Partial Orders
and
Fully Abstract Models
for
Concurrency

Uffe Henrik Engberg

March 26, 1990
Contents

Preface ................................................. v
Résumé (in Danish) .................................... vi

Presentation ............................................. 1

Introduction .......................................... 2

Overview and Basic Organization ................. 6

Summary of Results .................................. 8

Conclusion ............................................. 11

Bibliography .......................................... 13

I Testing Partial Orders .............................. 17

1 Semiwords: \( SW \) .................................. 18
  1.0 Preliminaries .................................... 18
  1.1 Basic Definitions ............................... 19
  1.2 Operations on \( SW \) ......................... 23
  1.3 Partial Orders on \( SW \) .................... 27
     1.3.1 Smoother Than ......................... 28
     1.3.2 Prefix of ............................... 37

2 Tree Semiwords: \( TSW \) ......................... 50
  2.0 Preliminaries .................................. 50
### 2.1 Basic Definitions

### 2.2 Operations on $T S W$

### 2.3 Partial Orders on $T S W$
- **2.3.1 Smoother Than**
- **2.3.2 Prefix of**

### 3 Semantics for a Simple Process Language: $PL$
- **3.1 Denotational Semantics**
- **3.2 Operational Semantics**
- **3.3 Full Abstractness**

### 4 Algebraic Characterization
- **4.1 Proof Systems**
- **4.2 Soundness**
- **4.3 Completeness**

### 5 Adding Recursion to $PL$
- **5.1 Denotational Semantics**
- **5.2 Operational Semantics**
- **5.3 Full Abstractness**

### Index

**II Tracing Partial Orders**

### 6 Pomsets
- **6.0 Basic Definitions**
- **6.1 Operations on Pomsets**
- **6.2 Two Partial Orders on Pomsets**
- **6.3 Sets of Pomsets**
- **6.4 Two Types of Pomset Properties**

### 7 $BL$—A Basic Process Language
- **7.1 General Semantics**
- **7.2 Operational Set-up**
7.3 Denotational Set-up ........................................ 155
7.4 Full Abstractness ............................................ 159
7.5 Summary ..................................................... 165
7.6 An Adequate Logic ........................................... 167

8 RBL—A Basic Process Language with Refinement ............... 172
  8.1 Operational Set-up ......................................... 173
  8.2 Denotational Set-up ......................................... 179
  8.3 Full Abstractness ............................................ 184
  8.4 Summary ..................................................... 189
  8.5 An Adequate Logic for RBL without Auto-Parallelism .......... 190

9 Adding Recursion to BL and RBL ................................. 203
  9.1 General Set-up ............................................... 204
  9.1.1 Denotations of Recursive Expressions .................... 204
  9.1.2 Contexts .................................................. 209
  9.1.3 \( \Sigma \)-precongruences ................................ 211
  9.1.4 Obtaining Algebraic Complete Partial Orders ............ 213
  9.2 Denotational Set-up ......................................... 216
  9.2.1 The Recursive Languages ................................. 216
  9.2.2 The Syntactic Finite Sublanguages ....................... 223
  9.3 Operational Set-up ......................................... 228
  9.3.1 The Recursive Languages ................................ 228
  9.3.2 The Syntactic Finite Sublanguages ....................... 229
  9.4 Full Abstractness ............................................ 230
  9.4.1 The Recursive Languages ................................. 230
  9.4.2 The Syntactic Finite Sublanguages ....................... 240

Index ............................................................. 249
Preface

In this thesis sets of labelled partial orders are employed as fundamental mathematical entities for modelling nondeterministic and concurrent processes thereby obtaining so-called noninterleaving semantics. Based on different closures of sets of labelled partial orders, simple algebraic languages are given denotational models fully abstract w.r.t. corresponding behaviourally motivated equivalences. Some of the equivalences are accompanied by adequate logics and sound axiomatisations of which one is complete.

The majority of the work was done with a scholarship at the computer science department, University of Aarhus, Denmark. The rest was carried out with grant-in-aid from the Danish Research Academy during a visit at the technical University of Munich, Germany, where I enjoyed the hospitality of Wilfrid Brauer and his concurrency group.

The thesis has grown out of inspiring and encouraging talks with my supervisor Mogens Nielsen to whom I give my special thanks. I am also grateful to Kim S. Larsen for the discussions we had when preparing a joint paper with him and Mogens Nielsen. I should like to thank Anders Gammelgård for our discussions and Karen Møller for her part of the typing. Last, not least, thanks go to my parents and my wife Ricarda.

Uffe Henrik Engberg
Resume

Den foreliggende licentiatafhandling placerer sig inden for området: semantiske modeller for parallele systemer. En gren heraf er semantisk beskrivelse af konkrete programmeringssprog, hvori parallelisme og nondeterminisme kan udtrykkes. Gennem den semantiske beskrivelse fastlægges hvilke processudtryk, der er ækvivalente, således, at det f.eks. giver mening at tale om, hvorvidt et processudtryk er en korrekt implementation af et andet, eller at en proces kan erstatte en anden i en given kontekst. Mange af studierne inden for området har taget udgangspunkt i mere abstrakte processprog som CCS, og de er blevet udstyret med forskellige former for semantik, eksempelvis: operationel, denotationel,aksiomatisk og logisk semantik.

Det er den operationelle semantik, som åbner mulighed for en intuitiv forståelse af, hvad en proces kan gøre, og hvilke egenskaber der er afgørende for, at to processer opfører sig ens – er ækvivalente. Ofte er intuitionen den, at procesækvivalensen fremkommer i en eksperimental opsætning, hvor en observatør udfører tests på nogle maskiner i henhold til en bestemt "protokol", og hvor, det en maskine kan gøre, er bestemt ud fra det tilhørende processudtryk.

For at sikre, at en semantik er i overensstemmelse med den operationelle intuition, er det derfor vigtigt med en præcis forbindelse til den operationelle semantik. Ved denotationelle semantikker er det formaliseret ved, at de denotationelle modeller er fuldt abstrakte m.h.t. de tilhørende operationelle ækvivalenser. D.v.s. de operationelle ækvivalenser er kongruenser (bevares i vilkårlige kontekster) og to processer giver anledning til de samme denotationer i modellerne, netop når processerne er operationelt ækvivalente (modellerne er "fully abstract"). For aktiomatiske semantikkers vedkommende er de tilsvarende begreber sundhed og fuldstændighed, hvor et bevissystem er sundt og fuldstændigt, når processer kan bevises ækvivalente, hvis og kun hvis de er operationelt ækvivalente. Ved logisk semantik forlanges typisk, at to processer tilfredsstiller de samme logiske formler, præcist når de er ækvivalente.

Den overvejende del, af de sædvanlige operationelle ækvivalenser, adskiller sig primært ved hvilken grad af nondeterminisme de er i stand til at skelne, og har som fællesstræk at (endelige) parallele processer er ækvivalente med tilsvarende rent nondeterministiske, men sekventielle processer – d.v.s. parallelisme reduceres til nondeterminisme. Flere af disse operationelle ækvivalenser er blevet karakteriseret logisk eller udstyret med sund og fuldstændige bevissystemer, og nogle af ækvivalenserne er givet denotationelle modeller, som baserer sig på abstraktioner over beregningstræer, og som er fuldt abstrakt m.h.t. ækvivalenserne. Derimod er sådanne resultater mere sjældne, når det drejer sig om de ækvivalenser, hvor parallelisme ikke reduceres til nondeterminisme.

Ved i stedet at fokusere på parallelisme og neglere de nondeterministiske aspekter, når
tests og protokoller for eksperimenterne fastlægges, gives der i afhandlingen flere forskellige operationelt definerede ækvivalenser for simple processprog, og ækvivalenserne udstyres med fuldt abstrakte modeller, hvor mængder af mærkede partielle ordninger, forkortet m.p.o.’er, fungerer som den naturlige modpart til beregningstræer.

Afhandlingen består af en indledende præsentation samt to hoveddele, der hovedsageligst adskiller sig ved om det er testene eller protokollerne, der gøres til genstand for variation, når de operationelle ækvivalenser defineres. De to dele er skrevet uafhængigt af hinanden og kan derfor også løses adskilt. Overordnet følger begge dele den samme linie. Først foretages en isoleret undersøgelse af de objekter, der senere skal danne baggrund for de denotationelle modeller. Derefter gives operationel og denotationel semantik til det pågældende sprog for endelige processer, og det bevises, at de stemmer overens. Hver del afsluttes med tilføjelse af rekursion, og de tidligere denotationelle resultater løftes til den ny opsætning.

I den første del af afhandlingen betragtes et meget simpelt processprog med kombinatorer for (sekventiel) præfiksning af atomare aktioner, nondeterministisk valg, og parallelssammensætning uden auto-parallelitet, d.v.s. at en atomar aktion kun kan optræde i én af to parallelle processor. Gennem et lidt usværligt transitionssystem og en fastlagt type tests, designet til at afdække parallelitet, opnås tre forskellige operationelle ækvivalenser ved at betragte, hvordan processor fra sproget kan reagere på eksperimenter med testene. Til ækvivalenserne knyttes denotationelle modeller, der baserer sig på en klasse af m.p.o.’er, kaldet semiord, der reflekterer fravær af auto-parallelitet, og det bevises, at modellerne er fuldt abstrakte m.h.t. ækvivalenserne. Generelle m.p.o.’er, og dermed også semiord, kan bl.a. sammenlignes via to forskellige (partielle) ordninger, som udtaler sig om, hvorvidt én m.p.o. er et præfiks, henholdsvis glattere/ mindre sekventiel end en anden m.p.o.. Det viser sig, at de denotationelle afbildninger kan udtrykkes som bestemte lukninger af en kanonisk associering af semiord til procesudtryk. Disse lukninger er præfikslukninger, som igen, alt afhængig af den aktuelle ækvivalens, er opad-/ nedad- konkavlukkede m.h.t. ”glatheds” relationen. Desuden gives et sundt bevissystem som ved en mindre udvidelse vises at være fuldstændigt for en af ækvivalenserne.

I den anden del betragtes et mere generelt processprog, der rummer mulighed for auto-parallelitet og sekventiel sammensætning af vilkårlige processor. Eksperimenter fastlægges her til at være maksimale sekvenser af direkte tests, og i stedet gøres de direkte tests til genstand for variation. Med en enkelt direkte test undersøges, om visse typer af aktioner kan udføres parallelt på én gang. Hver ”naturlig” mængde af direkte tests og tilhørende transitionssystem, giver anledning til en operationel ækvivalens, hvortil der knyttes en fuldt abstrakt model, der p.g.a. af det udvidede processprog, bygger på generelle m.p.o.’er. De denotationelle afbildninger følger samme mønster som i den første del, men de bestemte lukninger er her nedadlukninger, restrigeret til lagdelte m.p.o.’er, hvor hvert lag svarer til en af de direkte tests, som er mulige ved den aktuelle ækvivalens. Af disse resultater afledes, at ækvivalenserne danner et gitter med den almindelige (automatteoretiske) strengækvivalens i bunden, som den mindst nuancerede m.h.t. hvilke processor, der kan skelnes. Hver af disse ækvivalenser karakteriseres ved en Hennessy-Milner-lignende lineær modallogik.

Til processproget føjes en forfinelseskombinator, der til hver atomar aktion angiver et procesudtryk (uden forfinelseskombinatorer) som aktionen skal implementeres ved. På
en simpel måde indkopereres den ny kombinator i transitionssystemerne, og det bevises,
at den operationelle virkning er, som hvis de enkelte forfinelser på forhånd var tekstuelt
substitueret ind for de pågældende atomare aktioner. Derved bliver der mulighed for, at
forfinelser af parallele aktioner kan ”overlappe”, hvorfor ækvivalenserne ikke bevares un-
der den ny kombinator. Derfor studeres i stedet deres (største konsistente) kongruenser.
Herved opnås én enkelt mere nuanceret kongruens. Kongruensen gives en fuldt abstrakt
denotationel model, hvor den afgørende forskel er, at nedadlukningerne i stedet bliver
restringeret til m.p.o.’er, som ikke kan skelnes ved ”overlapning”. For et delprocessprog
uden auto-parallelitet karakteriseres kongruensen yderligere ved en modallogik, der, til
forskel for de ovennævnte, har en ekstra modaloperator, hvormed en slags delvis baglås
kan specificeres.

Sammenfattende kan det siges, at afhandlingen fremviser forskellige måder, hvorpå grader
af parallelisme ved processer kan skelnes, enten gennem forskellige operationelt motiverede
ækvivalenser, eller gennem de præordninger som ækvivalenserne er fremkommet af, og
at mærkede partielle ordninger på naturlig måde tjener som hjørnesten i de tilhørende
modeller.
Presentation
Introduction

Overall Background

During the last two decades attention to the area of concurrency has increased as program-
ning concepts for handling nondeterministic and concurrent systems have been introduced
while advances in hardware technology have made it realistic to use new programming
languages incorporating these concepts. A great deal of the research has been made in
order to achieve a good understanding of the meaning of concurrent systems and how to
reason about them, an understanding comparable to that of sequential systems where e.g.
the well-known axiomatic method of Hoare [Hoa69] is applicable for sequential programs.
The ongoing research has resulted in a multitude of models for concurrency, for exam-
ple Kahn-MacQueen networks [KM77], Mazurkiewicz traces [Maz77], Petri nets [Rei85],
event structures [NPW81, Win87] and different semantics for process algebras. The main
intention of this thesis is to contribute to this line of research.

Principal Confinement

Whereas it is standard to take the meaning of a sequential program as a function from
input to output there is no prevailing agreement on what the meaning of concurrent
programs should be. As De Nicola and Hennessy reason in [DNH84] it is necessary to
search for counterpart to functions when building semantic theories for concurrency. In
order not to obscure this task it is common practise to pay less attention to data aspects
of concurrent programs and in stead investigate the fundamentals of control since this
were the essential nature of concurrency lies. That is, in place of concrete programming
languages for concurrency, like Concurrent Pascal, Modula-2 and Ada, abstract languages
or process algebras containing combinators for the most fundamental notions of control –
sequential, nondeterministic and parallel composition – are taken as starting point for
the development of semantic theories for concurrency. This is also the case for the present
thesis and deliberately only process languages with these fundamental, more algebraic
combinators are studied. Prominent examples of larger process algebras which have been
equipped with a broad spectrum of theories are CCS [Mil80, Mil84] and TCSP [Hoa78,
BHR84].
General Requirement

Various forms of semantics for process algebras exist including: operational, denotational, axiomatic and logical – each of which contributes to knowledge and insight. Typically through labelled transition systems [Plo81] the operational semantics provide the means for an intuitive understanding of how concurrent processes behave and which processes are behaviourally equivalent. This is one of the main arguments when Milner (in e.g. [Mil83]) and many others argue that a semantic approach should be firmly based on an operational semantics. Consequently it will be a general requirement here too. Due to the importance of the requirement it has got an explicit formulation within the different types of semantics.

In case of denotational semantics it is formalized by the concept of a denotational map being fully abstract w.r.t. an associated behavioural equivalence. I.e. the interpretations of two processes in the denotational domain should be identified exactly when the processes are behaviourally equivalent.

As far as axiomatic semantics are concerned the analogous concepts are soundness and completeness – a proof system being sound when processes are provably equal only if they are behavioural equivalent, and complete if all such processes can be be proved equal.

Regarding semantics by logics one formulation of the requirement is adequacy. That means a logic is adequate when two processes satisfy the same set of formulas exactly when the processes are behaviourally undistinguishable.

Main Objective

The diversity of approaches to concurrency is also reflected in their attitude to the questions as to whether a linear or branching view of nondeterministic and concurrent systems should be taken, and whether concurrent processes should be reducible to purely nondeterministic, but sequential processes. When using a CCS/ TCSP like notation the first question can be illustrated by whether or not

\[(*) \quad a.(b + c) \text{ and } a.b + a.c \]

should be identified, and similarly for the second whether or not

\[(**) \quad a \parallel b \text{ and } a.b + b.a \]

should be distinguished. Changing from a look of controversy, the discussions around these questions seem now to have resulted in the understanding that there are no straight answers and that the attitude taken should depend on the situation at hand.

When concurrency is reduced to nondeterminism, concurrent processes are considered equivalent to ones with nondeterministic choice between different sequential shuffles of the individual processes as in (**) above, and the semantics are often described as being interleaving. For CCS, TCSP and other process algebras the question of a linear or branching view has here led to a whole spectrum of behavioural equivalences ranging from trace equivalence (in the classical language theoretic sense – not to be mistaken...
for Mazurkiewicz traces) [Hoa85, OH86], which identify say (⋆), over failure and testing
[BHR84, DNH84, OH86] to bisimulation equivalence [Mil80, Par81, Mil84], equivalences
which do not identify (⋆). Operationally these equivalences differ mainly in their view
of the branching structure of the labelled transition system associated with processes.
Through the study of degrees of branching some of the equivalences have been given
fully abstract denotational models where the counterparts to input-output functions (for
sequential programs) can be viewed as abstractions of computation trees (also called
synchronization trees) which in turn are slightly modified unfoldings of the corresponding
labelled transitions systems.

In other approaches concurrency is independent of nondeterminism and the processes of
(⋆) are distinguished. Among these approaches are the so-called partial order semantics
where causality, respectively concurrency, is represented by means of partial orderings
of actions. I.e. alternatively to computation trees, constructions containing labelled partial
orders (lpos for short) are proposed as counterparts to functions. These constructions
are often sets of some kind of lpos and so nondeterminism cannot be discriminated in
the semantics using them. However, it is possible in the denotational semantics based
on a generalization of lpos, labelled event structures, where nondeterminism is dealt with
by means of a conflict relation. See [BC87] for a good survey on the rôle of partial
orders in semantics for concurrency. Apart from step semantics, different proposals for
generalizations of existing behavioural equivalences (for nondeterminism) have been made
with time-based equivalence [Hen88b] and distributed bisimulation [CH88] among the
most discriminating. See also the final remarks of these papers. In the style of [Jon88,
Rei88] the situation can roughly be sketched as:

<table>
<thead>
<tr>
<th></th>
<th>(⋆)</th>
<th>(⋆)</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Trace</td>
<td>Set of words</td>
</tr>
<tr>
<td>≠</td>
<td>Bisimulation</td>
<td>Computation tree</td>
</tr>
<tr>
<td></td>
<td>Step</td>
<td>Set of lpos</td>
</tr>
<tr>
<td></td>
<td>Distributed</td>
<td>Event structure</td>
</tr>
<tr>
<td></td>
<td>Bisimulation</td>
<td></td>
</tr>
</tbody>
</table>

Behavioural process equivalence       Entity modelling processes

Whereas the work on interleaving semantics has led to a number of e.g axiomatisation and
full abstractness results, such results are more unusual when it comes to noninterleaving
semantics. Motivated by this and the suggestion of Pnueli [Pnu85] to study degrees of
concurrency in place of branching the main objective of the thesis is to explore the possi-
bilities of defining “natural” operational semantics for algebraic process languages which
open up opportunities for alternative semantics, especially for fully abstract denotational
models with lpos as main ingredient of the entities modelling processes. That is to say
we are seeking different behavioural equivalences where lpos come “naturally” in to the
 corresponding models, thereby capturing various degrees of nonsequentiality.

Possible Courses

Looking for ideas of how to modify behavioural equivalences such that the semantics is
not interleaving, it immediately appears to try to catch a property which intuitively seems
to be a distinctive characteristics of concurrency. To take an example one might argue that if a defect occurs in a subprocess then other concurrent subprocesses are able to run undisturbed (except of course if there is some dependence due to communication). If e.g. \( \mathcal{U} \) denotes the faulty process which cannot do any action and if besides the usual \( a.p \xrightarrow{a} p \) also the rule \( a.p \xrightarrow{a} \mathcal{U} \) is used in the definition of the action relation, then many of the known behavioural equivalences would distinguish say \((**)\). In the introduction and final remarks of [Hen88b], Hennessy discusses other ideas and in the same paper and in [CH88, Cas88] the ideas are successfully examined obtaining axiomatisations for generalizations of bisimulation equivalence. However, bearing in mind the difficulties in finding fully abstract models for bisimulation equivalence, we deliberately choose to study degrees of concurrency as “orthogonal” to the existing study of degrees of branching. Taking the lead of [HM80, Mil80, DNH84, Abr87] the intuition will be that of a behavioural equivalence arising in an experimental setting with observers performing tests according to some “protocol” on machines, with operational abilities defined in terms of labelled transition systems. Though omitting branching aspects, the various manners in which to capture degrees of branching can serve as a clue for capturing degrees of nonsequentiality. For example, instead of having tests with different strengths in discovering nondeterminism, tests may in different ways be geared towards parallelism (possibly by departing from the traditional labelled transition systems). Once tests capable of detecting some kind of concurrency are fixed, variations may be obtained by changing the “protocol” in the style of [DNH84]. Another direction to take is suggested in [Pnu85, BIM88] where increasing discriminating equivalences are obtained from a simple equivalence (trace) by considering the congruence when different combinators are added. So, finding combinators uncovering an aspect of concurrency, the congruence will be forced to take the aspect into account. These directions can be combined in several ways of which we have chosen two and elaborated each in a separate part of the thesis.
Overview and Basic Organization

The thesis is divided in two parts, which mainly differ in whether the tests or “protocols” of the experiments are subject to variations when the behavioural equivalences are defined.

In part I a particular kind of tests suitable to probe concurrency of processes is introduced for a simple process language, $PL$, and different equivalences are obtained by considering possible outcomes of the experiments. $PL$ contains combinators for prefixing of atomic actions, nondeterministic choice and parallel composition (without communication). The experiments and the labelled transition system is somewhat unconventional. Here an atomic action can be thought of intuitively as connected to a certain resource thereby excluding auto-parallelism [vGV87] (an atomic action can only occur in one of two parallel processes). When a signal, $a$, is submitted to initiate the action (ambiguously designated) $a$, this is noted such that other actions, possible the same, can be signaled to initiate. Each time the action $a$ is completed this is signaled by $\overline{a}$ as response. At first an attempt is made to signal a (multi) set of actions and if this turns out well a test is made on the signaled actions, where the language for specifying tests contains constructs for what Abramsky [Abr87] calls traces and copying. The process may accept the experiment if the actions can be signaled and the following test is successful, and may reject the experiment if the actions can be signaled and the test is not successful. The three equivalences, $\lll$, $\lll_{a}$ and $\lll_{r}$, are generated from the preorders $\ll$, $\ll_{a}$ and $\ll_{r}$ respectively, where $\ll$ is the intersection of $\ll_{a}$ and $\ll_{r}$, and one process is related via $\ll_{a}$ ($\ll_{r}$) to another if the experiments the first may accept (reject) also may be accepted (rejected) by the other.

Unlike in part one, the tests of the experiments in part II are varied when the different behavioural equivalence are introduced and the basic process language, $BL$, is slightly more general as auto-parallelism and full sequential composition is possible. Experiments are maximal sequences of direct tests and the variations arrise from the power admitted for the direct tests – with a single action tested as the weakest and a multiset the most powerful. For any “natural” fixed set of direct tests, $G$, processes are considered behaviourally equivalent, $\lll_{G}$ (actually generated from a preorder $\ll_{G}$), if they react identically to the same experiments. The equivalences are generalizations of the ordinary (maximal) trace equivalence which appears from the weakest direct tests.

Holding on to the behavioural equivalences $BL$ is extended to $RBL$ by adding a refinement combinator which makes it possible to prescribe through a map, called a $BL$-refinement, how atomic actions within the scope of the combinator should be refined or implemented in terms of basic processes of $BL$ (change of atomicity). Because the refinement combinator enables “overlapping” of refined actions, the equivalences are not preserved under the new combinator and their finer associated congruences, $\ll_{G}$, are considered. This part of the thesis is largely a continuation/ extension of [Lar88] and [NEL89] to cope with
auto-parallelism and recursion.

Both parts follow the same general line. At first lpos, or rather equivalence classes of lpos, are studied in their own right. Operations and the relations, prefixing and “smoother than” (where one lpo is smoother than another if the ordering relation of the first is a superset of that of the other lpo), are introduced and properties are derived – of course selecting certain topics in preparation for the models to come. In part I the study is actually confined to particular equivalence classes of lpos, called semiwords, where equally-labelled elements are demanded to be ordered, thereby reflecting absence of auto-parallelism. One important property of semiwords is that they have canonic representatives wherefore definitions and reasoning can be made directly in terms of these. Aiming at similar conditions for the general equivalence classes of lpos, pomsets, elements of representatives are in part II taken from a certain ground set and in fact pomsets can to some extent be handled as smoothly as semiwords. Together pomsets and semiwords will in the rest of the presentation be referred to simply as lpos.

After the initial study of lpos, operational and denotational semantics are given of the process language in question and a connection between them is established. More specifically, the denotational models, which build on different closures of sets of lpos, are proved to be fully abstract w.r.t. the corresponding operational equivalences. Besides this, alternative methods to reason about the processes are given, and links to the equivalences are shown.

Finally each part is ended by adding recursion to the process language, and both the operational semantics and the denotational characterizations are extended accordingly. In part II new behavioural equivalences, $\equiv_G$, come in by relaxing the maximality requirement of sequences (of direct tests). The new equivalences are not preserved in $BL$ or $RBL$ contexts, and their congruences, $\equiv_G^e$, are studied. For this purpose a new criterion – a language being expressive w.r.t. a preorder – which ensures algebraicity of precongruences is introduced. More technical prerequisites are necessary in part II and for the same reason they are treated more thoroughly there. For instance two ways of extending (denotational) relations to open expressions are compared and proofs (of results mentioned in [Hen83]) are made in full detail. Acquaintance with standard denotational techniques for dealing with recursion as presented in [Hen88a] is assumed.

The two parts of the thesis are written and may be read independently and hence there is a few differences in notation and some redundancy around the treatment of lpos. As a help for the reader each part is equipped with an index of the most used notions, definitions and symbols. To avoid repeating references a common bibliography is included at the end of this presentation of the thesis.
Summary of results

We shall here briefly state the results of the thesis and start out by looking at the syntactic
finite process languages (without recursion constructs) \( PL, BL \) and \( RBL \), where \( PL \) as
previously mentioned has combinators for prefixing, nondeterministic choice and parallel
composition (without auto-parallelism and communication), \( BL \) in addition has auto-
parallelism and full sequential composition and \( RBL \) a refinement combinator.

Operationally a new idea is introduced for \( PL \). In the labelled transition system control
is divided in two: at first nondeterministic choices are made during the act of signaling
actions to initiate. These are in turn later be completed and vanish from the configura-
tions.

For \( BL \) the operational capabilities are given via a more standard extended labelled
transition system in the style of [Nic87, Hen88a] where an internal step is used to resolve
(internal) nondeterministic choice. When it comes to \( RBL \) it turns out that a simple
operational “lazy substitution” of refinements can be given by means of the internal
step relation and this operational “substitution” is shown to coincide with the textual
substitutions of refinements.

Looking at the models, we draw the attention to the fact that they consist of finite sets
of lpos and that the denotational maps of the different models all can be regarded as
some kind of closure of the same canonical association of lpos to process expressions. In
addition the denotational maps admit simple compositional definitions, basically built in
terms of the operators used in the canonical maps and the relevant closure at the places
where the closure is not preserved.

For \( PL \) and \( \sqsubseteq_a \) (\( \sqsubseteq_r \), respectively) the closure used in the corresponding model, \( M_\chi \) (\( M_\kappa \)
or \( M_\nu \)), is the prefix- and convex (downwards or upwards) closure w.r.t. the “smoother
than” partial ordering of semiwords. The models are shown to provide suitable interpre-
tations of the behavioural equivalences through the full abstractness results. From the
models and examples it is seen that both \( \sqsubseteq_a \) and \( \sqsubseteq_r \) are strictly more abstract than \( \sqsubseteq \).
Furthermore, a sound proof system, \( DED_\pi \), is given which makes it possible to show
statements concerning “prefix-closure” as well as more ordinary algebraic properties of
the combinators such as commutativity and associativity of + and \( \parallel \). Extending \( DED_\pi \)
to \( DED_\delta \) by adding the axiom \( a.(x \parallel y) \leq a.x \parallel y \) a sound and complete proof system is
obtained for \( \sqsubseteq_a \) (or rather \( \sqsubseteq_a \)). In the style of [Hen88a] the results can be schematized:
Turning to BL and fixing a set of direct tests, \( G \), the closure of the the corresponding fully abstract model is the ordinary “smoother than” downwards closure of pomsets restricted to those pomsets which are “layered” and where each layer resembles a possible direct test from \( G \). Varying \( G \) it is seen that the equivalences form a lattice (in the sense of their ability to distinguish processes) with the usual trace/word equivalence, \( \triangleleft_w \), at the bottom and the unrestricted multiset equivalence, \( \triangleleft_M \), at the top. Each \( \triangleleft_G \)-equivalence is given an alternative characterization in terms of an adequate Hennessy-Milner like linear modal logic, \( \mathcal{L}_G \), containing a straightforward generalization of the “labelled” necessity modality (box) and atomic propositions expressing termination and non-termination. The results are sketched below:

The main observation for RBL is that when considering the largest congruences, \( \triangleleft_G \), contained in the equivalences, \( \triangleleft_G \), the addition of the refinement combinator collapse the lattice of equivalences into a strictly finer equivalence. Thereby also the result, \( \triangleleft_w^c = \triangleleft_M^c \), which looks like a similar result Hennessy notices in the final remarks of [Hen88b] for time-based bisimulation. The closure used in the fully abstract model for \( \triangleleft_G \) is again the downwards closure of pomsets, but instead restricted to those pomsets where of any two concurrent elements the successors of one also are successors of the other or vice versa. By removing auto-parallelism from RBL a sublanguage, \( RBL' \), is obtained which, beside resembling semiword based models, is equipped with an adequate logic, \( \mathcal{L}_G \). An extra modality for specifying a kind of semi-deadlock is here at disposal. The schematized results are:
SYNTAX:  
\[ RBL \supseteq RBL' \]

OPERATIONAL BEHAVIOUR:  
\[ G \]

DENOTATIONAL SEMANTIC:  
\[ M_{or} \]

LOGIC SYSTEM:  
\[ \mathcal{L}_G' \]

Now for the full process languages of \( PL, BL \) and \( RBL \) with recursion.

The transition systems for the different process languages are extended in the usual way to cope with recursion and in particular it is noticed for \( RBL \) that no extra (internal step) inference rule is needed for the interplay between the refinement combinator and the recursion constructor.

The models remain in principle the same but sets of lpos may now be infinite and the models, \( M^G_{or} \) and \( M^G_{or} \), for \( G \) w.r.t. \( BL \) and \( RBL \) respectively, separately carry information concerning approximating sequences. The domains of the finitary models are in a uniform way shown to be algebraic complete partial orders and the achieved models are proved to be fully abstract w.r.t. the corresponding behavioural equivalences. In this course a new criterion for algebraicity of precongruences turn out to be very useful.

\( PL \) can, modulo \( NIL \) and minor syntactic differences, be considered as a sublanguage of \( BL \) which in turn is a sublanguage of \( RBL \). Then from the pleasant fact that both the \( M^G_{or} \) and \( M^G_{il} \) model are expressed as abstractions over the downwards and prefix closure of a canonical association of lpos with expressions it follows that the relationship between the equivalences roughly can be illustrated as:

\[
\begin{array}{c}
  \mathbb{L}_a \searrow \\
  \mathbb{R}_a \searrow \\
  \mathbb{P}_a \searrow \\
  \mathbb{N}_a \searrow \\
  \mathbb{C}_a \searrow \\
  \mathbb{M}_a \searrow \\
  \mathbb{G}_a \searrow \\
  \mathbb{W}_a \searrow \\
  \mathbb{R}_a \searrow \\
  \mathbb{P}_a \searrow \\
  \mathbb{N}_a \searrow \\
  \mathbb{C}_a \searrow \\
  \mathbb{M}_a \searrow \\
  \mathbb{G}_a \searrow \\
  \mathbb{W}_a \searrow \\
  \mathbb{L}_a \searrow \\
\end{array}
\]

where \( \longrightarrow \) indicates that the equivalence on the left-hand side is strictly more abstract than the one on the right-hand side (the congruence of an equivalence is w.r.t. the language labelling the highest box the equivalence is contained in). Since the equivalence of the two parts only are compared here, we give two expressions, which illustrates that \( G \) w.r.t. \( RBL \) is strictly more abstract (on \( PL \)) than \( G \) (identified by \( G \) but not by \( G \)):

\[
(a \parallel c \parallel d) + (b \parallel a \parallel d \parallel c) + (a \parallel c \parallel b \parallel d) \quad \text{and} \quad (b \parallel a \parallel d \parallel c) + (a \parallel c \parallel b \parallel d)
\]

To sum up the achievements of the thesis one could say that means are brought about for discriminating degrees of concurrency in processes, either through different behavioural equivalences or through the preorders they are generated from, and that labelled partial orders in a natural way serve as cornerstones in the associated models.
Conclusion

The full abstractness results are obtained at the expense of simplified process languages and an undetailed view on branching. We shall here discuss a few ideas to redress some of the shortcomings and their impact on the results.

For PL the requirement of absence of auto-parallelism is crucial. This is best seen in the proofs of full abstractness which rely heavily on the fact that semiwords are characterized by their linearizations and no characterization of the pomsets that are identified by their linearizations is known. But by omitting auto-parallelism, it looks manageable to extend PL to BL and keep the results. Now consider what happens if a refinement combinator which does not introduce auto-parallelism is added, either to PL or the extension. Then it is unlikely that it will have any influence, at least not on the $\equiv_n$-equivalence, since two refined processes (without $+$), which can be distinguished by sequences, already are distinguished by the may-experiments on the unrefined processes.

Whereas the combinators of BL are quite simple this is by no means the case for the refinement combinator of RBL, but it suffers from an effective way to be specified. As it is now, a refinement is given by a function from the (infinite) set of atomic actions to the process expressions of BL. One way to go would be to introduce the notation $[a_1 \leadsto p_1, \ldots, a_n \leadsto p_n]$ for the refinement where all actions remain unrefined except that $a_1$ is refined to $p_1$, $a_2$ to $p_2$, etc. and only allow such refinements. Then it would not be possible to specify fission refinements as they are formulated now, but a closer look at the proofs, where these refinements are used, shows that refinements which “fission” on a finite set will do and so all the results go through. With the refinement combinator it is possible to imitate relabelling by considering the relabelling functions as a special class of BL-refinements (maps to individual atomic processes). Looking at the way relabelling usually is introduced in transition systems, the relabelling combinator is stactic in nature in contrast to the more dynamic nature of the refinement combinator, but this difference cannot be uncovered by the equivalences. Inaction ($NIL, SKIP$) seems also easy to include in RBL. The few proofs, where the refinements are assumed not to make actions disappear ($\varepsilon$-freeness), get more complicated. A (maybe unexpected) consequence of adding $NIL$ would be that expressions like $a$ and $a + NIL$ would be distinguished by $\equiv_G$ and also by the congruence of $\equiv_G$ (think of a context where the expressions are sequential composed by another action $b$). Once inaction is added to RBL it is no problem to simulate hiding of an action $a$; simply use the refinement combinator $[a \leadsto NIL]$. However the use of such an abstraction feature is limited as long as parallel processes cannot communicate – a matter we shall address next.

The extensions discussed until now stay so to say within the simplified view on branching.
But if we extend the parallel combinator of RBL such that e.g. synchronization shall happen on all common actions as in TCSP [BHR84] and we still look at maximal sequences, we would at once get a finer view, because the possibility of deadlock forces the model to reflect branching structure – see [Pnu85]. We have on purpose carried out this work on nonsequentiality “orthogonally” to existing work on branching, but it is an intriguing question, whether such an extension could be modeled by a smooth combination of e.g. the \( M_{or} \) model and the broom model of Pnueli – capturing aspects of nonsequentiality as well as branching.

We conclude by a simple example which indicates that such a combination in no way is straightforward to obtain. Suppose

\[
p = a \parallel b \quad \text{and} \quad q = a.b + b.a + a \parallel b
\]

Then \( p \) and \( q \) are identified in both the \( M_{or} \) model and the broom model, but \( p' = p[a \sim c.d] \) and \( q' = q[a \sim c.d] \) would be distinguishable in a parallel context with \( c.b.d - c \) is a possible maximal sequence of \( q' \parallel c.b.d \) whereas this is not the case for \( p' \parallel c.b.d \). Hence a “conjunction” of the two models would be to abstract for the congruence of \( \xi_G \) w.r.t the two combinators.
Bibliography


Part I

Testing Partial Orders
Chapter 1

Semiwords: SW

1.0 Preliminaries

Partial orders are often used to reflect causal relationships between events. In this chapter we shall present a special subclass of labelled partial orders called semiwords and find a number properties semiwords enjoys. Roughly speaking a semiword is a labelled partial order where the equal labelled elements are ordered. Before giving the exact definitions of labelled partial orders and semiwords we start out by a few mathematical and other conventions.

Propositions and definitions are numbered within chapters, e.g., definition 1.0.1 (the definition below) where the first number indicates the chapter it appears in and the second is the number of the definition.

If $\leq$ is a partial order over $A$ the downwards closure of an element $a \in A$ w.r.t. $\leq$ will be denoted $DC_\leq(a)$, i.e., $DC_\leq(a) = \{ b \in A \mid b \leq a \}$. Similar $UC_\leq(a)$ denote the upwards closure of $a$ w.r.t. $\leq$. We shall often use functions defined on sets, so in order not to write to many parenthesis we shall write $fS$ for the function application $f(S)$ where $S$ is a set and at the same time an element in the domain of $f$. The standard set, $\{1, \ldots, n\}$, will be denoted $\underline{n}$ and a tuple of the form $(t_1, \ldots, t_k)$ is abbreviated $\overline{t}$.

**Definition 1.0.1** Given a nonempty set $\Delta$, a *labelled partial ordering* (lpo for short) over $\Delta$ is a triple $(A, \leq, \beta)$, where $\beta : A \rightarrow \Delta$ is a mapping from $A$ into $\Delta$ and $\leq$ partially orders the set $A$ or equally $(A, \leq)$ is a poset, i.e., $\leq$ is a binary relation on $A$ which is reflexive, transitive and antisymmetric.

$A$ can be regarded as events, i.e., particular occurrences of actions and $\Delta$, the alphabet, as actions, or types of events.

**Definition 1.0.2** Two lpos $\rho = (A, \leq, \beta)$ and $\rho' = (A', \leq', \beta')$ are said to be isomorphic (written $\rho \cong \rho'$) iff there exists a bijection $\phi : A \rightarrow A'$ such that for all $a, b \in A : \beta(a) = \beta(\phi(a))$, and $a \leq b$ iff $\phi(a) \leq' \phi(b)$.

The equivalence class under $\cong$ of any lpo $\rho$ is denoted $[\rho]$ i.e., $[\rho] := \{ \rho' \mid \rho' \cong \rho \}$ and $\rho$ is
called a representative. If \( \rho = (A, \leq, \beta) \) we also write the corresponding equivalence class as \([A, \leq, \beta]\).

The subset of the quotient set of the lpos over \( \Delta \) by \( \cong \) where the posets of the representatives are finite are called the set of partial words over \( \Delta \) (written \( PW(\Delta) \)), i.e.,

\[
PW(\Delta) := \{[A, \leq, \beta] \mid (A, \leq) \text{ is a finite poset, } \beta : A \to \Delta \}.
\]

The subset of the partial words over \( \Delta \) where the equal labelled elements of the representatives are finitely ordered are called the set of partial words over \( \Delta \) (written \( PW() \)), i.e.,

\[
PW() := \{[A, \leq, \beta] \mid (A, \leq) \text{ is a finite poset, } \beta : A \to \Delta \}.
\]

The subset of the partial words over \( \Delta \) where the equal labelled elements of the representatives are linearly ordered are called the set of semiwords over \( \Delta \) (written \( SW() \) or \( SW \) for short), i.e.,

\[
SW() := \{[A, \leq, \beta] \mid \forall a, b \in A : \beta(a) = \beta(b) \Rightarrow a \leq b \lor b \leq a \};
\]

(\( \leq \) the partial order restricted to equal labelled elements satisfies the trichotomy law.) □

The semiwords were first introduced by Starke \[Sta81\] and reflects the idea that two occurrences (events) of the same action cannot be concurrent.

Though many of the following notions and results could be formulated and hold for \( PW() \) we prefer to introduce them for semiwords only. First of all because we are only concerned with semiwords in this work and second, because they have a particularly simple representation which we shall refer to as the canonic representatives.

Canonic representatives

According to Starke \[Sta81\] the canonic representatives can be characterized as follows:

Let \( A \) be a finite subset of \( \Delta \times \mathbb{N}^+ \) and \( \leq \) be a partial order on \( A \). Then \( (A, \leq) \) is the canonic representative of a semiword over \( \Delta \) iff for all \( a \in \Delta, i, j \in \mathbb{N}^+ \) it holds:

- \( SW1: (a, i) \in A \land 1 \leq j \leq i \Rightarrow (a, j) \in A \)
- \( SW2: (a, i), (a, j) \in A \Rightarrow ((a, i) \leq (a, j) \iff i \leq j) \)

Intuitively \( (a, i) \) denotes the \( i^{th} \) occurrence of the action \( a \), i.e., \( (a, i) \) is a label \( a \) with rank \( i \) (SW1). Since all equal labelled elements are linearly ordered (SW2) this gives sense and from Starke it follows that the mapping which for a canonic representative \( (A, \leq) \) gives the semiword \( [A, \leq, \beta] \), where \( \beta((a, i)) = a \), is an isomorphism.

In the sequel we will identify a semiword \( s = [\rho], \rho = (A, \leq, \beta) \) with its canonic representative which we denote \((A_s, \leq_s)\) or just \( (A, \leq) \) when it is clear from the context. We shall therefore refer to \( SW \) as the subclass of the lpos which satisfies \( SW1 \) and \( SW2 \).

1.1 Basic Definitions

Notationally, it will be convenient to let \( a^i \) denote \((a, i)\). If the rank of the element is unimportant for an argument or statement, we will simply omit the rank \( i \) of \( a^i \) and just write \( a \).

If not only equal labelled elements of a semiword but all elements are linearly ordered, we call it a word. Formally:
**Definition 1.1.1** Let $s \in SW$. $s$ is a word over $\Delta$ iff $\leq_s$ satisfies the trichotomy law:

$$\forall a, b \in A_s. \ a \leq_s b \lor b \leq_s a$$

The set of all words over $\Delta$ is denoted $W(\Delta)$ or $W$ for short. Notice $s \in W$ implies $\leq_s$ is total.

The one to one correspondence between $\Delta^*$ and $W$ should be clear (if not, see [Sta81]) and in the sequel we will often identify their members.

In order to introduce operators on $SW$ it will be useful to define a function $\psi$, which given a semiword $s$ and an action $a$ yields the maximal rank of an element of $s$ labelled by $a$. Because of $SW1$ this number equals the number of elements labelled with the action, so we can use this for the formal definition:

**Definition 1.1.2** $\psi: SW \times \Delta \rightarrow \mathbb{N}$ is defined by:

$$(s, a) \mapsto \{b^i \in A_s \mid b = a\}$$

This allows us to introduce some further notions.

**Definition 1.1.3** For a poset $(A, \leq)$ and a set $B$ the restriction of $(A, \leq)$ to $B$ (written $(A, \leq)|_B$) is defined to be the poset $(A|_B, \leq|_B^2)$.

$s$ is a direct subsemiword of $t$ iff $s$ is a semiword and $s = t|_{A_s}$.

If $s$ is a direct subsemiword of $u$ the complement semiword $t$ of $s$ (w.r.t. $u$) in $u$ is defined to be $t|_{A_u \setminus A_s}$ shifted left according to $s$. I.e.,

$$A_t := \{a^{-\psi(s,a)} \mid a^i \in A_u \setminus A_s\}$$

$$\forall a^i, b^j \in A_t. \ a^i \leq t \ b^j \iff a^{i+\psi(s,a)} \leq u|_{(A_u \setminus A_s)^2} b^{j+\psi(s,b)}$$

For convenience direct subsemiwords will be referred to simply as subsemiwords.

One could have defined a more general notion of subsemiword, but with this definition the subsemiword is directly represented by itself. Furthermore this definition suffice for our purpose.

**Example:** Let $u$ be the semiword $a \rightarrow b \rightarrow c \rightarrow d \rightarrow b$

$s_i$ below are subsemiwords of $u$ and $t_i$ complement semiword of $s_i$ in $u$. 

20
i) $s_i$  

a) $a \rightarrow d$  

b) $b \rightarrow c \rightarrow b$  

c) $a \leftarrow a \rightarrow d$  

d) $a \rightarrow b \leftarrow c \rightarrow d$  

e) $a \rightarrow b$  

f) $b$

The following semiwords are not subsemiwords of $u$:

\[
\begin{align*}
 & b \leftarrow a \rightarrow b, & b \rightarrow a \\
 & b \leftarrow c \rightarrow b, & b \leftarrow d \rightarrow b, & b \rightarrow a
\end{align*}
\]

From $t_a$ in the example we see that although a semiword is not a subsemiword of a semiword $u$, it can be a complement semiword. b) and c) shows that a complement semiword and a subsemiword may change rôle, whereas d) and e) shows it is not always the case. Also notice that although e.g., $b \rightarrow a$ is a direct part of the picture of $u$ (i.e., $b \rightarrow a$ is a subsemiword in a more general sense) it is neither a subsemiword of $u$ nor a complement semiword in $u$.

**Proposition 1.1.4**

a) If $A \subseteq A_s$ fulfills SW1 then $s|_A$ is a subsemiword.

b) The complement semiword in definition 1.1.3 is in fact a semiword.

**Proof**

a) We shall show that $\leq_{s|A^2}$ is a po and fulfills SW2.

Since $\leq_s$ is a po on $A_s$, it must be so on $A \cap A_s$ too. Similar the SW2-property must hold on $A \cap A_s$ also.

b) First we prove that $A_u$ fulfills SW1.

Let $a^i \in A_t$ and a $j$ such that $1 \leq j \leq i$. We shall prove $a^j \in A_t$.

By definition of $t$ it follows that $a^i \in A_t$ implies $a^{i+v(s,a)} \in A_u \setminus A_s \subseteq A_u$. Now $1 \leq j \leq i$
implies $1 \leq j + \psi(s, a) \leq i + \psi(s, a)$, so because $u$ is a semiword and therefore fulfills $SW1$ we have $a^{j+\psi(s,a)} \in A_u$. $a^{j+\psi(s,a)} \notin A_s$ because otherwise $j + \psi(s, a) \leq \psi(s, a)$ which obviously is impossible since $1 \leq j$. Hence $a^{j+\psi(s,a)} \notin A_s$. Then by definition of $t$: $a^{(j+\psi(s,a)) - \psi(s,a)} = a^j \in A_t$.

$\leq_u$ is a pos fulfilling $SW2$ so this holds for $\leq_{u | (A_u \setminus A_s)^2}$ too. By definition of $\leq_t$ this must then also be the case for $\leq_t$. □

**Definition 1.1.5** Two semiwords $s$ and $t$ are said to be disjoint iff $A_s \cap A_t = \emptyset$. □

Because $s, t$ disjoint implies $\forall a^i \in A_t \psi(s, a) = 0$ we get:

**Corollary 1.1.6**

a) If $s$ is a subsemiword of $u$ then: $\leq_s \subseteq \leq_u$.

b) and if $t$ is the complement semiword of $s$ in $u$ and disjoint to $s$ then $A_s \cup A_t = A_u$ and $\leq_t \subseteq \leq_u$.

**Definition 1.1.7** Given a poset $(A, \leq)$.

$a, b \in A$ are connected iff $a$ and $b$ are connected when considering the undirected graph of $\leq$, i.e., when $(a, b) \in (\leq \cup \leq^{-1})^+.$

A poset $(C, \leq')$ is a maximal connected component of a poset $(A, \leq)$ iff $(C, \leq') = (A, \leq)|_C$ and all elements of $C$ are connected and there is no $a \in C$ and $b \in A \setminus C$ which are connected. For the sake of convenience we will in the sequel just say connected component instead of maximal connected component.

The set of all connected components of a poset $(A, \leq)$ is denoted $\gamma((A, \leq))$.

Since $SW$ is a subclass of the posets, we can talk of the connect components of a semiword as well.

If $s$ is a semiword such that $\gamma(s) = \{\varepsilon, s\}$ we say that $s$ is a connected semiword.

□

It is not difficult to see:

**Corollary 1.1.8**

a) A connected component of a semiword is also a subsemiword of it.

b) For a semiword $s$, $\gamma(s)$ consists of mutual disjoint semiwords.

c) $s \mapsto \gamma(s)$ can be considered as a function $\gamma : SW \to \mathcal{P}(SW)$.

d) $\{\varepsilon, s\} \subseteq \gamma(s)$.

e) $\gamma(s) = \{\varepsilon, s\}$ iff $s$ is a connected component.

where $\mathcal{P}(A)$ denote the power set of $A$. 22
1.2 Operations on $SW$

In this section we shall introduce some of the operators on $SW$ presented by Starke in [Sta81] where he also displays the most fundamental properties of the operators.

**Nullary**

The semiword with canonic representative $(\emptyset, \emptyset)$ is denoted $\varepsilon$ and is called the empty (semi-) word. For every action $a \in \Delta$ we select a corresponding semiword which has the canonic representative $(\{a^1\}, \{(a^1, a^1)\})$ and denote it $a$.

**Corollary 1.2.1**

\[ a) \; \gamma(s) = \{\varepsilon\} \iff s = \varepsilon \]
\[ b) \; \gamma(a) = \{\varepsilon, a\}. \]

**Unary**

With the previous nullary operators and the concatenation of semiwords defined below, we easily derive an unary operator $a: : SW \rightarrow SW$ for every $a \in \Delta$. Namely:

**Definition 1.2.2** Let $a \in \Delta$. Then $a: : SW \rightarrow SW$ is defined by $s \mapsto a \cdot s$, where $a \cdot s$ means the the concatenation of $a$ and $s$.

From the properties of concatenation we derive:

**Corollary 1.2.3**

\[ a) \; a \cdot s = a \iff s = \varepsilon \]
\[ b) \; a \cdot s = b \cdot t \iff a = b, s = t \]
\[ c) \; a \cdot s \in W \iff s \in W \]

**Binary**

The definition of *concatenation* displayed as juxtaposition of the operands or placing a . (dot) between the operands is:

**Definition 1.2.4** Concatenation of semiwords, $\cdot : SW \times SW \rightarrow SW$, is defined by $(s, t) \mapsto s \cdot t = st = (A, \leq)$, where

\[ A = A_s \cup \{a_i^{\psi(s,a)} \mid a_i \in A_t\} \]
\[ \leq \subseteq A \times A \text{ is defined by:} \]
\[ \forall a^i, b^j \in A, a^i \leq b^j \iff \]
\[ i \leq \psi(s, a), j \leq \psi(s, b), a^i \leq_s b^j \]
\[ \text{or} \]
\[ i \leq \psi(s, a), \psi(s, b) < j \]
\[ \text{or} \]
\[ \psi(s, a) < i, \psi(s, b) < j, a_i^{\psi(s,a)} \leq t_b^{\psi(s,b)} \]
Notice that $A_s \subseteq A_{st}$ and $\forall a \in A_s \forall b \in A_{st} \setminus A_s. a \leq_{st} b$.

Example:

\[
\begin{align*}
&\quad\quad a \prec a \prec b, \quad b \prec b \prec c = a \prec a \prec b \prec b \prec c
\end{align*}
\]

**Corollary 1.2.5** For all $s, t, u \in SW$:

1. $st \in SW$ (well-defined)
2. $s(tu) = (st)u$ (associative)
3. $\varepsilon s = s = s\varepsilon$ (\(\varepsilon\) unit)
4. $st = su \Rightarrow t = u$ (left cancellation)
5. $ts = us \Rightarrow t = u$ (right cancellation)

Recalling the definition of subsemiword and complement semiword and inspecting the definition of concatenation, we immediately get:

**Corollary 1.2.6** $s$ is a subsemiword of $st$ and $t$ the complement semiword of $s$ in $st$.

The close connection between sub- and complement semiwords of a semiword and concatenation can be further illuminated by the following:

**Proposition 1.2.7** Let $s$ be a subsemiword of $u$. Define $t$ to be the complement semiword of $s$ (w.r.t. $u$). Then:

1. $A_u = A_{st}$
2. $\forall a, b \in A_s. a \leq_u b \iff a \leq_{st} b$
3. $\forall a \in A_s \forall b \in A_u \setminus A_s. a \leq_u b \Rightarrow a \leq_{st} b$
4. $\forall a, b \in A_u \setminus A_s. a \leq_u b \iff a \leq_{st} b$

Notice that we cannot conclude $u = st$ from a) – d). Later when dealing with partial orders on $SW$, we will see some conditions which ensure that there exist such $s$ and $t$. 

24
Proof
a) \(A_{st} = A_s \cup \{a^i \psi(s, a) \mid a^i \in A_t\} = A_s \cup \{a^i \psi(s, a) \mid a^i \in A_u \setminus A_s\} = A_s \cup \{(a^{-1} \psi(s, a)) \psi(s, a) \mid a^j \in A_u \setminus A_s\} = A_s \cup (A_u \setminus A_s) = A_u\), where the last equation follows from the fact that \(s\) being a subsemiword of \(u\) implies \(A_s \subseteq A_u\).

b) \(s\) is a subsemiword of \(u\) wherefore \(\leq_s\) agrees with \(\leq_u\) restricted to \(A_s\). \(s\) is also subsemiword of \(st\) and the result follows.

c) Because of a) we see \(A_u \setminus A_s = A_{st} \setminus A_s\), so the rest is trivial since we have already noticed by definition of concatenation that \(\forall a \in A_s \forall b \in A_{st} \setminus A_s\). \(a \leq_{st} b\) (no matter whether \(a \leq_u b\) or not).

d) Assume \(a\) and \(b\) actually are \(a^i\) and \(b^j\) respectively. Since \(a^i, b^j \in A_u \setminus A_s\) we have \(a^i \leq_u b^j \iff a^i \leq u|_{(A_u \setminus A_s)^2} b^j\) which by definition of \(t\) is equivalent to \(a^i \psi(s, a) \leq t b^j \psi(s, b)\) (notice that \(a^i, b^j \in A_u \setminus A_s, A_s \subseteq A_u \Rightarrow \psi(s, a) < i, \psi(s, b) < j\)). This again, by definition of concatenation, is equivalent to \(a^i \leq_{st} b^j\).

\[ \square \]

Proposition 1.2.8 Let \(u\) be a connected nonempty semiword. Then:

\[ \begin{align*} 
\text{a)} & \quad \gamma(ut) = \{\varepsilon, ut\} \\
\text{b)} & \quad \gamma(su) = \{\varepsilon, su\} \\
\text{c)} & \quad s, t \neq \varepsilon \Rightarrow \gamma(st) = \{\varepsilon, st\} 
\end{align*} \]

Proof
a) Assume \(\gamma(u) = \{\varepsilon, u\}\). By corollary 1.1.8.d) \(\{\varepsilon, ut\} \subseteq \gamma(ut)\). So what remains to be proved is \(r \in \gamma(ut) \Rightarrow r \in \{\varepsilon, ut\}\). One consequence of \(r \in \gamma(ut)\) is \(r = ut|_{A_r}\). If either \(A_r = \emptyset\) or \(A_r = A_{ut}\) the result is clear, so assume \(\emptyset \neq A_r \neq A_{ut}\).

Then there exist \(a \in A_r, b \in A_{ut} \setminus A_r\) and since \(r\) is a connected component of \(ut\), \(a\) and \(b\) cannot be connected. We look at the different possible memberships of \(a\) and \(b\) w.r.t. \(A_u\) and \(A_{ut}\).

\(a, b \in A_u \subseteq A_{ut}\): Since \(u\) is connected \(a\) and \(b\) must be connected—a contradiction.

\(a \in A_u, b \in A_{ut} \setminus A_u\): Then as noticed by concatenation \(a \leq_{ut} b\) and thereby connected—again a contradiction.

\(b \in A_u, a \in A_{ut} \setminus A_u\): Similar.

\(a, b \in A_{ut} \setminus A_u\): Since \(\{\varepsilon, u\} \neq \{\varepsilon\} \Rightarrow u \neq \varepsilon\) there exists a \(c \in A_u\). Again as noticed by concatenation \(c \leq_{ut} a\) and \(c \leq_{ut} b\). Hence \(a\) and \(b\) are connected and we get a contradiction again.

We have exhausted all possible memberships of \(a, b\) and each time got a contradiction, so the assumption \(\emptyset \neq A_r \neq A_{ut}\) was wrong. Hence \(\varepsilon, ut\) are the only connected components of \(ut\).

b) Similar.

c) Let \(a, b \in A_{st}\). If we can show that they are connected corollary 1.1.8.e) gives \(\gamma(st) = \{\varepsilon, st\}\). Three cases to consider.

\(a, b \in A_{st}\): Since \(t \neq \varepsilon\) we have a \(c \in A_{st} \setminus A_{st}\). By proposition 1.2.7.d) \(a \leq_{st} c, b \leq_{st} c\), so connected.
\(a \in A_s, b \in A_{st} \setminus A_s\): proposition 1.2.7 gives directly that they are connected.

\(a, b \in A_{st} \setminus A_s\): Since \(s \neq \varepsilon\) we have some \(c \in A_s\). Again by proposition 1.2.7 we have \(c \leq_{st} a, c \leq_{st} b\) and thereby connected.

\(\square\)

For words we have the following connection:

**Corollary 1.2.9** \(st \in W \iff s, t \in W\).

The parallel composition of semiwords is defined:

**Definition 1.2.10** Let \(s, t\) be two disjoint semiwords. Then the *parallel composition* of \(s\) and \(t\) is:

\[s \parallel t := (A_s \cup A_t, \leq_s \cup \leq_t)\]

\(\square\)

So parallel composition is only partially defined.

**Example:**

\[a \overset{\parallel}{\longrightarrow} b \overset{c}{\leftarrow} d = a \overset{\parallel}{\longrightarrow} b \overset{c}{\leftarrow} d\]

**Corollary 1.2.11** For all \(s, t, u \in SW\), mutual disjoint:

\[
\begin{align*}
a) & \ s \parallel t \in SW & \text{(well-defined)} \\
b) & \ s \parallel t = t \parallel s & \text{(commutative)} \\
c) & \ (s \parallel t) \parallel u = s \parallel (t \parallel u) & \text{(associative)} \\
d) & \ \varepsilon \parallel s = s = \varepsilon \parallel s & \text{($\varepsilon$ unit)} \\
e) & \ s \parallel t = s \parallel u \Rightarrow t = u & \text{(left cancellation)} \\
f) & \ t \parallel s = u \parallel s \Rightarrow t = u & \text{(right cancellation)}
\end{align*}
\]

Since \(\parallel\) is associative we can omit brackets. Furthermore because \(\parallel\) additionally is commutative and has \(\varepsilon\) as neutral element, we can even for a set \(D\) of semiwords write \(\|\{s \mid s \in D\}\) or just \(\|D\) for short to denote \(s_1 \| s_2 \| \ldots \| s_n\) where \(D = \{s_1, \ldots, s_n\}\). If \(D = \emptyset\) then \(\|D\) denotes \(\varepsilon\).

To avoid the proviso of disjointness of semiwords whenever writing expressions involving \(\parallel\) we will in the sequel tacitly assume this.
Corollary 1.2.12 If $s$ is a subsemiword of $u$ and $t$ the complement semiword of $s$ in $u$, similar for $s', t', u'$ then:

a) $s \parallel s'$ is a subsemiword of $u \parallel u'$ and

b) $t \parallel t'$ is the complement semiword of $s \parallel s'$ in $u \parallel u'$.

Proposition 1.2.13

a) $\gamma(s \parallel t) \setminus \{\varepsilon\} = \gamma(s) \setminus \{\varepsilon\} \uplus \gamma(t) \setminus \{\varepsilon\}$

b) $s = \parallel \gamma(s)$

where $\uplus$ means disjoint union of sets.

Proof

a) Trivial.

b) By induction on the size of $\gamma(s)$

$|\gamma(s)| = 1$: Since $\varepsilon \in \gamma(s)$ for all $s \in SW$ we have $s = \varepsilon$ and $\gamma(s) = \{\varepsilon\}$ from which the result follows.

$|\gamma(s)| > 1$: Then there is a $t \in \gamma(s)$ with $t \neq \varepsilon$. Clearly $r = s|_{A \cup A'}$ is a subsemiword of $s$ (corollary 1.1.8) and $s = t \parallel r$. Since $t \neq \varepsilon$ and hence $A_t \neq \emptyset$ we must have $|\gamma(r)| < |\gamma(s)|$ and the result then follows by applying the inductive hypothesis on $r$ and using a).

From this proposition and the other concerning $\gamma$ we obtain the following corollary.

Corollary 1.2.14

a) $a.s = t_1 \parallel t_2 \Rightarrow \begin{cases} t_1 = a.s, t_2 = \varepsilon \\
 or \\
 t_1 = \varepsilon, t_2 = a.s \end{cases}$

b) $s_1 \parallel s_2 = t_1 \parallel t_2$

$\downarrow$

$\exists t_i^j(\in \gamma(t_1) \cup \gamma(t_2)) i, j \in 2 \quad s_i = t_i^j, t_i = t_i^1 \parallel t_i^2, i \in 2$

c) $\varepsilon = s \parallel t \Rightarrow s = \varepsilon = t$.

1.3 Partial Orders on $SW$

There are more natural partial orders on $SW$ of which we shall see two in this section.
1.3.1 Smoother Than

The idea of one semiword, $s$, being smoother than another, $t$, i.e., $\leq_s$ is a refinement of $\leq_t$, can be captured formally as follows:

**Definition 1.3.1** Let $s, t \in SW$. Then $s$ is smoother than $t$ (written $s \preceq t$) iff $A_s = A_t$ and $\leq_s \supseteq \leq_t$. \hfill $\Box$

**Example:**

\[
\begin{array}{cccc}
  a & b & c \\ 
  \preceq & \preceq & \preceq \\
  a & b & c \\
\end{array}
\]

Both = and $\subseteq$ are partial orders so evidently:

**Corollary 1.3.2** $\subseteq$ partial orders $SW$.

**Corollary 1.3.3** If $s$ is a subsemiword of $u$, $t$ the complement semiword in $u$ and $s, t$ disjoint then $u \preceq s \parallel t$.

The truth of this is evident since $s, t$ disjoint implies $A_s \cup A_t = A_u$, $s$ subsemiword of $u$ implies $\leq_s \subseteq \leq_u$ and $t$ disjoint complement (sub)semiword of $s$ in $u$ implies $\leq_t \subseteq \leq_u$.

Looking at this corollary one might think that $s$ being a subsemiword of $u$ and $t$ the complement implies $st \preceq u$, but this is not in general true as can be seen from the following example.

**Example:** Let $u = a \searrow b \nearrow d$. Then $s = a \nearrow b \nearrow d$ is a subsemiword of $u$ and $t = c$ the complement semiword. But $st \not\preceq u$ because $c \leq u$ and $c \not\leq st$.

Later in proposition 1.3.28 we will see a sufficient condition for $st \preceq u$.

Having defined $\preceq$ we are able to define the set of linearizations or the smoothing of a semiword $s$, written $\lambda(s)$.

**Linearizations:** $\lambda$

**Definition 1.3.4** Define $\lambda: SW \rightarrow \mathcal{P}(W)$ by

\[ s \mapsto \{t \in W \mid t \preceq s\} \]

\hfill $\Box$

**Proposition 1.3.5** For all $s, t \in SW$ we have
Before we proceed with the proof we need some small lemmas.

**Lemma 1.3.6** \( \forall s \in SW \forall a, b \in A_s. (a \not\leq_s b, b \not\leq_s a \Rightarrow \exists t \in SW. t < s, a \leq_t b) \)

**Proof** The idea is to get a smoothing of \( \leq_s \) by adding \((a, b)\) to \( \leq_s \) and take the transitive closure. Given \( s \in SW \) and \( a, b \in A_s \) such that \( a \not\leq_s b, b \not\leq_s a \). Define \( t \) by \( A_t := A_s \), \( \leq_t := R^+ \), where \( R = \leq_s \cup Q, Q = \{(a, b)\} \). Clearly \( \leq_s \subseteq \leq_t \) and thereby \( t < s \), so the only problem is to see \( t \in SW \).

SW1 holds for \( t \) since \( A_t = A_s \). Because \( \leq_s \subseteq \leq_t \) and \( \leq_s \) is reflexive, SW2 holds for \( \leq_t \). By construction \( \leq_t \) is transitive.

Before considering the antisymmetry we prove:

\[(1.1) \forall c, d \in A_t. c \, R^n \, d, c \not\leq_s d \Rightarrow c \leq_s a, b \leq_s d.\]

by induction on \( n \).

\( n = 1 \): Because \( c \not\leq_s d \) we must have \( c \, Q \, d \). Then \( a = c, d = b \) and the result follows by the reflexivity of \( \leq_s \).

\( n > 1 \): Then \( c \, R^{n-1} \, e, e \, R \, d \) for some \( e \in A_t (= A_s) \). Two cases:

\( c \leq_s e \): From \( c \not\leq_s d \) and the transitivity of \( \leq_s \) we conclude \( e \not\leq_s d \). By hypothesis of induction \( e \leq_t a, b \leq_t d \) and the result follows.

\( c \not\leq_s e \): By hypothesis of induction \( c \leq_t a, b \leq_t e \). If \( e \leq_t d \) then transitivity gives \( b \leq_t d \) and if \( e \not\leq_t d \) the hypothesis yields \( b \leq_t d \) directly.

Now to see:

\[(1.2) \quad c \, R^n \, d, d \, R^m \, c \Rightarrow c = d\]

assume on the contrary that there exists \( c, d \in A_t \) such that \( c \, R^n \, d, d \, R^m \, c, c \neq d \). Since \( \leq_s \) is antisymmetric we have either \( c \not\leq_s d \) or \( d \not\leq_s c \). We investigate the different cases:

\( c \not\leq_s d, d \not\leq_s c \): Since \( c \, R^n \, d, d \, R^m \, c \) (1.1) gives \( c \leq_t a, b \leq_s d, d \leq_s a, b \leq_s c \). From \( b \leq_s c, c \leq_s a \) we get \( b \leq_s a \) which is a contradiction to \( b \not\leq_s a \).

\( c \leq_s d, d \not\leq_s c \): Since \( d \, R^n \, c \) we have \( d \leq_s a, b \leq_s c \). Collecting these facts: \( b \leq_s c \leq_s d \leq_s a \) and thereby \( b \leq_s a \)—again a contradiction.

\( c \not\leq_s d, d \leq_s c \): Similar.

We have exhausted the cases and each time got a contradiction, so the assumption was wrong. From (1.2) the antisymmetry of \( \leq_t \) then follows.

Since \( s \not\in W \) implies \( \exists a, b \in A_s. a \not\leq_s b, b \not\leq_s a \) it follows that:

**Corollary 1.3.7** \( \forall s \in SW : s \not\in W \Rightarrow \exists t \in SW. t < s. \)
Lemma 1.3.8 For all \( s \in SW \) we have: \( \lambda(s) = \{ t \in SW \mid t \leq s, \not\exists t'. t' < t \} \).

Proof

a) Immediate since \( s \in SW \).

b) We prove \( \not\exists t'. t' < t \). So assume on the contrary that there exists a \( t' < t \). Now \( t' < t \) implies \( s \leq t' \). But \( s \leq t' \) means \( \exists a, b \in A_{t'} \mid a \leq b \). We cannot have \( b \leq a \) since this, by \( a \leq t \), implies \( b \leq t \). But \( a \leq t \) means \( \exists s \in SW \mid a \leq s \). By the previous lemma we have \( s \leq t \) and from \( a \leq t \) we conclude by lemma 1.3.6 that there exists a \( t' \in SW \) such that \( t' < t \). By the transitivity of \( \leq \) we get \( t \leq s \).

Before proving the rest of the proposition we need one more little lemma.

Lemma 1.3.9 For all \( s \in SW \) and \( a, b \in A_s \) we have:

a) \( a \leq s \Rightarrow \forall t \in \lambda(s). a \leq t b \)

b) \( a \not\leq s \Rightarrow \exists t \in \lambda(s). b \leq t a \)

Proof

a) Immediate since \( t \in \lambda(s) \Rightarrow t \leq s \Rightarrow s \subseteq t \).

b) We look at two cases of \( s \).

\( s \in W \): Then \( s \) satisfies the trichotomy law. Choose \( t = s \).

\( s \not\in W \): There are two possibilities. Either \( b \leq s \) or \( b \not\leq s \). If \( b \leq s \) the result follows from a) and \( \lambda(s) \not= \emptyset \). If \( b \not\leq s \) we conclude by lemma 1.3.6 that there exists a \( t' \) such that \( t' < s \) and \( b \leq t' a \). From a) then \( \forall t \in \lambda(t'). b \leq t a \). We have already proved \( t' \leq s \Rightarrow \lambda(t') \subseteq \lambda(s) \) so we are done.
Proof (of proposition 1.3.5.a continued)

We shall prove \( \lambda(s) \subseteq \lambda(t) \) \( \Rightarrow \) \( s \preceq t \) which is equivalent to \( s \not\preceq t \) \( \Rightarrow \) \( \lambda(s) \not\subseteq \lambda(t) \). Assume \( s \not\preceq t \). Two possibilities.

\( A_s \neq A_t \): Then clearly \( \lambda(s) \not\subseteq \lambda(t) \) since in general \( \forall r \in \lambda(s) \). \( A_r = A_s \).

\( A_s = A_t \): Then we must have \( \leq_s \not\subseteq \leq_t \). That means there exists \( a, b \in A_t \) such that \( a \leq_t b \) but \( a \not\leq_s b \). By b) of the previous lemma \( a \not\leq_s b \) implies \( \exists s' \in \lambda(s) \), \( b \leq_{s'} a \). We cannot have \( s' \in \lambda(t) \). Suppose on the contrary we have \( s' \in \lambda(t) \). From a) of the same lemma we have \( a \leq_t b \) implies \( \forall t' \in \lambda(t), a \leq_{t'} b \), hence also \( a \leq_{s'} b \). By the antisymmetry of \( \leq_{s'} \): \( a \leq_{s'} b, b \leq_{s'} a \) implies \( a = b \). Then \( a \not\leq_s b \) means \( a \not\leq_s a \)—a contradiction to the reflexivity of \( \leq_s \).

From this proposition it follows that \( s = t \) iff \( \lambda(s) = \lambda(t) \) and since \( \lambda(s) \) is a finite set of words a natural question is “Why not just consider finite sets of words instead of semiwords?” The main reason we are using semiwords, as opposed to finite sets of words over \( \Delta \), is the semiwords agreement with our intuition of concurrency—the partial order reflecting the dependencies between occurrences of actions and the lack of such a dependency the concurrency. Furthermore they have a very simple graphical representation which supports the intuition. Also the formal mathematical representation (canonic representation) is simple. A consequence is simplified definitions and proofs. Another technical reason is that not every set of words \( T \) (with same multiset) have an \( s \in SW \) such that \( \lambda(s) = T \). For a more detailed discussion of—and look at pos see [Pra86].

The next proposition is concerned with the \( \preceq \)-monotonicity of \( \cdot \) and \( \| \).

**Proposition 1.3.10**

\[ \begin{align*}
\text{a) } & s \leq t \Leftrightarrow sr \leq tr \Leftrightarrow rs \leq rt & \text{b) } & s \leq t \Leftrightarrow r \parallel r \leq t \parallel r.
\end{align*} \]

Proof

a) We look at the different implications one by one.

\( s \leq t \Rightarrow sr \leq tr \): Given \( s \leq t \). Shall prove \( sr \leq tr \) or equally \( A_{sr} = A_{tr}, \leq_{tr} \subseteq \leq_{sr} \). Since \( s \leq t \Rightarrow A_s = A_t \) we see by the definition of concatenation that, in order to prove \( A_{sr} = A_{tr} \), it is enough to prove \( \{ a^{i+\psi(t, a)} \mid a^i \in A_r \} = \{ a^{i+\psi(s, a)} \mid a^i \in A_r \}, \) but this follows directly from \( A_s = A_t \Leftrightarrow \forall a \in \Delta, \psi(s, a) = \psi(t, a) \). To see \( \leq_{tr} \subseteq \leq_{sr} \) we must prove \( \forall a^i, b^j \in A_{tr} (= A_{sr}), a^i \leq_{tr} b^j \Rightarrow a^i \leq_{sr} b^j \). We look at the cases: \( i \leq \psi(t, a), j \leq \psi(t, b) \): This implies by the definition of concatenation \( a^i \leq_t b^j \) which by \( s \leq t \) implies \( a^i \leq_s b^j \). Since \( \psi(s, a) = \psi(t, a), \psi(t, b) = \psi(s, b) \) the result follows for this case. The remaining cases \( (i \leq \psi(t, a), \psi(t, b) < j) \) and \( (\psi(t, a) < i, \psi(t, b) < j) \) follows similarly.

\( s \leq t \Rightarrow rs \leq rt \): Similar.
$$sr \leq tr \Rightarrow s \leq t: A_s \subseteq A_t$$: Assume on the contrary \( A_s \not\subseteq A_t \). Then \( \exists a^i \in A_s, a^i \not\in A_t \). This implies \( \psi(t,a) < i \leq \psi(s,a) \). Clearly \( a^{i+\psi(t,a)} = a^k \in A_{sr} \). Now \( a^k \not\in A_t \) because \( \psi(t,a) < \psi(s,a) \leq k \). Also \( a^k \not\in \{a^{i+\psi(t,a)} | a^i \in A_t \} \).

If not, then \( a^{k-\psi(t,a)} \in A_t \). This gives us \( k - \psi(t,a) \leq \psi(r,a) \) or equivalently \( \psi(r,a) + \psi(s,a) - \psi(t,a) \leq \psi(r,a) \) and thereby \( \psi(s,a) \leq \psi(t,a) \) which contradicts \( \psi(t,a) < \psi(s,a) \). So \( a^k \not\in A_{tr} \). But this contradicts \( A_{sr} = A_{tr} \), so the assumption was wrong and we have \( A_s \subseteq A_t \).

Similarly, we see \( A_t \subseteq A_s \), wherefore \( A_s = A_t \). This also implies \( \psi(s,a) = \psi(t,a) \) for all \( a^i \in A_s (= A_t) \), hence by the definition of concatenation and the fact that \( \leq tr \subseteq \leq sr \), it follows \( \leq t \subseteq \leq s \).

\( rs \leq rt \Rightarrow s \leq t: \) In general we have \( A_r \) and \( \{a^{i+\psi(r,a)} | a^i \in A_u \} \) are disjoint, so \( A_{rs} = A_{rt} \) implies \( \{a^{i+\psi(r,a)} | a^i \in A_s \} = \{a^{i+\psi(r,a)} | a^i \in A_t \} \), hence \( A_s = A_t \). Similar as above we see \( \leq t \subseteq \leq s \).

b) Obvious, since in general the disjointness of \( s \) and \( t \) gives \( A_{s\|t} = A_s \uplus A_t, \leq s\|t = \leq s \uplus \leq t \).

From this proposition and the transitivity of \( \leq \) we immediately get:

**Corollary 1.3.11** If \( s_i \leq t_i \) for \( i \in 2 \) then:

\( \forall i: s_1 s_2 \leq t_1 t_2 \)

The commutativity of \( \| \) is used in seeing b).

**Proposition 1.3.12** For semiwords \( s, t \) and \( u \): \( u \leq st \Rightarrow \exists s' \leq s, t' \leq t, u = s't' \).

**Proof** Given \( u \leq st \). Define \( s' := u|_{A_s} \).

\( s \) is a subsemword of \( st \) wherefore \( A_s \) fulfills SW1. Since \( u \leq st \Rightarrow A_u = A_{st} \) and \( A_s \subseteq A_{st} \) it follows that \( A_s \subseteq A_u \). Hence \( s' \) is a subsemword of \( u \). Then we can define \( t' \) to be the complement semiword of \( s' \) w.r.t. \( u \).

\( u = s't' \): Since \( s' \) is a subsemword of \( u \) and \( t \) the complement proposition 1.2.7.a) gives \( A_u = A_{s't'} \) and b) - d) most of \( \leq u = \leq s't' \). Only three cases remains to be proved and this in a situation with \( a \in A_{s'}, b \in A_u \setminus A_{s'} \). \( u \leq st \) implies \( A_u = A_{st} \) and by definition of \( s' \) we have \( A_{s'} = A_s \), so the situation can be read as \( a \in A_{s'} = A_s, b \in A_{s't'} \setminus A_{s'} = A_{st} \setminus A_s \). As noticed by the definition of concatenation we then have

\[
(1.3) \quad a \leq s't' \quad b \quad a \leq_{st} b
\]

\( a \leq s't' \ L a \leq u b \): From \( u \leq st \) we also have \( \leq st \subseteq \leq u \), wherefore (1.3) gives \( a \leq u b \).

\( b \leq s't' \ L b \leq a \): Trivially true because \( b \not\leq s't' a \). To see this assume otherwise \( b \leq s't' a \) and (1.3) together with the antisymmetry of \( \leq s't' \) would give \( a = b \) which contradicts \( a \) and \( b \) belonging to disjoint sets.

\( b \leq u a \Rightarrow b \leq s't' a \): We cannot have \( b \leq u a \) since (1.3) and the first case implies \( a \leq u b \) which then if \( b \leq u a \) would lead to a contradiction as in the last case.
Proposition 1.3.13

a) \( u \in \lambda(st) \iff \exists s' \in \lambda(s) \exists t' \in \lambda(t). u = s't' \)

b) \( u \in \lambda(s \parallel t) \iff \exists s' \in \lambda(s) \exists t' \in \lambda(t). u \preceq s' \parallel t', \overline{\beta}u'. u' < u. \)

Proof

a) \( \Rightarrow \): By lemma 1.3.8 \( u \in \lambda(st) \) implies \( u \preceq st, \overline{\beta}u'. u' < u. \) From \( u \preceq st \) we see from the last proposition that \( \exists s' \preceq s, \exists t' \preceq t. u = s't', \) so if we can prove \( \exists s'' \prec s' \) and \( \exists t'' \prec t' \) we are done since, then again by lemma 1.3.8 we have \( s'' \in \lambda(s) \) and \( t'' \in \lambda(t). \)

To see \( \exists s'' \prec s' \) assume on the contrary \( \exists s'' \prec s'. \) Then by proposition 1.3.10 \( s'' \prec s' \prec t' \prec u \) a contradiction to \( \overline{\beta}u'. u' < u. \) Similarly, we prove \( \exists t'' \prec t'. \)

\( \Leftarrow \): \( s' \in \lambda(s), t' \in \lambda(t) \) implies \( s' \preceq s \) and \( t' \preceq t \) and by corollary 1.3.11 \( u = s't' \preceq st. \) So in order to have \( u \in \lambda(st) \) we just need to prove \( \exists s'' \prec s' \) or \( \exists t'' \prec t' \) a contradiction to \( s' \in \lambda(s) \) or \( t' \in \lambda(t). \)

b) \( \Rightarrow \): \( u \in \lambda(s \parallel t) \) implies \( \overline{\beta}u'. u' < u. \) Since \( u \in \lambda(s \parallel t) \) means \( u \preceq s || t, \) we have \( A_u = A_s \sqcup A_t, \leq_s \sqcup \leq_t \subseteq \leq_u. \) So \( s' := u|_{A_u}, t' := u|_{A_t} \) are indeed subsemwords of \( u. \)

At first we prove \( s' \preceq s \) and \( t' \preceq t. \) To see \( s' \preceq s \) notice \( A_v = A_u|_{A_s} = A_s \) and \( \leq_{s||t} \subseteq \leq_u \) implies \( \leq_{s||t}|_{A_s} = \leq_s. \) Since \( \leq_s = \leq_{s||t}|_{A_s^2} \) we have \( \leq_s \subseteq \leq_s \) and thereby \( s' \preceq s \). \( t' \preceq t \) is shown similarly.

Now to see \( s' \in \lambda(s) \) we just need to prove \( s' \in W; \) i.e., \( \forall a, t \in A_s', a \leq_s b \lor b \preceq a. \) Let \( a, b \in A_s' \subseteq A_s || A_t \) be given. Since \( u \in \lambda(s || t) \) and thereby \( u \in W \) we have \( a \leq_u b \lor b \leq_u a. \) W.l.o.g. assume \( a \leq_u b. \) From \( a, b \in A_s \) we see \( a \leq_u b \) implies \( a \leq_{u|_{A_s^2}} b \) or what is the same \( a \leq_{s'} b. \) The proof of \( t' \in W \) is done in the same way.

To complete this implication we finally have to show \( u \preceq s' \parallel t' \) or what comes to the same: since \( A_u = A_s \sqcup A_t = A_{s'} \cup A_{t'} \) that \( \leq_{s'} \cup \leq_{t'} \subseteq \leq_u. \) But this follows evidently since \( \leq_{s'} = \leq_{u|_{A_{s'}}} \) and \( \leq_{t'} = \leq_{u|_{A_{t'}}}. \)

\( \Leftarrow \): Because \( \overline{\beta}u'. u' < u \) is assumed we see from lemma 1.3.8 that all we have to show is that \( u = s' \parallel t' \preceq s \parallel t. \) Since \( s' \in \lambda(s) \) and \( t' \in \lambda(t) \) imply \( s' \preceq s \) and \( t' \preceq t \) we immediately get the result from corollary 1.3.11. \( \square \)

Corollary 1.3.14

a) \( \lambda(s)\lambda(t) = \lambda(st) \)

b) \( a.\lambda(t) = \lambda(a.t) \)

\( \preceq \)-downwards closure: \( \delta \)

In the following we are concerned with the full \( \preceq \)-downward closure. We will abbreviate \( DC_\preceq(s) \) by \( \delta(s). \) Notice that \( \delta(s) \) is a finite set since \( A_s \) is finite and so only finitely many refinements of \( \leq_s \) are possible. Also \( \lambda(s) = W \cap \delta(s). \) Both \( \delta \) and \( \lambda \) are extended to sets of semiwords in the natural way. E.g., if \( S \) is a set of semiwords then \( \delta S = \bigcup_{s \in S} \delta(s). \)
Proposition 1.3.15

a) \( s \in \delta(s) \)

b) \( \delta(\varepsilon) = \{ \varepsilon \} \) and \( \forall a \in \Delta. \delta(\underline{a}) = \{ \underline{a} \} \)

c) \( \delta(st) = \delta(s)\delta(t) \)

d) \( \delta(s) \parallel \delta(t) \subseteq \delta(s \parallel t) \)

Before giving the proof we observe the following immediate consequence:

Corollary 1.3.16

a) \( \delta(a.s) = a.\delta(s) \)

b) \( \delta(s \parallel t) = \delta(\delta(s) \parallel \delta(t)) \)

b) of corollary 1.3.16 is seen as follows: \( \subseteq \) from d) implies \( \delta(\delta(s) \parallel \delta(t)) \subseteq \delta(\delta(s) \parallel t) \) and \( \supseteq \) by \( s \parallel t \in \delta(s) \parallel \delta(t) \Rightarrow \delta(s \parallel t) \subseteq \delta(\delta(s) \parallel \delta(t)) \).

Proof (of proposition 1.3.15)

a) By the reflexivity of \( \leq \).

b) Follows directly from a) and the fact that \( A_{\varepsilon} = \emptyset \) (\( A_{\underline{a}} = \{ a^1 \} \)) allows no refinement of \( \leq_{s} = \emptyset \) (\( \leq_{\underline{a}} = \{ (a^1, a^1) \} \)).

c) Clear from proposition 1.3.12 and corollary 1.3.11

d) \( u \in \delta(s) \parallel \delta(t) \Rightarrow (\exists s', t'. s' \leq s, t' \leq t, u = s' \parallel t') \Rightarrow u = s' \parallel t' \leq s \parallel t \Rightarrow u \in \delta(s \parallel t) \). □

\( \leq \)-upwards closure: \( \nu \)

Similar to the abbreviation of \( DC_{\leq}(s) \) by \( \delta(s) \) we abbreviate \( UC_{\leq}(s) \) upwards closure of \( s \) w.r.t. \( \leq \) by \( \nu(s) \) and extend \( \nu \) to sets in the natural way. We have already seen that \( \delta(s) \) is a finite set. The same turns out to be true for \( \nu(s) \) because \( A_{s} \) is finite and so only finitely many coarsenings of \( \leq_{s} \) is possible. Whereas every po consistent refinement (i.e., it is reflexive, antisymmetric, transitive) of \( \leq_{s} \) to \( \leq_{s'} \) yields a semiword \( s' \), this is not the case for every po consistent coarsening. For example, if

\[
s = (\{ a^1, a^2 \}, \{(a^1, a^2), (a^1, a^1), (a^2, a^2)\})
\]

then the only possible po consistent coarsening of \( \leq_{s} \) is

\[
\{(a^1, a^1), (a^2, a^2)\}
\]

which isn’t a semiword (violates SW2).

Before we continue with properties of \( \nu \) we prove:

Proposition 1.3.17 If \( s \) and \( t \) are disjoint we have

\[
s \parallel t \leq u \Rightarrow \exists s', t'. s \leq s', t \leq t' \text{ and } u = s' \parallel t'
\]
Proof. Given \( s \parallel t \leq u \). Define \( s' := u|_{A_s} \) and \( t' := u|_{A_t} \).

At first we show \( u = s' \parallel t' \).

By definition \( s \parallel t \leq u \) means \( A_{s\parallel t} = A_u \) and \( \leq_{s\parallel t} \supseteq \leq_u \). From \( A_{s\parallel t} = A_s \cup A_t \) we see \( A_u = A_s \cup A_t, \) so \( A_u = A_u|_{A_s \cup A_t} = A_{s'} \cup A_{t'} \). Hence \( A_{s'} \cup A_{t'} \) fulfills SW1 and by proposition 1.1.4 \( s' \parallel t' \) are subsemiswords of \( u \). Clearly they are disjoint, so \( s' \parallel t' \) well-defined. Since \( A_{s'\parallel t'} = A_{s'} \cup A_{t'} \) we have \( A_{s'\parallel t'} = A_u. \) \( s \parallel t \leq u \) implies \( \leq_u \subseteq \leq_{s'\parallel t'} \subseteq \leq_{s'} \subseteq \leq_{s'} \cup \leq_{t'} \subseteq \leq_{s'\parallel t'} \) which on second thoughts is seen to imply \( \leq_u \subseteq \leq_{A_{s'} \cup A_{t'}} \subseteq \leq_{s'\parallel t'} \). But \( \leq_u \subseteq A_{s'} \cup A_{t'} \). Clearly they are disjoint, so \( s' \parallel t' \) are subsemiswords of \( u \) with second thoughts is seen to imply \( s' \parallel t' \).

Secondly we prove \( s \leq s' \) and \( t \leq t' \).

To see \( s \leq s' \) at first notice \( A_s = A_{s'} \) by construction, so the proof of \( s \leq s' \) reduces to \( s \leq s' \subseteq s \). \( s \parallel t \leq u \) implies \( \leq_u \subseteq \leq_{s\parallel t} \subseteq \leq_{s'}, \) which again implies \( \leq_u |_{A_{s'}} \subseteq \leq_{A_{s'}} \). Since \( \leq_{s'} \subseteq \leq_u |_{A_{s'} \cup A_{t'}} \) and \( \leq_{s\parallel t} |_{A_{s'} \cup A_{t'}} \subseteq s \) we are done.

\( t \leq t' \) is seen in the same way.

We are now ready to state and prove the following properties of \( u \).

Proposition 1.3.18

a) \( s \in v(s) \)

b) \( v(\varepsilon) = \{\varepsilon\} \) and \( \forall a \in A \, v(a) = \{a\} \)

c) \( v(s) v(t) \subseteq v(st) \)

d) \( v(s \parallel t) = v(s) \parallel v(t) \)

Corollary 1.3.19

a) \( v(st) = v(v(s)v(t)) \)

b) \( v(a.s) = v(a.v(s)) \)

\( \subseteq \) of a) is seen from \( st \in v(s)v(t) \) and \( \supseteq \) from c) of the proposition using \( v(v(st)) = v(st) \).

Proof (of proposition 1.3.18)

a) Follows from the reflexivity of \( \leq \).

b) \( A_\varepsilon = \emptyset \) allows no coarsening. No coarsening of \( \{(a^1, a^1)\} \) is po consistent—fails the reflexivity.

c) \( u \in v(s)v(t) \Rightarrow \exists s', t', s \leq s', t \leq t', s't' \Rightarrow \) (corollary 1.3.11) \( st \leq s't' = u \Rightarrow u = s't' \in v(st) \).

d) \( \subseteq \) follows from the last proposition and \( \supseteq \) from corollary 1.3.11

Notice that in general \( \delta(s \parallel t) \neq \delta(s) \parallel \delta(t) \) and \( v(st) \neq v(s)v(t) \) when \( s, t \neq \varepsilon \). This can be seen by \( st \in \delta(s \parallel t) \) but \( st \not\in \delta(s) \parallel \delta(t) \) if \( s, t \neq \varepsilon \) and if \( s \) and \( t \) are disjoint and different from the empty (semi)word. Under the same conditions \( s \parallel t \subseteq v(st) \) but \( s \parallel t \not\subseteq v(s)v(t) \).
**≤-convex closure:** χ

The preceding up- and downwards closures w.r.t. ≤, δ and v, were defined for single semiwords and extended to sets in the natural way. This cannot be done in the same way for the convex closure, χ, we are going to define now.

**Definition 1.3.20** Let T be a (finite) set of semiwords. Then the convex closure of T written $\chi T$ is defined by:

$$\chi T := \{ s \in SW \mid \exists t, t' \in T. t \leq s \leq t' \}$$

From the definition of χ it appears:

**Corollary 1.3.21** $\chi T = \delta T \cap v T$.

As for δ and v we derive some fundamental properties of χ.

**Proposition 1.3.22** For $S, T \subseteq SW$ we have

a) $T \subseteq \chi T$

b) $\chi \{s\} = \{s\}$ for $s \in SW$

c) $\chi S \chi T \subseteq \chi (ST)$

d) $\chi S \parallel \chi T \subseteq \chi (S \parallel T)$

e) $\chi S \cup \chi T \subseteq \chi (S \cup T)$

Since $\chi \chi S = \chi S$ we can use a) to derive the opposite inclusions of c) – e) and so obtain:

**Corollary 1.3.23**

a) $\chi (ST) = \chi (\chi S \chi T)$

b) $\chi (S \parallel T) = \chi (\chi S \parallel \chi T)$

c) $\chi (S \cup T) = \chi (\chi S \cup \chi T)$

**Proof** (of proposition 1.3.22)

Now first notice that in general if □—an operator between sets—can be considered as the natural extension of an operator, □, between members of these sets, then:

(1.4) $(A \cap B) \Box (C \cap D) \subseteq (A \Box C) \cap (B \Box D)$.

a) – b) Immediate.

c) $\chi S \chi T = (\text{corollary 1.3.21}) (\delta S \cap v S)(\delta T \cap v T) \subseteq (\text{by (1.4)}) (\delta S \delta T) \cap (v S v T) = (\text{proposition 1.3.15.c}) \delta (ST) \cap v (ST) = \chi (ST)$.

d) $\chi S \parallel \chi T = (\delta S \cap v S) \parallel (v T \cap v T) \subseteq (\delta S \parallel \delta T) \cap (v S \parallel v T) \subseteq (\text{proposition 1.3.15.d}) \delta (S \parallel T) \cap v (S \parallel T) = \chi (S \parallel T)$.

e) $\chi S \cup \chi T = (\delta S \cup v S) \cup (\delta T \cap v T) \subseteq (\delta S \cup \delta T) \cap (v S \cup v T) = \delta (S \cup T) \cap v (S \cup T) = \chi (S \cup T)$. □
That the opposite inclusion in c) – e) of proposition 1.3.22 does not hold as can be seen by the following counter examples. Let \( S = \{ \varepsilon, a \rightarrow b \rightarrow c \} \) and \( T = \{ \varepsilon, \frac{a}{c} \} \). Then \( S \cup T \subseteq ST, S \parallel T \) and \( s = a \frac{b}{c} \in \chi(ST), \chi(S \parallel T), \chi(S \cup T) \). But \( \chi S = S \) and \( \chi T = T \), so \( s \not\in \chi S, \chi S \parallel \chi T, \chi S \cup \chi T \).

Now for a special version of corollary 1.3.23.

**Proposition 1.3.24**

\[
a) \ \chi(S \cup \{ \varepsilon \}) = \chi S \cup \chi \{ \varepsilon \} = \chi S \cup \{ \varepsilon \} \quad \text{b) } \chi(\{s\}T) = \{s\}\chi T
\]

**Proof**

a) Evident since \( t = \varepsilon \) is the only semiword for which \( t \leq \varepsilon \) or \( \varepsilon \leq t \).

b) \( \supseteq \): \( \{s\} \chi T = \chi(\{s\}T) \subseteq (\text{by proposition 1.3.22.c) } \chi(\{s\}T). \)

\( \subseteq \): Let \( u \in \chi(\{s\}T) \) be given. We shall prove \( u \in \{s\} \chi T \).

If \( u \in \chi(\{s\}T) \) implies \( \exists t, t' \in T. s t \leq u \leq s t' \). From proposition 1.3.12 and \( u \leq s t' \), we see that there exists \( v, v' \) such that \( u = vv', v \leq s, v' \leq t' \). Hence \( st \leq u \) means \( st \leq vv' \). Again by proposition 1.3.12 there must exists \( s_v \) and \( s_{v'} \) such that \( st = s_v s_{v'} \) and \( s_v \leq v, s_{v'} \leq v' \). Now \( s_v \leq v \leq s \) implies \( A_{s_v} = A_s \). Clearly \( A_{s_v} = A_s \), and \( st = s_v s_{v'} \) implies \( s = s_v, t = s_{v'} \). This again means \( s \leq v \leq s, t \leq v' \leq t' \). \( s \leq v \leq s \) gives \( v = s \), so \( u = sv' \) for a \( v \) that \( t \leq v' \leq t' \) or equivalently \( us = v' \) for a \( v' \in \chi(\{t, t'\}) \subseteq \chi T \), so \( u \in \{s\} \chi T \).

**Corollary 1.3.25** \( \chi a.T = a.\chi T \)

### 1.3.2 Prefix of

We are now going to introduce another partial order on SW which shall be the generalization to SW of the well-known prefix partial order on \( \Delta^* (\cong W) \). It will turn out that in general \( s \) is a prefix of \( st \). As for \( \Delta^* \) we have that \( s \) being a prefix of \( t \in W \) implies that there exists a \( t' \) such that \( st' = t \), but this is not in general true for arbitrary \( t \in SW \! \). 

**Definition 1.3.26** \( s \) is a **prefix** of the semiword \( t \) (written \( s \subseteq t \)) iff \( s \) is a subsemiword of \( t \) and \( DC_{ \leq s}(A_s) \subseteq A_s \) i.e., \( \forall a^i \in A_s(\subseteq A_t) \forall b^j \in A_t. b^j \leq_t a^i \Rightarrow b^j \in A_s. \) 

Notice that for a subsemiword \( s \) of \( t \) \( DC_{ \leq s}(A_s) \subseteq A_s \) iff \( UC_{ \leq s}(A_t \setminus A_s) \subseteq A_t \setminus A_s \). This makes the connection with the definition of the prefix-po in [Pra86] for pomsets clear. We adopt his abbreviation \( \pi(s) \) for \( DC_{ \subseteq}(s) \) – the \( \subseteq \)-downwards closure of \( s \).

**Example:** If \( s = a \frac{d}{b} \frac{e}{c} \) then:
\[
\begin{align*}
a \overset{b}{\rightarrow} c \subseteq s, \text{ but } t = \begin{array}{c}
\overset{a}{\rightarrow} c \\
\overset{e}{\rightarrow} \end{array} e \not\subseteq s
\end{align*}
\]

because \( e \in A_t, d \leq s e \) and \( d \not\in A_t \).

**Proposition 1.3.27** \( \sqsubseteq \) is a po on \( SW \).

**Proof** Antisymmetry: \( s \sqsubseteq t, t \sqsubseteq s \) implies \( A_s \subseteq A_t, A_t \subseteq A_s \). Hence \( A_s = A_t \). Then of course \( \leq_t|A_s^2 = \leq_t|A_t^2 = \leq_t \). Since \( s \sqsubseteq t \) implies \( \leq_s = \leq_t|A_s^2 \) we have \( \leq_s = \leq_t \) and therefore \( s = t \).

Reflexivity: \( s \) is a subsemiword of \( s \), and the rest is immediate.

Transitivity: Given \( s \sqsubseteq t \subseteq u \). \( s \sqsubseteq t \) implies \( s \) is a subsemiword of \( t \), so \( A_s = A_t|A_s^2 \subseteq A_t \) and \( \leq_s = \leq_t|A_s^2 \). Similar we see \( A_t \subseteq A_u \) and \( \leq_t = \leq_u|A_t^2 \) from \( t \subseteq u \).

Because \( A_s \subseteq A_t \subseteq A_u \) we have \( A_u|A_s = A_t|A_s = A_s \). We also have \( \leq_s = \leq_t|A_s^2 = (\leq_u|A_t^2)|A_s^2 = (\text{since } A_s \subseteq A_t) \leq_u|A_s^2 \). Hence \( s = u|A_s \), \( s \) is a semiword wherefore \( A_s \) fulfills \( SW_1 \), so by proposition 1.1.4 \( s \) is a subsemiword of \( u \).

In order to have \( s \sqsubseteq u \) it now remains to prove \( DC_{\leq_u}(A_s) \subseteq A_s \). Let \( b \in A_s, a \in A_u \) be given such that \( a \leq_u b \). As \( A_s \subseteq A_t \) we have \( b \in A_t \). Since \( DC_{\leq_u}(A_t) \subseteq A_t \) it follows that \( a \in A_t \), so \( (a, b) \in \leq_s|A_t^2 = \leq_t \). Hence \( a \leq_t b \). Because \( DC_{\leq_t}(A_s) \subseteq A_s \) it then also follows that \( a \in A_s \).

We now present the proposition promised at example on page 28 which gives a sufficient condition for \( st \prec u \).

**Proposition 1.3.28** If \( s \sqsubseteq u \) and \( t \) is the complement semiword of \( s \) in \( u \) then \( st \leq u \)

**Proof** By proposition 1.2.7 it is only necessary to prove

\[
\forall a \in A_s, b \in A_u \setminus A_s, b \leq_u a \Rightarrow b \leq_{st} a
\]

This follows directly from the fact that \( b \not\leq_u a \) for all \( a \in A_s, b \in A_u \setminus A_s \). Assume on the contrary \( \exists a \in A_s, b \in A_u \setminus A_s, b \leq_u a \). Then \( b \in DC_{\leq_u}(A_s) \) and \( b \not\in A_s \) which contradicts the definition of \( s \sqsubseteq u \).

From the proof it is seen that \( \{ s \mid s \sqsubseteq u \} \) exactly is the set, \( S \), of subsemiwords of \( u \) for which \( s \in S \) iff \( st \leq u \), where \( t \) is the complement semiword of \( s \) in \( u \). So in this way we have found another characterization of \( \sqsubseteq \) (this alternative characterization would not hold with a more general notion of subsemiwords). That \( s \sqsubseteq u \) or rather \( DC_{\leq_u}(A_s) \subseteq A_s \) is a necessary condition was illustrated in the example on page 28.

**Proposition 1.3.29**

\begin{align*}
a) \quad & a.(s \parallel t) \preceq a.s \parallel t \\
b) \quad & s(t \parallel u) \preceq st \parallel u \\
c) \quad & (s \parallel s')(t \parallel t') \preceq st \parallel s't'
\end{align*}
provided the expressions are defined.

c) can be visualized as follows: \[ \frac{s}{s'} \leq t \leq \frac{s}{s'} t' \setminus \frac{s}{s'} t' . \]

**Proof**
a) is a special case of b) which in turn is a special case of c).

b) corollary 1.2.6 and corollary 1.2.12 are easily seen to hold for prefixes too. I.e., \( s \sqsubseteq st \), \( t \) complement semiword of \( s \) in \( st \) etc. So \( s \parallel s' \sqsubseteq st \parallel s't' \) and \( t \parallel t' \) is the complement semiword of \( s \parallel s' \) in \( st \parallel s't' \). The result then follows from proposition 1.3.28 above. \( \square \)

**Proposition 1.3.30** Let \( s, t, u \in SW \). Then:

\[
a.u \preceq s \parallel t \iff \begin{cases} \exists s'. a.s' \preceq s, u \preceq s' \parallel t, a^1 \notin A_t \\
or \exists t'. a.t' \preceq t, u \preceq s \parallel t', a^1 \notin A_s \end{cases}
\]

**Proof**

if: We only look at the case \( \exists s'. a.s' \preceq s, u \preceq s' \parallel t, a^1 \notin A_t \) since the other is handled totally symmetric. By corollary 1.3.11 \( a.s' \preceq s \) implies \( a.s' \parallel t \preceq s \parallel t \) since \( a^1 \notin A_t \) and \( s', t \) are disjoint. By the same corollary we obtain \( a.u \preceq a.(s' \parallel t) \) from \( u \preceq s' \parallel t \). Using proposition 1.3.29 we see \( a.(s' \parallel t) \preceq a.s' \parallel t \), so collecting the facts we establish \( a.u \preceq a.(s' \parallel t') \preceq a.s' \parallel t \preceq s \parallel t \) from which the result follows by the transitivity of \( \preceq \).

only if: Consider \( A_\alpha \cap A_s \) and \( A_\alpha \cap A_t \). One of the intersections must be empty - otherwise \( s \) and \( t \) would not be disjoint which is assumed for \( s \parallel t \) to make sense. W.l.o.g. assume the latter is the case i.e., \( a^1 \notin A_t \).

Since \( a.u \preceq s \parallel t \) implies \( A_{a.u} = A_s \parallel t = A_s \cup A_t \) we get \( a^1 \in A_s \) from \( a^1 \notin A_t \) and \( a^1 \in A_{a.u} \). So \( \alpha \) is a subsemiword of \( s \) and furthermore \( \alpha \subseteq s \). Let \( \alpha \) be the complement semiword of \( \alpha \) in \( s \). By proposition 1.3.28 \( a.s' = \alpha s' \preceq s \).

To see \( u \preceq s' \parallel t \) we prove \( A_u = A_s \parallel t \cup A_t \) and \( \alpha s' \parallel t \subseteq u \).

\( A_u = A_s \parallel t \cup A_t \): By corollary 1.2.6 \( u \) is the complement semiword of \( \alpha \) in \( au = a.u \), so by definition of complement semiword \( A_u = \{ b^i \psi(a.b) \mid b^i \in A_{a.u} \setminus \alpha \} \) (since \( a.u \preceq s \parallel t \)) \( \{ b^i \psi(a.b) \mid b^i \in A_{s' \parallel t} \setminus \alpha \} = \) (since \( a \notin \alpha \)) \( \{ b^i \psi(a.b) \mid b^i \in A_{s' \parallel t} \setminus \alpha \} \) (because \( a^1 \notin A_t \) \( b^i \in \alpha \cup A_t \)) \( \alpha \). Clearly only the following two cases can come into consideration.

\( b^i, c^j \in A_s \): Then \( b^i \preceq s' \parallel t \) \( c^j \) implies \( b^i \preceq s' \parallel t \) \( c^j \). Now \( b^i \preceq s' \parallel t \) \( c^j \) \( \Rightarrow \) (again by definition of \( s' \) being the complement of \( \alpha \) in \( s \)) \( b^i \psi(a.b) \preceq s \psi(a.b) \preceq c^j \). Now \( a.u \preceq s \parallel t \) \( b^i \psi(a.b) \preceq s \parallel t \) \( c^j \). Clearly only the following two cases can come into consideration.

\( b^i, c^j \in A_t \): Then \( b^i \preceq s' \parallel t \) \( c^j \) implies \( b^i \preceq s' \parallel t \) \( c^j \). Using \( a.u \preceq s \parallel t \) we then see \( b^i \preceq a.u \preceq c^j \). As earlier \( a \notin \alpha \) \( b^i, c^j \in A_t \Rightarrow \psi(a.b) = 0 = \psi(a.c) \), so from \( b^i \preceq a_u \preceq c^j \) and the definition of concatenation we get \( b^i \psi(a.b) \preceq c^j \).
This proposition can be specialized to $W$.

**Proposition 1.3.31** Let $s, t, u \in W$. Then

\[
a.u \leq s \parallel t \iff \begin{cases} 
\exists s' \in W. a.s' = s, u \preceq s' \parallel t, a^1 \notin A_t \\
\text{or} \\
\exists t' \in W. a.t' = t, u \preceq t', a^1 \notin A_s
\end{cases}
\]

**Proof**

*if:* Immediate from the previous proposition since $a.s' = s$ implies $a.s' \preceq s$, so by the previous proposition $a.u \preceq s \parallel t$.

*only if:* By the same proposition $a.u \preceq s \parallel t$ gives $\exists s'. a.s' \preceq s, u \preceq s' \parallel t, a^1 \notin A_t$ or $\exists t'. a.t' \preceq t, u \preceq t', a^1 \notin A_s$. The result then follows since $s, t \in W$ and $a.s' \preceq s, a.t' \preceq t$ implies $s', t' \in W$.

**Proposition 1.3.32**

\[
\begin{align*}
\text{a)} & \quad s \sqsubseteq t \iff us \sqsubseteq ut & \text{b)} & \quad s \sqsubseteq t \implies s \sqsubseteq tu \\
\text{c)} & \quad s \sqsubseteq t \iff s \parallel u \sqsubseteq t \parallel u
\end{align*}
\]

**Proof**

*a)* Each implication is proven separately.

$s \sqsubseteq t \implies us \sqsubseteq ut$: Given $s \sqsubseteq t$. We shall prove that $us$ is a subsemiword of $ut$ and that the $\leq_{ut}$-downwards closure of $A_{us}$ is contained in $A_{us}$.

Since $s \sqsubseteq t$ implies $s = t|_{A_u}$ it follows that $us$ is a subsemiword of $ut$ if we can prove that in general:

\[
(ut)|_{A_{us}} = u(t|_{A_u}) \quad (= us)
\]

At first observe that since

\[
\begin{align*}
A_{ut} &= A_u \uplus \{a^i \psi(u, a) \mid a^i \in A_t\} \quad \text{and} \\
A_{us} &= A_u \uplus \{a^i \psi(u, a) \mid a^i \in A_s\}
\end{align*}
\]

we have:

\[
\begin{align*}
A_{ut}|_{A_{us}} &= A_u \uplus \{a^i \psi(u, a) \mid a^i \in A_t\}|_{(a^i \psi(u, a)|a^i \in A_s)} \\
&= A_u \uplus \{a^i \psi(u, a) \mid a^i \in A_t|_{A_s}\} \\
&= A_u \uplus \{a^i \psi(u, a) \mid a^i \in A_{t|A_s}\}
\end{align*}
\]

It is now evident that $\leq_{ut}|_{A_{us}} = \leq_{u(t|_{A_u})}$ by looking at the definition of concatenation thereby establishing (1.5).

It remains to prove $DC_{ut}(A_{us}) \subseteq A_{us}$. So let $a^i \in A_{us}$ and $b^j \in A_{ut}$ with $b^j \preceq_t a^i$ be given. If $b^j \in A_u$ then clearly $b^j \in A_{us}$. So assume $b^j \notin A_u$, that is $\psi(u, b) < j$. Then
\[ b^i \leq_{ut} a^i \] implies \[ b^i - \psi(u, b) \leq_t a^i - \psi(u, a) \]. From this and \( s \subseteq t \) it follows that \( b^i - \psi(u, b) \in A_s \).

By definition of concatenation then \( b^i = b^{(i - \psi(u, b)) + \psi(u, b)} \in A_{us} \).

\[ \text{us} \subseteq \text{ut} \Rightarrow s \subseteq t: \text{us} \subseteq \text{ut} \text{ implies } \text{us} = \text{ut}|_A \text{ and from (1.5) } (\text{ut})|_{A_{us}} = u(t|_A) \text{ so we can conclude } s = t|_A \text{ a subsemiword of } t. \]

Now let \( a^i \in A_i, b^i \in A_i \) be given such that \( b^i \leq_t a^i \). Define \( k := i + \psi(u, a) \) and \( l := j + \psi(u, b) \). Since \( a^i \in A_i, b^i \in A_i \Rightarrow i, j \geq 1 \) we have \( \psi(u, b) < l \) and \( \psi(u, a) < k \).

Clearly \( b^i \leq_t a^i \) is the same as \( b^{i - \psi(u, b)} \leq_t a^{i - \psi(u, a)} \) so by the definition of concatenation we have \( b^i \leq_{ut} a^k \).

Since \( a^i \in A_i \Rightarrow a^k \in A_{us} \) we have from \( \text{us} \subseteq \text{ut} \) and \( b^i \leq_{ut} a^k \) that \( b^i \in A_{us} \). Since \( \psi(u, b) < l \) it follows that \( b^{i - \psi(u, u)} \in A_s \) which by the definition of \( l \) means \( b^i \in A_s \).

b) \( s \subseteq t \Rightarrow s \subseteq tu \): Given \( s \subseteq t \). We shall prove that \( s \) is subsemiword of \( tu \) and \( DC_{\leq_{tu}}(A_s) \subseteq A_s \).

Now \( s \subseteq t \) implies \( s = t|_A \), which again implies \( A_s \subseteq A_t \). Hence \( A_s \subseteq A_t \cup A_{tu} \setminus A_t = A_{tu} \) and since \( s \) is a semiword, \( A_s \) fulfills SW1, wherefore \( s \) is a subsemiword of \( tu \) (by proposition 1.1.4).

To prove \( DC_{\leq_{tu}}(A_s) \subseteq A_s \) let \( a \in A_s \) and \( b \in A_{tu} \) be given such that \( b \leq_{tu} a \).

\( b \) cannot be in \( A_{tu} \setminus A_t \). If it was \( a \in A_t \subseteq A_t \) would imply \( a \leq_{tu} b \) as noticed by the definition of concatenation. By the antisymmetry of \( \leq_{tu} \) and we would then get \( a = b \) — a contradiction to \( a, b \) belonging to disjoint sets.

So \( b \in A_t \). By definition \( b \leq_{tu} a \) and \( s \in A_s \subseteq A_t \) then implies \( b \leq_t a, b \in A_t \) then follows from \( s \subseteq t \).

c) At first notice \( s \subseteq t \) implies \( A_s \subseteq A_t \), wherefore \( t \) disjoint from \( u \) implies \( s \) disjoint from \( u \) — so well-defined under the proviso. The rest follows directly from \( A_{s|ut} = A_s \cup A_t \). \( \square \)

**Corollary 1.3.33** If \( s_i \subseteq t_i \text{ for } i \in 2 \) then \( s_1 \| s_2 \subseteq t_1 \| t_2 \) provided \( t_1 \) and \( t_2 \) are disjoint.

**Proposition 1.3.34**

\[
\begin{align*}
\text{a) } u \subseteq st & \Rightarrow \begin{cases} u \subseteq s \\
 \exists t' \subseteq t. u = st'
\end{cases} \\
\text{b) } u \subseteq s \parallel t & \Rightarrow \exists s' \subseteq s, t' \subseteq t. u = s' \parallel t'
\end{align*}
\]

**Proof**

\[ a) \ u \subseteq st \Rightarrow u \subseteq s \text{ or } \exists t' \subseteq t. u = st': \]

Given \( u \subseteq st \). Then \( u = st|_{A_u} \) and \( DC_{\leq_{ut}}(A_u) \subseteq A_u \). Since \( u = st|_{A_u} \) implies \( A_u \subseteq A_{st} = A_u \cup (A_{st} \setminus A_u) \) we have two principal cases:

\( A_u \subseteq A_s \): Claim: then \( u \subseteq s \). Clearly \( st|_{A_u} = s|_{A_u} \) wherefore \( u \) is a subsemiword of \( s \).

To see \( DC_{\leq_s}(A_u) \subseteq A_u \) let \( a^i \in A_u \) and \( b^i \in A_u \) be given such that \( b^i \leq_s a^i \). Since \( a^i \in A_u \subseteq A_s, b^i \in A_s \) implies \( i \leq \psi(s, a), j \leq \psi(s, b) \) we have from \( b^i \leq_s a^i \) that \( b^i \leq_{st} a^i \).

Then \( a^i \in A_u \) and from \( DC_{\leq_{st}}(A_u) \subseteq A_u \) it follows that \( b^i \in A_u \) and we are done for this case.

\( A_u \nsubseteq A_s \) but \( A_u \nsubseteq A_{st} \setminus A_s \): At first we prove: \( A_u \nsubseteq A_u(\nsubseteq A_{st}) \). Let \( a \in A_u \) be given. \( A_u \cap (A_{st} \setminus A_u) \) cannot be empty since this would imply \( A_u \subseteq A_s \) which we assume is not the case, so there exists a \( b \in A_u \cap (A_{st} \setminus A_s) \). As noticed at the definition of concatenation
Because $A_s \subseteq A_u$ and $s, u$ are semiwords, it follows that $s$ is a subsemiword of $u$, and so $s = u|_{A_s}$. Then we can define $t'$ to be the complement semiword of $s$ w.r.t. $u$. Recall that this means:

$$A_{t'} := \{a^{i-\psi(s,a)} \mid a^i \in A_u \setminus A_s\}$$

$$\forall a^i, b^j \in A_{t'} \colon a^i \leq_{t'} b^j \iff a^{i+\psi(s,a)} \leq_u (A_u \setminus A_s)^2 b^{j+\psi(s,b)}$$

Notice $a^k \in A_{t'}$ iff $a^{k+\psi(s,a)} \in A_u \setminus A_s$. By the definition of concatenation it follows that $A_{u'} = A_u \cup \{a^{k+\psi(s,a)} \mid a^k \in A_{t'}\} = A_s \cup \{a^{k+\psi(s,a)} \mid a^k \in A_u \setminus A_s\} = A_u \cup \{a^i \mid a^i \in A_u \setminus A_s\} = A_u \cup (A_u \setminus A_s) = A_u$—the last equation is a consequence of $s$ being a subsemiword of $u$.

We now want to prove $\leq_u = \leq_{u'}$, i.e., $\forall a^i, b^j \in A_u (= A_{u'})$. $a^i \leq_u b^j \iff a^i \leq_{u'} b^j$.

$: \Rightarrow$ Given $a^i, b^j \in A_u$ such that $a^i \leq_u b^j$.

Since $s$ is a subsemiword of $u$ we can compare $i$ and $j$ with $\psi(s,a)$ and $\psi(s,b)$.

- $i \leq \psi(s,a), j \leq \psi(s,b)$: Then $a^i, b^j \in A_s$. Hence $a^i \leq_{u\setminus A_s} b^j$ and by definition of $s$ we have $a^i \leq_s b^j$. From the definition of concatenation we see that this implies $a^i \leq_{u'} b^j$.

- $i \leq \psi(s,a), \psi(s,b) < j$: Follows directly by the definition of concatenation.

- $\psi(s,a) < i, \psi(s,b) < j$: Then $a^i, b^j \notin A_s$ and so $a^i, b^j \in A_u \setminus A_s$. From $a^i \leq_{u\setminus A_s} b^j$ we then conclude $a^i \leq_{u\setminus (A_u \setminus A_s)^2} b^j$ which by the definition of $\leq_{u'}$ implies $a^{i-\psi(s,a)} \leq_{u'} b^{j-\psi(s,b)}$.

By the definition of concatenation we now get $a^i \leq_{u'} b^j$.

- $\psi(s,a) < i, \psi(s,b) < j$: Then $a^i \in A_u \setminus A_s$ and $b^j \in A_s$.

Now $u \sqsubseteq st$ implies $A_u \subseteq A_u \cup (A_u \setminus A_s)$ and since $A_s \subseteq A_u$ it follows that $a^i \in A_u \setminus A_s$ implies $a^i \in A_{st} \setminus A_s$. From $u \sqsubseteq st$ we also see $a^i \leq_{u\setminus A_s} b^j$ only if $a^i \leq_{st} b^j$. On the other hand we noticed at the definition of concatenation that $b^j \in A_s, a^i \in A_{st} \setminus A_s$ implies $b^j \leq_{st} a^i$. Since $\leq_{st}$ is antisymmetric we must have $a^i = b^j$—a contradiction to $a^i \in A_u \setminus A_s$ and $b^j \in A_s$, so we can rule out this case.

$: \Leftarrow$ Given $a^i, b^j \in A_{u'} (= A_u)$ such that $a^i \leq_{u'} b^j$.

By the definition of concatenation one of the following cases must hold.

- $i \leq \psi(s,a), j \leq \psi(s,b), a^i \leq b^j$: Since $s$ is a subsemiword of $u$ this implies $a^i \leq_u b^j$.

- $i \leq \psi(s,a), \psi(s,b) < j$: Then $a^i \in A_s \subseteq A_u, b^j \in A_u \setminus A_s$. Similar as above we see $b^j \in A_u \setminus A_s$ implies $b^j \in A_{st} \setminus A_s$, wherefore $a^i \leq_{st} b^j$. Since $a^i, b^j \in A_u \subseteq A_{st}$ we then also have $a^i \leq_{st\setminus A_s^2} b^j$. Because $u \sqsubseteq st$ implies $\leq_u = \leq_{st\setminus A_s^2}$ this means $a^i \leq_u b^j$.

- $\psi(s,a) < i, \psi(s,b) < j, a^{i-\psi(s,a)} \leq_{st} b^{j-\psi(s,b)}$: By definition of $t'$ this implies:

  $$a^{(i-\psi(s,a))+\psi(s,a)} \leq_{u\setminus (A_u \setminus A_s)^2} b^{(j-\psi(s,b))+\psi(s,b)}$$

  $$a^i \leq_{u\setminus (A_u \setminus A_s)^2} b^j$$

  Obviously $\leq_{u\setminus (A_u \setminus A_s)^2} \leq_u$ only if $a^i \leq_u b^j$, so we have now proved: $u = st'$ for the defined $t'$. Then $u \sqsubseteq st$ reads $st' \sqsubseteq st$ which by the last proposition implies $t' \sqsubseteq t$. 

42
b) \( u \subseteq s \parallel t \Rightarrow \exists s' \subseteq s, t' \subseteq t. \ u = s' \parallel t' \):
\( u \subseteq s \parallel t \) implies \( u = (s \parallel t)|_{A_u} \) which—because \( s \) and \( t \) are disjoint—equals \( s|_{A_u} \parallel t|_{A_u} \). Let \( s' = s|_{A_u}, t' = t|_{A_u} \). We then have \( u = s' \parallel t' \) and \( s' \parallel t' \subseteq s \parallel t \). Since \( A_{s'} \subseteq A_s, A_{t'} \subseteq A_t \) and \( s, t \) are disjoint, it is trivial to see that \( s' \subseteq s \) and \( t' \subseteq t \). \( \square \)

**Proposition 1.3.35**

\[
\begin{align*}
a) & \ \varepsilon, s \in \pi(s) & b) & \ \pi(\varepsilon) = \{\varepsilon\}, \forall a \in \Delta. \ \pi(\varepsilon) = \{\varepsilon, a\} \\
c) & \ \pi(st) = \pi(s) \cup \{s\} \pi(t) & d) & \ \pi(s \parallel t) = \pi(s) \parallel \pi(t)
\end{align*}
\]

**Proof**

a) Clearly, \( \varepsilon, s \) are subsemiwords of \( s \) and \( DC_{\leq,}(A_\varepsilon) = DC_{\leq,}(\emptyset) = \emptyset = A_\varepsilon \) so as \( DC_{\leq,}(A_s) \subseteq A_s \).

b) \( \pi(\varepsilon) = \{\varepsilon\} \) is evident, and from a) we have \( \{\varepsilon, a\} \subseteq \pi(\varepsilon) \). Since \( \varepsilon, a \) are the only possible subsemiwords of \( \varepsilon \) it follows that \( \pi(\varepsilon) \subseteq \{\varepsilon, a\} \).

c) \( \subseteq \) follows from a) of the last proposition. \( \{s\} \pi(t) \subseteq \pi(st) \) follows from proposition 1.3.32.a) and \( \pi(s) \subseteq \pi(st) \) from b) of the same.

d) \( \subseteq \) follows from b) of the last proposition and \( \supseteq \) from the last corollary. \( \square \)

From c) and \( \pi(\varepsilon) = \{\varepsilon\} \) we immediately get the corollary:

**Corollary 1.3.36** \( \pi(a.s) = a.\pi(s) \cup \{\varepsilon\} \)

The next lemma concerned with pos will be used intensively in the proof of proposition 1.3.38 below.

**Lemma 1.3.37** Let \( B \) be a subset of \( A \) and \( \leq \) a po on \( A \) such that \( DC_{\leq}(B) \subseteq B \). Furthermore let \( Q \) be a relation such that *either*

\[
\begin{align*}
a) & \ Q \text{ is antisymmetric, transitive and } \leq|_{B^2} \subseteq Q \subseteq B^2 \\
& \text{ or } \\
b) & \ Q \subseteq A \times (A \setminus B)
\end{align*}
\]

Define \( R \) to be \( \leq \cup Q \). Then \( R^+ \) is a po on \( A \) and \( DC_{R^+}(B) \subseteq B \).

**Proof** Notice that no matter whether a) or b) holds \( Q \) is not defined on \( (A \setminus B) \times A \).

Then we can prove:

\[
(1.6) \quad b \in A \setminus B, a \in B \Rightarrow \lnot(b R^a a)
\]

by induction on \( n \).

\( n = 1 \): Assume on the contrary \( b \in A \setminus B, a \in B \) and \( b R a \). Since \( Q \) is not defined on \( (A \setminus B) \times A \) we see that \( b R a \) implies \( b \leq a \)—a contradiction to \( DC_{\leq}(B) \subseteq B \).

\( n > 1 \): Again suppose on the contrary \( b \in A \setminus B, a \in B \) and \( b R^a a \). Then \( b R^{a-1} c \) and \( c R a \) for some \( c \in A \). Two cases:
$c \in B$: By hypothesis of induction $\neg(b R^{n-1} c)$—a contradiction to $b R^{n-1} c$.

$c \notin B$: Then $c \in A \setminus B$ and by hypothesis $\neg(c R a)$—a contradiction.

$DC_{R^+}(B) \subseteq B$ now directly follows from (1.6).

Since $\leq$ is reflexive on $A^2$ and $\leq \subseteq R^+$ this must be the case for $R^+$ too. By definition $R^+$ is transitive. We look at a) and b) separately when proving the antisymmetry of $R^+$.

a) Assume $\leq_{|B^2} \subseteq Q \subseteq B^2$ and $Q$ transitive, antisymmetric. At first we prove:

\[
(1.7) \quad a, b \in B, a R^n b \Rightarrow a Q b
\]

$n = 1$: Follows directly from $\leq_{|B^2} \subseteq Q$.

$n > 1$: Then $a R^{n-1} c, c R b$ for some $c \in A$. We must have $c \in B$. Otherwise we would have $c \in A \setminus B$ and from (1.6): $\neg(c R b)$—a contradiction. So $c \in B$. Then by hypothesis $a Q c, c Q b$ and from the transitivity of $Q$: $a Q b$.

Next we prove:

\[
(1.8) \quad a, b \in A \setminus B, a R^n b \Rightarrow a \leq b
\]

$n = 1$: We must have $a \leq b$ since $Q$ is not defined on $(A \setminus B)^2$.

$n > 1$: Then $a R^{n-1} c, c R b$ for some $c \in A$. By (1.6) we cannot have $c \in B$, so $c \in A \setminus B$. By hypothesis of induction and the transitivity of $\leq$ we get $a \leq b$.

From (1.6) – (1.8) and the antisymmetry of $Q$ and $\leq$ we get:

\[
(1.9) \quad \forall a, b \in A. a R^n b, b R^m a \Rightarrow a = b
\]

and thereby also the antisymmetry of $R^+$.

b) Assume $Q \subseteq A \times (A \setminus B)$. At first we prove:

\[
(1.10) \quad a, b \in B, a R^n b \Rightarrow a \leq b
\]

$n = 1$: Follows directly since $Q$ is not defined on $B^2$.

$n > 1$: Then $a R^{n-1} c, c R b$. By (1.6) we must have $c \in B$. (1.10) then follows by hypothesis and transitivity of $\leq$.

Similar we prove:

\[
(1.11) \quad a, b \in A \setminus B, a R^n b \Rightarrow a \leq b
\]

From (1.6), (1.10), (1.11) and the antisymmetry of $\leq$ we get (1.9).

Notice that this lemma (with the b) proviso) also could have been used to prove $R^+$ in lemma 1.3.6 to be a po on $A_s$ by letting $B = DC_{\leq_s}(\{a\})$.

The next proposition says that $\pi$ distributes over $\delta$ and $\lambda$ and “partly” over $\nu$.

**Proposition 1.3.38**
a) $\pi\delta(s) = \delta\pi(s)$

b) $\pi\lambda(s) = \lambda\pi(s)$

c) $\pi\upsilon(s) \supseteq \upsilon\pi(s)$

**Proof**

a) $\pi\delta(s) \subseteq \delta\pi(s)$: Let $t \in \pi\delta(s)$ be given. Then there exists a $t' \in \delta(s)$ such that $t \subseteq t'$. $t' \in \delta(s)$ implies $t' \subseteq s$. Consider $u$ defined by $u := s|_{A_t}$. We shall prove that $u$ is a semiword and $t \subseteq u \subseteq s$.

Since $t \subseteq t' \subseteq s$ implies $A_t \subseteq A_{t'} = A_s$, $u$ must be a subsemiword of $s$ with $A_u = A_t$.

In order to prove $u \subseteq s$ we then just need to prove $DC_{\leq s}(A_u) \subseteq A_u$ or what is the same $DC_{\leq s}(A_t) \subseteq A_t$. Let $a \in A_t$ and $b \in A_s$ be given such that $b \leq s_a$. We shall prove $b \in A_t$.

Since $t' \subseteq s$ implies $\leq_s \subseteq \leq_{t'}$ we have $b \leq_{t'} a$. Because $t \subseteq t'$ implies $DC_{\leq_{t'}}(A_t) \subseteq A_t$ we get $b \in A_t$.

Next we prove $t \subseteq u$. Since $A_t = A_u$ we just need to prove $\leq_u \subseteq \leq_t$. Because $t' \subseteq s$ implies $\leq_s \subseteq \leq_{t'}$ we get from $A_t \subseteq A_{t'} = A_s$ that $\leq_u = \leq_s|_{A_t^2} \subseteq \leq_{t'}|_{A_t^2}$. Since $t \subseteq t'$ gives $\leq_t = \leq_{t'}|_{A_t^2}$ we are finished.

\[\delta\pi(s) \subseteq \pi\delta(s):\] Let $t \in \delta\pi(s)$ be given. I.e., there exists a $t'$ such that $t \subseteq t' \subseteq s$. We shall find a semiword $u$ such that $t \subseteq u \subseteq s$.

Define $u$ by $A_u := A_s$ and $\leq_u := R^+$, where $R = (\leq_s \cup \leq_t)$.

We first want to examine if $u$ is a semiword. Since $A_u = A_s$ it fulfills SW1, and because $\leq_s \subseteq SW2$, it follows that $\leq_u$ does so too provided $\leq_u$ is a pos. Now $t' \subseteq s$ implies $\leq_{t'} = \leq_s|_{A_t^2}$ and $t \subseteq t'$ implies $A_t = A_{t'}$, $\leq_{t'} \subseteq \leq_t$, so $\leq_s|_{A_t^2} \subseteq \leq_t$ and we see that a) of lemma 1.3.37 is satisfied. Also $DC_{\leq_s}(A_t) \subseteq A_t$ because $DC_{\leq_s}(A_{t'}) \subseteq A_{t'}$ and $A_t = A_{t'}$.

From the lemma we can then conclude $u$ is a semiword and $DC_{\leq_u}(A_t) \subseteq A_t$.

Now clearly $A_u = A_s$ and $\leq_s \subseteq \leq_u$, so $u \subseteq s$.

To see $t \subseteq u$ notice that $A_t = A_{t'} = A_s|_{A_{t'}} = A_u|_{A_{t'}} = A_u|_{A_t}$ and $\leq_t \subseteq \leq_u|_{A_t^2}$ by definition of $\leq_u$.

$\leq_u|_{A_t^2} \subseteq \leq_t$ follows from (1.7) of lemma 1.3.37. So $t$ is a subsemiword of $u$ and we already know $DC_{\leq_u}(A_t) \subseteq A_t$.

b) $\pi\lambda(s) \subseteq \lambda\pi(s)$: Follows exactly as $\subseteq$ of a), just notice that for the given $t$ no $t'$ exists such that $t' < t$.

$\lambda\pi(s) \subseteq \pi\lambda(s)$: Here we cannot take over the corresponding proof of a) directly, since the semiword $u$ constructed there not necessarily belongs to $\lambda(s)$. For the $u$ of a) we know that $t \subseteq u \subseteq s$. The idea is now to choose a $u' \in \lambda(u) \subseteq \lambda(s)$ and prove $t \subseteq u' \subseteq s$.

But we have to be careful in choosing $u'$—not every $u'$ of $\lambda(u)$ will do. On the way to find $u'$ we define a $v \preceq u$ which will ensure that every $u' \in \lambda(v)$ will have $t$ as prefix. Let $Q = \{(a,b) \mid a \in A_t, b \in A_u \setminus A_t\}$, $R = \leq_u \cup Q$, and define $v$ by $A_v := A_u, \leq_v := R^+$. In this way every smoothing of $v$ will have $t$ as prefix.

Of course we shall at first prove that $v$ is indeed a semiword.

Notice that $\leq_u$ and $Q$ are contained in $R \subseteq \leq_v$. Clearly SW1 and SW2 holds for $v$ because $u \in SW$ and $A_v = A_u, \leq_u \subseteq \leq_v$. Since $t \subseteq u$ we have $DC_{\leq_u}(A_t) \subseteq A_t$ and by construction $Q$ satisfies b) of lemma 1.3.37 (with $A = A_u, \leq = \leq_u$ and $B = A_t$). Hence we conclude that $v$ is a semiword.
Clearly $v \leq u \leq s$. By proposition 1.3.5 $\lambda(v) \neq \emptyset$, so chose a $u' \in \lambda(v)$. Then $u' \leq s$.

To see $t \subseteq u'$ notice that $A_{u'} = A_u$, hence $A_t = A_u|A_t$. Clearly $\leq u|A_t^2 = \leq u'|A_t^2$ since no more refinements of $\leq u|A_t^2$ were possible, because $\leq t = \leq u|A_t^2$ and $t \in W$. That means $\leq t = \leq u'|A_t^2$ wherefore $t$ is a subsemiword of $u'$, so we just need to prove $DC_{\leq u'}(A_t) \subseteq A_t$. Suppose this is not the case. Then there is some $s \in A_u \setminus A_t$, $b \in A_t$ such that $a \leq_w b$.

Now $R^+ \subseteq \leq u$ implies by lemma 1.3.9 that every $v'$ of $\lambda(v)$ has $R^+ \subseteq \leq u'$. Especially we have $R^+ \subseteq \leq u'$. Hence also $Q \subseteq \leq u'$. Now $a \in A_t, b \in A_u \setminus A_t$ implies a $Q, b$ wherefore $a \leq_w b$. By antisymmetry of $\leq u$: $a = b$—which contradicts that $a, b$ belongs to disjoint sets.

c) $v\pi(s) \subseteq \pi v(s)$: Let $t \in v\pi(s)$ be given. I. e., there exists $t'$ such that $t' \subseteq s$ and $t' \leq t$.

The problem is now to find a $u$ such that $s \leq u$ and $t \subseteq u$. The idea is to define $u$ such that it is the least extension of $t$ to the elements of $A_n(\supseteq A_t)$ such that $u$ is a semiword. Define $u$ by $A_u := A_s$ and $\leq u := R^+$, where $R = \subseteq \cup \subseteq \subseteq$ and $\subseteq = \{(d, a) \in A_s^2 | (d, b) = (c', c'' \text{ for some } c \in \Delta \text{ and } \subseteq = \{d, c'' \})\}$.

At first we want to show that $u$ is a semiword.

Since $A_s = A_s$ and $s \in SW$ we have $A_s$ fulfills $SW1$. Notice that $\subseteq \leq s$ is the least po on $A_n$ which satisfies $SW2$. Because $\subseteq \subseteq R^+$ we see that $R^+$ fulfills $SW2$ if $R^+$ is a po. Since $\subseteq s$ satisfies $SW2$ and $\subseteq$ is the least po that does so we have $\subseteq \subseteq s$. Then we see $DC_{\leq s}(A_t) \subseteq A_t$ because $DC_{\leq s}(A_t') \subseteq A_t, A_t = A_t$. Also $\subseteq |A_t^2$ is the least po on $A_t$ which satisfies $SW2$, so $\subseteq |A_t^2 \subseteq \subseteq$ and a) of lemma 1.3.37 is satisfied (with $A = A_s$, $Q = \subseteq \subseteq \subseteq$ and $B = A_t$). Hence we conclude that $u$ is a semiword and $DC_{\leq u}(A_t) \subseteq A_t$.

Since $\subseteq \subseteq s$ and $\subseteq s \subseteq \subseteq \subseteq \subseteq = \subseteq |A_t^2 \subseteq \subseteq$ it follows that $\subseteq u \subseteq \subseteq s$ wherefore $s \leq u$.

To see $t \subseteq u$, at first notice $A_t = A_{u'} = A_s|A_t' = A_u|A_t' = A_u|A_t$. $\subseteq \subseteq \subseteq \subseteq \subseteq \subseteq \subseteq$ by definition of $\subseteq u$. And from (1.7) of lemma 1.3.37 $\subseteq u |A_t^2 \subseteq \subseteq$ follows. So $t$ is a subsemiword of $u$ and we already know $DC_{\leq u}(A_t) \subseteq A_t$.

It is easy to see that $\pi v(s) \not\subseteq v\pi(s)$. Take for instance the semiword $s = a\overrightarrow{c}b$. Then for $t' = a$ we have $t' \in v(s)$ and $t = b \in \pi(t')$. Hence $t \in v\pi(s)$. If $t$ should belong to $v\pi(s)$ there should be an $s'$ such that $t' \subseteq t$ and $s' \subseteq s$. Now $s' \subseteq t$ implies $A_{s'} = \{b\}$. But there is no prefix $s'$ of $s$ with $A_{s'} = \{b\}$ because $a^1 \in DC_{\leq s'}(b^1)$ and therefore also should be included in $A_{s'}$.

By $\chi$ it even gets worse. In general we have neither $\pi \chi S \subseteq \chi \pi S$ nor $\chi \pi S \subseteq \pi \chi S$. The latter can be seen by the example $S = \left\{ a\overrightarrow{c}b\overrightarrow{d}c, a\overrightarrow{b}, a\overrightarrow{c}b \right\}$, $a \leq b \in \pi \chi S$ and the former by $S = \left\{ a\overrightarrow{c}b\overrightarrow{c}c, a\overrightarrow{b}c \right\}$, $a \rightarrow c \in \pi \chi S$.

In the next propositions the interrelation between the connected components of semiwords which are in $\subseteq$.

**Proposition 1.3.39** $s \leq t \Rightarrow \forall u \in \gamma(s) \exists D \subseteq \gamma(t). u \leq || D$.

**Proof** Induction on the number of connected components of $s$.  

46
In general $|\gamma(s)| = 1$: Then $s = \varepsilon$ and therefore also $t = \varepsilon$. Chose $D = \{\varepsilon\} = \gamma(t)$.

$|\gamma(s)| > 1$: Then there is an $s' \in \gamma(s)$, $s' \neq \varepsilon$, and we can write $s = s' \parallel s''$, where $s'' = |\gamma(s) \setminus \{s'\}$. By proposition 1.3.17 we have $s = s' \parallel s'' \leq t$ implies $\exists t' \leq t''$, $s'' \leq t''$, $t = t'' \parallel t''$. From proposition 1.2.13.a we see $\gamma(t) = \gamma(t') \cup \gamma(t'')$. Hence for $s'$ we can chose $D = \gamma(t') \subseteq \gamma(t)$ and get $s' \leq t' = |\gamma(t')| = |D|$. This settles the case if $u = s'$.

So it remains to prove $\forall u \in \gamma(s) \setminus \{s'\} \exists D \subseteq \gamma(t)$. $u \leq |D|$. $s' \in \gamma(s)$ implies $\gamma(s') = \{\varepsilon, s'\}$, so from $s' \neq \varepsilon$ and proposition 1.2.13.a we get $\gamma(s) \setminus \{s'\} = \gamma(s'') \cup \{\varepsilon\}$, $\gamma(s'') \cup \{\varepsilon\} = (\gamma(s'') \cup \{\varepsilon\}) \setminus \{\varepsilon\}$, $s'' \leq t''$, $t''$ is a set of semiwords we let $\gamma(s')$. From proposition 1.2.13.a) we see

$\gamma(t) = \gamma(t') \cup \gamma(t'')$. Hence for $s'$ we can chose $D = \gamma(t') \subseteq \gamma(t)$ and get $s' \leq t' = |\gamma(t')| = |D|$. This settles the case if $u = s'$. Now proposition 1.2.13 gives $|\gamma(s') \setminus \{\varepsilon\}|$. From proposition 1.2.13.a) we see $|\gamma(s') \setminus \{\varepsilon\}| = |\gamma(s') \setminus \{\varepsilon\}| + |\gamma(s'') \setminus \{\varepsilon\}|$. Since $t'' = t''$ only if $\gamma(t'') \subseteq \gamma(t)$ it follows that it is enough to prove $\forall u \in \gamma(s') \exists D \subseteq \gamma(t'')$. $u \leq |t''|$. We have $s'' \leq t''$, so we get the wanted directly by hypothesis of induction if we can prove $|\gamma(s'')| < |\gamma(s')|$. Now proposition 1.2.13 gives $|\gamma(s') \setminus \{\varepsilon\}| = |\gamma(s') \setminus \{\varepsilon\}| + |\gamma(s'') \setminus \{\varepsilon\}|$. So $|\gamma(s'') \setminus \{\varepsilon\}| = |\gamma(s') \setminus \{\varepsilon\}| - |\gamma(s') \setminus \{\varepsilon\}| = (\gamma(s') = \{\varepsilon, s'\}, s' \neq \varepsilon) |\gamma(s') \setminus \{\varepsilon\}| - 1$. Because in general $\varepsilon \in \gamma(v)$ for arbitrary $v$ we have $|\gamma(s'')| = |\gamma(s')| - 1 < |\gamma(s)|$. □

In general $\gamma(s) \neq \emptyset$ wherefore we also have $s \leq t \Rightarrow \exists u \in \gamma(s) \exists D \subseteq \gamma(t)$. $u \leq |D|$. If $D$ is a set of semiwords we let $A_D$ denote $\cup_{s \in D} A_s$ in the following proposition and it’s proof.

**Proposition 1.3.40** Given $s, t$ such that $A_s = A_t$ and for each $s' \in \gamma(s)$ a $D_{s'} \subseteq \gamma(t)$ with $A_{s'} = A_{D_{s'}}$. Then:

$$\gamma(t) = \bigcup_{s' \in \gamma(s)} D_{s'}$$

**Proof** Let $D$ denote $\cup_{s' \in \gamma(s)} D_{s'}$. Because each $D_{s'} \subseteq \gamma(t)$ we clearly have $D \subseteq \gamma(t)$.

To see $D \supseteq \gamma(t)$ assume on the contrary that there exists a $t' \in \gamma(t)$ such that $t' \notin D$. At first notice $t' \in \gamma(t)$ implies $A_{t'} \subseteq A_t$.

Next we prove $t' \neq \varepsilon$. Because $\varepsilon \in \gamma(s)$ we have a $D_{\varepsilon} \subseteq \gamma(t)$ with $A_{\varepsilon} = \emptyset = A_{D_{\varepsilon}}$. Since $u = \varepsilon$ is the only semiword with $A_u = \emptyset$ we must have $D_{\varepsilon} = \{\varepsilon\}$. Hence $\varepsilon \in D$ and from $t' \notin D$ we then see $t' \neq \varepsilon$.

Because $D \subseteq \gamma(t)$ and $\gamma(t)$ consists of disjoint semiwords $t' \in \gamma(t) \setminus D$ must imply $A_{t'} \cap A_{t'} = \emptyset$ for every $t'' \in D$. From $t \neq \varepsilon$ and thereby $A_{t'} \neq \emptyset$ we then conclude $A_{t'} \not\subseteq A_D$. But this implies $A_{t'} \not\subseteq A_D = \bigcup_{s' \in \gamma(s)} A_{s'} = A_s = A_t$ which contradicts $A_{t'} \subseteq A_t$. □

Because $s \leq t$ only if $A_s = A_t$ we have the following.

**Corollary 1.3.41** Given $s, t$ such that $s \leq t$ and for each $s' \in \gamma(s)$ a $D_{s'} \subseteq \gamma(t)$ with $s' \leq |D_{s'}|$. Then:

$$\gamma(t) = \bigcup_{s' \in \gamma(s)} D_{s'}$$

**Proposition 1.3.42** $a.s \leq t$, $|\gamma(t)| > 3$ implies $\exists u. a.s < u < t$
Proof Since \( t \) is finite and thereby \(|\gamma(t)|\) too we will by repeated use of proposition 1.3.30 find a \( t_1 \in \gamma(t) \) such that \( a.t'_{1} \leq t_{1}, s \preceq t'_{1} \) \(|(\gamma(t) \setminus \{t_1\})| \) for some \( t'_1 \). Because \( t_1 \in \gamma(t) \) and \(|\gamma(t)| > 3 \) we can write \( \|\gamma(t) \setminus \{t_1\}\| \) as \( t_2 \| t_3 \) for some \( t_2, t_3 \neq \varepsilon \). So we have \( a.s \preceq t_1 \| t_2 \| t_3 \), \( a.t'_{1} \preceq t_{1}, s \preceq t'_{1} \| t_2 \| t_3 \).

Now chose \( u = a.(t'_{1} \| t_2) \| t_3 \).

From proposition 1.3.29.a we have \( a.((t'_{1} \| t_2) \| t_3) \preceq a.(t'_{1} \| t_2) \| t_3 = u \). Since \( t_3 \neq \varepsilon \) we have \( \gamma(a.t'_{1} \| t_2) \| t_3) \neq \gamma(u) \), so \( a.(t'_{1} \| t_2) \| t_3 < u \). From \( s \preceq t'_{1} \| t_2 \| t_3 \) we see \( a.s \preceq a.(t'_{1} \| t_2) \| t_3) \).

Hence \( a.s < u \). Again from proposition 1.3.29 we get \( a.(t'_{1} \| t_2) \preceq a.t'_{1} \| t_2 \) and thereby \( u = a.(t'_{1} \| t_2) \| t_3 \preceq (a.t'_{1} \| t_2) \| t_3 \). As \( t_2, t_3 \neq \varepsilon \) we conclude \( u < a.t'_{1} \| t_2 \| t_3 \). Now \( a.t'_{1} \preceq t_1 \) implies \( a.t'_{1} \| t_2 \| t_3 \preceq t_1 \| t_2 \| t_3 = t \), so also \( u < t \).

The definition of the relation \( \prec \) for a po \((A, \preceq)\) is:

\[
\forall a, b \in A. \ a \prec b \iff a \prec b \text{ and } \exists c \in A. \ a \prec c \prec b.
\]

That is \( a \prec b \) means \( a \) is an immediate predecessor of \( b \) in the relation \( \prec \). The \( \prec \) might be empty some pos though \( \preceq \) is not, but for \( \preceq \) on SW we have \( \prec^+ \equiv \prec \text{ and } \prec^- \equiv \preceq \).

This is seen as follows. Let \( s \prec t \). This means \( A_s = A_t \) and \( s \in \delta(t) \), \( t \in \upsilon(s) \). So every semiword \( u \) of \( a \prec \)-path from \( s \prec t \) must be in \( \delta(t) \cap \upsilon(s) \). As noticed earlier \( \delta \) and \( \upsilon \) are in general finite, so all such path’s are finite as well wherefore there exists \( 0 \leq n \), and some \( u_i, i \in \mathbb{N} \) such that \( s \prec u_1 \prec u_2 \ldots \prec u_n \prec t \). Clearly then \( \prec^+ = \prec \text{ and } \prec^- = \preceq \).

The lately proved properties allow us to show a implication of \( s \prec t \).

**Proposition 1.3.43** \( s \prec t \) implies \( \exists s' \in \gamma(s) \setminus \{\varepsilon\} \exists D \subseteq \gamma(t). \gamma(s) \setminus \{s'\} = \gamma(t) \setminus D, s' \prec \| D \).

**Proof** Clearly the proof must find an \( s' \in \gamma(s) \setminus \{\varepsilon\} \) such that

\[
(1.12) \quad \exists D_{s'} \subseteq \gamma(t). \ s' \prec \| D_{s'}
\]

Notice that there is no \( t \) with \( \varepsilon \prec t \). For the same reason (1.12) cannot hold for \( s' = \varepsilon \) neither.

At first we prove that there is at most one \( s' \in \gamma(s) \setminus \{\varepsilon\} \) such that (1.12) holds.

Assume on the contrary that there are (at least) two different nonempty connected components of \( s \) for which (1.12) holds. I.e., assume \( \exists s', s'' \in \gamma(s) \setminus \{\varepsilon\} \exists D_{s'}, D_{s''} \subseteq \gamma(t). \ s' \neq s'', s' \prec \| D_{s'}, s'' \prec \| D_{s''}. \)

By proposition 1.3.39 we find a \( D_u \subseteq \gamma(t), u \leq \| D_u \) for every \( u \in \gamma(s) \). Let \( v = \|\{u \mid u \in \gamma(s) \setminus \{s', s''\}\} = \| \gamma(s) \setminus \{s', s''\} \) and \( v' = \|\{D_u \mid u \in \gamma(s) \setminus \{s', s''\}\} \). Clearly \( s = s' \| s'' \| v \) and \( v \leq v' \) so by corollary 1.3.41 we have \( D_{s'} \cup D_{s''} \cup \{D_u \mid u \in \gamma(s) \setminus \{s', s''\}\} = \gamma(t) \) and thereby \( (\| D_{s'} \| (\| D_{s''} \| v') \| v' = t \). From \( s' \prec \| D_{s'} \), \( s'' \prec \| D_{s''} \) and \( v \leq v' \) we now get \( s = s' \| s'' \| v \prec (\| D_{s'} \| s'' \| v \prec (\| D_{s'} \| (\| D_{s''} \| (\| D_{s''} \| v' = t\) — a contradiction to \( s \prec t \).

Next we prove that there is at least one \( s' \in \gamma(s) \setminus \{\varepsilon\} \) such that (1.12) holds.

Assume on the contrary there is no such \( s' \). As noticed (1.12) does not hold for \( s' = \varepsilon \), so we can in fact assume (1.12) not to hold for \( s' \in \gamma(s) \).

From proposition 1.3.39 we see \( \forall s' \in \gamma(s) \exists D_{s'} \). \( s' \preceq \| D_{s'} \). Since there by assumption

48
is no \( s' \in \gamma(s) \) such that (1.12) holds this implies \( \forall s' \in \gamma(s). s' = \| D_{s'} \). This has as consequence \( \gamma(s') = D_{s'} \) and \( A_{s'} = A_{D_{s'}} \) for all \( s' \in \gamma(s) \). Then by proposition 1.3.40 we have \( \gamma(t) = \bigcup_{s' \in \gamma(s)} D_{s'} \) from which we get: \( \gamma(t) = \bigcup_{s' \in \gamma(s)} D_{s'} = \bigcup_{s' \in \gamma(s)} \gamma(s') = \gamma(s) \), so \( s = t \) which contradicts \( s \prec t \).

Now let \( s' \in \gamma(s) \setminus \{ \varepsilon \} \) be the only one for which (1.12) holds and \( D_{s'} \) the corresponding subset of \( \gamma(t) \).

We know \( s' \neq \varepsilon \), so we might define \( D = D_{s'} \setminus \{ \varepsilon \} \) and still have \( D \neq \emptyset \), \( s' \prec \| D \). Using proposition 1.3.39 again we have \( \forall u \in \gamma(s) \exists D_u \subseteq \gamma(t). u \preceq \| D_u \). Since \( s' \) is the only semiword of \( \gamma(s) \) with \( s' \prec \| D \) we have \( u = \| D_u \) for \( u \in \gamma(s) \setminus \{ s' \} \). From proposition 1.3.40 we now get \( \gamma(t) = D \cup \bigcup_{u \in \gamma(s) \setminus \{ s' \}} D_u \). \( D \) is disjoint to \( \bigcup_{u \in \gamma(s) \setminus \{ s' \}} D_u \). If not then we have a \( v \in D \cap D_u \) for some \( v \in \gamma(s) \setminus \{ s' \} \). Because \( \varepsilon \not\in D \) we have \( v \neq \varepsilon \). This together with \( v \in D \cap D_u \) implies \( A_{D} \cap A_{D_u} \neq \emptyset \). Since \( A_D = A_{s'} \) and \( A_u = A_{D_u} \) this also means \( A_{s'} \cap A_u \neq \emptyset \) which contradicts \( u, s' \in \gamma(s) \setminus \{ \varepsilon \} \) and \( \gamma(s) \) consisting of disjoint semiwords. Hence \( \gamma(t) = D \cup \bigcup_{u \in \gamma(s) \setminus \{ \varepsilon \}} D_u \). So \( \gamma(t) \setminus D = \bigcup_{u \in \gamma(s) \setminus \{ s' \}} D_u = (\text{because } u = \| D_u \) \( \cup \bigcup_{u \in \gamma(s) \setminus \{ s' \}} (\gamma(u) = (\text{because } s' \neq \varepsilon) \gamma(s) \setminus \{ s' \} \).

The conclusion of the first three steps of the proof is now:
\( s \prec t \) implies \( \exists s' \in \gamma(s) \setminus \{ \varepsilon \} \exists D \subseteq \gamma(t). \gamma(s) \setminus \{ s' \} = \gamma(t) \setminus D, s' \prec \| D \), so the final step is to prove \( s' \prec \| D \).

Assume on the contrary that there exists a \( u \) with \( s' \prec u \prec \| D \). Then \( s = s' \prec (\| \gamma(s) \setminus \{ s' \}) \prec u \prec (\| \gamma(s) \setminus \{ s' \}) \prec (\| D \) \( \| \| \gamma(s) \setminus \{ s' \}) = (\| D \) \( \| \gamma(t) \setminus D = \| \gamma(t) = t \)—a contradiction to \( s \prec t \!). \( \square \)
Chapter 2

Tree Semiwords: \( TSW \)

2.0 Preliminaries

We are now going to define a particular subclass of semiwords called tree semiwords which can be seen as reflecting non-synchronized behaviour.

Definition 2.0.1 A poset \( t \) of \( \Delta \times \mathbb{N}^+ \) is a tree-semiword if \( t \) fulfills SW1, SW2 and:

\[
T: \forall a, b, c \in A_t. a \leq_t c, b \leq_t c \Rightarrow a \leq_t b \lor b \leq_t a
\]

The class of tree-semiwords over \( \Delta \) is denoted \( TSW(\Delta) \) (TSW for short).

Corollary 2.0.2 \( W \subseteq TSW \subseteq SW \)

Technically it is convenient to introduce the notion of a rooted tree-semiword.

Definition 2.0.3 \( r \) is a rooted tree-semiword if \( r \) is a tree-semiword and:

\[
RT: \exists a \in A_r. \forall b \in A_r. a \leq_r b
\]

The class of rooted tree-semiwords over \( \Delta \) is denoted \( RTSW(\Delta) \) (RTSW for short).

Corollary 2.0.4 \( W \setminus \{\varepsilon\} \subseteq RTSW \subseteq TSW \).

It would be nice if we could carry over all the definitions and results of semiwords to the subclass of tree-semiwords. Unfortunately, this cannot be done entirely, the main reason being that though a construction from some tree-semiwords yields a semiword, it is not ensured to be a tree-semiword. The most conspicuous example is that the concatenation of two tree-semiwords does not necessarily give a tree-semiword.

Therefore, we will briefly repeat the definitions and results of the previous chapter, making a few changes and necessary additions. Whenever a result or definition of this chapter is referred (as e.g., corollary 1.2.14) later on and it is not stated here explicit it is because it carry over directly from chapter 1 (of course with SW changed to TSW). To emphasize that it is a tree-semiword version the reference will be subscribed with a \( T \) like: proposition\( T \).
2.1 Basic Definitions

The definition of restriction, subsemiword and complement semiword of semiwords can directly be carried over to tree-semiwords.

Proposition 1.1.4 now says that $s|_{A^2}$ and the complement semiword are tree-semiwords (if $A \subseteq A_s$ and $A$ fulfill $SW_1$).

Proof From the corresponding semiword proof we know they are semiwords, so we only have to show that they have the $T$-property:

At first notice that in general for a poset $(A, \leq)$ having the $T$-property, any poset $(B, \leq|_{B^2})$ where $B \subseteq A$, has the $T$-property too. From the corresponding semiword proof we already know that $s|_{A}$ is a semiword, and since it is a restriction of a tree-semiword $s$ we know that $\leq s|_{A^2}$ fulfills $T$ and we are done with a).

For b) we also know that $t$ is a semiword. But $\leq t$ is just $\leq u|(A_t\setminus A_s)^2$ shifted left according to $s$ so $\leq t$ must fulfill $T$ too.

Also the definition of connected components of a semiword and the belonging results can be carried over. Since we already know that a connected component is a subsemiword, we only have to observe that it is a restriction, hence having the $T$-property too, wherefore it is a tree-semiword.

Having the notion of rooted tree-semiwords we can get a finer view of tree-semiwords. We extend corollary 1.1.8 with:

f) A nonempty connected component (of a tree-semiword) is a rooted tree-semiword.

This is perhaps not totally obvious, so we prove it:

Proof Let $s$ be a connected component of a tree-semiword. We already know that it is a tree-semiword so we shall prove that $\leq s$ have the $RT$-property. Define $R := (\leq s \cup \leq s^{-1})$.

That $s$ is connected means $\forall b, c \in A_s, b R^+ c$.

To continue we need an intermediate result:

\[(2.1) \quad b R^+ c \Rightarrow \exists a(\in A_s), a \leq s b, a \leq s c\]

We prove this by proving $b R^n c \Rightarrow \exists a. a \leq_s b, a \leq_s c$ by induction on $n$.

$n = 1$: I.e., $b R c$. This means either $b \leq_s c$ or $c \leq_s b$. Let $a$ equal $b$ in the former case and $c$ in the latter.

$n > 1$: Then there exists a $d$ such that $b R d R^{n-1} c$. Using the hypothesis of induction on $d R^{n-1} c$ we find $a'(\in A_s)$ such that $a' \leq_s d, a' \leq_s c$. We now look at the possibilities of $b R d$.

$b \leq_s d$: Since $s \in TSW$ we have $a' \leq_s d, b \leq_s d \Rightarrow a' \leq_s b \lor b \leq_s a'$. In the latter case choose $a = b$ and in the former $a = a'$. By reflexivity and transitivity of $\leq_s$ we are then done.
\(d \leq_{s} b\): Then let \(a = a'\). We then have \(a \leq_{s} c\) and by transitivity of \(\leq_{s}\) also \(a \leq_{s} b\).

The next is:

\[\exists a \in A_{s} \forall b \in B. a \leq_{s} b \text{ if } \emptyset \neq B \subseteq A_{s}\]

We prove it by induction on the size of \(B\). Since \(B \neq \emptyset\) the induction basis must be:

\(|B| = 1\): Then \(B = \{b\}\) for some \(b \in A_{s}\). By reflexivity of \(\leq_{s}\) we have \(b \leq_{s} b\). Choose \(a = b\). Because \(B \subseteq A_{s}\) we are done.

\(|B| > 1\): Pick out some \(b \in B\). Use the inductive hypothesis on \(B \setminus \{b\}\) to find a \(c \in A_{s}\) such that \(\forall d \in B \setminus \{b\}, c \leq_{s} d\). Because \(s\) is connected \(b R^{+} c\). Then by (2.1) there exists an \(a \in A_{s}, a \leq_{s} b, a \leq_{s} c\). By transitivity of \(\leq_{s}\): \(\forall d \in B \setminus \{b\}, a \leq_{s} d\). Hence also \(\forall b \in B, a \leq_{s} b\).

With the last result \(b)\) now follows directly by noticing that \(s\) nonempty implies \(A_{s} \neq \emptyset\) and that \(A_{s}\) is a subset of itself.

\[\square\]

### 2.2 Operations on \(TSW\)

**Nullary**

We have already noticed that \(W \subseteq TSW\), so especially \(\varepsilon, a \in TSW\).

**Binary**

We have already noticed that concatenation does not carry over as it is.

In fact we have:

\[\forall s \in SW \setminus W \forall t \in SW. t \neq \varepsilon \Rightarrow st \notin TSW\]

**Proof** \(s \in SW \setminus W\) implies that there exist \(a, b \in A_{s}\) such that \(a \not\leq_{s} b, b \not\leq_{s} a\). Since \(t \neq \varepsilon\) there exists a \(c^{t} \in A_{t}\). Then \(d = c^{v(s,c)+t} \in A_{st} \setminus A_{s}\). As noticed by definition 1.2.4 we have \(a \leq_{st} d, b \leq_{st} d\). Since \(\leq_{s}\) and \(\leq_{st}\) agree on \(A_{s}, a \not\leq_{s} b, b \not\leq_{s} a\) implies \(a \not\leq_{st} b, b \not\leq_{st} a\). Hence \(st\) is not a tree-semiword.

As a consequence of this we must restrict the domain of concatenation from \(TSW \times TSW\) to \(W \times TSW\).

The properties carry over. The only one we will dwell on is that \(st\) in fact is a tree-semiword when \(s \in W\). In order not to write \(s \in W\) whenever we consider \(st\) for \(s, t \in TSW\) we take it as a convention from now on.

\[st\] is a tree-semiword:

**Proof** We already know that \(st\) is a semiword, so we shall convince ourselves that \(\forall a, b, c \in A_{st}, a \leq_{st} c, b \leq_{st} c \Rightarrow a \leq_{st} b \lor b \leq_{st} a\) (T-property). Let us consider the membership of \(c\).
\(c \in A_s\): Then of course \(a, b \in A_s\) and the result follows because \(s\) and \(st\) agree on \(A_s\).

\(c \not\in A_s\): I.e., \(c \in A_{st} \setminus A_s\). If both \(a, b \in A_s\) then \(a \leq s\) or \(b \leq s\) \(a\) because \(s \in W\), hence \(a \leq_{st} b\) or \(b \leq_{st} a\). If both \(a, b \in A_{st} \setminus A_s\) we get the result from \(t\) being a tree-semiword and the correspondence between \(\leq_{st}\) and \(\leq_t\). If \(a \in A_s\), \(b \in A_{st} \setminus A_s\) we already know \(a \leq_{st} b\). Similarly if \(a \in A_{st} \setminus A_s\), \(b \in A_s\).

So \(st\) is indeed a tree-semiword.

Whereas we had to restrict the definition of concatenation in order to get the tree-semiwords as results this is not the case for parallel composition. The definition and the results can be carried over.

We conclude this section by a proposition which bring light to the connection between \(\text{TSW}(\mathcal{RTSW})\) and its operators.

**Proposition 2.2.15**

\[\begin{align*}
&\text{a) } \forall s \in \text{SW}. s \in \mathcal{RTSW} \text{ iff } \exists a \in \Delta, \exists t \in \text{T SW}. s = a.t \\
&\text{b) } \text{Every } t \in \text{T SW} \text{ can be generated from } \varepsilon, || \text{ and } a. (a \in \Delta)
\end{align*}\]

**Proof**

\[\begin{align*}
&\text{a) } \text{if: } \text{We already have that } a.t \in \text{T SW} \text{ and in general } \forall a \in A_s \forall b \in A_{st} \setminus A_s. a \leq_{st} b, \text{ so especially for } a^1 \in A_s \subseteq A_{st} \text{ we have } \forall b \in A_{a.t} \setminus A_a. a^1 \leq_{a.t} b. \text{ Hence } \leq_{a.t} \text{ fulfills RT and } a.t \in \mathcal{RTSW}.
\\
&\text{only if: } \text{Given } s \in \mathcal{RTSW}. \text{ By definition there exists an } a \in A_s \text{ such that } \forall b \in A_s. a \leq_s b. \text{ Clearly } a \text{ must have rank } 1, \text{ so } \{a^1\} \text{ fulfills SW1. Then } a = s|_{\{a^1\}} \text{ is a subtree-semiword of } s. \text{ Let } t \text{ be the complement tree-semiword of } a \text{ w.r.t. } s. \text{ So we have } t \in \text{T SW}. \text{ What remains to prove is that } s = a.t. \text{ Clearly } A_s = A_{a.t}. \leq_s = \leq_{a.t} \text{ is seen by noticing } b^j \neq a^1 \Leftrightarrow b^j \in A_s \setminus \{a^1\} \Leftrightarrow \psi(a, b) < j \text{ and looking at the definition of concatenation and complement tree-semiword.}
\\
&\text{b) } \text{Follows by induction, directly from } t = \varepsilon \parallel (|| \gamma(t) \setminus \{\varepsilon\}), \text{ proposition 1.2.13, corollary 1.1.8 and a) above.}
\end{align*}\]

### 2.3 Partial Orders on TSW

#### 2.3.1 Smoother Than

The definition of smoother than and linearization carries over. However it is worth remarking that the \(\preceq\)-downwards closure of a tree-semiword \(t\) within \(\text{SW}\) is not contained in \(\text{T SW}\). E.g., with

\[t = \frac{c}{a} \rightarrow b \in \text{T SW} \text{ and } s = \frac{c}{a} \gg b \in \text{SW}\]

we have \(s \preceq t\) in \(\text{SW}\) but \(s \not\in \text{T SW}\).
So it is clear that some care must be taken when using \( \leq \) on TSW, especially when constructing a new (tree-)semiword which is claimed to be smoother than another tree-semiword.

We will now pick out the cases where the difference is significant. One of the most conspicuous cases is in fact the first lemma:

**Lemma 2.3.6** \( \forall s \in TSW \forall a, b \in A_s. (a \varleq_s b, b \varleq_s a \Rightarrow \exists t \in TSW. t < s, a \leq_t b) \)

**Proof** Whereas we before just added \((a, b)\) to \(\leq_s\) taking the transitive closure we cannot do this any longer, as can be seen from the example above. In general there there can be more least refinements of \(\leq_s\) containing \((a, b)\). E.g., in (2.2) above \(c \rightarrow c \rightarrow b\) and \(c \rightarrow a \rightarrow b\) are two such least refinements of \(\leq_s\). So we can just as well choose in what way to refine \(\leq_s\). By the new idea \((a, b)\) still is added to \(\leq_s\) but not necessarily directly. We consider two cases.

\(a, b\) in \(A_s\) are not connected:

By corollary 1.1.8.f) the connected component which \(b\) belongs to is a rooted tree-semiword. So let \(d\) denote the root and we have \(d \leq_s b\). Now define \(A_t = A_s, \leq_t = Q^+\), where \(Q = (\leq_s \cup \{(a, d)\})\). Clearly \(a \leq_t b\) and \(\leq_s \subseteq \leq_t\). As in the proof for semiwords we see that \(A_t\) fulfills SW1, \(\leq_t\) fulfills SW2 and is transitive so as reflexive. Now for the antisymmetry:

We shall show \(f Q^+ g, g Q^+ f \Rightarrow f = g\).

Since \(a, d\) belongs to two different connected components of \(s\) we cannot have \(a \leq_s d\) or \(d \leq_s a\). Hence \(f Q^+ g\) implies \(f \leq_s g\) or \(f \leq_s (a, d) \leq_s g\). Similar for \(g Q^+ f\). So there are four cases to consider. If \(f \leq_s g, g \leq_s f\) we get \(f = g\) from the antisymmetry of \(\leq_s\).

The remaining cases can be excluded since they all implies \(d \leq_s a\) which as noticed is impossible.

It remains to show the \(T\)-property of \(\leq_t\). Suppose \(f Q^+ h, g Q^+ h\). We shall show \(f Q^+ g\) or \(g Q^+ f\). Again there are four cases:

\(f \leq_s h, g \leq_s h\): Follows form \(\leq_s\) having the property.

\(f \leq_s h, g \leq_s (a, d) \leq_s h\): Then \(f \leq_s d\) or \(d \leq_s f\). In the former case we must have \(f = d\) by the way \(d\) is chosen. But then \(g \leq_s (a, d) f\) i.e., \(g Q^+ f\). In the latter case \(g \leq_s (a, d) \leq_s f\).

\(f \leq_s (a, d) \leq_s h, g \leq_s h\): Symmetric.

\(f \leq_s (a, d) \leq_s h, g \leq_s (a, d) \leq_s h\): Then \(f \leq_s a, g \leq_s a\) and the result follows.

Now suppose \(a, b\) are connected.

I.e., \(a R^+ b\), where \(R = (\leq_s \cup \leq_s^{-1})\). By corollary 1.1.8.f) this component \(a\) and \(b\) belongs to is a rooted tree-semiword so there exists a \(c' \in A_s\) such that \(c' \leq_s a, c' \leq_s b\). Let \(c\) be the element of \(A_s\) such that \(c' \leq_s c, c \leq_s a, c \leq_s b\) and there is no \(c''\) with \(c \leq_s c''\) such that \(c'' \leq_s a, c'' \leq_s b\), i.e., \(c\) is the greatest lower bound of \(a, b\) w.r.t. \(\leq_s\) (exists since \(DC_{\leq_s}(s)\) and \(DC_{\leq_s}(b)\) are finite). Since we are dealing with trees, the paths leading from \(c\) to \(a\) and \(b\) must be unique. Let \(d\) denote the first element after \(c\) on the path to \(b\). The situation is as illustrated:

\[
\cdots c \ldots a \underline{d} \cdots b
\]
We are now ready to define $t$:

$$A_t = A_s \text{ and } \leq_t = Q^+,$$ where $Q = \leq_s \cup \{(a,d)\}$. By construction we immediately have $a \leq_t b$ and $\leq_s \subseteq \leq_t$. As above we immediately see all the properties of $t$ in order to be a semiword except for antisymmetry. To see this we first need some intermediate results.

$f Q^+ a \Rightarrow f \leq_s a$: Suppose $f \not\leq_s a$. Then it must be possible to write the path of $Q$ establishing $f Q^+ a$ as: $f Q^n(a,d) \leq_s a$. This means $d \leq_s a$. By the way $d$ is chosen we also have $d \leq_s b$. But this contradicts the way $c$ is chosen.

$d Q^+ g \Rightarrow d \leq_s g$: Similarly we see that $d \leq_s (a,d) Q^m g$, and the contradiction is obtained in the same way.

Now if $f \not\leq_s g$ and $d Q^+ g$ then $f Q^n a$ and $d Q^m g$. From the above we see that this implies $f \leq_s a$ and $d \leq_s g$. The antisymmetry can now be seen as in the proof for semiwords.

What remains to prove is that $\leq_t$ fulfills $T$. Assume $f Q^+ h$ and $g Q^+ h$. We shall prove either $f Q^+ g$ or $g Q^+ f$.

$f \leq_s h, g \leq_s h$: Then we get it from $\leq_s$ having the property.

$f \not\leq_s h, g \not\leq_s h$: Here we by definition know that $f \leq_s a$ and $g \leq_s a$, so the $T$-property follows again.

$f \leq_s h, g \not\leq_s h$: Then we have $g \leq_s a, d \leq_s h$. From the former case we conclude $d \leq_s f$ or $f \leq s d$. If $d \leq_s f$ we have $g \leq_s (a,d) \leq_s f$ i.e., $g Q^+ f$. For $f \leq_s d$ notice that any path of $\leq_s$ from $f$ to $d$ must go through $c$ since $c$ is the immediate predecessor of $d$, so $f \leq_s c$ (may be $f = c$). By the way $d$ is chosen $c \leq_s a$, so $f \leq_s a$. From this and $g \leq_s a$ we conclude $f \leq_s g$ or $g \leq_s f$. If $f = d$ clearly $g \leq_s (a,d) f$ or equally $g Q^+ f$.

$f \not\leq_s h, g \leq_s h$: Is handled symmetrically.

The next lemmas and corollaries carry over directly, so proposition 1.3.5 also holds for tree-semiwords.

The propositions concerning concatenation have to be modified a little. If we take over the formulation $s \leq t \Leftrightarrow sr \leq tr \Leftrightarrow rs \leq rt$ in proposition 1.3.10, it is trivial because it only is defined when $s, t \in W$. Instead we have the proposition.

**Proposition 2.3.10**

a) $s \leq t \iff rs \leq rt$

The a) part of the next corollary can be left out since it is just a) of the proposition, because $s_1 \leq t_1$ and $s_1, t_1 \in w$ implies $s_1 = t_1$. From this we also get that the next proposition is formulated:

**Proposition 2.3.12**

$u \leq sl \Rightarrow \exists t'. t \leq s t'$

The proof can be carried over with the addition $s' \leq s \in W$ implies $s' = s$.  

55
Proposition 2.3.13

a) \( u \in \lambda(st) \iff \exists t' \in \lambda(t), u = st' \)

b) \( u \in \lambda(s \parallel t) \iff \exists s' \in \lambda(s) \exists t' \in \lambda(t), u \preceq s' \parallel t', \not\exists t'' \cdot u'. u' < u \)

**Proof**

a) The proof of semiwords can be used directly because it only uses previous results which we know hold for tree-semiwords.

b) Also this proof can be carried over.

Since \( s \in W \) implies \( \delta(s) = \{s\} \), c) of proposition 1.3.15 can be read \( \delta(st) = s\delta(t) \). The proofs are identical.

We now turn to the \( \preceq \)-upwards closure \( v \). proposition 1.3.17 is the same. c) in the next proposition does not carry over because \( v(s)v(t) \) is not defined for all \( s \in W \). Instead it should be

**Proposition 2.3.18**

\( c) \{s\}v(t) \subseteq v(st) \).

**Proof** The same.

The corollary reads \( v(st) = v(\{s\}v(t)) \), but for \( s = a \) we do have \( v(a.t) = v(v(\{a\}v(t)))! \).

The definition of \( \chi \) and the associated results carry over smoothly.

### 2.3.2 Prefix of

The definition of prefix can be used directly, so as e.g., the proof that \( \sqsubseteq \) is a po an \( TSW \).

The propositions concerning \( \sqsubseteq/\pi \) alone and the proofs of these all carry over because when constructing new semiwords these are either subsemiwords or complement semiwords and we know that if these constructions derive from tree-semiwords, we will also have subtree-semiwords or complement tree-semiwords respectively. Remember that when we write \( u \sqsubseteq st \Rightarrow u \sqsubseteq s \) or \( \exists t' \sqsubseteq t. u = st' \) in proposition 1.3.34 we still **presume s to be a word**.

The matter is rather different when it comes to the relations between \( \lambda, \delta, v \) and \( \pi \), i.e., proposition 1.3.38. In the proofs, semiwords are constructed where it is not obvious that the constructions yield tree-semiwords when they are constructed from such.

**Proof** The only proof which we can take over directly is \( \pi\lambda(s) = \lambda\pi(s) \) because \( \lambda(s) \subseteq W \) and \( s \in W, t \sqsubseteq s \) implies \( t \in W \), such that the constructions yield tree-semiwords when \( s \in TSW \). We now look at the other proofs one by one.
\( \pi \delta (s) \subseteq \delta \pi (s) \): The constructed \( u \) is to be a subsemiword of \( s \). Since \( s \in TSW \) we know that \( u \in TSW \) and the proof can be reused directly.

It is not easy to see that the constructed \( u \) in the proof of the other inclusion is a subtree-semiword, but it will turn out that it in fact is a tree-semiword which we will now prove.

\( \delta \pi (s) \subseteq \pi \delta (s) \): With \( t \preceq t' \subseteq s, A_u = A_s \) and \( \leq u = R^+ \), where \( R = \leq u \cup \leq t \) and under the assumption \( t, s \in TSW \). It is already proved to be a semiword so what remains is to prove \( a R^+ c, b R^+ c \Rightarrow a R^+ b \) or \( b R^+ a \) (\( T \)-property). We will prove this by proving:

\[
a R^n c, b R^m c \Rightarrow l < n + m \quad \text{and} \quad (a R^l b \text{ or } b R^l a)
\]

by induction on \( n + m \).

\( n + m = 2 \): Then \( a R c, b R c \). Look at the different possibilities.

- \( a \leq s, c \leq s \): Since \( s \in TSW \) we have \( a \leq s b \) or \( b \leq s a \), hence also \( a R b \) or \( b R a \).
- \( a \leq t, b \leq t \): Similar.
- \( a \leq s, c \leq t \): Since \( \leq t \) only is defined on \( A_t \subseteq A_s \) we conclude \( c \in A_t \) for \( b \leq t \).
- \( a \leq t \): We get \( A_t = A_t' \), hence \( c \in A_t' \) and \( t \subseteq s \) gives us that \( DC_{\leq t}(c) \subseteq A_t' \), so \( a \in A_t' = A_t \) and \( a \leq c \) by definition of \( \subseteq \). Again from \( t \preceq t' \) we see \( \leq \subseteq \leq t \) and therefore \( a \leq t \).
- \( a \leq t, c \leq s \): Symmetric.

\( n + m > 2 \): Then either \( n \geq 2 \) or \( m \geq 2 \). W.l.o.g. assume \( n \geq 2 \). This implies that there exists \( d \) such that \( a R d, d R^{n-1} c \). Using the hypothesis on \( d R^{n-1} c, b R^m c \) we get \((d R^0 b \text{ or } b R^l d)\) and \( l' < n + m - 1 \). We look at the two cases:

- \( d R^l b \): Then from \( a R d \) we get \( a R^l b \), where \( l = l' + 1 < n + m - 1 + 1 = n + m \).
- \( b R^l d \): We have \( a R d \), and since \( l' + 1 < m + n \) we can use the hypothesis of induction on this to obtain \( a R^l \) or \( b R^l a \), where \( l < l' + 1 < m + n \).

\( v \pi (s) \subseteq \pi v (s) \): The situation is that for a given \( t \in v \pi (s) \) where \( s, t \in TSW \) a semiword \( u \) with \( s \preceq u \) and \( t \subseteq u \) is defined by \( u = (A_u, (\leq \cup \leq t)^+) \), where \( \leq = \{(a, b) \in A_u^2 \mid (a, b) = (c', c^i) \text{ for some } c \in \Delta \text{ and } i \leq j\} \). So it just remains to prove \( u \) has the \( T \)-property in order to get \( u \in TSW \). Let \( a, b, c \) be given such that \( a \leq u \) and \( b \leq u \). We shall prove \( a \leq u \) or \( b \leq u \).

We consider two main cases:

- \( c \in A_t \): Clearly \( a \leq u \) and \( b \leq u \) then implies \( a \leq t \) and \( b \leq t \). Since \( t \in TSW \) it follows that \( a \leq u \) or \( b \leq u \) and so \( a \leq u \) or \( b \leq u \).

- \( c \not\in A_t \): There are actually four subcases:

  - \( a, b \not\in A_t \): Then \( a, b, c \) must be equal labelled and so are ordered by definition of \( \leq u \).
  - \( a \in A_t, b \not\in A_t \): By \( a \leq u \) and construction of \( \leq u \) from \( \leq \) it then follows that there is an element \( c' \) labelled like \( c \) with \( a \leq c' \leq u \). From \( b \not\in A_t \) and \( b \leq u \) follows \( c' \leq b \leq c \) so \( a \leq u \).
  - \( a \not\in A_t, b \in A_t \): Symmetrically as in the last case we here see \( b \leq u \).
  - \( a, b \in A_t \): As above we see there are elements \( c' \) and \( c'' \) of \( A_t \) labelled like \( c \) such that \( a \leq c \) and \( b \leq c \). Since \( c' \) and \( c'' \) are equally labelled either \( c' \leq c'' \) or \( c'' \leq c' \). W.l.o.g. assume the former. Then \( a \leq c'' \) and \( b \leq c'' \) and the result follows from \( t \in TSW \).
The remaining of chapter 1 carries over. Now to a proposition special for tree-semiwords.

**Proposition 2.3.44** $s \prec t$ implies $\exists u \in \gamma(s), D \subseteq \gamma(t), \gamma(s) \setminus \{u\} = \gamma(t) \setminus D$ and for some $a, b \in \text{Act}, s', s'', t' \in TSW$ either

a) $u = a.(s' || b.s'')$, $D = \{a.s', b.s''\}$

or

b) $u = a.s', D = \{a.t'\}, s' \prec t'$

**Proof** We already have $s \prec t \Rightarrow \exists u \in \gamma(s) \setminus \{\varepsilon\}, D \subseteq \gamma(t), \gamma(s) \setminus \{u\} = \gamma(t) \setminus D, u \prec || D$ from proposition 1.3.43, so it is enough to prove $u \prec || D$ and $u \in \gamma(s) \setminus \{\varepsilon\}$ implies a) or b).

Now since $u$ is a nonempty connected component of $s$ it is (by corollary 1.1.8.f) a rooted tree-semiword. Hence $u = a.u'$ for some $u' \in TSW$. Since $u \neq \varepsilon, \varepsilon \in \gamma(s)$ and $\gamma(s) \setminus \{u\} = \gamma(t) \setminus D$ we have $\varepsilon \notin D$, so $\gamma(|| D) = D \uplus \{\varepsilon\}$. Then by proposition 1.3.42 we see $a.u' \prec || D$ implies $|D| \leq 2$. Since $\varepsilon \notin D$, $D$ must consist of nonempty connected components. By corollary 1.1.8.f) then $D = \{c.s', b.s''\}$ or $D = \{c.t'\}$ for some $b, c \in \text{Act}, s', s'', t' \in TSW$.

$D = \{c.t'\}$: Then $u \prec || D$ reads $a.u' \prec c.t'$. Clearly then $a = c$ and $u' \prec t'$. Chose $s' = u'$.

$D = \{c.s', b.s''\}$: By proposition 1.3.30 we get w.l.o.g.: $a.u' \prec c.s' || b.s'' \Rightarrow \exists v. a.v \prec c.s', u' \preceq v || b.s''$, $\{a^1 \notin A_{b.s''}\}$. We examine the cases of $\preceq$.

$a.v = c.s', u' = v || b.s'':$ Then $a = c, v = s, u' = s' || b.s'$ and $D = \{a.s', b.s''\}$.

$a.v \prec c.s', u' = v || b.s'':$ a) By proposition 1.3.29.a) we see $a.u' = a.(v || b.s'') \prec a.v || b.s''$ since $|\gamma(a.u')| = 2, |\gamma(a.v || b.s'')| = 3 \Rightarrow \gamma(a.u') \neq \gamma(a.v || b.s'') \Rightarrow a.u' \neq a.v || b.s''$. Now $v \prec s'$ implies $a.v || b.s'' \prec a.s' || b.s'' = c.s' || b.s'' = || D$, so $a.u \prec a.v || b.s'' = || D$ which contradicts $u = a.u' \prec || D$ wherefore this case can be ruled out.

$a.v \prec c.s', u' \prec v || b.s'':$ Then $a.u' \prec (a.(v || b.s'')) \prec a.v || b.s'' = c.s' || b.s'' = || D$. A contradiction.

$a.v \prec c.s', u' \prec v || b.s'':$ As the previous case.

□
Chapter 3

Semantics for a Simple Process Language: $PL$

In this chapter we shall give three different semantics to a simple process language, $PL$, for describing finite nondeterministic processes which in turn is a restricted subset of the basic language, $BL$, obtained as the term algebra for the signature $\Sigma$—essentially the operators (symbols) from the chapters with semiwords/tree-semiwords. The restriction will be that processes only can be parallel composed when they have no action symbols in common. This restriction is mainly technical motivated, but can also be seen as reflecting the idea that an atomic action cannot be duplicated (however it may reinitiated). The restriction allows us to define the different interpretations of parallel composition of processes on the basis of the corresponding partial defined parallel composition of tree-semiwords.

3.1 Denotational Semantics

The concrete signature, $\Sigma$, from which $BL$ is derived as the term algebra is:

**Definition 3.1.1** $\Sigma$ is defined by:

\[
\begin{align*}
\Sigma_0 &= \{NIL\} \\
\Sigma_1 &= \{a, \} \quad \text{where } a \in Act \\
\Sigma_2 &= \{+, \|\} \\
\Sigma_n &= \emptyset \quad n > 3
\end{align*}
\]

$Act$ is a set of abstract atomic action symbols fixed throughout the rest of this part.

Writing binary operators as usual as infixes and the unary as prefixes, $BL$ can be considered defined from the following BNF-like schema:

\[p ::= NIL \mid a.p, \ a \in Act \mid p + p \mid p \| p\]

To formalize the restriction we shall impose on the processes we for every $p \in BL$ we define it’s sort, $L(p)$, or label set as follows:
Definition 3.1.2 Let $L : BL \rightarrow \mathcal{P}(\text{Act}) (= \mathcal{L})$ be defined by:

\[
\begin{align*}
\text{NIL} & \mapsto \emptyset \\
a.p & \mapsto \{a\} \cup L(p) \\
p + q & \mapsto L(p) \cup L(q) \\
p \parallel q & \mapsto L(p) \cup L(q)
\end{align*}
\]

Whit this in the hand we can define the process language, $PL$, as those terms of $BL$ where every subterm of the form:

\[p \parallel q\]

satisfies:

\[L(p) \cap L(q) = \emptyset\]

That is parallel composition is only allowed between processes with different sorts.

The next step will be to define the three interpretations of the terms from $PL$ by means of corresponding $\Sigma$-po algebras as explained in [Hen85a].

However because of the restriction on $PL$ some modifications are needed. Formally the semantics should be given within a theoretical framework which address the question of giving semantics to terms with certain sorts as e.g., in sorted algebras [GTWW77]. This would to the opinion of the author obscure the presentation unnecessarily, since these questions not are the main concern of this thesis. So under the conviction that the presentation easily (but lengthly) could be given within such a framework, we shall merely on the way state the most important changes which arrise.

Common to the carriers of the three $\Sigma$-po algebras is that they consists of closures of prefix-closed sets of tree-semiwords over $\text{Act}$ (i.e., $\Delta = \text{Act}$). The differences between the carriers derive from the chosen closures which all are based on the smoother than relation ($\preceq$) between single tree-semiwords. The three closures are $\delta, v$ and $\chi$ respectively. We denote the three carriers by $C_\delta, C_v$ and $C_\chi$ respectively. Formally:

Definition 3.1.3 For $\star$ in $\{\delta, v, \chi\}$ we define:

\[C_\star := \{S \neq \emptyset \mid \exists T \subseteq TSW(\text{Act}). T \text{ is finite}, S = \star(\pi T)\}\]

and call it the $\star$-carrier.

It would have been nicer to define $C_\star$ as the finite $\star$ and $\pi$ closed subsets of $TSW(\text{Act})$ ($TSW$ for short), but from proposition 1.3.38 and the comments there we see that this only could be done for $\star = \delta$.

In the sequel $\mathcal{P}_f(A) \subseteq \mathcal{P}(A)$ will denote the finite sets of the power set. With this notation we can read $C_\star$ as:

\[\{S \mid \exists T \in \mathcal{P}_f(TSW) \setminus \emptyset. S = \star(\pi T)\}\]
Corollary 3.1.4 For $\star$ in $\{\delta, \nu, \chi\}$ we have:

\[ \forall T \in C_\star, \star(T) = T \]

For each of these carriers we are going to define a interpretation $\Sigma\star$ of the symbols of the signature $\Sigma$ as a function from $C_\star^n$ to $C_\star$ where $n$ is the rank of the symbol in question. Most of the definitions of these functions will lean on the corresponding functions defined on single tree-semiwords and the $\star$-closure properties.

**Definition 3.1.5** The sort of a nonempty set of tree-semiwords, $S$, ambiguously denoted $L(S)$, is defined by:

\[ L(\{s\}) \mapsto \{a \mid a^i \in A_\star\} \left(= \{a \mid \psi(s, a) > 0\}\right) \]

\[ L(S \cup T) \mapsto L(S) \cup L(T) \]

\[ \square \]

If $S$ is a singleton set $\{s\}$ we often just write $L(s)$ in place of $L(\{s\})$, so $L$ can be considered defined on TSW also. Notice that because tree-semiwords satisfies SW1 we have for arbitrary tree-semiwords $s$ and $t$: $A_s \cap A_t = \emptyset$ iff $L(s) \cap L(t) = \emptyset$. I.e., $s$ and $t$ are disjoint iff there sorts are disjoint. Also remark that $L(\varepsilon) = \emptyset$.

For each carrier, $C_\star$, the function, $\|\star$, corresponding to the interpretation of $\|$ will then be partially defined: $S \|_\star T$ is only defined when $L(S) \cap L(T) = \emptyset$. But due to the restriction on terms from $PL$ it will be ensured that the interpretations are defined.

We are now ready to define the interpretations of the operator symbols.

**Definition 3.1.6** With $S$ and $T$ considered to be elements of the appropriate $C_\star$-carrier we define:

\[
\Sigma_{\delta} : \quad \begin{align*}
NIL_{\delta} & = \{\varepsilon\} \\
a_{\delta}S & = a.S \cup \{\varepsilon\} \\
S +_{\delta} T & = S \cup T \\
S \|_{\delta} T & = \delta(S \| T) \quad \text{provided } L(S) \cap L(T) = \emptyset
\end{align*}
\]

\[
\Sigma_{\nu} : \quad \begin{align*}
NIL_{\nu} & = \{\varepsilon\} \\
a_{\nu}S & = \nu(a.S) \cup \{\varepsilon\} \\
S +_{\nu} T & = S \cup T \\
S \|_{\nu} T & = S \| T \quad \text{provided } L(S) \cap L(T) = \emptyset
\end{align*}
\]

\[
\Sigma_{\chi} : \quad \begin{align*}
NIL_{\chi} & = \{\varepsilon\} \\
a_{\chi}S & = a.S \cup \{\varepsilon\} \\
S +_{\chi} T & = \chi(S \cup T) \\
S \|_{\chi} T & = \chi(S \| T) \quad \text{provided } L(S) \cap L(T) = \emptyset
\end{align*}
\]
The result of $S +_* T$, $S, T$ in $C_*$, $*$ in $\{\delta, v\}$ is easily seen to be a member of $C_*$ since $\delta$ and $v$ distributes over $\cup$ for arbitrary sets. But $\chi$ does in general not distribute over $\cup$ for arbitrary sets, not even for $\chi$-closed sets, as can be seen from the following example.

**Example:** Let $S = \{a \rightarrow b \rightarrow c\}$, $T = \begin{pmatrix} a \\ b \\ c \end{pmatrix}$. Then $\chi(S) = S$, $\chi(T) = T$ and $s = a < b < c \in \chi(S \cup T)$, but $s \notin S \cup T$.

Therefore we define $S +_\chi T$ to be $\chi(S \cup T)$.

**Proposition 3.1.7** The operators of $\Sigma_*$, $*$ in $\{\delta, v, \chi\}$ are well-defined. I.e., they are functions from $C_*^n$ to $C_*$ for every $*$.

**Proof** Notice at first that for all $op_* \in \Sigma_*^n$ and $*$ in $\{\delta, v, \chi\}$ $op_*$ is defined on $C_*^n$ under proviso. What remains is to find a $U \in P_f(TSW)$ for every $op_* \in \Sigma_*^n$ and $S \in C_*$ such that $op_*(S) = *$($\pi U$), because then $op_*(S) \in C_*$.

$NIL_*$: Let $U = \{\varepsilon\}$. For every $*$-closure we have: $*$(($\pi U$)) = $*$($\pi \{\varepsilon\}$) = $*$($\{\varepsilon\}$) = $\{\varepsilon\}$ = $NIL_*$.

$a_* S$: $S \in C_*$ implies there exists a $S' \in P_f(TSW)$ such that $S = *$(($\pi S'$)). Let $U = a.S' \in P_f(TSW)$.

$* = \delta$: $\delta \pi U = \delta \pi a.S' = \text{(corollary}_{T} 1.3.36) \delta (a.\pi S' \cup \{\varepsilon\}) = \delta a.\pi S' \cup \{\varepsilon\} = \text{(corollary}_{T} 1.3.16) a.\delta \pi S' \cup \{\varepsilon\} = a.S \cup \{\varepsilon\} = a.S$.

$* = v$: $v \pi U = \ldots = v(a.\pi S' \cup \{\varepsilon\}) \text{(corollary}_{T} 1.3.19) va.v \pi S' \cup \{\varepsilon\} = va.S \cup \{\varepsilon\} = a_v.S$.

$* = \chi$: $\chi \pi U = \ldots = \chi(a.\pi S' \cup \{\varepsilon\}) = \text{(proposition}_{T} 1.3.24) \chi a.\pi S' \cup \{\varepsilon\} = \text{(corollary}_{T} 1.3.25) a.\chi \pi S' \cup \{\varepsilon\} = a.S \cup \{\varepsilon\} = a.\chi S$.

$S +_* T$: $S, T \in C_*$ implies $\exists S', T' \in P_f(TSW)$. $S = *$(($\pi S'$)), $T = *$(($\pi T'$)). Let $U = S' \cup T'$.

$* \in \{\delta, v\}$: Since $\pi, \delta$ and $v$ distributes over $\cup$ the result is immediate.

$* = \chi$: $\chi \pi (S' \cup T') = \chi(\pi S' \cup \pi T') = \text{(corollary}_{T} 1.3.23) \chi(\pi S' \cup \chi \pi T') = \chi(S \cup T) = S +_\chi T$.

$S \|^*_T$: Suppose $S$ and $T$ are disjoint. Furthermore let $S'$ and $T'$ be as in the case $S +_* T$ and let $U = S' \parallel T'$.

$* = \delta$: $\delta \pi U = \delta(\pi S' \parallel T') = \text{(proposition}_{T} 1.3.35) \delta(\pi S' \parallel \pi T') = \text{(corollary}_{T} 1.3.16) \delta(\pi S' \parallel \delta \pi T') = \delta(S \parallel T) = S \|^*_\delta T$.

$* = v$: $v \pi U = \ldots = v(\pi S' \parallel \pi T') = \text{(proposition}_{T} 1.3.18) v \pi S' \parallel v \pi T' = S \parallel T = S \|^*_v T$.

$* = \chi$: $\chi \pi U = \ldots = \chi(\pi S' \parallel \pi T') = \text{(corollary}_{T} 1.3.23) \chi(\pi S' \parallel \chi \pi T') = \chi(S \parallel T) = S \|^*_\chi T$.

\[\square\]
We now introduce a very simple $\Sigma$-po algebra $A_\pi$ which in this and later chapters will prove useful in establishing properties of the $A_\pi$-algebras (based $C_\pi$) we are going to introduce in a moment.

**Definition 3.1.8** Let $C_\pi := \mathcal{P}_f(TSW) \setminus \emptyset$ and for $S, T \in C_\pi$ define:

$$
\Sigma_\pi :
\begin{align*}
NIL_\pi &= \{\varepsilon\} \\
\alpha_\pi S &= a.S \cup \{\varepsilon\} \\
S +_\pi T &= S \cup T \\
S \|_\pi T &= S \| T
\end{align*}
$$

provided $L(S) \cap L(T) = \emptyset$

Clearly the operators of $\Sigma_\pi$ are well-defined and monotone w.r.t. $\subseteq$, so $A_\pi = (C_\pi, \preceq_\pi, \Sigma_\pi)$, where $\preceq_\pi = \subseteq$, is indeed a $\Sigma$-po algebra. Of course $\preceq_\pi$-monotonicity of $\|_\pi$ is relative to the carrier upon which $\|_\pi$ is defined. That is $\|_\pi$ is e.g., left $\preceq_\pi$-monotone in the sense that for $S, S', T \in C_\pi$ and $L(S) \cap L(T) = \emptyset = L(S') \cap L(T)$ we have:

$$S \preceq_\pi S' \text{ implies } S \|_\pi T \preceq_\pi S' \|_\pi T$$

Monotonicity for parallel composition under this proviso will be indicated by writing: (relative) monotone.

Also $C_\delta, C_v, C_\chi \subseteq C_\pi$ wherefore we can formulate the following proposition which displays the close connection between operators of $\Sigma_\pi$ and $\Sigma_\pi$.

**Proposition 3.1.9** Let $\ast$ be in $\{\delta, v, \chi\}$. For all $op_\ast \in \Sigma_\ast$, $\tilde{S} \in C_\ast$ we have:

$$op_\ast(\tilde{S}) = \ast op_\pi(\tilde{S})$$

**Proof** In most of the operator cases we use corollary 3.1.4.

$NIL_\pi$: Evident since $NIL_\pi = \{\varepsilon\}$ and $\delta(\varepsilon) = v(\varepsilon) = \chi(\varepsilon) = \{\varepsilon\}$.

$\delta$: $\alpha_\delta S = a.S \cup \{\varepsilon\} = a.\delta S \cup \{\varepsilon\} = \delta a.S \cup \{\varepsilon\} = \delta(a.S \cup \{\varepsilon\}) = \delta a_\pi S$.

$\gamma$: $\alpha_\gamma S = \nu a.S \cup \{\varepsilon\} = \nu(a.S \cup \{\varepsilon\}) = \nu a_\pi S$.

$\chi$: $\alpha_\chi S = a.S \cup \{\varepsilon\} = a.\chi S \cup \{\varepsilon\} = \chi a.S \cup \{\varepsilon\} = (\text{corollary} \ T \ 1.3.25) \ \chi a.S \cup \{\varepsilon\} = (\text{proposition} \ T \ 1.3.24) \ \chi(a.S \cup \{\varepsilon\}) = \chi a_\pi S$.

$\delta$: $S +_\delta T = S \cup T = \delta S \cup \delta T = \delta(S \cup T) = \delta(S +_\pi T)$.

$\gamma$: Similar.

$\delta$: $S +_\delta T = \chi(S \cup T) = \chi(S +_\pi T)$.

$\|_\delta$: $S \|_\delta T = \delta(S \| T) = \delta(S \|_\pi T)$.

$\|_\gamma$: $S \|_\gamma T = S \| T = \nu S \| \nu T = \nu(S \| T) = \nu(S \|_\pi T)$.

$\|_\chi$: As $S \|_\chi T$.
The next to define is the partial order $\mathfrak{A}$ on $C^*$. 

**Definition 3.1.10** For every $*$ in $\{\delta, v, \chi\}$ define the $\mathfrak{A}$—the partial order over $C^*$—to be the set inclusion ($\subseteq$). I.e.,

$$\forall S, T \in C^*. S \mathfrak{A} T \iff S \subseteq T$$

Clearly it is a partial order and $\{\varepsilon\}$ is a least element in every $C^*$. 

Since the $\delta$-, $v$- and $\chi$-closures in general are monotone w.r.t. $\subseteq$ we immediately from proposition 3.1.9 and $op_*$ being monotone get:

**Corollary 3.1.11** All $op_* \in \Sigma^*$ are (relative) monotone on $C^*$ (w.r.t. $\mathfrak{A}$) for all $*$ in $\{\delta, v, \chi\}$ (with the modification that $S \parallel_* T$ only is defined when $L(S) \cap L(T) = \emptyset$).

From the preceding and this corollary we then also have:

**Corollary 3.1.12** For every $*$ in $\{\delta, v, \chi\}$ $A_* = (C^*, \mathfrak{A}, \Sigma^*)$ is a $\Sigma$-po algebra.

Our different models, $M_*$, then consists of these $\Sigma$-po algebras and denotational maps, $[\_]_*$ given below:

**Definition 3.1.13** The interpretation, $[\_]_*$, in the $M_*$ model of terms from PL is defined compositionally (on the basis of $A_*$) as follows:

$$
[NIL]_* = NIL_* \\
[a.p]_* = a_*[p]_* \\
[p + q]_* = [p]_* +_* [q]_* \\
[p \parallel q]_* = [p]_* \parallel_* [q]_*
$$

From $L(p) = L([p]_*)$ and $p \parallel q \in PL$ only if $L(p) \cap L(q) = \emptyset$ it is seen that the definition is well-defined.

### 3.2 Operational Semantics

The operational semantics we are going to define are based on a labelled transition system (lts for short) which determines a process’s ability to develop from one configuration to another. For the purpose of this we define the set of possible configurations. In the definition, actions of a set of atomic complementary action symbols, $Act$, disjoint but equipotent to $Act$ is used. Furthermore a bijective map $*: Act \rightarrow Act$.
Definition 3.2.1 $\mathcal{BL}$ is defined to be the least set $C$ which satisfies:

\[
\begin{align*}
\mathcal{BL} & \subseteq C \\
\bar{a}.p & \in C \quad \text{if } p \in C \text{ and } a \in \text{Act} \\
p_1 \parallel p_2 & \in C \quad \text{if } p_1, p_2 \in C
\end{align*}
\]

$L$ is extended to $\mathcal{BL}$ by: $L(\bar{a}.q) = \{a\} \cup L(q)$.

The configuration language, $\mathcal{CL}$, is defined to be the subset of $\mathcal{BL}$ where every subterm of the form $p \parallel q$ has $L(p) \cap L(q) = \emptyset$.

Notice

i) $L$ “forgets” whether a label belongs to $\text{Act}$ or $\overline{\text{Act}}$. So $L(p)$ for $p \in \mathcal{CL}$ could be defined as taking $L$ on $p' \in PL$, where $p'$ is $p$ with all $\overline{\text{v}}$’s striped of.

ii) $PL \subseteq \mathcal{CL}$ and $p + q \in \mathcal{CL}$ only if $p, q \in PL$.

In the sequel we will have the implicit requirements $L(p) \cap L(q) = \emptyset$ whenever writing $p \parallel q$.

What remains to define for the lts over $\mathcal{CL}$ and $\text{Act} \cup \overline{\text{Act}}$ is the action relation.

Definition 3.2.2 Let $\rightarrow \subseteq \mathcal{CL} \times (\text{Act} \cup \overline{\text{Act}}) \times \mathcal{CL}$ (writing $p \xrightarrow{y} q$ for $(p, y, q) \in \rightarrow$) be the least relation over $\mathcal{CL}$ which satisfies:

\[
\begin{align*}
1) \quad & a.p \xrightarrow{a} \bar{a}.p \\
2) \quad & \bar{a}.p \xrightarrow{\bar{a}} p \\
3) \quad & \begin{array}{c}
p \xrightarrow{b} p' \\
\bar{a}.p \xrightarrow{b} \bar{a}.p'
\end{array} \\
4) \quad & \begin{array}{c}
p + q \xrightarrow{a} p' \\
q + p \xrightarrow{a} p'
\end{array} \\
5) \quad & \begin{array}{c}
p \parallel q \xrightarrow{y} p' \parallel q \\
q \parallel p \xrightarrow{y} q \parallel p'
\end{array}
\end{align*}
\]

where $y \in \text{Act} \cup \overline{\text{Act}}$ and $a, b, \ldots$ range over $\text{Act}$.

Corollary 3.2.3 $p \xrightarrow{y} p' \Rightarrow L(y) \subseteq L(p)$, $L(p') \subseteq L(p)$.

The fact that $p \xrightarrow{y} p'$ implies $L(p') \subseteq L(p)$ gives the well-definedness of the relation since then $p, q$ disjoint implies $p', q$ disjoint too in 5).

Proposition 3.2.4 $\forall a \in \text{Act} \forall p \in \mathcal{CL}. \left|\left\{q \mid p \xrightarrow{\bar{a}} q\right\}\right| \leq 1$
**Proof** Induction on the structure of $p$.

$p \in PL$: Then no subterm $r$ of $p$ is of the form $b.r'$, $b \in Act$. By inspection of definition 3.2.2 we clearly have $\{q \mid p \xrightarrow{\overline{a}} q\} = \emptyset$.

$p = \overline{b}.p'$: Two cases: $a \neq b$: Again by inspection of the definition we see $\overline{b}.p' \xrightarrow{\overline{a}}$ or equivalently $\{q \mid p = \overline{b}.p' \xrightarrow{\overline{a}} q\} = \emptyset$. $a = b$: We see $\overline{a}.p' \xrightarrow{\overline{a}} q$ implies $q = p'$.

$p = p_1 \parallel p_2$: By inspection and the disjointness of $p_1$ and $p_2$ we see $p = p_1 \parallel p_2 \xrightarrow{\overline{a}} q$ implies that exactly one of the two cases $p_1 \xrightarrow{\overline{a}} p'_1$, $q = p'_1 \parallel p_2$ or $p_2 \xrightarrow{\overline{a}} p'_2$, $q = p_1 \parallel p'_2$ hold, so the cardinality of $\{q \mid p \xrightarrow{\overline{a}} q\}$ is equal to the cardinality of $\{p' \mid p_1 \xrightarrow{\overline{a}} p'\}$ in the former case and $\{p' \mid p_2 \xrightarrow{\overline{a}} p'\}$ in the latter. By the inductive hypothesis these will be less than or equal to one.

\[\Box\]

Intuitively one can think of the $a.p$ as the process which can be signaled to initiate action $a$ and thereby transforming to $\overline{a}.p$. This term again represents a process which contains an action $a$ signaled to initiate and which can signal it’s completion by transforming by $\overline{a}$ into $p$. The inference rule 3) says that more actions can be signaled to initiate before earlier signaled actions them selves signal there completion. The term $p + q$ represents the process which can act either as $p$ or $q$, and $p \parallel q$ represents the process which can act both as $p$ and as $q$, so actions of one subprocess can be signaled to initiate or complete independent of the other.

**Example:** $a.NIL + b.(a.NIL \parallel b.NIL) \xrightarrow{b} \overline{b}.(a.NIL \parallel b.NIL)$

\[\xrightarrow{a} \overline{b}.(\overline{a}.NILEN \parallel b.NIL)\]

\[\xrightarrow{b} \overline{b}.(\overline{a}.NILEN \parallel \overline{b}.NILEN)\]

\[\xrightarrow{b} \overline{a}.NILEN \parallel \overline{b}.NILEN\]

\[\xrightarrow{b} NILEN \parallel \overline{b}.NILE\]

\[\xrightarrow{b} NILEN \parallel NILE\]

We extend the (atomic) action relation to strings over $Act \cup \overline{Act}$ by:

\[p \xrightarrow{z} p' \text{ iff } \begin{cases} z = \varepsilon, \ p' = p \\
\text{or} \\
z = az', \ p \xrightarrow{a} p'', \ p'' \xrightarrow{z'} p'
\end{cases}\]

where $z \in (Act \cup \overline{Act})^*$.

On the basis of this and the notion of experiment we define how two processes are operational semantically related.

We consider two statements about a process $p$ and an experiment $e$:

- $p$ may accept $e$
- $p$ may reject $e$
The relations can then be defined as follows:

**Definition 3.2.5** For processes \( p, q \in PL \) we define

\[
\begin{align*}
p \preceq_a q & \iff \text{ for all experiments } e: p \text{ may accept } e \implies q \text{ may accept } e \\
p \preceq_r q & \iff \text{ for all experiments } e: p \text{ may reject } e \implies q \text{ may reject } e \\
p \preceq q & \iff p \preceq_a q \text{ and } p \preceq_r q
\end{align*}
\]

The next thing to consider is which experiments we will allow and when a process may accept/ reject an experiment.

An *experiment*, \( e \), will be split out into two. First a set of actions \( A \) are signaled to initiate and second a test \( t \) is done on these. So an experiment, \( e \), is a pair: \((A, t)\).

In fact the signaled set of actions can be considered as a *multiset* over \( Act \) because we want to be able to signal the same action more than once. If \( A \) is a finite multiset over \( Act \) and \( a \in Act \) let \( |A|_a \) denote the number of \( a \)'s in \( A \). For a tree-semiword \( s \) and such a multiset \( A \) we write:

\[
A_s \cong A \iff \forall a \in Act. \, |A|_a = \psi(s, a)
\]

For a multiset \( A \) and a process \( p \in PL \) the set of possible configurations we can obtain by signaling \( A \) is \( D(A, p) \) defined as follows:

**Definition 3.2.6** \( D(A, p) := \{p' \in CL \mid \exists w \in W. A_w \cong A, \, p \xrightarrow{w} p'\} \)

Recall that \( W \) and \( Act^* \) are isomorphic, so it gives sense to write \( p \xrightarrow{w} p' \) for a \( w \in W \). Notice that nondeterministic choices are made when signaling \( A \) to initiate.

The next to decide is the test language \( TL \). We will only allow tests on the actions which are signaled to initiate, so the language must be based on \( Act \). It shall be possible to test the order in which the process can signal completions of the actions previously signaled to initiate, so if \( t \) is a test \( \bar{a}.t \) is a test too. If \( p \) is a configuration the test \( t \land t' \) denotes the test whether *both* the test \( t \land t' \) are possible on \( p \). Similar \( t \triangledown t' \) is the test whether *either* \( t \lor t' \) are possible. A test is ended with \( \top \) to notify that the test was possible.

**Definition 3.2.7** The *test language*, \( TL \), is defined by the schema

\[
t ::= \top \mid \bar{a}.t, \, a \in Act \mid t \land t \mid t \triangledown t
\]
Observe that for a test like \( t \land t' \) or \( t \triangledown t' \) there is no restrictions on the sorts of \( t \) and \( t' \).

\( \top \)—the successful test—is one of the two possible outcomes of a test. The other—the unsuccessful test—is denoted \( \bot \). When having a test like \( t \land t' \) one subtest can turn out to be successful and the other unsuccessful, so during the total test, subconfigurations like \( \top \land \bot \) are possible.

**Definition 3.2.8** The test configurations, \( TC \), are defined by the schema:

\[
o := (t, p), \quad t \in TL, \quad p \in CL \mid \top \mid \bot \mid o \land o \mid o \triangledown o
\]

A test is finished when it is known to be successful or unsuccessful, i.e., when one of the test configurations \( \top \) or \( \bot \) are reached. The relation between the different test configurations is determined by the test relation \( \rightarrow \subseteq TC \times TC \) defined below.

**Definition 3.2.9** Let \( p, p' \in CL, \quad t \in TL, \quad o, o', o'' \in TC \) in the following.

**Axioms:**

1) \( \square : (t \square t', p) \rightarrow (t, p) \square (t', p) \quad \text{for} \quad \square \in \{ \land, \triangledown \} \)

2) \( (\top, p) \rightarrow \top \)

3) \( \land \top : o \land \top \rightarrow o \quad \top \land : \top \land o \rightarrow o \)

\( \land \bot : o \land \bot \rightarrow \bot \quad \bot \land : \bot \land o \rightarrow \bot \)

\( \triangledown \top : o \triangledown \top \rightarrow \top \quad \top \triangledown : \top \triangledown o \rightarrow \top \)

\( \triangledown \bot : o \triangledown \bot \rightarrow o \quad \bot \triangledown : \bot \triangledown o \rightarrow o \)

**Inferences:**

4) \( \square : o \rightarrow o' \quad \text{for} \quad \square \in \{ \land, \triangledown \} \)

\[o \square o'' \rightarrow o' \square o''
\]

\[o'' \square o \rightarrow o'' \square o'\]

5) \( p \xrightarrow{\bar{a}} p', \quad a \in Act\) \( (\bar{a}.t, p) \rightarrow (t, p') \)

6) \( p \xrightarrow{\bar{a}}, \quad a \in Act\) \( (\bar{a}.t, p) \rightarrow \bot \)

\( p \xrightarrow{\bar{a}} \) is just a shorthand notation for \( \exists p' : p \xrightarrow{\bar{a}} p' \).

**Example:**

\[
(\bar{a}.\bar{b}.\top \land \bar{b}.\bar{a}.\top, \bar{a}.\text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL}))
\]

\[
(\bar{a}.\bar{b}.\top, \bar{a}.\text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL})) \land (\bar{b}.\bar{a}.\top, \bar{a}.\text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL}))
\]

\[
(\bar{b}.\top, \text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL})) \land (\bar{b}.\bar{a}.\top, \bar{a}.\text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL}))
\]

\[
(\bar{b}.\top, \text{NIL} \parallel \bar{b}.(\text{c.NIL} + d.\text{NIL})) \land (\bar{a}.\bar{a}.\top, \bar{a}.\text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\bar{b}.\top, \text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\top \land (\bar{a}.\bar{a}.\top, \bar{a}.\text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\top \land \text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\top \land \text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\top \land \text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]

\[
(\top \land \text{NIL} \parallel (\text{c.NIL} + d.\text{NIL}))
\]
Notice that this only is one of many possible derivation that leads $\top$.

A test configuration $o$ is called terminal iff $o \not\rightarrow (i.e., \not\exists o' \in TC. o \rightarrow o')$. In a moment we will show that the only possible terminal test configuration is exactly one of $\top$ and $\bot$, such that a test is either successful or unsuccessful, and cannot be both. In this sense our notion of test is well-defined.

The fact that the terminal configurations are $\{\top, \bot\}$ and that one and only one of these can be reached from a test configuration, $o$, has as consequence:

$$\forall p \in CL \forall t \in TL. (t, p) \rightarrow^* \top \Leftrightarrow (t, p) \not\rightarrow^* \bot$$

An experiment $e$ can now be considered as $(A, t)$, where $A$ is the multiset over $Act$, which determines the actions that should be signaled to initiate, such that a test can be run on them, and $t$ is the actual test to run.

Informally a process $p$ may accept the experiment $e = (A, t)$ if

a) It gives sense to run the test, i.e., the actions of $A$ can be signaled.

b) One of the processes $p'$ obtainable from $p$ by signaling $A$ to initiate, pass the test $t$ successfully.

Similar $p$ may reject $(A, t)$ if under the same conditions as above one of the obtainable processes $p'$ pass the the test $t$ unsuccessfully. Notice that we may have a process $p$ and experiment $e$ such that $p$ may accept $e$ and $p$ may reject $e$! Also notice that the two statements are not dual. I.e., we do not have $p$ may accept $e$ implies $p$ may reject $e$ (where $p$ may accept $e$ means it is not the case that $p$ may accept $e$). This is because the reason why $p$ may accept $e$ can be that it does not make sense to run the test $t$, in which case we have $p$ may reject $e$ too. Formally:

**Definition 3.2.10** Denote the set of experiments by $E$. I.e., $e \in E$ iff $e = (A, t)$, $A$ is a finite multiset over $Act$ and $t \in TL$.

Let $p \in PL$ and $e = (A, t)$ be an experiment. Then:

a) $p$ may accept $e$ iff $\exists q \in D(A, p). (t, q) \rightarrow^* \top$

b) $p$ may reject $e$ iff $\exists q \in D(A, p). (t, q) \rightarrow^* \bot$

Example:

$a.NIL + b.a.NIL + a.NIL \parallel b.NIL$ may accept $(\{a, b\}, \bar{a}.\bar{b}.\top \& \bar{b}.\bar{a}.\top)$

$a.b.NIL + b.a.NIL$ may accept $(\{a, b\}, \bar{a}.\bar{b}.\top \& \bar{b}.\bar{a}.\top)$

$a.b.NIL + b.a.NIL + a.NIL \parallel b.NIL$ may reject $(\{a, b\}, \bar{a}.\bar{b}.\top \nabla (\bar{b}.\bar{a}.\top \& \bar{a}.\bar{b}.\top)$)

$a.NIL \parallel b.NIL$ may reject $(\{a, b\}, \bar{a}.\bar{b}.\top \nabla (\bar{b}.\bar{a}.\top \& \bar{a}.\bar{b}.\top)$)

69
So we have now formally defined what was used in the definition of the three testing preorders \(\mathcal{E}_a\), \(\mathcal{E}_r\) and \(\mathcal{E}\) on \(PL\). In the following \(\equiv\) shall denote the equivalence of \(\mathcal{E}\). Similar for the other preorders.

**Example:**

\[
\begin{align*}
 a.b.NIL + b.a.NIL &\not\equiv a.b.NIL + b.a.NIL + a.NIL \parallel b.NIL \\
 a.NIL &\not\equiv a.b.NIL \\
 a.b.c.NIL + a.(b.NIL \parallel c.NIL) + a.NIL \parallel b.c.NIL &\equiv a.b.c.NIL + a.NIL \parallel b.c.NIL
\end{align*}
\]

As indicated by:

\[
\begin{align*}
 a.(b.NIL + c.NIL) &\not\equiv a.b.NIL + a.c.NIL
\end{align*}
\]

non of the equivalences are able to distinguish nondeterminism.

That \(p\) may accept \(e\) and \(p\) may reject \(e\) are not dual can now also formally easily be seen:

\[
\neg(\exists q \in D(A,p). (t,q) \rightarrow^{*} \top) \iff \forall q \in D(A,p)(t,q) \not\Rightarrow^{*} \top
\]

\[
\iff \forall q \in D(A,p). (t,q) \rightarrow^{*} \bot
\]

\[
\iff \exists q \in D(A,p). (t,q) \rightarrow^{*} \bot
\]

Before we give the promised proof of one and only one terminal configuration for every test configuration we prove the following lemma, which also clears the rôle of \(\&\) and \(\triangleright\).

**Lemma 3.2.11**

a) \(o_1 \& o_2 \rightarrow^{*} \top \iff o_1 \rightarrow^{*} \top \text{ and } o_2 \rightarrow^{*} \top\)

b) \(o_1 \& o_2 \rightarrow^{*} \bot \iff o_1 \rightarrow^{*} \bot \text{ or } o_2 \rightarrow^{*} \bot\)

c) \(o_1 \triangleright o_2 \rightarrow^{*} \top \iff o_1 \rightarrow^{*} \top \text{ or } o_2 \rightarrow^{*} \top\)

d) \(o_1 \triangleright o_2 \rightarrow^{*} \bot \iff o_1 \rightarrow^{*} \bot \text{ and } o_2 \rightarrow^{*} \bot\)

**Proof**

a) *only if:* We prove it by proving \(o_1 \& o_2 \rightarrow^{n} \top\) only if \(o_1 \rightarrow^{*} \top \land o_2 \rightarrow^{*} \top\) by induction on \(n\) (\(n = 0\) impossible because \(\top\) is not of the form \(t \& t'\)).

\(n = 1:\) By inspection of definition 3.2.9 we see \(o_1 \& o_2 \rightarrow \top\) implies \(o_1 = \top = o_2\). Then of course also \(o_1 \rightarrow^{*} \top\) and \(o_2 \rightarrow^{*} \top\).

\(n > 1:\) \(o_1 \& o_2 \rightarrow o, o \rightarrow^{n-1} \top\). Looking at the definition again we see that only 3) \(\& \top, \top \&\) or 4) \(\&\) can come under discussion for the move \(o_1 \& o_2 \rightarrow o\).
3) & $= \top$, $\vee$: W.l.o.g. assume $o_1 \& o_2 \to o$ is $o_1 \& \top \to o_1$. Then clearly $o_2 \to^* \top$ and $o_1 \to^{n-1} \top$ which implies $o_1 \to^* \top$.

4) $\&$: Two cases: $o_1 \to o'_1$, $o = o'_1 \& o_2$ and $o_2 \to o'_2$, $o = o_1 \& o'_2$.

For the former then $o \to^{n-1} \top$ means $o'_1 \& o_2 \to^{n-1} \top$. By hypothesis of induction $o'_1 \to^* \top \land o_2 \to^* \top$. $o_1 \to o'_1, o'_1 \to^* \top$ implies $o_1 \to^* \top$, so we are done for this case.

The latter is handled symmetric. This also completes the inductive step.

If: It is enough to prove $o_1 \to^n \top \land o_2 \to^m \top$ implies $o_1 \& o_2 \to^{n+m} \top$ by induction on $n + m$.

$n + m = 0$: Then $o_1 = \top = o_2$. No matter whether we use 3) & $= \top$ or 3) $\top \&$ we get $o_1 \& o_2 \to \top$ and thereby $o_1 \& o_2 \to^* \top$.

$n + m > 0$: We split out in two subcases:

$m > 0$: This implies $o_2 \to o'_2 \to^{m-1} \top$. By hypothesis of induction $o_1 \& o'_2 \to^* \top$. Using 4) $\&$ we now get $o_1 \& o_2 \to o_1 \& o'_2$, hence $o_1 \& o_2 \to^* \top$.

$n > 0$: Then $o_1 \to o'_1 \to^{n-1} \top$. Similar to the case $n = 0$.

b) -d): Similar to a).

Now for an $o \in TC$ let $B(o)$ denote the set of terminal configurations i.e.,

$$B(o) := \{ o' \in TC \mid o \to^* o', o' \neq \}$$

So what we shall prove is that $\{ \top, \bot \}$ equals the terminal configurations and $|B(o)| = 1$ which follows from the proposition below.

**Proposition 3.2.12** Let $o \in TC$. Then

a) $\{ \top, \bot \} = \text{terminal configurations}$  

b) $o \to^* \top \lor o \to^* \bot$

where $\lor$ means that exactly one of the possibilities are true.

**Proof**

a) By inspection of definition 3.2.9 we easily see that $\top$ and $\bot$ are the only possible terminal configurations.

b) We split the proof out in two, i) $o \to^* \top \lor o \to^* \bot$ and ii) $o \to^* o', o' \in \{ \top, \bot \} \Rightarrow B(o) = \{ o' \}$ from which b) can be seen.

i) We prove $o \to^* \top \lor o \to^* \bot$ by induction on the structure of $o$.

- $o = \top$ or $o = \bot$ (basis): Immediate.

- $o = o_1 \& o_2$: By hypothesis of induction $o_i \to^* \top \lor o_i \to^* \bot$ for $i \in \{2\}$. Four cases:
  
  - $o_1 \to^* \top, o_2 \to^* \top$: Using the if part of lemma 3.2.11 a) we get $o = o_1 \& o_2 \to^* \top$.
  
  - In the three other cases we get $o \to^* \bot$ using the if part of b) in the lemma.

- $o = o_1 \nabla o_2$: Similar.

- $o = (t, p), t \in TL, P \in CL$: To prove this part of the inductive step we use induction on the structure of $t$.  

71
t = \top: 2) of definition 3.2.9 gives us \( o = (t, p) = (\top, p) \rightarrow \top \) hence \( o \rightarrow^* \top \).

\( t = a.t' \): Clearly either 5) and 6) can be used. In the latter case we directly get \( o \rightarrow^* \perp \). In the former we have \((a.t', p) \rightarrow (t', p')\). By hypothesis of induction either \((t', p') \rightarrow^* \top \) or \((t', p') \rightarrow^* \perp \) and the result follows.

\( t = t_1 \& t_2 \): Using 1)& we get \( o = (t_1 \& t_2, p) \rightarrow (t_1, p) \& (t_2, p) \). By the hypothesis of induction \((t_1, p) \rightarrow^* \top \lor (t_2, p) \rightarrow^* \perp \) for \( i \in 2 \). Similar as in the case \( o = o_1 \& o_2 \) we get \((t_1, p) \& (t_2, p) \rightarrow^* \top \lor (t_1, p) \& (t_2, p) \rightarrow^* \perp \). Hence also \( o \rightarrow^* \top \lor o \rightarrow^* \perp \).

\( t = t_1 \lor t_2 \): Similar.

ii) Since \( o' \in \{\top, \perp\} \) we can part the proof in two:

\( o' = \top \): Then we have to prove \( o \rightarrow^* \top \Rightarrow B(o) = \{\top\} \). We will do this by induction on the structure of \( o \). Since \( \top \not\rightarrow^* \top \) and \( o \rightarrow^* \top \) implies \( \top \in B(o) \) we only need to prove \( o \rightarrow^* \top \Rightarrow B(o) \subseteq \{\top\} \).

\( o = \top \) or \( o = \perp \) (basis): Looking at definition 3.2.9 we see \( \perp \not\rightarrow^* \top \) and \( B(\top) = \{\top\} \).

\( o = o_1 \& o_2 \): By the only if part of lemma 3.2.11 we have \( o_1 \rightarrow^* \top \) and \( o_2 \rightarrow^* \top \). By hypothesis of induction we then have \( B(o_1) = \{\top\} \) and \( B(o_2) = \{\top\} \). Now assume \( B(o) \not\subseteq \{\top\} \). Since \( \top \) and \( \perp \) are the only terminal configurations (by \( a \)) this implies \( o = o_1 \& o_2 \rightarrow^* \perp \) and by the only if part of lemma 3.2.11.b) we have \( o_1 \rightarrow^* \perp \) or \( o_2 \rightarrow^* \perp \), which contradicts \( B(o_1) = \{\top\} \) and \( B(o_2) = \{\top\} \). So \( B(o) \not\subseteq \{\top\} \).

\( o = o_1 \lor o_2 \): Symmetric.

\( o = (t, p), t \in TL, p \in CL \): This part of the inductive step is also proved by structural induction, but this time on the structure of \( t \).

\( t = \top \): Looking at definition 3.2.9 we see that only 2) can be used, hence \( B(o) = B(\top) = \{\top\} \).

\( t = a.t' \): Clearly only 5) or 6) can come under discussion. Two cases depending on \( p \).

\( \exists q, p \rightarrow a q \): By proposition 3.2.4 there is at most one such \( q \). Hence \( B(o) = B((t', q)) \). By hypothesis of induction \( B((t', q)) \subseteq \{\top\} \).

\( \not\exists q, p \rightarrow a q \): This case can be excluded since \( o \rightarrow \perp \) is the only possibility, \( \perp \not\rightarrow^* \top \) and we assume \( o \rightarrow^* \top \).

\( t = t_1 \& t_2 \): The only possibility is \( o = (t_1 \& t_2, p) \rightarrow (t_1, p) \& (t_2, p) = o' \rightarrow^* \top \) and \( B(o) = B(o') \). Similar as in the case \( o = o_1 \& o_2 \) we get \( B(o') \subseteq \{\top\} \).

\( t = t_1 \lor t_2 \): Symmetric.

\( o' = \perp \): Similar as the case \( o' = \top \), but now \( t = \top \) can be excluded and in \( t = a.t' \), \( \not\exists q, p \rightarrow a q \) no longer can be ignored. In this last case we get \( o = (a.t, p) \rightarrow \perp \) and \( B(o) = B(\perp) = \{\perp\} \), which is wanted.

\( \square \)

In the following we will investigate other properties of our test language which will be useful in the following sections.

The most important property is that every test \( t \) has a normal form which we define in a moment. On the way to show this we introduce some notation.

72
Definition 3.2.13 For \( t, t' \in TL \) we let \( t \equiv t' \) denote:

\[
\forall p \in CL. B((t, p)) = B((t', p))
\]

or equivalently by the last proposition:

\[
\forall p \in CL. (t, p) \rightarrow^* \top \iff (t', p) \rightarrow^* \top
\]

\[ \square \]

Proposition 3.2.14 \( \equiv \) is an equivalence relation on \( TL \) such that for all \( t, t', t'' \in TL \):

\[
\begin{align*}
\text{a)} & \quad t \& t' \equiv t' \& t \quad & \text{b)} & \quad t \triangledown t' \equiv t' \triangledown t \\
\text{c)} & \quad t \& (t' \& t'') \equiv (t \& t') \& t'' \\
\text{d)} & \quad t \triangledown (t' \triangledown t'') \equiv (t \triangledown t') \triangledown t'' \\
\text{e)} & \quad (t \triangledown t') \triangledown (t \triangledown t'') \\
\text{f)} & \quad (\vec{a}, (t \& t')) \equiv (\vec{a}t \& \vec{a}t') \\
\text{g)} & \quad (\vec{a}, (t \triangledown t')) \equiv \vec{a}t \triangledown \vec{a}t' \\
\text{h)} & \quad t \equiv t' \implies \begin{cases} \vec{a}, & \vec{a}t \equiv \vec{a}t' \\ & \& t \& t'' \equiv t' \& t'' \\ & (\triangledown) & t \triangledown t'' \equiv t' \triangledown t'' \end{cases}
\end{align*}
\]

Proof

That \( \equiv \) is an equivalence relation is immediate from the definition.

a) – e) follows by lemma 3.2.11 from the similar properties of \( \land \) and \( \lor \). This is not the case with f) – h).

f) We shall prove \((\vec{a}, (t \& t'), p) \rightarrow^* \top \iff (\vec{a}t \& \vec{a}t', p) \rightarrow^* \top\).

if: By definition 3.2.9 \((\vec{a}t \& \vec{a}t', p) \rightarrow^* \top\) implies \((\vec{a}, t, p) \& (\vec{a}t', p) \rightarrow^* \top\) which by lemma 3.2.11 implies \((\vec{a}t, p) \rightarrow^* \top\) and \((\vec{a}t', p) \rightarrow^* \top\). Now \((\vec{a}t, p) \rightarrow^* \top\) implies \((\vec{a}t', p) \rightarrow (t', p') \rightarrow^* \top\) where \(p \xrightarrow{\vec{a}} p'\). Similar \((\vec{a}, t', p) \rightarrow (t', p') \rightarrow^* \top\), where \(p \xrightarrow{\vec{a}} p'\). By proposition 3.2.4 we must have \(p' = p''\). Hence \((t, p') \rightarrow^* \top\), \((t', p') \rightarrow^* \top\) and \(p \xrightarrow{\vec{a}} p'\). Using lemma 3.2.11 again we get \((t, p') \& (t, p') \rightarrow^* \top\). From this and \(p \xrightarrow{\vec{a}} p'\), 5), 1) & of definition 3.2.9 we see \((\vec{a}, (t \& t'), p) \rightarrow (t \& t', p') \rightarrow (t, p') \& (t, p') \rightarrow^* \top\).

only if: By inspection of definition 3.2.9 we see \((\vec{a}, (t \& t'), p) \rightarrow^* \top\) implies \((\vec{a}, t \& t'), p) \rightarrow (t \& t'), p' \rightarrow (t, p') \& (t, p') \rightarrow^* \top\), where \(p \xrightarrow{\vec{a}} p'\). From this definition 3.2.9 directly gives us: \((\vec{a}t \& \vec{a}t', p') \rightarrow (\vec{a}t, p) \& (\vec{a}t', p) \rightarrow (t, p') \& (t, p') \rightarrow^* \top\).

\(\text{g)}\) Similar, but here proposition 3.2.4 is not necessary!

h) Assume \( t \equiv t' \).

\(\vec{a}.\): Let \( p \in PL, \vec{a} \in Act \) be given. Shall show that \((\vec{a}t, p) \rightarrow^* \top \iff (\vec{a}t', p) \rightarrow^* \top\). This is evident from proposition 3.2.4.

\&): Let a \( p \in CL \) and \( t'' \in TL \) be given. Shall show \((t \& t''', p) \rightarrow^* \top \iff (t \& t''', p) \rightarrow^* \top\).

if: \((t \& t''', p) \rightarrow^* \top\) implies \((t \& t''', p) \rightarrow (t', p) \rightarrow^* \top\). By lemma 3.2.11
this implies $(t', p) \rightarrow^* \top$ and $(t'', p) \rightarrow^* \top$. From $t \cong t'$ we then have $(t, p) \rightarrow^* \top$. Reversing the arguments we obtain $(t \& t'', p) \rightarrow^* \top$.

\(\nabla\): Similar.

\[\nabla\) Definition 3.2.15 \(t \in TL\) is a test normal form iff \(t\) is of the form
\[
\&_{j \in \mathbb{N}} \left( \nabla \right. \left. k \in n_j \overline{w_{jk} \top} \right), w_{jk} \in W \text{ for } k \in n_j, 1 \leq n_j \text{ with } j \in n, 1 \leq n
\]

where \(\overline{w}\) simply is the string \(w\) with every \(a\) in \(w\) exchanged with \(\bar{a}\). □

By a) – d) of proposition 3.2.14 it gives sense to use the notational convenience of \(\&\) and \(\nabla\) in the definition of a normal form.

\[\nabla\) Proposition 3.2.16 For every \(t \in TL\) there is a normal form \(t' \in TL\) such that \(t \cong t'\).

\[\nabla\) Proof We can transform \(t\) into a normal form \(t'\) by using a) – h) of proposition 3.2.14. At first we use e) and f) to get all \&’s of \(t\) out on the outmost level such that we obtain a \(t'' = \&_{j \in \mathbb{N}} t_j\), where \(t_j\) is built from \(\nabla, \bar{a} \in \text{Act}\) and \(\top\) for \(j \in \mathbb{N}\). Then for every \(j \in \mathbb{N}\) transform \(t_j\) by g) into \(t'_j = \nabla_{k \in n_j} w_{jk} \top\). By the congruence h) it should be clear that \(t' = \&_{j \in \mathbb{N}} t'_j \cong t\). □

By inspection of definition 3.2.9 the following corollary is evident.

\[\nabla\) Corollary 3.2.17 For all \(p \in CL, w \in W\) we have:
\[
\exists q. \ p \xrightarrow{\bar{w}} q \text{ iff } \bar{w} \top \in TL, (\bar{w} \top, p) \rightarrow^* \top
\]

\[\nabla\) Proposition 3.2.18 Let \(t\) be on the normal form: \(\&_{j \in \mathbb{N}} (\nabla_{k \in n_j} \bar{w}_{jk} \top)\). Then for all \(p \in CL:\)
\[
(t, p) \rightarrow^* \top \text{ iff } \forall j \in \mathbb{N} \exists k \in n_j. (\bar{w}_{jk} \top, p) \rightarrow^* \top
\]

\[\nabla\) Proof Follows immediately from lemma 3.2.11. □

\[\nabla\) 3.3 Full Abstractness

The aim of this section is to show that the denotational and operational semantics corresponds or rather that \(\llbracket s, \llbracket v\) and \(\llbracket \chi\) are fully abstract w.r.t. \(\llbracket a, \llbracket r\) and \(\llbracket\) respectively. Formally we want to prove:
Theorem 3.3.1  Operational Characterization Theorem If \( p, q \in PL \) then \( \preceq_a, \preceq_r \) and \( \preceq \) are (relative) precongruences and

\[
\begin{align*}
\delta) \quad [p]_\delta \not\subseteq [q]_\delta & \iff p \preceq_a q \\
\nu) \quad [p]_\nu \not\subseteq [q]_\nu & \iff p \preceq_r q \\
\chi) \quad [p]_\chi \not\subseteq [q]_\chi & \iff p \preceq q
\end{align*}
\]

As for Hennessy [Hen85a] it will prove convenient to introduce some more plain relations, \( \ll_* \), on processes from \( PL \), which are entirely defined on basis of the lts, and which coincide with the three testing preorders.

The rest of this section is devoted to definitions and intermediate results necessary to prove the Semantic Characterization Theorem of these relations and the Operational Characterization Theorem.

At first we define a map, \( \bar{\theta} \), on configurations which associates in a natural way a tree-semiword with the “barred” part of a configuration. I.e., for a configuration, \( p, \bar{\theta}(p) \) gives a tree-semiword which reflects the causal order in which initiated actions can signal there completion.

Definition 3.3.2

a) Let \( \bar{\theta} : CL \rightarrow TSW \) be defined inductively as follows:

\[
\begin{align*}
p & \mapsto \varepsilon & \text{if } p \in PL \\
\bar{a}.p & \mapsto a.\bar{\theta}(p) \\
p \parallel q & \mapsto \bar{\theta}(p) \parallel \bar{\theta}(q) & \text{if either } p \not\in PL \text{ or } q \not\in PL
\end{align*}
\]

b) Let \( \bar{\Theta} : PL \rightarrow \mathcal{P}(TSW) \) be defined by: \( \bar{\Theta}(p) := \{\bar{\theta}(q) \mid \exists w \in W. p \xrightarrow{w} q\} \)

For arbitrary sets of semiwords we use the following notation:

\[
\begin{align*}
S \ll_a T & \iff \forall s \in S \exists t \in T. s \leq t \\
S \ll_r T & \iff \forall s \in S \exists t \in T. t \leq s \\
S < T & \iff S \ll_a T \text{ and } S \ll_r T
\end{align*}
\]

We can now formulate the three alternative preorders.

Definition 3.3.3 Let \( p, q \in PL \). Then \( \ll_a, \ll_r \) and \( \ll \) are defined as follows:

\[
p \ll_* q \iff \bar{\theta}(p) \ll_* \bar{\theta}(q)
\]

where * as usual is either left out or one of \( a, r \).

In the future we will mostly omit the comment about *.
Theorem 3.3.4 Semantic Characterization Theorem For all $p, q$ in $PL$:

$$p \subseteq q \iff p \ll_* q$$

The first step in the prove of this theorem is a rewriting of $p \subseteq q$.

Lemma 3.3.5 For $* = a \ (*=r)$ and $o_a = \top \ (o_r = \bot)$ we have:

$$p \subseteq_* q \iff \forall (A, t) \forall p' \in D(A, p) \exists q' \in D(A, q). (t, p') \rightarrow^* o_a \Rightarrow (t, q') \rightarrow^* o_*$$

Proof We only prove the case $* = a$, since the case $* = r$ follows in exactly the same way.

We prove $p \subseteq_a q \iff a$ by proving $p \not\subseteq_a q \iff \neg a$.

$p \not\subseteq_a q$ (by definition)

$$\neg (\forall (A, t) (\exists p' \in D(A, p). (t, p') \rightarrow^* \top) \Rightarrow (\exists q' \in D(A, q). (t, q') \rightarrow^* \top)) \iff$$

$$\exists (A, t) (\exists p' \in D(A, p). (t, p') \rightarrow^* \top) \land (\forall q' \in D(A, q). (t, q') \rightarrow^* \bot) \iff$$

$$\exists (A, t) \exists p' \in D(A, p) \forall q' \in D(A, q). (t, p') \rightarrow^* \top \land (t, q') \rightarrow^* \bot \iff$$

$$\neg a).$$

In these derivations we used $(t, p) \rightarrow^* \top$ iff $(t, p) \not\rightarrow^* \bot$ which was a consequence of proposition 3.2.12.

The next step is to prove that the $\forall t$-quantifier can be moved past $\exists q' \in D(A, q)$ in $*$) of the last lemma.

Lemma 3.3.6 For $* = a \ (*=r)$ and $o_a = \top \ (o_r = \bot)$ we have:

i) $\forall (A, t) \forall p' \in D(A, p) \exists q' \in D(A, q). (t, p') \rightarrow^* o_a \Rightarrow (t, q') \rightarrow^* o_*$

iff

ii) $\forall A \forall p' \in D(A, p) \exists q' \in D(A, q) \forall t'. (t', p') \rightarrow^* o_a \Rightarrow (t', q') \rightarrow^* o_*$

Proof We prove $\neg i)_* \iff \neg ii)_*$ for the two cases of $*$. I.e.,

$$\neg i)_* \exists (A, t) \exists p' \in D(A, p) \forall q' \in D(A, q). (t, p') \rightarrow^* o_a \land (t, q') \not\rightarrow^* o_*$$

iff

$$\neg ii)_* \exists A \exists p' \in D(A, p) \forall q' \in D(A, q) \exists t'. (t', p') \rightarrow^* o_a \land (t', q') \not\rightarrow^* o_*$$

If $D(A, q) = \emptyset$ the result is trivial, so assume $D(A, q) \neq \emptyset$ in the following.
Lemma 3.3.7

Let \( \text{Act} \cup \overline{\text{Act}} \). For \( p_1, p_2 \in CL \) we have for \( z \in Z \):

\[
p_1 \parallel p_2 \xrightarrow{z} q \quad \text{iff} \quad \exists z_i \in Z, q_i, p_i \xrightarrow{z_i} q_i \quad \text{for} \ i \in 2 \quad \text{and} \quad q = q_1 \parallel q_2, z \preceq z_1 \parallel z_2
\]

Notice that \( W \subseteq Z \) and \( \overline{W} \subseteq Z \), but \( Z \not\subseteq W \cup \overline{W} \).

**Proof**

Both of the implications in this lemma is proved by induction on the length of \( z \).

- if: \( z = \varepsilon \): Then \( z = \varepsilon \preceq z_1 \parallel z_2 \) implies \( z_1 = \varepsilon = z_2 \), so \( q_i = p_i \) for \( i \in 2 \) and \( q = q_1 \parallel q_2 \). By definition \( p_1 \parallel p_2 \xrightarrow{\varepsilon} p_1 \parallel p_2 = q \), so ok.

- if: \( z \neq \varepsilon \): Then \( z = a.z' \) for some \( a \in \text{Act} \cup \overline{\text{Act}} \) and \( z' \in Z \). By proposition 1.3.31 this implies \( \exists z'_1, a.z'_1 = z_1, z' \preceq z'_1 \parallel z_2 \) or \( \exists z'_2, a.z'_2 = z_2, z' \preceq z'_1 \parallel z'_2 \). W.l.o.g. assume the former is true. Then \( p_1 \xrightarrow{z} q_1 \) is the same as \( p_1 \xrightarrow{a.z'_1} q_1 \), so there exists some \( p'_1 \) such that \( p_1 \xrightarrow{a} p'_1 \xrightarrow{z'_1} q_1 \). Since \( |z'| < |z| \) we can use the inductive hypothesis to get \( p_1 \parallel p_2 \xrightarrow{z'} q_1 \parallel q_2 = q \). From the inference rule 5) of definition 3.2.2 we obtain \( p_1 \parallel p_2 \xrightarrow{a} p'_1 \parallel p_2 \) from \( p_1 \xrightarrow{a} p'_1 \), so \( p_1 \parallel p_2 \xrightarrow{a.z'} q \) or equivalently \( p_1 \parallel p_2 \xrightarrow{z} q \).

Only if:

- if: \( z = \varepsilon \): Then \( q = q_1 \parallel p_2 \) and the result should be clear.

- if: \( z \neq \varepsilon \): Then \( z = a.z' \) for some \( a \in \text{Act} \cup \overline{\text{Act}} \) and \( p_1 \parallel p_2 \xrightarrow{a} q' \xrightarrow{z'} q \) for some \( q' \in CL \). Looking at definition 3.2.2 we see that 5) must have been used to ensure \( p_1 \parallel p_2 \xrightarrow{a} q' \). W.l.o.g. assume this is obtained from \( p_1 \xrightarrow{a} p'_1 \) such that \( q' = p'_1 \parallel p_2 \). By
Lemma 3.3.8 For all $p \in CL$ and $w \in W$ we have:

$$w \in \pi \lambda \bar{\theta}(p) \iff \exists p'. p \xrightarrow{\bar{w}} p'$$

**Proof** We will prove $w \in \pi \lambda \bar{\theta}(p) \iff \exists p'. p \xrightarrow{\bar{w}} p'$ by induction on the structure of $p$.

- $p \in PL$: We split this case out in two depending of whether $w = \varepsilon$ or not.

  $w = \varepsilon$: Since $p \in PL$ we have $\bar{\theta}(p) = \varepsilon$, so $w = \varepsilon \in \{\varepsilon\} = \lambda(\varepsilon) = \pi \lambda(\varepsilon) = \pi \lambda \bar{\theta}(p)$.

  Also $\exists p'. p \xrightarrow{\varepsilon} p'$, since $\varepsilon = \varepsilon$ and $p \xrightarrow{\varepsilon} p$ by definition.

  $w \neq \varepsilon$: We neither have a $w' \in \pi \lambda \bar{\theta}(p)$ with $w' \neq \varepsilon$ nor any $p'$ such that $p \xrightarrow{\bar{w}} p'$, when $p \in PL$ and $w \neq \varepsilon$.

  To see the latter assume on the contrary $\exists p'. p \xrightarrow{\bar{w}} p'$, $w \neq \varepsilon$ implies $w = a.w'$ for some $w' \in W$ and $a \in Act$. So $\bar{w} = \bar{a}.w'$. Then $p \xrightarrow{\bar{w}} p'$ can be written $p \xrightarrow{\bar{a}.w'} p'$ and implies $p \xrightarrow{\bar{a}} p'' \xrightarrow{\bar{w}'} p'$ for some $p'' \in CL$. But it is easily seen that $p \in PL$, $p \xrightarrow{\bar{y}}$ implies $y \in Act$, which contradicts $p \xrightarrow{\bar{a}} p''$, $a \in Act$.

  The former is seen from $\pi \lambda \bar{\theta}(p) = \{\varepsilon\}$ as shown in the case $w = \varepsilon$.

- $p = \bar{a}.p''$, $a \in Act$: At first notice $\pi \lambda \bar{\theta}(p) = \pi \lambda \bar{\theta}(\bar{a}.p'') = (\text{by definition of } \bar{\theta}) \pi \lambda a.\bar{\theta}(p'') = (\text{deduced from proposition}_T \text{1.3.13} \pi a.\lambda \bar{\theta}(p'') = (\text{corollary}_T \text{1.3.36}) a.\pi \lambda \bar{\theta}(p'') \cup \{\varepsilon\}$.

  We show the implications separately.

  $\Rightarrow$: $w \in \pi \lambda \bar{\theta}(p)$ implies $w \in a.\pi \lambda(p'')$ or $w = \varepsilon$ which again implies $w = a.w'$ or $w = \varepsilon$, where $w' \in \pi \lambda \bar{\theta}(p'')$. By definition $p \xrightarrow{\varepsilon} p = p'$ which handles the former case. In the latter we use the hypothesis of induction to get $\exists p'. p'' \xrightarrow{\bar{w}} p'$. Using 3) of definition 3.2.2 we have $p = a.p'' \xrightarrow{\bar{a}} p'$. Hence $p \xrightarrow{\bar{w}} p'$.

  $\Leftarrow$: Looking at definition 3.2.2 we see that $p = \bar{a}.p'' \xrightarrow{\bar{w}} p'$ implies $\bar{w} = \varepsilon$ or $\bar{w} = \bar{a}.w'$ for some $w'$ such that $p'' \xrightarrow{\bar{w}'} p'$. If $\bar{w} = \varepsilon$ we have $w = \varepsilon = \varepsilon$, so $w \in a.\pi \lambda \bar{\theta}(p'') \cup \{\varepsilon\} = \pi \lambda \bar{\theta}(p)$. In the other case $\bar{w} = \bar{a}.w'$, the hypothesis of induction gives us $w' \in \pi \lambda \bar{\theta}(p'')$, so $w = a.w' \in a.\pi \lambda \bar{\theta}(p'') \cup \{\varepsilon\} = \pi \lambda \bar{\theta}(p)$.

$p = p_1 \parallel p_2$: At first notice that by lemma 3.3.7 we have:

$$p_1 \parallel p_2 \xrightarrow{\bar{w}} p' \iff \exists p_1', p_2' \in CL, \bar{w}_1, \bar{w}_2 \in \overline{W} \text{ such that } p_1 \xrightarrow{\bar{w}_i} p_i' \text{ for } i \in 2, \bar{w} \preceq \bar{w}_i \parallel \bar{w}_2, p = p_1' \parallel p_2'. $$

Clearly we also have $\bar{w} \preceq \bar{w}_1 \parallel \bar{w}_2 \iff w \preceq w_1 \parallel w_2$. The two implications:

$\Rightarrow$: We have $\pi \lambda \bar{\theta}(p_1 \parallel p_2) = (\text{by proposition}_T \text{1.3.38 and definition of } \bar{\theta}) \lambda(\pi \bar{\theta}(p_1)) \parallel \bar{\theta}(p_2)) = \lambda(\pi \bar{\theta}(p_1) \parallel \pi \bar{\theta}(p_2))$, so using proposition $T \text{1.3.13}$ we get $w \in \pi \lambda \bar{\theta}(p)$ implies $\exists w_i \in \lambda \pi \bar{\theta}(p_1), i \in 2$ such that $w \preceq w_1 \parallel w_2$. Since $\lambda \pi \bar{\theta}(p_1) = \pi \lambda \bar{\theta}(p_1)$ for $i \in 2$ we can use the hypothesis to get $\exists p_i', p_i \xrightarrow{\bar{w}'} p_i'$ for $i \in 2$. Then, as noticed, $p = p_1 \parallel p_2 \xrightarrow{\bar{w}} p'_1 \parallel p'_2 = p'$.
We see above that

\[ (t;p_k) \Rightarrow \text{from proposition } T 1.3.13 \]

\[ w \in \lambda(\pi\hat{\theta}(p_1) \parallel \pi\hat{\theta}(p_2)) \]

which, as seen above, is the same as \( w \in \pi\lambda\hat{\theta}(p) \).

\[ \square \]

**Lemma 3.3.9** For \( p, q \in CL \) we have

\[ \hat{\theta}(p) \leq \hat{\theta}(q) \text{ iff } A_{\hat{\theta}(p)} = A_{\hat{\theta}(q)} \text{ and } \forall t. (t,p) \rightarrow^* \top \Rightarrow (t,q) \rightarrow^* \top \]

**Proof**

If: Assume \( A_{\hat{\theta}(p)} = A_{\hat{\theta}(q)} \) and \( \forall t. (t,p) \rightarrow^* \top \Rightarrow (t,q) \rightarrow^* \top \). We shall prove \( \hat{\theta}(p) \leq \hat{\theta}(q) \). By proposition \( T 1.3.5 \) it is enough to prove \( \lambda\hat{\theta}(p) \subseteq \lambda\hat{\theta}(q) \). Let \( w \in \lambda\hat{\theta}(p) \) be given. Then also \( A_w = A_{\hat{\theta}(p)} \) and \( w \in \pi\lambda\hat{\theta}(p) \), \( w \in W \), so by lemma 3.3.8 \( \exists p'. p \rightarrow^i p' \) and from corollary 3.2.17 \( w \top \in TL, (w\top, p) \rightarrow^* \top \). By the assumption then \( (w\top, q) \rightarrow^* \top \). Using the same lemmas in the opposite direction we get \( w \in \pi\lambda\hat{\theta}(q) \). We cannot conclude \( w \in \lambda\hat{\theta}(q) \) directly. Assume on the contrary \( w \) is a proper prefix of some \( w' \in \lambda\hat{\theta}(q) \). Then \( A_w \subset A_{w'} \). In general \( \forall s \in \lambda(t) \). Assume \( A_w = A_t \), so \( A_{\hat{\theta}(p)} = A_w \subset A_{w'} = A_{\hat{\theta}(q)} \) which contradicts the assumption \( A_{\hat{\theta}(p)} = A_{\hat{\theta}(q)} \). Hence \( w \in \lambda\hat{\theta}(q) \).

Only if: Assume \( \hat{\theta}(p) \leq \hat{\theta}(q) \). By definition \( A_{\hat{\theta}(p)} = A_{\hat{\theta}(q)} \). Let \( t \in TL \) be given such that \( (t,p) \rightarrow^* \top \). We shall prove \( (t,q) \rightarrow^* \top \). By proposition 3.2.16 \( t \) can be chosen to be on normal form. I.e.,

\[ t = \bigvee_{j \in \mathbb{N}} \bigwedge_{k \in n_j} \bar{w}_{jk} \top, w_{jk} \in W \text{ for } k \in n_j \text{ and } j \in \mathbb{N} \]

Then by proposition 3.2.18 \( (t,p) \rightarrow^* \top \) implies \( \forall j \in \mathbb{N} \exists k \in n_j. (\bar{w}_{jk} \top, p) \rightarrow^* \top \) and by corollary 3.2.17 \( \forall j \in \mathbb{N} \exists k \in n_j. \exists p_{jk}. p \rightarrow^i p_{jk} \). By lemma 3.3.8 \( w_{jk} \in \pi\lambda\hat{\theta}(p) \) for \( j \in \mathbb{N}, k \in n_j \). Now \( \hat{\theta}(p) \leq \hat{\theta}(q) \) \( \Rightarrow \) (by proposition \( T 1.3.5) \lambda\hat{\theta}(p) \subseteq \lambda\hat{\theta}(q) \Rightarrow \pi\lambda\hat{\theta}(p) \subseteq \pi\lambda\hat{\theta}(q) \), so \( w_{jk} \in \pi\lambda\hat{\theta}(q) \) and using lemma 3.3.8 again \( \exists q_{jk}. q \rightarrow^i q_{jk} \). Reversing the arguments from above we get \( (t,q) \rightarrow^* \top \).

\[ \square \]

**Lemma 3.3.10** For all \( p \in PL \) and \( w \in W \) we have \( p \rightarrow^w q \) implies \( w \in \lambda\hat{\theta}(q) \).

For the proof of the lemma we shall temporarily assume to work with semiwords and not just tree-semiwords.

**Proof** Actually we prove the stronger result:

\[ (3.2) \text{ for all } p \in CL \text{ and } w \in W. p \rightarrow^w q \Rightarrow \hat{\theta}(p)w \leq \hat{\theta}(q) \]

from which the lemma follows since \( p \in PL \Rightarrow \hat{\theta}(p) = \varepsilon \text{ and } w \leq \hat{\theta}(q), w \in W \Rightarrow w \in \lambda\hat{\theta}(q) \). Notice that though \( \hat{\theta}(p) \in TSW \) we do not necessarily have \( \hat{\theta}(p)w \in TSW \). However this does not change the truth of (3.2).

To prove (3.2) we first prove:

\[ (3.3) \text{ for all } p \in CL, a \in Act. p \rightarrow^a q \Rightarrow \hat{\theta}(p)aq \leq \hat{\theta}(q) \]

by induction on the structure of \( p \) considered as a member of \( PL \).
\( p \in PL \): Then \( \bar{\theta}(p) = \varepsilon \), so we shall prove \( a \leq \bar{\theta}(q) \). This again will be proved by induction on the structure of \( p \).

\( p = NIL \): Looking at definition 3.2.2 we see \( p \to^b q \) for all \( b \), so we are done for this case.

\( p = b \cdot p' \): Recalling \( p \in PL \) when inspecting definition 3.2.2 we see that only 1) can could have been used to obtain \( p \to^a q \) and \( b = a \), so \( q = \bar{a}p' \). Since \( p = a \cdot p' \in PL \Rightarrow p' \in PL \) we have \( \bar{\theta}(q) = \bar{\theta}(\bar{a}p') = a \cdot \varepsilon = a \).

\( p = p_1 + p_2 \): Only 4) could have been used. W.l.o.g. assume \( p_1 \to^a q \). By hypothesis of induction \( a \leq \bar{\theta}(q) \).

\( p = p_1 \parallel p_2 \): Here only 5) can come under discussion. W.l.o.g. assume \( p_1 \to^a p'_1 \), \( q = p'_1 \parallel p_2 \). By hypothesis of induction \( a \leq \bar{\theta}(p'_1) \). Since \( p_2 \in PL \) we have \( a \leq \bar{\theta}(p'_1) = \bar{\theta}(p'_1) \parallel \varepsilon = \bar{\theta}(p'_1) \parallel \bar{\theta}(p_2) = \bar{\theta}(p'_1 \parallel p_2) = \bar{\theta}(q) \). This concludes the inductive step in the proof of \( \bar{\theta}(p) \leq \bar{\theta}(q) \) for \( p \in PL \).

\( p = b \cdot p' \): Inspecting definition 3.2.2 we see \( p = b \cdot p' \to^a q \), \( a \in Act \) implies \( p' \to^a q' \), where \( q = b \cdot q' \). By induction \( \bar{\theta}(p') \leq \bar{\theta}(q') \). By congruence of \( \leq \) we then have \( \bar{\theta}(p) a \leq \bar{\theta}(p'_1) \). Since \( L(p_1) \cap L(p_2) = \emptyset \) and we in general for \( r \in CL \) have: \( L(\bar{\theta}(r)) \subseteq L(r) \) and \( r \to^b r' \Rightarrow b \in L(r), L(r') \subseteq L(r) \) it follows that \( \bar{\theta}(p_1) a \) and \( \bar{\theta}(p_2) \) are disjoint so as \( \bar{\theta}(p'_1) \) and \( \bar{\theta}(p_2) \). Therefore \( \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \parallel \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \) are well-defined and members of TSW. So we can use proposition \( 1.3.29 \) to get \( \bar{\theta}(p_1) \parallel \bar{\theta}(p_2) \leq \bar{\theta}(p_1) \parallel \bar{\theta}(p_2) \). From the congruence of \( \leq \) we get \( \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \leq \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \). By transitivity of \( \leq \): \( \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \leq \bar{\theta}(p_1) a \parallel \bar{\theta}(p_2) \). So \( \bar{\theta}(p) a \parallel \bar{\theta}(p_2) = (\bar{\theta}(p_1) a \parallel \bar{\theta}(p_2)) \leq \bar{\theta}(p'_1) \parallel \bar{\theta}(p_2) \). Therefore, \( \bar{\theta}(p) a \parallel \bar{\theta}(p'_1) \parallel \bar{\theta}(p_2) = \bar{\theta}(q) \) thereby finishing the inductive step in the proof of (3.3).}

Having established (3.3) it is easy to prove (3.2) by induction on the length of \( w \).

\( w = \varepsilon \): Then \( q = p \). Clearly \( \bar{\theta}(p) \varepsilon = \bar{\theta}(q) \).

\( w = a \cdot w' \) for some \( a \in Act \), \( w' \in W \): Then \( p \to^a p' \to^w q \) for some \( p' \in CL \). By hypothesis of induction \( p' \to^w q \Rightarrow \bar{\theta}(p') w' \leq \bar{\theta}(q) \). From (3.3) we have \( p \to^a p' \Rightarrow \bar{\theta}(p) a \leq \bar{\theta}(p') \). Hence \( \bar{\theta}(p) a \cdot w' \leq \bar{\theta}(p') w' \) and by transitivity \( \bar{\theta}(p) w = \bar{\theta}(q) \)

\[ \square \]

**Lemma 3.3.11** For \( * = a \) \((* = r)\) and \( a_\alpha = \top \) \((o_\varepsilon = \bot)\) and all \( p, q \in PL \) we have:

\[ \ast \forall A \forall p' \in D(A, p) \exists q' \in D(A, q) \forall t \cdot (t, p') \to^* a_\ast \Rightarrow (t, q') \to^* o_\ast \]

iff

\[ \Theta(p) <_* \Theta(q) \]

**Proof**

\( * = a \): if: Let a multiset \( A \) over \( Act \) and a \( p' \in D(A, p) \) be given. We shall find a \( q' \in D(A, q) \) such that

\[ \forall t \cdot (t, p') \to^* \top \Rightarrow (t, q') \to^* \top \]
By definition $p' \in D(A, p)$ implies $\exists w \in W. p \xrightarrow{w} p', A_w \cong A$ and then from the definition of $\Theta$ we have $\theta(p') \in \bar{\Theta}(p)$. The assumed premise is $\bar{\Theta}(p) \prec_a \bar{\Theta}(q)$, so $\exists s_q \in \bar{\Theta}(q). \bar{\theta}(p') \preceq s_q$. Again by definition of $\bar{\Theta}$ we know that there exists $w' \in W$ and a $q'$ such that $q \xrightarrow{w'} q'$, $\bar{\theta}(q') = s_q$. By lemma 3.3.9 $\bar{\theta}(p') \preceq \bar{\theta}(q')$ implies (3.4) and $A_{\bar{\theta}(p')} = A_{\bar{\theta}(q')}$. So if we can prove $q' \in D(A, q)$ we have proved this implication. From lemma 3.3.10 we see $w' \in W$, $q \xrightarrow{w'} q'$ implies $w' \in \lambda\bar{\theta}(q')$. Hence $A_{w'} = A_{\bar{\theta}(q')} = A_{\bar{\theta}(p')} \cong A$. All in all we have $w' \in W$, $A_{w'} \cong A$ and $q \xrightarrow{w'} q'$, so $q' \in D(A, q)$.

only if: Let $s_p \in \bar{\Theta}(p)$ be given. By definition of $\prec_a$ we shall find an $s_q \in \bar{\Theta}(q)$ such that $s_p \preceq s_q$. $s_p \in \bar{\Theta}(p)$ means $\exists w \in W \exists p'. p \xrightarrow{w} p', \bar{\theta}(p') = s_p$. By definition of $\cong$ we have $A \cong A_w$ for the multiset $A$ over $\text{Act}$ defined by $|A|_a = \psi(w, a)$ for all $a \in \text{Act}$. So $p' \in D(A, p)$. Assuming the premise of the implication to be true there exists a $q' \in D(A, q)$ such that (3.4) holds. $q' \in D(A, q)$ implies $\exists w' \in W. q \xrightarrow{w'} q', A_{w'} \cong A$, so for all $a \in \text{Act}$. $\psi(w', a) = |A|_a$, wherefore for all $a \in \text{Act}. \psi(w', a) = \psi(w, a)$ and thereby $A_{w'} = A_w$. By lemma 3.3.10 we see $w' \in \lambda\bar{\theta}(q')$ and $w \in \lambda\bar{\theta}(p')$, so $A_{\bar{\theta}(q')} = A_{w'} = A_w = A_{\bar{\theta}(p')}$. This and (3.4) together with lemma 3.3.9 gives $\bar{\theta}(p') \preceq \bar{\theta}(q')$. Defining $s_q := \bar{\theta}(q')$ this reads $s_p = \bar{\theta}(p') \preceq \bar{\theta}(q') = s_q$. Now $s_q \in \bar{\Theta}(q)$ since $q \xrightarrow{w'} q'$, $w' \in W$ by definition of $\Theta$ implies $\bar{\theta}(q') \in \bar{\Theta}(q)$.

$* = r$: The proof is similar as for the case $* = a$, with the difference that another version of lemma 3.3.9 is used:

$$\bar{\theta}(q) \preceq \bar{\theta}(p) \iff A_{\bar{\theta}(q)} = A_{\bar{\theta}(p)}$$

(3.5) $\forall t. (t, p) \rightarrow^* \bot \Rightarrow (t, q) \rightarrow^* \bot$

To see this from lemma 3.3.9 notice that by proposition 3.2.12 we in general have

$$\neg((t, r) \rightarrow^* \top) \iff (t, r) \rightarrow^* \bot$$

wherefore (3.5) is equivalent to $\forall t. (t, q) \rightarrow^* \top \Rightarrow (t, p) \rightarrow^* \top$. $\square$

**Proof** of Semantic Characterization Theorem

Since for $* = a$ and $* = r$ we have $\bar{\Theta}(p) \prec_a \bar{\Theta}(q) \iff p \ll_a q$ we get the theorem from lemma 3.3.5, lemma 3.3.6 and lemma 3.3.11 in the cases $* = a$ and $* = r$. Finally: $p \ll q \iff p \ll_a q$ and $p \ll_r q \iff p \ll_a q$ and $p \ll_r q \iff p \ll q$. $\square$

We have already seen the close connection between the operators of $A_\delta$ and $A_\pi$ for $* \in \{\delta, \nu, \chi\}$ and soon we shall investigate how the interpretations of $PL$ in $A_\pi$ are related to the interpretation of $PL$ in $A_\pi$ which we now define in exactly the same way as we did in definition 3.1.13 for the $A_\pi$ algebras.

**Definition 3.3.12** $[\_]_\pi: PL \rightarrow C_\pi$ is recursively defined as follows:

$$[\text{NIL}]_\pi = \text{NIL}_\pi$$
$$[a.p]_\pi = a_\pi [p]_\pi$$
$$[p + q]_\pi = [p]_\pi +_\pi [q]_\pi$$
$$[p \parallel q]_\pi = [p]_\pi \parallel_\pi [q]_\pi$$
Recalling definition 3.1.8 where the interpretations in the $\Sigma$-po algebra $A_\pi$ of the operator symbols where defined, we see that definition 3.3.12 can be read as:

$\left[\text{NIL}\right]_\pi = \{\varepsilon\}$

$[a.p]_\pi = a.[p]_\pi \cup \{\varepsilon\}$

$[p + q]_\pi = [p]_\pi \cup [q]_\pi$

$[p \parallel q]_\pi = [p]_\pi \parallel [q]_\pi$

**Lemma 3.3.13** For all $p \in PL : \tilde{\Theta}(p) = [p]_\pi$

**Proof** Induction on the structure of $p$.

$p = \text{NIL}$: Since $\text{NIL} \not\rightarrow$ we have $\tilde{\Theta}(\text{NIL}) = \{\tilde{\theta}(\text{NIL})\} = \{\varepsilon\} = [\text{NIL}]_\pi$.

$p = a.p'$: $\tilde{\Theta}(p) = \tilde{\Theta}(a.p') = \{\tilde{\theta}(q) | \exists w \in W. a.p' \overset{w}{\rightarrow} q\} = \{\tilde{\theta}(q) | \exists w \in W. w \neq \varepsilon, a.p' \overset{w}{\rightarrow} q\} \cup \{\tilde{\theta}(p)\}$ (since $p \in PL$)

(3.6) \quad \{\tilde{\theta}(q) | \exists w \in W. w \neq \varepsilon, a.p' \overset{w}{\rightarrow} q\} \cup \{\varepsilon\}

Inspecting definition 3.2.2 we see $a.p' \overset{w}{\rightarrow} q$, $w \neq \varepsilon$ iff $w = a.w'$, $p' \overset{w'}{\rightarrow} q'$ and $q = a.q'$ for some $w' \in W$. So (3.6) $= \{\tilde{\theta}(a.q') | \exists w' \in W. p' \overset{w'}{\rightarrow} q'\} \cup \{\varepsilon\}$ (by definition of $\tilde{\theta}$) $\{\tilde{\theta}(q') | \exists w' \in W. p' \overset{w'}{\rightarrow} q'\} \cup \{\varepsilon\}$ =

(3.7) \quad a.\tilde{\Theta}(p') \cup \{\varepsilon\}

By hypothesis of induction $\tilde{\Theta}(p') = [p']_\pi$, so (3.7) equals $[a.p']_\pi$.

$p = p_1 + p_2$: $\tilde{\Theta}(p) = \tilde{\Theta}(p_1 + p_2) =$

(3.8) \quad \{\tilde{\theta}(q) | \exists w \in W. p_1 + p_2 \overset{w}{\rightarrow} q\}

Again from definition 3.2.2 we see $p_1 + p_2 \overset{w}{\rightarrow} q$ iff either $p_1 \overset{w}{\rightarrow} q$ or $p_2 \overset{w}{\rightarrow} q$, so (3.8) equals $\{\tilde{\theta}(q) | \exists w \in W. p_1 \overset{w}{\rightarrow} q\} \cup \{\tilde{\theta}(q) | \exists w \in W. p_2 \overset{w}{\rightarrow} q\}$. The result then follows directly from the hypothesis.

$p = p_1 \parallel p_2$: $\tilde{\Theta}(p) = \tilde{\Theta}(p_1 \parallel p_2) =$

(3.9) \quad \{\tilde{\theta}(q) | \exists w \in W. p_1 \parallel p_2 \overset{w}{\rightarrow} q\}

The next to see is that (3.9) equals

(3.10) \quad \{\tilde{\theta}(q_1 \parallel q_2) | \exists w_1, w_2 \in W \exists q_1, q_2 \in CL. p_1 \overset{w_1}{\rightarrow} q_1 \parallel q_2 \overset{w_2}{\rightarrow} q_2\}

$\subseteq$: Follows directly from lemma 3.3.7.

$\supseteq$: Let $s$ in (3.10) be given. Then there exists $w_i \in W, q_i \in CL, p_i \overset{w_i}{\rightarrow} q_i$ for $i \in 2$. Since $p_1 \parallel p_2$ is well-defined we have $L(p_1) \cap L(p_2) = \emptyset$. By corollary 3.2.3 then $q_1 \parallel q_2$ and $w_1 \parallel w_2$ are well-defined too, so we can define $q := q_1 \parallel q_2$. From proposition 1.3.5.b we know $\lambda(w_1 \parallel w_2) \neq \emptyset$, so there exists a $w \in W. w \geq w_1 \parallel w_2$. Then we can use lemma 3.3.7 to get $p_1 \parallel p_2 \overset{w}{\rightarrow} q$. Hence $s = \tilde{\theta}(q_1 \parallel q_2) = \tilde{\theta}(q)$ in (3.9). Clearly (3.10) equals $\{\tilde{\theta}(q_1) | \exists w_1 \in W. p_1 \overset{w_1}{\rightarrow} q_1\} \parallel \{\tilde{\theta}(q_2) | \exists w_2 \in W. p_2 \overset{w_2}{\rightarrow} q_2\} = \tilde{\Theta}(p_1) \parallel \tilde{\Theta}(p_2)$ from which the result follows directly from the hypothesis of induction.
Lemma 3.3.14 For all $S \subseteq SW$ (hence also for $S \subseteq TSW$) we have

\[ \delta S <_a S <_a \delta S \]
\[ vS <_r S <_r vS \]
\[ \chi S < S < \chi S \]

Proof $\delta S <_a S$ and $vS <_r S$ follows directly from the definition of $\delta$ and $v$. $S <_a \delta S$, $S <_r vS$ and $S < \chi S$ follows from the reflexivity of $\leq$ and $S \subseteq \delta S, vS, \chi S$.

$\chi S < S$: We shall prove $\chi S <_a S$ and $\chi S <_r S$. But this is evident since by definition $s \in \chi S$ implies $t \leq s \leq t'$ for some $t, t' \in S$. \hfill \square

Combining the last two lemmas we immediately have:

Corollary 3.3.15 For all $p, q \in PL$:

\[ p \ll_a q \iff \delta[p]_\pi <_a \delta[q]_\pi \]
\[ p \ll_r q \iff v[p]_\pi <_r v[q]_\pi \]
\[ p \ll q \iff \chi[p]_\pi < \chi[q]_\pi \]

Lemma 3.3.16 For $\star$ in $\{\delta, v, \chi\}$ and all $p \in PL$:

\[ \star[p]_\pi = *[p]_\pi \]

Proof If we for $\overline{S} \in C_\pi$ have:

(3.11) \[ \star op_\pi(\overline{S}) = *op_\pi(S) \]

then $\star$ is easily proved by induction on the structure of $p$ as indicated here: $*[op_\pi(p)]_\pi = op_\pi([p]_\pi)$ (by proposition 3.1.9) $\star op_\pi([p]_\pi) = \star op_\pi(\overline{S})$ (hypothesis of induction) $\star op_\pi(\overline{S}) = \star op_\pi([p]_\pi)$ (by (3.11)) $\star op_\pi([p]_\pi) = *op_\pi(S)$.

In proving (3.11) we use the properties of (sets of) tree semiwords, the fact that $\delta$ and $v$ distributes over $\cup$ and the closure properties of $\star$:

(3.12) \[ \star \star S = \star S \]

for arbitrary sets of semiwords $S$.

The proof of (3.11):

$op_\pi = NIL_\pi$: Trivial.

$op_\pi = a_\pi$: $*a_\pi \star \overline{S} = *a_\pi(S)$, which by the $\cup$-distributivity of $\delta, v$ and for $\star = \chi$: Proposition $T \ 1.3.24.a$ equals $*a_\pi \star S \cup \star \{\varepsilon\}$. By corollary $T \ 1.3.19.b$ in the case of $\star = v$ and (3.12), corollary $T \ 1.3.16.a$, corollary $T \ 1.3.25$ in the other cases we see that this quantity is the same as $*a_\pi \star S \cup \star \{\varepsilon\}$. With the same arguments as above this equals $*a_\pi \star (S \cup \{\varepsilon\}) = *a_\pi S$. 83
\( op_\pi = +_\pi: \star(S +_\pi T) = \star(S \cup \pi T) \) (by (3.12), \( \cup \)-distributivity of \( \delta, \nu \) and in the case \( \star = \chi \): corollary \( T 1.3.23.c \)).

\( op_\pi = ||_\pi: \star(S ||_\pi T) = \star(S \| \pi T) \). Using corollary \( T 1.3.16.b \) when \( \star = \delta \), (3.12) and proposition \( T 1.3.18.d \) in the case \( \star = \nu \) and finally for \( \star = \chi \): corollary \( T 1.3.23.b \), we see \( \star(S ||_\pi T) = \star(S \| T) = \star(S ||_\pi T) \).

\( \square \)

**Lemma 3.3.17**

\( \delta \) \( \forall S, T \in C_\delta . S \triangleleft_a T \iff S \sqsubseteq_\delta T \)

\( v \) \( \forall S, T \in C_\nu . S \triangleleft_r T \iff S \sqsubseteq_\nu T \)

\( \chi \) \( \forall S, T \in C_\chi . S \triangleleft T \iff S \sqsubseteq_\chi T \)

**Proof** Recall at first corollary \( 3.1.4 \) that for \( \star \in \{ \delta, \nu, \chi \}: \star(T) = T . \)

\( \delta \) if: Assume \( S \sqsubseteq_\delta T \) or equivalently \( S \subseteq T . \) We shall show \( \forall s \in S \exists t \in T . s \subseteq t . \) Let \( s \in S \) be given. Since \( S \subseteq T \) and \( \subseteq \) is reflexive we can chose \( t = s . \)

only if: Assume \( S \triangleleft_a T \). We shall show \( S \subseteq T . \) Let a \( s \in S \) be given. By assumption there exists a \( t \in T \) such that \( s \subseteq t . \) Since \( \delta T = T \) we have \( s \in T \) too.

\( v \) Similar.

\( \chi \) if: Assume \( S \sqsubseteq_\chi T . \) We shall prove \( S \triangleleft_a T \) and \( S \triangleleft_r T . \) This follows as for \( \delta \) and \( v \).

only if: Assume \( S \triangleleft T . \) We shall prove \( S \subseteq T . \) Let \( s \in S . \) From \( S \triangleleft_a T \) we see \( \exists t' \in T . s \triangleleft t' \) and from \( S \triangleleft_r T \) \( \exists t \in T . t \triangleleft s . \) Hence \( \exists t, t' \in T . t \triangleleft s \triangleleft t' \) and thereby \( s \in \chi T . \) Since for \( \chi T = T \) the result follows.

\( \square \)

From the last two lemmas and corollary 3.3.15 it follows:

**Corollary 3.3.18** For all \( p, q \in PL : \)

\( \delta \) \( p \triangleleft_a q \iff [p]_\delta \sqsubseteq_\delta [q]_\delta \)

\( v \) \( p \triangleleft_r q \iff [p]_v \sqsubseteq_\nu [q]_v \)

\( \chi \) \( p \triangleleft q \iff [p]_\chi \sqsubseteq_\chi [q]_\chi \)

We are now in a position to prove the operational characterization theorem on page 75.
\textbf{Proof} (of Operational Characterization Theorem)

From the Semantic Characterisation Theorem and corollary 3.3.18 it follows that e.g., $\leq_\alpha$ and $\leq_\delta$ agrees on $PL$. By corollary 3.1.11 the different operators of $\Sigma_\delta$ are (relative) $\leq_\delta$-monotone, so from the compositional definition of $\llbracket \cdot \rrbracket_\delta$ we deduce that $\leq_\delta$ is a (relative) precongruence. Because of the agreement between $\leq_\alpha$ and $\leq_\delta$ this must be the case for $\leq_\alpha$ too. Similar for the other preorders. \hfill $\Box$ \\

Before ending the section we shall prove that the denotational maps can denote any element of the relevant domain—a fact which will be used in the next chapter.

**Proposition 3.3.19** $\llbracket \cdot \rrbracket_\star : PL \rightarrow C_\star$ is surjective for $\star \in \{\delta, \nu, \chi \}$.

**Proof** Given a $S \in C_\star$. We shall find a $p \in PL$ with $[p]_\star = S$. $S \in C_\star$ implies that there exists a $T \in \mathcal{P}_f(TSW) \setminus \emptyset$ such that $\star \pi T = S$. Because by lemma 3.3.16 $[p]_\star = \star [p]_\pi$ we see that it is enough to find a $p$ such that $[p]_\pi = \pi T$ since then $[p]_\star = \star [p]_\pi = \star \pi T = S$. Now $\pi$ is $\cup$-distributive, so $\pi T = \cup_{t \in T} \pi(t)$. Hence we are done if we for every $t \in T$ can find a $p_t \in PL$ such that $[p_t]_\pi = \pi(t)$, because then we can chose $p$ to be $\Sigma t \in T p_t$ ($T$ is finite) and get $[p]_\pi = (\text{by definition of } +_\pi \text{ and proposition } T 1.3.35.c) \cup_{t \in T} [p_t]_\pi = \cup_{t \in T} \pi(t) = \pi T$.

We will now find such a $p_t$ for a given $t$ by induction on the size of $t$.

**Base:** Clearly $t = \varepsilon$. Let $p_t = NIL$. Then $[p]_\pi = NIL_\pi = \{\varepsilon\} = \pi(\varepsilon) = \pi(t)$.

**Inductive step:** Then $\gamma(t) \neq \{\varepsilon\}$ and $t = \varepsilon \cup (|| \gamma(t) \setminus \{\varepsilon\} || = || \gamma(t) \setminus \{\varepsilon\} ||$. By corollary $T 1.1.8.1$ $t' \in R TSW$ for every $t' \in \gamma(t) \setminus \{\varepsilon\}$.

If $\gamma(t) \setminus \{\varepsilon\}$ only consists of one rooted tree-semiword, $t'$, proposition $T 2.2.15.a$ gives us $\exists t'' \in TSW. t' = a. t''$. By hypothesis of induction we can find a $p'' \in PL$ such that $[p'']_\pi = \pi(t'')$. Let $p = a. p''$. Then $[p]_\pi = a. \pi [p'']_\pi = a. \pi(t'') \cup \{\varepsilon\} = (\text{by corollary } T 1.3.36) \pi(a. t'') = \pi(t') = \pi(|| \gamma(t) \setminus \{\varepsilon\} ||) = \pi(t)$.

If $\gamma(t) \setminus \{\varepsilon\}$ consists of more than one rooted tree-semiword we clearly can write $t$ as $t_1 \parallel t_2$, where $t_1, t_2$ are nonempty tree-semiwords of size less than $t$. By hypothesis we find $p_i \in PL$. $[p_i]_\pi = t_i$ for $i \in 2$. Let $p = p_1 \parallel p_2$. Then $[p]_\pi = [p_1]_\pi \parallel_\pi [p_2]_\pi = \pi(t_1) \parallel \pi(t_2) = (\text{by proposition } T 1.3.35) \pi(t_1 \parallel t_2) = \pi(t)$.

Finally we will briefly compare the equivalences. Since $\leq$ is defined as the intersection of $\leq_\alpha$ and $\leq_\nu$, it is immediate from the full abstraction results of this section, that both $\llbracket \cdot \rrbracket_\delta$ and $\llbracket \cdot \rrbracket_\nu$ is as abstract as $\llbracket \cdot \rrbracket_\chi$. By the two process terms:

\[ p_1 = a. b. NIL + a. NIL \parallel b. NIL \parallel b. NIL \]

it follows that $\llbracket \cdot \rrbracket_\delta$ is strictly more abstract than $\llbracket \cdot \rrbracket_\chi$ (identified by $\llbracket \cdot \rrbracket_\delta$ but not by $\llbracket \cdot \rrbracket_\chi$). That $\llbracket \cdot \rrbracket_\nu$ also is strictly more abstract than $\llbracket \cdot \rrbracket_\chi$ is seen from $p_1$ and

\[ p_3 = a. b. NIL \]

The same examples can be used to see that in general $\llbracket \cdot \rrbracket_\delta$ is not as abstract as $\llbracket \cdot \rrbracket_\nu$ and vice versa; $p_1$ and $p_2$ are identified by $\llbracket \cdot \rrbracket_\delta$ but not by $\llbracket \cdot \rrbracket_\nu$, and conversely with $p_1$ and $p_3$. 

\textit{85}
Chapter 4

Algebraic Characterization

The purpose of this chapter is to introduce two proof systems which on a purely syntactical level enables us to reason about how processes of \( PL \) can be interrelated via the different operational precongruences. The idea will be that if a certain relation between two process terms can be shown in the proof system then they will be related via the corresponding test preorder in the same way.

As for the previous chapter most of the concepts are as described in [Hen85a] and we shall also use some of the results.

Given a set of variables, \( X \), and an arbitrary \( \Sigma \)-po algebra, \( A = (C_A, \leq_A, \Sigma_A) \), an \( A \)-assignment is a mapping \( \rho_A : X \rightarrow C_A \), and from the proof of the freeness theorem we know a structural defined unique extension of \( \rho_A \) to the term algebra for \( \Sigma(X) \). If \( BL \) is extended to the term algebra for the signature with variables, the corresponding extensions of the different \( A_* \)-assignments would not always be well-defined. Some modifications are therefore necessary.

For a set of variables, \( X \), \( BL \) is extended to include terms with variables from \( X \) simply by extending the signature \( \Sigma \) to \( \Sigma(X) \) by augmenting \( \Sigma_0 \) with \( X \). The so obtained term algebra is denoted \( BL(X) \). We shall assume that each variable, \( x \in X \), has an associated sort/label set, \( L_x \) and furthermore that there is an infinite number variables for every possible sort (finite subset of \( Act \)). Extending the map \( L \) from \( BL \) to \( BL(X) \) by letting \( L(x) = L_x \) for every \( x \in X \) we can similarly as \( PL \) was extracted from \( BL \) define \( PL(X) \)—the open process terms—to be those terms of \( BL(X) \) where every subterm of form \( t \parallel t' \) satisfies \( L(t) \cap L(t') = \emptyset \). To emphasize the possibility of variables we shall often use \( t, t', \ldots \) to denote terms from \( PL(X) \).

An \( A_* \)-assignment, \( \rho_A_* \), is now defined to be a mapping \( X \rightarrow C_* \) such that for all \( x \in X \) we have:

\[
\rho_A_*(x) = S \Rightarrow L(x) = L(S)
\]

If \( \rho_A_* \) is extended to \( PL(X) \) in the same way as in the freeness theorem, \( \rho_A_* \) is in this way ensured to be well-defined (and unique). Notice that \([p]_* = \rho_A_*(p)\) for all \( A_* \)-assignments, \( \rho_A_* \), if \( p \in PL \).

The same goes for syntactic substitutions, i.e., \( PL(X) \)-assignments. That is \( \rho : X \rightarrow \)
PL(X) is a PL(X)-assignment if for every \( x \in X \):
\[
\rho(x) = t \Rightarrow L(x) = L(t)
\]
The extension of a PL(X)-assignment to syntactic substitution will be written postfix.

### 4.1 Proof Systems

In this section we are going to formulate two proof systems \( \text{DED}_\delta \) and \( \text{DED}_\pi \) respectively. These proof systems \( \text{DED}_\delta \) and \( \text{DED}_\pi \) will contain the (usual) inference rules for (relative) precongruence, instantiation, transitivity, reflexivity as well as the inference rule for basic inequations:

<table>
<thead>
<tr>
<th>Reflexivity: ( t \leq t )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transitivity: ( t \leq t', t' \leq t'' \Rightarrow t \leq t'' )</td>
</tr>
<tr>
<td>Substitutivity: ( a.t \leq a.t' ) ( t_1 \leq t'_1, t_2 \leq t'_2 \Rightarrow t_1 + t_2 \leq t'_1 + t'_2 )</td>
</tr>
<tr>
<td>( t_1 \leq t'_1 ), ( t_2 \leq t'_2 ) ( t_1 \parallel t_2 \leq t'_1 \parallel t'_2 ) provided ( L(t_1) \cap L(t_2) = \emptyset = L(t'_1) \cap L(t'_2) )</td>
</tr>
<tr>
<td>Instantiation: ( t \leq t' ) for every PL(X)-assignment ( \rho )</td>
</tr>
<tr>
<td>Inequations: ( t \leq t' ) for every ( (\pi-) ) ( \delta )-inequation ( t \leq t' )</td>
</tr>
</tbody>
</table>

The \( \delta \)-inequations and \( \pi \)-inequations respectively are as displayed below:
\[ \pi\text{-inequations:} 
+1 \quad x + (y + z) = (x + y) + z 
+2 \quad x + y = y + x 
+3 \quad x = x + \text{NIL} 
+4 \quad x = x + x 
\]

\[ ||1 \quad x || (y || z) = (x || y) || z 
||2 \quad x || y = y || x 
||3 \quad x = x || \text{NIL} 
\]

. + \quad a.(x + y) = a.x + a.y 
||+ \quad x || (y + z) = x || y + x || z 

\[ +5 \quad x \leq x + y \]

\[ \delta\text{-inequations: } \pi\text{-inequations and} \]
\[ \delta.|| \quad a.(x || y) \leq a.x || y \]

So the inequations are just relations between terms of \( PL(X) \).

More generally for a proof system \( DED(E) \) of inequations as described by Hennessy [Hen85a], where \( E \) is the set of basic inequations we have the following notions.

The inequations inference rule gives a statement \( t \leq t' \) for every \( t \leq t' \) in \( E \). These statements together with \( t \leq t \) obtained by the reflexivity inference rule (with no premise) will be denoted the axioms.

A proof, \( P \), is a sequence of statements

\[ t_1 \leq t'_1, t_2 \leq t'_2, \ldots, t_n \leq t'_n \]

where each statement \( t_i \leq t'_i, i \in \mathbb{N} \) is derived by applying the inference rules to statements earlier in the sequence. Clearly each \( t_i \leq t'_i, i \in \mathbb{N} \) has its own proof which is a part of \( P \). We denote it by \( P_{t_i \leq t'_i} \).

We will say that a proof has the \textit{simple instantiation property} if instantiation only is used on the axioms.

Later in section 4.3 we shall see that if \( P \) is a proof of \( t \leq t' \) then there is another proof \( P' \) of \( r \leq t' \) with this property.

Notice that \( DED_{\delta} \) and \( DED_{\pi} \) just are special cases of \( DED(E) \) with \( E = \delta\text{-inequations} \) and \( E = \nu\text{-inequations} \) respectively.

If the statement \( t \leq t' \) can be proved in \( DED_{\pi} \) we shall write this as \( \vdash_{\pi} t \leq t' \). Similar for the statements of \( DED_{\delta} \). Since the \( \pi\text{-inequations} \) are contained in the \( \delta\text{-inequations} \) it follows that \( \vdash_{\pi} t \leq t' \) implies \( \vdash_{\delta} t \leq t' \). As mentioned in the beginning to the chapter the proof systems can be used to deduce how processes can be operationally related. This will be more accurately addressed in the next two sections.
4.2 Soundness

In this section we shall see that all statements proved in $DED_\pi$ will hold for the different operational (relative) precongruences. More precisely $DED_\pi$ is sound w.r.t. $\preceq_\pi$ over $PL(X)$ in the sense:

$$\vdash_\pi t \leq t' \text{ implies } t \preceq_\pi t'$$

where $\preceq_\pi$ is extended from $PL$ to $PL(X)$ in the usual way by letting:

$$t \preceq_\pi t' \iff \rho_{PL} t \preceq_\pi \rho_{PL} t' \text{ for all } PL\text{-assignments } \rho_{PL}$$

Furthermore the larger proof system, $DED_\delta$, will also be sound w.r.t. $\preceq_\pi$.

**Theorem 4.2.1 (Soundness)**

$DED_\delta$ is sound w.r.t. $\preceq_\pi$ over $PL(X)$.

$DED_\pi$ is sound w.r.t. $\preceq_\pi$ over $PL(X)$.

$DED_\pi$ is sound w.r.t. $\preceq_\pi$ over $PL(X)$.

**Proof** For an arbitrary $PL$-assignment, $\rho_{PL}$, it is easy to see from the substitution lemma (see [Hen85a]) that the $A_\ast$-assignment, $\rho_{A_\ast}$, given by $\rho_{A_\ast}(x) = \llbracket \rho_{PL}(x) \rrbracket_\ast$, fulfills

(4.1) $$\forall t \in PL(X). \rho_{A_\ast}(t) = \llbracket t \rho_{PL} \rrbracket_\ast$$

Conversely it is, due to the surjectivity of $\llbracket \rrbracket_\ast$ as seen in proposition 3.3.19, also possible for any given $A_\ast$-assignment, $\rho_{A_\ast}$, to find a $PL$-assignment such that $\llbracket \rho_{PL}(x) \rrbracket_\ast = \rho_{A_\ast}$. From the substitution lemma, (4.1) then holds. Consequently for $t, t' \in PL(X)$ we have

$$\llbracket t \rho_{PL} \rrbracket_\ast \subseteq \llbracket t' \rho_{PL} \rrbracket_\ast \text{ for all } PL\text{-assignments } \rho_{PL}$$

$$\text{iff } \rho_{A_\ast}(t) \subseteq \rho_{A_\ast}(t') \text{ for all } A_\ast\text{-assignments } \rho_{A_\ast}$$

The theorem is then immediate from the following proposition and the full abstractness results of the preceding chapter. The denotational preorders, $\mathfrak{A}_\ast$, are extended to $PL(X)$ by:

$$t \preceq_\ast t' \iff \rho_{A_\ast}(t) \subseteq \rho_{A_\ast}(t') \text{ for all } A_\ast\text{-assignments } \rho_{A_\ast}$$

\[ \square \]

**Proposition 4.2.2**

\(\delta\) $DED_\delta$ is sound w.r.t. $\mathfrak{A}_\delta$ over $PL(X)$.

\(v\) $DED_\pi$ is sound w.r.t. $\mathfrak{A}_v$ over $PL(X)$. 89
In the Proof A equivalently by the definition of PL if for all A is surjective and agrees with A. Similar considerations for the A to Hennessy enough to show every t; t′ implies ρAδ(t) ≤ ρAδ(t′) for every Aδ-assignment ρAδ. To do this it is according to Hennessy enough to show Aδ satisfies the set of δ-inequations. I.e., we shall show for every t ≤ t′ in the δ-inequations and every Aδ-assignment, ρAδ, that ρAδ(t) ≤ ρAδ(t′) or equivalently by the definition of Aδ that ρAδ(t) ⊆ ρAδ(t′). Since \([t]\)δ by proposition 3.3.19 is surjective and agrees with Aδ on closed terms this follows from the substitution lemma if for all PL-assignments, ρPL, and all t ≤ t′ in the δ-inequations: \([t];\)δ ≤ \([t];\)δ.

Similar considerations for the v) and χ) case.

δ): We look at the (in)equations one by one.

+1: We shall show that for all p, q, r ∈ PL, \([p + (q + r)]\)δ = \([(p + q) + r]\)δ.

This is seen as follows: \([p + (q + r)]\)δ = \([p]δ + \delta [q + r]δ = \ldots = [p]δ ∪ ([q]δ ∪ [r]δ) = ([p]δ ∪ [q]δ ∪ [r]δ) = \ldots =\).

+2: Similar.


For all r ∈ PL we have \(ε ∈ [r]δ\) and thereby also \(\{ε\} ⊆ (\{r\})δ\). Hence by lemma 3.3.16

\(\{ε\} ⊆ [r]δ\)

Using r = p in (4.2) we get \(⊇\) too.

+4: Evident.

‖1: The proof of this case is not as obvious as for +1. Let p, q, r ∈ PL be given.

\([p \parallel (q \parallel r)]δ = \ldots = \delta ([p]δ ∥ \delta ([q]δ ∥ [r]δ)) = (\text{corollary 3.1.4}) \delta ([p]δ ∥ \delta ([q]δ ∥ [r]δ)) = (\text{corollary 1.3.16}) \delta ([p]δ ∥ ([q]δ ∥ [r]δ)) = (\text{corollary T}) \delta ([p]δ ∥ ([q]δ ∥ [r]δ)) = \ldots = ([p \parallel q] ∥ [r]δ)\).

‖2: By the commutativity of ∥.

‖3: \([p \parallel NIL]δ = \ldots =\)

\(\delta ([p]δ ∥ \{ε\}) = \delta [p]δ\)

= (corollary 3.1.4)[p]δ. Equation (4.3) follows from \(\{ε\}\) being neutral to ∥ on \(P(TSW)\) which again is inherited from (corollary T 1.2.11) ε being neutral to ∥ on TSW.


‖+: Similar, but with the additional use of the ∪-distributivity of δ.

+5: We shall show \([p]δ ⊆ [p + q]δ\) which evidently is true since \([p + q]δ = [p]δ ∪ [q]δ\).

δ .: At first we show that in general for S, T ⊆ TSW we have

\(a.δ(S ∥ T) ⊆ δ((a.S ∪ \{ε\}) ∥ T)\)
Clearly \( \delta(a.S \parallel T) \subseteq \delta((a.S \cup \{\varepsilon\}) \parallel T) \), so it is enough to show \( a.\delta(S \parallel T) \subseteq \delta(a.S \parallel T) \).

Let \( u \in a.\delta(S \parallel T) \). Then \( u = a.u' \) where \( u' \in \delta(S \parallel T) \) so there is exists some \( s \in S, t \in T \) such that \( u' \preceq s \parallel t \). By congruence of \( \preceq \), \( u = a.u' \preceq a.(s \parallel t) \).

From proposition 1.3.29 we have \( a.(s \parallel t) \preceq a.s \parallel t \) wherefore \( u \preceq a.s \parallel t \). Hence \( u \in \delta(a.S \parallel T) \).

Letting \( S = [p]_\delta, T = [q]_\delta \) in (4.4) it reads \( a.\delta([p]_\delta \parallel [q]_\delta) \subseteq \delta((a.[p]_\delta \cup \{\varepsilon\}) \parallel [q]_\delta) \).

Now \( \delta((a.[p]_\delta \cup \{\varepsilon\}) \parallel [q]_\delta) = \delta([a.p]_\delta \parallel [q]_\delta) = [a.p \parallel q]_\delta \) and \( a.\delta([p]_\delta \parallel [q]_\delta) = a.[p \parallel q]_\delta \), so we have \( a.[p \parallel q]_\delta \subseteq [a.p \parallel q]_\delta \).

By (4.2) we have \( \{\varepsilon\} \subseteq [a.p \parallel q]_\delta \), too, so \( [a.(p \parallel q)]_\delta = a.[p \parallel q]_\delta \cup \{\varepsilon\} \subseteq [a.p \parallel q]_\delta \).

\( v \): The arguments are almost as for the \( \delta \)-case just using the properties of \( v \) instead.

\( +1 - +4 \): As in the case \( \delta \)

\( \parallel 1 - \parallel 2 \): Similar to \(+1\) and \(+2\) because \( \parallel v \), does not have explicit \( v \)-closure.

\( \parallel 3 \): Follows with the same arguments as in \( \delta \).

\( .+: \) This case is a little different from the \( \delta \)-case because we have explicit \( v \)-closure in the definition of \( a.v \), but it is just as easy though.

\( a.(p \parallel q)_v = \ldots = va.([p]_v \cup [q]_v) \cup \{\varepsilon\} = v(a.[p]_v \cup a.[q]_v) \cup \{\varepsilon\} = (\cup\text{-distributivity of } v) va.([p]_v \cup va.[q]_v) \cup \{\varepsilon\} = (va.[p]_v \cup \{\varepsilon\}) \cup (va.[q]_v \cup \{\varepsilon\}) = \ldots = [a.p + a.q]_v. \)

\( \parallel +: \) As the \( .+ \)-case.

\( +5 \): Similar to the \( \delta \)-case.

\( \chi \): Here we deduce:

\( +1 \): Suppose \( p, q, r \in PL \). Then \( [p + (q + r)]_\chi = \chi([p]_\chi \cup \chi([q]_\chi \cup [r]_\chi)) = (\text{corollary}_T 1.3.23) \chi([p]_\chi \cup [q]_\chi \cup [r]_\chi) = \chi(\chi([p]_\chi \cup [q]_\chi) \cup [r]_\chi) = \ldots = [(p + q) + r]_\chi \)

\( +2 \): Direct from definition of \( [.]_\chi \) and commutativity of \( \cup \).

\( +3 \): Similar arguments as for \(+3\) of \( \delta \) but without \( \chi \).

\( +4 \): Evident.

\( \parallel 1 - \parallel 3 \): Similar to the corresponding cases of \( \delta \) but corollary \( T 1.3.16 \) is used in stead of corollary \( T 1.3.23 \).

\( .+: \) \( a.(p \parallel q)_\chi = \ldots = a.\chi([p]_\chi \cup [q]_\chi) \cup \{\varepsilon\} = (\text{proposition}_T 1.3.24.a) \chi(a.[p]_\chi \cup a.[q]_\chi \cup \{\varepsilon\}) = \chi([a.p]_\chi \cup [a.q]_\chi) = [a.p + a.q]_\chi. \)

\( +5 \): From \( a \) of proposition \( T 1.3.22\) follows \( [p]_\chi \subseteq \chi([p]_\chi) \) so evidently \( [p]_\chi \subseteq \chi([p]_\chi \cup [q]_\chi) = [p + q]_\chi \) for any \( p, q \in PL \).

\[ \square \]

### 4.3 Completeness

We shall now see that \( DED_\delta \) is powerful enough to derive any \( \preceq_a \)-relationship between two process.
**Theorem 4.3.1 (Completeness)**

DED is complete w.r.t. \( \preceq_\pi \) over PL. I.e.,

\[
\forall p, q \in PL. p \preceq_\pi q \implies \vdash_\pi p \leq q
\]

**Proof**  Follows from \( [\ ]_\pi \) being fully abstract w.r.t. \( \preceq_\pi \) and the proposition below. \( \square \)

From the \( \pi \)-inequations it appears that in \( DED_\pi \) statements concerning prefix \((-\)closures) as well as more ordinary algebraic properties such as commutativity and associativity can be proved. With the extra inequation, \( \delta \| \) it becomes possible to deal with \( \delta \)-closures. Looking at the inequation \( \delta \| \) it is then tempting to replace it with

\[
v.\| \ a.x \| y \leq a.(x \| y)
\]

in order to obtain a complete proof system, \( DED_v \), for \( \preceq_v \). However this inequation would not be sound as can be seen by considering the instantiation of \( v.\| \):

\[
a.b.NIL \| c.NIL \leq a.(b.NIL \| c.NIL)
\]

Then we would have \( a.b.NIL \| c.NIL \) may reject \( \{c\}, a.\top \), but \( a.(b.NIL \| c.NIL) \) may reject \( \{c\}, a.\top \). This can just as easy be seen denotationally: \( c \in [a.b.NIL \| c.NIL]_v \), but \( c \not\in [a.(b.NIL \| c.NIL)]_v \).

We could obtain a more powerful proof system for \( \preceq_v \) than \( DED_\pi \) by adding the sound inequations:

\[
a.x \| y \leq a.(x \| y) + a.NIL \| y
\]

\[
a.x \| y \leq a.(x \| y) + y
\]

Still we would not be able to prove e.g., \( a.b.NIL \| c.NIL \leq a.b.c.NIL + a.NIL \| c.NIL \).
Of course still more inequations could be added, but we have not been able to find a complete set, wherefore we stick to \( DED_\pi \) which is sound for all three preorders.

**Proposition 4.3.2** \( DED_\delta \) is complete w.r.t. \( \preceq_\delta \) over PL.

The proof can of course not be done so directly as the soundness proof and some auxiliary propositions are needed. To motivate these and the necessary extra definitions below we will at first outline the proof—the full proof is on page 108.

**Proof**  (sketch)  
The main idea for proving

\[
p \preceq_\delta q \Rightarrow \vdash_\delta p \leq q
\]

is to reduce \( p \) (via \( \vdash_\delta \)) to a sum, \( p' \), of composition forms (terms without \( + \)) with the property that there is exactly one summand for each tree semiword in the denotation of \( p \) in the \( M_\delta \) model. If the same is done for \( q \) thereby obtaining \( q' \) then the premise of (4.5)
and definition of \( \delta \) ensures that \( q' \) can be proved equal to a term of the form \( p' + q'' \) and so the consequence of (4.5) follows by applying +5. The sum of composition forms with the desired property is obtained through more stages. At first a sum of composition forms is obtained essentially using the axioms for distributivity, + and \( \parallel + \). Then all prefixes of the composition forms are added by use of +3 whereupon the composition forms corresponding to the downwards closure of the sum are included via \( \delta \parallel \). Finally all duplicates (up to commutativity and associativity of \( \parallel \)) of the composition forms are removed by means of +4 (idempotent) in order to get the one to one correspondence with the denotation. \( \Box \)

**Definition 4.3.3**

Let \( p \) be a process from our process language \( (p \in PL) \). At first we define two fundamental sublanguages of \( PL \).

\( p \) is a *composition form* \((p \in cf)\) is inductively defined by:

\[
\begin{align*}
NIL & \in cf \\
 a.p & \in cf \quad \text{if} \quad p \in cf \\
 p \parallel q & \in cf \quad \text{if} \quad p, q \in cf
\end{align*}
\]

\( p \) is a *sumnormal form* \((p \in snf)\) is defined by:

\[
\begin{align*}
\text{cf} & \subseteq \text{snf} \\
p + q & \in \text{snf} \quad \text{if} \quad p, q \in \text{snf}
\end{align*}
\]

We can now define the set of *summands*, \( S(p) \), of a sumnormal form \( p \).

Let \( S: \text{snf} \rightarrow \mathcal{P}(\text{cf}) \) be defined by:

\[
\begin{align*}
 p & \mapsto \{p\} \quad \text{if} \quad p \in \text{cf} \\
p_1 + p_2 & \mapsto S(p_1) \cup S(p_2)
\end{align*}
\]

\( p \) is a *minimal sumnormal form* \((p \in msnf)\) is defined by:

\[
\begin{align*}
\text{cf} & \subseteq \text{msnf} \\
p_1 + p_2 & \in \text{msnf} \quad \text{if} \quad p_1, p_2 \in \text{msnf} \text{ and } S(p_1) \cap S(p_2) = \emptyset
\end{align*}
\]

We denote the set of *syntactic “deterministic” prefixes* of a term \( p \) by \( \mathcal{P}(p) \). Formally:

\( \mathcal{P}: \text{PL} \rightarrow \mathcal{P}(\text{cf}) \) is defined by:

\[
\begin{align*}
\text{NIL} & \mapsto \{\text{NIL}\} \\
a.p & \mapsto a.\mathcal{P}(p) \cup \{\text{NIL}\} \\
p_1 + p_2 & \mapsto \mathcal{P}(p_1) \cup \mathcal{P}(p_2) \\
p_1 \parallel p_2 & \mapsto \mathcal{P}(p_1) \parallel \mathcal{P}(p_2)
\end{align*}
\]

We define \( p \) is a *prefix form* \((p \in pf)\) by
\[ p \in \text{pf} \]
\[ \text{iff} \quad p \in \text{snf} \text{ and if this is the case } P(p) = S(p). \]

Notice

i) \( \forall p \in \text{snf}. S(p) \neq \emptyset \).

ii) \( p \in \text{msnf} \Rightarrow p \in \text{snf} \).

iii) \( (p \in \text{snf}, (p \equiv a.p' \text{ or } p \equiv p_1 \ || \ p_2)) \Rightarrow p, p', p_1, p_2 \in \text{cf} \).

iv) \( a.P(p) \) in the definition of \( P \) shall be considered as the natural extension of the operator symbol \( a \) to cover sets as well. I.e., \( a.P(p) = \{a.q \mid q \in P(p)\} \).

Whereas the functions in the last definition mapped to sets of terms they will map to tree-semiwords and sets of these in the following.

Definition 4.3.4
We define \( \theta : \text{cf} \rightarrow \text{TSW} \) by:

\[
\begin{align*}
\text{NIL} & \mapsto e \\
a.p & \mapsto a.\theta(p) \\
p_1 \ || \ p_2 & \mapsto \theta(p_1) \ || \ \theta(p_2)
\end{align*}
\]

and \( \Theta : \text{snf} \rightarrow \mathcal{P}(\text{TSW}) \) by:

\[ \Theta(p) := \{\theta(q) \mid q \in S(p)\} = \theta S(p), \]

where \( \theta \) is extended in the natural way to sets.

Notice that the ambiguity arising in using \( a \) both for terms and for semiwords is solved in the definition of \( \theta \) when fixing \( \theta \)'s domain and codomain.

We introduce some notational convenience. We will say that two terms \( p, q \) are sum congruent written \( \vdash p =_s q \) if we can show \( \vdash p = q \) by +1, +2 and the other inference rules. Similar \( \vdash p =_c q \) means that \( \vdash p = q \) can be shown using \( \parallel 1 \) and \( \parallel 2 \) and we say that \( p \) and \( q \) are composition congruent. Often we will omit \( \vdash \) and just write \( p = q \) instead of \( \vdash p = q \). This has as consequence that we also write e.g., \( \vdash p =_s q \) as \( p =_s q \). To avoid confusion we use \( \equiv \) for syntactic equality between terms instead of =. Furthermore \( p =_i q \) (for idempotent) means only +4 together with the other inference rules are used in the proof of \( p = q \) and \( p =_n q \) (\( n \) for neutrality w.r.t. \( \parallel \)\) that only \( \parallel 3 \) is used. We will also use
combinations of these as e.g., \( p =_s q \).

Finally for \( A; B \in \mathcal{P}(cf) \) we let \( A \subseteq_{nc} B \) denote \( \forall p \in A \exists q \in B. \vdash p =_{nc} q \).

Most of the following proofs will be induction on the structure of some closed term \( p \) considered as a member of \( PL \) or as being a composition-, sumnormal- or minimal sumnormal form. We will then often just list the different cases to consider—of course starting with the basic cases.

But we will have some proofs which are by induction on the proof of some statement \( p \leq q \)—actually on the length of the proof. To this end the following general lemma is useful.

**Lemma 4.3.5** Let \( DED(E) \) be a proof system of inequations \( E \). Furthermore let \( P \) be a proof of \( t \leq t' \). Then there exists a proof \( P' \) of \( t \leq t' \) with the simple instantiation property.

**Proof** We will use induction on the length, \( |P| \), of the proof \( P \) of \( t \leq t' \).

\( |P| = 1 \): Then \( t \leq t' \) is an inequation of \( E \) or \( t = t' \) and we cannot have used instantiation, so we can let \( P' := P \).

\( |P| > 1 \): Assume \( |P| = n \) and the proof \( P \) is

\[ t_1 \leq t'_1, t_2 \leq t'_2, \ldots, t_n \leq t'_n \]

Now look at the last inference rule used to obtain \( t_n \leq t'_n \). Two cases depending on whether \( t_n \leq t'_n \) is obtained by instantiation or not.

No instantiation used:

Assume \( t_n \leq t'_n \) is obtained from \( t_{i_1} \leq t'_{i_1}, \ldots, t_{i_k} \leq t'_{i_k}, i_l \in \mathbb{N} \) by some inference rule. Since \( |P_{t_{i_l}} \leq t'_{i_l}| < |P| = n \) for \( l \in \mathbb{N} \) we can use the hypothesis of induction to find proofs \( P'_{t_{i_l} \leq t'_{i_l}} \) of \( t_{i_l} \leq t'_{i_l} \) with the simple instantiation property. Let

\[ P' := P'_{t_{i_1} \leq t'_{i_1}}, \ldots, P'_{t_{i_k} \leq t'_{i_k}}, t_n \leq t'_n \]

where the last statement is obtained by the inference rule. Clearly \( P' \) is a proof of \( t_n \leq t'_n \) with the desired property.

Instantiation used:

Here we have two subcases.

\( t_n \leq t'_n \) is an instantiation of an axiom.

Assume this axiom is \( t_j \leq t'_j, j \in \mathbb{N} \). We see that

\[ P' := t_j \leq t'_j, t_n \leq t'_n \]

is a proof of \( t_n \leq t'_n \) with the simple instantiation property.

\( t_n \leq t'_n \) is an instantiation, but not of an axiom.

Assume it is a \( \rho \)-instantiation of \( t_j \leq t'_j, j \in \mathbb{N} \), i.e., \( t_n \equiv t_j \rho, t'_n \equiv t'_j \rho \). Since \( t_j \leq t'_j \) is not an axiom some inference rule must have been used to derive \( t_j \leq t'_j \). We look at the different possibilities.
transitivity: Assume $t_j \leq t_j'$ is obtained from $t_k \leq t_k'$ and $t_1 \leq t_1'$ where $t_j \equiv t_k$, $t_j' \equiv t_k'$, $t_1 \equiv t_1$ and $k, l \in j - 1$. Then $P_{i_1 \leq t_1'} \leq t_1' \rho$ and $P_{t_1' \leq t_1'} \leq t_1' \rho$, where the last statement is obtained by a $\rho$-instantiation, are proofs of $t_1 \rho \leq t_1' \rho$ and $t_1' \rho \leq t_1' \rho$. Now $k, l < j < n$ implies $k + 1, l + 1 < n$ so the length of these proofs are less than $n$ wherefore we can use the hypothesis to get proofs $P_{t_1' \rho \leq t_1' \rho}$ and $P_{t_1' \rho \leq t_1' \rho}$ with the property. Then since $t_1' \rho \equiv t_1 \rho$

$$P' := P_{t_1 \rho \leq t_1' \rho}, P_{t_1' \rho \leq t_1' \rho}, t_1 \rho \leq t_1' \rho$$

is a proof of $t_n \leq t_1'$ with the property.

substitutivity (congruence): Suppose $t_j \equiv f(t_{i_1}, ..., t_{i_k})$, $t_j' \equiv f(t_{i_1}', ..., t_{i_k}')$ for some $f$ in $\Sigma$ of rank $k$ and that $t_j \leq t_j'$ is obtained from $t_{i_1} \leq t_{i_1}', ..., t_{i_k} \leq t_{i_k}'$. Similar as in the case of transitivity we by substitutivity find a proof

$$P' := P_{t_{i_1} \rho \leq t_{i_1}' \rho}, ..., P_{t_{i_k} \rho \leq t_{i_k}' \rho}, f(t_{i_1} \rho, ..., t_{i_k} \rho) \leq f(t_{i_1}' \rho, ..., t_{i_k}' \rho)$$

of $f(t_{i_1} \rho, ..., t_{i_k} \rho) \leq f(t_{i_1}' \rho, ..., t_{i_k}' \rho)$ with the simple instantiation property. Now since substitution $\rho$ is a homomorphism we see $f(t_{i_1} \rho, ..., t_{i_k} \rho) \equiv (f(t_{i_1}, ..., t_{i_k}) \rho) \equiv t_j \rho \equiv t_n$ and similar $f(t_{i_1}' \rho, ..., t_{i_k}' \rho) \equiv t_n$. So $P'$ is actually a proof of $t_n \leq t_n'$.

Instantiation: Assume $t_j \leq t_j'$ is obtained by $\rho'$-instantiation of $t_k \leq t_k'$, $k \in j - 1$. I.e., $t_j \equiv t_k \rho'$ and $t_j' \equiv t_k' \rho'$. We have a proof $P'_{t_k \rho \leq t_k'}$ for $t_k \leq t_k'$. The length of the proof

$$P_{t_k \leq t_k'}, t_k \rho \circ \rho' \leq t_k' \rho \circ \rho'$$

(where the last statement is a $\rho \circ \rho'$-instantiation of $t_k \leq t_k'$) is less than or equal to $k + 1$. Since $k + 1 \leq j < n$ we can use the hypothesis to find a proof $P'_{t_k \rho \circ \rho' \leq t_k' \rho \circ \rho'}$ with the simple instantiation property. By the Substitution lemma (of Hennessy [Hen85a]) $t_k \rho \circ \rho' \equiv (t_k \rho') \rho \equiv t_\rho \equiv t_n$ and similar $t_k' \rho \circ \rho' \equiv t_n'$. So

$$P' := P_{t_k \rho \circ \rho' \leq t_k' \rho \circ \rho'}$$

is a proof of $t_n \leq t_n'$ with the desired property.

The advantage of this lemma is that when proving some property on the basis of the length of a proof of a statement $p \leq q$ where $p, q \in PL$ we can assume that the proof has the simple instantiation property. Since $p$ and $q$ are closed terms the instantiation must be closed too. This means that we can leave out instantiation in our considerations if we instead consider closed instantiations of the axioms when dealing with these.

The first lemma shows that an action prefix of a sumnormal form within the proof system can be distributed over the summands thereby obtaining a sumnormal form.

**Lemma 4.3.6** $p' \in \text{snf} \Rightarrow \exists p \in \text{snf.} \quad \vdash p = a.p', S(p) = a.S(p')$

**Proof** Induction on the structure of $p$ considered as a sumnormal form.
Lemma 4.3.8  Let $p' \in \text{cf}$: Let $p \equiv a.p'$. Reflexivity gives $\vdash p \equiv a.p' = a.p'$. We also have $p' \in \text{cf} \Rightarrow p \equiv a.p' \in \text{cf}$ and thereby $p \in \text{snf}$. So since $S(p') = \{p'\}$ we have $S(p) = \{p\} = \{a.p'\} = a.\{p'\} = a.S(p').$

$p' \equiv p'_1 + p'_2, p'_1, p'_2 \in \text{snf}$: By hypothesis of induction $\exists p_i \in \text{snf}$. $\vdash p_i = a.p'_i, S(p_i) = a.S(p'_i)$ for $i \in 2$. Let $p \equiv p_1 + p_2$. We have:

$\vdash p \equiv p_1 + p_2 = a.p'_1 + a.p'_2$ by congruence

$= a.(p'_1 + p'_2)$ by $+$

$= a.p'$

It only remains to show $S(p) = a.S(p')$. Notice $p \in \text{snf}$ because $p_1, p_2 \in \text{snf}$. So $S(p) = S(p_1 + p_2) = S(p_1) \cup S(p_2) = a.S(p'_1) \cup a.S(p'_2) = a.(S(p'_1) \cup S(p'_2)) = a.S(p'_1 + p'_2) = a.S(p')$.

$\square$

In the next lemma a composition form parallel composed with a sumnormal form is distributed in over the summands.

Lemma 4.3.7  $p_1 \in \text{cf}, p_2 \in \text{snf} \Rightarrow \exists p \in \text{snf}. \vdash p = p_1 \parallel p_2, S(p) = \{p_1\} \parallel S(p_2)$.

Proof  The proof is here by induction on the structure of $p_2$

$p_2 \in \text{cf}$: Let $p \equiv p_1 \parallel p_2$. We have $\vdash p \equiv p_1 \parallel p_2 = p_1 \parallel p_2$.

$p_1, p_2 \in \text{cf} \Rightarrow p_1 \parallel p_2 \in \text{cf} \subseteq \text{snf}$ and thereby also $p \in \text{cf}$. As $p, p_2 \in \text{cf}$ we have $S(p) = \{p\}$ and $S(p_2) = \{p_2\}$. Then $S(p) = \{p\} = \{p_1 \parallel p_2\} = \{p_1\} \parallel \{p_2\} = \{p_1\} \parallel S(p_2)$.

$p_2 \equiv q_1 + q_2, q_1, q_2 \in \text{snf}$: By hypothesis of induction:

$\exists p'_1 \in \text{snf}. \vdash p'_1 = p_1 \parallel q_i, S(p'_1) = \{p_1\} \parallel S(q_i)$ for $i \in 2$

$\vdash p \equiv p'_1 + p'_2$. We have: $\vdash p \equiv p'_1 + p'_2 = p_1 \parallel q_1 + p_2 \parallel q_2$ by congruence

$= p_1 \parallel (q_1 + q_2)$ by $+$

$\equiv p_1 \parallel p_2$

Notice $p \in \text{snf}$, because $p'_1, p'_2 \in \text{snf}$, so $S$ is defined on $p$. Then $S(p) = S(p'_1) \cup S(p'_2) = (\{p_1\} \parallel S(q_1)) \cup (\{p_2\} \parallel S(q_2)) = \{p_1\} \parallel (S(q_1) \cup S(q_2)) = \{p_1\} \parallel S(p_2)$.

$\square$

The last lemma easily generalize to sumnormal forms.

Lemma 4.3.8  $p_1, p_2 \in \text{snf} \Rightarrow \exists p \in \text{snf}. \vdash p = p_1 \parallel p_2, S(p) = S(p_1) \parallel S(p_2)$.

Proof  

$p_1 \in \text{cf}$: Use the last lemma to find $p$. From $p_1 \in \text{cf}$ it follows $S(p_1) = \{p_1\}$, so $S(p) = \{p_1\} \parallel S(p_2) = S(p_1) \parallel S(p_2)$.

$p_1 \equiv q_1 + q_2, q_1, q_2 \in \text{snf}$: We can use the hypothesis of induction on $q_1, q_2$ to get $q'_1, q'_2 \in \text{snf}$ such that $\vdash q'_i = q_i \parallel p_2$ and $S(q'_i) = S(q_i) \parallel S(p_2)$ for $i \in 2$. Let $p \equiv q'_1 + q'_2$. Then:
\[ \vdash p \equiv q_1 + q_2 = q_1 \parallel p_2 + q_2 \parallel p_2 \] by congruence
\[ =_c p_2 \parallel q_1 + p_2 \parallel q_2 \]
\[ = p_2 \parallel (q_1 + q_2) \] by \[\parallel+\]
\[ \equiv p_2 \parallel p_1 =_c p_1 \parallel p_2 \]

Furthermore \[S(p) = S(q_1') \cup S(q_2') = (S(q_1) \parallel S(p_2)) \cup (S(q_2) \parallel S(p_2)) = (S(q_1) \cup S(q_2)) \parallel S(p_2) = S(p_1) \parallel S(p_2).\]

\[ \square \]

We are now in a position to show that any term can be reduced to a sum normal form.

**Proposition 4.3.9** \( p \in PL \Rightarrow \exists q \in \text{snf}. \vdash p = q. \)

**Proof** We use induction on the structure of \( p \) considered as a member of \( PL. \)

\( p \equiv NIL: NIL \in \text{snf} \) by definition so \( q \equiv NIL \) and we have the result by reflexivity.

\( p \equiv a.p': \) By hypothesis of induction \( \exists q' \in \text{snf}. \vdash p' = q'. \) By congruence \( \vdash p \equiv a.p' = a.q'. \)

From lemma 4.3.6 we find a \( q \in \text{snf} \) such that \( \vdash q = a.q' \) and the result follows by transitivity.

\( p \equiv p_1 + p_2: \exists q_i \in \text{snf}. \vdash q_i = p_i \) for \( i \in \mathbb{2} \) by hypothesis of induction. Let \( q \equiv q_1 + q_2. \) By congruence then \( \vdash p \equiv p_1 + p_2 = q_1 + q_2 \equiv q. \)

\( p \equiv p_1 \parallel p_2: \) From the hypothesis we get \( \exists q_i \in \text{snf}. \vdash q_i = p_i \) for \( i \in \mathbb{2}. \) By congruence \( \vdash q_1 \parallel q_2 = p_1 \parallel p_2 \equiv p. \) The result then follows from lemma 4.3.8.

\[ \square \]

The next lemma merely states that no extra terms are gained by applying \( P \) more than once.

**Lemma 4.3.10** For \( p \in PL \) we have \( PP(p) = P(p). \)

If \( D \) is a set of terms, \( PD, \) is as usual to understand as the natural extension of \( P \) (defined on single terms) to sets of terms. We will write \( P(p) \) for a single term and e.g., \( P\{p\} \) when considering sets of terms.

**Proof** By induction on the structure of \( p \in PL. \)

\( p \equiv NIL: P(p) = P(NIL) = \{NIL\} = P\{NIL\} = PP(p). \)

\( p \equiv a.p': \) Here we have:

\[ P(p) = P(a.p') = \{NIL\} \cup a.P(p') \]
\[ = \{NIL\} \cup a.P(p') \tag{by hypothesis of induction} \]
\[ = \{NIL\} \cup \bigcup_{q \in P(p')} a.P(q) \]
\[ = \{NIL\} \cup \bigcup_{q \in P(p')} (a.P(q) \cup \{NIL\}) \]
\[ = P\{NIL\} \cup \bigcup_{q \in P(p')} a.P(q) \]
\[ = \bigcup_{q \in P\{a.p'\} \cup \{NIL\}} P(q) \]
\[ = PP(a.p') = PP(p). \]
\[ p \equiv p_1 + p_2: \, P(p) = P(p_1 + p_2) = P(p_1) \cup P(p_2) = (\text{by hypothesis}) \, PP(p_1) \cup PP(p_2) = P(P(p_1) \cup P(p_2)) = PP(p_1 + p_2) = PP(p). \]

\[ p \equiv p_1 \parallel p_2: \text{ We deduce:} \]
\[ P(p) = P(p_1) \parallel P(p_2) \]
\[ = PP(p_1) \parallel PP(p_2) \quad (\text{by hypothesis}) \]
\[ = \{ p'_1 \parallel p'_2 \mid (p'_1, p'_2) \in PP(p_1) \times PP(p_2) \} \]
\[ = \{ p'_1 \parallel p'_2 \mid (p'_1, p'_2) \in \bigcup_{(q_1, q_2) \in P(p_1) \times P(p_2)} P(q_1) \times P(q_2) \} \]
\[ = \bigcup_{(q_1, q_2) \in P(p_1) \times P(p_2)} P(q_1) \parallel P(q_2) \]
\[ = \bigcup_{q \in P(p_1) \parallel P(p_2)} P(q) \]
\[ = \bigcup_{q \in P(p_1) \parallel P(p_2)} P(q) = PP(p). \]

\[ \square \]

**Lemma 4.3.11** \( p \in \text{snf} \Rightarrow P(p) = PS(p). \)

**Proof** Induction on the structure of \( p \) considered as a sumnormal form.

\( p \in \text{cf} \): Then \( S(p) = \{ p \} \) wherefore \( P(p) = P\{ p \} = PS(p). \)

\( p \equiv p_1 + p_2, \, p_1, p_2 \in \text{snf} \): We see \( P(p) = P(p_1) \cup P(p_2) = (\text{by hypothesis}) \, PS(p_1) \cup PS(p_2) = P(S(p_1) \cup S(p_2)) = PS(p_1 + p_2) = PS(p). \)

\[ \square \]

From the last to lemmas we get:

**Lemma 4.3.12** If \( p \in \text{snf} \) and there exists a \( q \in PL \) such that \( S(p) = P(q) \) then \( p \in \text{pf} \).

That is if \( p \) is a sumnormal form with summands equal to the prefixes of some other term then the summands of \( p \) are already closed under prefix.

**Proof** \( p \in \text{snf} \) so we only have to show \( S(p) = P(p) \). But this is easily seen:
\[ S(p) = P(q) \quad \text{by assumption} \]
\[ = PP(q) \quad \text{by lemma 4.3.10} \]
\[ = PS(p) \quad \text{by assumption again} \]
\[ = P(p) \quad \text{by lemma 4.3.11 and the fact that } p \in \text{snf}. \]

\[ \square \]

**Lemma 4.3.13** \( p' \in \text{pf} \Rightarrow \exists p \in \text{pf}. \vdash p = a.p' \).

Similar as the lemmas leading to proposition 4.3.9 we shall now see how different operators can be distributed in over prefix forms to obtain new prefix forms.

**Proof** As \( p' \in \text{pf} \) implies \( p' \in \text{snf} \) we can use lemma 4.3.6 to find a \( q \in \text{snf} \) such that
\[ \vdash q = a.p' \quad \text{and} \]
\[ (4.6) \quad S(q) = a.S(p') \]

99
Now let \( p \equiv q + \text{NIL} \). By +3 we have \( p \equiv q + \text{NIL} = q = a.p' \). It just remains to show \( p \in \text{pf} \). Notice that because \( q \in \text{snf} \) we have \( p \equiv q + \text{NIL} \in \text{snf} \). We show \( S(p) = P(a,p') \).

\[
S(p) = S(q) \cup S(\text{NIL}) \\
= a.S(p') \cup \{\text{NIL}\} \quad \text{by (4.6)} \\
= a.P(p') \cup \{\text{NIL}\} \quad \text{because } p' \in \text{pf} \\
= P(a,p')
\]

Since \( p \in \text{snf} \) we can deduce \( p \in \text{pf} \) from \( S(p) = P(a,p') \) and lemma 4.3.12.

\[\square\]

Lemma 4.3.14 \( p_1, p_2 \in \text{pf} \) implies \( \exists p \in \text{pf}. \vdash p = p_1 \| p_2. \)

\[\textbf{Proof} \quad p_1, p_2 \in \text{pf} \) implies \( p_1, p_2 \in \text{snf} \) so we can use lemma 4.3.8 to find a \( p \in \text{snf} \) such that \( \vdash p = p_1 \| p_2 \) and

\[
\text{(4.7)} \quad S(p) = S(p_1) \| S(p_2)
\]

To get the result it remains to show \( p \in \text{pf} \). Since \( S(p) = (\text{by (4.7)}) S(p_1) \| S(p_2) = (\text{because } p_1, p_2 \in \text{pf}) P(p_1) \| P(p_2) = P(p_1 \| p_2) \) and \( p \in \text{snf} \) we obtain \( p \in \text{pf} \) by lemma 4.3.12.

\[\square\]

Lemma 4.3.15 \( p \in \text{cf} \) implies \( \exists q \in \text{pf}. \vdash p = q. \)

\[\textbf{Proof} \quad \text{Induction on the structure of } p \text{ considered as a composition form.}\]

\( p \equiv \text{NIL}: \) Let \( q \equiv \text{NIL} \). \( P(\text{NIL}) = \{\text{NIL}\} = S(\text{NIL}) \) and by reflexivity \( \vdash p \equiv \text{NIL} = \text{NIL} = q.\)

\( p \equiv a.p', p' \in \text{cf}: \) By hypothesis there exists a \( q' \in \text{pf}. \vdash p' = q'. \) Using congruence we get \( \vdash p \equiv a.p' = a.q' \). As \( q' \in \text{pf} \) lemma 4.3.13 gives us a \( q \in \text{pf} \) such that \( \vdash q = a.q' \). We see \( \vdash p = q \) and \( q \in \text{pf}, \)

\( p \equiv p_1 \| p_2, p_1, p_2 \in \text{cf}: \) There exists \( p'_1, p'_2 \in \text{pf} \) such that \( \vdash p_1 = p'_1 \) and \( \vdash p_2 = p'_2 \) by hypothesis of induction. By congruence then \( \vdash p \equiv p_1 \| p_2 = p'_1 \| p'_2. \) As \( p'_1, p'_2 \in \text{pf} \) lemma 4.3.14 gives us a \( q \in \text{pf} \) with \( \vdash q = p'_1 \| p'_2 \) from which the result follows.

\[\square\]

It now easily follows that any sumnormal form can be reduced to a prefix form.

Proposition 4.3.16 \( p \in \text{snf} \) implies \( \exists q \in \text{pf}. \vdash p = q. \)

\[\textbf{Proof} \quad \text{Induction on the structure of } p \text{ considered as a sumnormal form.}\]

\( p \in \text{cf}: \) By the last lemma.

\( p \equiv p_1 + p_2, p_1, p_2 \in \text{snf}: \) By hypothesis and congruence we find \( q_1, q_2 \in \text{pf} \) such that \( \vdash p \equiv p_1 + p_2 = q_1 + q_2. \) Let \( q \equiv q_1 + q_2, q_1, q_2 \in \text{pf} \Rightarrow q_1, q_2 \in \text{snf} \) wherefore \( q \in \text{snf} \). We just have to show \( S(q) = P(q) \) in order to have \( q \in \text{pf}. \)

\[
S(q) = S(q_1) \cup S(q_2) \\
= P(q_1) \cup P(q_2) \quad \text{because } q_1, q_2 \in \text{pf} \\
= P(q_1 + q_2) = P(q).
\]
With the next lemma it is possible (by commutativity and associativity) to bring any
summand of a sumnormal form, \( p \), to the front of \( p \).

**Lemma 4.3.17** Let \( p \in \text{snf} \). Then \( q \in S(p) \) implies \( p \equiv q \) or \( \exists q' \in \text{snf} \). \( p =_s q + q' \).

**Proof**

\( p \in \text{cf} \): In this case we have \( S(p) = \{ p \} \) so \( q \in \{ p \} \) implies \( p \equiv q \).

\( p \equiv p_1 + p_2, p_1, p_2 \in \text{snf} \): Here we have \( S(p) = S(p_1) \cup S(p_2) \) so \( q \in S(p) \) gives us two cases
to consider.

\( q \in S(p_1) \): Using the hypothesis of induction we also have two possibilities to consider here. \( p_1 \equiv q_2 \): Chose \( q' \equiv p_2 \). By reflexivity \( p \equiv p_1 + p_2 \equiv q + q' \).

\( \exists q'' \in \text{snf} \): \( \vdash p_1 =_s q + q'' \): Chose \( q' \equiv q'' + p_2 \). We have:

\[ q \equiv q + (q'' + p_2) \]

\[ \vdash q + q' \]

\( q' \in \text{snf} \) follows from \( q'' \in \text{snf} \) and \( p_2 \in \text{snf} \).

\( q \in S(p_2) \): Similar but with additional use of +2.

Any two sumnormal forms which are equal up to idempotents, commutativity and asso-
ciativity have the same summands. Formally:

**Lemma 4.3.18** Let \( p, q \in \text{snf} \). Then \( \vdash p =_s q \) implies \( S(p) = S(q) \).

**Proof** For the purpose of this proof it is convenient to extend \( S \) defined on \( \text{snf} \) to \( S' \) defined on \( \text{PL} \). Let \( S' : \text{PL} \rightarrow \mathcal{P}(\text{cf}) \) be defined by:

\[
\begin{align*}
NIL & \rightarrow \{ \varepsilon \} \\
a.p & \rightarrow a.S'(p) \\
p_1 + p_2 & \rightarrow S'(p_1) \cup S'(p_2) \\
p_1 \parallel p_2 & \rightarrow S'(p_1) \parallel S'(p_2)
\end{align*}
\]

A number of subproofs are necessary, but they are all inductive and quite trivial so we only give the principal line.

\( S' \) well-defined i.e., \( S'(p) \in \text{cf} \) for all \( p \in \text{PL} \) is proved by induction on the structure of \( p \).
That \( S' \) coincide with \( S \) on \( \text{snf} \) is shown by first proving

\[ p \in \text{cf} \Rightarrow S'(p) = S(p) \tag{4.8} \]

by induction on the structure of \( p \) considered as a member of \( \text{cf} \) and next

\[ p \in \text{snf} \Rightarrow S'(p) = S(p) \]
also by structural induction, but this time with \( p \) considered as a sumnormal form, using (4.8) in the basis.

Finally for \( p, q \in PL \):

\[
\vdash p =_{si} q \Rightarrow S'(p) = S'(p)
\]

is shown by induction on the number of inferences used to prove \( p =_{si} q \). Since \( S' \) coincide with \( S \) on \( \text{snf} \) the lemma then follows as a special case.

The reason that we need the extension is that if we tried to prove the lemma directly by induction on the inferences we would get the following problem by transitivity:

If \( p, q \in \text{snf} \) and \( p =_{si} q \) was proved using \( p =_{si} r \) and \( r =_{si} q \) we cannot be sure that \( r \in \text{snf} \) and therefore could not use the hypothesis. \( \Box \)

By the definition of \( \Theta \) we then have:

**Corollary 4.3.19** If \( p, q \in \text{snf} \) then \( \vdash p =_{si} q \) implies \( \Theta(p) = \Theta(q) \).

**Lemma 4.3.20** \( p \in \text{msnf} \) and \( \vdash p =_{s} q \) implies \( q \in \text{msnf} \).

**Proof** Shown by induction on the number of inferences used to infer \( p =_{s} q \). At first we consider the axioms.

Reflexivity: Evident since then \( p = q \).

\( +1: \) \( p \equiv p_1 + (p_2 + p_3) = (p_1 + p_2) + p_3 \equiv q \). \( p \in \text{msnf} \) implies \( p_1, p_2, p_3 \in \text{msnf} \),

\( S(p_1) \cap S(p_2 + p_3) = S(p_1) \cap (S(p_2) \cup S(p_3)) = \emptyset \) and \( S(p_2) \cap S(p_3) = \emptyset \). Clearly then also \( S(p_1 + p_2) \cap S(p_3) = \emptyset \) and \( S(p_1) \cap S(p_2) = \emptyset \) so \( q \in \text{msnf} \).

\( +2: \) Similar arguments using \( \cap \) commutative.

Inferences:

Transitivity: Assume \( \vdash p =_{s} q \) is obtained from \( \vdash p =_{s} r \) and \( \vdash r =_{s} q \). Using the hypothesis of induction on \( \vdash p =_{s} r \) we have \( r \in \text{msnf} \) so we can use the induction once more to get \( q \in \text{msnf} \).

Congruences:

\( a. \) Assume \( p' =_{s} q' \) is used to show \( p \equiv a.p' =_{s} a.q' \equiv q \). Now \( p \equiv a.p' \in \text{msnf} \) implies \( p' \in \text{cf} \). Since \( p' =_{s} q' \) it then follows that \( q' \in \text{cf} \) and thereby \( q \equiv a.q' \in \text{cf} \). Hence \( q \in \text{msnf} \).

\( +. \) W.l.o.g. assume \( p' =_{s} q' \) is used to infer \( p \equiv p' + r =_{s} q' + r \equiv q \). \( p \equiv p' + r \in \text{msnf} \) implies \( p', r \in \text{msnf} \) and \( S(p') \cap S(r) = \emptyset \). By hypothesis of induction we then from \( p' =_{s} q' \) get \( q' \in \text{msnf} \). Then since \( \text{msnf} \subseteq \text{snf} \): \( p' =_{s} q' \) by lemma 4.3.18 implies \( S(p') = S(q') \). Hence \( S(q') \cap S(r) = \emptyset \) and we conclude \( q \in \text{msnf} \).

\( \|: \) Similar arguments as in the \( a.-\)case. \( \Box \)

We now show that any sumnormal form can be minimalized.

**Proposition 4.3.21** For \( p \in \text{snf} \) there exists a \( q \in \text{msnf} \) such that \( \vdash p =_{si} q \).
Proof  At first we prove:

\[(4.9) \quad q_1, q_2 \in \text{msnf} \Rightarrow \exists q \in \text{msnf}. \vdash q_1 + q_2 =_{si} q \]

by induction on the number \( n = |S(q_1) \cap S(q_2)| \)

\( n = 0: \) I.e., \( S(q_1) \cap S(q_2) = \emptyset \) and \( q_1, q_2 \in \text{msnf} \). Let \( q \equiv q_1 + q_2 \). Clearly \( q \in \text{msnf} \) and by reflexivity \( \vdash q_1 + q_1 = q \).

\( n > 0: \) Chose a \( q' \in S(q_1) \cap S(q_2) \). By lemma 4.3.17 we obtain for \( i \in 2: \) \( q_i \equiv q' \land \exists q'_i \in \text{snf}. \vdash q_i =_{s} q' + q'_i \), so there are four subcases to consider.

\( q_1 \equiv q': \) We then also have \( q_1 =_{s} q' + q'_2 \). By congruence \( q_1 + q_2 =_{s} q' + q'_2 + q'_2 = q \), the result follows since \( q_2 \in \text{msnf} \).

\( q_1 \equiv q', (\exists q'_2 \in \text{snf}. \vdash q_2 =_{s} q' + q'_2): \) We get \( \vdash q_1 + q_2 =_{s} q' + q'_2 + q'_2 =_{s} q' + q'_2 + q'_2 =_{s} q_1 + q_2 \). By congruence \( q_1 + q_2 =_{s} q \), the result follows since \( q_2 \in \text{msnf} \).

\( (\exists q'_1 \in \text{snf}. \vdash q_1 =_{s} q' + q'_1), q_2 =_{s} q' \): Symmetric.

\( \exists q'_1 \in \text{snf}. \vdash q_1 =_{s} q' + q'_1 \) for \( i \in 2: \) We get \( \vdash q_1 + q_2 =_{s} (q' + q'_1) + (q' + q'_2) =_{s} (q' + q') + (q'_1 + q'_2) =_{s} (q' + q'_1) + q'_2 =_{s} q_1 + q_2 \). According to lemma 4.3.20 we have \( q'_1 + q'_2 \in \text{msnf} \) because \( q_2 \in \text{msnf} \) and \( \vdash q_2 =_{s} q' + q'_2 \). \( q' + q'_2 \in \text{msnf} \) gives us \( S(q') \cap S(q'_2) = \emptyset \). Hence \( |S(q_1) \cap S(q'_2)| < n \). \( q' + q'_2 \in \text{msnf} \) also gives \( q'_2 \in \text{msnf} \) so we can use the hypothesis of induction on \( q_1, q'_2 \) to find a \( q \in \text{msnf} \) such that \( \vdash q_1 + q'_2 =_{s} q \). All together we then have \( \vdash q_1 + q'_2 =_{si} q \) thereby completing the inductive step in proving (4.9).

With (4.9) we can now prove the proposition by induction on the structure of \( p \) considered as a sumnormal form.

\( p \in cf: \) We then also have \( p \in \text{msnf} \) and can chose \( q \equiv p \).

\( p \equiv p_1 + p_2, p_1, p_2 \in \text{snf}: \) Using the hypothesis of induction on \( p_1, p_2 \) we can find \( q_1, q_2 \in \text{msnf} \) such that \( \vdash p_1 =_{si} q_1 \) and \( \vdash p_2 =_{si} q_1 \). By congruence \( \vdash p =_{si} q_1 + q_2 \) and from (4.9) we find a \( q \in \text{msnf} \). \( \vdash q_1 + q_2 =_{si} q \). The result then follows by transitivity.

The next lemma establish the first connection to the denotations of terms.

Lemma 4.3.22 For \( p \in PL \) we have \([p]_\pi = \theta P(p)\).

Proof  Induction on the structure of \( p \) considered as a member of \( PL \).

\( p \equiv NIL: \) We see \([p]_\pi = NIL_\pi = \{\varepsilon\} = \{\theta (NIL)\} = \theta \{NIL\} = \theta P(NIL)\).

\( p \equiv a. p': \) \([p]_\pi = a.[p']_\pi \cup \{\varepsilon\} = (\text{induction and the definition of } \theta) \ a. \theta P(p') \cup \{\theta (NIL)\} = \{a.\theta (q) \mid q \in P(p') \cup \{\theta (NIL)\}\} = (\text{definition of } \theta) \ \{\theta (a.q) \mid q \in P(p') \cup \{\theta (NIL)\}\} = \theta a. P(p') \cup \{\theta (NIL)\} = \theta (a. P(p') \cup \{NIL\}) = \theta P(a.p') = \theta P(p)\).

\( p \equiv p_1 + p_2: \) \([p]_\pi = [p_1]_\pi \cup [p_2]_\pi = (\text{hypothesis of induction}) \theta P(p_1) \cup \theta P(p_2) = (\text{because } \theta \text{ is extended to sets in the natural way}) \ \theta (P(p_1) \cup P(p_2)) = \theta P(p)\).
Corollary 4.3.23 If \( p \in \text{pf} \) then \( \lbrack p \rbrack_\pi = \Theta(p) \).

Lemma 4.3.24 If \( \vdash p = q' \) and \( q' \in S(q) \) for a \( q \in \text{snf} \) then \( \vdash q = p + q \).

Proof By lemma 4.3.17 it is enough to consider the following two possibilities:

\( q' \equiv q \): By +4 we have \( \vdash q \equiv q' = q' + q' \equiv q' + q \). As \( \vdash p = q' \) we get by congruence \( \vdash q' + q = p + q \) from which the result follows.

\( \exists q'' \in \text{snf}. \vdash q = s \cdot q'' + q'' \): Again by +4 we have \( \vdash q' = q' + q' \) so by congruence \( \vdash q = (q' + q') + q'' = s \cdot q' + (q' + q') = s \cdot q' + q \). From \( \vdash p = q' \) and congruence we now get \( \vdash q' + q = p + q \) and thereby \( \vdash q = p + q \).

Proposition 4.3.25 Suppose \( p \in \text{msnf} \) and \( q \in \text{snf} \). Then \( S(p) \subseteq_{nc} S(q) \) implies \( \vdash p + q = q \).

Proof Induction on the quantity \( n = \lvert S(p) \rvert \).

\( n = 1 \): From \( p \in \text{msnf} \) follows \( S(p) = \{ p \} \). Now \( \{ p \} \subseteq_{nc} S(q) \) means \( \exists q' \in S(q). \vdash p =_{nc} q' \). The result then follows from the last lemma.

\( n > 1 \): Chose a \( p_1 \in S(p) \). As \( \lvert S(p) \rvert > 1 \) lemma 4.3.17 gives us that there exists a \( p_2 \in \text{snf} \) such that \( \vdash p = s \cdot p_1 + p_2 \). By lemma 4.3.20 we get \( p_1 + p_2 \in \text{msnf} \) from \( p \in \text{msnf} \). According to the definition of \( \text{msnf} \) we then also have \( p_1, p_2 \in \text{msnf} \) and \( S(p_1) \cap S(p_2) = \emptyset \). Since \( S(p_1) \neq \emptyset \neq S(p_2) \) then \( \lvert S(p_1) \rvert < \lvert S(p_1) \cup S(p_2) \rvert = \lvert S(p_1 + p_2) \rvert = \lvert S(p) \rvert = n \) for \( i \in \mathbb{Z} \). Because \( p_1, p_2 \in \text{msnf} \) and \( S(p_1), S(p_2) \subseteq S(p_1) \cup S(p_2) = S(p) \subseteq_{nc} S(q) \) the hypothesis of induction gives us

\[
\vdash p_1 + q = q \quad \text{and} \quad \vdash p_2 + q = q.
\]

Then \( \vdash p + q = s \cdot (p_1 + p_2) + q \\
= (p_1 + p_2) + (q + q) \quad \text{by +4} \\
= (p_1 + q) + (p_2 + q) \\
= q + q \quad \text{by congruence and (4.10) by +4} \\
= q
\]

Lemma 4.3.26 \( \theta(p) = \varepsilon, p \in \text{cf} \Rightarrow p =_{nc} \text{NIL} \)

Proof
Lemma 4.3.27 If \( p \in \text{cf} \) and \( \theta(p) = a.s \) then there exists a \( p' \in \text{cf} \) such that \( \theta(p') = s \) and \( \vdash p =_n a.p' \).

**Proof**

\[ p \equiv \text{NIL}: \theta(p) = \varepsilon \neq a.s \text{ so the premise is not fulfilled.} \]

\[ p \equiv a.q: \theta(p) = b.\theta(q). \text{ From corollary}_T 1.2.3.b \text{ we get } a = b \text{ and } s = \theta(q). \text{ So } p \equiv a.q. \]

Then just chose \( p' \equiv q \). As \( p \in \text{cf} \) it follows that \( p' \equiv q \in \text{cf} \). By reflexivity \( p =_n a.p' \).

\[ p \equiv q \parallel r: \theta(p) = \theta(q) \parallel \theta(r) \text{ so by corollary}_T 1.2.14.a \text{ we then get w.l.o.g. } \theta(q) = a.s \text{ and } \theta(r) = \varepsilon. \]

Now \( p \in \text{cf} \Rightarrow q \in \text{cf} \) so we can use the hypothesis of induction to find a \( p' \in \text{cf} \) such that \( \vdash q =_n a.p' \) and \( \theta(p') = s \). By lemma 4.3.26 \( \theta(r) = \varepsilon \Rightarrow \vdash r =_n \text{NIL} \). Then by congruence and \( \|3\): \( \vdash p \equiv q \parallel r =_n q \equiv \text{NIL} =_n q =_n a.p' \).

Lemma 4.3.28 If \( p \in \text{cf} \) then \( \theta(p) = s \parallel t \) implies that there exists \( p_1, p_2 \in \text{cf} \) such that \( \theta(p_1) = s, \theta(p_2) = t \) and \( \vdash p =_{\text{nc}} p_1 \parallel p_2 \).

**Proof**

\[ p \equiv \text{NIL}: \theta(p) = \varepsilon \text{ and } s \parallel t = \varepsilon \Rightarrow s = \varepsilon = t. \]

Let \( p_1 \equiv p_2 \equiv \text{NIL} \in \text{cf}. \text{ It is seen that } \theta(p_1) = s \text{ and } \theta(p_2) = t. \text{ The result then follows by } \|3\.

\[ p \equiv a.p': \theta(p) = a.\theta(p'). \text{ By corollary}_T 1.2.14.a \text{ we from } a.\theta(p') = s \parallel t \text{ get either} \]

\[ \text{a)} s = a.\theta(p'), t = \varepsilon \text{ or} \]

\[ \text{b)} s = \varepsilon, t = a.\theta(p'). \]

We look at the two possibilities separately.

\[ \text{a)} \text{ Let } p_1 \equiv a.p' \in \text{cf and } p_2 \equiv \text{NIL} \in \text{cf}. \text{ We have } \theta(p_1) = a.\theta(p') = s, \theta(p_2) = \theta(\text{NIL}) = \varepsilon = t \text{ and } \vdash p \equiv p_1 =_n p_1 \parallel \text{NIL} \text{ by } \|3 \equiv p_1 \parallel p_2. \]

\[ \text{b)} \text{ Symmetric with the addition that } \|2 \text{ is used too.} \]

\[ p \equiv q_1 \parallel q_2: \theta(p) = \theta(q_1) \parallel \theta(q_2). \text{ By corollary}_T 1.2.14.b \text{ } \theta(q_1) \parallel \theta(q_2) = s \parallel t \text{ implies that there exists } s', s'', t', t'' \text{ such that } s = s' \parallel s'', t = t' \parallel t'', \theta(q_1) = s' \parallel t', \theta(q_2) = s'' \parallel t''. \]

Using the induction on the last two equations we find \( q_1', q_1'', q_2', q_2'' \in \text{cf} \) such that

\[ (4.11) \quad q_1 =_{\text{nc}} q_1' \parallel q_1'', q_2 =_{\text{nc}} q_2' \parallel q_2'' \]

105
and \( \theta(q'_1) = s'_1, \theta(q''_1) = s''_1, \theta(q'_2) = t'_1, \theta(q''_2) = t''_1 \). Now let \( p_i = q'_i \parallel q''_i, i \in \mathbb{2} \). Evidently
\( \theta(p_1) = \theta(q'_1) \parallel \theta(q''_1) = s'_1 \parallel s''_1 = s \) and similar \( \theta(p_2) = t \). We also have:
\[ p \equiv q_1 \parallel q_2 =_{nc} (q'_1 \parallel q''_1) \parallel (q'_2 \parallel q''_2) \]
by congruence and (4.11)
\[ \equiv p_1 \parallel p_2 \]
As \( q'_1, q''_1, q'_2, q''_2 \in \text{cf} \) it follows that \( p_1, p_2 \in \text{cf} \). This concludes the inductive step.

\[ \square \]

**Proposition 4.3.29** Let \( p, q \in \text{cf} \). Then \( \theta(p) = \theta(q) \) implies \( \vdash p =_{nc} q \).

**Proof**
\[ p \equiv NIL: \quad \theta(p) = \theta(NIL) = \varepsilon. \]
By lemma 4.3.26 it is seen that \( \theta(q) = \varepsilon \) implies \( q =_n NIL \)
and thereby \( \vdash p =_{nc} q \).
\[ p \equiv a.p': \quad \text{By lemma 4.3.27 and } \theta(q) = a.\theta(p') = \theta(p), q \in \text{cf} \text{ we can find a } q' \in \text{cf} \text{ such that } q =_n a.q' \text{ and } \theta(q') = \theta(p'). \]
As \( p \in \text{cf} \Rightarrow p' \in \text{cf} \) and \( q' \in \text{cf} \) we can use the induction to get \( q' =_{nc} p' \). By congruence then \( a.q' =_{nc} a.p' \equiv p \). As \( q =_n a.q' \) we have \( \vdash q =_{nc} p \).
\[ p \equiv p_1 \parallel p_2: \quad \text{From lemma 4.3.28 and } \theta(q) = \theta(p_1) \parallel \theta(p_2), q \in \text{cf} \text{ we see that there exists } q_1, q_2 \in \text{cf} \text{ such that } q =_{nc} q_1 \parallel q_2 \text{ and } \theta(q_1) = \theta(p_1), \theta(q_2) = \theta(p_2). \]
As \( p \in \text{cf} \Rightarrow p_1, p_2 \in \text{cf} \) we can use the hypothesis of induction to get \( \vdash p_i =_{nc} q_i, i \in \mathbb{2} \). By congruence then \( q =_{nc} q_1 \parallel q_2 =_{nc} p_1 \parallel p_2 \equiv p \).

\[ \square \]

It should be noticed that we up til now only have been using the \( \pi \)-inequalities. To emphasise this we will write \( \vdash_\pi p = q \) whenever this is the case. So we could actual rewrite all the previous properties with this addition. If in addition to the \( \pi \)-inequalities also \( \delta.\parallel \) is used in proving \( p = q \) we write \( \vdash_\delta p = q \).

**Proposition 4.3.30** Let \( q \in \text{cf}, \theta(q) = t \) and \( s \in TSW \). Then \( s \prec t \Rightarrow \exists p \in \text{cf}. \vdash_\delta p \leq q, \theta(p) = s \)

**Proof** Recall proposition \( 2.3.44: \)
\[ s \prec t \text{ implies } \exists u \in \gamma(s), D \subseteq \gamma(t), \gamma(s) \setminus \{u\} = \gamma(t) \setminus D \]
and for some \( a, b \in \text{Act}, s', s'', t' \in TSW \) either
\[ a) \quad u = a.(s' \parallel b.s''), D = \{a.s', b.s''\} \]
or
\[ b) \quad u = a.s', D = \{a.t'\}, s' \prec t' \]
This natural suggests to make the proof in the size of \( A_{\theta(s)} \). Letting \( t = \theta(q) \) we see from above that there is two cases to consider.
a) Then \( t \) can be written as \( a.s' \parallel b.s'' \parallel (\parallel \gamma(t) \setminus D) \). Since \( \theta(q) = t \) we can use lemma 4.3.28 to find \( q_1, q_2, q_3 \in \text{cf} \) such that

\[
(4.12) \quad \vdash_\pi q = q_1 \parallel q_2 \parallel q_3
\]

and \( \theta(q_1) = a.s', \theta(q_2) = b.s'', \theta(q_3) = (\parallel \gamma(t) \setminus D) \). We can then by lemma 4.3.27 find \( q_1', q_2' \) such that

\[
(4.13) \quad \vdash_\pi q_1 = a.q_1', \vdash_\pi q_2 = b.q_2'
\]

and \( \theta(q_1') = s', \theta(q_2') = s'' \). Let \( p := a.(q_1' \parallel b.q_2') \parallel q_3 \). We have:

\[
\vdash_\delta p \equiv a.(q_1' \parallel b.q_2') \parallel q_3
\]

\[
= \leq a.q_1' \parallel b.q_2' \parallel q_3 \quad \text{by } \delta \parallel \text{ and congruence}
\]

\[
= \pi q_1 \parallel q_2 \parallel q_3 \quad \text{by (4.13) and congruence}
\]

\[
= \pi q \quad \text{by (4.12)}
\]

We also have \( \theta(p) = \theta(a.(q_1' \parallel b.q_2') \parallel q_3) = a.(\theta(q_1') \parallel b.\theta(q_2') \parallel \theta(q_3)) = a.(s' \parallel b.s'') \parallel (\parallel \gamma(t) \setminus D) = u \parallel (\parallel \gamma(t) \setminus D) = u \parallel (\parallel \gamma(s) \setminus \{u\}) = \parallel \gamma(s) = s \). In the proof of this case we actually did not use the inductive hypothesis.

b) In this case we can write \( t \) as \( a.t' \parallel (\parallel \gamma(t) \setminus \{a.t'\}) \). As in the a)-case we find \( q_1, q_2, q_1' \in \text{cf} \) such that

\[
(4.14) \quad \vdash_\pi q = q_1 \parallel q_2, \vdash_\pi q_1 = a.q_1'
\]

and \( \theta(q_1) = a.t', \theta(q_2) = (\parallel \gamma(t) \setminus \{a.t'\}), \theta(q_1') = t' \).

Clearly \( |A_{s'}| < |A_s| \) so we can use the inductive assumption to find \( p' \in \text{cf} \) such that

\[
(4.15) \quad \vdash_\delta p' \leq q_1'
\]

and \( \theta(p') = s' \). Let \( p := a.p' \parallel q_2 \). We have:

\[
\vdash_\delta p \equiv a.p' \parallel q_2
\]

\[
= \leq a.q_1' \parallel q_2 \quad \text{by (4.15) and congruence}
\]

\[
= \pi q_1 \parallel q_2 \quad \text{by the second part of (4.14)}
\]

\[
= \pi q \quad \text{by the first part of (4.14)}
\]

Finally: \( \theta(p) = a.\theta(p') \parallel \theta(q_2) = a.s' \parallel (\parallel \gamma(t) \setminus \{a.t'\}) = u \parallel (\parallel \gamma(s) \setminus \{u\}) = s \).

This also completes the inductive step. \( \square \)

**Proposition 4.3.31** Let \( q \in \text{snf} \). Then \( \exists p \in \text{snf} \). \( \vdash_\delta p = q, \Theta(p) = \delta \Theta(q) \)

**Proof** At first we prove an intermediate result. Notice that since \( \prec^+ = \prec \) we see \( s \prec t \) implies \( s \prec^n t \) for some \( n \geq 1 \). Then we can use induction on \( n \) to prove:

\[
(4.16) \quad q \in \text{cf}, s \prec \theta(q) \Rightarrow \exists p \in \text{cf}. \vdash_\delta p \leq q, \theta(p) = s
\]

\( n = 1 \): I.e., \( s \prec \theta(q) \). The result follows from proposition 4.3.30 above.

\( n > 1 \): Then there exists some \( u \) such that \( s \prec^{n-1} u, u \prec \theta(q) \). By proposition 4.3.30 there exists some \( r \in \text{cf} \). \( \vdash_\delta r \leq q, \theta(r) = u \). Hence \( s \prec^{n-1} \theta(r) \), so by hypothesis \( \exists p \in \text{cf} \). \( \vdash_\delta p \leq r, \theta(p) = s \). Then by transitivity \( \vdash_\delta p \leq q \).

We now prove the proposition by induction on the structure of \( q \) considered as a sum-normal form.
\( q \in \text{cf} \): Then we by (4.16) have for every \( s < \theta(q) \) a \( p_s \in \text{cf} \) such that \( \vdash \delta p_s \leq q, \theta(p_s) = s \).

Since in general \( S \)-closure is finite and thereby \( \{ s \mid s < \theta(q) \} \) too, we have by congruence \( \vdash \delta \sum_{s \prec \theta(q)} p_s \leq \sum_{s \prec \theta(q)} q \) and \( \vdash \delta q + \sum_{s \prec \theta(q)} p_s \leq q + \sum_{s \prec \theta(q)} q \). +4 gives us \( \vdash \delta q + \sum_{s \prec \theta(q)} q = q \), so \( \vdash \delta q + \sum_{s \prec \theta(q)} p_s \leq q \) and from +5 we then deduce \( \vdash \delta q + \sum_{s \prec \theta(q)} p_s = q \). Now let \( p := q + \sum_{s \prec \theta(q)} p_s \). Then \( \vdash \delta p = q \) and \( \Theta(p) = \{ \theta(q) \} \cup \{ \theta(p_s) \mid s < \theta(q) \} = \{ \theta(q) \} \cup \{ s \mid s < \theta(q) \} = \{ s \mid s < \theta(q) \} = \delta \Theta(q) = \delta \{ \theta(q) \} = \delta \{ \theta(q') \mid q' \in \{ q \} \} = (\text{since } q \in \text{cf}) \delta(\delta(\theta(q') \mid q' \in \{ q \})) = \delta \Theta(q).

\( q \equiv q_1 + q_2, q_1, q_2 \in \text{snf} \): By hypothesis of induction we know that there exists \( p_1 \in \text{snf} \). \( \vdash \delta p_1 = q_1, \Theta(p_1) = \delta \Theta(q_1) \) for \( i \in 2 \). Define \( p := p_1 + p_2 \in \text{snf} \). Then \( \vdash \delta p \equiv p_1 + p_2 = q_1 + q_2 \equiv q \) by congruence. Furthermore \( \Theta(p) = \Theta(p_1 + p_2) = \{ \theta(p') \mid p' \in S(p_1) \cup S(p_2) \} = \Theta(p_1) \cup \Theta(p_2) = \delta \Theta(q_1) \cup \delta \Theta(q_2) = \delta(\Theta(q_1) \cup \Theta(q_2)) = \delta \Theta(q_1 + q_2) = \delta \Theta(q).

Finally we are ready to prove the completeness in full detail.

**Proof** (of proposition 4.3.2) From the previous lemmas and propositions we at first show:

\[(4.17) \quad p \in PL \Rightarrow \exists q \in \text{msnf}. \vdash q = p, [p]_\delta = \Theta(q)\]

Given \( p \in PL \). From proposition 4.3.9 we find a \( p_1 \in \text{snf} \). \( \vdash \pi p = p_1 \). Then from proposition 4.3.16 also a \( p_2 \in \text{pf} \) is obtained such that \( \vdash \pi p_1 = p_2 \). Since \( \vdash \pi p = p_2 \) and the proof system is sound we have \( [p]_\delta = [p_2]_\delta \). Now by proposition 4.3.31 we find a \( p_3 \in \text{snf} \) such that \( \vdash \delta p_2 = p_3 \) and \( \delta \Theta(p_2) = \Theta(p_3) \). Furthermore proposition 4.3.21 gives us a \( q \in \text{msnf} \). \( \vdash \pi p_3 =_{si} q \). By corollary 4.3.19 then \( \Theta(p_3) = \Theta(q) \). Collecting the facts we have \( \vdash \delta p = q \) and

\[\begin{align*}
[p]_\delta &= [p_2]_\delta \\
&= \delta[p_2]_\pi \quad \text{by lemma 3.3.16} \\
&= \delta \Theta(p_2) \quad \text{by corollary 4.3.23 and the fact that } p_2 \in \text{pf} \\
&= \Theta(p_3) = \Theta(q)
\end{align*}\]

thereby establishing (4.17).

Now for the completeness. Assume \( [p]_\delta \not\subseteq [q]_\delta \).

By (4.17) we can find \( p', q' \in \text{msnf} \). \( \vdash \delta p = p', \vdash \delta q' = q \) and \( [p]_\delta = \Theta(p'), [q]_\delta = \Theta(q') \). Then by definition of \( \not\subseteq \) we have \( [p]_\delta \not\subseteq [q]_\delta \implies \Theta(p') \not\subseteq \Theta(q') \) which by the definition of \( \Theta \) is the same as \( \{ \theta(p'') \mid p'' \in S(p') \} \subseteq \{ \theta(q'') \mid q'' \in S(q') \} \) or equivalently

\[(4.18) \quad \forall p'' \in S(p') \exists q'' \in S(q'). \theta(p'') = \theta(q'')\]

Since \( \theta(p'') = \theta(q'') \), \( p'', q'' \in \text{cf} \) by proposition 4.3.29 implies \( \vdash \pi p'' =_{nc} q'' \) we see that (4.18) can be written as

\[S(p') \subseteq_{nc} S(q')\]

Because \( p', q' \in \text{msnf} \subseteq \text{snf} \) we can use proposition 4.3.25 to deduce \( \vdash \pi p' + q' = q' \). By +5 we have \( \vdash \pi p' \leq p' + q' \) so \( \vdash \pi p' \leq q' \). Now \( p' \) and \( q' \) where found such that \( \vdash \delta p = p' \) and \( \vdash \delta q' = q \) wherefore \( \vdash \delta p \leq q \) by transitivity and we are done. \( \square \)
Chapter 5

Adding Recursion to $PL$

In this chapter we shall extend our process language, $PL$, to include recursively defined processes and investigate their semantics as in chapter 3.

All concepts introduced in the previous chapters should be adjusted to the new set-up and also new concepts are needed. We will make good use of the results obtained in these chapters so as results and notions of Hennessy [Hen85b] which the reader will be assumed to be acquainted with. Also some notions of [Hen87a]) are used.

5.1 Denotational Semantics

In order to define interpretations of the recursively defined processes the $\Sigma$-po algebras are extended to $\Sigma$-domains. The signature remains the same, but the function symbol $NIL$ of rank 0 is nominated the role of $\Omega$ in [Hen87a], i.e., the syntactical representative of the least element of the interpretations. The carriers are in principle the old ones but extended to include infinite sets as well. Formally:

**Definition 5.1.1** For $\ast$ in $\{\delta, v, \chi\}$ the carrier $C_\ast$ is defined by:

$$C_\ast = \{S \neq \emptyset \mid \exists T \subseteq TSW. S = \ast(\pi T)\}$$

Notice that we have left out the restriction of finiteness. The elements of the $\ast$-carriers are of course $\ast$-closed so corollary 3.1.4 remains true in this set-up. Also the definition and results definition 3.1.5—corollary 3.1.11 carry over since the proofs there makes no use of the finiteness of the sets. The only deviation is of course in definition 3.1.8 where elements of $C_\pi$ now may be infinite sets as well.

The purpose of the next propositions is to show that the carriers are algebraic complete partial orders (algebraic cpos for short).

**Proposition 5.1.2** For each $\ast$ in $\{\delta, v, \chi\}$ the pair $(C_\ast, \preceq_\ast)$ is an algebraic cpo with least element $\bot_{C_\ast} = \{\varepsilon\}$ and every nonempty subset $D$ of $C_\ast$ has a lub $\bigvee_\ast D = \ast(\bigcup_{d \in D} d)$
or for short \( \bigcup D = \star \bigcup_{d \in D} d \). In the case of \( \star \) in \( \{ \delta, v \} \) \( \bigcup_* D \) actually equals \( \bigcup_{d \in D} d \). Furthermore the compact elements are the finite sets of \( C_* \).

**Proof** At first we show \((C_*, \subseteq)\) to be a cpo. Recall that \( \mathfrak{A} \) simply is \( \subseteq \). Already in chapter 3 it was noticed (though for finite sets) that \( \{ \varepsilon \} \) is the least element \( \bot_{C_*} \) of \( C_* \) and that \((C_*, \subseteq)\) is a partial order. Now let \( D \) be a nonempty subset of \( C_* \).

\[
\bigcup D = \bigcup_{d \in D} d \in C_* \text{ is seen as follows: } d \in C_* \text{ implies } d = \star(\pi T_d) \text{ for some } \emptyset \neq T_d \subseteq TSW. \quad \text{Clearly } \bigcup_{d \in D} T_d \subseteq TSW \text{ and is nonempty since } D \text{ is. Then } \star(\pi(\bigcup_{\delta \in D} T_d)) = \star \bigcup_{d \in D} \pi T_d = (\text{in case of } \chi; \text{corollary } 1.3.23) \star \bigcup_{d \in D}(\pi T_d) = \star \bigcup_{d \in D} d. \]

Notice that we actually have

\[
(5.1) \quad \bigcup_* D = \bigcup_{d \in D} d \text{ if } \star \in \{ \delta, v \} \text{ since } \delta \text{ and } v \text{ distributes over } \cup.
\]

Also \( \bigcup D \) is a lub of \( D \): Obviously \( \bigcup D \) is a ub for \( D \). Suppose there exists a ub \( e \in C_* \) of \( D \), i.e., \( \forall d \in D. d \subseteq e \). We shall show \( \bigcup D \subseteq e \) and do this by proving \( t \in \bigcup D \) implies \( t \in e \). By (5.1) this is clear in the case of \( \star \in \{ \delta, v \} \). So we are left with the case \( \star = \chi \).

Now \( t \in \bigcup D = \chi \bigcup_{d \in D} d \) implies \( \exists d, d' \in D. t \in \chi\{d, d'\} \). By assumption \( d, d' \subseteq e \) and since \( e \in C_\chi \) it must be \( \chi \)-closed, so \( \chi\{d, d'\} \subseteq e \) and thereby also \( t \in e \).

Since every directed set is nonempty we then have that \((C_*, \subseteq)\) is a cpo.

Let \( a \in C_* \) be a finite set. We shall show that \( a \) is compact. That is for a directed set \( D \) such that \( a \subseteq \bigcup D \) there exist an \( e \in D \) such that \( a \subseteq e \). At first we prove

\[
\forall t \in a \exists d_t \in D. t \in d_t
\]

Assume \( t \in a \). Since \( a \subseteq \bigcup D \) we have \( t \in \bigcup D \) wherefore \( \exists d, d' \in D. t \in \star\{d, d'\} \). \( D \) directed implies that there exists a \( d_t \in D \) such that \( d, d' \subseteq d_t \). Because \( d_t \) is \( \star \)-closed we must have \( \star\{d, d'\} \subseteq d_t \) and hence also \( t \in d_t \).

Now since \( a \) is finite \( D_a = \{d_t \mid t \in a\} \) must be finite too. The directedness of \( D \) implies there is a ub \( e \) of \( D_a \) in \( D \) and clearly \( a \subseteq e \). So \( a \) is indeed compact.

Conversely we show that all compact elements are finite sets.

Suppose \( a \in C_* \) is an infinite set. Then there exists a infinite subset \( T \) of \( TSW \) such that \( a = \star \pi T \). Since \( T \) is infinite it contains a countable infinite subset \( T' = \{t_n\}_{n \in \mathbb{N}} \) of different tree semiwords. For all \( n \in \mathbb{N} \) define

\[
\begin{align*}
T_n &= \bigcup_{i \leq n}\{t_i\} \\
S_n &= \{s \in T \mid \forall j > n. A_{t_j} \not\supseteq A_s\} \\
d_n &= \star\pi(T_n \cup S_n)
\end{align*}
\]

At first we show that \( D = \{d_n\}_{n \in \mathbb{N}} \) forms an increasing chain.

Since \( T_n \subseteq T_{n+1} \) and \( S_n \subseteq S_{n+1} \) it follows that \( d_n \subseteq d_{n+1} \), so it remains to show that the chain is increasing i.e.,

\[
(5.2) \quad \forall n \exists m > n. d_n \subseteq d_m
\]

Let \( n \) be given. In general for an arbitrary finite set \( V \) of (finite) semiwords there is only finite many semiwords \( s \) such that there is a \( t \in V \) with \( A_s \subseteq A_t \). So because \( T_n \) is finite and \( T' \) is infinite there then exists a \( t_m \in T' \) such that \( \forall t \in T_n. A_{t_m} \not\supseteq A_t \). It follows that

110
$m > n$ and $t_m \in d_m$. Also $t_m \notin d_n$.

Assume on the contrary $t_m \in d_n$. Then there exists $s, t \in T_n \cup S_n$ such that $t_m \in \star \pi \{s, t\}$. So either $A_m \subseteq A_t$ or $A_m \subseteq A_i$. Consider $A_m \subseteq A_s$. $s$ must come from $T_n$ or $S_n$. If $s \in T_n$ this means $A_m \subseteq A_t$ for some $t_i \in T_n$. But this is impossible by the way $t_m$ is chosen. If $s \in S_n$ we have $\forall j > n, A_{t_i} \nsubseteq A_s$, especially $A_{t_m} \nsubseteq A_s$ as $m > n$. Again a contradiction. The case $A_m \subseteq A_i$ is ruled out in the same way. Hence the assumption was false.

Now where were we know that $D$ is an increasing chain we convince ourselves that $a \subseteq \bigsqcup_s D$

To begin with we show:

$$\forall t \in T \exists n. t \in T_n \cup S_n$$

If $t \in T$ then $t = t_n$ for some $n$ and $t \in T_n$. So suppose $t \in T \setminus T'$. There is only finite many $t_i \in T'$ with $A_{t_i} \subseteq A_i$. So choose $n$ to be the $i$ of the last $t_i$ with $A_{t_i} \subseteq A_i$. Then $\forall j > n, A_{t_i} \nsubseteq A_i$ and so $t \in S_n$.

Next to show $a \subseteq \bigsqcup_s D$ let an $s \in a$ be given. Since $\forall n, d_n \subseteq \bigsqcup_s D$ it is enough to find a $d_m$ such that $s \in d_m$ in order to have $s \in \bigsqcup_s D$. $s \in a$ implies $\exists t_i, t \in T. s \in \pi \{t, t_i\}$. Using (5.3) we obtain $n$ and $n'$ for $t$ and $t_i$ respectively. W.l.o.g. assume $n \leq n'$. Then $T_n \cup S_n \subseteq T_{n'} \cup S_{n'}$ and $t, t' \in T_{n'} \cup S_{n'}$ wherefore $\pi \{t, t'\} \subseteq d_{n'}$. So $s \in \pi \{t, t'\} \subseteq d_{n'}$ and we can chose $m = n'$ to get the desired $d_m$.

We can now return to the question of the compactness of $a$. Since $a$ is assumed to be compact and $a \subseteq \bigsqcup_s D$ there should exists a $d_n \in D$ such that $a \subseteq d_n$. By (5.2) there exists a $m$ such that $d_m \subseteq d_m$ or by the proof of (5.2) $\exists t_m, t_m \notin d_n$. But this is a contradiction to $t_m \in T' \subseteq a$, so our assumption of $a$ being an infinite set was wrong.

Knowing how the compact elements of $C_\star$ looks like it easier to show ($C_\star, \subseteq$) algebraic.

We shall show $\forall a \in C_\star. a = \bigsqcup_s D_a$, where $D_a = \{d \mid d \subseteq a, d \text{ compact}\}$.

Since $\{\varepsilon\} \subseteq a$ for all $a$ it follows that $D_a$ is nonempty, so $\bigsqcup_s D_a$ is defined. It is clear that $\bigsqcup_s D_a \subseteq a$ as $a$ is a upper bound for $D_a$. To see the other inclusion let $a \in \bigsqcup_s D_a$. We prove $t \in \bigsqcup_s D_a$. $t \in a = \pi \{s, s'\}$ implies there exists $s, s' \in T$ such that $t \in \pi \{s, s'\}$. $\pi \{s, s'\} \in D_a$ follows from $\pi \{s, s'\} \subseteq a$ and the finiteness of $\pi \{s, s'\}$. In general $d \in D$ implies $d \subseteq \bigsqcup_s D$ so we have $t \in \pi \{s, s'\} \subseteq \bigsqcup_s D_a$.

In order to see that we actually have obtained $\Sigma$-domains corollary 3.1.11 must be strengthened to:

**Proposition 5.1.3** All $op_\star \in \Sigma_\star$ are (relative) continuous on $C_\star$ w.r.t. $\bigsqcup_\star$ for each $\star$ in \{$\delta, \nu, \chi$\}.

**Proof** Since the operators of $\Sigma_\star$ are natural extensions to sets they evidently are continuous w.r.t. $\bigsqcup_\star$ ($\subseteq$). The proofs of (3.11) and (3.12) on page 83 can be carried over to infinite sets wherefore we get:

$$\pi \{s, s'\} = \pi \{s, s'\}$$

where $op_\pi \in \Sigma_\pi$ and $\pi \{s, s'\} \subseteq C_\pi$. It is then an easy matter to show the operators to be continuous. E.g., suppose $D$ is a nonempty subset of $C_\star$. Then

111
That we in general for a nonempty set, $D$, of sets of tree-semiwords have:

$$\chi \chi \bigcup D = \chi \bigcup \chi D \quad (\text{by } S \in D)$$

follows from c) of corollary 1.3.23. Similarly it is shown that $+ \ast$ and $\| \ast$ are left and right continuous ($\| \ast$ under the usual proviso).

\begin{center}
\textbf{Corollary 5.1.4} For each $\ast$ in \{$\delta, v, \chi$\} $A_\ast = (C_\ast, \Delta_\ast, \Sigma_\ast)$ is a $\Sigma$-domain.
\end{center}

We now proceed by defining the language, $RPL(X)$, of the recursive process terms. $RPL(X)$ can be considered as the extension of $PL(X)$ obtained by adding constructors for recursion.

The recursive terms over $\Sigma$, $RBL(X)$, is the terms obtained from the following schema:

$$t ::= NIL \mid x, x \in X \mid a. t, a \in Act \mid t + t \mid t \| t \mid recx. t, x \in X$$

$RPL(X)$ is then defined to be those terms of $RBL(X)$ where every subterm $t$ meets the usual requirement:

$$t = t_1 \| t_2 \Rightarrow L(t_1) \cap L(t_2) = \emptyset \quad (5.5)$$

and the additional requirement:

$$t = recx. t' \Rightarrow L(x) = L(t') \quad (5.6)$$

where $L$ is extended to $RBL(X)$ by defining $L(recx. t) = L(x) \cup L(t)$.

$FV(t)$ is defined in the normal way to be the free variables in $t$. The recursive processes is the subset of $RPL(X)$ with no free variables and is denoted $RPL$.

\begin{center}
\textbf{Notice}
\end{center}

\begin{itemize}
    \item[i)] $PL(X)$, $PL \subseteq RPL(X)$ and $PL = PL(X) \cap RPL$.
    \item[ii)] Since $recx. p$ intuitively stands for the “solution” of the equation $x = p$ the requirement in c) of equal sorts of $x$ and $p$ is natural.
\end{itemize}

\begin{center}
\textbf{\textit{We shall refer to } } PL(X) \textbf{\textit{as the syntactic finite process terms.}}
\end{center}

The definitive consequences of the restriction on the arguments to $\|$ become clear by the introduction of $recx$. as can be seen by the following example.
Example: Consider the term \( p = a.NIL \| b.x \in PL(X) \). No matter what sort \( x \) might have \( \text{rec}\, x.\, p \) cannot be a (legal) term of \( RPL(X) \) because the sort of \( x \) must contain \( a \) and \( b \) in which case we would not have \( p \in PL(X) \). As a consequence \( \text{rec}\, x.\) can only prefix terms containing \( \| \) if at most one of the arguments has nonempty sort. This means that only terms like \( \text{rec}\, x.\, (\text{NIL} \| p) \| \text{NIL} \) are possible. On the other hand \( RPL(X) \) can contain terms like \( (\text{rec}\, x.\, a.x + b.x) \| (\text{rec}\, y.\, c.y + d.y) \).

The denotational maps \([ \_ ]_\delta \), \([ \_ ]_v \) and \([ \_ ]_x \) are given in the standard way by means of environments as described by Hennessy. Of course the environments shall be modified as the \( A_\tau \)-assignments in chapter 4. Notice that \([ \_ ]_x \) is independent of the environment when used on closed terms.

### 5.2 Operational Semantics

With the definitions and results of section 3.2 extended in the natural way to the new setting we can take over most of them. We will in the following briefly state the main differences.

\( \overline{RBL}(X) \) is defined to be the least set \( C \) such that:

\[
\begin{align*}
RBL(X) \subseteq C \\
\bar{a}.t \in C & \quad \text{if } t \in C \text{ and } a \in \text{Act} \\
t_1 \| t_2 \in C & \quad \text{if } t_1, t_2 \in C
\end{align*}
\]

and \( RCL(X) \)—the recursive configuration terms—are defined to be the terms of \( \overline{RBL}(X) \) that satisfies (5.5) and (5.6) above. The recursive process configurations \( RCL \) are simply the closed terms of \( RCL(X) \).

Of course definition 3.2.2 has to have an inference rule for \( \text{rec}\, x.\, - \):

\[
6) \quad \frac{p[\text{rec}\, x.\, p/x] \xrightarrow{a} q, a \in \text{Act}}{\text{rec}\, x.\, p \xrightarrow{a} q}
\]

and the test configurations, \( TC \), shall be changed to \( RTC \) in order to include recursive process configurations (in \( (t, p) \)). The test language, \( TL \), remain unchanged. The rest of the corresponding section of chapter 3 extends smoothly.

### 5.3 Full Abstractness

The map \( \overline{\theta} : CL \rightarrow TSW \) associating tree-semiwords with configurations is extended to \( RCL \rightarrow TSW \) simply by letting \( \overline{\theta}(p) = \varepsilon \) if \( p \in RPL \) and otherwise keeping it’s compositional definition (page 75).

In this section we shall also use the notions of algebraic relations and syntactic preorders as explained in [Hen87a].
For \( t, t' \in RPL(X) \) we write \( t \preceq t' \) to mean that \( t \) is a syntactic approximation to \( t' \) where \( \preceq \) is defined to be the least (relative) \( \Sigma \)-precongruence over \( RPL(X) \) which satisfies:

\[
\text{NIL} \preceq t \\
t[\text{rec}. t/x] \preceq \text{rec}. t
\]

For every \( t \in RPL(X) \), \( \text{Fin}(t) \) denotes \( \{ t' \in PL(X) \mid t' \preceq t \} \); i.e., \( \text{Fin}(t) \) is the syntactical finite approximations to \( t \). \( \preceq \) is extended to \( RCL(X) \) by taking it to be the least (relative) \( \Sigma \)-precongruence over \( RCL(X) \) which satisfies (5.7) above.

A relation \( R \) over \( RPL \) is algebraic if for all \( t, u \in RPL \):

\[
t R u \iff \forall t' \in \text{Fin}(t) \exists u' \in \text{Fin}(u). t' R u'
\]

In the following it will prove useful to be able to limit the number of experiments necessary to distinguish processes. To this end we first investigate the possibilities to reduce the size of a test \( t \) in an experiment \((A, t)\) on a process \( q \) without affecting the outcome of the experiment.

Looking at definition 3.2.9 we get some ideas. As an example consider inference rule 6) and the test \( a:t \). If \( p \not\rightarrow \frac{\overline{a}}{a} \) the test \( a:/> \) would have the same outcome. Whether \( p \not\rightarrow \frac{\overline{a}}{a} \) or \( p \not\rightarrow \frac{\overline{a}}{a} \) depends naturally on \( p \), but if we can find some criterions under which we can deduce \( p \not\rightarrow \frac{\overline{a}}{a} \) for all \( a \in Act \) we can certainly reduce the test.

Now if we have signaled the multiset of actions \( A \) getting to \( p \) (i.e., \( p \in D(A, q) \)) it should be clear that \( p \not\rightarrow \frac{\overline{a}}{a} \) implies \( |\overline{a}| \leq |A| \). Hence we can make the reduction whenever we are sure that at least \( |A| \) signaled actions have been tested. i.e., if \( \overline{a}.t \) is a subterm of the test \( t' \) in the experiment \((A, t')\) and the “path” leading to \( \overline{a}.t \) is \(|A| \) long we can replace \( \overline{a}.t \) by \( \overline{a}.\top \). So this limits the necessary depth of a test \( t' \) in an experiment \((A, t')\).

Another idea to reduce the set of experiments has it’s roots in the same inference rule. Consider the same example as above. Clearly it makes no difference to the outcome of the test if we replace \( a \) with \( b \) in \( \overline{a}.t \) as long as we are sure \( p \not\rightarrow \frac{\overline{b}}{b} \).

These considerations leads to the following definition and proposition.

**Definition 5.3.1** Given \( c \in Act \) and \( B \subseteq Act \) we successively for each \( n \in \mathbb{N} \) define \( f^n_{c,B} : TL \rightarrow TL \) structurally as follows:

\[
\begin{align*}
n = 0: & \quad \top \mapsto \top \\
 & \quad \overline{a}.t \mapsto \overline{c}.\top \\
 & \quad t \triangle t' \mapsto f^0_{c,B}(t) \triangle f^0_{c,B}(t') \quad \text{for } \triangle \in \{\&., \triangledown\} \\

n > 0: & \quad \top \mapsto \top \\
 & \quad \overline{a}.t \mapsto \overline{b} \cdot f^{n-1}_{c,B}(t) \quad \text{where } b = \begin{cases} a & \text{if } a \in B \\ c & \text{otherwise} \end{cases} \\
 & \quad t \triangle t' \mapsto f^n_{c,B}(t) \triangle f^n_{c,B}(t') \quad \text{for } \triangle \in \{\&., \triangledown\}
\end{align*}
\]

\[\square\]
It should be clear that $f_{c,B}^n$ is well-defined.

Notice

i) A subsequent test to $\bar{a}$. is discarted and $\top$ inserted when $f_{c,B}^0$ is applied.

ii) If $\bar{a}$ occurs in $f_{c,B}^n(t)$ then $a \in B \cup \{c\}$.

\[ \square \]

**Proposition 5.3.2** Let $p \in RPL$ and $(A, t) \in E$. If $c \not\in L(p)$ then:

$$p \text{ may } x (A, t) \iff p \text{ may } x (A, f_{c,L(A)}^{|A|}(t))$$

where $x$ either is accept or reject.

Before we prove this proposition we need the following definition and lemma. We will define a function, $\overline{ad}$, which given a $p \in RCL$ gives an upper bound of the length of $w \in \text{Act}$ where $p \xrightarrow{w}$. Notice there must not be any initiation of actions in the sequence ($\overline{w} \in \overline{\text{Act}}$). Latter we need the action depth, $ad$, of a closed syntactical finite term ($\in PL$) too so we introduce this notion here too. It will also be convenient with a function, $\overline{L}$, which yields the label set corresponding to the actions signaled to initiate.

**Definition 5.3.3** The action depth, $ad$, of a process and the barred depth, $\overline{ad}$, of a recursive process configuration is defined as follows:

$$
\begin{align*}
\text{ad} : PL & \longrightarrow N \\
\text{NIL} & \mapsto 0 \\
a.p & \mapsto 1 + \text{ad}(p) \\
p + q & \mapsto \max\{\text{ad}(p), \text{ad}(q)\} \\
p \parallel q & \mapsto \text{ad}(p) + \text{ad}(q)
\end{align*}
$$

$$
\begin{align*}
\overline{\text{ad}} : RCL & \longrightarrow N \\
p & \mapsto 0 \\
\bar{a}.p & \mapsto 1 + \overline{\text{ad}}(p) \\
p \parallel q & \mapsto \overline{\text{ad}}(p) + \overline{\text{ad}}(q)
\end{align*}
$$

and the map $\overline{L} : RCL \longrightarrow \text{Act}$ is given by:

$$
\begin{align*}
p & \mapsto \emptyset \quad \text{ if } p \in RPL \\
\bar{a}.p & \mapsto \{a\} \cup \overline{L}(p) \\
p \parallel q & \mapsto \overline{L}(p) \cup \overline{L}(q)
\end{align*}
$$

\[ \square \]

So $\overline{ad}$ actually estimates the necessary maximal “length” of a test.

The following lemma tells that nothing is lost in reducing the test as informally argued previously.
Lemma 5.3.4 For \( p \in RCL \), \( B \subseteq \text{Act} \) and \( c \in \text{Act} \) such that \( \overline{\text{ad}}(p) \leq n \), \( \bar{L}(p) \subseteq B \), \( c \not\in B \) we have:

\[(t, p) \rightarrow^* \top \text{ iff } (f^n_{c,B}(t), p) \rightarrow^* \top\]

Proof

only if: Assume \( (t, p) \rightarrow^* \top \). The proof will be by induction on \( n \).

\( n = 0 \): Then clearly \( p \not\rightarrow \) for all \( b \in \text{Act} \). We proceed by induction on the structure of \( t \).

\( t = \top \): Follows directly from \( f^0_{c,B}(\top) = \top \).

\( t = \bar{a}.t' \): Since \( p \not\rightarrow \) we can exclude this case.

\( t = t' \land t'' \): By lemma 3.2.11 \((t' \land t'\prime) \rightarrow^* \top \) implies \((t', p) \rightarrow^* \top \) and \((t'', p) \rightarrow^* \top \).

Since \( f^n_{c,B}(t' \land t'\prime) = f^0_{c,B}(t') \land f^0_{c,B}(t'\prime) \) the result now follows using the hypothesis of induction and lemma 3.2.11.

\( t = t' \land t'' \land t'' \): Similar.

\( n > 0 \): Again we use structural induction.

\( t = \top \): Immediate.

\( t = \bar{a}.t' \): \((\bar{a}.t', p) \rightarrow^* \top \) implies \( p \not\rightarrow \bar{a}' \), \((t', p) \rightarrow^* \top \). Clearly \( \overline{\text{ad}}(p') \leq n - 1 \), \( a \in \bar{L}(p') \subseteq B \) and by corollary 3.2.3 \( L(p') \subseteq L(p) \). Hence \( \bar{L}(p') \subseteq B \) and we can use the hypothesis of (structural or natural) induction to get \((f^n_{c,B}(t'), p') \rightarrow^* \top \).

Since \( a \in B \) we have \( f^n_{c,B}(\bar{a}.p') = \bar{a}.f^n_{c,B}(t') \) and from \( p \not\rightarrow \bar{a}' \) it then follows that \((f^n_{c,B}(\bar{a}.t'), p') \rightarrow^* \top \).

\( t = t' \land t'' \land t'' \): Similar as in the case \( n = 0 \) using the hypothesis of structural induction.

if: Suppose \((f^n_{c,B}(t), p) \rightarrow^* \top \) for some \( t \). Again we use natural induction on \( n \).

\( n = 0 \): As for the other implication we use structural induction.

\( t = \top \): Trivial.

\( t = \bar{a}.t' \): Then \( f^0_{c,B}(t) = \bar{c}.\top \). Since \( p \not\rightarrow \) for all \( d \in \text{Act} \) when \( \overline{\text{ad}}(p) \leq 0 \) and because \((\bar{c}.\top, p) \rightarrow^* \top \) implies \( p \rightarrow \bar{c}' \) for some \( \bar{c}' \) we can exclude this case.

\( t = t' \land t'' \): We have \( f^0_{c,B}(t' \land t'\prime) = f^0_{c,B}(t') \land f^0_{c,B}(t'\prime) \). From the assumption then \((f^0_{c,B}(t') \land f^0_{c,B}(t''\prime), p) \rightarrow^* \top \) and from lemma 3.2.11 \((f^0_{c,B}(t'), p) \rightarrow^* \top \) and \((f^0_{c,B}(t''\prime), p) \rightarrow^* \top \).

Using the hypothesis and the same lemma we get \((t' \land t''\prime, p) \rightarrow^* \top \).

\( t = t' \land t'' \land t'' \): Similar.

\( n > 0 \): Structural induction on \( t \).

\( t = \top \): Immediate.

\( t = \bar{a}.t' \): Then \( f^n_{c,B}(t) \) equals \( b.f^n_{c,B}(t') \), where \( b = a \) if \( a \in B \) and \( b = c \) otherwise.

\((\bar{a}.f^n_{c,B}(t'), p) \rightarrow^* \top \) implies \((f^n_{c,B}(t'), p') \rightarrow^* \top \) where \( p \not\rightarrow b' \). But \( p \not\rightarrow b' \) clearly implies \( \bar{L}(p') \subseteq \bar{L}(p) \), \( b \in \bar{L}(p) \subseteq B \) and \( \overline{\text{ad}}(p') \leq \overline{\text{ad}}(p) \) so by induction \((t', p') \rightarrow^* \top \). Since \( b \in B \) and \( c \not\in B \) we have \( b = a \), i.e., \( p \not\rightarrow b' \) and thereby \((\bar{a}.t', p) \rightarrow^* \top \).
\( t = t' \& t'' , t' \triangleleft t'' \): Similar to the case \( n = 0 \).

With this lemma we can give the promised proof.

**Proof** of proposition 5.3.2.
Assume \( p \in RPL, (A,t) \in E \) and \( c \not\in L(p) \). We show:

\[
p \text{ may } x (A,t) \iff p \text{ may } x (A,F_{c,L(A)}^{[A]}(t))
\]

only if: Suppose \( p \) may \( x (A,t) \). Then \( \exists q \in D(A,p),(t,q) \rightarrow^* \top \). Clearly \( q \in D(A,p) \) implies \( d(q) = |A| \) and \( L(q) = L(A) \). Because \( L(q) \subseteq L(p) \) and \( c \not\in L(p) \) we can then use lemma 5.3.4 to get \((f_{c,L(A)}^{[A]}(t),q) \rightarrow^* \top \) and we are done for this implication.

if: Similar.

The next step in reducing the number of experiments is to observe that we just as well can use test normal forms in the experiments. Finally notice that all duplicates in a test normal form can be removed without affecting the outcome of the test. This leads to the following definition.

**Definition 5.3.5** \( t \in TL \) is a reduced test normal form iff

a) \( t \) is a test normal form. I.e., \( t = \&_{j \in n}(\nabla_{k \in n_{t_j}} w_{j,k}) \).

b) \( \forall j \in n \forall k,l \in n_{t_j} k \neq l \Rightarrow w_{j,k} \neq w_{j,l} \).

c) \( \forall i,j \in n, i \neq j \Rightarrow \{w_{i1},\ldots,w_{im_i}\} \neq \{w_{j1},\ldots,w_{jn_j}\} \).

From lemma 3.2.11 we get:

**Corollary 5.3.6**

a) \( t \& t \cong t \)

b) \( t \nabla t \cong t \)

With this, proposition 3.2.14 and proposition 3.2.16 we easily get:

**Proposition 5.3.7** For every \( t \in TL \) there is a reduced test normal form \( t' \in TL \) such that \( t \cong t' \).

**Proof** At first we use proposition 3.2.16 to find a test normal form \( t'' \) such that \( t \cong t'' \). Then if there exists \( j \in n \) and \( k,l \in n_{t_j} \) such that \( k \neq l \) and \( w_{j,k} = w_{j,l} \) we use proposition 3.2.14 and proposition 5.3.6.b) to remove e.g., \( \overline{w_{j,k}} \top \). Iterating this we eventually get a test normal form which fulfills b) of definition 5.3.5. Finally use proposition 3.2.14.a)-d) and corollary 5.3.6.a) to obtain a reduced test normal form \( t' \) with \( t'' \cong t' \). By transitivity then \( t \cong t' \). 

\( \square \)
Using the definition of may x with x equal to either accept or to reject it is easy to see that:

\[(5.8) \quad t \cong t' \implies \forall p \in RPL \forall A. p \text{ may } x (A, t) \iff p \text{ may } x (A, t')\]

**Definition 5.3.8** Let \( F_c \) denote the set \{\((A, t) \in E \mid t \text{ is a reduced test normal form and } \exists t' \in TL, t = f_{c, L(A)}^{|A|}(t')\}\).

For \( p, q \in PL \) we then write:

\[ p \Preceq^F_c q \iff \forall e \in F_c. p \text{ may } x e \iff q \text{ may } x e \]

So if we consider \( p, q \) and \( c \not\in L(p) \cup L(q) \) then \( F_c \) denotes according to the previous ideas a reduced set of experiments sufficient to characterize a testing preorder between \( p \) and \( q \). Formally we have:

**Proposition 5.3.9** Given \( p, q \in RPL \) and \( c \in Act \) such that \( c \not\in L(p) \cup L(q) \). Then

\[ p \Preceq^F_c q \iff p \Preceq^E_X q \]

**Proof**

*only if*: Immediate since \( F_c \subseteq E \).

*if*: Assume \( p \Preceq^F_c q \) and \( p \text{ may } x (A, t) \) for some \((A, t) \in E\). Since \( c \not\in L(p) \) we have according to proposition 5.3.2 \( p \text{ may } x (A, f_{c, L(A)}^{|A|}(t)) \). By proposition 5.3.7 there exists a reduced test normal form \( t' \) such that \( t' \cong f_{c, L(A)}^{|A|}(t) \), so by (5.8) \( p \text{ may } x (A, t') \). Inspecting the proof of proposition 5.3.7 and definition 5.3.5 we see that in the process of converging \( f_{c, L(A)}^{|A|}(t) \) to \( t' \) we get a member of \( F_c \). Consequently by our assumption, \( p \Preceq^F_c q \), we see \( q \text{ may } x (A, t') \). Now using (5.8) we get \( q \text{ may } x (A, f_{c, L(A)}^{|A|}(t)) \). Since \( c \not\in L(q) \) we can also use proposition 5.3.2 to obtain \( q \text{ may } x (A, t) \).

So far when trying to limit the set of experiments we have concentrated on the test part of it. We now search for conditions which can limit the set of actions to signal when the process is known to be syntactical finite. The following lemma gives some limits for a class of processes.

**Lemma 5.3.10** Suppose \( p \in PL \) (i.e., finite or equally contains no occurrences of \( \text{rec } x \).) then there exists an \( n \in \mathbb{N} \) such that \( p \text{ may } x (A, t) \implies L(A) \subseteq L(p), \quad |A| \leq n \).

**Proof** The first consequent \( L(A) \subseteq L(p) \) is immediate from corollary 3.2.3 and independent of \( n \). The second is seen from \( p \) being finite as follows:

Let \( n \) be the maximal action depth of \( p \) \((n = \text{ad}(p))\) and suppose \( p \text{ may } x (A, t) \). This means \( \exists q \in D(A, p). (t, q) \longrightarrow^* \top / \bot \). By definition \( q \in D(A, p) \) implies \( \exists w \in W. A_w \cong A, p \overset{w}{\longrightarrow} q \). We show

\[ p \overset{w}{\longrightarrow} q \implies |w| \leq \text{ad}(p) \]
by natural induction on the size, ad(p), and the result follows.

\( \text{ad}(p) = 0 \): Inspecting the definition of ad we see that \( \text{ad}(p) = 0 \) implies \( p \) is either NIL or combinations of NIL through + or \( \| \). So \( p \xrightarrow{w} q \) must mean \( q = p \) and \( w = \varepsilon \). But \( |\varepsilon| = 0 \) so we are done.

\( \text{ad}(p) > 0 \): We use structural induction on \( p \).

\( p = \text{NIL} \): Then \( p \xrightarrow{w} q \) implies \( w = \varepsilon \)—ok.

\( p = a.p' \): \( a.p' \xrightarrow{w} q \) implies \( w = aw' \) and \( p' \xrightarrow{w'} q' \) for some \( q' \) such that \( \tilde{a}.q' = q \). Since \( \text{ad}(p') \leq 1 + \text{ad}(p') = \text{ad}(p) \) we by hypothesis get \( |w'| \leq \text{ad}(p') \). Clearly then \( |w| \leq \text{ad}(p) \).

\( p = p_1 + p_2 \): W.l.o.g. assume \( p \xrightarrow{w} q \) is due to \( p_1 \xrightarrow{w} q \). By the hypothesis of structural induction \( |w| \leq \text{ad}(p_1) \). Since \( \text{ad}(p_1 + p_2) \geq \text{ad}(p_1) \) we are done for this case.

\( p = p_1 \parallel p_2 \): The case \( w = \varepsilon \) is trivial, so suppose \( w = aw' \). Clearly \( p_1 \parallel p_2 \xrightarrow{aw'} q \) implies either \( p_1 = a.p'_1 \) and \( p'_1 \parallel p_2 \xrightarrow{w'} q'_1 \parallel q_2, q = \tilde{a}.q'_1 \parallel q_2 \) or similar for \( p_2 \). Suppose the former is true. Then since \( \text{ad}(p'_1 \parallel p_2) = \text{ad}(p'_1) + \text{ad}(p_2) < \text{ad}(a.p'_1 \parallel p_2) \) we by hypothesis of natural induction get \( |w'| \leq \text{ad}(p'_1 \parallel p_2) \). But \( |w| = |aw'| = 1 + |w'| \leq 1 + \text{ad}(p'_1 \parallel p_2) = \text{ad}(p) \) and we have concluded the inductive step.

\( \square \)

The following statements will elucidate some of the (mainly operational) implications when two terms are related via the syntactic preorder.

**Lemma 5.3.11**

a) \( p \preceq q, q \in RPL \Rightarrow p \in RPL \)

b) \( \tilde{\vartheta}(p) \neq \varepsilon \Rightarrow p \neq \text{NIL} \)

c) \( p \xrightarrow{a} q \Rightarrow \tilde{\vartheta}(q) \neq \varepsilon \)

**Proof**
a) is proved by induction on the length of the proof of \( p \preceq q \).

b) follows immediately from \( \text{NIL} \in RPL \) and the definition of \( \tilde{\vartheta} \).

c) follows by induction on the number of rules \( p \xrightarrow{a} q \) is obtained with.

\( \square \)

Because recursion constructors only occurs in processes \( p \in RPL \) we cannot have \( q = \text{rec} \times q' \) for a \( q' \in RCL \setminus RPL \). This enables us to deduce:

**Corollary 5.3.12** If \( p \in RCL \setminus RPL \) then:

a) \( p = \tilde{a}.p' \preceq q \) implies \( q = \tilde{a}.q' \) where \( p' \preceq q' \)

b) \( p = p_1 \parallel p_2 \preceq q \) implies \( q = q_1 \parallel q_2 \) where \( p_i \preceq q_i \) for \( i \in \mathbb{2} \)
It cause no problems to prove by structural induction:

**Corollary 5.3.13** If \( p \in RCL \) then there is a \( p' \in CL \) such that:
\[
p' \preceq p \text{ and } \bar{\theta}(p') = \bar{\theta}(p)
\]

**Lemma 5.3.14** Suppose \( \bar{\theta}(p_1 \parallel p_2) = \bar{\theta}(q_1 \parallel q_2) \) and \( p_1 \preceq q_1, p_2 \preceq q_2 \) for \( p_1, p_2, q_1, q_2 \in RCL \). Then \( \bar{\theta}(p_i) = \bar{\theta}(q_i) \) for \( i \in 2 \).

**Proof** Let arbitrary \( p, q \in RCL \) be given. An easy induction on the length of the proof of \( p \preceq q \) shows that \( \bar{\theta}(p) \) must be a prefix of \( \bar{\theta}(q) \):
\[
p \preceq q \Rightarrow \bar{\theta}(p) \subseteq \bar{\theta}(q)
\]

Hence \( \bar{\theta}(p_i) \subseteq \bar{\theta}(q_i) \) for \( i \in 2 \). We cannot have \( \bar{\theta}(p_1) \subseteq \bar{\theta}(q_1) \) since this clearly would imply \( \bar{\theta}(p_1 \parallel p_2) = \bar{\theta}(p_1 \parallel p_2) \subseteq \bar{\theta}(q_1 \parallel q_2) = \bar{\theta}(q_1 \parallel q_2) \) contradicting the assumption of the lemma. So we must have \( \bar{\theta}(p_1) = \bar{\theta}(q_1) \). In the same way we infer \( \bar{\theta}(p_2) = \bar{\theta}(q_2) \). \( \square \)

**Proposition 5.3.15** For \( w \in Act^* \), \( p \in RPL \) and \( q \in RCL \) we have:
\[
p \xrightarrow{w} q
\]
\[
\Downarrow
\exists p' \in PL, q' \in CL. p \succeq p' \xrightarrow{w} q', \bar{\theta}(q') = \bar{\theta}(q)
\]

**Proof** With the basic case trivial and lemma 5.3.16 in the inductive step we prove:
\[
w \in Act^*, p, q \in RCL, p \xrightarrow{w} q \succeq q' \in CL \text{ and } \bar{\theta}(q) = \bar{\theta}(q')
\]
\[
\Downarrow
\exists p' \in CL. \bar{\theta}(p) = \bar{\theta}(p') \text{ and } p \succeq p' \xrightarrow{w} q'
\]

by induction on \(|w|\). From this the proposition follows using corollary 5.3.13 and lemma 5.3.11. \( \square \)

**Lemma 5.3.16** For \( a \in Act \) and \( p, q \in RCL \) we have:
\[
P \xrightarrow{a} q \succeq q' \in CL \text{ and } \bar{\theta}(q) = \bar{\theta}(q')
\]
\[
\Downarrow
\exists p' \in CL. \bar{\theta}(p) = \bar{\theta}(p') \text{ and } p \succeq p' \xrightarrow{a} q'
\]

**Proof** By a) of lemma 5.3.11 we know \( p \xrightarrow{a} q \) only if \( \bar{\theta}(q) \neq \varepsilon \), so from \( \bar{\theta}(q) = \bar{\theta}(q') \) and b) of the same lemma we get \( q' \neq NIL \). We will use this fact when proving the lemma by induction on the number, \( n \), of rules used to infer \( p \xrightarrow{a} q \).

\( n = 1 \): since \( a \in Act \) only one rule comes into consideration: \( p = a.r \xrightarrow{a} \tilde{a}.r = q \). From \( NIL \neq \tilde{q}' \in CL \) and \( q' \leq \tilde{a}.r \) follows \( \tilde{q}' = \tilde{a}.r' \) where \( r' \preceq r \) and \( r' \in CL \). \( p = a.r \) implies \( r \in RPL \) and by \( r' \preceq r \) and \( r' \in CL \) then \( r' \in PL \). With \( p' = a.r' \) therefore \( q' \in PL \subseteq CL \) and \( p' \xrightarrow{a} q' \). Because both \( p \) and \( p' \) belongs to \( RPL \) we have \( \bar{\theta}(p) = \varepsilon = \bar{\theta}(p') \).

\( n > 1 \): We consider each inference rule in turn.
Now for the inductive step we consider the inference rules one by one.

Proof

Proposition 5.3.17 If \( w \in \text{Act}^* \), \( p, q \in \text{RPL} \) and \( r \in \text{RCL} \) then:

\[
p \preceq q, p \xrightarrow{w} r \Rightarrow \exists s \in \text{RCL}. q \xrightarrow{w} s, \bar{\theta}(r) = \bar{\theta}(s)
\]

Proof Proved along the lines of proposition 5.3.15 but using lemma 5.3.18 below in place of lemma 5.3.16.

Lemma 5.3.18 If \( a \in \text{Act} \) and \( p, p' \in \text{RCL} \) then:

\[
\bar{\theta}(p) = \bar{\theta}(p') \text{ and } p \succeq p' \xrightarrow{a} q'
\]

Proof By induction on the length of the proof of \( p' \preceq p \). There are three cases in the basis:

\( p = p' \): Let \( q = q' \).

\( p' = \text{NIL} \): Then \( p' \not\rightarrow \) and the implication holds vacuously.

\( p' = p''[\text{rec} \times. p''/x] \) and \( p = \text{rec} \times. p'' \): By the recursion rule \( \text{rec} \times. p'' = p \xrightarrow{a} q' \) follows directly from \( p''[\text{rec} \times. p''/x] \xrightarrow{a} q' \). Let \( q = q' \).

Now for the inductive step we consider the inference rules one by one.

\( p' \preceq p'' \), \( p'' \preceq p \): By induction on the structure of \( p' \in \text{RCL} \) (from the definition of \( \text{RCL} \) as extracted from \( \text{RBL} \)) we prove:

\[
\bar{\theta}(p') = \bar{\theta}(p'') = \bar{\theta}(p)
\]

\( p' \in \text{RPL} \): By definition of \( \bar{\theta} \) then \( \bar{\theta}(p') = \varepsilon \) and so \( \bar{\theta}(p) = \varepsilon \). Hence also \( p \in \text{RPL} \). From \( p'' \preceq p \) and lemma 5.3.11 then \( p'' \in \text{RPL} \). Therefore \( \bar{\theta}(p'') = \varepsilon = \bar{\theta}(p') = \bar{\theta}(p) \).
3.3.9 we get $(p' = \bar{a}.r')$. By corollary 5.3.12 $p' \triangleq p''$ then implies $p'' = \bar{a}.r''$ and $r' \triangleq r''$. Using the corollary once more we get $p = \bar{a}.r$ where $r'' \triangleq r$. From $a.\bar{q}(r) = \bar{q}(p') = \theta(p) = a.\bar{q}(r)$ clearly $\bar{q}(r') = \bar{q}(r)$ so by the hypothesis of structural induction $\bar{q}(r') = \bar{q}(r''') = \bar{q}(r)$. Therefore also $\bar{q}(p') = a.\bar{q}(r') = \bar{q}(p') = \bar{q}(p)$.

$p' = p'_1 \parallel p'_2$: We can assume $p' \in RCL \setminus RPL$ since we already have dealt with the case $p' \in RPL$. Similar as above we then from corollary 5.3.12 get $p'' = p''_1 \parallel p''_2$ and $p = p_1 \parallel p_2$ where $p'_1 \triangleq p''_1 \triangleq p_i$ for $i \in 2$. From lemma 5.3.14 we then conclude $\bar{q}(p'_1) = \bar{q}(p_i)$ and the rest follow by induction as in the last case.

Now where we know $\bar{q}(p') = \bar{q}(p'') = \bar{q}(p)$ we can use the main hypothesis of induction to find a $q''$ such that $p'' \xrightarrow{a} q'' \triangleq q'$ and $\bar{q}(q'') = \bar{q}(q')$. Again by induction $\exists q. p \xrightarrow{a} q \triangleright q''$, $\bar{q}(q) = \bar{q}(q'')$. Then also $q'' \triangleright q$ and $\bar{q}(q) = \bar{q}(q')$.

$p' = b.r'$, $p = b.r$ and $r' \triangleq r$: $b.r' \xrightarrow{a} q'$ implies $a = b$ and $q' = \bar{a}.q'$. From $p' = a.r'$ and $p = a.r$ follows $r, r' \in RPL$ so $\bar{q}(r') = \varepsilon = \bar{q}(r)$. Then choose $q = \bar{a}.r$ and we clearly have $q'' \triangleright q$ and $\bar{q}(q') = \bar{q}(q)$ so as $p = a.r \xrightarrow{a} \bar{a}.r = q$.

$p' = \bar{b}.r'$, $p = \bar{b}.r$ and $r' \triangleq r$: $\bar{b}.r' \xrightarrow{a} q'$ only if $r \xrightarrow{a} s'$ where $q' = \bar{b}.s'$. Then $\bar{q}(p) = \bar{q}(p')$ implies $\bar{q}(r) = \bar{q}(r')$ so by induction $r \xrightarrow{a} s$ for some $s \triangleright s'$ with $\bar{q}(s) = \bar{q}(s')$. With $q = \bar{b}.s$ then $p = \bar{b}.r \xrightarrow{a} q \triangleright \bar{b}.s' = q'$ and $\bar{q}(q) = \bar{q}(q')$.

$p' = p'_1 + p'_2$, $p = p_1 + p_2$ and $p'_1 \triangleq p_i$: W.l.o.g. we assume $p' \xrightarrow{a} q'$ derives from $p'_1 \xrightarrow{a} q'$. From the form of $p$ and $p'$ we deduce $p_i, p'_1 \in RPL$ and therefore $\bar{q}(p_i) = \varepsilon = \bar{q}(p'_1)$. By hypothesis of induction we get a $q$ such that $p \xrightarrow{a} q \triangleright q'$ and $\bar{q}(q) = \bar{q}(q')$. Because $p \xrightarrow{a} q$ this case is settled.

$p' = p'_1 \parallel p'_2$, $p = p_1 \parallel p_2$ and $p'_1 \triangleq p_i$: Suppose $p' \xrightarrow{a} q'$ is due to $p'_1 \xrightarrow{a} q'_1$ where $q' = q'_1 \parallel p'_2$. From $\bar{q}(p) = \bar{q}(p')$, $p'_1 \triangleq p_i$ and lemma 5.3.14 we get $\bar{q}(p_i) = \bar{q}(p'_1)$ so by induction $p_i \xrightarrow{a} q$ for some $q \triangleright q'_1$ with $\bar{q}(q) = \bar{q}(q')$. Letting $q = q_1 \parallel q_2$ it follows from $q'_1 \triangleq q_1$ and $p'_1 \triangleq p_i$ that $q' \triangleq q$. Also $\bar{q}(q) = \bar{q}(q_1) \parallel \bar{q}(p_2) = \bar{q}(q'_1) \parallel \bar{q}(p_2) = \bar{q}(q')$. Because $p_i \xrightarrow{a} q$ we by the rules for $\parallel$ directly have $p \xrightarrow{a} q$. The other case where $p'_2 \xrightarrow{a} q'_2$ is symmetric.

We are now in a position to prove the fundamental proposition:

**Proposition 5.3.19** For $p \in RPL$, $e \in E$ we have:

$p \triangleright e \Rightarrow \exists d \in Fin(p). d \triangleright e$

**Proof** Assume $e = (A, t)$. $p \triangleright e \Rightarrow \exists d \in Fin(p)$ implies that there exists $q \in D(A, p). (t, q) \xrightarrow{*} o_x$ where $o_x = \top$ if $x = \text{accept}$ and $o_x = \bot$ if $x = \text{reject}$. Now $q \in D(A, p)$ implies $p \xrightarrow{w} q$ for some $w$ such that $A_w \cong A$. Then from proposition 5.3.15 above there exists $d \in Fin(p)$ and $q' \in RCL$ such that $d \xrightarrow{w} q'$ and $\bar{q}(q) = \bar{q}(q')$. Clearly $q' \in D(A, d)$ and from lemma 3.3.9 we get $(t, q) \xrightarrow{*} o_x$ implies $(t, q') \xrightarrow{*} o_x$. Hence $d \triangleright e$ (A, t) = e.

We take full advantage of the previous results in the proof of the following.
Proposition 5.3.20 Let $d$ be a syntactical finite process (i.e., $d \in PL$ and so contains no occurrences of $\text{rec.}$) and $p \in RPL$. If $d \sqsubseteq_x p$ then $d \sqsubseteq_x d'$ for some $d' \in \text{Fin}(p)$ ($x = a/ \text{accept or } x = r/ \text{reject}).$

The proof is much like an equivalent proof of Hennessy with some minor adjustments to our set-up.

Proof Since $L(q)$ in general for $q \in RPL$ is finite there is a finite set $B \subseteq \text{Act}$ such that $L(d), L(p) \subseteq B$. Because $\text{Act}$ is infinite we can chose a $c \not\in B$. According to proposition 5.3.9 we have $q \sqsubseteq_x q'$ iff $q \sqsubseteq_x q'$ for arbitrary $q, q'$ with $L(q), L(q') \subseteq B$. Now because $d$ is finite lemma 5.3.10 ensures us a $n$ such that $d$ may $x (A, t) \Rightarrow L(A) \subseteq L(d), |A| \leq n$. So let $F_c(d) = \{(A, t) \in F_c : L(A) \subseteq L(d), |A| \leq n\}$. Clearly then $d$ may $x e$ and $e \in F_c$ implies $e \in F_c(d)$. Therefore to show $d \sqsubseteq_x q$ for a $q$ with $L(q) \subseteq B$ it is sufficient to show $q$ may $x e$ for those $e$ in $F_c(d)$ such that $d$ may $x e$. Let $F'_c$ denote this subset of $F_c(d)$.

A simple argument shows that $F_c(d)$ is finite and hence also $F'_c$. Since $L(d)$ is finite there is only finite many multisets $A$ with $L(A) \subseteq L(d)$ and $|A| \leq n$. Also for a given finite $A$ there can only be finite many $t$'s with $L(A, t) \subseteq F_c$. $(A, t) \in F_c$ implies $t$ is a test normal form i.e., of the form $t = \&_{j \in \mathbb{N}}(\nabla_{k \in \mathbb{N}_+} \bar{w}_{jk} \top)$. Furthermore since there exists a $t'$ such that $t = f_{c, L(A)}(t')$ we must have $|w_{jk}| \leq |A| + 1$ and $L(w_{jk}) \subseteq L(A) \cup \{c\}$. There is only finitely many strings with this property. Since there is no duplicates of the strings in $\nabla_{k \in \mathbb{N}_+} \bar{w}_{jk} \top$ there can only be finitely many on this form. Similar we see that there are finitely many tests of the form $\&_{j \in \mathbb{N}}(\nabla_{k \in \mathbb{N}_+} \bar{w}_{jk} \top)$. Hence a finite number of $e \in F_c$.

By the assumption of the lemma we know $d \not\sqsubseteq_x p$ and so $p$ may $x e$ for every $e \in F'_c$. From the previous proposition (proposition 5.3.19) we find a $d(e) \in \text{Fin}(p)$ for every $e \in F'_c$. $D = \{d(e) : e \in F'_c\}$ must be finite since $F'_c$ has the same property. Then because $D \subseteq \text{Fin}(p)$ and $\text{Fin}(p)$ is directed we can take $d'$ to be an upper bound of $D$ and $d'$ may $x e$ for every $e \in F'_c$. In general $q \in \text{Fin}(p)$ implies $L(q) \subseteq L(p)$ and it follows that $L(d') \subseteq B$ so we have $d \sqsubseteq_x d'$.

Proposition 5.3.21 The test preorders $\sqsubseteq_a$ and $\sqsubseteq_r$ extends $\preceq$ on $RPL$. I.e., $\preceq \subseteq \sqsubseteq_a, \sqsubseteq_r$.

Proof Suppose $p, q \in RPL$ and $p \preceq q$. Given $(A, t) \in E$ such that $p$ may $x$ we shall show $q$ may $x$ in order order to have $p \sqsubseteq_x q$. Similar as in the proof of proposition 5.3.19 we see it is enough to show

$$\forall p' \in D(A, p) \exists q' \in D(A, q) \bar{\theta}(p') = \bar{\theta}(q')$$

But this follows immediately from proposition 5.3.17.

Proposition 5.3.22 The test preorders $\sqsubseteq_x$ are algebraic over $RPL$, where $x$ either is $a$, $r$ or left out.

With the results obtained so far the proof is very similar to a corresponding proof of Hennessy.
Proposition 5.3.25  Given a \( \Sigma \)-domain, \( A \), assume the functions preserve \( \text{Fin}(A) \). Then the denotational preorder, \( \leq_A \), arising from \( A[] \) is algebraic on \( \text{REC}_\Sigma \). I.e.,

\[
A[p] \leq_A A[q] \text{ iff } \forall d \in \text{Fin}(p) \exists e \in \text{Fin}(q). A[d] \leq_A A[e]
\]
Proof

if: A consequence of the proposition above is \(\forall e \in \text{Fin}(q). A[e] \leq_A A[q]\) so from the assumption of the implication we get \(\forall d \in \text{Fin}(p). A[d] \leq_A A[q]\). Hence also \(\forall_A A[\text{Fin}(p)] \leq_A A[q]\). Since \(\forall_A A[\text{Fin}(p)]\) by Hennessy equals \(A[p]\) this actually reads \(A[p] \leq_A A[q]\).

only if: \(d \in \text{Fin}(p)\) and \(p \in \text{REC}_\Sigma\) implies \(d \in \text{FREC}_\Sigma\). So if \([\,\,]\) on elements of \(\text{FREC}_\Sigma\) yields elements of \(\text{Fin}(A)\) (i.e., finite elements) we get this implication as follows:

\(A\) is a \(\Sigma\)-domain and \(\forall d \in \text{Fin}(p). A[d] \leq_A A[p] \leq_A A[q]\) so \(\forall d \in \text{Fin}(p). A[d] \leq_A A[q]\). Because \(A[q] = \forall_A A[\text{Fin}(q)]\), \(\text{Fin}(q)\) is directed and \(A[d]\) is assumed to denote a finite element there exists an \(e \in \text{Fin}(q)\) such that \(A[d] \leq_A A[e]\).

We owe to show \(t \in \text{FREC}_\Sigma\) implies \(A[t] \in \text{Fin}(A)\). Using as hypothesis the assumption of the proposition that \(\forall f \in \Sigma\) of arity \(k\) we have \(f_A(A[t] ) \in \text{Fin}(A)\) where \(A[t] \in \text{Fin}(A)^k\) this easily follows by induction on the structure of \(t\). \[\square\]

From the results in chapter 3 and the characterization of the finite elements in proposition 5.1.2 it is seen that \(\textit{op}_*\) preserve finite elements in \(C_*\) when \(\ast \in \{\delta, v, \chi\}\). We then have:

**Corollary 5.3.26** The denotational preorders \(\preceq_\delta\), \(\preceq_v\) and \(\preceq_\chi\) are algebraic on \(\text{RPL}\).

Notice that from the proof of proposition 5.3.25 above it appears that \([\,\,]\) denotes finite elements when restricted to \(\text{PL}\) and by 3.3.19 on page 85 all finite elements are denotable by terms of \(\text{PL}\) so our different domains are actually finitary.

With the corollary it is now an easy matter to show the denotational models are fully abstract w.r.t. the corresponding preorders.

**Theorem 5.3.27** If \(p, q \in \text{RPL}\) then the different test preorders \(\preceq_a, \preceq_r\) and \(\preceq\) are (relative) precongruences and:

\[
\begin{align*}
\delta) & \quad [p]_{\delta} \preceq_{\delta} [q]_{\delta} \iff p \preceq_a q \\
v) & \quad [p]_{v} \preceq_{v} [q]_{v} \iff p \preceq_r q \\
\chi) & \quad [p]_{\chi} \preceq_{\chi} [q]_{\chi} \iff p \preceq q
\end{align*}
\]

**Proof** From the last corollary we know that \(\preceq_\delta, \preceq_v\) and \(\preceq_\chi\) are algebraic and from proposition 5.3.22 we also know that this is the case for \(\preceq_a, \preceq_r\) and \(\preceq\) so by the corresponding result for syntactic finite processes, theorem 3.3.1, \(\delta) - \chi)\) then follows.

The test preorders are seen to be precongruences because they now are know to agree with the corresponding denotational preorders which in turn are precongruence since the denotational maps are built from (relative) continuous and thereby monotone operators. \[\square\]
Index

Constants

⊥, 68
⊤, 67

Functions

[ ]ₚ, 81
[ ]ₜ, [ ]ᵥ, [ ]ₓ, 64

Operations

a., 23
∥, see semiword, parallel composition of

akin, 67
akin, 67
∇, 67
ψ, 27
▽, 71

Relations

α →, 65
γ →, 65
→, 68
<, see immediate predecessor
≤ₜ, ≤ₜ, ≤, 75
≤ₜ, ≤ₜ, ≤, 75
∈ₚ, 118
∈ₚ, ∈ₜ, ∈, 67
∈ₚ, 63
∈ₜ, ∈ₜ, ∈ₓ, 64
∈, see semiword, prefix of
≤, see smoother than
≤, see syntactic approximation
=ₚ, =ₜ, =, =ₚ, =ₚ, =ₚ, 94
∈ₚ, ≤ₚ, 95
Σ (between tests), 73
Γₜ, 88
Γₚ, 88

Alphabetical

Aₚ, 63
Aₜ, Aₜ, Aₓ, 64
Act, 59
Act, 64
action depth, 115
ad, 115
ad, 115
algebraic (a relation being ~), 114
alphabet, 18
a', 19
assignment, 86

B, 71
barred depth, 115
BL, 59, 60
BL(X), 86
BL, 65

Cₜ, Cᵥ, Cₓ, 60, 109
Cₚ, 63
canonic representative, 19
carrier, 60
cf, see composition form
χ, 36
CL, 65
composition congruent, 94
composition form, 93
concatenation, 23
connected elements, 22
cpo, 109

D(A,p), 67
DC₂, see downwards closure
DEDₜ, DEDₚ, 87
Δ, see alphabet
δ, 33
downwards closure, 18
e, see semiword, empty
$\theta$, 94
$\Theta$, 75
$\hat{\theta}$, 75, 113
$TL$, 67
trichotomy law, 20
$TSW$, see semiword, tree-

$UC$, see upwards closure
$v$, 34
upwards closure, 18

variables, 86

$W$, see word
word, 20

$X$, see variables
Part II

Tracing Partial Orders
Chapter 6

Pomsets

As mentioned in the presentation, the concept of labelled partial orders will be central for the models we are going to present. The basic idea is that labelled partial orders will represent individual behaviours of processes. In particular we will look at pomsets. We shall use the interpretation and graphical representation of pomsets from [Gra81]. That is

\[ a \prec b \bowtie d \]

is used to represent a behaviour of a process with four action occurrences, where the \( d \) occurrence is causally dependent on the others, the \( b \) occurrence is causally dependent on \( a \), but not on \( c \), a.s.o.

6.0 Basic Definitions

Pomsets are usually defined as isomorphism classes of labelled partial orders ([Gis88, Pra86]). We will look at labelled partial orders, also known as labelled posets, over an action alphabet \( \Delta \)—a countably infinite alphabet (fixed throughout the rest of this part of the thesis). We assume \( \Delta \) to be disjoint from \( \mathbb{N} \)—the nonnegative integers.

The labelled partial orders are defined on basis of a fixed ground set which is assumed to be closed under pairing and containing \( \mathbb{N} \) and \( \Delta \) (See e.g., [Hen87c] for a solution to the simple set equation \( S = \mathbb{N} \cup \Delta \cup (S \times S) \)).

Definition 6.0.1 Labelled Poset

A subset, \( X \), of the ground set together with a partial order (reflexive, transitive and antisymmetric), \( \leq \), and a labelling function \( \ell : X \to \Delta \) is called a labelled poset (lpo for short) and denoted \( \langle X, \leq, \ell \rangle \).

Given a lpo \( p \) then \( X_p = X, \leq_p = \leq \) and \( \ell_p = \ell \) if \( p = \langle X, \leq, \ell \rangle \).

The set of all such lpos is denoted \( \text{LPO} \).

Given two lpos, a morphism \( f : \langle X, \leq, \ell \rangle \to \langle X', \leq', \ell' \rangle \) of labelled posets is a function \( f : X \to X' \) such that
\[ x \leq y \Rightarrow f(x) \leq f(y) \quad \text{for all } x, y \in X \]
\[ \ell(x) = \ell(f(x)) \quad \text{for all } x \in X \]

An isomorphism \( f : p \rightarrow q \) of labelled posets is a bijection \( f : X_p \rightarrow X_q \) such that \( f \) and \( f^{-1} \) are morphisms of labelled posets (then also \( x \leq_p y \iff f(x) \leq_q f(y) \)). If such an isomorphism exists between \( p \) and \( q \) we write \( p \cong q \).

The empty lpo, \( \langle \emptyset, \emptyset, \emptyset \rangle \), is denoted \( \varepsilon \).

That \( LPO \) indeed is a set follows from the ground set being one. Observe that we use \( x, y, \ldots \) to range over elements of \( X_p \), where \( p \) is a lpo.

\( x \) and \( y \) are said to be concurrent/causally independent in a lpo \( p \),

\[ x \text{ co}_p y \iff x \not\leq_p y \text{ and } y \not\leq_p x \]

Notice that \( \text{co}_p \) is not reflexive! We say that \( Y \subseteq X_p \) is a \( \text{co}_p \)-set if all the elements of \( Y \) are concurrent in \( p \), i.e., if \( \text{co}_p|_{Y^2} = (Y \times Y) \setminus \{ (y, y) \mid y \in Y \} \) or alternatively if \( \leq_p|_{Y^2} = \{ (y, y) \mid y \in Y \} \).

If \( Y \) is a set and \( p = \langle X, \leq, \ell \rangle \) an lpo then the restriction of \( p \) to \( Y \), \( p|_Y \), is the lpo \( \langle X|_Y, \leq|_Y, \ell|_Y \rangle \).

For \( x \in X_p \) we sometimes (ambiguously) abbreviate \( p|_{\{x\}} \) by \( x \).

The definition of pomsets emerge almost immediately from that of lpos.

**Definition 6.0.2** Pomsets

The equivalence class of a lpo \( p \) under \( \cong \) is denoted \([p]\) and \( p \) is called a representative of the equivalence class. I.e., \([p] = \{ q \in LPO \mid q \cong p \} \). Whenever an lpo is denoted by a single symbol, \( p \), we define for convenience \( p \) to be \([p]\).

The set of all pomsets is then the quotient set of \( LPO \) by \( \cong \), \( LPO/\cong \) and is denoted \( \mathbb{P} \).

A pomset \( p \) is contained in pomset \( q \) if a representative of \( p \) can be embedded in a representative of \( q \). Formally: \( p \) is a subpomset of \( q \), written \( p \hookrightarrow q \), iff \( \exists Y. p = [q|_Y] \).

We have defined the notion of subpomset by means of a single representative so one should check that the definition is independent of what representative used in the definition. E.g., if \( q \cong q' \) it is easy to see that \( q|_Y \cong q'|_Y \), where \( Y' \) is the subset of \( X_{q'} \) isomorphic to \( Y \cap X_q \) under the lpo isomorphism holding between \( q \) and \( q' \). It will often be left to the reader to check that definitions regarding pomsets are well-defined in this sense.

For a pomset \( p \) and a set of pomsets \( Q \) we denote by \( Q(p) \) those pomsets of \( Q \) which are contained in \( p \), i.e., \( Q(p) = \{ q \in Q \mid q \hookrightarrow p \} \).

**Example:** If \( p \) is the pomset represented in (6.1) then e.g.,

\[ p \hookrightarrow p, \quad a \rightarrow c \rightarrow d \hookrightarrow p, \quad a \rightarrow d \hookrightarrow p \]

and

\[ \left\{ c, a \rightarrow d, \begin{matrix} b & \rightarrow & d \\ c & \rightarrow & d \end{matrix} \right\} \subseteq \mathbb{P}(p) \]
We overload notation and use $\varepsilon$ and $a$ to denote the empty pomset $\emptyset$ and the singleton pomset $\{a\}$ respectively. Similarly $a^n$ denote the multisingleton pomset $\{(a, k) \mid 1 \leq k \leq n\}$, $\{(x, x) \mid x = (a, k), 1 \leq k \leq n\}$, $\{(a, k) \rightarrow a\}$.

For a set of pomsets $P$ we adopt the notation $P^\varepsilon$ for $P \cup \{\varepsilon\}$.

Below we list different types of pomsets we shall deal with together with the symbols we tend to use for them.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A, B, \ldots \in M$</td>
<td>the multiset pomsets: ${p \in P \mid p \neq \varepsilon$ and $X_p$ is a co-set}</td>
</tr>
<tr>
<td>$A, B, \ldots \in S$</td>
<td>the set pomsets: ${p \in M \mid \forall x, y \in X_p, x \neq y \Rightarrow \ell_p(x) \neq \ell_p(y)}$</td>
</tr>
<tr>
<td>$a^n, b^n, \ldots \in N$</td>
<td>the multisingle multisingleton ${p \in M \mid \forall x, y \in X_p, \ell_p(x) = \ell_p(y)}$</td>
</tr>
<tr>
<td>$a, b, \ldots \in \Delta$</td>
<td>the singleton pomsets: ${p \in M \mid \forall x, y \in X_p, x = y}$</td>
</tr>
</tbody>
</table>

Notice that we by this notation have $a = a^1 = \{a\}$. The reader is obliged to sort out from the context the ambiguity arising from this notation in return for a more tractable presentation. The reader should also be aware that the sets $\Delta, N, S$ and $M$ are defined not to contain the empty pomset $\varepsilon$. As already stated e.g., $M$ is augmented with $\varepsilon$ by writing $M_{\varepsilon}$. In continuation with the notation above we then also have $\varepsilon = a^0 = \emptyset$.

It will not be necessary to deal with infinite pomsets in the following so we will throughout the rest of this part assume pomsets to be finite. More precisely: we shall only consider pomsets $p$ where $X_p$ is finite.

Having restricted ourselves to finite pomsets we can now for a pomset associate a unique multiplicity function over $\Delta$ which for each action tells how many elements in the pomsets that are labelled whit this action.

**Definition 6.0.3 Multiplicity Function**

A multiplicity function, $m$, (over $\Delta$) is a map $m : \Delta \rightarrow N$. $m$ is said to be finite if $m$ is 0 everywhere except on a finite subset of $\Delta$.

The set of multiplicity functions are partially ordered by

$$m \leq m' \text{ iff } \forall a \in \Delta. m(a) \leq m'(a)$$

Given a lpo $p$ the associated multiplicity function, $m_p$, is defined by $\forall a \in \Delta. m_p(a) = |\{x \in X_p \mid \ell_p(x) = a\}|$.

The multiplicity function, $m_p$, of a pomset $p$ is simply $m_p$.

The preorder induced on pomsets by the partial order $\leq$ on multiplicity functions is (ambiguously) denoted $\leq$ and defined by $p \leq q$ iff $m_p \leq m_q$.

It is easy to see that every finite set $M$ of multiplicity functions has a lub (least upper bound) $\bigvee M = m'$ where $m'$ is given by $\forall a \in \Delta. m'(a) = \max_{\leq}\{m(a) \mid m \in M\}$. If in addition every $m \in M$ is finite then so is $\bigvee M$. Also $m_p$ is finite for every $p \in P$ because we only deal with finite pomsets.
Observe that multisets are nothing else than a pomset representation of multiplicity functions. It is mainly for convenience that we have chosen to work with both notions.

**Definition 6.0.4 Pomset Property**

A lpo property, $P_\ast$, is $\cong$-invariant if it is preserved under lpo isomorphism:

$$p \cong q, P_\ast(p) \implies P_\ast(q)$$

$P_\ast$ is a *pomset property* if it is induced from a $\cong$-invariant lpo property, $Q_\ast$, in the following way:

$$P_\ast(p) \iff Q_\ast(p)$$

Observe that the $\cong$-invariance ensures the notion of pomset property to be well-defined. In the sequel we shall make no distinction between a pomset property and the lpo property it is induced from.

An example of a pomset property, $P_\ast$, is where $P_\ast(p)$ demands $\preceq_p$ to satisfy the trichotomy law: $\forall x, y \in X_p, x \preceq_p y$ or $y \preceq_p x$, i.e., that $\preceq_p$ shall be total. The set of pomsets having this property is denoted $W$ (words) and we write the property as $P_w$. Pomsets of $W$ are by Gischer [Gis88] alternatively called tonsets. We shall often write $w$ for $w \in W$, because of the one to one correspondence between $\Delta^*$ and $W$ (see [Sta81]).

We now give an example of a type of pomset property that can be defined in terms of a set of nonempty multisets.

**Definition 6.0.5 Multiset Induced Pomset Property**

Given $D \subseteq M$ we say that a pomset $p$ has the $P_{M\subseteq D}$-*property* if the (nonempty) multisets contained in $p$ are from $D$. Formally

$$P_{M\subseteq D}(p) \iff M(p) \subseteq D$$

The $P_{M\subseteq D}$-*pomsets* are those with the $P_{M\subseteq D}$-property and they are denoted $P_{M\subseteq D}$. 

It is easy to see that $P_{M\subseteq D}$ actually is a pomset property because it is induced from the lpo property:

$$P_{M\subseteq D}(p) \iff [p|_Y] \in D \text{ for every nonempty } co_p\text{-set } Y \subseteq X_p$$

and $co$-sets are preserved by $\cong$.

**Example:** Suppose $p = \frac{a}{b} \frac{c}{b} \frac{b}{a}$ and $q = \frac{a}{b} \frac{b}{c} \frac{b}{c}$. Then $P_{M\subseteq S}(p)$ because

$$M(p) = \{a, b, c, \frac{a}{b}, \frac{b}{c}\} \subseteq S$$

but $P_{M\subseteq S}(q)$ does not hold because $(b^2 \notin S)$

$$M(q) = \{a, b, c, \frac{b}{b}, \frac{a}{c}, \frac{a}{b} \} \not\subseteq S$$
6.1 Operations on Pomsets

Pomsets have been equipped with a variety of operations ([Gra81, Sta81, Gis88, Pra86]). In this part of the thesis we need only a few of these. Just as pomsets were defined on the basis of labelled posets we shall do so with the operations. The following two are both natural generalizations of concatenation of words: sequential and parallel composition.

**Definition 6.1.1 Sequential Composition of Pomsets**

Given two pomsets, \( p \) and \( q \). Their sequential composition, \( p \cdot q \), is obtained (informally) by taking their disjoint union (component wise), and making all elements of \( q \) causally dependent on all elements of \( p \). Formally:

\[
\text{For two lpos } p_0 \text{ and } p_1 \text{ we define their sequential composition } p_0 \cdot p_1 = (X, \leq, \ell), \text{ where } \\
X \text{ is the set } \{0\} \times X_{p_0} \cup \{1\} \times X_{p_1} \\
\leq \text{ is the partial order defined by } \\
\langle i, x \rangle \leq \langle j, y \rangle \text{ iff } i = j \text{ and } x \leq_{p_i} y \\
\text{ or } i = 0, j = 1 \\
\ell \text{ is the function } (i, x) \mapsto \ell_{p_i}(x)
\]

For two pomsets \( p_0 \) and \( p_1 \) we then define \( p_0 \cdot p_1 \) to be \([p_0 \cdot p_1]\). □

**Example:**

\[
a \xrightarrow{b} a \cdot c \xrightarrow{d} = a \xrightarrow{b} c \xrightarrow{d}
\]

**Proposition 6.1.2** Suppose \( p, p_0, p_1 \) and \( p_2 \) are lpos. Then

- \( p \cdot \varepsilon \cong p \cong \varepsilon \cdot p \)
- \( (p_0 \cdot p_1) \cdot p_2 \cong p_0 \cdot (p_1 \cdot p_2) \)

**Proof** To see the last property use as isomorphism (from the left hand side to the right hand side) the function given by:

\[
\langle 0, \{0, x\} \rangle \mapsto \langle 0, x \rangle \\
\langle 0, \{1, x\} \rangle \mapsto \langle 1, \{0, x\} \rangle \\
\langle 1, x \rangle \mapsto \langle 1, \{1, x\} \rangle
\]

□

As a corollary we immediately get that for pomsets \( \cdot \) is associative and has \( \varepsilon \) as left and right neutral element.

**Definition 6.1.3 Parallel Composition of Pomsets**

Given two pomsets, \( p \) and \( q \), their parallel composition, \( p \times q \), is simply the union (component wise) of \( p \) and \( q \). Formally:

For lpos \( p_0 \) and \( p_1 \) we define \( p_0 \times p_1 = (X, \leq, \ell) \), where
\[ X \text{ is the set } \{0\} \times X_{p_0} \cup \{1\} \times X_{p_1} \]
\[ \leq \text{ is the partial order defined by } \]
\[ \langle i, x \rangle \leq \langle j, y \rangle \text{ iff } i = j \text{ and } x \leq_{p_i} y \]
\[ \ell \text{ is the function } \langle i, x \rangle \mapsto \ell_{p_i}(x) \]

For pomsets \( p_0 \) and \( p_1 \) we define \( p_0 \times p_1 \) to be \([p_0 \times p_1] \).

Example:
\[
\begin{align*}
  a & \xleftarrow{b} a \times c \xrightarrow{d} d \\
  a & \xleftarrow{b} a \\
  c & \xrightarrow{d}
\end{align*}
\]

**Proposition 6.1.4** Suppose \( p, p_0, p_1 \) and \( p_2 \) are lpos. Then

- \( p \times \varepsilon \cong p \cong \varepsilon \times p \)
- \( p_0 \times p_1 \cong p_1 \times p_0 \)
- \( p_0 \times (p_1 \times p_2) \cong (p_0 \times p_1) \times p_2 \)

**Proof** The second property is seen by using as isomorphism the function given by:
\[
\begin{align*}
  \langle 0, x \rangle & \mapsto \langle 1, x \rangle \\
  \langle 1, x \rangle & \mapsto \langle 0, x \rangle
\end{align*}
\]
and the other properties are proved as in the last proposition. \( \square \)

So for pomsets \( \times \) is associative, commutative and has \( \varepsilon \) as left and right neutral element.

The next operator refines the different elements of a pomset into different pomsets (a formalization of the concept of “change of atomicity”).

**Example:** Consider the pomset \( a \xleftarrow{b} b \). Suppose we would like to refine the upper occurrence of \( b \) to \( e \xrightarrow{d} d \), the lower to \( c \xrightarrow{a} a \) and the \( a \) occurrence to \( b \xrightarrow{a} a \). Call this refinement \( \pi \) and the associated operator \( \langle \pi \rangle \)—then we would expect:
\[
\begin{align*}
  a & \xleftarrow{b} \langle \pi \rangle \\
  b & \xrightarrow{d} d \\
  a & \xrightarrow{e} a \\
  c & \xrightarrow{a}
\end{align*}
\]

Actually it does not make sense talk about the upper, lower, etc. occurrence of \( b \) in a pomset, but for a particular representative each individual element can be replaced by “its own” pomset (representative) thus obtaining the representative of, a new pomset. We now give a definition of this construction and then in a moment utilize this for the definition of a function from pomsets.

The construction is not as simple as the others and we need to introduce some additional notions.
Definition 6.1.5 Particular Refinement

Let \( p \) be a lpo. A particular refinement for \( p \) is a mapping \( \pi_p : X_p \to LPO \).

Given a lpo \( p \) and a particular refinement (p.ref. for short), \( \pi_p \), for \( p \), we can construct a new lpo, \( p<\pi_p> \), as follows: \( p<\pi_p> \) is \( \langle X, \leq, \ell \rangle \), where

\[
X \quad \text{is the set } \{ \langle x, x' \rangle \mid x \in X_p, x' \in X_{\pi_p(x)} \}
\]

\[
\leq \quad \text{is the partial order defined by}
\]

\[
\langle x, x' \rangle \leq \langle y, y' \rangle \iff x \leq_p y \text{ and } \quad x = y \Rightarrow x' \leq_{\pi_p(x)} y'
\]

\[
\ell \quad \text{is the function } \langle x, x' \rangle \mapsto \ell_{\pi_p(x)}(x')
\]

Notice that \( p<\pi_p> \) is a finite lpo. Following the idea of Gischer [Gis84] we introduce the following lpos

\[
C = \langle \{0, 1\}, \{\{0, 0\}, \{0, 1\}, \{1, 1\}\}, i \mapsto a_i \rangle \text{ i.e., } [C] = a_0 \rightarrow a_1
\]

\[
S = \langle \{0, 1\}, \{\{0, 0\}, \{1, 1\}\}, i \mapsto a_i \rangle \text{ i.e., } [S] = \begin{array}{c} a_0 \\ a_1 \end{array}
\]

where \( a_0 \) and \( a_1 \) are two fixed elements of \( \Delta \). For lpos \( p_0 \) and \( p_1 \) let \( \pi_C(p_0,p_1) \) denote the p.ref. for \( C \) given by \( \pi_C(p_0,p_1)(i) = p_i \) for \( i = 0, 1 \) and similar for \( \pi_S(p_0,p_1) \).

From the definitions it immediately follows that sequential and parallel composition can be derived from particular refinements of \( C \) and \( S \) in the following sense:

\[
p \cdot q = C<\pi_C(p,q)>
\]

\[
p \times q = S<\pi_S(p,q)>
\]

Therefore also \( p \cdot q = [C<\pi_C(p,q)>] \) and \( p \times q = S<\pi_S(p,q)> \). That is to say with the words of Gischer [Gis88] \cdot and \times are pomset definable operations on pomsets. Gischer actually make refinement into a operation itself (called substitution) but it would not allow the type of refinements we shall need. We therefore prefer to postpone the definition of the pomset refinement operation to the section dealing with sets of pomsets.

6.2 Two Partial Orders on Pomsets

The first relation on pomsets we are going to present is used to compare the “concurrency” of two pomsets.

Definition 6.2.1 \( \preceq \)-ordering on Pomsets

The preorder, \( \preceq \), on lpos is defined: \( p \preceq q \iff \) there exists bijective function \( X_q \to X_p \) which also is a morphism of lpos.

This preorder induce a partial order, ambiguously denoted \( \preceq \), on pomsets as follows:

\[
p \preceq q \iff p \preceq q
\]
\( p \preceq q \) can be read: the pomset \( p \) is smoother than \([\text{Gra81}]\) subsumed by \([\text{Gis88}]\) less nonsequential than the pomset \( q \).

Notice that for lpos \( p \) and \( q, \ p \preceq q \) does not imply \( p \cong q \). It is also useful to observe that \( p \preceq q \) implies \( m_p = m_q \).

**Example:**

\[
\begin{array}{c@{}c@{}c}
 a & \to b & c \\
 c & \to b & c \\
 c & \to d & c \\
\end{array}
\]

\( \preceq \)-downwards closure of a pomset \( p \), \( \{p' \in P \mid p' \preceq p\} \), is denoted \( \delta(p) \). Suppose \( P_\ast \) is a property of pomsets then \( \delta_\ast(p) \) will be a shorthand for the semi \( \preceq \)-downwards closure \( \{p' \in P \mid p' \preceq p \text{ and } P_\ast(p')\} \). E.g., \( \delta_\omega(p) = \{p' \in P \mid p' \preceq p \} \) and \( P_\omega(p') \} = \{p' \in W \mid p' \preceq p\} \). Though we might have \( p \not\in \delta_\ast(p) \) for some pomset property \( P_\ast \), we call it the \( \delta_\ast \)-closure.

From the definition of p.ref. we directly see:

**Proposition 6.2.2** Let lpos \( p \) and \( q \) be given with \( q = \langle X_p, \leq_p, \ell_p \rangle \) and \( \leq_q \subseteq \leq_p \). Furthermore suppose \( \pi \) and \( \pi' \) are p.ref.'s for both \( p \) and \( q \) \( \langle X_p = X_q \rangle \) such that \( \forall x \in X_p, X_{\pi'(x)} = \langle X_{\pi(x)}, \leq, \ell_{\pi(x)} \rangle, \leq \subseteq \leq_{\pi(x)} \). Then

\[
\begin{align*}
 p<\pi & \preceq q<\pi \\
 p<\pi & \preceq p<\pi'
\end{align*}
\]

The following alternative characterization of \( \preceq \) will often be more convenient to work with.

**Proposition 6.2.3** For pomsets \( p \) and \( q \) we have:

a) \( p \preceq q \) iff \( p = \langle X_{q'}, \leq_p, \ell_{q'} \rangle \) and \( \leq_p \supseteq \leq_{q'} \) for some \( q' \in q \)

b) \( p \preceq q \) iff \( \langle X_{p'}, \leq_q, \ell_{p'} \rangle = q \) and \( \leq_{p'} \supseteq \leq_q \) for some \( p' \in p \)

**Proof** Observe at first that in general \( p \preceq q \iff p \preceq r \) and \( p \cong q \iff p \cong r \) and \( p \preceq r \).

a) \( \text{id}_{X_{q'}} \) is a label preserving bijective function from \( X_{q'} \) to \( X_p \) because \( X_p = X_{q'} \) and \( \ell_p = \ell_{q'} \). By \( \leq_p \subseteq \leq_{q'} \) it is also order preserving. Hence \( p \preceq q' \) and since \( q' \in q \) means \( q' \cong q \) we get \( p \preceq q \) and so \( p \preceq q \).

only if: By definition \( p \preceq q \) implies the existence of a bijective function \( f : X_q \to X_p \) which also is a morphism of lpos. Then define \( q' \) to be \( \langle X_p, \leq_{q'}, \ell_{p'} \rangle \) where \( \leq_{q'} \) is given by

\[
x \leq_{q'} y \text{ iff } f^{-1}(x) \leq f^{-1}(y)
\]

Clearly \( q' \cong q \) and \( q' \in q \). Also \( p = \langle X_{q'}, \leq_p, \ell_{q'} \rangle \) so it remains to show \( \leq_{q'} \subseteq \leq_p \). Assume \( x \leq_{q'} y \). Then \( f^{-1}(x) \leq f^{-1}(y) \) by definition of \( q' \) and because \( f \) is bijective and a morphism of lpos therefore \( x = f \circ f^{-1}(x) \leq_p f \circ f^{-1}(y) = y \).

b) is proved similar. \( \square \)
With the alternative characterization of $\preceq$, proposition 6.2.2 above and the observations made by the definition of particular refinement we get:

\begin{equation}
(6.2)
\end{equation}

\begin{equation}
\cdot \text{ and } \times \text{ are } \succeq\text{-monotone in their left and right arguments}
\end{equation}

Similar we with appropriate p.ref.'s deduce from the above example that:

\begin{equation}
(6.3)
\end{equation}

We now turn to the second partial order on pomsets.

**Definition 6.2.4** $\sqsubseteq$-**ordering on Pomsets**

Given two pomsets $p$ and $q$. Then $p$ is a *prefix* of $q$, $p \sqsubseteq q$, if $p$ is a subpomset of $q$ and the elements of $p$ only dominates the elements of $p$ in $q$. Formally:

The lpo preorder, $\sqsubseteq$, is defined $p \sqsubseteq q$ iff there exists a $\leq_q$-downwards closed set $Y$ such that $p$ is isomorphic to the restriction of $q$ to $Y$. I.e.,

\[ p \sqsubseteq q \iff \exists Y. p \cong q|_Y \text{ and } \{ x \in X_q \mid \exists y \in Y. x \leq_q y \} \subseteq Y \]

The partial order, $\sqsubseteq \subseteq P \times P$, is induced from the lpo preorder by:

\[ p \sqsubseteq q \iff p \sqsubseteq q \]

$\pi$ is defined to be the function which for a pomset $p$ gives the $\sqsubseteq$-downwards closure of $p$:

\[ \pi(p) = \{ p' \in P \mid p' \sqsubseteq p \}. \]

That $p \sqsubseteq q$ implies $p \preceq q$ follows from $p \cong q|_Y$. Notice that $p \sqsubseteq p$ and $p \sqsubseteq q$ implies $m_p \leq m_q$. Also observe that $\{ x \in X_q \mid \exists y \in Y. x \leq_q y \} \subseteq Y$ just is a formalization of: $Y$ is $\leq_q$-downwards closed.

**Example:**

\[ a \xleftarrow{b} \xrightarrow{c} b \xrightarrow{d} \]

but $a \xrightarrow{b} d \not\subseteq a \xrightarrow{b} c \xrightarrow{d}$

As for the partial order $\preceq$ there is an alternative characterization of $\sqsubseteq$:

**Proposition 6.2.5** For pomsets $p$ and $q$ we have:

a) $p \sqsubseteq q$ iff $p' = q|_{X_{p'}}$ for some $p' \in p$ with $\{ x \in X_q \mid \exists y \in X_{p'}. x \leq_{q'} y \} \subseteq X_{p'}$

b) $p \sqsubseteq q$ iff $p = q'|_{X_p}$ for some $q' \in q$ with $\{ x \in X_q \mid \exists y \in X_p. x \leq_{q'} y \} \subseteq X_p$

**Proof** a), b) if: Immediate because $= \subseteq \cong$.

For the only if direction of a) and b) we by definition have

\[ \exists Y. p \cong q|_Y \text{ and } \{ x \in X_q \mid \exists y \in Y. x \leq_q y \} \subseteq Y \]

W.l.o.g. we can assume $Y \subseteq X_q$ (because $q|_Y = q|_{(X_q \cap Y)}$).
a) only if: Define $p'$ to be $q|_Y$. Obviously $p'$ is a representative of $p$ and because $Y$ is a subset of $X_q$ we have $Y = X_q|_Y = X_{p'}$. Hence $p' = q|_Y = q|_{X_{p'}}$ and $X_{p'}$ is $\leq_{p'}$-downwards closed.

b) only if: Here we shall find a representative of $q$ which $p$ is a part of. The idea will be to find a representative $q''$ of $q$ which has no elements in common with $p$ and then just replace that part of $q''$ which is isomorphic to $p$ with $p$ to obtain $q'$. The elements of $p$ are from the ground set which are composed of two-tuples. Clearly the “size”, $|x|$, of an element $x$ can be determined as follows

$$|x| = \begin{cases} |x_0| + |x_1| & \text{if } x = \langle x_0, x_1 \rangle \\ 1 & \text{otherwise (}x \in \mathbb{N} \text{ or } x \in \Delta) \end{cases}$$

If $p$ is empty we can just choose $q' = q$ so assume it is not. Then since we work with finite pomsets/lpos it make sense choose a $z \in X_q$ with maximal size according to the measure above. Define $q'' = \langle X, \leq, \ell \rangle$ where

- $X$ is the set $\{\langle x, z \rangle \mid x \in X_q\}$
- $\leq$ is the partial order defined by $\langle x, z \rangle \leq \langle y, z \rangle$ iff $x \leq_y y$
- $\ell$ is the function $\langle x, z \rangle \mapsto \ell_q(x)$

Evidently $q''$ is a representative of $q$ and $p$ is a lpo isomorphic to $q''|_{Y_z}$ where $Y_z$ is the $\leq_{q''}$-downwards closed set $\{\langle x, z \rangle \mid x \in Y\}$. By construction all elements of $X_{q''}$ have size greater than those of $X_p$ and so they cannot have any elements in common. The required $q'$ is then obtained by replacing all elements from $X_{q''}$ which under the lpo isomorphism equals the elements of $X_p$ with these corresponding elements of $X_p$.

With this alternative characterization it is useful to observe for lpos $p$ and $q$:

- $\{0\} \times X_p = X_{p\cdot0} = X_{p\cdot\varepsilon}$ and $\{1\} \times X_p = X_{\varepsilon\cdot p} = X_{\varepsilon\cdot x_p}$
- $Y \subseteq X_{p\cdot\varepsilon}$ implies $(p \cdot q)|_Y = (p \cdot \varepsilon)|_Y$
- $X_{p\cdot\varepsilon} \subseteq Y$ implies $(p \cdot q)|_Y \cong p \cdot (\varepsilon \cdot q)|_Y$
- $Y \subseteq X_{p\times\varepsilon}$ implies $(p \times q)|_Y = (p \times \varepsilon)|_Y$ (symmetric for $\varepsilon \times p$)
- $X_{p\times\varepsilon} \subseteq Y$ implies $(p \times q)|_Y \cong p \times (\varepsilon \times q)|_Y$ (symmetric for $\varepsilon \times p$)

Then evidently a pomset is a prefix of two parallel composed pomsets if it itself is the parallel composition of two prefixes of the two parallel composed pomsets. It is also easy to see $p \sqsubseteq q$ implies $p \sqsubseteq q \cdot r$ and $r \cdot p \sqsubseteq r \cdot q$. It takes more effort to prove the “reverse”:

**Proposition 6.2.6** If $p \sqsubseteq q \cdot r$ then either $p \sqsubseteq q$ or there exists a pomsets $r'$ such that $p = q \cdot r'$ and $r' \sqsubseteq r$. 

139
Proposition 6.2.7 Given two pomsets \( \mathbf{p} \) and \( \mathbf{q} \). Then

\[ \mathbf{p} \subseteq \mathbf{q} \Rightarrow \exists \mathbf{r} \in \mathbf{P}. \mathbf{p} \cdot \mathbf{r} \preceq \mathbf{q} \]

Proof Assume \( \mathbf{p} \subseteq \mathbf{q} \). By the alternative characterization of prefix we can find a representative \( p' \) of \( \mathbf{p} \) such that \( p' = \mathbf{q} |_{X_{p'}} \) and \( X_{p'} \) is \( \leq_{q} \)-downwards closed. Define \( r \) to be \( q |_{(X_q \setminus X_{p'})} \) and \( q' = (X, \preceq, \ell) \), where

\[ X = \text{the set } \{0\} \times X_{p'} \cup \{1\} \times (X_q \setminus X_{p'}) \]
\[ \preceq \text{ is the partial order defined by } \langle i, x \rangle \preceq \langle j, y \rangle \text{ iff } x \preceq y \]
\[ \ell \text{ is the function } \langle i, x \rangle \mapsto \ell_q(x) \]

Clearly \( q' \cong q \)--i.e., \( q' \) is a representative of \( \mathbf{q} \)--and from the definition of \( p', r \) and lpo sequential composition we see \( X_{p',r} = X_{q'} \) and \( \ell_{p',r} = \ell_{q'} \). To see \( \leq_{q} \subseteq \leq_{q'} \) assume \( \langle i, x \rangle \leq \langle j, y \rangle \). Then \( x \preceq y \) and if \( i = 0 = j \) we have \( x, y \in X_{p'} \) so \( x \preceq y \) then follows from \( \leq_{p'} \leq_{q} |_{X_{p'}} \). Similar if \( i = 1 = j \). If \( i = 0 \) and \( j = 1 \) then \( \langle i, x \rangle \preceq_{p',r} \langle j, y \rangle \) by the definition of \( p', r \). We are left with the case \( i = 1 \) and \( j = 0 \). This means \( x \preceq y \), \( x \in X_q \setminus X_{p'} \) and \( y \in X_{p'} \) but this is impossible because \( X_{p'} \) is \( \leq_{q} \)-downwards closed and we can exclude this case. So we actually have \( q' = (X_{p',r}, \leq_{q}, \ell_{p',r}) \) and \( \leq_{p',r} \supseteq \leq_{q'} \). The alternative characterization of \( \preceq \) then gives us \( p' \cdot r = [p' \cdot r] \preceq q' = q \) as we wanted. 

With this result it is easy to prove:

Proposition 6.2.8 For pomsets \( \mathbf{p} \) and \( \mathbf{q} \) we have

\[ \exists \mathbf{r}. \mathbf{p} \preceq \mathbf{r} \subseteq \mathbf{q} \iff \exists \mathbf{s}. \mathbf{p} \subseteq \mathbf{s} \preceq \mathbf{q} \]
In [Pra86, page 49] Pratt outlines an alternative proof. He defines prefix in another, but equivalent way: \( p \) is a prefix of \( q \) if \( \exists Y. p \equiv q|_Y \) and \( X_q \setminus Y \) is \( \leq_q \)-upwards closed. So this proposition can be seen as just a reformulation of his theorem.

**Proof** *only if*: Assume \( p \preceq r \subseteq q \). By the previous proposition we know there is a pomset \( r' \) such that \( r \cdot r' \subseteq q \). From \( p \preceq r \) and \( \preceq \)-monotonicity of \( \cdot \) then \( p \cdot r' \subseteq q \). But \( p \subseteq p \cdot r \) so we can just choose \( s = p \cdot r' \).

*if*: Suppose \( p \subseteq s \subseteq q \). Then there are representatives \( p' \) and \( q' \) of \( p \) and \( q \) respectively such that \( p' = s|_{X_{p'}} \), \( X_{p'} \) is \( \leq_s \)-downwards closed and \( q' = (X_s, \leq_{q'}, \ell_s) \) with \( \leq_s \supseteq \leq_{q'} \). Define \( r \) to be \( q'|_{X_{p'}} \). Then \( r \) is a lpo and to see that \( X_{p'} \) is \( \leq_r \)-downwards closed assume \( x \leq_r y \) in \( X_{p'} \). Then \( x \leq_{q'} y \) and from \( \leq_{q'} \subseteq \leq_s \) also \( x \leq_s y \). \( x \in X_{p'} \) follows now from the \( \leq_s \)-downwards closure of \( X_{p'} \). Hence \( r \subseteq q' \). We also have \( r = (X_{p'}, \leq_{q'}|_{X_{p'}}, \ell_{p'}) \), so from \( \leq_{q'} \subseteq \leq_s \) then \( \leq_r = \leq_{q'}|_{X_{p'}^2} \subseteq \leq_s|_{X_{p'}^2} = \leq_{p'} \). Thus \( p' \preceq r \subseteq q' \) and \( p = p' \preceq r \subseteq q' = q \). □

### 6.3 Sets of Pomsets

Sets of pomsets and operators on them are used extensively in the models we shall present, so we briefly treat them here. The two operations on pomsets \( \cdot \) and \( \times \) generalize to sets in the natural way e.g., \( P \cdot Q = \{ p \cdot q \mid p \in P, q \in Q \} \). We shall use \( \cup \) to denote the normal set union and \( \mathcal{P}(\cdot) \) the powerset operator. Also for a pomset property \( P \), \( \delta \) generalize to sets: \( \delta_s(Q) = \bigcup_{q \in Q} \delta_s(q) \).

The previously mentioned refinement operator for pomsets is defined using the particular refinement construction for lpos.

**Definition 6.3.1** Refinements

A \( \mathcal{P}(P) \)-refinement is a mapping \( \varrho : \Delta \rightarrow \mathcal{P}(P) \).

We say that a \( \mathcal{P}(P) \)-refinement, \( \varrho \), is \( \varepsilon \)-free iff \( \forall a \in \Delta. \varepsilon \notin \varrho(a) \) and \( \varrho \) is image finite if \( \varrho(a) \) is finite for every \( a \in \Delta \).

A particular refinement \( \pi_p \) for a lpo \( p \) is consistent with a \( \mathcal{P}(P) \)-refinement \( \varrho \) iff

\[
\forall x \in X_p. [\pi_p(x)] \in \varrho(\ell_p(x))
\]

The mapping associated with \( \varrho \) is now defined as \( \langle \varrho \rangle : P \rightarrow \mathcal{P}(P) \) with \( p\langle \varrho \rangle = \{ [p\pi_p] \mid \pi_p \text{ is a } \varrho \text{-consistent p.ref. for } p \} \) and generalized to sets of pomsets by \( P\langle \varrho \rangle = \bigcup_{p \in P} P\langle \varrho \rangle \).

For a finite lpo \( p \) and image finite refinement \( \varrho \) we notice that there is only finitely many different \( \varrho \)-consistent p.ref. for \( p \) (up to \( \cong \) in each \( x \in X_p \)) and consequently in general \( p\langle \varrho \rangle \) is a finite set of pomsets because we only work with finite pomsets. Also \( P\langle \varrho \rangle \) must be finite if \( P \) is a finite set of pomsets and \( \varrho \) is a image finite refinement.

**Example**: Consider the same pomset as in the example for particular refinement on page 135. Suppose \( \varrho \) is a \( \mathcal{P}(P) \)-refinement such that \( a \mapsto \{ b \rightarrow a \} \) and \( b \mapsto \{ c \rightarrow a, d \rightarrow a \} \).
Then
\[ a \prec_b b \prec_a = \begin{pmatrix} b \rightarrow a \leftarrow d & d \\ a \rightarrow c \leftarrow e & b \rightarrow a \leftarrow c \\ a \rightarrow d \leftarrow e & d \rightarrow b \leftarrow a \end{pmatrix} \]

Whereas it was quite obvious that \( \cdot \) and \( \times \) defined operations on sets of pomsets this is not so easy to see for \( \prec \). But we now prove that \( \prec \) actually defines a operation on sets of pomsets.

**Proposition 6.3.2** \( \prec \) is well-defined.

**Proof** From the definition of \( \prec \) we clearly see that is enough to show:

If \( \pi_p \) is a \( \prec \)-consistent p.ref. for a lpo \( p \) then \( p \cong q \) implies the existence of a \( \prec \)-consistent p.ref., \( \pi_q \), for \( q \) such that \( p \prec \pi_p \cong q \prec \pi_q \).

Let \( f \) be an isomorphism of lpos from \( q \) to \( p \). If \( \pi_p \) is a p.ref. for \( p \) then \( \pi_q := \pi_p \circ f \) is a p.ref. for \( q \). Also \( \pi_q \) is consistent with \( \prec \) because:

\[
\forall x \in X_p, [\pi_p(x)] \in f(\ell_p(x))
\]

\[
\Downarrow
\forall x \in f(X_q), [\pi_p(x)] \in \ell_p(x)
\]

\[
\Downarrow
\forall x \in X_q, [\pi_p(f(x))] \in [\ell_p(f(x))]
\]

\[
\Downarrow
\forall x \in X_q, [\pi_q(x)] \in [\ell_q(x)]
\]

To see \( p \prec \pi_p \cong t \prec \pi_q \) we show \( g : X_q \prec \pi_q \longrightarrow X_p \prec \pi_p \) given by \( \langle x, x' \rangle \mapsto \langle f(x), x' \rangle \) is an isomorphism of lpos.

It is seen from: \( g(X_q \prec \pi_q) = \{g(\langle x, x' \rangle) \mid \langle x, x' \rangle \in X_q \prec \pi_q\}\)

\[
= \{\langle f(x), x' \rangle \mid x \in X_q, x' \in X_{\pi_p(f(x))}\}
\]

\[
= \{\langle x, x' \rangle \mid x \in f(X_q), x' \in X_{\pi_p(x)}\} = X_p \prec \pi_p
\]

Clearly \( g \) is bijective and \( g^{-1} \) is \( \langle x, x' \rangle \mapsto \langle f^{-1}(x), x' \rangle \).

We have \( \ell_q \prec \pi_q(\langle x, x' \rangle) = \ell_{\pi_q(x)}(x') = \ell_{\pi_p(f(x))}(x') = \ell_p \prec \pi_p(\langle f(x), x' \rangle) = \ell_p \prec \pi_p(g(\langle x, x' \rangle)), \)

so \( g \) is label preserving and since:
\[ \langle x, x' \rangle \leq_{q<\pi_q>} \langle y, y' \rangle \]

by construction of \( q<\pi_q> \)

\[ x \leq_{q} y \text{ and } x = y \Rightarrow x' \leq_{\pi_q(x)} y' \]

\[ f(x) \leq_{p} f(y) \text{ and } f(x) = f(y) \Rightarrow x' \leq_{\pi_p(f(x))} y' \]

\[ f(x), x' \leq_{p<\pi_p>} \langle f(y), y' \rangle \]

by construction of \( p<\pi_p> \)

\[ g(\langle x, x' \rangle) \leq_{p<\pi_p>} g(\langle y, y' \rangle) \]

by definition of \( g \)

\( g \) is also order preserving and therefore a morphism of lpos. Similarly it is seen that \( g^{-1} \) is a morphism of lpos.

The difference between our refinement operation and Gischer’s substitution can be illustrated by the following example.

**Example:** Suppose \( p = a \rightarrow a \) and \( g \) is a \( P(P) \)-refinement with \( g(a) = \{b, c\} \). Then

\[
p<\varphi> = \left\{ b \rightarrow b, b \xleftarrow{c} c, d \rightarrow d, c \xrightarrow{d} d \right\}
\]

whereas the result by Gischer substitution would be

\[
\left\{ b \rightarrow b, c \xrightarrow{d} d \right\}
\]

The different operations enjoy a number of properties, many of them inherited from the corresponding properties of pomsets. Some of them are listed in:

**Proposition 6.3.3**

- \( \cdot, \times \) and \( \cup \) are associative
- \( \times \) and \( \cup \) are commutative
- \( \{\varepsilon\}<\varphi> = \{\varepsilon\}, \{a\}<\varphi> = g(a) \) and \( <\varphi> \) distributes over \( \cdot, \times \) and \( \cup \)

That \( <\varphi> \) distributes over \( \cdot \) may seem surprising. But if \( \pi \) is a \( g \)-consistent p.ref. for \( p_0 \cdot p_1 \) then one can find \( g \)-consistent p. refinements, \( \pi_{p_0} \) for \( p_0 \) and \( \pi_{p_1} \) for \( p_1 \) (just define \( \pi_{p_i}(x) = \pi(\langle i, x' \rangle) \) for \( i = 0, 1 \)) such that

\[
(p_0 \cdot p_1)<\varphi> \cong p_0<\pi_{p_0}> \cdot p_1<\pi_{p_1}>
\]

(the map \( \langle i, x' \rangle \mapsto \langle i, x' \rangle \) is an isomorphism from \( X_{p_0<\pi_{p_0}> <\pi_{p_1}>} \) to \( X_{p_0<\pi_{p_0}> <\pi_{p_1}>} \)).

Then we have \( [p_0<\pi_{p_0}> <\pi_{p_1}>] = [p_0<\pi_{p_0}> <\pi_{p_1}>] \). And of course then also \( (p_0 \cdot p_1)<\varphi> = p_0<\varphi> \cdot p_1<\varphi> \) which generalize to sets as well.

The partial order \( \subseteq \) on sets will be central to our models. \( \cup \) and natural extensions to sets are \( \subseteq \)-monotone, so we get:
Proposition 6.3.4 The operators \( \cdot, \cup, \times, \langle \cdot \rangle \) and \( \delta_\ast \) are \( \subseteq \)-monotone in all their arguments.

6.4 Two Types of Pomset Properties

The first type of pomset properties we shall consider is those where the property of a pomset is inherited to all subpomsets.

Definition 6.4.1 Hereditary Pomset Properties

A pomset property, \( P \), is hereditary, iff

\[
\forall p \in P. \ P(p) \implies p \implies P(q)
\]

A pomset being a singleton/ multiset/ set/ multiset pomset are examples of hereditary pomset properties because \( p \leftarrow q \) implies \( M(p) \subseteq M(q) \). Also the \( P_w \)-property (page 133) is hereditary.

The following three propositions relates hereditary pomset properties with sequential and parallel composition of pomsets.

Proposition 6.4.2 Let \( P \) be a hereditary pomset property. Then

\[
q \preceq p_0 \cdot p_1 \implies q \leftarrow p \implies P(q)
\]

Proof We prove the proposition for lpos which then generalizes to pomsets. Let there be given lpos \( p_0, p_1 \) and \( q \) such that \( q \preceq p_0 \cdot p_1 \) and \( P(q) \).

\( q \preceq p_0 \cdot p_1 \) implies the existence of a bijection \( f : X_{p_0 \cdot p_1} \longrightarrow X_q \) which also is a morphism of lpos.

By definition of \( \cdot \) we have \( X_{p_0 \cdot p_1} = \{0\} \times X_{p_0} \cup \{1\} \times X_{p_1} \) so we define \( q_i \) to be \( X_{p_i} \subseteq X_{p_0} \times X_{p_1} \) is defined by:

\[
x \leq q_i \iff f(\langle i, x \rangle) \leq f(\langle i, y \rangle)
\]

With this definition of \( q_i \) we only have to prove \( \leq_{p_i} \subseteq \leq_{q_i} \) in order to have \( q_i \leq p_i \). This is seen as follows:

\[
x \leq_{p_i} y \implies \langle i, x \rangle \leq_{p_0 \cdot p_1} \langle i, y \rangle \quad \text{definition of } p_0 \cdot p_1
\]

\[
\implies f(\langle i, x \rangle) \leq f(\langle i, y \rangle) \quad f \text{ is order preserving}
\]

\[
\implies x \leq_{q_i} y \quad f \text{ is bijective and label preserving, so we just have to show that } f \text{ and } f^{-1} \text{ preserve order: At first notice}
\]
\[ \langle i, x \rangle \leq_{q_0 \cdot q_1} \langle i, y \rangle \iff x \leq_{q_1} y \]
\[ \iff f(\langle i, x \rangle) \leq_{q} f(\langle i, y \rangle) \] definition of \( q_0 \cdot q_1 \)
\[ \iff f(\langle i, x \rangle) \leq_{q} f(\langle i, y \rangle) \] definition of \( q_i \)

Now suppose \( \langle i, x \rangle \leq_{q_0 \cdot q_1} \langle j, y \rangle \), \( i \neq j \). By definition of \( q_0 \cdot q_1 \) then \( i = 0, j = 1 \). But then also \( \langle i, x \rangle \leq_{p_0 \cdot p_1} \langle j, y \rangle \) and since \( f \) preserves the order of \( p_0 \cdot p_1 \) then \( f(\langle i, x \rangle) \leq_{q} f(\langle j, y \rangle) \).

Suppose now \( f(\langle i, x \rangle) \leq_{q} f(\langle j, y \rangle), i \neq j \). If \( i = 0 \) and \( j = 1 \) we by definition of \( q_0 \cdot q_1 \) also have \( \langle i, x \rangle \leq_{q_0 \cdot q_1} \langle j, y \rangle \). This settles the case because \( i = 1 \) and \( j = 0 \) would lead to a contradiction as follows:

If \( i = 1 \) and \( j = 0 \) we have \( \langle j, y \rangle \leq_{p_0 \cdot p_1} \langle i, x \rangle \) and \( \langle j, y \rangle \neq \langle i, x \rangle \). Since \( f \) preserves the order of \( p_0 \cdot p_1 \) and is injective we get \( f(\langle j, y \rangle) \leq_{q} f(\langle i, x \rangle) \) and \( f(\langle j, y \rangle) \neq f(\langle i, x \rangle) \). But this together with \( f(\langle i, x \rangle) \leq_{q} f(\langle j, y \rangle) \) contradicts the antisymmetry of \( \leq_{q} \).

It remains to show \( P_{*}(q_0) \) and \( P_{*}(q_1) \). Clearly \( q_{i} |_{f(\langle i \rangle \times x_{n})} \cong q_{i} \) so because \( P_{*} \) is hereditary and invariant under \( \cong \) the result follows. \( \square \)

**Proposition 6.4.3** Let \( P_{*} \) be a hereditary pomset property. Then

\[ q \preceq_{p_0 \times p_1} P_{*}(q) \]
\[ \Downarrow \]
\[ \exists q_{0}, q_{1} \text{. } q \preceq_{q_0 \times q_1} \text{ and } q_{i} \preceq_{p_i} P_{*}(q_{i}) \text{ for } i = 0, 1 \]

**Proof** The definitions of \( q_0 \) and \( q_1 \) so as the arguments are exactly as in the proof of the previous proposition, except that \( \cdot \) has to be exchanged to \( \times \) and we cannot infer

\[ f(\langle i, x \rangle) \leq_{q} f(\langle j, y \rangle), i \neq j \Rightarrow \langle i, x \rangle \leq_{q_0 \times q_1} \langle j, y \rangle \]

because \( i \neq j \) implies \( \langle i, x \rangle \sim_{q_0 \times q_1} \langle j, y \rangle \). For the same reason the proposition just states

\[ q \preceq_{q_0 \times q_1} \]

\( \square \)

**Proposition 6.4.4** Let \( P_{*} \) be hereditary pomset property. Then

\( a) \) \( \delta_{*}(p_0 \cdot p_1) \subseteq \delta_{*}(p_0) \cdot \delta_{*}(p_1) \)

\( b) \) \( \delta_{*}(p_0 \times p_1) = \delta_{*}(\delta_{*}(p_0) \times \delta_{*}(p_1)) \)

**Proof**

a) Suppose \( q \in \delta_{*}(p_0 \cdot p_1) \)—i.e., \( q \preceq_{p_0 \cdot p_1} P_{*}(q) \). Then by the last but one proposition there exists pomsets \( q_0 \) and \( q_1 \) such that \( q = q_0 \cdot q_1 \) and \( q_{i} \preceq_{p_i} P_{*}(q_{i}) \) for \( i = 0, 1 \). This implies \( q_{i} \in \delta_{*}(p_{i}) \) for \( i = 0, 1 \) and \( q = q_0 \cdot q_1 \in \delta_{*}(p_0) \cdot \delta_{*}(p_1) \).

b) We prove each inclusion in turn.

\( \subseteq \): Suppose \( q \in \delta_{*}(p_0 \times p_1) \). Then \( P_{*}(q) \) and \( q \preceq_{p_0 \times p_1} \). Using the last proposition we find pomsets \( q_0 \) and \( q_1 \) such that \( q \preceq_{q_0 \times q_1} \) and \( q_{i} \preceq_{p_i} P_{*}(q_{i}) \) for \( i = 0, 1 \). This gives \( q_0 \times q_1 \in \delta_{*}(p_0) \times \delta_{*}(p_1) \). From \( P_{*}(q) \) and \( q \preceq_{q_0 \times q_1} \) we then conclude \( q \in \delta_{*}(\delta_{*}(p_0) \times \delta_{*}(p_1)) \) as desired.

\( \supseteq \): Given \( q \in \delta_{*}(\delta_{*}(p_0) \times \delta_{*}(p_1)) \). Then \( P_{*}(q) \) and \( q \preceq_{q_0 \times q_1} \) for some \( q_{i} \in \delta_{*}(p_{i}) \) and \( i = 0, 1 \). This implies \( q_0 \preceq_{p_0} \) and \( q_1 \preceq_{p_1} \), so from the \( \preceq \)-monotonicity of \( \times \)

\[ q \preceq_{q_0 \times q_1} \preceq_{p_0 \times p_1} \preceq_{p_0 \times p_1} \]

\( q \in \delta_{*}(p_0 \times p_1) \) then follows from \( P_{*}(q) \).
Proposition 6.4.5 If $P_*$ is hereditary pomset property then $\pi \delta_*(p) \subseteq \delta_*\pi(p)$.

Proof Let a $q \in \pi \delta_*(p)$ be given. This means there is a $s$ such that $P_*(s)$ and $q \subseteq s \leq p$. $q \subseteq s$ implies $q \rightarrow s$, so because $P_*$ is hereditary we also have $P_*(q)$. By proposition 6.2.8 there is a pomset $r$ with $q \leq r \subseteq p$. Hence $q \in \delta_*\pi(p)$.

Notice that $P_*$ being hereditary was not used in $\supseteq$ of b) of proposition 6.4.4 and if we had closed the right hand side of a) similarly as in b) we would obtain equality.

But we shall deal with a certain type of pomset properties where it will not be necessarily to close in this way in order to obtain equality. For this type one can deduce/ synthesize the property for the sequential composition of two pomsets if they both have the property.

Definition 6.4.6 Dot Synthesizable Pomset Properties

A pomset property, $P_*$, is dot synthesizable, iff

\[(6.4) \forall p, q \in P. P_*(p) \text{ and } P_*(q) \text{ implies } P_*(p \cdot q)\]

The following proposition states a condition that ensures a pomset property to be dot synthesizable.

Proposition 6.4.7 A pomset property $P_*$ is dot synthesizable if

for every lpo $p$ and $Y \subseteq X_p$ with $\forall x \in X_p \setminus Y \forall y \in Y. x \epsilon_0 p y$ we have:

\[(6.5) \Downarrow P_*(p)\]

Proof We show that $p \cdot q$ has the $P_*$-property if $P_*$ fulfills the condition. By definition of $p \cdot q$ we have $(0, z) \leq_{p \cdot q} (1, v)$ for all $z \in X_p$ and $v \in X_q$ and as a consequence

$\forall x \in \{0\} \times X_p \forall y \in \{1\} \times X_q. x \epsilon_0 p \cdot q y$

We also have $P_*((p \cdot q)|_{X_{p \cdot q}})$ so from $p \equiv \epsilon \cdot p = (p \cdot q)|_{X_{p \cdot q}}$ we see $P_*((p \cdot q)|_{X_{p \cdot q}})$. Similar we get $P_*((p \cdot q)|_{X_{p \cdot q}})$. Using (6.5) we then conclude $P_*(p \cdot q)$.

With this proposition it is easy to prove that the multiset induced pomset properties are examples of dot synthesizable pomset properties:

Proposition 6.4.8 The multiset induced pomset properties are dot synthesizable.
Proof Given a set of multisets, $D$, we show that $P_{M \subseteq D}$ satisfies the condition in proposition 6.4.7 above. Let $p$ be any lpo and $Y$ a subset of $X_p$ with $\forall x \in X_p \setminus Y \forall y \in Y \omicron \phi_y$. The latter of course implies that any (nonempty) $co_p$-set, $Z$, must be contained in either $X_p \setminus Y$ or $Y$. So if $P_{M \subseteq D}(p|_{X_p \setminus Y})$ and $P_{M \subseteq D}(p|_Y)$ we conclude that $[p|_{Z}]$ must be contained in $D$, and so $P_{M \subseteq D}(p)$ as desired. 

That the condition in proposition 6.4.7 is not necessary for a pomset property to be dot synthesizable can be seen from the following example.

Example: Let $P_{ex}$ be the dot synthesizable pomset property defined by: $P_{ex}(p)$ iff every element of $p$ has an immediate neighbour with the same label. E.g., $P_{ex}(\epsilon)$ and $P_{ex}(a \xrightarrow{a} b \xrightarrow{b})$ but neither $P_{ex}(a)$ nor $P_{ex}(p)$ where $p = a \rightarrow b \rightarrow a \rightarrow b$. If $Y$ is the two elements of $X_p$ labelled with $b$ then $\forall x \in X_p \setminus Y \forall y \in Y \omicron \phi_y$. Also $[p|_{X_p \setminus Y}] = a \rightarrow a$ and $[p|_Y] = b \rightarrow b$ which both have the $P_{ex}$-property. As already stated $P_{ex}(p)$ does not hold wherefore the condition of the proposition is not satisfied.

Of course we cannot be sure that $\delta_*(p)$ is nonempty no matter whether we have to do with hereditary or dot synthesizable pomset properties. Take for instance the pomset property which is not fulfilled by any pomset. The next proposition states a condition which ensures $\delta_*(p)$ not to be empty.

**Proposition 6.4.9** Let $P_*$ be a dot synthesizable pomset property such that $P_*(\epsilon)$ and for every singleton pomset $a$, $P_*(a)$. Then $\delta_*(p) \neq \emptyset$ for every pomset $p$.

**Proof** Let a pomset $p$ be given. The proof is by induction on the number of elements in $p$. The basis $p = \epsilon$ holds by the assumption of the proposition. So assume $p \neq \epsilon$. We can then choose an $x \in X_p$ minimal w.r.t. $\leq_p$. Then $\{x\}$ is $\leq_p$-downwards closed and by the alternative characterization of prefix then $a := [p|_{\{x\}}] \subseteq p$. By proposition 6.2.7 we find a $p'$ such that $a \cdot p' \leq p$. Clearly $p'$ must have less elements than $p$, so by hypothesis of induction $\exists q \in \delta_*(p')$—i.e., $q \leq p'$ and $P_*(q)$. From the $\leq$-monotonicity of $\cdot$ then $a \cdot q \leq a \cdot p'$ and $p$. By the assumption of the proposition $P_*(a)$ and we know $P_*(q)$ so $P_*(a \cdot q)$ follows from proposition 6.4.7. Hence $a \cdot q \in \delta_*(p)$. 

As an example of the use of this proposition consider the pomset property $P_w$—a pomset being a word. Using proposition 6.4.7 one from the definition (trichotomy law) easily sees that $P_w$ is dot synthesizable. Also the other assumptions of the lemma are fulfilled, so we conclude $\delta_w(p) \neq \emptyset$ for every pomset $p$.

**Proposition 6.4.10** Let $P_*$ be a dot synthesizable pomset property. Then:

$$\delta_*(p_0 \cdot p_1) \supseteq \delta_*(p_0) \cdot \delta_*(p_1)$$

**Proof** Given $q \in \delta_*(p_0) \cdot \delta_*(p_1)$. Then $q = p_0' \cdot p_1'$ for some $p_i' \in \delta_*(p_i)$ and $i = 0, 1$. This implies $P_*(p_i')$ and $p_i' \leq p_i$ for $i = 0, 1$, so as a consequence of the $\leq$-monotonicity of $\cdot$ then $p_0' \cdot p_1' \leq p_0 \cdot p_1$, and $P_*(p_0' \cdot p_1')$ since $P_*$ is dot synthesizable. Hence $q \in \delta_*(p_0 \cdot p_1)$.
So if a pomset property, $P_\ast$, is both hereditary and dot synthesizable we from this proposition and a) of proposition 6.4.4 see:

$$\delta_\ast(p_0 \cdot p_1) = \delta_\ast(p_0) \cdot \delta_\ast(p_1)$$

If in addition $P_\ast$ holds for $\varepsilon$ and the singleton pomsets we from proposition 6.4.5 and the following proposition get:

$$\delta_\ast \pi(p) = \pi \delta_\ast(p)$$

In the following chapters we shall only meet such pomset properties.

**Proposition 6.4.11** Suppose $P_\ast$ is a dot synthesizable pomset property holding for $\varepsilon$ and the singleton pomsets. Then for every pomset $p$:

$$\delta_\ast \pi(p) \subseteq \pi \delta_\ast(p)$$

**Proof** Suppose $q \in \delta_\ast \pi(p)$. Then $P_\ast(q)$ and there is a pomset $r$ with $q \preceq r \sqsubseteq p$. As in the proof of proposition 6.2.8 we can find a $r'$ such that $q \cdot r' \preceq p$. We presume the same of $P_\ast$ as in proposition 6.4.9, so there is a $p' \in \delta_\ast(r')$. Hence $P_\ast(p')$ and by the $\preceq$-monotonicity of $\cdot$ also $q \cdot p' \preceq q \cdot r' \preceq p$. $P_\ast(q \cdot p')$ follows from $P_\ast(q)$ and $P_\ast(p')$. Because $q \sqsubseteq q \cdot p'$ we actually have $q \in \pi \delta_\ast(p')$. \qed
Chapter 7

BL—A Basic Process Language

As mentioned in the presentation we shall study degrees of nonsequentiality as “orthogonal” to existing study of branching, and as a consequence hereof the process expressions we shall use will be from a very basic language, BL, over the abstract set of action symbols, Δ, containing a combinator for internal nondeterminism beside combinators for sequencing and parallelism with auto-parallelism (but without communication).

BL consists of expressions of the form:

\[ E ::= a \quad \text{individual process labelled } a \in \Delta \]
\[ E_0 ; E_1 \quad \text{sequential composition of } E_0 \text{ and } E_1 \]
\[ E_0 \oplus E_1 \quad \text{internal nondeterministic composition of } E_0 \text{ and } E_1 \]
\[ E_0 \parallel E_1 \quad \text{parallel composition of } E_0 \text{ and } E_1. \]

In all models to come these binary operators are associative, a fact we shall make use of in examples together with the combinator precedence:

\[ \oplus < \parallel < ; \]

7.1 General Semantics

In the tradition as initiated in [HM80] our starting point will be the idea of an observer experimenting by doing tests on a black-box containing a process.

Tests consists in pushing buttons until some bulb is lightning up indicating the termination of the process. A direct test could be to try to push a button and a full test can then be considered as a maximal sequence of direct tests.

Within the branching tradition a widespread technique to increase an observers capability to distinguish nondeterministic processes is to provide the observer more sophisticated,
but natural means of making direct tests—e.g., in the readiness semantic where it is directly possible to test which buttons one successfully could push. How powerful these capabilities should be depends on the purpose and application [OH86]. In the line of this we shall look for natural direct tests which puts the observer in a position to discriminate degrees of nonsequentiality by processes, but remains faithful to the idea of an observer pushing buttons on a black-box.

Keeping the analogy of a human observer the weakest form of an direct test must be that of an observer pushing buttons using just one finger. But also simultaneously observations are conceivable [Mil80]. Clearly some power of the direct test is gained if the observer uses two fingers at the same time thereby enabling the observer to direct test whether two different labelled individual processes could be started at the same time. Another approach would be to realize the force used to push the button—reflecting how many individual processes with equal label could be started at the same time. These two directions for increasing the power of the direct test seems to span the possibilities for an observer experimenting through pushing buttons by the fingers. Of course the combination of these directions opens up for a large variety with one button direct tests at one extreme and finite many button push with realized force for each, at the other extreme. It is difficult to argue which one to choose in this spectrum and in the end it must be a matter of application. As an example of one application consider the situation where more processes have access to a common store. Here it would be suitable if only direct tests with at most one write in the common store is possible.

On the basis of sequences of direct test equivalences on a simple language, BL, will be defined. We can then investigate what consequences a choice of direct tests can have. However for an extension, RBL, of BL which allows change of atomicity, we shall later see that the actual choice is irrelevant if the equivalences are demanded to be congruences.

We now formalize the direct tests and add some “natural” requirements.

**Definition 7.1.1** A set of direct tests, \( G \), is a set of nonempty multisets satisfying:

\[
\Delta \subseteq G \\
A \iff B, B \in G \Rightarrow A \in G_e
\]

The first demand says that an observer at least should be capable of doing the weakest direct test: push one button. The second demand means that an observer capable of doing one direct test also should be able to do any weaker direct test.

Evidently our tests resembles the sequences of firing steps used to express nonsequential behaviours of processes in Petri nets [Rei85].

It is possible to carry through more quibbling observations as the partial order observations of [DM87] and in [BC87] transitions like \((a:b \parallel a)\) are possible. However one might argue that it is difficult to give “natural” intuition supporting such observations.
7.2 Operational Set-up

The sequence of direct test which can be performed will be build up from the direct test relation \( \Rightarrow_G \) holding through an \( A \in G \) between configurations, with each \( BL \)-expression being a possible start configuration. Configurations are expressions from \( CL \), which is almost like \( BL \) with \( \Delta \) extended with \( \dagger \) (a symbol distinct from those of \( \Delta \)). Intuitively \( \dagger \) represents the extinct action. Formally \( CL \) is defined to be the least set \( C \) satisfying:

\[
\begin{align*}
\dagger & \in C \\
BL & \subseteq C \\
E_0 ; E_1 & \in C \quad \text{if } E_0 \in C \text{ and } E_1 \in BL \\
E_0 || E_1 & \in C \quad \text{if } E_0, E_1 \in C
\end{align*}
\]

The construction of \( CL \) reflects the idea that control cannot pass ; before all previous actions are extinct.

**Example:** \( a \parallel (\dagger ; b) \in CL \) but \( \dagger \oplus a \not\in CL \) and \( a ; (\dagger ; b) \not\in CL \).

We shall often prove properties by induction on the structure of an \( E \in CL \). Strictly speaking we then first prove the property for expressions from \( BL \) and then look at \( \dagger \) and sequential/parallel composition afterwards. This implies that e.g., \( E = E_0 ; E_1 \) shall be treated two times with the only difference that for the first time we can assume \( E_0 \) not to contain \( \dagger \). We will therefore treat these cases together except at rare occasions where the distinction is crucial. The same applies for \( E = E_0 || E_1 \).

So \( \Rightarrow_G \) is actually a subset of \( CL \times G \times CL \). If \( \langle E, A, E' \rangle \in \Rightarrow_G \) we write this as \( E \xrightarrow{A} G E' \). One can think of this as \( E \) can evolve to \( E' \) when the direct test \( A \) is performed.

We shall follow DeNicola [Nic87] and Hennessy [Hen88a] when defining \( \Rightarrow_G \). Hennessy does this in an extended labelled transition system by means of a relation \( \Rightarrow \), which reflects the step of an internal computation, and by a relation \( \xrightarrow{G} \) for an external computation step corresponding to a direct test. The slight deviation from Hennessy in defining the relation, \( \Rightarrow \), for internal steps are manily due to differences in the languages considered.
**Definition 7.2.1** \( \rightarrow \subseteq CL \times CL \) and \( \rightarrow_G \subseteq CL \times G \times CL \) are defined as the least relations satisfying the following axioms and inference rules.

\[
\begin{array}{c}
a \rightarrow_G \uparrow \\
\hline
E_0 \rightarrow_G E_0' \\
\hline
E_0 \parallel E_1 \rightarrow_G E_0' \parallel E_1 \\
E_1 \parallel E_0 \rightarrow_G E_1' \parallel E_0'
\end{array}
\]

\[
\begin{array}{c}
E_0 \rightarrow_G E_0' ; E_1 \rightarrow_G E_0' ; E_1 \\
\hline
E_0 \rightarrow_G E_0' ; E_1 \rightarrow_G E_0' ; E_1 \rightarrow_G E_0' , A_0 \times A_1 \in G
\end{array}
\]

\[
\begin{array}{c}
\uparrow ; E \rightarrow E \\
\hline
E_0 \oplus E_1 \rightarrow E_0 \\
E_0 \oplus E_1 \rightarrow E_1
\end{array}
\]

\[
\begin{array}{c}
\uparrow \parallel E \rightarrow E \\
\hline
E_0 \parallel \uparrow \rightarrow E_0' \\
E_1 \parallel E_0 \rightarrow E_1' \parallel E_0'
\end{array}
\]

In this way an internal step either resolves an internal nondeterministic choice or removes an extinct action. The idea of using \( \rightarrow \) for other purposes than resolving internal nondeterministic choices is not inconsistent with Hennessy—he also uses \( \rightarrow \) to unfold recursive definitions.

Notice that the definition of \( \rightarrow_G \) is well-defined because of the premise \( A_0 \times A_1 \in G \) in the rule for a composed action and because we assume \( \Delta \subseteq G \) (for \( a \rightarrow_G \uparrow \)).

**Example:** Let \( G \) be a set of direct test containing \( a^2 \). Then

\[
a ; b \parallel a ; d \rightarrow_G \uparrow ; b \parallel \uparrow ; d \rightarrow \uparrow ; b \parallel \uparrow ; d \rightarrow_G \uparrow ; b \parallel \uparrow \rightarrow b \parallel \uparrow \rightarrow \uparrow
\]

The test relation, \( A \rightarrow_G \), is now defined as \( \rightarrow \rightarrow^* \rightarrow \rightarrow^* \rightarrow \rightarrow^* \rightarrow \rightarrow^* \) and for a sequence of direct tests \( s \in G^* \) and \( E, E' \in CL \) we define:

\[
E \rightarrow_G E', s = A_1 A_2 \ldots A_n
\]

iff

\[
\exists E_1, \ldots, E_n \in CL \exists A_1, \ldots, A_n \in G, n \geq 0.
\]

\[
E \rightarrow_G E_1 \rightarrow_G E_2 \rightarrow_G \ldots \rightarrow_G E_n = E'
\]

where the case \( n = 0 \) means \( E \rightarrow E' \).

With this notion of sequences of direct test it follows that any maximal sequence, \( s \), of direct is of the form \( E \rightarrow_G \uparrow \), so we can define our basic operational preorder:

**Definition 7.2.2** \( \preceq_G \subseteq BL \times BL \)
Notice that as expected the equivalence of $\preceq_G$, $\preceq_G$, identifies $a ; (b \oplus c)$ and $a ; b \oplus a ; c$.

Throughout this section we will fix $G$ and so will leave it out as a subscript of $\rightarrow_G$ and $\Rightarrow_G$ except when dealing with certain $G$'s. This will also be the case in the remaining sections whenever the direct test set $G$ in question is clear from the context.

Given a concrete sequence of internal and external steps, written $\Rightarrow$, we define its length as the total number of steps in the sequence. If $E$ under this sequence evolves to $E'$ we also write this as $E \Rightarrow E'$. This allows us to make induction on the length of a concrete sequence.

As a first result notice that by an easy induction on the length of $\Rightarrow$ (where $\Rightarrow$ is a concrete sequence for $E \Rightarrow E'$) one can prove:

**Proposition 7.2.3** Suppose $E \in BL$, $E_0, E_1 \in CL$ and $E_0 \Rightarrow E_0'$. Then

- $E_0 ; E \Rightarrow E_0' ; E$
- $E_0 \parallel E_1 \Rightarrow E_0' \parallel E_1$
- $E_1 \parallel E_0 \Rightarrow E_1 \parallel E_0'$

Since we only have a combinator for internal nondeterministic choice a natural question to raise is whether a processes reacts successfully to a test *iff* one of the syntactic “controlled behaviours” of it does. Such a behaviour can be regarded as a deterministic process ($\in DBL$) or configuration ($\in DCL$)—deterministic in the sense that no internal nondeterminism is explicit present in the form of a $\oplus$-combinator, but of course their might be indirectly as in $a \parallel a$. A behaviour would in Petri net terms correspond to a possible process/ concurrent behaviour of a Petri net system—more accurately it would correspond to to an occurrence net of a place/ transition net [BF88]. Formally:

**Definition 7.2.4** Behaviours

The set of configuration *behaviours*, $DCL$, is defined to be the $\oplus$-free expressions of $CL$. Similar $DBL = DCL \cap BL$ is the set of process behaviours.

The behaviours of a configuration expression is given by the map $\text{Beh} : CL \rightarrow \mathcal{P}(DCL)$ defined as follows:

- $\text{Beh}(\dagger) = \{\dagger\}$
- $\text{Beh}(a) = \{a\}$
- $\text{Beh}(E_0 ; E_1) = \text{Beh}(E_0) \cup \text{Beh}(E_1)$
- $\text{Beh}(E_0 \oplus E_1) = \text{Beh}(E_0) \cup \text{Beh}(E_1)$
- $\text{Beh}(E_0 \parallel E_1) = \text{Beh}(E_0) \parallel \text{Beh}(E_1)$

153
where $\text{Beh}(E_0) \land \text{Beh}(E_1)$ denotes $\{E'_0 ; E'_1 \mid E'_0 \in \text{Beh}(E_0), E'_1 \in \text{Beh}(E_1)\}$. Similar for $\parallel$.

Notice that $E \in BL$ implies $\text{Beh}(E) \subseteq DBL$.

Because $BL \subseteq CL$ we from the proposition below deduce a positive answer to the question whether a processes reacts successful to a test iff one of the syntactic “controlled behaviours” of it does.

**Proposition 7.2.5** For a configuration $E \in CL$ and $s \in G^*$ we have

$$E \leftrightarrow + \iff \exists F \in \text{Beh}(E). F \leftrightarrow +$$

Because in general $\dagger \in \text{Beh}(E)$ iff $E = \dagger$ this proposition is immediate from:

**Proposition 7.2.6** Given $s \in G^*$ and configurations $E$ and $F'$. Then:

$$\exists E' \Rightarrow E', F' \in \text{Beh}(E') \downarrow \exists F \in \text{Beh}(E). F \Rightarrow F'$$

**Proof** Both implication are proven by induction on the length of $\Rightarrow$ using the following three propositions. $\square$

**Proposition 7.2.7** Given configurations $E$ and $F'$ we have:

$$\exists E'. E \rightarrow E', F' \in \text{Beh}(E') \downarrow \exists F \in \text{Beh}(E). F \rightarrow^* F'$$

**Proof** By induction on the structure of $E$.

$E = \dagger$ or $E = a$: In both cases $E$ cannot do any internal step so the implication holds vacuously.

$E = E_0 ; E_1$: According to the definition of $\rightarrow$ there are two subcases:

$E_0 = \dagger$: Then $E' = E_1$ and $F' \in \text{Beh}(E_1)$. Now $\text{Beh}(E) = \dagger \cup \text{Beh}(E_1)$ so $F := \dagger \cup F' \in \text{Beh}(E)$ and of course $D \rightarrow F'$.

$E_0 \rightarrow E'_1$: I.e., $E' = E'_0 ; E_1$, so $F \in \text{Beh}(E')$ means $F' = F'_1 ; F_1$ for some $F'_0 \in \text{Beh}(E'_0)$, $F_1 \in \text{Beh}(E_1)$. By induction $\exists F_0 \in \text{Beh}(E_0). F_0 \rightarrow^* F'_0$. By proposition 7.2.3 this implies $F := F_0 ; F_1 \rightarrow^* F'_0 ; F_1 = F'$. Since $F_i \in \text{Beh}(E_i)$ for $i = 0, 1$ we also have $F \in \text{Beh}(E)$.

$E = E_0 \oplus E_1$: By definition of $\rightarrow$ then either $E' = E_0$ or $E' = E_1$. W.l.o.g. assume $E' = E_0$. Then $F \in \text{Beh}(E')$ means $F' \in \text{Beh}(E_0) \subseteq \text{Beh}(E_0) \cup \text{Beh}(E_1) = \text{Beh}(E_0 \oplus E_1)$ so we can just choose $F = F'$ since $F' \rightarrow^0 F'$. 

154
\( E = E_0 \parallel E_1 \): Again according to the definition of \( \rightarrow \rightarrow \) there are four possibilities. If the internal step \( E_0 \parallel E_1 \rightarrow \rightarrow E' \) derives from one of the axioms for \( \parallel \) the proof goes similar/symmetric as in the first subcase of \( E = E_0 ; E_1 \) and if it derives from one of the inference rules for \( \parallel \) it goes as in the second subcase.

\[ \square \]

**Proposition 7.2.8** If \( E \) and \( F' \) are configurations then:

\[ \exists F \in \text{Beh}(E). \; F \rightarrow \rightarrow F' \]

\[ \Downarrow \]

\[ \exists E'. \; E \rightarrow \rightarrow^* E', F' \in \text{Beh}(E') \]

**Proof** By induction on the structure of \( E \).

\( E = \uparrow \) or \( E = a \): Then \( \text{Beh}(E) = \{E\} \) and \( F = E \). Since \( E \) of this form can do no internal step the implication holds trivially.

\( E = E_0 ; E_1 \); \( F \in \text{Beh}(E_0 ; E_1) \) means \( F = F_0 ; F_1 \) where \( F_i \in \text{Beh}(E_i) \) for \( i = 0, 1 \).

\( E_0 = \uparrow \). Since \( F_0 \in \text{Beh}(\uparrow) \) implies \( F_0 = \uparrow \) we see that \( F \rightarrow \rightarrow F' \) implies \( F' = F_1 \).

Let \( E' = E_1 \). We have \( E = \uparrow ; E_1 \rightarrow \rightarrow E' \) with \( F' \in \text{Beh}(E') \) as desired.

\( E_0 \neq \uparrow \). Now \( F_0 \in \text{Beh}(E_0) \), \( E_0 \neq \uparrow \) implies \( F_0 \neq \uparrow \). Inspecting the definition of \( \rightarrow \rightarrow \) we see that \( F \rightarrow \rightarrow F' \) must be due to \( F_0 \rightarrow \rightarrow F'_0 ; F' = F_1 \). By hypothesis of induction there exists \( E'_0 \) such that \( E_0 \rightarrow \rightarrow^* E'_0 \) and \( F'_0 \in \text{Beh}(E'_0) \). Then also \( E \rightarrow \rightarrow^* E'_0 ; E_1 \) and \( F' \in \text{Beh}(E'_0) \); \( \text{Beh}(E_1) \). Choosing \( E' = E'_0 ; E_1 \) we are done.

\( E = E_0 \oplus E_1 \); \( F \in \text{Beh}(E) \) means \( F \in \text{Beh}(E_0) \) or \( F \in \text{Beh}(E_1) \). Suppose w.l.o.g. \( F \in \text{Beh}(E_0) \). Then by induction \( \exists E'. \; E_0 \rightarrow \rightarrow^* E', F' \in \text{Beh}(E') \). By definition of \( \rightarrow \rightarrow \) then also \( E_0 \oplus E_1 \rightarrow \rightarrow E_0 \rightarrow \rightarrow^* E' \) and thereby \( E \rightarrow \rightarrow^* E' \).

\( E = E_1 \parallel E_1 \): Has similar/symmetric subcases to those of \( E = E_0 ; E_1 \).

\[ \square \]

By an easy induction on the structure of \( E \) one can prove:

**Proposition 7.2.9** For configurations \( E \) and \( F' \) we have:

\[ \exists E'. \; E \xrightarrow{A} E', F' \in \text{Beh}(E') \]

\[ \Downarrow \]

\[ \exists F \in \text{Beh}(E). \; F \xrightarrow{A} F' \]

### 7.3 Denotational Set-up

The well-known trace models (not necessarily maximal) e.g., [OH86, Hoa85] are based on sets of sequences of actions from \( \Delta \) (words) and using the shuffle operator when dealing with \( \parallel \). These and related models can be viewed as abstractions of computations trees canonical associated with the process expressions. In the trace models for the equivalence
corresponding to the smallest set of direct tests the abstraction would consist in taking the set of words which constitute the paths from the root to the leaves of the computation tree as illustrated in:

\[
\begin{align*}
\text{Computation tree} & \quad \text{Set of words} \\
\text{a} \parallel \text{b} \oplus \text{a} ; \text{b} & \mapsto \{ \text{ab}, \text{ba} \} \\
\text{a} ; \text{b} \oplus \text{b} ; \text{a} & \mapsto \{ \text{ab}, \text{ba} \}
\end{align*}
\]

With these models in mind it offers itself for the generalized traces, to look for models based on sets of sequences of direct tests, i.e., subsets of \( G^* \). However because immediate tests are directed towards discovering concurrency as mirrored in pomsets and in order to clear the way for the more complicated model in the next chapter we shall devise corresponding models in the pomset framework. So the idea is to obtain a similar picture as above using pomsets in stead.

**Example:** If we intuitively think of \( \varphi \) as associating pomsets to expressions and \( \delta_\varphi \) gives the linearizations of pomsets we expect:

\[
\begin{align*}
\text{Set of} & \quad \text{Set of} \\
\text{pomsets} & \quad \text{pomsets} \\
\text{a} \parallel \text{b} \oplus \text{a} ; \text{b} & \overset{\varphi}{\mapsto} \begin{cases} 
\text{a} \\
\text{b}, \\
\text{a} \rightarrow \text{b}
\end{cases} \overset{\delta_\varphi}{\mapsto} \begin{cases} 
\text{a} \rightarrow \text{b}, \\
\text{b} \rightarrow \text{a}
\end{cases} \\
\text{a} ; \text{b} \oplus \text{b} ; \text{a} & \overset{\varphi}{\mapsto} \begin{cases} 
\text{a} \rightarrow \text{b}, \\
\text{b} \rightarrow \text{a}
\end{cases} \overset{\delta_\varphi}{\mapsto} \begin{cases} 
\text{a} \rightarrow \text{b}, \\
\text{b} \rightarrow \text{a}
\end{cases}
\end{align*}
\]

To make this picture precise and generalize to an arbitrary set of direct tests, \( G \), we shall at first look for pomsets which only contains multisets from \( G \). From \( G \subseteq M \) and the definition of multiset induced pomset properties we know that \( P_{M \subseteq G} \) are the pomsets we are looking for.

For arbitrary pomset properties, \( P_\varphi \) and \( P_{\varphi'} \), we denote \( P_\varphi \cap P_{\varphi'} \) by \( P_{\varphi,\varphi'} \) and similar for the \( \delta_\varphi \)-closure we for a pomset \( p \) denote \( \delta_\varphi(p) \cap P_{\varphi,\varphi'} \) by \( \delta_{\varphi,\varphi'}(p) \).

Notice that \( \delta_{\varphi,\varphi'}(p) \) alternatively may be written as \( \{ q \in P \mid q \preceq p, P_\varphi(p) \text{ and } P_{\varphi'}(p) \} \).

Next it seems natural to seek a pomset property reflecting the general nature of when the multisets of a pomset are in sequence, i.e., pomsets of the form \( A_1 \cdot A_2 \cdot \ldots \cdot A_n \). Pomsets of this form can be considered as “layered” in the sense that they may be viewed as “a linear order on top of a set of completely unordered pomsets (the individual multisets, \( A_i \)'s)”. One way to formalize this property is the following:
**Definition 7.3.1** \( \mathcal{P}_{\text{and}} \)-Property for Pomsets

A pomset \( \mathcal{P} \) is said to have the \( \mathcal{P}_{\text{and}} \)-property, \( \mathcal{P}_{\text{and}}(\mathcal{P}) \), iff for all \( x, x', y, y' \) in \( X_p \) we have:

\[
\begin{align*}
\text{if } \mathcal{P} &\vdash x \preceq_p y \text{ then } \\
\forall z. x \preceq_p z \Rightarrow y \preceq_p z
\end{align*}
\]

\[
\begin{align*}
\text{and } \\
\forall z. y \preceq_p z \Rightarrow x \preceq_p z
\end{align*}
\]

**Example:** \( a \xleftarrow{c} b \xrightarrow{d} c \) has the \( \mathcal{P}_{\text{and}} \)-property, \( a \xleftarrow{c} b \xrightarrow{d} d \) and \( a \xleftarrow{c} b \) has not.

**Proposition 7.3.2** The \( \mathcal{P}_{\text{and}} \)-property is hereditary and dot synthesizable.

**Proof** Because of the universal quantification of \( x \) and \( y \) in the definition of \( \mathcal{P}_{\text{and}}(\mathcal{P}) \) and because the partial order just is restricted in subpomsets it follows that the \( \mathcal{P}_{\text{and}} \)-property is hereditary.

It is also dot synthesizable as can be seen by using proposition 6.4.7: Let a lpo \( p \) and a subset \( Y \) of \( X_p \) given such that \( \mathcal{P}_{\text{and}}(p|_{X_p \setminus Y}), \mathcal{P}_{\text{and}}(p|_Y) \) and

\[
\forall x \in X_p \setminus Y \forall y \in Y. x \not\preceq_p y
\]

Suppose on the contrary \( \neg\mathcal{P}_{\text{and}}(p) \). By definition there must be \( x, y, z \in X_p \) with \( y \preceq_p x \preceq_p z \) and \( y \not\preceq_p z \). \( y \not\preceq_p z \) cannot mean \( z \preceq_p y \) because we then would get \( x \preceq_p y \) contradicting \( y \preceq_p x \), so actually:

\[
\forall x \in X_p \setminus Y \forall y \in Y. x \preceq_p z \preceq_p y
\]

If \( x, y \) and \( z \) all are in one of the two sets \( X_p \setminus Y \) and \( Y \), we get a contradiction to \( \mathcal{P}_{\text{and}}(p|_{X_p \setminus Y}) \) and \( \mathcal{P}_{\text{and}}(p|_Y) \). Otherwise one element must be in one of the sets and the remaining two in the other set. From (7.2) we see that at least one of the two elements belonging to the same set must be concurrent to the element in the other set—a contradiction to (7.1).

**Proposition 7.3.3** The \( \mathcal{P}_{\text{and}} \)-property has the following alternative characterization:

\[
(p \neq \varepsilon \text{ and } \mathcal{P}_{\text{and}}(p)) \iff \exists n \geq 1 \exists A_1, \ldots, A_n \in M. A_1 \cdot \ldots \cdot A_n = p
\]

From this and the definition of \( \mathcal{P}_{\mathcal{M} \subseteq \mathcal{G}, \text{and}} \) we immediately get:

\[
(p \neq \varepsilon \text{ and } \mathcal{P}_{\mathcal{M} \subseteq \mathcal{G}, \text{and}}(p)) \iff \exists n \geq 1 \exists A_1, \ldots, A_n \in \mathcal{G}. A_1 \cdot \ldots \cdot A_n = p
\]

We abbreviate \( \mathcal{P}_{\mathcal{M} \subseteq \mathcal{G}, \text{and}} \) by \( \mathcal{P}_G \). \( \mathcal{P}_{\mathcal{M} \subseteq \mathcal{D}} \) is clearly hereditary so from the propositions 6.4.8 and 7.3.2 we get:

**Corollary 7.3.4** The \( \mathcal{P}_G \)-property is hereditary and dot synthesizable.
With the above biimplications it is not hard to see that $G^*$ and $P_G$ coincide (are isomorphic), and as a consequence we shall often identify them in the sequel. So there is hope that we can base our models on subsets of $P_G$.

It only remains to establish a connection from $BL$-expressions to nonempty subsets of $P_G$. To this end we introduce a canonical map which give a natural association of sets of pomsets with $BL$-expressions.

**Definition 7.3.5 Canonical Pomset Association**

The *canonical associated pomsets* of a $BL$-expression is given by the map $\varphi : BL \rightarrow \mathcal{P}(P \setminus \{\varepsilon\}) \setminus \emptyset$ defined compositionally as follows:

- $\varphi(a) = \{a\}$
- $\varphi(E_0 ; E_1) = \varphi(E_0) \cdot \varphi(E_1)$
- $\varphi(E_0 \oplus E_1) = \varphi(E_0) \cup \varphi(E_1)$
- $\varphi(E_0 \parallel E_1) = \varphi(E_0) \times \varphi(E_1)$

**Example:**

$$\varphi((a \oplus b) ; (a \parallel c)) = \left\{ a \prec_c a, b \prec_c a \right\}$$

We can then let denotations in our models go via this map:

**Definition 7.3.6** $[\_]_G : BL \rightarrow \mathcal{P}(P_G \setminus \{\varepsilon\}) \setminus \emptyset$ with $[E]_G = \delta_G(\varphi(E))$.  

So, our $G$-model is finite sets of $P_G$-pomsets partially ordered by inclusion: $\subseteq$.

It is easy to check that the maps of the example on page 156 are correct and that they composed correspond to the denotational map just defined.

$[\_]_G$ together with the partial order induces a denotational preorder $\preceq_G$ over $BL$ by:

$$E_0 \preceq_G E_1 \text{ iff } [E_0]_G \subseteq [E_1]_G$$

Having models using sets of sequences from $\Delta^*$ in mind it is not hard to come up with:

**Theorem 7.3.7** $[\_]_G$ can be defined compositionally by:

- $[a]_G = \{a\}$
- $[E_0 ; E_1]_G = [E_0]_G \cdot [E_1]_G$
- $[E_0 \oplus E_1]_G = [E_0]_G \cup [E_1]_G$
- $[E_0 \parallel E_1]_G = \delta_G([E_0]_G \times [E_1]_G)$

Notice that $\delta_G$ here acts as the natural generalization of the shuffle/zip operator for $\Delta^*$.

**Proof** At first notice that corollary 7.3.4 enables us to apply the propositions 6.4.4 and 6.4.10 in the proof. We look at the different cases:
a: Evident by inspection of the definitions.

$E_0; E_1$: Follows directly from the fact that $\delta_G$ distributes over pomset sequential composition (proposition 6.4.4 and 6.4.10). See also the last case.

$E_0 \oplus E_1$: Similar because $\delta_G$ is a natural generalization to sets and therefore distributes over sets.

$E_0 \land E_1$: Similar because $G$ is a natural generalization to sets and therefore distributes over sets.

$E_0 \land E_1$:

\[
\begin{align*}
[E_0 \land E_1]_G & = \delta_G(\psi(E_0 \land E_1)) & \text{definition of } [\sqcap]_G \\
& = \delta_G(\psi(E_0) \times \psi(E_1)) & \text{definition of } \psi \\
& = \delta_G(\delta_G(\psi(E_0)) \times \delta_G(\psi(E_1))) & \text{proposition 6.4.4} \\
& = \delta_G([E_0]_G \times [E_1]_G) & \text{definition of } [\sqcap]_G
\end{align*}
\]

\[\square\]

### 7.4 Full Abstractness

The first proposition says that $\s_G$ is inherited in all $BL$-contexts.

**Proposition 7.4.1** $\s_G$ is a precongruence over $BL$.

**Proof** From theorem 7.3.7 we know a compositional definition of $[\sqcap]_G$ using $\subseteq$-monotone operators (proposition 6.3.4), and hence $\s_G$ is a precongruence.

**Theorem 7.4.2** $[\sqcap]_G$ is fully abstract w.r.t. $\preceq_G$, because

a) $\preceq_G$ is a precongruence w.r.t. $BL$

b) $E_0 \preceq_G E_1$ iff $E_0 \s_G E_1$

**Proof** a) is a consequence of proposition 7.4.1 and b) which in turn is a direct consequence of the proposition below.

**Proposition 7.4.3** For every $E_0, E_1 \in BL$ we have

\[
[E_0]_G \subseteq [E_1]_G \text{ iff } E_0 \preceq_G E_1
\]

**Proof** In the last section we saw:

\[
(p \neq \varepsilon \text{ and } P_G(p)) \text{ iff } \exists n \geq 1 \exists A_1, \ldots, A_n \in G. A_1 \cdot \ldots \cdot A_n = p
\]

from which we immediately get:

\[
P_G(p) \text{ iff } \exists n \geq 1 \exists A_1, \ldots, A_n \in G. A_1 \cdot \ldots \cdot A_n = p
\]

Recalling our convention to identify $G^*$ and $P_G$ we then from lemma 7.4.4 below get for $E \in BL$:

\[
(7.3) \quad [E]_G = \{A_1 \cdot \ldots \cdot A_n \in G^* \mid n \geq 1, E \overset{A_1}{\Rightarrow} \ldots \overset{A_n}{\Rightarrow} \uparrow\} = \{s \in G^* \mid E \overset{\uparrow}{\Rightarrow}\}
\]

with $\overset{\uparrow}{\Rightarrow}$ interpreted according to the convention below.

The proof is now a simple matter: $[E_0]_G \subseteq [E_1]_G$ iff $\{s \in G^* \mid E_0 \overset{\uparrow}{\Rightarrow}\} \subseteq \{s \in G^* \mid E_1 \overset{\uparrow}{\Rightarrow}\}$ iff $E_0 \preceq_G E_1$. \[\square\]
For simplicity of the following lemmas we shall temporarily adopt the notation $E \rightarrow^0 E'$ to mean $E = E'$ wherefore $E \rightarrow^0 E'$ also means $E \rightarrow^* E'$. For the same reason the lemmas are formulated slightly stronger than needed where they are used.

**Lemma 7.4.4** Given $E \in BL$ and multisets $A_1, \ldots, A_n \in G_\varepsilon (n \geq 1)$. Then

$$E \xrightarrow{A_1} \ldots \xrightarrow{A_n} \uparrow \iff \exists p \in \varphi(E). A_1 \cdot \ldots \cdot A_n \preceq p$$

**Proof** Before proving each implication separately notice that $p \in \varphi(E)$ implies $p \neq \varepsilon$ for $E \in BL$ and that every subexpression of $E \in BL$ belongs to $BL$ too.

**If:** By induction on the structure of $E$.

$E = a$: $\varphi(a) = \{a\}$ and we have $p = a$. Clearly $A_1 \cdot \ldots \cdot A_n \preceq a$ implies that exactly one $A_i = \{a\}$, the rest of them equal to $\emptyset$. The result then follows from $a \xrightarrow{\emptyset} \ldots \xrightarrow{\emptyset} a \xrightarrow{\emptyset} \uparrow$. $E = E_0 \cup E_1$: From $\varphi(E) = \varphi(E_0) \cdot \varphi(E_1)$ we then see $p = p_0 \cdot p_1$ where $p_i \in \varphi(E_i)$ for $i = 0, 1$. By lemma 7.4.7 $A_1 \cdot \ldots \cdot A_n \preceq p_0 \cdot p_1$ implies $n \geq 2$ and the existence of a $1 \leq j < n$ such that $A_1 \cdot \ldots \cdot A_j \preceq p_0$ and $A_{j+1} \cdot \ldots \cdot A_n \preceq p_1$. By hypothesis of induction then $E_0 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow$ and $E_1 \xrightarrow{A_{j+1}} \ldots \xrightarrow{A_n} \uparrow$. By proposition 7.2.3 then $E_0 ; E_1 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow ; E_1$. Since $\uparrow ; E_1 \rightarrow^* E_1$ we get $E_0 ; E_1 \xrightarrow{A_1} \ldots \xrightarrow{A_j} E_1 \xrightarrow{A_{j+1}} \ldots \xrightarrow{A_n} \uparrow \uparrow$ as desired.

$E = E_0 \cup E_1$: $p \in \varphi(E) = \varphi(E_0) \cup \varphi(E_1)$ implies $p \in \varphi(E_0)$ or $p \in \varphi(E_1)$. Suppose w.l.o.g. $p \in \varphi(E_0)$. By hypothesis of induction $E_0 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow$ so from the rules of $\rightarrow^*$ then also $E_0 \cup E_1 \rightarrow^* E_0 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow$.

$E = E_0 \parallel E_1$: $p \in \varphi(E) = \varphi(E_0) \times \varphi(E_1)$ implies $p = p_0 \times p_1$ for some $p_0 \in \varphi(E_0)$ and $p_1 \in \varphi(E_1)$. According to proposition 7.4.10 $A_1 \cdot \ldots \cdot A_n \preceq p_0 \times p_1$ implies the existence of multisets $A_1^0, \ldots, A_n^0, A_1^1, \ldots, A_n^1$ such that $A_i^1 \cdot \ldots \cdot A_n^1 \preceq p_i$ for $i = 0, 1$ and $A_j = A_i^0 \times A_i^1$ for $j = 1, \ldots, n$. This means $A_j^1 \lhd A_j$ so because $G$ has the closure property:

$$B \hookrightarrow C, C \in G \Rightarrow B \in G_\varepsilon$$

the $A_j^1$s actually belongs to $G_\varepsilon$. Hence we can use the hypothesis of induction to see $E_i \xrightarrow{A_i^1} \ldots \xrightarrow{A_i^n} \uparrow$ for $i = 0, 1$. By proposition 7.2.3 then $E_0 \parallel E_1 \xrightarrow{A_1^1} \ldots \xrightarrow{A_j^1} \uparrow \parallel \uparrow \uparrow$ and the result follows from $\parallel \parallel \rightarrow^* \uparrow$.

**Only if:** We shall also prove this implication by induction on the structure of $E$.

$E = a$: Since $a \not\rightarrow^*$ and $a \xrightarrow{A} F$ implies $A = a$ and $F = \uparrow$ there is exactly one $A_i = a$ and the rest equal to $\emptyset$. Recalling $\emptyset(= \varepsilon)$ neutral to $\cdot$ we see from $a \preceq a$ and $\varphi(a) = \{a\}$ that we are done.

$E = E_0 \cup E_1$: Because $E_0 \in BL$ we cannot have $E_0 \rightarrow^* \uparrow$. For purely structural reasons we cannot for any $E_0$ and $E_1$ have $E_0 \cup E_1 = \uparrow$ neither, so using lemma 7.4.5 on $E_0 ; E_1 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow$ we deduce $n \geq 2$ and the existence of a $1 \leq j < n$ such that $E_0 \xrightarrow{A_1} \ldots \xrightarrow{A_j} \uparrow$ and $E_1 \xrightarrow{A_{j+1}} \ldots \xrightarrow{A_n} \uparrow$. By hypothesis then $A_1 \cdot \ldots \cdot A_j \preceq p_0$
and $A_{j+1} \ldots A_n \preceq p_1$ where $p_i \in \varphi(E_i)$ for $i = 0, 1$. By $\preceq$-monotonicity of $\cdot$ then $A_1 \ldots A_j \cdot A_{j+1} \ldots A_n \preceq p_0 \cdot A_{j+1} \ldots A_n \preceq p_0 \cdot p_1 \in \varphi(E_0) \cdot \varphi(E_1)$.

$E = E_0 \oplus E_1$: Inspecting the definition of $\Rightarrow$ and $\Rightarrow_1$ one easily sees that $E_0 \oplus E_1 \Rightarrow_1 \ldots \Rightarrow_1$ implies $E_0 \oplus E_1 \Rightarrow \supseteq E_{\Rightarrow_1} \ldots \Rightarrow_1$ where $F = E_0$ or $F = E_1$. The result then follows from the hypothesis of induction and definition of $\varphi$.

$E = E_0 \parallel E_1$: Then $E_0 \parallel E_1 \Rightarrow_1 \ldots \Rightarrow_1$. Closings $E = \uparrow$ in lemma 7.4.6 we see that there are $A_1^0, \ldots, A_n^0, A_1^1, \ldots, A_n^1 \in G_c$ such that $A_j = A_j^0 \times A_j^1$ for $j = 1, \ldots, n$ and $E_i \Rightarrow_1 \ldots \Rightarrow_1$ for $i = 0, 1$. Using the hypothesis of induction together with (6.3):

$$(p \times q) \cdot (p' \times q') \preceq (p \cdot p') \times (q \cdot q')$$

the desired result is then obtained similarly as in the case $E = E_0 \parallel E_1$.

Notice that we only used the closure property of $G$ in the if part of the proof.

**Lemma 7.4.5** Suppose $n \geq 1$ and $E_0 \parallel E_1 \Rightarrow_1 \ldots \Rightarrow_1 E$ for $A_1, \ldots, A_n \in G_c$ and $E_0 \parallel E_1 \in CL$. Then either

a) $E_0 \Rightarrow \uparrow; E_1 \Rightarrow_1 \ldots \Rightarrow_1 E$ or

b) $E_0 \Rightarrow_1 \ldots \Rightarrow_1 E_1 \Rightarrow_1 \ldots \Rightarrow_1 E$ for a $1 \leq j < n$ or

c) $E_0 \Rightarrow_1 \ldots \Rightarrow_1 E_1 \Rightarrow \uparrow E$ or

d) $E_0 \Rightarrow_1 \ldots \Rightarrow_1 E_0', E_1 = E$ for some $E_0' \in CL$

**Proof** At first we prove by natural induction for arbitrary $m$ and $F, E_0 \parallel E_1 \in CL$:

$$E_0 \parallel E_1 \Rightarrow \Rightarrow^m F$$

(7.4)

$\downarrow$

i) $E_0 \Rightarrow \uparrow; E_1 \Rightarrow \Rightarrow F$ or

ii) $E_0 \Rightarrow \Rightarrow F_0'; F_0' \parallel E_1 = F$

$m = 0$: Here $E_0 \parallel E_1 = F$ and we can choose $F_0' = E_0$.

$m > 0$: Then for some $H \in CL$ we have $E_0 \parallel E_1 \Rightarrow H \Rightarrow \Rightarrow^{m-1} F$. According to the definition of $\Rightarrow$ there are two cases:

1. $E_0 = \uparrow$ and $H = E_1$: I.e., $E_1 \Rightarrow \Rightarrow^{m-1} F$ and i) holds.

2. $E_0 \Rightarrow H_0$ and $H_0; E_1 = H$: From the hypothesis of induction used on $H_0; E_1 \Rightarrow \Rightarrow^{m-1} F$ we get either $(H_0 \Rightarrow \Rightarrow \Rightarrow \Rightarrow H_0, E_1 \Rightarrow \Rightarrow F)$ or $(H_0 \Rightarrow \Rightarrow F_0', F_0'; E_1 = F)$. In the former case i) is established because $E_0 \Rightarrow H_0 \Rightarrow \Rightarrow \Rightarrow$ and in the latter case we get ii).

Using (7.4) we can now prove the lemma by induction on $n$.

$n = 1$: If $A_1 = \emptyset$ we have $E_0 \parallel E_1 \Rightarrow \Rightarrow E$ and we can use (7.4) to see that c) or d) holds.

So assume $A_1 \neq \emptyset$ and we have $E_0 \parallel E_1 \Rightarrow \Rightarrow F \Rightarrow_1 F' \Rightarrow \Rightarrow E$ for some $F, F' \in CL$.

We consider two case according to (7.4):
i) Here we have $E_0 \rightarrow^* \uparrow; E_1 \rightarrow^* F \xrightarrow{A_1} F' \rightarrow^* E$ and a) holds.

ii) $E_0 \rightarrow^* F''_0; E_1 \xrightarrow{A_1} F' \rightarrow^* E$. Since $A_1 \neq \emptyset$ we must have $F'' = F''_0; E_1$ where $F''_0 \xrightarrow{A_1} F''$. Using (7.4) on $E_1 \rightarrow^* E$ we see $F'' \rightarrow^* \uparrow; E_1 \rightarrow^* E$ or $F'' \rightarrow^* E$ in the former case we have $E_0 \rightarrow^* F'' \xrightarrow{A_1} F' \rightarrow^* E$ and c) holds. In the latter case $E_0 \rightarrow^* F'_0 \xrightarrow{A_1} F'' \rightarrow^* E_0; E_0' ; E_1 = E$ so here d) holds.

$n > 1$: Then $E_0 ; E_1 \xrightarrow{A_1} F \xrightarrow{A_2} \ldots \xrightarrow{A_n} E$. From the case $n = 1$ we know that for $E_0 ; E_1 \xrightarrow{A_1} F$ there are the following three main possibilities:

$E_0 \rightarrow^* \uparrow; E_1 \xrightarrow{A_1} F$: Then also $E_1 \xrightarrow{A_2} \ldots \xrightarrow{A_n} E$ and a) holds.

$E_0 \xrightarrow{A_1} \uparrow; E_1 \rightarrow^* F$: Here we get $E_1 \xrightarrow{A_2} \ldots \xrightarrow{A_n} E$ and b) is established with $j = 1$.

$E_0 \xrightarrow{A_1} F''_0; E_1 = F$: The hypothesis of induction used on $F''_0; E_1 \xrightarrow{A_2} \ldots \xrightarrow{A_n} E$ yields:

a') $F''_0 \rightarrow^* \uparrow; E_1 \xrightarrow{A_2} \ldots \xrightarrow{A_n} E$ or

b') $F''_0 \xrightarrow{A_2} \ldots \xrightarrow{A_n} \uparrow; E_1 \xrightarrow{A_j \uparrow} \ldots \xrightarrow{A_n} E$ for a $2 \leq j < n$ or

c') $F''_0 \xrightarrow{A_3} \ldots \xrightarrow{A_n} \uparrow; E_1 \rightarrow^* E$ or

d') $F''_0 \xrightarrow{A_2} \ldots \xrightarrow{A_n} E_0; E_0' ; E_1 = E$

Clearly we have $E_0 \xrightarrow{A_1} \uparrow$ in the case a') thereby getting b) for $j = 1$. In the remaining cases b'), c') and d') we directly get b), c) and d) respectively.

\[ \Box \]

**Lemma 7.4.6** Suppose $n \geq 1$ and $E_0 \parallel E_1 \xrightarrow{A_1} \ldots \xrightarrow{A_n} E$ for $A_1, \ldots, A_n \in \mathcal{G}_e$ and $E_0 \parallel E_1 \in CL$. Then there are $E'_0, E'_1 \in CL$ and $A'_0, \ldots, A'_n \in \mathcal{G}_e$ such that

$A_j = A'_j \times A'_j$ for $j = 1, \ldots, n$ and

$E_i \xrightarrow{A'_i} \ldots \xrightarrow{A'_n} E'_i$ for $i = 0, 1$ and

$E'_0 \parallel E'_1 = E$ or $E'_0, E'_1 = \{\uparrow, E\}$

**Proof** At first we by natural induction prove for arbitrary $m$ and $E_0, E_1 \in CL$ that $E_0 \parallel E_1 \rightarrow^m E$ implies the existence of some $E'_0, E'_1 \in CL$ such that:

(7.5)

$$E_0 \rightarrow^* E'_0; E_1 \rightarrow^* E'_1 \text{ and } E'_0 \parallel E'_1 = E \text{ or } \{E'_0, E'_1\} = \{\uparrow, E\}$$

$m = 0$: Then $E = E_0 \parallel E_1$ and we can choose $E'_i = E_i$.

$m > 0$: This means $E_0 \parallel E_1 \rightarrow F \rightarrow^m E$. According to the definition of $\rightarrow$ there are four cases:

$E_0 = \uparrow$ and $E_1 = F$: Choose $E'_0 = \uparrow, E'_1 = E$ and we are done.

$E_1 = \uparrow$ and $E_0 = F$: Symmetric.
$E_0 \rightarrow F'_0$ and $F'_0 \parallel E_1 = F$: Use the hypothesis of induction on $F'_0 \parallel E_1 \rightarrow^{m-1} E$ to find $E'_0, E'_1$ such that $F'_0 \rightarrow^* E'_0, E_1 \rightarrow^* E'_1$ and $(E'_0 \parallel E'_1 = E$ or $\{E'_0, E'_1\} = \{\dag, E\})$. The result then follows because $E_0 \rightarrow^* F'_0 \rightarrow^* E_0$ or equally $E_0 \rightarrow^* E_0$.

$E_1 \rightarrow F'_1$ and $E_0 \parallel F'_1 = F$: Symmetric to the last case.

Next we prove the lemma by induction on $n$:

$n = 1$: We have $E_0 \parallel E_1 \rightarrow^* F \xrightarrow{A_1} H \rightarrow^* E$. We can now use (7.5) on $E_0 \parallel E_1 \rightarrow F$ to find $F'_0, F'_1 \in CL$ such that $E_i \rightarrow^* F'_i$ for $i = 0, 1$ and $(F'_0 \parallel F'_1 = F$ or $\{F'_0, F'_1\} = \{\dag, F\})$. According to this there are two subcases:

\{F'_0, F'_1\} = \{\dag, F\}: Suppose w.l.o.g. $F'_0 = \dag$ and $F'_1 = F$. Choosing $E'_0 = \dag, E'_1 = E$ and $A_1 = \emptyset, A'_1 = A_1$ we are done because $E_0 \rightarrow^* E'_0$ implies $E_0 \parallel E'_0$ or equally $E'_0 \xrightarrow{A'_0} E'_0$ and because $E_1 \rightarrow^* F \xrightarrow{A_1} H \rightarrow^* E$ implies $E_1 \xrightarrow{A_0} E'_1$.

$n > 1$: Then there must be a $F \in CL$ and $1 \leq j < n$ such that $E_0 \parallel E_1 \xrightarrow{A_1} \ldots \xrightarrow{A_j} F \xrightarrow{A_{j+1}} \ldots \xrightarrow{A_n} E$. By induction there are $F'_0, F'_1$ and $A'_i, \ldots, A'_n$ for $i = 0, 1$ such that $E_i \xrightarrow{A'_i} \ldots \xrightarrow{A'_n} F'_1$ and $(F'_0 \parallel F'_1 = F$ or $\{F'_0, F'_1\} = \{\dag, F\})$. Two cases:

\{F'_0, F'_1\} = \{\dag, F\}: Suppose w.l.o.g. $F'_0 = \dag$ and $F'_1 = F$. Choosing $E'_0 = \dag, E'_1 = E$ and $A'_0 = \emptyset, A'_1 = A_k$ for $k = j + 1, \ldots, n$ we have $F'_0 = \dag \parallel \ldots \parallel \dag = E'_0$ or equally $F'_0 \xrightarrow{A'_0} \ldots \xrightarrow{A'_n} E'_0$ from which we get the result.

$E_0 \parallel F'_1 = F$: Then we can apply the hypothesis of induction once more and find the desired $E'_0, E'_1$ and remaining $A'_k$ for $i = 0, 1$ and $k = j + 1, \ldots, n$.

\[\square\]

**Lemma 7.4.7** Suppose $A_1, \ldots, A_n \in M_\varepsilon$ and $p_0, p_1 \neq \varepsilon$. Then $A_1 \cdot \ldots \cdot A_n \leq p_0 \cdot p_1$ implies $n \geq 2$ and there is a $1 \leq j < n$ such that

$$A_1 \cdot \ldots \cdot A_j \leq p_0 \text{ and } A_{j+1} \cdot \ldots \cdot A_n \leq p_1$$

**Proof** Let $q = A_1 \cdot \ldots, A_n$. By proposition 6.4.2 there are $q_0$ and $q_1$ such that $q = q_0 \cdot q_1$ where $q_i \leq p_i$ for $i = 0, 1$. Since $p_i \neq \varepsilon$ we also have $q_i \neq \varepsilon$. The lemma then follows from

$$A_1 \cdot \ldots \cdot A_n = p_0 \cdot p_1, p_0, p_1 \neq \varepsilon$$

(7.6)

\[n \geq 2, \exists 1 \leq j < n. A_1 \cdot \ldots \cdot A_j = p_0, A_{j+1} \cdot \ldots \cdot A_n = p_1\]

which we prove by induction on $n$: 163
\( n = 1 \): The situation is \( A_1 = p_0 \cdot p_1 \). The premise cannot hold since \( p_0, p_1 \neq \varepsilon \) implies that there are at least two ordered elements of \( p_0 \cdot p_1 \), but this contradicts \( A_1 = p_0 \cdot p_1 \) because the elements of \( A_1 \) are unordered \( (A_1 \in M_\varepsilon) \).

\( n > 1 \): Equally \( n \geq 2 \). Since \( p_0 \neq \varepsilon \) we can apply proposition 7.4.8 below to find a \( p'_0 \) such that \( A_1 \cdot p'_0 = p_0 \) and \( A_2 \cdot \ldots \cdot A_n = p'_0 \cdot p_1 \).

If \( p'_0 = \varepsilon \) we have \( p_0 = A_1 \) and \( A_2 \cdot \ldots \cdot A_n = \varepsilon \cdot p_1 = p_1 \) wherefore we can choose \( j = 1 \).

Otherwise if \( p'_0 \neq \varepsilon \) we can use the induction to find \( 2 \leq j < n \) such that \( A_2 \cdot \ldots \cdot A_j = p'_0 \) and \( A_{j+1} \cdot \ldots \cdot A_n = p_1 \). Since \( p_0 = A_1 \cdot p'_0 = A_1 \cdot A_2 \cdot \ldots \cdot A_j \) we are done. \( \square \)

**Proposition 7.4.8** For \( A \in M_\varepsilon \) and a pomset \( p_0 \neq \varepsilon \) we have

\[
A \cdot q = p_0 \cdot p_1
\]

\[ \Downarrow \]

\[ \exists p'_0 . A \cdot p'_0 = p_0, q = p'_0 \cdot p_1 \]

**Proposition 7.4.9** Suppose \( p \cdot q \leq r_0 \times r_1 \). Then there exists \( p_0, p_1, q_0, q_1 \) such that \( p_i \cdot q_i \leq r_i \) for \( i = 0, 1 \) and \( p \leq p_0 \times p_1 \) and \( q \leq q_0 \times q_1 \).

**Proof** We prove it for lpos and the proposition follows immediately.

\( p \cdot q \leq r_0 \times r_1 \) means that there exists a bijection \( f : X_{r_0 \times r_1} \rightarrow X_{p \cdot q} \) which also is a morphism of lpos.

By definition of \( \cdot \) and \( \times \) we have \( X_{p \cdot q} = \{0\} \times X_p \cup \{1\} \times X_q \) and \( X_{r_0 \times r_1} = \{0\} \times X_{r_0} \cup \{1\} \times X_{r_1} \). So define \( p_i \) as \( p \) restricted to \( \{x \in X_p \mid \langle 0, x \rangle \in f(\{i\} \times X_{r_i})\} \) for \( i = 0, 1 \) and similar for \( q_0 \) and \( q_1 \).

\( p_0 \cdot q_0 \leq r_0 \): Consider \( g : X_{r_0} \rightarrow X_{p_0 \cdot q_0} \) given by \( g(x) = f(\langle 0, x \rangle) \). It is easy to see that \( g \) is order preserving.

\( g \) order preserving: From \( f \) being order preserving and \( x \leq y \Rightarrow \langle 0, x \rangle \leq \langle 0, y \rangle \) we see \( f(\langle 0, x \rangle) \leq f(\langle 0, y \rangle) \). \( f(\langle 0, x \rangle) \) is of the form \( \langle i, x' \rangle \) and \( f(\langle 0, y \rangle) \) is of the form \( \langle j, y' \rangle \). According to the definition of \( \leq \) then

\[
\begin{align*}
(i = 0 & \Rightarrow j, x' \leq y') \text{ or } (i = 1 \Rightarrow j, x' \leq y') \text{ or } (i = 0, j = 1) \\
\end{align*}
\]

In the case \( x' \leq y' \) we have \( x' \in X_p \), so we must have \( x' \in X_{p_0} \). Also \( y' \in X_{p_0} \). Similar considerations in the case \( x' \leq y' \) leads us to:

\[
\begin{align*}
(i = 0 & \Rightarrow j, x' \leq_{p_0} y') \text{ or } (i = 1 \Rightarrow j, x' \leq_{q_0} y') \text{ or } (i = 0, j = 1) \\
\end{align*}
\]

But then \( g(x) = f(\langle 0, x \rangle) = \langle i, x' \rangle \leq_{p_0 \cdot q_0} \langle j, y' \rangle = f(\langle 0, y \rangle) = g(y) \).

\( g \) is directly seen to be label preserving: \( \ell_{r_0}(x) = \ell_{r_0 \times r_1}(\langle 0, x \rangle) = \ell_{p \cdot q}(f(\langle 0, x \rangle)) = \ell_{p_0 \cdot q_0}(g(x)) \).

\( p_1 \cdot q_1 \leq r_1 \): similar as above.

\( p \leq p_0 \times p_1 \): Let \( g : X_{p_0 \times p_1} \rightarrow X_p \) be given by \( g(\langle i, x \rangle) = x \). This time it is easy to see that \( g \) is a morphism of lpos.

\( g \) injective: Assume \( \langle i, x \rangle \neq \langle j, y \rangle \). We are done if \( x = y \) and \( i \neq j \) is impossible. So suppose on the contrary \( x = y, i \neq j \). Now \( \langle i, x \rangle \in X_{p_0 \times p_1} \Rightarrow x \in X_{p_i} \Rightarrow (0, x) \in \)
Proposition 7.4.10 If \( \text{only if} \): Suppose \( G \) and \( g \). Then \( G \) is surjective if \( G \) is surjective. Hence the direct observation sets form a lattice under the inclusion relation; the meet/glub being intersection and join/lub being union.

From \( g(i, x) = x \) the result then follows.

\[ q \leq q_0 \times q_1 \] Similar as the last case.

\[ \text{Proposition 7.4.10} \] If \( n \geq 1 \) and \( A_1, \ldots, A_n \in M_c \) and \( A_1 \cdots A_n \leq p_0 \times p_1 \) then there exists multisets \( A_1^0, \ldots, A_n^0 \) and \( A_1^1, \ldots, A_n^1 \) such that \( A_j = A_j^0 \times A_j^1 \) for \( j = 1, \ldots, n \), and \( A_1^1 \cdots A_n^1 \leq p_i \) for \( i = 0, 1 \).

\[ \text{Proof} \] By induction on \( n \).

\( n = 1 \): \( A_1 \leq p_0 \times p_1 \) clearly implies \( p_0 \) and \( p_1 \) are multisets and \( A_1 = p_0 \times p_1 \). Chose \( A_1^0 = p_0 \) and \( A_1^1 = p_1 \).

\( n > 1 \): \( A_1 \cdot (A_2 \cdot \ldots \cdot A_n) \leq p_0 \times p_1 \) implies by the previous proposition the existence of pomsets \( q_0, q_1, r_0, r_1 \) such that \( q_0 \cdot r_0 \leq p_0, q_1 \cdot r_1 \leq p_1, A_1 \leq q_0 \times q_1 \) and \( A_2 \cdot \ldots \cdot A_n \leq r_0 \times r_1 \). The last implies by hypothesis of induction that there are multisets \( A_2^0, \ldots, A_n^0 \) and \( A_2^1, \ldots, A_n^1 \) with \( A_2^0 \cdot \ldots \cdot A_n^0 \leq r_0, A_2^1 \cdot \ldots \cdot A_n^1 \leq r_1 \) and \( A_1^0 \times A_2^0 = A_i \) for \( i = 2, \ldots, n \). From the case \( n = 1 \) we see that there exists \( A_0 \) and \( A_1 \) with \( A_0 \leq q_0, A_1 \leq q_1 \) and \( A_1 = A_0 \times A_1^1 \). By monotonicity and transitivity of \( \leq \) we now get

\[ A_0^0 \cdot (A_2^0 \cdot \ldots \cdot A_n^0) \leq q_0 \cdot (A_2^0 \cdot \ldots \cdot A_n^0) \leq q_0 \cdot r_0 \leq p_0 \]

and similar for the \( A_1^1 \)’s.

7.5 Summary

In this section we show how the different G-semantics are related and some concrete examples of G-semantics are given.

At first notice that if \( G \) and \( G' \) are sets of direct observations then so are \( G \cup G' \) and \( G \cap G' \). Hence the direct observation sets form a lattice under the inclusion relation; the meet/glub being intersection and join/lub being union.

This carry over to models as follows:

\[ \text{Proposition 7.5.1} \] \( G \subseteq G' \) iff \( =_G \subseteq =_G \)

\[ \text{Proof} \]

only if: Suppose \( G \subseteq G' \). Then \( G = G' \cap G \) and \( P_G = P_{G'} \cap P_G \). By definition of \( \llbracket \cdot \rrbracket_G \) and \( \llbracket \cdot \rrbracket_{G'} \) therefore \( \llbracket \cdot \rrbracket_G = \llbracket \cdot \rrbracket_G' \cap P_G \). As a consequence \( \llbracket G' \rrbracket \subseteq \llbracket G \rrbracket \) and \( =_G' \subseteq =_G \).

165
if: We start out by some general observations for an arbitrary set of direct test $G$. If $A$ is any multiset let $E_A$ be a $BL$-expression obtained by parallel composing the atomic actions, i.e., $\varphi(E_A) = \{A\}$. From section 7.3 we know $p \neq \varepsilon$, $p \in PG$ iff $\exists n \geq 1 \exists A_1, \ldots, A_n \in G$. $A_1 \cdot \ldots \cdot A_n = p$, so for all $p \in PG \setminus \{\varepsilon\}$ there are $E_{A_1}, \ldots, E_{A_n}$ such that $\{p\} = \varphi(E_{A_1} ; \ldots ; E_{A_n})$. Composing such expressions with $\oplus$ one can then for any finite and nonempty subset, $P$, of $PG \setminus \{\varepsilon\}$ find an expression $EP$ with $\varphi(EP) = P$.

The proof of the implication is now by contradiction. Suppose $=_{G'} \subseteq =_G$ but $G \nsubseteq G'$. Then there is an $A \in G$ with $A \not\subseteq G'$. Of course then $A \in PG$ and if $P = \delta_G(A)$ we have $[EP]_G = \delta_G(P) = \delta_{G'}(A) = [EA]_{G'}$—i.e., $EP =_{G'} EA$. By assumption then also $EP =_G EA$. But clearly $A \in [EA]_G$, so we must have $A \in [EP]_G$ too. $A \in [EP]_G$ means $A \in \delta_{G'}(\delta_G(A))$ wherefore there must be a $p \in PG$ such that $A \leq p \leq A$. Hence $A = p \in PG$—a contradiction to $A \not\subseteq PG$.  

From this proposition it immediately follows that $G \subset G'$ implies $\llbracket A \rrbracket_G$ is strictly more abstract than $\llbracket A \rrbracket_{G'}$ (i.e., all expressions identified by $\llbracket A \rrbracket_G$ are identified by $\llbracket A \rrbracket_{G'}$ and there is some expressions identified by $\llbracket A \rrbracket_{G'}$ but not by $\llbracket A \rrbracket_G$).

In fact the lattice of our $G$-models has a least and a greatest model (in the sense of their ability to distinguish expressions). The least model is of course the one generated from $G = \Delta$ and the largest the one generated from $G = M$. It is not hard to see that $P_\Delta$ agrees with $W$—the set of pomsets which are words (see page 133). As a consequence hereof we shall in the following (chapter) subscript with $w$ rather than $\Delta$ when concerning the minimal model/ the operational semantics obtained by the least set of direct tests.

The variation of the operational semantics arising from the different sets of direct test manifest itself in the inference rule for a composed step: $\frac{A_0 \times A_1}{A_0 \cdot A_1} G$. For $W$ the inference rule totally vanish and for $M$ it becomes

$$
\frac{E_0 \overset{A_0}{\longrightarrow}_M E_0', E_1 \overset{A_1}{\longrightarrow}_M E_1'}
{E_0 \parallel E_1 \overset{A_0 \parallel A_1}{\longrightarrow}_M E_0' \parallel E_1'}
$$

I.e., no restrictions.

As examples of what models we might find in between the least and largest models we end this section by giving two almost contrasting models.

Starke [Sta81] has introduced one natural candidate for semiwords, i.e., to some extend half a word, half a (unordered) pomset. He defines a semiword to be a pomset, $p$, where all equally labelled elements are ordered: $P_{sw}(p)$ iff for all $x, y$ in $X_p$ we have:

$$
x \text{ if } co_p \text{ then } \ell_p(x) \\
y \text{ } \ell_p(y)
$$

One might just as well take the opposite standpoint and define a semiwords to be a pomset, $p$, where all unequally labelled elements are ordered: $P_{sw}^{-}(p)$ iff for all $x, y$ in $X_p$ we have:
\[ \text{if } x \preceq y \text{ then } \ell_p(x) = \ell_p(y) \]

Notice that both properties are hereditary and dot synthesizable. However the candidate of Starke enjoys a number of nice properties: If \( P_{sw}(p) \) then there is a canonic representative, \( \hat{p} \in LPO \), of \( p \) (in the sense that \( p = [\hat{p}] \) and if \( p = q \) then \( \hat{p} = \hat{q} \)) and furthermore the partial order, \( \preceq \), on such semiwords may be characterized by \( p \preceq q \iff \hat{p} \preceq \hat{q} \iff \delta_w(p) \subseteq \delta_w(q) \). We shall later in the next chapter see some consequences of a pomset having the \( P_{sw} \)-property.

On second thoughts one soon realizes that

\[ P_{sw, and} = P_S \text{ and } P_{sw, and} = P_N \]

So the most general linearizations of the semiwords of Starke corresponds to sequences of sets (of \( \Delta \)) whereas the most general linearisations of the other type of semiwords corresponds to sequences of multisingletons (of \( \Delta \)).

The inference rules for the composed step for these two sets of direct tests are particular simple:

\[
\frac{E_0 \xrightarrow{A_0} S E_0', E_1 \xrightarrow{A_1} S E_1', A_0 \cup A_1 \subseteq \Delta}{E_0 \parallel E_1 \xrightarrow{A_0 \cup A_1} S E_0' \parallel E_1'}
\]

and

\[
\frac{E_0 \xrightarrow{a^n \in N} E_0', E_1 \xrightarrow{a^n \in N} E_1'}{E_0 \parallel E_1 \xrightarrow{a^n \in N} E_0' \parallel E_1'}
\]

### 7.6 An Adequate Logic

In this section we shall for each \( G \)-semantics of \( BL \) give an adequate logic \( L_G \). A logic for our process language will be a set of (logic) formulae together with a satisfaction relation which for each process and formula tells whether the process satisfies the formula. In the sense of Hennessy and Milner [HM80] such a logic is adequate for a \( G \)-semantics iff processes are identified by the \( G \)-semantics (by the equivalence, \( \preceq_G \), of \( \preceq_G \)) exactly when they satisfy the same set of formulae in the logic (\( L_G \)).

The branching aspect is on purpose left out of account and brought about partly by having only a combinator for internal nondeterminism and partly by constructing the operational preorders on the basis of sequences of direct tests. Pnueli [Pnu85] regards the latter as taking the linear view and shows how a linear time logic can be appropriate in this situation. In agreement with this we will define a process to satisfy a formula if all the “syntactic controlled” behaviours of the process satisfies the formula. We shall in a moment make these notions precise.

The different logics will share the same set of logic formulae, \( L_p \), but have individual satisfaction relation \( |=_G \)—one for each logic \( L_G \). Mainly for proof technical reasons we
shall define $\mathcal{L}_g$ as a subset of a larger formula language, $\mathcal{L}$, and base $\models_G$ on a larger satisfaction relation for $\mathcal{L}$.

The set of formulae, $\mathcal{L}$, is defined in the BNF-like way:

$$f ::= \text{tt} \mid ff \mid \triangledown \mid \bigtriangleup \mid \lozenge f, A \in M \mid \Box f, A \in M$$

and $\mathcal{L}_g \subseteq \mathcal{L}$ is taken to be those formulae with no occurrence of the modality $\lozenge$.

Similarly as for Hennessy-Milner logic [HM85] we for each $G$-semantic define a satisfaction relation, $\models_G$, between behaviours from $DCL$ (see page 153) and formulae of $\mathcal{L}$ using the definitions of the operational $G$-semantics. The modalities $\lozenge$ and $\bigtriangleup$ can be considered as generalizations of the corresponding Hennessy-Milner modalities (with $A = \{a\}$). $\text{tt}$ ($ff$) has the standard interpretation that it always (never) is satisfied by a process. A process satisfies $\triangledown$ if it is terminated (i.e., no external computation step is possible) whereas $\bigtriangleup$ indicates that the process is alive. Formally:

**Definition 7.6.1** $\models_G \subseteq DCL \times \mathcal{L}$ is defined inductively:

1. $E \models_G \text{tt}$ for all $E \in DCL$
2. $E \models_G \triangledown$ iff $\forall a \in A. E \not\models_G$
3. $E \models_G \bigtriangleup$ iff $\exists a \in A. E \not\models_G$
4. $E \models_G \lozenge f$ iff $\exists E'. E \models_G E'$ and $E' \models_G f$
5. $E \models_G \Box f$ iff $\forall E'. E \models_G E'$ implies $E' \models_G f$

where $E \not\models_G$ means $\exists E' \in CL. E \not\models_G E'$.

Following the linear logic tradition we now for each logic, $\mathcal{L}_G$, say that a process $E \in BL$ satisfies a formula $f \in \mathcal{L}_g$,

$$E \models_G f \text{ iff } \forall E' \in \text{Beh}(E). E' \models_G f$$

The set of formulae from $\mathcal{L}_g$ which is satisfied in a logic $\mathcal{L}_G$ by an $E \in BL$ will be denoted $\mathcal{L}_G(E)$. I.e.,

$$\mathcal{L}_G(E) = \{ f \in \mathcal{L}_g \mid E \models_G f \}$$

It will facilitate the proof of the the adequacy of the different $\mathcal{L}_G$ logics to introduce some additional notions.

At first we give a syntactic map $\underline{\cdot} : \mathcal{L} \rightarrow \mathcal{L}$ which yield the “dual” of a formula. For each $f \in \mathcal{L}$ define $\overline{f}$ by induction on the structure of $f$:

$$
\begin{align*}
\overline{\text{tt}} &= \text{tt} \quad & \overline{f} &= ff \\
\overline{\bigtriangleup} &= \triangledown \quad & \overline{\triangledown} &= \bigtriangleup \\
\overline{\lozenge f} &= \Box \overline{f} \quad & \overline{\Box f} &= \lozenge \overline{f}
\end{align*}
$$

168
Clearly $\overline{f} = f$ and an easy induction on the structure of $f$ shows that $f$ and $\overline{f}$ are dual (for every satisfaction relation $|=G$) in the sense that

\[(7.7) \quad E \not|=G f \iff E |=G \overline{f} \]

for all configuration behaviours $E \in DCL$. Now define

$$\mathcal{L}_g = \{ f \in \mathcal{L} \mid f \in \mathcal{L}_g \}$$

and for every $E \in BL$

$$\mathcal{L}_G(E) = \{ f \in \mathcal{L}_g \mid \exists E' \in \text{Beh}(E). E' |=G f \}$$

Notice that $\mathcal{L}_g$ are the formulae of $\mathcal{L}$ whit no occurrence of the modality $A$.

The following lemma display the close relationship between $\mathcal{L}_G(\_)$ and $\mathcal{L}_G(\_)$.

**Lemma 7.6.2** For all $E_0, E_1 \in BL$ we have:

$$\mathcal{L}_G(E_0) \subseteq \mathcal{L}_G(E_1) \iff \mathcal{L}_G(E_0) \supseteq \mathcal{L}_G(E_1)$$

**Proof** We start out by inferring for an arbitrary formula $f \in \mathcal{L}_g$ and process $E \in BL$:

\[
\begin{align*}
& f \notin \mathcal{L}_G(E) \iff E \not|=G f \\
& \quad \iff \exists E' \in \text{Beh}(E). E' \not|=G f \quad \text{definition of } \mathcal{L}_G(\_) \\
& \quad \iff \exists E' \in \text{Beh}(E). E' |=G \overline{f} \quad \text{by (7.7)} \\
& \quad \iff \overline{f} \in \mathcal{L}_G(E) \quad \text{definition of } \mathcal{L}_G(\_)
\end{align*}
\]

Using the lemma below the proof is now merely logic rewriting:

\[
\begin{align*}
\mathcal{L}_G(E_0) & \subseteq \mathcal{L}_G(E_1) \\
\implies & \forall f \in \mathcal{L}_g. f \in \mathcal{L}_G(E_0) \Rightarrow f \in \mathcal{L}_G(E_1) \\
\implies & \forall f \in \mathcal{L}_g. \overline{f} \in \mathcal{L}_G(E_0) \Rightarrow \overline{f} \in \mathcal{L}_G(E_1) \quad \square \\
\implies & \forall f \in \mathcal{L}_g. f \notin \mathcal{L}_G(E_0) \Rightarrow f \notin \mathcal{L}_G(E_1) \\
\implies & \mathcal{L}_G(E_0) \supseteq \mathcal{L}_G(E_1)
\end{align*}
\]

The adequacy of the different $\mathcal{L}_G$ logics can now be seen from:

**Theorem 7.6.3** (Linear Logic Characterization)

For all $E_0, E_1 \in BL$:

$$E_0 \leq_G E_1 \iff \mathcal{L}_G(E_0) \supseteq \mathcal{L}_G(E_1)$$

**Proof** Immediate from the preceding lemma and the following. \qed
Lemma 7.6.4 Suppose $E_0, E_1 \in BL$. Then

$$E_0 \preceq_G E_1 \iff \overline{\mathcal{Z}}_G(E_0) \subseteq \overline{\mathcal{Z}}_G(E_1)$$

Proof From the definition of $|=_G$ it is almost trivial to prove for $F \in DCL$, $f \in \overline{\mathcal{Z}}_g$ and $n \geq 1$ that

(7.8) $F |=_G \Box \ldots \Box f \iff \exists F', F \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G F', F' |=_G f$

(7.9) $F |=_G \nabla \iff F \rightarrow^* \uparrow$

by induction on $n$ in the case of (7.8) and induction on the structure of $E$ in case of (7.9).

The if part of the lemma now follows the definition of $\preceq_G$ and by deducing for $E \in BL$ and $n \geq 1$

(7.10) $E \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G \uparrow \iff \Box \ldots \Box \nabla \in \overline{\mathcal{Z}}_G(E)$

as follows: $E \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G \uparrow$

$\iff \exists F \in \text{Beh}(E). F \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G \uparrow$ \hspace{1cm} \text{proposition 7.2.5}

$\iff \exists F', F \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G F', F' \rightarrow^* \uparrow$ \hspace{1cm} \text{definition of $\xrightarrow{A_{i_n}}_G$}

$\iff \exists F \in \text{Beh}(E). F |=_G \Box \ldots \Box \nabla$ \hspace{1cm} \text{by (7.8) and (7.9)}

$\iff \Box \ldots \Box \nabla \in \overline{\mathcal{Z}}_G(E)$ \hspace{1cm} \text{definition of $\overline{\mathcal{Z}}_G(E)$}

For the only if direction let an $f \in \overline{\mathcal{Z}}_G(E_0)$ be given. We consider each possible appearance of $f$ in turn.

At first notice that $F \in \text{Beh}(E)$ and $E \in BL$ implies $F \in DBL$, and that any $F \in DBL$ is capable of doing at least one action. So because $E_0 \in BL$ it follows that $f = \nabla$ is impossible and if $f = \Box$ we also have $f \in \overline{\mathcal{Z}}_G(E_1)$ since $E_1 \in BL$.

If $f$ is satisfied by no behaviour wherefore it should be clear that $f$ cannot belong to $\overline{\mathcal{Z}}_G(E_1)$ because $E_1 \in BL$.

If $f = \Box \ldots \Box \uparrow$ then evidently $f \in \overline{\mathcal{Z}}_G(E_1)$ and if $f$ is of the form $\Box \ldots \Box \nabla$ the result follows from (7.10) and the definition of $\preceq_G$.

Now suppose $f$ is of the form $\Box \ldots \Box \Box$ for some $A_1, \ldots, A_i$ and $i \geq 1$. This means there is a $F \in \text{Beh}(E_0)$ such that $F |=_G \Box \ldots \Box \Box$ and from (7.8) we conclude $F \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G F'$ for some $F' \in DCL$. Using proposition 7.2.6 we get $E_0 \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G E_0'$ for some $E_0'$ (with $E_0' \in \text{Beh}(E_0')$). Since we only have finite processes in $BL$ and therefore also finite configurations there must be some $A_{i_1+1}, \ldots, A_{i_n} \in G$ such that $E_0' \xrightarrow{A_{i_1+1}}_G \ldots \xrightarrow{A_{i_n}}_G \uparrow$.

The premise $E_0 \preceq_G E_1$ then gives $E_1 \xrightarrow{A_{i_1}}_G \ldots \xrightarrow{A_{i_n}}_G \uparrow$ and by (7.10) thus $\Box \ldots \Box \ldots \Box \Box \nabla \in \overline{\mathcal{Z}}_G(E_1)$. A simple induction on $i$ then shows that this implies $\Box \ldots \Box \Box \Box \uparrow \in \overline{\mathcal{Z}}_G(E_1)$.

We are left with the case where $f$ is of the form $\Box \ldots \Box \Box$. Using (7.8) it is easy to see
by looking at the definition of $F' \models G \triangleright$ that

$$\bigotimes \cdots \bigotimes \triangleright \in \mathcal{T}_G(E) \iff \exists a \in \Delta. \bigotimes \cdots \bigotimes \text{tt} \in \mathcal{T}_G(E)$$

for $E \in BL$ and $i \geq 1$. Hence this case is reduced to the one we just have dealt with. □

From the proof it is evident that the lemma still would hold if $\mathcal{L}_g$ only had formulae of the form $\bigotimes \cdots \bigotimes \triangleright$, and consequently a logic with formulae of $\mathcal{L}$ which only contained $\triangleright$ and the modality $\bigcirc$ would be sufficient to obtain the linear logic characterization. So why not be be content with this smaller formula language? Pnueli [Pnu85] argues that one advantage of logic is the ability to deal with partial specifications. Clearly there is more freedom to give partial specifications in the larger formula language. With little extra effort we could even include disjunction in $\mathcal{L}_g$ without affecting the lemma. (7.7) would also hold if we added conjunction to $\mathcal{L}_g$ and would therefore obtain the characterization for this extended logic too.

Let us end this section by making the note that we easily could have obtained an alternative logic characterization of $\models_G$ by choosing as formula language $\mathcal{L}_g$ and for each $G$ take as satisfaction relation $\models' \subseteq CL \times \mathcal{L}_g$ with definition as 7.6.1 (on $\mathcal{L}_g$) but with expressions from $CL$ and not just $DCL$. We could then for $E \in BL$ let $\mathcal{L}_G(E)$ be $\{ f \in \mathcal{L}_g \mid E \models'_G f \}$. From proposition 7.2.6 and the form of the formulae of $\mathcal{L}_g$ it should be clear that lemma 7.6.4 still would hold for the changed set-up and could therefore serve as logic characterization. The reason why we have chosen to give the linear logic characterization is twofold. In the first place we want to show that a linear view (perhaps not surprisingly) is sufficient to capture the $G$-semantics. Secondly it prepares for a later logic characterization which we could not make so easily in the changed set-up.
Chapter 8

RBL—A Basic Process Language with Refinement

It is well-known ([BC87, vGV87, Hen87b]) that a distinction between concurrency and interleaving may be captured by adding a combinator to the process language, changing the atomicity of actions.

To give a simple concrete example assume the processes $E = \text{Topneg} \ ; \ \text{Topneg}$ and $F = \text{Topneg} \ || \ \text{Topneg}$ when run accesses a nonempty stack of logical values. With \text{Topneg} having the obvious effect on the stack $E$ and $F$ will run without problems leaving the stack as it was. If however \text{Topneg} is refined to $\text{Pop} ; \text{Pushneg}$ (again with the effect as suggested by the name) in $E$ and $F$ getting $E'$ and $F'$ respectively the things change. There will be no difference between $E$ and $E'$, but when $F'$ is run the value of the top element may have changed and in the case where the stack consists of one element stack underflow may occur.

We look at the different semantics for $BL$ introduced in the previous chapter 7 and investigate the consequences of adding a combinator allowing an expansion of an individual action into a process.

Formally define a $BL$-refinement to be a mapping $\varphi : \Delta \to BL$.

For each $BL$-refinement $\varphi$ we introduce a combinator, $[\varphi]$, into our language, with the operational meaning that $E[\varphi]$ behaves operationally just like $E$ with all $a$-occurrences substituted by $\varphi(a)$. We denote this extended language by $RBL$. The combinator precedence will be the same as for $BL$ except that $[\varphi]$ binds stronger than the binary combinators.

In spite of we have not given a more explicit formulation of substitution yet, we shall look at an example which illustrates not only the idea of substitution but also a consequence for the preorders.

Example: Let $\varphi(a) = a \ ; \ b$, $\varphi(b) = b$, and $E_0 = a \ || \ b$, $E_1 = a \ ; \ b \oplus b \ ; a$. The “substituted” expressions $F_0$ and $F_0$ of $E_0[\varphi]\sigma$ and $E_1[\varphi]\sigma$ respectively will then be

$$F_0 = a \ ; \ a \ || \ b \ \text{and} \ F_1 = a \ ; \ a \ ; b \oplus b \ ; a \ ; a$$

Clearly $E_0 \preceq_w E_1$ but $F_0 \nrightarrow_w \uparrow$ and $F_1 \nrightarrow_w$. Hence $\preceq_w$ will not be a precongruence for $RBL$!
Though the example illustrates that \( \preceq_w \) will not be a precongruence for \( RBL \) we cannot use it to conclude the same for \( \preceq_G \) in general since many of the \( G \)-semantics would distinguish \( E_0 \) and \( E_1 \), e.g., \( E_0 \not\sim_S E_1 \).

Our question is here: What is the precongruence associated with \( \preceq_G \) for \( RBL \), \( \preceq_w \)? In the next section we give the different operational \( G \)-semantics and derive some results. We then pursue the question for \( \preceq_w \) in the succeeding section through different considerations, gradually arriving at a model fully abstract with respect to \( \preceq_w \). From the model considerations it then turns out that \( \preceq_w \) equals \( \preceq_c^n \) for every \( G \)-semantics.

8.1 Operational Set-up

We shall give different operational semantics for \( RBL \) similar as in the chapter with \( BL \). In fact the extended labelled transition system will be the same except that the configuration language \( RCL \) now is the least set \( C \) satisfying:

\[
\begin{align*}
\uparrow \in C \\
RBL \subseteq C \\
E_0 ; E_1 \in C & \text{ if } E_0 \in C \text{ and } E_1 \in RBL \\
E_0 \parallel E_1 \in C & \text{ if } E_0, E_1 \in C
\end{align*}
\]

and the definition of \( \rightarrow \) is augmented in order to cope with \( [\varrho] \):

\[
\begin{align*}
\displaystyle{a[\varrho] \rightarrow \varrho(a)} & \\
\displaystyle{(E_0 ; E_1)[\varrho] \rightarrow E_0[\varrho] ; E_1[\varrho]} & \frac{E \rightarrow E'}{E[\varrho] \rightarrow E'[\varrho]} \\
\displaystyle{(E_0 \oplus E_1)[\varrho] \rightarrow E_0[\varrho] \oplus E_1[\varrho]} & \\
\displaystyle{(E_0 \parallel E_1)[\varrho] \rightarrow E_0[\varrho] \parallel E_1[\varrho]}
\end{align*}
\]

Notice that there is no rule to deal with a case like \( E[\varrho'][\varrho] \). This is not necessary because the \( [\varrho] \)-inference rule allows “substitution” of \( [\varrho'] \) in \( E \) by internal steps before starting with \( [\varrho] \).

**Example:** Suppose \( \varrho'(b) = c ; d \) and \( \varrho(c) = e \). Then

\[
\begin{align*}
(a \parallel b)[\varrho'][\varrho] & \rightarrow (a[\varrho'] \parallel b[\varrho'])[\varrho] \\
& \rightarrow (a[\varrho'] \parallel c ; d)[\varrho] \\
& \rightarrow a[\varrho'][\varrho] \parallel (c ; d)[\varrho] \\
& \rightarrow a[\varrho'][\varrho] \parallel c[\varrho] ; d[\varrho] \\
& \rightarrow a[\varrho'][\varrho] \parallel e ; d[\varrho] \rightarrow_G \ldots
\end{align*}
\]
Notice that the “substitution” not necessarily has to follow a unique route. E.g., above it is also possible with: \(a[\ell]\|b[\ell]'[\ell]\|c; d[\ell]'[\ell]\|c')\rightarrow a[\ell]'[\ell]\|b'[\ell]'[\ell]\|c; d[\ell]'[\ell]\|c')\rightarrow \ldots\)

The definitions of \(\frac{A}{G}\rightarrow G, \frac{S}{G}\rightarrow G\) and \(\leq G\) are generalized to \(RBL\) in the obvious way. We keep the convention to leave out the subscript \(G\) in \(\rightarrow G\) and \(\Rightarrow G\) except for certain \(G\)’s.

Proposition 7.2.3 extends smoothly to \(RBL\) with one addition:

**Proposition 8.1.1** Suppose \(E \in RBL, E \rightarrow^* E'\) and \(E_0, E_1 \in RCL, E_0 \Rightarrow E'_0\). Then

- \(E_0; E \Rightarrow E'_0; E\)
- \(E_0 \| E_1 \Rightarrow E'_0 \| E_1\)
- \(E_1 \| E_0 \Rightarrow E'_1 \| E'_0\)
- \(E\ell \Rightarrow^* E'[\ell]\)

We will now make it more precise what we mean by substitution. The substitution is “performed” by a compositionally defined mapping \(\sum: RCL \rightarrow CL\), using \(\{g\} : BL \rightarrow BL\) which (also compositionally) performs a single substitution in a refinement free expression. Because of their syntactic nature we write them postfix. The definitions of \(\sum\) and \(\{g\}\) are in full:

\[\begin{align*}
\sum = \sum
\ &a \sum = a \\
\ &a\ell = a(\ell) \\
\ &\{g\} = \{g\} \\
\ &\{g\} = \{g\} + \{g\} \\
\ &\{g\} \| \{g\} = \{g\} \| \{g\} \\
\ &\{g\} \| \{g\} = \{g\} \| \{g\} \\
\ &\{g\} \{g\} = \{g\}\{g\} \\
\&\{g\} = \{g\} \\
\end{align*}\]

Notice that \(\sum\) when restricted to \(RBL\) yield a map \(\sum: RBL \rightarrow BL\). Because configurations only contains expressions like \(E\ell\) when \(E \in RBL\) we then do not need a case for \(\{g\}\) similar to \(\sum = \sum\) and we conclude that the definitions are well-defined.

**Example:** Suppose \(\ell'([a] = b; c, \ell(b) = a; b\) and otherwise \(\ell'(e) = \ell(e) = e\). Then

\[(a \| b)[\ell'][\ell]\sum = (b \| b)[\ell]\{g\} = a; b \| a\; b.\]

The rest of this section is devoted the proof of the following proposition which essentially states: \(E \in RBL\) behaves operationally as if the refinements were substituted in advance:

**Proposition 8.1.2** Suppose \(E \in RBL\). Then for \(s \in G^*:\)

\[E \Rightarrow^* \sum \iff E\sum \Rightarrow^* \sum\]

174
Proof The proposition follow directly from:

\( E \in RCL, E \not\Rightarrow E' \)

(8.1) \[ E_1 \underoverset{\not\Rightarrow}{\sigma} E_2 \]

and

\( E \in RCL, E \not\Rightarrow E' \)

(8.2) \[ \exists E'' \in RCL, E \not\Rightarrow E'', E'' \sigma = E' \]

which both are proven by induction on the length of \( \not\Rightarrow \) using the following two lemmas in the inductive step of (8.1) and lemma 8.1.6 in the inductive step of (8.2).

\[ \square \]

Lemma 8.1.3 For \( E \in RCL \) we have: \( E \rightarrow E' \) implies \( E \sigma \rightarrow^{\ast} E' \sigma \)

Proof Induction on the structure of \( E \).

\( E = \top \) or \( E = a \): Then \( E \nrightarrow \) and the implication holds vacuously.

\( E = E_0 \cup E_1 \): From the definition of \( \nrightarrow \) we see that there are two cases:

- \( E_0 = \top \) and \( E' = E_1 \): We have \( (\top ; E_1) \sigma = \top ; E_1 \sigma \rightarrow E_1 \sigma = E' \).
- \( E_0 \rightarrow E'_0 \) and \( E' = E'_0 \); \( E_1 \): By induction \( E_0 \sigma \rightarrow^{\ast} E'_0 \sigma \), so from proposition 8.1.1 then \( E \sigma = E_0 \sigma ; E_1 \sigma \rightarrow^{\ast} E'_0 \sigma ; E_1 \sigma = E' \).

\( E = E_0 \cup E_1 \): W.l.o.g. assume \( E \rightarrow E_0 = E' \). Clearly \( E \sigma = E_0 \sigma \cup E_1 \sigma \rightarrow E_0 \sigma = E \sigma \).

\( E = E_0 \| E_1 \): Similar and symmetric to the case \( E = E_0 \cup E_1 \).

\( E = F[\sigma] \): In each case when the internal step derives from an axiom one easily from the definition of \( \sigma \) and \( \{\sigma \} \) show \( F[\sigma] \sigma = E' \sigma \). Since \( E' \sigma \rightarrow^{\ast} E' \sigma \) the result then follows. It remains to look at the case where the internal step derives from the inference rule. Here we have \( F \rightarrow F' \) and \( E' = F'[\sigma] \). By hypothesis of induction \( F \sigma \rightarrow^{\ast} F' \sigma \). Since \( F \sigma \in BL \) we can use (8.3) below to get \( (F[\sigma]) \sigma = F\sigma\{\sigma\} \rightarrow^{\ast} F'\sigma\{\sigma\} = F'[\sigma] \sigma = E' \sigma \) as desired.

We used

(8.3) \[ n \geq 0, E \in BL, E \rightarrow^{n} E' \text{ implies } E\{\sigma\} \rightarrow^{\ast} E'\{\sigma\} \]

which is proved by induction on \( n \) and in the inductive step one prove (8.3) for \( n = 1 \) by induction on the structure of \( E \). The arguments are identical to the ones used above except that the things get easier because \( E \in BL \) and we therefore do not have to deal with \( \sigma \) and the cases \( E = \top \) and \( E = F[\sigma] \).

\[ \square \]

Lemma 8.1.4 For \( E \in RCL \) we have: \( E \rightarrow E' \) implies \( E \sigma \rightarrow A E' \)

Proof By induction on the structure of \( E \).

\( E = \top \): Trivially true.

\( E = a \): The only possibility is \( A = a \) and \( E' = \top \). We have \( a \sigma = a \rightarrow^{\top} \top = \top \sigma \).

175
Lemma 8.1.6. If \( E = E_0; E_1 \): According to the definition of \( \overset{A}{\rightarrow} \) we can only have \( E_0; E_1 \overset{A}{\rightarrow} E' \) if 
\( E_0 \overset{A}{\rightarrow} E_0' \) and \( E' = E_0'; E_1 \). By hypothesis of induction \( E_0\sigma \overset{A}{\rightarrow} E_0'\sigma \). Using the inference rule for \( \); we get \( E\sigma = E_0\sigma ; E_1\sigma \overset{A}{\rightarrow} E_0'\sigma ; E_1\sigma = E'\sigma \).

\( E = E_0 \oplus E_1 \overset{A}{\rightarrow} \) is not defined for expressions of this form so the implication holds trivially.

\( E = E_0 \parallel E_1 \): There are three potential ways the step could have been produced. If only one part, \( E_0 \) or \( E_1 \), is involved the argument follow the case \( E = E_0 \parallel E_1 \). Otherwise we have \( E' = E_0' \parallel E_1' \) where \( A = A_0 \times A_1 \in G \) and \( E_i \overset{A_i}{\rightarrow} E_i' \) for \( i = 0, 1 \). By hypothesis of induction we then get \( E\sigma = E_0\sigma \parallel E_1\sigma \overset{A}{\rightarrow} E_0'\sigma \parallel E_1'\sigma = E'\sigma \).

\( E = F[\varrho] \): There are no axioms or inference rules for \( \overset{A}{\rightarrow} \) when \( E \) is of this form.

\( \square \)

In the statement and proofs of the lemma to follow we shall make extensive use of some special subsets \( RBL \) and \( RCL \) of the process expressions and the configuration expressions respectively. The idea is that \( E \in RBL \subseteq RCL \) if no internal step can bring any refinement combinator of \( E \) “inwards” in \( E \). Similar if \( E \in RCL \). Looking at the rules for internal steps dealing with the refinement combinator one soon realize that the refinement then only can appear in the scope of a \( \oplus \)-combinator or the right hand side of a \( \parallel \)-combinator. This leads to the following inductive definition.

**Definition 8.1.5** \( RCL \) is the least subset \( C \) of \( RCL \) which satisfies:

\[
\begin{align*}
CL & \subseteq C \\
E_0 ; E_1 & \in C \quad \text{if } E_0 \in C \text{ and } E_1 \in RBL \\
E_0 \oplus E_1 & \in C \quad \text{if } E_0, E_1 \in RBL \\
E_0 \parallel E_1 & \in C \quad \text{if } E_0, E_1 \in C
\end{align*}
\]

\( RBL \) is \( RBL \cap RCL \), i.e., \( RBL \) is the process expressions of \( RCL \) or equally those expressions of \( RCL \) that contains no \( \uparrow \).

**Example:** \( a[\varrho] \oplus (b ; c[\varrho]) \in RCL \) but \( a[\varrho] \parallel (b ; c)[\varrho] \notin RCL \) because \( a[\varrho] \overset{A}{\rightarrow} \varrho(a) \) and \( [\varrho] \) can be moved in over \( b ; c \).

**Lemma 8.1.6** If \( E \in RCL \) then

a) \( E\sigma \rightarrow E' \) implies \( \exists E'' \in RCL. E \rightarrow E'', E''\sigma = E' \)

b) \( E\sigma \overset{A}{\rightarrow} E' \) implies \( \exists E'' \in RCL. E \overset{A}{\rightarrow} E'', E''\sigma = E' \).

**Proof**

a) Using lemma 8.1.7 we find a \( F \in RCL \) fulfilling \( E \rightarrow F \) and \( F\sigma = E\sigma \). So \( F\sigma \rightarrow E' \). Since \( F \in RCL \) we can use lemma 8.1.9 to find a \( E'' \) with \( F \rightarrow E'' \) and \( E''\sigma = E' \). Together we now have \( E \rightarrow F \rightarrow E'' \) and \( E''\sigma = E' \).
b) Given $E\sigma \xrightarrow{A} E'$. As in a) we find a $F \in \overline{RCL}$ such that $E \rightarrow^* F$ and $F\sigma \xrightarrow{A} E'$. Because $F \in \overline{RCL}$ lemma 8.1.10 then yields $F \xrightarrow{A} E''$ for a $E''$ with $E''\sigma = E'$. Collecting the facts we have $E \rightarrow^* F \xrightarrow{A} E''$—i.e. $E \Rightarrow E''$ and $E''\sigma = E'$.

\[ \square \]

The next lemma states that the refinement combinators can be brought entirely “inwards” by internal steps.

**Lemma 8.1.7** Given $E \in RCL(RBL)$ there exists a $E' \in \overline{RCL}(RBL)$ such that $E \rightarrow^* E'$ and $E\sigma = E\sigma$.

**Example:** $E = a[\alpha] \parallel (b ; c)[\alpha] \rightarrow \varphi(a) \parallel (b ; c)[\alpha] \rightarrow^* \varphi(a) \parallel (\varphi(b) ; c[\alpha]) = E' \in \overline{RCL}$ and $E\sigma = \varphi(a) \parallel (\varphi(b) ; (c[\alpha]\sigma)) = E'\sigma$.

**Proof** By induction on the structure of $E$.

$E = \dagger$: Then $E \in RCL$. But also $E' := E = \dagger \in CL \subseteq \overline{RCL}$.

$E = a$: Just choose $E' = a \in BL \subseteq \overline{RBL} \subseteq \overline{RCL}$.

$E = E_0 ; E_1$: Here we must have $E_0 \in RCL(RBL)$ and $E_1 \in RBL$. By hypothesis of induction there is an $E_0' \in RCL(RBL)$ such that $E_0 \rightarrow^* E_0'$ and $E_0'\sigma = E_0\sigma$.

Let $E' = E_0' ; E_1$. From proposition 8.1.1 then $E_0 ; E_1 \rightarrow E'$ and of course $E' \in \overline{RCL}(RBL)$. Also $E'\sigma = E_0'\sigma ; E_1\sigma = E_0\sigma ; E_1\sigma = E\sigma$.

$E = E_0 \parallel E_1$: Clearly we can choose $E' = E$ here.

$E = F[a]$: $E$ can only be of this form when $F \in RBL$. By hypothesis of induction $F \rightarrow^* F'$ for a $F' \in RBL$ with $F'\sigma = F\sigma$. Since $F' \in RBL$ we can use the following lemma to find a $E' \in RBL$ such that $F'[a] \rightarrow^* E'$ and $F'[a]\sigma = E'\sigma$.

From proposition 8.1.1 then $E = F[a] \rightarrow^* F'[a] \rightarrow^* E'$. We also have $E'\sigma = F'[a]\sigma = (F'\sigma)\{a\} = (F\sigma)\{a\} = F[a]\sigma = E\sigma$ as desired.

\[ \square \]

We need a measure, $h$, for the inductive proof of the next lemma. Intuitively $h$ measure the number of internal steps necessary to move a refinement combinator $[\alpha]$ from outside “entirely inwards” in an expression $E \in \overline{RBL}$—i.e., if $h(E) = n$ then there is a $E'$ such that $E[\alpha] \rightarrow^* E' \in \overline{RBL}$. $h(E) = 3$ in the example above. Formally $h: \overline{RBL} \rightarrow \mathbb{N}^+$ is given by:

\[
\begin{align*}
h(a) &= 1 \\
h(E_0 ; E_1) &= 1 + h(E_0) \\
h(E_0 \otimes E_1) &= 1 \\
h(E_0 \parallel E_1) &= h(E_0) + h(E_1)
\end{align*}
\]

Notice that we do not have to define $h$ for expressions of the form $E[\alpha]$ because they cannot belong to $\overline{RBL}$. 

177
Lemma 8.1.8 If \( E \in RBL \) then there is an \( E' \in RBL \) such that
\[
E[\rho] \rightsquigarrow^* E' \text{ and } E'\sigma = E[\rho]\sigma
\]

Proof By induction on \( h(E) \).

\( h(E) = 1 \): There are two cases:

\( E = a \): Then \( E[\rho] \rightsquigarrow \rho(a) \). Choose \( E' = \rho(a) \). Since \( E' \in BL \subseteq RBL \) by definition of \( BL \)-refinements we have \( E'\sigma = E' = \rho(a) = a\{\rho\} = E\sigma \).

\( E = E_0 \uplus E_1 \): Then \( E[\rho] \rightsquigarrow E_0[\rho] \uplus E_1[\rho] =: E' \in RBL \) and the result follows from the compositional nature of \( \sigma \) and \( \{\rho\} \).

\( h(E) > 1 \): Again there are two cases:

\( E = E_0 \uplus E_1 \): Then \( h(E_0) < h(E) \) and of course \( E_0 \in RBL \), so we can use the hypothesis of induction to find an \( E'_0 \) such that \( E'_0[\rho] \rightsquigarrow^* E'_0 \) and \( E'_0\sigma = E_0[\rho]\sigma \). Choosing \( E' = E'_0 \uplus E_1[\rho] \) we get from proposition 8.1.1 \((E_0; E_1)[\rho] \rightsquigarrow E_0[\rho]; E_1[\rho] \rightsquigarrow^* E' \) and \( E'\sigma = E'_0\sigma; E_1[\rho]\sigma = E_0[\rho]\sigma; E_1[\rho]\sigma = (E_0\sigma; E_1\sigma)\{\rho\} = (E_0; E_1)\sigma\{\rho\} = E[\rho]\sigma \).

\( E = E_0 \parallel E_1 \): Here both \( h(E_0) \) and \( h(E_1) \) are less than \( h(E) \) so we can apply the hypothesis of induction on both and obtain the result with similar arguments as in the last case.

\[ \square \]

Lemma 8.1.9 Assume \( E \in RCL \). Then
\[
E\sigma \rightsquigarrow E' \\
\downarrow \\
\exists E''. E \rightsquigarrow E'', E''\sigma = E'
\]

Proof By induction (from the definition of \( RCL \)).

\( E \in CL \): Then \( E\sigma = E \rightsquigarrow E' \in CL \). Hence also \( E'\sigma = E' \) and we can chose \( E'' = E' \).

\( E = E_0 \uplus E_1, E_0 \in RCL \) and \( E_1 \in RBL \): We have \( E\sigma = E_0\sigma; E_1\sigma \rightsquigarrow E' \). According to the definition of \( \rightsquigarrow \) there are two subcases to consider:

\( E_0\sigma \uparrow \) and \( E' = E_1\sigma \): \( E_0\sigma = \uparrow \) implies \( E_0 = \uparrow \). Letting \( E'' = E_1 \) we get \( E_0; E_1 \uparrow \rightsquigarrow E'' \) and \( E''\sigma = E_1\sigma = E' \).

\( E_0\sigma \downarrow \) and \( E' = E_0\sigma \uplus E_1\sigma \): By hypothesis \( \exists E''_0. E_0 \uparrow \rightsquigarrow E''_0, E''_0\sigma = E''_0 \). With \( E'' = E''_0 \uplus E_1 \) we get \( E_0; E_1 \uparrow \rightsquigarrow E'' \) and \( E''\sigma = E''_0\sigma; E_1\sigma = E''_0; E_1\sigma = E' \).

\( E = E_0 \parallel E_1 \) and \( E_0, E_1 \in RBL \): Here the situation is \( E\sigma = E_0\sigma \uplus E_1\sigma \). Inspecting the definition of \( \rightsquigarrow \) we see that there only is two possibilities. Assume w.l.o.g. that \( E' = E_0\sigma \). Since \( E_0 \parallel E_1 \uparrow \rightsquigarrow E_0 \) the result then follows if we let \( E'' = E_0 \).

\( E = E_0 \parallel E_1 \) and \( E_0, E_1 \in RCL \): Similar/ symmetric argument as in the case \( E = E_0; E_1 \).

\[ \square \]

Lemma 8.1.10 Suppose \( E \in RCL \) and \( A \in G \). Then

178
\[ E \sigma \xrightarrow{A} E' \]
\[ \exists E'', E \xrightarrow{A} E'', E'' \sigma = E' \]

**Proof**  By induction (from the definition of \( RCL \)).

\( E \in CL \): Then \( E \sigma = E \xrightarrow{A} E' \in CL \), so \( E' \sigma = E' \) and we choose \( E'' = E' \).

\( E = E_0 ; E_1, E_0 \in RCL \) and \( E_1 \in RBL \): We have \( E \sigma = E_0 \sigma ; E_1 \sigma \xrightarrow{A} E' \). Since there only is one rule for \( \xrightarrow{A} \) and expressions of this form we deduce \( E' = E_0' ; E_1 \sigma \) where \( E_0 \sigma \xrightarrow{A} E_0' \). Because \( E_0 \in RCL \) we can use the hypothesis of induction to find an \( E_0'' \) such that \( E_0 \xrightarrow{A} E_0'' \) and \( E_0'' \sigma = E_0' \). From the same rule we then get \( E_0 ; E_1 \xrightarrow{A} E_0'' ; E_1 \). Choose \( E'' = E_0'' ; E_1 \) and we have \( E'' \sigma = E_0'' \sigma ; E_1 \sigma = E_0' ; E_1 \sigma = E' \) as we want.

\( E = E_0 \oplus E_1 \) and \( E_0, E_1 \in RBL \): This means \( E \sigma = E_0 \sigma \oplus E_1 \sigma \), but there is no rule for \( \xrightarrow{A} \) and expressions of this form wherefore the implication holds trivially.

\( E = E_0 \parallel E_1 \) and \( E_0, E_1 \in RBL \): Here we have \( E \sigma = E_0 \sigma \parallel E_1 \sigma \). Three inference rules shall be taken into consideration. The ones with only one of \( E_0 \sigma \) and \( E_1 \sigma \) involved in \( \xrightarrow{A} \) goes similar/symmetric as in the case \( E = E_0 ; E_1 \). If both are involved we have \( E' = E_0 \parallel E_1 \) where \( A = A_0 \times A_1 \) and for \( i = 0, 1 \), \( E_i \sigma \xrightarrow{A_i} E_i' \) and \( A_i \in G \). Since \( E_0, E_1 \in RCL \) the hypothesis of induction can be applied to get for each \( i \) an \( E_i'' \) such that \( E_i \xrightarrow{A_i} E_i'' \) and \( E_i'' \sigma = E_i' \). Since \( A \in G \) then also \( E_0 \parallel E_1 \xrightarrow{A_0 \times A_1} E_0'' \parallel E_1'' \). Choosing \( E'' = E_0'' \parallel E_1'' \) this reads \( E \xrightarrow{A} E'' \) and we see \( E'' \sigma = E_0'' \sigma \parallel E_1'' \sigma = E_0' \parallel E_1' = E' \) and we are done.

\( 8.2 \) Denotational Set-up

As mentioned in the introduction to this chapter we will start out by searching a model for \( \preceq_w^c \). To this end \([\cdot]_w \) is extended to \( RBL \) by letting

\[ [E]_w = [E \sigma]_w \text{ for } E \in RBL \]

The induced denotational preorder \( \preceq_w \) then also extends to \( RBL \) and it follows from proposition 8.1.2 and proposition 7.4.3 that \( \preceq_w = \preceq_w \) on \( RBL \).

In the case of \( \preceq_w ^c \) it is much harder (than in the previous chapter) to see intuitively that \( \preceq_w ^c \) should have a pomset based model at all—and if so what it should look like. But following the pattern from the previous chapter we shall be looking for a model in which the denotation of \( E \) is expressible as \( \delta_e(\wp(E)) \) for a suitable pomset property \( P_e \)—but which?

Playing with examples, one soon realizes that a refinement combinator is quite a powerful tool in distinguishing expressions, because much of the information represented in \( \wp(E) \) may be reflected by suitable refinement combinator \([\wp]\), in the sense of “overlapping”
occurrences of $\varphi$-images of concurrent elements in $p \in \varphi(E)$ (as indicated on page 172 in the example of $\preceq_w$ not being a precongruence for $RBL$). So clearly fewer identifications should be made. Through examples like:

\[ (a \parallel c) ; (b \parallel d) \xrightarrow{\varphi} \left\{ \begin{array}{ll}
\begin{array}{ll}
a & \rightarrow b \\
\downarrow & \\
c & \rightarrow d
\end{array}
\end{array} \right\} \rightarrow \delta \left( \varphi\left( E_0 \right) \right) \]

one might be led to the conjecture that $\delta(\varphi(E))$ ordered under inclusion could be a model for $\preceq_w$. However, this is not the case, as can be seen by looking at the example:

**Example:**

\[ E_0 = (a ; (b \parallel d)) \parallel c \parallel (a \parallel (c ; (b \parallel d))) \xrightarrow{\varphi} \left\{ \begin{array}{ll}
\begin{array}{ll}
a & \rightarrow b \\
\downarrow & \\
c & \rightarrow d
\end{array}
\end{array} \right\} \rightarrow \delta\left( \varphi\left( E_0 \right) \right) \]

\[ E_1 = E_0 \oplus a ; b \parallel c ; d \xrightarrow{\varphi} \left\{ \begin{array}{ll}
\begin{array}{ll}
a & \rightarrow b \\
\downarrow & \\
c & \rightarrow d
\end{array}
\end{array} \right\} \rightarrow \delta\left( \varphi\left( E_1 \right) \right) \]

The inequallity follows from:

\[ p = \begin{array}{ll}
a & \rightarrow b \\
c & \rightarrow d
\end{array} \in \delta(\varphi(E_1)), \quad \begin{array}{ll}
a & \rightarrow b \\
c & \rightarrow d
\end{array} \notin \delta(\varphi(E_0)) \]

We do not intend to prove operational that $E_0 \preceq_w E_1$ and $E_1 \preceq_w E_0$ (it will follow easily from the denotational characterization to be developed), but invite the reader to find convincing arguments for this fact.

So, presumable $p$ should not belong to the denotation of $E$ in a model for $\preceq_w$. Intuitively, an argument could be that no single linearization of a refinement version from $p<\varphi>$ can reflect the full structure of $p$, in the sense that if the images of $a$ and $d$ overlap in such a linearization (reflecting $a$ and $d$ being concurrent) then the image of $c$ must precede that of $b$, and vice versa. Following this intuition one may look for a property expressing when the full structure of a pomset may be reflected in a single linearization of a refined version of it (in the “overlapping” sense).
Based on our example, we suggest the following formalization of this property—expressed as a slight modification of the $P_{and}$-property.

**Definition 8.2.1 $P_{or}$-Property for Pomsets**

A pomset $p$ is said to have the $P_{or}$-property, $P_{or}(p)$ if for all $x, x', y, y'$ in $X_p$ we have:

if $x <_p x'$ and $y <_p y'$ then

$x <_p y'$ and $y <_p x'$

**Example:** $a \rightarrow b \leftarrow c \rightarrow d$ has the $P_{or}$-property, $a \rightarrow b \leftarrow c \rightarrow d$ has not.

$P_{or}$ has an alternative characterization (used extensively in the following), the proof of which is trivial:

**Proposition 8.2.2** A pomset $p = [p]$ has the $P_{or}$-property if for all $x, x', y, y'$ in $X_p$:

if $x <_p x'$ and $y <_p y'$ then

$x <_p y'$ and $y <_p x'$

**Proposition 8.2.3** The $P_{or}$-property is hereditary and dot synthesizable.

**Proof** With the alternative characterization the proposition is proved with similar argumentation as the $P_{and}$-property was proved in proposition 7.3.2 to be hereditary and dot synthesizable.

After these manoeuvres we now give the denotation of an expression $E \in RBL$, $\llbracket E \rrbracket_{or}$, in the model we informally arrived at, that is, finite sets of $P_{or}$-pomsets partial ordered under inclusion.

**Definition 8.2.4** $\llbracket \cdot \rrbracket_{or} : RBL \rightarrow \mathcal{P}(P_{or} \setminus \{\varepsilon\}) \setminus \emptyset$ is defined by $\llbracket E \rrbracket_{or} = \delta_{or}(\varphi(E\sigma))$.

As usual the induced denotational preorder is denoted $\preceq_{or}$.

Returning to the example on page 180 using $\preceq_{or}$ in stead of $\delta$ we now get:

$$E_0 = (a ; (b \parallel d)) \parallel c \oplus a \parallel (c ; (b \parallel d)) \xrightarrow{\nu} \begin{cases} a \rightarrow b \\ c \rightarrow d, \\ a \rightarrow b \\ c \rightarrow d \end{cases} \xrightarrow{\delta_{or}} \delta_{or}(\varphi(E_0))$$

$$\leq_{or}^c$$

$$E_1 = E_0 \oplus a ; b \parallel c ; d \xrightarrow{\nu} \begin{cases} a \rightarrow b \\ c \rightarrow d, \\ a \rightarrow b \\ c \rightarrow d \end{cases} \xrightarrow{\delta_{or}} \delta_{or}(\varphi(E_1))$$
where the equality follows from:

$$\delta_{or}(\varphi(E_1)) = \delta_{or}(\varphi(E_0)) \cup \delta_{or}\left(\left\{\frac{a \rightarrow b}{c \rightarrow d}\right\}\right) = \delta_{or}(\varphi(E_0)) \cup \delta_{or}\left(\left\{\frac{a \rightarrow b}{c \rightarrow d}, \frac{a \rightarrow b}{c \rightarrow d}\right\}\right) = \delta_{or}(\varphi(E_0))$$

For each $BL$-refinement $\varphi$ we associate the corresponding $\mathcal{P}(P)$-refinement $\varphi(\varphi)$, by letting $(\varphi(\varphi))(a) = \varphi(\varphi(a))$. Notice that $\varphi(\varphi)$ is $\varepsilon$-free.

**Theorem 8.2.5** $[\mathbb{L}]_{or}$ has the following compositional definition:

$$
\begin{align*}
[a]_{or} &= \{a\} \\
[E_0; E_1]_{or} &= [E_0]_{or} \cdot [E_1]_{or} \\
[E_0 \circ E_1]_{or} &= [E_0]_{or} \cup [E_1]_{or} \\
[E_0 || E_1]_{or} &= \delta_{or}(\langle E_0, E_1 \rangle) \\
[E(\varphi)]_{or} &= \delta_{or}(\langle E, \varphi(\varphi) \rangle)
\end{align*}
$$

**Proof** Similar to that of theorem 7.3.7, but also using the compositional nature of $\sigma$. The case $E[a]$ is more difficult, so we use some lemma’s proved in the sequel.

$$
\begin{align*}
[E[a]]_{or} &= \delta_{or}(\langle E[a] \rangle) \\
&= \delta_{or}(\langle \varphi(\varphi) \rangle) \\
&= \delta_{or}(\langle \varphi, \varphi(\varphi) \rangle) \\
&= \delta_{or}(\langle \varphi, \varphi(\varphi) \rangle) \\
&= \delta_{or}(\langle \varphi(\varphi) \rangle)
\end{align*}
$$

**Lemma 8.2.6** Let $P$ be a set of posets and $\varphi$ an $\varepsilon$-free $\mathcal{P}(P)$-refinement. Then

$$
\delta_{or}(\langle \varphi(\varphi) \rangle) = \delta_{or}(\langle \varphi \rangle)
$$

**Proof** Clearly it is enough to prove $\delta_{or}(\langle \varphi(\varphi) \rangle) = \delta_{or}(\langle \varphi \rangle)$ for a single $\mathcal{P}(P)$-refinement $\mathcal{P}$. Each inclusion is proven separately.

To see $\delta_{or}(\langle \varphi(\varphi) \rangle) \subseteq \delta_{or}(\langle \varphi \rangle)$ let $q \in \delta_{or}(\langle \varphi(\varphi) \rangle)$. Then $P_{or}(q)$ and there exists a $q' \in \delta_{or}(\langle \varphi \rangle)$ such that $q \leq q'$. Therefore $q' \in \delta_{or}(\langle \varphi \rangle)$ for some $p' \in \delta_{or}(\langle \varphi \rangle)$ and we have $p' \not> p$. But by the nature of $\langle \varphi \rangle$ this implies $\forall r' \in p' \not> r \exists r \in p \not< r$. Hence there exists a $r \in p \not< r$ such that $q \leq q' \leq r$. Since $P_{or}(q)$ we have $q \in \delta_{or}(\langle \varphi \rangle)$.

$\delta_{or}(\langle \varphi(\varphi) \rangle) \supseteq \delta_{or}(\langle \varphi \rangle)$: Suppose $q \in \delta_{or}(\langle \varphi \rangle)$. This means $P_{or}(q)$ and $q \leq [p, \pi_p]$, where $\pi_p$ is a $\varepsilon$-consistent particular refinement for a representative, $p$, of $\varphi$. Since $\delta_{or}(\langle \varphi(\varphi) \rangle) \supseteq \delta_{or}(\langle \varphi \rangle)$, we have $q \in \delta_{or}(\langle \varphi \rangle)$. So it is enough to find an $p' \in \delta_{or}(\langle \varphi \rangle)$ such that $q \leq [p', \pi_{p'}]$, where $\pi_{p'}$ is a $\varepsilon$-consistent particular refinement for a representative, $p'$, of $\varphi$.

By proposition 6.2.3 $q \leq [p, \pi_p]$ implies the existence of a representative, $q$, of $q$ such that $q = \langle X_p, \leq_q, \ell_q \rangle$ and $\leq_q \supseteq \leq_q \supseteq \leq_{p, \pi_p}$.

Define $p := \langle X_p, \leq_{p'}, \ell_{p'} \rangle$, where $\leq_{p'}$ is the reflexive closure of $\leq_{p'} \subseteq X^2_p$ defined by:

$$
\begin{align*}
(x < y) &\iff \forall (x, x'), (y, y') \in X_p. (x, x') < q (y, y')
\end{align*}
$$

182
That is, we order elements \( x, y \) in \( p' \) if and only if all elements from \( \pi_p(y) \) are causally dependent on all elements \( \pi_p(x) \) in \( q \).

To see that \( p' \) in fact is a lpo notice that \( \leq p' \) by definition is reflexive, clearly also transitive and the antisymmetry is seen from (8.4), the \( \varepsilon \)-freeness of \( \pi_p \) (a consequence of \( \varepsilon \) being \( \varepsilon \)-free) and the antisymmetry of \( \leq q \).

\[ X_p = X_{p'} \text{ and } \ell_p = \ell_{p'} \text{ so } p' \preceq p \text{ follows by proving } \leq p' \supseteq \leq p. \] By definition \( x \leq p y \) then \( x \neq y \), so by the construction of \( p < \pi_p > \) we have \( \forall (x, x'), (y, y') \in X_{p < \pi_p >} \) \( (x, x') <_{p < \pi_p >} (y, y') \) and from \( \leq q \supseteq \leq p < \pi_p > \) this implies \( \forall (x, x'), (y, y') \in X_q \) \( (x, x') <_q (y, y') \). By definition of \( <_p \) then \( x <_{p'} y \).

If \( p' \) have the \( P_{or} \)-property it then follows that \( p' \in \delta_{or}(p) \).

Assume that \( p' \) does not have the \( P_{or} \)-property. That is \( X_{p'} \) contain elements \( x_1, x_2, y_1, y_2 \) such that:

\[
\begin{align*}
(8.5) & \quad x_1 <_{p'} y_1 \\
(8.6) & \quad \text{and} \quad x_2 <_{p'} y_2
\end{align*}
\]

From the definition of \( p' \), the \( \varepsilon \)-freeness of \( \varepsilon \) and (8.6) it then follows that there exists \( x_1', x_2', y_1', y_2' \) such that:

\[
\begin{align*}
(8.7) & \quad (x_1, x_1') \not< q (y_2, y_2') \\
& \quad \text{and} \quad (x_2, x_2') \not< q (y_1, y_1')
\end{align*}
\]

From (8.5) then:

\[
(8.8) \quad (x_1, x_1') <_q (y_1, y_1')
\]

But from (8.7) and (8.8) it follows that:

\[
\langle x_1, x_1' \rangle \not<_q \langle x_2, x_2' \rangle
\]

and we have a contradiction to the fact that \( q \) has the \( P_{or} \)-property.

It remains to prove \( q \preceq [p' < \pi_p >] \) for some \( p \)-consistent \( p \text{-ref.} \), \( \pi_{p'}, \) for \( p' \). Since \( X_p = X_{p'} \), \( \pi_p \) is also a \( p \text{-ref.} \) for \( p' \) and we know that it is \( p \)-consistent. For the same reason \( X_{p' < \pi_p >} = X_{p < \pi_p >} = X_q \) and similarly \( \ell_{p' < \pi_p >} = \ell_q \).

Next we show \( \leq q \supseteq \leq p' < \pi_p > \). Assume \( \langle x, x' \rangle \leq p' < \pi_p > (y, y') \). By construction of \( p' < \pi_p > \) this implies \( x <_{p'} y \) or \( (x = y, x' \leq_{\pi_p}(x) y') \). In the former case (8.4) directly gives \( \langle x, x' \rangle <_q (y, y') \) and in the latter case we have \( \langle x, x' \rangle <_{p < \pi_p >} (x, y') \) from the construction of \( p < \pi_p > \). Since \( \leq q \supseteq \leq p < \pi_p > \) this implies \( \langle x, x' \rangle <_q (x, y') \). Hence \( \leq q \supseteq \leq p' < \pi_p > \).

Collecting the facts we can use proposition 6.2.3 again to conclude \( q \preceq [p' < \pi_p >] \) as desired.

**Lemma 8.2.7** \( \varphi(E\{q\}) = (\varphi(E))\langle \varphi(q) \rangle \) for \( E \in BL \).

**Proof** By induction on the structure of \( E \).

183
\[ E = a: \varphi(a\{g\}) = \varphi(\varphi(a)) = (\varphi(\varphi))(a) = \{a\}<\varphi(\varphi)> = \varphi(a)<\varphi(\varphi>).\]

\[ E = E_0; E_1: \quad \varphi(E\{g\}) = \varphi(E_0\{g\}) \cdot \varphi(E_1\{g\}) \quad \text{definition of } \varphi \text{ and } \{g\}\]
\[ = (\varphi(E_0)<\varphi(\varphi)>)(\varphi(E_1)<\varphi(\varphi)>) \quad \text{hypothesis}\]
\[ = (\varphi(E_0) \cdot \varphi(E_1))<\varphi(\varphi)> \quad \text{proposition 6.3.3}\]
\[ = \varphi(E)<\varphi(\varphi)> \quad \text{definition of } \varphi\]

\[ E = E_0 \oplus E_1 \text{ and } E = E_0 \parallel E_1: \text{ Similar.}\]

### 8.3 Full Abstractness

The connection between \([\_\_\_]_\text{or}\) and \([\_\_\_]_\text{w}\) is indicated by:

**Proposition 8.3.1** \([E]_w = \delta_w([E]_\text{or})\) for \(E \in RBL\).

**Proof** \([E]_w = \delta_w(\varphi(E\sigma))\) \quad \text{definition}\]
\[ = \delta_w(\delta_{\text{or}}(\varphi(E\sigma))) \quad \text{since } \delta_w \circ \delta_{\text{or}} = \delta_w\]
\[ = \delta_w([E]_\text{or}) \quad \text{by definition}\]

Furthermore:

**Proposition 8.3.2** \([\_\_\_]_\text{or}\) is a precongruence.

**Proof** Similar to proposition 7.4.1, but with the additional case of \(<g>\), which also is \(\subseteq\)-monotone (proposition 6.3.4).

And in fact:

**Theorem 8.3.3** The denotation \([\_\_\_]_\text{or}\) is fully abstract w.r.t. \(\preceq^c_w\) on \(RBL\).

**Proof** We show that \([\_\_\_]_\text{or}\) is the largest precongruence contained in \(\preceq_w\) or equivalently the largest precongruence contained in \([\_\_\_]_\text{w}\).

By proposition 8.3.2 \([\_\_\_]_\text{or}\) is a precongruence and the containment is seen as follows:
\[ E_0 \trianglelefteq \_\_\text{or} E_1 \Rightarrow [E_0]_\text{or} \subseteq [E_1]_\text{or} \quad \text{by definition}\]
\[ \Rightarrow \delta_w([E_0]_\text{or}) \subseteq \delta_w([E_1]_\text{or}) \quad \delta_w \text{ is } \subseteq\text{-monotone}\]
\[ \Rightarrow [E_0]_w \subseteq [E_1]_w \quad \text{proposition 8.3.1}\]
\[ \Rightarrow E_0 \trianglelefteq \_\_\text{w} E_1 \quad \text{by definition}\]

To show that \([\_\_\_]_\text{or}\) is the largest precongruence contained in \([\_\_\_]_\text{w}\) is harder, so we have deferred the crux of the matter to lemma 8.3.4 below, from which we see \(E_0 \trianglelefteq \_\_\text{or} E_1\) implies that there exists a \(BL\)-refinement \(g\) such that \(E_0[g] \trianglelefteq \_\_\text{w} E_1[g]\). This means that any preorder contained in \([\_\_\_]_\text{w}\) larger than \([\_\_\_]_\text{or}\) would not be a precongruence w.r.t. this combinator: \([g]\).
In the following we will need a special type of refinements—fission refinements—which splits an atomic action into two. In this way the original action is no longer atomic. Our notation is inspired by Hennessy [Hen87b].

For each \( k \in \mathbb{N}^+ \) we shall in the sequel denote the “standard” finite set, \( \{1, \ldots, k\} \), by \( k \). Now let a finite multiplicity function, \( m \), be given. Because \( m \) is finite we can define \( n(m) = \max\{k \mid k = 1 \text{ or } \exists a \in \Delta. m(a) = k\} \in \mathbb{N}^+ \).

Since \( \Delta \) is infinite, but countable, there exists an injective function \( h: \Delta \times \{S, F\} \times n(m) \rightarrow \Delta \). I.e., \( \forall a, a' \in \Delta \forall i, i' \in \{S, F\} \forall k, k' \in n(m) \).

\[
(a, i, k) \neq (a', i', k') \Rightarrow h((a, i, k)) \neq h((a', i', k'))
\]

For convenience we shall abbreviate \( h((a, i, k)) \) by \( a_{ik} \).

With such a function we associate a BL-refinement, \( g \), by defining for all \( a \in \Delta \):

\[
g(a) = a_{S1} \cdot a_{F1} \oplus \ldots \oplus a_{Sn(m)} \cdot a_{Fn(m)}
\]

and call it a \( m \)-fission refinement.

The corresponding \( \varepsilon \)-free \( P(P) \)-refinement, (ambiguously denoted) \( g \), has

\[
g(a) = \{a_{S1} \cdot a_{F1}, \ldots, a_{Sn(m)} \cdot a_{Fn(m)}\}
\]

and is also called a \( m \)-fission refinement.

We shall refer to \( a_{Sk} \) and \( a_{Fk} \) as a fission pair of the \( m \)-fission refinement \( g \). I.e., the pair \( a_{Sk} \) and \( a_{Fk} \) is a fission of \( a \).

If \( \pi_p \) is a \( g \)-consistent p.ref. for a lpo \( p \) we can define two functions \( \pi_p^{\mathcal{S}}, \pi_p^{\mathcal{F}} : X_p \rightarrow X_{p<\pi_p>} \) as follows: \( \pi_p^{\mathcal{S}} \) (respectively \( \pi_p^{\mathcal{F}} \)) is that element \( \langle x, x' \rangle \) where \( x' \in X_{\pi_p(x)} \) and \( \ell_{p<\pi_p>} \pi_p(x') = a_{Sk} \) (respectively \( \ell_{p<\pi_p>} \pi_p(x') = a_{Fk} \)) for some \( k \in n(m) \), \( a = \ell_p(x) \). We will drop the superscript, \( \pi_p \), when it is clear from the context. Due to the construction of \( p<\pi_p> \) and the definition of \( g \) from \( h \) (fulfilling (8.9)) we have:

\[
\begin{align*}
x_S &= y_S \iff x = y \iff x_F = y_F \\
\ell_{p<\pi_p>} \pi_p(x_S) &= a_{Sk} \iff \ell_{p<\pi_p>} \pi_p(x_F) = a_{Fk} \\
\ell_p(x) &= a \iff \exists k \in n(m), \ell_{p<\pi_p>} \pi_p(x_S) = a_{Sk} \\
\ell_{p<\pi_p>} \pi_p(x_S) &= a_{Sk} \iff \exists y \in X_p, y_S = x \\
\ell_{p<\pi_p>} \pi_p(x_S) &= b \iff \exists a \in \Delta, k \in n(m), b = a_{Sk} \text{(namely: } a = \ell_p(x))
\end{align*}
\]

\[
\begin{align*}
x < p \iff y \iff x_F < p<\pi_p> y_S \\
x_S < p<\pi_p> x_F
\end{align*}
\]

Suppose \( p \) is a lpo with \( m_p \leq m \) (i.e \( \forall a \in \Delta. m_p(a) \leq m(a) \)). Then there clearly are \( g \)-consistent p. refinements, \( \pi_p \), injective in the sense:

\[
\forall x, y \in X_p. x \neq y \Rightarrow [\pi_p(x)] \neq [\pi_p(y)]
\]

We call such a \( \pi_p \) for a \( g \)-consistent particular fission refinement for \( p \).
Lemma 8.3.4 Given $E_0 \in RBL$. Then there exists a refinement combinator, $[\varrho]$, such that

$$\forall E_1 \in RBL. \ [E_0]_\text{or} \not\subseteq [E_1]_\text{or} \Rightarrow [E_0[\varrho]]_w \not\subseteq [E_1[\varrho]]_w$$

**Proof** Let $m$ be the finite multiplicity function which is the lub for $\{m_p \mid p \in [E_0]_\text{or}\}$ (finite set). Choose a $m$-fission refinement $\varrho$. The associated refinement combinator, $[\varrho]$, is the one we are after. To see this let an arbitrary $E_1 \in RBL$ be given such that $[E_0]_\text{or} \not\subseteq [E_1]_\text{or}$. The proof is by contradiction. Assume on the contrary $[E_0[\varrho]]_w \subseteq [E_1[\varrho]]_w$, $[E_0]_\text{or} \not\subseteq [E_1]_\text{or}$ only if there is a $p \in [E_0]_\text{or}$ such that $p \not\in [E_1]_\text{or}$. $p \in [E_0]_\text{or}$ implies $P_{\text{or}}(p)$ and by definition also $m_p \leq m$. By lemma 8.3.6 there is a $w \in \delta_w(p<\varrho>)$ which is $p$-reflecting.

Now $w \in \delta_w(p<\varrho>)$ and $p \in [E_0]_\text{or}$ implies $w$ in $\delta_w([E_0]_\text{or}<\varrho>)$ which, because $\delta_w \circ \delta_{\text{or}} = \delta_w(\delta_{\text{or}}([E_0]_\text{or}<\varrho>))$. By theorem 8.2.5 and proposition 8.3.1 then also $w \in [E_0[\varrho]]_w$ and so $w \in [E_1[\varrho]]_w$ by the assumption. Reversing the arguments we find a $q \in [E_1]_\text{or}$ such that $w$ is a linearization of a pomset, $r$, of $q<\varrho>$. Because $w$ is $p$-reflecting we then deduce from lemma 8.3.5 that $p \preceq q$. Since $P_{\text{or}}(p)$ and $[E_1]_\text{or}$ is $\delta_{\text{or}}$-closed then $p \in [E_1]_\text{or}$—a contradiction.

Lemma 8.3.5 Let a fission refinement $\varrho$ be given and suppose $w'$ is $p'$-reflecting. If $w' \preceq r \in q<\varrho>$ then $p' \preceq q$.

**Proof** To see $p' \preceq q$ we at first elucidate the situation. $w'$ being $p'$-reflecting implies there are representatives $w$ of $w'$ and $p$ of $p'$ together with a $\varrho$-consistent p. fission ref., $\pi_p$, such that

$$w = \langle X_{p<\pi_p>}, \leq_w, \ell_{p<\pi_p>} \rangle \leq_w \leq_{p<\pi_p>}$$

We also have $w' \preceq r \in q<\varrho>$ Therefore there is a $\varrho$-consistent p.ref., $\pi_q$, and a morphism of lpos $f : q<\pi_q> \rightarrow w$.

We shall find a morphism $g : q \rightarrow p$. Define

$$g(x) = y \text{ iff } \exists y \in X_p, y_{\pi_q}^p = f(x_{\pi_q}^q)$$

(8.18)
This gives sense since \( X_q \xrightarrow{\pi_q} X_{q<\pi_q} \xrightarrow{f} X_w = X_{p<\pi_p} \xrightarrow{\pi_p} X_p \).

To see that (8.18) actually defines a function \( g : X_q \rightarrow X_p \) we shall prove that there for a given \( x \in X_q \) is one and only one \( y \in X_p \) such that \( y_S = f(x_S) \). At first we notice from \( f \) being label preserving and \( \ell_w = \ell_{p<\pi_p} \) that:

\[
\forall z \in X_{q<\pi_q}, \ell_{q<\pi_q}(z) = \ell_{p<\pi_p}(f(z))
\]

one: \( \pi_q \) is \( \varphi \)-consistent, so by (8.14) \( \ell_{q<\pi_q}(x_S) = a_S \) for some \( a \) and \( k \). From (8.19) then \( \ell_{p<\pi_p}(f(x_S)) = a_S \) and by (8.13) there exists a \( y \in X_p \) with \( y_S = f(x_S) \).

only one: Follows directly from (8.10).

Before continuing we prove

\[
f(x_S) = g(x)_S, \quad f(x_F) = g(x)_F
\]

The first equation holds by definition of \( g \) and the second is seen from the first as follows: \( f(x_S) = g(x)_S \) implies \( \ell_{p<\pi_p}(f(x_S)) = \ell_{p<\pi_p}(g(x)_S) \) which by (8.19) is the same as \( \ell_{q<\pi_q}(x_S) = \ell_{p<\pi_p}(g(x)_S) \). Because \( \pi_p \) and \( \pi_q \) both are \( \varphi \)-consistent we from (8.14) get \( \ell_{q<\pi_q}(x_S) = a_S = \ell_{p<\pi_p}(g(x)_S) \) for some \( a \) and \( k \), so by (8.11) and (8.19) then \( \ell_{p<\pi_p}(f(x_F)) = \ell_{p<\pi_p}(g(x)_F) \). Now \( \pi_p \) is also a \( \varphi \)-consistent p. fission ref. for \( p \), so we conclude \( f(x_F) = g(x)_F \) from (8.17).

As the next step we show \( g \) to be bijective.

\( g \) injective: \( x \neq y \Rightarrow x_S \neq y_S \) by (8.10)
\[
\Rightarrow f(x_S) \neq f(y_S) \quad \text{ \( f \) injective}
\]
\[
\Rightarrow g(x)_S \neq g(y)_S \quad \text{by (8.20)}
\]
\[
\Rightarrow g(x) \neq g(y) \quad \text{by (8.10)}
\]

\( g \) surjective: Given \( y \in X_p \). By (8.14) then \( \ell_{p<\pi_p}(y_S) = a_S \) for some \( a \) and \( k \). Since \( f \) is surjective and label preserving there is an \( x' \in X_{q<\pi_q} \) with \( f(x') = y_S \) and \( \ell_{q<\pi_q}(x') = a_S \). From (8.13) we see that there must be an \( x \in X_q \) with \( x_S = x' \).

In proving \( g \) to be a morphism of lpos it remains to show that \( g \) is label and order preserving.

\( g \) label preserving: Suppose \( x \in X_q \) and \( \ell_q(x) = b \). Then from (8.12) \( \ell_{q<\pi_q}(x_S) = b_S \) for some \( k \), and therefore \( b_S = \ell_{p<\pi_p}(f(x_S)) = \ell_{p<\pi_p}(g(x)_S) \) by (8.19) and (8.20). Using (8.12) again we obtain \( \ell_p(g(x)) = b = \ell_q(x) \).

\( g \) order preserving: Assume \( x \leq_q y \). In the case \( x = y \) the result follows from the reflexivity of \( \leq_p \). In the case \( x <_q y \) we have

\[
g(x)_F <_w g(y)_S
\]

because \( x <_q y \Rightarrow x_F <_q <_p y_S \) by (8.15)
\[
\Rightarrow f(x_F) <_w f(y_S) \quad \text{\( f \) is order preserving}
\]
\[
\Rightarrow g(x)_F <_w g(y)_S \quad \text{by (8.20)}
\]

We cannot have \( g(y)_S <_p g(x)_F \) since it by (8.15) and (8.16) would imply \( g(y)_S <_p <_p g(x)_F \) which in turn from \( \leq_{p<\pi_p} \subseteq \leq_w \) would imply \( g(y)_S <_w g(x)_F \) — contradicting (8.21).

\( g(x) \circ_p g(y) \) can also be excluded since we then from the fact that \( w \) is \( p \)-reflecting would get \( g(y)_S <_w g(x)_F \) — again contradicting (8.21). Hence we are left with \( g(x) <_p g(y) \) as the only possibility and we are done.
For a pomset $p$, let in the sequel $M_p \subseteq X_p$ denote the set of minimal elements of $p$ (w.r.t. $\leq_p$).

We state and prove the lemma referred to in the proof of lemma 8.3.4.

**Lemma 8.3.6** Let $p$ be a pomset with the $P_{or}$-property and $m_p \leq m$, where $m$ is some finite multiplicity function over $\Delta$. Also let $\varrho$ be a $m$-fission refinement. Then there exists a linearization $w$ of $p^{<\varrho}$ (i.e., $w \in \delta_w(p^{<\varrho})$) which is $p$-reflecting under $\varrho$.

**Proof** If $p = \varepsilon$ it is trivial that $w = \varepsilon$ will do, so we can assume $p \neq \varepsilon$ in the following. Since $m_p \leq m$ there is a $\varrho$-consistent p. fission ref., $<\pi_p>$, for $p$. The result is then a consequence of the corresponding statement for lpos:

Let $\pi_p$ be a p. fission ref. for $p \neq \varepsilon$. Assume the minimal elements $M_p$ of $p$ listed in some arbitrary order are: $x_1, \ldots, x_n$. Then there exists an $p$-reflecting linearization $w$ of $p^{<\pi_p}>$ isomorphic to a lpo of the form:

$$x_1S \cdot \ldots \cdot x_nS \cdot v$$

The proof is by induction on the size of $X_p$.

The basis, $X_p$ a singleton, is clear.

So assume $|X_p| > 1$. From proposition 8.3.7 we can find an element $x_i \in M_p$ such that $x_i$ is dominated in $X_p$ by all successors of $M_p$. Consider now the lpo, $p'$, obtained by deleting $x_i$ from $p$.

Notice that $M_p \setminus \{x_i\}$ is a subset of the minimal elements of $p'$, hence we may list $M_{p'}$ as follows:

$$x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n, y_1, \ldots, y_k$$

Clearly $\pi_{p'} = \pi_p|_{X'_p}$ is a $\varrho$-consistent p. fission ref. for $p'$, so because the $P_{or}$ property is inherited to $p'$ we can use the inductive hypothesis to find a $p'$-reflecting linearization $w'$ of $p'^{<\pi_{p'}}> \text{ isomorphic to a lpo of the form}$

$$x_1S \cdot \ldots \cdot x_{i-1}S \cdot x_{i+1}S \cdot \ldots \cdot x_nS \cdot y_1S \cdot \ldots \cdot y_kS \cdot v'$$

Since $x_i$ is minimal in $p$ there are no other elements before $x_iS$ and $x_iF$ in $p^{<\pi_p}$, and so $x_iS \cdot x_iF \cdot w'$ is isomorphic to a possible linearization of $p^{<\pi_p}>$. By the way $x_i$ was chosen, the elements concurrent to $x_i$ are exactly $M_p \setminus \{x_i\}$. Then $x_iS$ and $x_iF$ are concurrent to $x_1S, \ldots, x_{i-1}S, x_{i+1}S, \ldots, x_nS$ in $p^{<\pi_p}>$, from which it follows that

$$x_1S \cdot \ldots \cdot x_iS \cdot \ldots \cdot x_nS \cdot x_iF \cdot y_1S \cdot \ldots \cdot y_kS \cdot v'$$

must be isomorphic to a linearization, $w$, of $p^{<\pi_p}>$, which quite easily is seen to be $p$-reflecting as desired. 

**Proposition 8.3.7** Let $p$ be a nonempty lpo with the $P_{or}$-property and $M$ a subset of the minimal elements $M_p$ of $p$. Then there is an element $z$ of $M$ dominated by all the successors of $M$. 
Proof By induction on the size of $M$. The basis where $M = \{ z \}$ is evident, and for the inductive step choose an $x \in M$. By hypothesis of induction we can find a $y \in M \setminus \{ x \}$, which is dominated by the successors of $M \setminus \{ x \}$. If $y$ is dominated by the successors of $x$ too, we can choose $z = y$. Otherwise, since $p$ has the $P_{or}$-property, and minimal elements are mutual concurrent, the successors of $y$ must dominate $x$. But the successors of $y$ are also the successors of $M \setminus \{ x \}$ and we can choose $z = x$. □

8.4 Summary

Let us sum up the abstractness results we have proved in this chapter. If we let $\equiv$ denote the equivalence associated with an operational preorder $\preceq$, and if we extend $\llbracket G \rrbracket$ to $RBL$ in the same simple way as $\llbracket w \rrbracket$ were extended in section 8.2, we get the following immediate corollary:

Corollary 8.4.1 For all $E_0, E_1 \in RBL$:

\begin{align*}
E_0 \lessdot w E_1 & \quad \text{iff} \quad \llbracket E_0 \rrbracket_w = \llbracket E_1 \rrbracket_w \\
E_0 \lessdot G E_1 & \quad \text{iff} \quad \llbracket E_0 \rrbracket_G = \llbracket E_1 \rrbracket_G \\
E_0 \gtrdot w E_1 & \quad \text{iff} \quad \llbracket E_0 \rrbracket_{or} = \llbracket E_1 \rrbracket_{or}
\end{align*}

It follows from $P_M = P_{and}$, $P_{and}(p) \Rightarrow P_{or}(p)$ and definitions that $\llbracket \cdot \rrbracket_M$ is as abstract as $\llbracket \cdot \rrbracket_{or}$ on $BL$. The following two expressions:

\begin{align*}
E_0 &= a ; b \parallel c \\
E_1 &= (a \parallel c) ; b \oplus a ; (b \parallel c)
\end{align*}

show that $\llbracket \cdot \rrbracket_M$ it is strictly more abstract than $\llbracket \cdot \rrbracket_{or}$ (identified by $\llbracket \cdot \rrbracket_M$, but not by $\llbracket \cdot \rrbracket_{or}$).

Furthermore, from the full abstractness results, the fact that $P_w \Rightarrow P_G \Rightarrow P_M (= P_{and}) \Rightarrow P_{or}$, and the examples in the summary from the last chapter we get:

Corollary 8.4.2 For all $E_0, E_1 \in RBL$:

$$(E_0 \lessdot w E_1) \Rightarrow (E_0 \lessdot G E_1) \Rightarrow (E_0 \lessdot w E_1)$$

and none of the implications hold in the other direction except of course in the last implication if $G$ equals $w$.

At this stage, it seems very natural to ask what would have happened, if we had chosen to look for a denotational characterization of the $RBL$ congruence associated with the different operational $G$-sequences rather than action-sequences, i.e., a characterization of $\lessdot G$ on $RBL$. Operationally it seems hard to say anything directly. However, from $\lessdot G \subseteq \lessdot w$ follows $\lessdot G \subseteq \lessdot w$ so $\lessdot w$ is at least as large as the largest precongruence contained in $\lessdot G$. $\lessdot w$ is a precongruence by definition and by $\lessdot w \subseteq \lessdot G$ it then follows that $\lessdot w$ is the largest precongruence contained in $\lessdot G$, i.e., $\lessdot G = \lessdot w$. From the last corollary but one we then obtain the full abstractness result:

Corollary 8.4.3 For all $E_0, E_1 \in RBL$:

\begin{align*}
E_0 \lessdot G E_1 & \quad \text{iff} \quad E_0 \lessdot w E_1 \\
E_0 \gtrdot G E_1 & \quad \text{iff} \quad \llbracket E_0 \rrbracket_{or} = \llbracket E_1 \rrbracket_{or}
\end{align*}
8.5 An Adequate Logic for RBL without Auto-Parallelism

Along the lines of section 7.6 we would in this section for RBL like to find an adequate linear logic $L_r$ for $\varepsilon_G$. Unfortunately it seems insurmountable to devise such a logic for the full RBL language. We shall therefore confine ourselves to search for an adequate logic for the sublanguage, RBL', of RBL where processes have no auto-parallelism (see [vGV87]). That is the same action may not occur on both sides of a $\parallel$-combinator as in e.g., $a \parallel b \parallel c \parallel a$. Similar BL' will be the sublanguage of BL with no auto-parallelism.

Formally let $L(E)$ denote the sort/label set/set of actions of an $E \in BL$. Then BL' is those expressions of BL where all subexpressions of the form $E_0 \parallel E_1$ fulfills $L(E_0) \cap L(E_1) = \emptyset$. A BL'-refinement will be a mapping $\varrho : \Delta \rightarrow BL'$ with the additional requirement:

$$a \neq b \Rightarrow L(\varrho(a)) \cap L(\varrho(b)) = \emptyset$$

RBL' is those $E \in RBL$ where $E \varrho \in BL'$ and if $[\varrho]$ is a refinement combinator of $E$ then $\varrho$ is a BL'-refinement. Due to the restrictions on the expressions of BL' and RBL' the congruence and full abstractness results to follow should be modified accordingly.

The decisive importance of BL' and RBL' is that the canonical pomset association map, $\varphi$, when used on expressions of BL' yield pomsets with the $P_{sw}$-property (i.e., semwords) we encountered in section 7.5. Recall that in a $P_{sw}$-pomset all equally labelled elements are ordered. Consequently one can speak of the $i^{th}$ occurrence of a label $a \in \Delta$. The corresponding element of $\hat{p}$ will be $\langle i, a \rangle \in X_{\hat{p}}$, where $\hat{p}$ is the canoncic representative of $p$ we mentioned in section 7.5. For more details on this matter see Starke [Sta81]. From the results there a stronger version of the alternative characterization of $\preceq$ on $P_{sw}$-pomsets appears: If $p, q \in P_{sw}$ then

$$p \preceq q \iff X_{\hat{p}} = X_{\hat{q}} \text{ and } \leq_{\hat{p}} \supseteq \leq_{\hat{q}}$$

Another of the $P_{sw}$-pomsets characteristics is that:

if $p \in P$, $q \in P_{sw}$ and $p \preceq q$ then $p \in P_{sw}$.

Combining this with the fact that $\varphi(E) \in P_{sw}$ when $E \in BL'$ we see that the denotational maps of the different models when restricted to BL' respectively RBL' only yield $P_{sw}$-pomsets. We can therefore choose to work with canoncic representatives in stead.

To this end denote the set of lpos which are the canoncic representatives of some $P_{sw}$-pomset by $SW$ and call $SW$ the set of semwords (over $\Delta$). For $p, q \in SW$ the partial order then becomes:

$$p \preceq q \iff X_p = X_q \text{ and } \leq_p \supseteq \leq_q$$

and the pomset operations inherits to $SW$ via the canoncic representatives. E.g., $p \times q := p \hat{\times} q$ where the $\times$-operator under $\hat{\cdot}$ is the lpo parallel composition introduced in section 6.1. In order to insure this to be well-defined $p$ and $q$ must be disjoint. This will be assumed henceforth when writing $p \times q$. Because as we noticed in section 7.5 the $P_{sw}$-property is both hereditary and dot synthesizable it follows that all the purely denotational results carry over to BL' and RBL' (see also section 6.4).
On the operational side nothing changes except that $CL$ and $RCL$ are changed accordingly. But due to the nature of the process expressions we now focus on there is no point in regarding set of direct tests larger than $S \subseteq P_{sw}$. We shall therefore assume

$$\Delta \subseteq G \subseteq S$$

The full abstractness results of course holds for $BL'$. It is however not so obvious that $\preceq^G$ will be fully abstract with the semiword version of $[\_]$ or. That this is the case can be seen by passing through section 8.3 with semiwords in mind and observing that the $BL$-refinement used in lemma 8.3.4 actually is a $BL'$-refinement. The result for $\preceq^S$ is also reported in [NEL89]. There a simpler $BL'$ refinement without $\oplus$ is used in the lemma corresponding to lemma 8.3.4 (this only works for semiwords—see the conclusion of that paper). Furthermore direct definitions of the different semiword operations is given—especially the definition of the refinement operator is not strait forward.

We will now introduce the linear logics. For $BL'$ and $\preceq^G$ we can use the logic $L_G$ from section 7.6. Since $G \subseteq S$ it will do with modalities $\boxdot$ and $\Box$ where $A \in S$.

For $RBL'$ a stronger modal language is needed. We shall also denote this language by $L$ and define it to be the formulae obtained from:

$$f ::= tt \mid ff \mid \nabla \mid \Delta \mid \Box f \mid \Box f \mid \Diamond f \mid \Diamond f$$

where $A$ can be any element of $S$.

$L_r \subseteq L$ is defined to be those formulae with no occurrence of the modalities $\boxdot$ and $\Box$. The intuition behind $\Box f$ is a kind of semi-deadlock. I.e., a process satisfies $\Box f$ if it either is what Stirling [Sti85] calls $a$-deadlocked for some $a \in A$ or it satisfies $f$. Dually a process satisfies $\boxdot f$ if it is able to perform every $a \in A$ and it also satisfies $f$. Formally:

**Definition 8.5.1** $\models_G DCL' \times L$ is defined:

- $E \models_G tt$ for all $E \in DCL'$
- $E \models_G \nabla$ iff $\forall a \in \Delta, E \not\models_G f$
- $E \models_G \Delta$ iff $\exists a \in \Delta, E \models_G f$
- $E \models_G \Box f$ iff $A \subseteq \{a \in \Delta \mid E \not\models_G f\}$ and $E \models_G f$
- $E \models_G \Diamond f$ iff $\exists a \in A, E \not\models_G f$ or $E \models_G f$
- $E \models_G \Box f$ iff $E \models_G E' \models_G E'$ and $E' \models_G f$
- $E \models_G \Diamond f$ iff $\forall E'. E \models_G E' \models_G E'$ implies $E' \models_G f$

As in section 7.6 we say that a process $E' \in BL'$ satisfies a formula $f \in L_r$,

$$E \models_G f \iff \forall E' \in \text{Beh}(E). E' \models_G f$$
With the syntactic substitution $\sigma : RBL' \rightarrow BL'$ it is then possible to extend $\models_G$ further to $RBL'$ as follows: $E \in RBL'$ satisfies a formula $f \in \mathcal{L}_r$,

$$E \models_G f \iff E\sigma \models_G f$$

For $E \in RBL'$ we define:

$$\mathcal{L}_G'(E) = \{ f \in \mathcal{L}_r \mid E \models_G f \}$$

and in order to prove the adequacy of $\mathcal{L}_G'$ w.r.t. $\not\models_G$ on $RBL'$ we shall also introduce for $E \in RBL'$:

$$\overline{\mathcal{L}}_G'(E) = \{ f \in \overline{\mathcal{L}}_r \mid \exists E' \in \text{Beh}(E\sigma). E' \models_G f \}$$

where $\overline{\mathcal{L}}_r$ is $\{ \overline{f} \in \mathcal{L} \mid f \in \mathcal{L}_r \}$ and $\overline{\_} : \mathcal{L} \rightarrow \mathcal{L}$ is the syntactic map of section 7.6 extended to this larger formula language by:

$$\overline{\mathcal{A}}f = \mathcal{A}\overline{f} \quad \overline{\mathcal{U}}f = \overline{\mathcal{A}}\overline{f}$$

That is the formulae of $\overline{\mathcal{L}}_r$ are:

$$f ::= \text{tt} \mid ff \mid \bigtriangledown \mid \mathcal{A}f \mid \mathcal{G}f$$

Similarly as we proved lemma 7.6.2 we here for $E_0, E_1 \in RBL'$ get:

$$\overline{\mathcal{L}}_G'(E_0) \subseteq \overline{\mathcal{L}}_G'(E_1) \iff \mathcal{L}_G'(E_0) \supseteq \mathcal{L}_G'(E_1) \quad (8.22)$$

We are now ready to give the theorem from which that adequacy of $\mathcal{L}_G'$ follows:

**Theorem 8.5.2** (Linear Logic Characterization) For all $E_0, E_1 \in RBL'$:

$$E_0 \not\leq_G E_1 \iff \mathcal{L}_G'(E_0) \supseteq \mathcal{L}_G'(E_1)$$

**Proof** Immediate from (8.22) and lemma 8.5.17 at the end of the section which states:

$$\overline{\mathcal{L}}_G'(E_0) \subseteq \overline{\mathcal{L}}_G'(E_1) \iff E_0 \not\leq_G E_1 \quad (8.23)$$

Lemma 7.6.4 corresponding to (8.23) for $BL$ was proved through operational argumentation. This is not so easily done here, but if we introduce a satisfaction relation based on semiwords we can utilize our knowledge of the models characterizing $\not\leq_G$. For this purpose we introduce some additional concepts and conventions for semiwords.

As for the proof of lemma 8.3.6 $M_p$ will for $p \in SW$ denote the minimal elements of $X_p$ w.r.t. $\leq_p$. Suppose $a \in \Delta$ is the label of an element of $X_p$. Due to the nature of $p \in SW$ the first occurrence of an element of $X_p$ labelled with $a$ will then be $\langle 1, a \rangle \in M_p$. That is $\langle 1, a \rangle$ is the unique element of $M_p$ labelled $a$ (see page 190). For $A \subseteq \Delta$ or equally $A \in S$ we can therefore make the convention to identify $A$ with the uniquely determined set $\{ \langle 1, a \rangle \mid a \in A \}$ such that it is sensible to write e.g., $A \subseteq M_p$ (or $a \in M_p$ for that matter). Obviously we then have:
Corollary 8.5.3

a) $M_p = \emptyset$ iff $p = \varepsilon$

b) $M_a = \{a\}$ ($=a$)

c) $M_{p,q} = \left\{ \begin{array}{ll} M_p & \text{if } p \neq \varepsilon \\ M_q & \text{if } p = \varepsilon \end{array} \right.$

d) $M_{p \times q} = M_p \cup M_q$

With the conventions we can if $A \subseteq M_p$ define the complement semiword, $\overline{A}^p$, of $A$ in $p$ to be the semiword $\hat{q}$ where $q$ is $p|(X_p \setminus A)$. The construction of $\overline{A}^p$ could of course be done directly from $A$ and $p$.

Example: Suppose $A = \{a, b\}$ and $p = a \xrightarrow{a} b \xrightarrow{c} b$. Then $M_p = \{a, b, c\}$ and $\overline{A}^p = a \xrightarrow{a} b \xrightarrow{c} c$. On second thoughts one realize the truth of

Corollary 8.5.4

a) $\overline{\overline{A}}^p = p$, $\overline{\overline{A}}^p = \varepsilon$

b) $A \subseteq M_p$ implies $\overline{A}^{p \times q} = \overline{A}^p \cdot q$

c) $A \subseteq M_p$, $B \subseteq M_q$ and $p$ disjoint to $q$ implies

$A \cup B \subseteq M_{p \times q}$ and $\overline{A} \times \overline{B}^{p \times q} = \overline{A}^p \times \overline{B}^q$

It is also easy to observe that $A \subseteq M_p$ and $A \subseteq M_q$ implies:

$\overline{A}^p \preceq \overline{A}^p$ iff $X_p \setminus A = X_q \setminus A$ and $\leq_p|_{(X_p \setminus A)^2} \supseteq \leq_q|_{(X_q \setminus A)^2}$

Proposition 8.5.5

a) $p \preceq q \Rightarrow M_p \subseteq M_q$

b) $p \preceq q$ and $A \subseteq M_p$ implies $\overline{A}^p \preceq \overline{A}^q$

Proof

a) Assume on the contrary that there exists a $x \in M_p$ such that $x \not\in M_q$. Since $X_p = X_q$ and $x \in M_p$ we have $x \in X_q$. Hence $x \not\in M_q$ implies there is a $y \in X_q$ with $y <_q x$. But then from $\leq_p \supseteq \leq_q$ also $y <_p x$—a contradiction to $x \in M_p$.

b) From a) and the hypothesis of the implication we see $A \subseteq M_q$. Hence $\overline{A}^q$ is well-defined. $p \preceq q$ gives us $X_p = X_q$ and $\leq_p \supseteq \leq_q$. b) is then immediate from (8.24).
With the ability to regard elements of $S$ as minimal elements of a semiword and with the notion of complement semiwords we can define the semiword satisfaction relation $|=_{SW}^G$:

**Definition 8.5.6** $|=_{SW}^G \subseteq SW \times \overline{C}_r$ is defined inductively by:

\[
\begin{align*}
    p &\vdash_{SW}^G \text{tt} \quad \text{for all } p \in SW \\
    p &\vdash_{SW}^G \text{\n} \iff p = \varepsilon \\
    p &\vdash_{SW}^G \Delta \quad \iff p \neq \varepsilon \\
    p &\vdash_{SW}^G A \hat{f} f \iff A \subseteq M_p \text{ and } p \vdash_{SW}^G f \\
    p &\vdash_{SW}^G A f \hat{f} f \iff A \in G, A \subseteq M_p \text{ and } \mathcal{F} \vdash_{SW}^G f
\end{align*}
\]

$|=_{SW}^G$ is extended to $\mathcal{P}(SW) \times \overline{C}_r$ by letting:

\[
P \vdash_{SW}^G f \iff \exists p \in P, p \vdash_{SW}^G f
\]

At first we want to establish a connection between the operational based satisfaction relation $|=_G$ (restricted to $RBL' \times \overline{C}_r$) and the semiword based $|=_{SW}^G$. $\varphi$ is therefore extended to $CL'$ by keeping it’s compositional definition, but adding $\varphi(\dagger) = \varepsilon$ for the extinct action. Naturally $\varphi(E) = \{p\}$ when $E \in BL'$ so we shall often identify $\varphi(E)$ with $p$ in such situations. To get the connection some lemmas are needed.

**Lemma 8.5.7** Suppose $E, E' \in DCL'$. Then:

a) $E \Rightarrow^* E'$ implies $\varphi(E) = \varphi(E')$

b) $\varphi(E) = \varepsilon$ iff $E \Rightarrow^* \dagger$

c) $E \vdash_G \n \iff E \Rightarrow^* \dagger$

d) $\{a \in \Delta \mid E \vdash_G a\} = M_{\varphi(E)}$

**Proof**

a) Induction in the structure of $E$ using the fact that $\varepsilon$ is neutral to $\cdot$ and $\times$ on $SW$.

b) The *only if* part follows by a trivial induction on the structure of $E$ and the *if* part is just a special case of a)

c) (7.9) on page 170.

d) By induction on the structure of $E$.

$E = \dagger$: $\{a \in \Delta \mid \dagger \vdash_G a\} = \emptyset = M_{\varepsilon} = M_{\varphi(E)}$.

$E = b$: $\{a \in \Delta \mid \vdash_G a\} = \{b\} = M_b = M_{\varphi(E)}$.

$E = E_0 ; E_1$: We consider two subcases:
$E_0 \nrightarrow^* \vdash$: From lemma 7.4.5 and proposition 7.2.3 it then follows that \( \{ a \in \Delta \mid E \circlearrowleft \} = \{ a \in \Delta \mid E_0 \circlearrowleft \} \) which by hypothesis of induction equals \( M_{\psi(E_0)} \).

Now from b) \( E_0 \nrightarrow^* \vdash \) implies \( \varphi(E_0) \neq \varepsilon \), so corollary 8.5.3 gives \( M_{\psi(E_0)} = M_{\psi(E_0) \circ \varphi(E_1)} \). By definition of \( \varphi \) then also \( M_{\psi(E_0)} = M_{\psi(E)} \).

\( E_0 \nrightarrow^* \vdash: \) Similar we see \( \{ a \in \Delta \mid E \circlearrowleft \} = \{ a \in \Delta \mid E_0 \circlearrowleft \} \cup \{ a \in \Delta \mid E_1 \circlearrowleft \} \) and \( \varphi(E_0) = \varepsilon \), so the result follows in the same way but using the hypothesis on \( E_1 \) instead.

\( E = E_0 \parallel E_1: \) This time we see \( \{ a \in \Delta \mid E \circlearrowleft \} = \{ a \in \Delta \mid E_0 \circlearrowleft \} \cup \{ a \in \Delta \mid E_1 \circlearrowleft \} \) from lemma 7.4.6 and proposition 7.2.3. The rest then follows along the lines above.

\( \square \)

**Lemma 8.5.8** For \( E \in DCL' \):

a) \( E \circlearrowleft \) \( E' \) implies \( A \subseteq M_{\psi(E)} \) and \( \overline{A}^{\psi(E)} = \varphi(E') \)

b) \( A \subseteq M_{\psi(E)} \) and \( A \in \mathbb{G} \) implies \( \exists E'. \) \( E \circlearrowleft \) \( E' \)

**Proof**

a) \( E \circlearrowleft \) \( E' \) only if there are \( F \) and \( F' \) such that \( E \nrightarrow^* \vdash \) \( F \xrightarrow{a} F' \nrightarrow^* \vdash \) \( E' \), so from a) of the preceding lemma we see it is enough to prove

\[
E \xrightarrow{A} \mathbb{G} \ E' \implies A \subseteq M_{\psi(E)} \text{ and } \overline{A}^{\psi(E)} = \varphi(E')
\]

By induction of the size of \( E \xrightarrow{A} \mathbb{G} \ E' \) one easily shows \( a \in A \) implies \( E \xrightarrow{a} \mathbb{G} \), so \( A \subseteq \{ a \in \Delta \mid E \circlearrowleft \} \). By d) of the preceding lemma therefore \( A \subseteq M_{\psi(p)} \).

We now know that \( \overline{A}^{\psi(E)} \) is well-defined and it make sense to prove

\[
E \xrightarrow{A} \mathbb{G} \ E' \implies \overline{A}^{\psi(E)} = \varphi(E')
\]

by induction on the size, \( m \), of \( E \xrightarrow{A} \mathbb{G}_m \mathbb{E} \). Only the inductive step is interesting. We consider the different rules one by one.

\( E = a \xrightarrow{A} \mathbb{G}_{m+1} \mathbb{E}: \) Clearly \( A = a \) and \( E' = \vdash \). Now \( \overline{a} = \varepsilon \) and \( \varphi(a) = a \), so \( \overline{a}^{\psi(E)} = \varepsilon = \varphi(E') \).

\( E = E_0 \parallel E_1 \xrightarrow{A} \mathbb{G}_{m+1} \mathbb{E}_1 \mathbb{E}_1 = \mathbb{E}' \) where \( E_0 \xrightarrow{A} \mathbb{G}_m \mathbb{E}_0 \mathbb{E}_1 : \) By induction \( \overline{A}^{\psi(E_0)} = \varphi(E_0) \). So by corollary 8.5.4 c) \( \overline{A}^{\psi(E_0)} = \overline{A}^{\psi(E_0)} \cdot \varphi(E_1) = \overline{A}^{\psi(E_0)} \cdot \varphi(E_1) = \varphi(E_0) \cdot \varphi(E_1) = \varphi(E') \).

\( E = E_0 \parallel E_1 \xrightarrow{A} \mathbb{G}_{m+1} \mathbb{E}' \): There are three ways \( E_0 \parallel E_1 \xrightarrow{A} \mathbb{G}_{m+1} \mathbb{E}' \) could be obtained, and each case is proved essential as above but this time using corollary 8.5.4 d).

b) By induction on the structure of \( E \).

\( E = \vdash: \) \( \varepsilon \notin \mathbb{G} \) by definition of \( \mathbb{G} \) so \( A \neq \varepsilon \) (\( = \emptyset \)) and we cannot have \( A \subseteq M_{\psi(E)} = \emptyset \).

\( E = a: \) \( M_{\psi(a)} = a \) and we must have \( A = a \). From \( a \xrightarrow{A} \mathbb{G} \vdash \) the implication follows with \( E' = \vdash \).

195
Lemma 8.5.9
now be stated and proved:
With the previous two lemmas the connection between the two satisfaction relations can
be first we prove a restricted/modified version of the proposition:

\[ (8.25) \]

The proof of this will be by induction on the structure of \( f \).

\[ f = \text{tt or } f = \text{ff}: \text{ Evident.} \]

\[ f = \nabla: E \models_G f \iff \forall a \in \Delta. E \not\models_G a \quad \text{definition of } \models_G \]

\[ \iff M_{p(E)} = \emptyset \quad \text{lemma 8.5.7} \]

\[ \iff \varphi(E) = \varepsilon \quad \text{corollary 8.5.3} \]

\[ \iff \models_G \varphi(E) \quad \text{definition of } \models_G \]

\[ f = \Delta: \text{ With the last case we see: } E \models_G \Delta \iff E \not\models_G \nabla \iff \varphi(E) \not\models_G \nabla \iff \models_G \varphi(E) \]

\[ f = \overline{\Delta} a: \text{ By hypothesis of induction } E \models_G g \iff \varphi(E) \not\models_G g \]

With the previous two lemmas the connection between the two satisfaction relations can
now be stated and proved:

**Lemma 8.5.9** If \( f \in \overline{\mathcal{L}_r} \) and \( E \in RBL' \) then

\[ E \models_G f \iff \varphi(E \sigma) \models_{SW} f \]

**Proof** At first we prove a restricted/modified version of the proposition:
If \( E \in DCL' \) and \( f \in \overline{\mathcal{L}_r} \) then

\[ (8.25) \]

\[ f \in \overline{\mathcal{L}_G'}(E) \iff \varphi(E) \models_{SW} f \]

The proof of this will be by induction on the structure of \( f \).

\[ f = \text{tt or } f = \text{ff}: \text{ Evident.} \]

\[ f = \nabla: E \models_G f \iff \forall a \in \Delta. E \not\models_G a \quad \text{definition of } \models_G \]

\[ \iff M_{p(E)} = \emptyset \quad \text{lemma 8.5.7} \]

\[ \iff \varphi(E) = \varepsilon \quad \text{corollary 8.5.3} \]

\[ \iff \models_G \varphi(E) \quad \text{definition of } \models_G \]

\[ f = \Delta: \text{ With the last case we see: } E \models_G \Delta \iff E \not\models_G \nabla \iff \varphi(E) \not\models_G \nabla \iff \models_G \varphi(E) \]

\[ f = \overline{\Delta} a: \text{ By hypothesis of induction } E \models_G g \iff \varphi(E) \not\models_G g \]

and from lemma 8.5.7
\[ \{a \in \Delta \mid E \models_G a \} = M_{p(E)} \], so the result follows by the similarity of the definitions
of \( \models_G \) and \( \models_{SW} \).
\( f = \Diamond : \) Follows directly from lemma 8.5.8, hypothesis of induction and the definitions of \( \models_G \) and \( \models^\text{SW}_G \).

With (8.25) the lemma now follows:

\[
\begin{align*}
\text{If } & f \in \overline{\mathcal{L}}_G(E) \iff \exists E' \in \text{Beh}(E\sigma). E' \models_G f & \text{definition of } \overline{\mathcal{L}}_G \\
\text{iff } & \exists E' \in \text{Beh}(E\sigma). \varphi(E') \models_G f & (8.25) \text{ above} \\
\text{iff } & \exists p \in \varphi(E\sigma). p \models^\text{SW}_G f & (8.26) \text{ below} \\
\text{iff } & \varphi(E\sigma) \models^\text{SW}_G f & \text{by extension of } \models^\text{SW}_G \text{ to sets}
\end{align*}
\]

In the deduction we used

\[(8.26) \quad \forall E \in \mathcal{B}L', \varphi(E) = \{ \varphi(E') \mid E' \in \text{Beh}(E) \}\]

which follows by induction on the structure of \( E \) using the compositional nature of \( \varphi \). \( \square \)

It is appropriate here to recall the note at the end of section 7.6 where it was pointed out that an alternative logic characterization of \( \leq_G \) (on \( \mathcal{B}L' \)) could be obtained from \( \mathcal{L}_g \) by pretending definition 8.5.1 of \( \models_G \) was for \( \mathcal{B}L' \times \mathcal{L}_g \) and not just \( D\mathcal{L}' \times \mathcal{L}_g \). The reason was that for \( f \in \mathcal{L}_g \) and \( E \in \mathcal{C}L' \) one would have:

\[ E \models_G f \iff \exists E' \in \text{Beh}(E). E' \models_G f \]

For the extended logic language here this would not be true. Just consider \( E = a \oplus b \) and \( f = \Box \text{tt} \) where \( A = \{ a, b \} \). Then \( E \models_G f \), but for all \( E' \in \text{Beh}(E) = \{ a, b \} \) \( E' \not\models_G f \).

With the last lemma we can now concentrate fully on properties of the semiword based satisfaction relation.

**Lemma 8.5.10** Suppose \( f \in \mathcal{L}_r \) and \( p, q \in \text{SW} \). Then

\[ p \models^\text{SW}_G f, p \preceq q \text{ implies } q \models^\text{SW}_G f \]

**Proof** Induction on the structure of \( f \).

- \( f = \text{tt, ff, } \lor, \Delta : \) Either trivial or follows directly from \( \varepsilon \preceq p \iff p = \varepsilon \).

- \( f = \Box g : \) Then \( A \subseteq M_p \) and \( p \models^\text{SW}_G g \). By induction \( q \models^\text{SW}_G g \) and from proposition 8.5.5 a) \( M_p \subseteq M_q \), so \( A \subseteq M_q \) and we get the result.

- \( f = \Diamond g : \) This implies \( A \subseteq M_p \) and \( A' \models^\text{SW}_G g \). By proposition 8.5.5 b) it follows from \( p \preceq q \) that \( A' \) is well-defined and \( A' \preceq A' \). By hypothesis of induction then \( A' \models^\text{SW}_G g \). Using proposition 8.5.5 a) we have \( A \subseteq M_q \), so \( q \) actually satisfies \( \Diamond g = f \) as desired.

\( \square \)

**Lemma 8.5.11** If \( p \in \text{SW} \) then \( p \in \text{max}_{\preceq}(\delta_{\text{or}}(q)) \) implies \( M_p = M_q \)

**Proof** Assume \( p \in \text{max}_{\preceq}(\delta_{\text{or}}(q)) \).
\( \subseteq: \) The assumption implies \( p \leq q \), so this inclusion follows from proposition 8.5.5.

\( \supseteq: \) Given \( x \in M_q \). Suppose \( x \not\in M_p \). We show that this leads to a contradiction by finding a \( r \in \delta_{or}(q) \) such that \( p < r \). Define \( r \) to be \( \langle X_p, \leq_r, \ell_p \rangle \), where \( \leq_r = \leq_p \setminus \{ (y, x) \mid y <_p x \} \).

By a moment's reflection one sees that \( \leq_r \) defines a partial order. \( p \leq r \) follows by definition and \( p < r \) from \( x \in M_r \) and \( x \not\in M_p \). Since we only have removed relations leading to \( x \) we see from \( x \in M_q \) and \( p \leq q \) that \( r \leq q \) must hold. It remains to show that \( r \) has the \( P_{or} \)-property. Let \( y, y', z, z' \in \overline{X_r} \) be given such that

\[
\begin{align*}
    y &<_r y' \\
    z &<_r z' \\
    y &\text{co}_r z
\end{align*}
\]

(8.27)

We shall then show \( y \leq_r z' \) or \( z \leq_r y' \). From \( p \leq r \) and (8.27) we see \( y <_p y' \), \( z <_p z' \) and since relations are removed from \( \leq_p \) to obtain \( \leq_r \) iff they lead to \( x \), we conclude \( x \neq y', z' \).

If \( y \text{co}_p z \) then \( P_{or}(p) \) implies \( y \leq_p z' \) or \( z \leq_p y' \). Since \( x \neq y', z' \) we must then also have \( y \leq_r z' \) or \( z \leq_r y' \).

It remains to consider \( y \phi_p z \)–i.e., either \( y \leq_p z \) or \( z \leq_p y \). Suppose \( y \leq_p z \). Since \( z <_p z' \) the transitivity of \( \leq_p \) yields \( y \leq_p z' \) and from \( x \neq z' \) we then conclude \( y \leq_r z' \). Similar for \( z \leq_p y \). \( \square \)

**Lemma 8.5.12** Suppose \( A \in S \) and \( P \subseteq SW \) has the property \( \forall p \in P \). \( A \subseteq M_p \). Then

\[
\max_{\leq}\{ \overline{A'} \mid p \in P \} \subseteq \{ \overline{A'} \mid p \in \max_{\leq} P \}.
\]

**Proof** Given \( \overline{A'} \in \max_{\leq}\{ \overline{A'} \mid p \in P \} \). I.e., \( q \in P \) and there is no \( p \in P \) such that \( \overline{A'} < \overline{A'} \) by (8.24):

\[
\exists p \in P. \ X_q \setminus A = X_p \setminus A, \ \leq_q|_{(X_q \setminus A)^2} \supset \leq_p|_{(X_p \setminus A)^2}
\]

So if \( p \in P \) and \( q \leq p \) we must have \( \leq_q|_{(X_q \setminus A)^2} = \leq_p|_{(X_p \setminus A)^2} \). Hence the partial order of any maximal element \( r \) of \( P \) \((r \in \max_{\leq} P)\) above \( q \) \((q \leq r)\) agrees on \( X_q \setminus A \), wherefore \( \overline{A'} = \overline{A'} \) and we are done. \( \square \)

With \( P = \{ a \rhd b, a \lhd b \} \) and \( A = a \) it follows that the right hand side of the inclusion in the lemma may be different from the left hand side.

**Lemma 8.5.13** If \( A \subseteq M_q \) and \( P = \{ p \in \delta_{or}(q) \mid A \subseteq M_p \} \) then \( \delta_{or}(\overline{A'}) = \{ \overline{A'} \mid p \in P \} \).

**Proof**

\( \subseteq: \) Given \( r \in \delta_{or}(\overline{A'}) \), i.e., \( P_{or}(r) \) and \( r \leq \overline{A'} \). Let \( p = A \cdot r \). Then \( \overline{A'} = r \) and we have \( \overline{A'} \leq \overline{A'} \) or equally by (8.24):

\[
X_q \setminus A = X_p \setminus A, \ \leq_q|_{(X_q \setminus A)^2} \supset \leq_p|_{(X_p \setminus A)^2}
\]

Then \( X_p = X_q \) and because the elements of \( A \) in \( p \) are below all other elements of \( X_p \) we conclude \( p \leq q \). The \( P_{or} \)-property is dot synthesizable so from \( P_{or}(A) \) and \( P_{or}(r) \) follows

198
\(P_{or}(A \cdot r),\) i.e., \(p = A \cdot r \in \delta_{or}(q)\). Since \(A \subseteq M_{A \cdot r} = M_p\) then \(p \in P\) and this implication is settled.

\(\geq:\) \(p \in \delta_{or}(q)\) implies \(P_{or}(p)\) and \(p \leq q\). Since \(A \subseteq M_p\) we get \(\overline{A}^p \leq \overline{A}'\). The \(P_{or}\)-property inherits to \(\overline{A}'\), so \(\overline{A}' \in \delta_{or}(\overline{A}')\). \(\square\)

If \(A \subseteq M_q\) we from the last two lemmas see that:

\[(8.28)\quad \max(\delta_{or}(\overline{A}')) \subseteq \{\overline{A}' | p \in \max\{p \in \delta_{or}(q) | A \subseteq M_p\}\}\]

Since \(A \subseteq M_q\) we by lemma 8.5.11 also have:

\[(8.29)\quad \max(\delta_{or}(q)) = \max(\delta_{or}(q))\]

Combining (8.28) and (8.29) then:

**Corollary 8.5.14** If \(A \subseteq M_q\) and \(q \in SW\) then

\[
\max(\delta_{or}(\overline{A}')) \subseteq \{\overline{A}' | p \in \max(\delta_{or}(q))\}
\]

That the converse inclusion does not hold can be seen as follows. Let \(q = \frac{a \rightarrow b}{c \rightarrow d}\) and \(p = \frac{a \rightarrow b}{c \rightarrow d}\). Then \(p \in \max(\delta_{or}(q))\) and \(\pi' = \frac{c \rightarrow b}{d \rightarrow d}\) but \(\pi' \notin \max(\delta_{or}(\pi')) = \left\{\left\{\frac{b}{c \rightarrow d}\right\}\right\}\).

**Lemma 8.5.15** Given \(f \in \overline{L}_r\) and \(p \in SW\). Then

\(p \models_{G}^{SW} f\) implies \(\exists q \in \delta_{or}(p). q \models_{G}^{SW} f\)

**Proof** The lemma follows by proving the stronger

\(p \models_{G}^{SW} f\) implies \(\exists q \in \max(\delta_{or}(p)). q \models_{G}^{SW} f\)

by induction in the structure of \(f\).

\(f = \text{tt, ff, } \lor, \Delta:\) Either trivial or follows from \(\max(\delta_{or}(p)) = \{\varepsilon\}\) iff \(p = \varepsilon\).

\(f = \square g:\) Then \(A \subseteq M_p\) and \(p \models_{G}^{SW} g\). By hypothesis of induction there is a \(q \in \max(\delta_{or}(p))\) such that \(q \models_{G}^{SW} g\). By lemma 8.5.11 we have \(M_q = M_p\), so also \(q \models_{G}^{SW} \square g\).

\(f = \Diamond g:\) Here we must have \(A \in G, A \subseteq M_p\) and \(\overline{A}' \models_{G}^{SW} g\). Using the hypothesis of induction we get a \(q' \in \max(\delta_{or}(\overline{A}'))\) such that \(q' \models_{G}^{SW} g\). By corollary 8.5.14 then \(q' \in \{\overline{A}' | q \in \max(\delta_{or}(p))\}\). I.e., there is a \(q \in \max(\delta_{or}(p))\) such that \(\overline{A}' = q'\) and thereby \(\overline{A}' \models_{G}^{SW} g\). Because \(q \in \max(\delta_{or}(p))\) and \(A \subseteq M_p\) we can use lemma 8.5.11 to see \(A \subseteq M_q\). Hence \(q \models_{G}^{SW} \Diamond g\) and \(q \in \max(\delta_{or}(p))\) as desired.

\(\square\)
Lemma 8.5.16 Suppose \( p \in SW \) and \( P_{or}(p) \). Then there is a \( f_p \in \mathcal{L}_r \) such that:

a) \( p \models^w_{SW} f_p \)

b) \( q \models^G_{SW} f_p \) implies \( p \preceq q \)

Proof The proof is by induction on the size of \( p \) (i.e., of \( X_p \)).

In the basis \( X_p = \emptyset \) we can only have \( p = \varepsilon \). Choose \( f_p = \nabla \). By definition \( \varepsilon \models^w_{SW} \nabla \).

For the inductive step we can assume the proposition to hold for all semiwords of size less than the given \( p \). Now consider \( M_p \). Because \( P_{or}(p) \) we by proposition 8.3.7 know there is a \( a \in M_p \) such that

\[
(8.30) \quad \forall x \in M_p, x <_p y \Rightarrow a <_y y
\]

Denote \( \overline{a} \) by \( p' \). Then the size of \( p' \) is less than the size of \( p \) and we can apply the hypothesis of induction to find a \( f_{p'} \in \mathcal{L}_r \) such that \( p' \models^w_{SW} f_{p'} \) and \( q' \models^G_{SW} f_{p'} \) implies \( p' \preceq q' \). Since \( M_p \neq \emptyset \) (when \( X_p \neq \emptyset \)) we can define:

\[
f_p = \bigtriangleup \bigotimes f_{p'}, \text{ where } A = M_p
\]

Clearly \( P \models^w_{SW} f_p \). So let a \( q \) be given such that \( q \models^G_{SW} f_p \). This means \( A \subseteq M_q \) and \( q' = \overline{a} \models^G_{SW} f_{p'} \). From the hypothesis of induction we know \( p' \preceq q' \) or equally \( \overline{a} \preceq \overline{a} \).

This means by (8.24):

\[
(8.31) \quad X_q \setminus \{a\} = X_p \setminus \{a\} \quad \text{and} \quad (8.32) \quad \leq_q (X_q \setminus \{a\})^2 \supseteq \leq_p (X_p \setminus \{a\})^2
\]

From \( a \in M_p \), \( a \in M_q \) and (8.31) follows \( X_p = X_q \), so we just have to prove \( \leq_p \supseteq \leq_q \) in order to obtain \( p \preceq q \). By the reflexivity of \( \leq_p \) is suffice to show \( x <_q y \) implies \( x <_p y \) for given \( x, y \in X_q (= X_p) \). We distinguish three cases:

\( x, y \neq a \): Follows from (8.32).

\( x \neq a, y = a \): I.e., \( x <_q a \). But this contradicts \( a \in M_q \), so we can exclude this case.

\( x = a, y \neq z \): Then \( x <_q y \) reads \( a <_q y \) wherefore \( y \not\in M_q \). Because \( M_p = A \subseteq M_q \) this also implies \( y \not\in M_p \). Hence there is a \( z \in M_p \) such that \( z <_p y \). By (8.30) then \( a <_p y \) as we wish.

We can now prove the crucial lemma used in the proof of the linear logic characterization of \( \preceq_G \).

Lemma 8.5.17 For \( E_0, E_1 \in RBL' \) we have:

[Boxed mathematical expression]

Proof only if: \( E_0 \preceq_G E_1 \) implies \( [E_0]_{or} \subseteq [E_1]_{or} \) by the full abstractness result of section 8.4.

Now let \( f \in \mathcal{L}_G(E_0) \) be given. Then by lemma 8.5.9: \( \varphi(E_0 \sigma) \models^w_{SW} f \) which means there is
a \varphi(E_0) such that \varphi(E_0) \supseteq \varphi(E_1) and \varphi(E_1) \supseteq \varphi(E_2). Hence from lemma 8.5.15 \( p \models_{w} f \) for some \( p \in \delta_{or}(q) \). This means there is a \( p \in [E_0]_{or} \) with \( p \models_{w} f \). Because \( [E_0]_{or} \subseteq [E_1]_{or} \) then also \( p \in [E_1]_{or} \) and by definition of \( [\_]_{or} \) there must be a \( r \in \varphi(E_1) \) such that \( p \leq r \). So from lemma 8.5.10 \( r \models_{w} f \) and \( \varphi(E_1) \models_{w} f \). Using lemma 8.5.9 again we get \( f \in \mathcal{L}_{r}(E_1) \).

If: \( [E_0]_{or} \subseteq [E_1]_{or} \) implies \( E_0 \subseteq E_1 \) so it is enough to prove \( p \in [E_1]_{or} \) for a given \( p \in [E_0]_{or} \). Then \( P_{or}(p) \) and by the previous lemma there is \( f_{p} \in \mathcal{L}_{r} \) such that \( p \models_{w} f_{p} \) and \( q \models_{w} f_{p} \) implies \( p \leq q \). Clearly \( p \models_{w} f_{p} \) implies \( p \models_{w} f_{p} \) if \( p \in [E_0]_{or} \). Then \( \mathcal{L}_{r} \) gives \( r \models_{w} f_{p} \). \( p \models_{w} f_{p} \) means there is a \( r \in \varphi(E_0) \) such that \( p \leq r \). Lemma 8.5.10 gives \( r \models_{w} f_{p} \) because \( p \models_{w} f_{p} \), so from lemma 8.5.9 then \( f_{p} \in \mathcal{L}_{r}(E_0) \). Hence also \( f_{p} \in \mathcal{L}_{r}(E_1) \) by assumption. As above we see that there is a \( q \in \varphi(E_1) \) such that \( q \models_{w} f_{p} \). By the way \( f_{p} \) was chosen then \( p \leq q \). Because \( P_{or}(p) \) we finally have \( p \in [E_1]_{or} \).

From this proof and lemma 8.5.16 \( ( p \models_{w} f_{p} ) \) it appears that \( \mathcal{L}_{r} \) actually would suffice to characterize \( \subseteq \) and a closer look at \( f_{p} \) shows that formulae generated by:

\[(8.33)\]

\[ f ::= \nabla \mid \Box f, A \in S \mid \bigotimes f, a \in \Delta \]

would do. This can of course not come surprisingly because we from the full abstractness result already know \( \subseteq = \subseteq \) for any set of direct tests \( G (\Delta \subseteq G \subseteq S) \). It can also be seen from the ability of \( \Box \) and \( \bigotimes \) to simulate the effect of the modality \( \bigotimes \) under the satisfaction relation \( \models_{G} \) where \( A \in G \).

**Example:** Suppose \( A = \{a, c\} \) and \( A \in G \). Then

\[ E \models_{G} \bigotimes tt \iff E \models_{w} \bigotimes \bigotimes tt \]

The reason why formulae from \( \mathcal{L}_{r} \) are used in place of just those of \( (8.33) \) is as in section 7.6 because they provide more freedom in specifications. How forcible formulae can be of course depends on the available satisfaction relation.

**Example:** Suppose \( A \) and \( G \) are as in the example above. Furthermore let

\[ E_0 = (a \parallel c) ; e \text{ and } E_1 = (a \parallel c) ; a ; e \oplus a ; (a \parallel c) ; e \]

Then the formula

\[ f = \bigotimes \bigotimes \bigotimes tt \]

would be sufficient to distinguish \( E_0 \) and \( E_1 \): \( E_0 \models_{G} f \) but \( E_1 \not\models_{G} f \). If only formulae of \( (8.33) \) could be used then a formula like

\[ \bigotimes \bigotimes \bigotimes \bigotimes \nabla \]

should be used to differentiate \( E_0 \) and \( E_1 \).

Notice by the way that no formulae of \( \mathcal{L}_{G} \) can distinguish \( E_0 \) and \( E_1 \).
Striving towards more freedom in specifications it is tempting when looking at the definition for $E \models^G \exists f$ to turn the modality $\exists$ into an atomic proposition with $E \models^G \exists A \iff A \subseteq \{a \in \Delta \mid E \models^G a\}$ and add conjunction to $L_r$ (disjunction to $L_r$ respectively). However this would make the modal logic too strong which can be seen as follows.

Let $E_0 = (a \parallel c) ; a \parallel e \oplus c \parallel a ; (a \parallel e)$ and $E_1 = a ; a \parallel c ; e$. Then $E_0 \not\models^G E_0 \oplus E_1$ (can be seen from the denotations) but with $f = \Box \Diamond \Diamond \Diamond \Diamond \top \land \Diamond \Diamond \Diamond \Diamond \top$ we would have $E_0 \oplus E_1 \models^L_r f$ and $E_0 \not\models^L_r f$.

One way out would be to restrict the admissible formulae to be those where any subformula of the form $f \land g$ would have either $f \not\models^L_r f'$ or $g \not\models^L_r g'$.

We end the section with a comment regarding the logic characterization of $\not\models^G$ for the full $RBL$ language.

Without problems the formula language could be extended to include formulae like $\exists f$ where $A \in M$ and not just $A \in S$ as it is now. Taking the same definition of $\models^G$ but as if it was for $DCL$ one could similarly extend $\models^G$ to $RBL$ and obtain a logic for $RBL$. Nevertheless the logic would not be strong enough to characterize $\not\models^G$ on $RBL$: If $E_0 = a ; a \parallel a ; a$ and $E_1 = a ; a \parallel a \oplus (a \parallel a) ; (a ; a)$ then $E_0 \not\models^G E_1$ but no formula of $L_r$ (or dually $L_r$) would be able to distinguish $E_0$ and $E_1$ as can be seen by an easy case analysis. How, if possible, to characterize $\not\models^G$ on $RBL$ remains open.
Chapter 9
Adding Recursion to $BL$ and $RBL$

In this chapter we shall equip the process languages $BL$ and $RBL$ of the two preceding chapters with constructors for recursion in order to deal with infinite behaviours. The crucial constructors will be of the form $recx$. If $E$ is an expression with $x$ at some places (where an action could have been) then one can roughly think of $recx. E$ as the process which evolve like $E$ until an $x$ is meet where after it (repeatedly) can evolve like $recx. E$. Example:

$$E = recx. (a \oplus b ; x) \overset{b}{\Rightarrow} E \overset{b}{\Rightarrow} E \overset{a}{\Rightarrow} \uparrow$$

But of course $E$ could just as well evolve infinitely performing $b$'s

$$E \overset{b}{\Rightarrow} E \overset{b}{\Rightarrow} \ldots \overset{b}{\Rightarrow} E \overset{b}{\Rightarrow} \ldots$$

With our notion of (finite) maximal sequences of direct tests we would still be able to distinguish recursive $BL$ processes like:

$$recx. (a \oplus b ; x) \text{ and } recx. (c \oplus b ; x)$$

because they obviously can do different maximal sequences. On the other hand there will be no way to distinguish the processes:

(9.1) $$recx. (a ; x) \text{ and } recx. (b ; x)$$

This is satisfactory if nontermination is viewed as unimportant and only termination matters. Taking the opposite point of view, disregarding termination, they must be distinguished. One way to go would be to find some notion of infinite sequences. Against this one might argue that it breaks with the principle of finite observability: no (human) experimenter can carry out infinite sequences of direct tests. But there seems no reason to inhibit the experimenter from recording prefixes of a (possibly maximal) sequence. The preorder arising when the experimenter is endowed with this capability will be denoted $\leq$ as opposed to $\preceq$ from the previous chapters. The associated equivalence, $\equiv$, of $\leq$ will be able to distinguish the expressions of (9.1) but in return identify

$$recx. (b \oplus b ; x) \text{ and } recx. (b ; x)$$

which on the contrary would not be identified by $\equiv$—the equivalence of $\preceq$. The appropriate equivalence depends on what view is taken. However there is the serious drawback of $\leq$
that it is not a precongruence— not even on $BL$:

$$a \oplus a ; b \subseteq a ; b \text{ but } (a \oplus a ; b) ; c \not\subseteq (a ; b) ; c$$

We will therefore also be interested in $\preceq^c$— the largest precongruence contained in $\preceq$.

Recalling the operational semantics of $BL$ we know that a sequence of $E ; F$ involving actions from $F$ must contain a maximal sequence of $E$. The models from the previous chapters all to some extend mirrored maximal sequences so here we already get the clue that the previous models must be incorporated in the models which shall capture $\preceq^c$ (for the various operational $G$-semantics). For the recursive $RBL$ processes we shall similarly look for models characterizing $\preceq^c$ and $\preceq^c$.

There are standard ways of giving denotational semantics to recursive expressions and from the previous chapters we have an god idea of how the models for the preorders should look like. We will therefore in this chapter take the opposite angel and start out by constructing the infinite models and then use the finite parts of the models as link to the operational semantics.

## 9.1 General Set-up

In this section the definitions and results necessary for the remaining sections are introduced. We shall assign meaning to recursive expressions as done by Hennessy in [Hen88a]. Except for borrowing his notation and some results the section is intended to be self contained.

### 9.1.1 Denotations of Recursive Expressions

Given an infinite set, $X$, of variables and a signature, $\Sigma$, containing $\Omega$ which intuitively represent the completely undefined process. The language of recursive expressions over $\Sigma$, $REC_\Sigma(X)$, is the least set which satisfies

$$X \subseteq REC_\Sigma(X)$$

$$f(t_1, \ldots, t_k) \in REC_\Sigma(X) \text{ if } t_1, \ldots, t_n \in REC_\Sigma(X) \text{ and } f \in \Sigma \text{ is of arity } k$$

$$recx. t \in REC_\Sigma(X) \text{ if } t \in REC_\Sigma(X) \text{ and } x \in X$$

The syntactically finite expressions are denoted $FREC_\Sigma(X)$—i.e., those expressions of $REC_\Sigma(X)$ with no occurrences of $\text{rec}x$. for any $x \in X$. A variable $x$ is free in $t$ if $x$ is not within the scope of a $\text{rec}y.$ combinator where $y = x$, and an expression $t$ is called open (closed) if $t$ contains (no) free variables. The set of free variables of an expression $t$ is denoted $FV(t)$ and closed expressions of $REC_\Sigma(X)$ and $FREC_\Sigma(X)$ are denoted $REC_\Sigma$ and $FREC_\Sigma$ respectively.

A syntactic substitution, $\rho$, is a $REC_\Sigma(X)$-assignment, i.e., a map from $X$ to $REC_\Sigma(X)$, and is extended is extended to $REC_\Sigma(X)$ in the usual way, possible with renaming of bound variables to avoid clashes. $\rho[x \rightarrow t]$ denotes the substitution which maps $x$ to $t$ and otherwise is identical to $\rho$. $[t/x]$ is a shorthand for $I[x \rightarrow t]$ where $I$ is the identity substitution.

204
An expression $t$ is a approximation to $u$ if it is in the relation $\preceq$, where the syntactic preorder, $\preceq$, is defined to be the least $\Sigma$-precongruence over $REC_\Sigma(X)$ which satisfies:

$$\Omega \preceq t \quad t[rec. t/x] \preceq rec. t$$

For every $t \in REC_\Sigma(X)$, $\text{Fin}(t)$ denotes $\{t' \in FREC_\Sigma(X) \mid t' \preceq t\}$. Intuitively $\text{Fin}(t)$ is the set of syntactic finite approximations to $t$ and the meaning of $t$ can thought of as the limit of these approximations.

Having syntactic finite approximations the notion of algebraic relations can be introduced:

A relation $R$ over $REC_\Sigma$ is algebraic if for all $t, u \in REC_\Sigma$:

$$t R u \iff \forall t' \in \text{Fin}(t) \exists u' \in \text{Fin}(u). t' R u'$$

The preorder $\preceq$ enjoys the following properties ([Hen88a, page 218]):

- $\preceq$ is a partial order on $FREC_\Sigma$
- $t \preceq u$ implies $t \rho \preceq u \rho$
- $\text{Fin}(t)$ is directed w.r.t. $\preceq$

A $\Sigma$-domain, $A$, is a triple $\langle A, \leq_A, \Sigma_A \rangle$ where

- $\langle A, \leq_A \rangle$ is an algebraic complete partial order (algebraic cpo for short)
- for each $f$ in $\Sigma$ of arity $k$ there is an associated continuous function $f_A : A^k \rightarrow A$ in $\Sigma_A$.
- $\Omega_A$ is the least element $\bot_A$ of $\langle A, \leq_A \rangle$

We shall use $\text{Fin}(A)$ to denote the compact elements of $A$.

Given a $\Sigma$-domain, $A$, the expressions of $REC_\Sigma(X)$ are assigned a meaning using environments over $A$. An environment is an $A$-assignment, $\rho_A$, i.e., a map from $X$ to $A$, and similar as for syntactic substitution $\rho_A[a/x]$ is the $A$-assignment which maps $x$ to $a$ and otherwise equals $\rho_A$. The set of all environments, $(X \rightarrow A)$, is denoted $\text{ENV}_A$. Two $A$-environments $\rho$ and $\rho'$ from $\text{ENV}_A$ are ordered by the induced pointwise ordering:

$$\rho \preceq \rho' \iff \forall x \in X. \rho(x) \leq_A \rho'(x)$$

**Proposition 9.1.1** $\text{ENV}_A$ is an algebraic complete partial order and the compact elements of $\text{ENV}_A$ are those $\rho_A$ where there exists a finite subset $Y$ of $X$ such that

a) $\forall x \in X \setminus Y. \rho_A(x) = \bot_A$

b) $\forall x \in Y. \rho_A(x) \in \text{Fin}(A)$

205
Notice that \( \perp_A \) is a compact elements so \( \rho_A \in \text{Fin}(\text{ENV}_A) \) actually implies \( \rho(x) \in \text{Fin}(A) \) for every \( x \in X \).

**Proof** For convenience we will in this proof use \( f, g, \ldots \) for elements of \( \text{ENV}_A \). By Hennessy [Hen88a, page 123] \( (X \longrightarrow A) \) is a complete partial order. We show that the compact elements of \( \text{ENV}_A \) are as described above. It is then a simple matter to check that every element is the lub of the compact elements below it.

For every compact element \( f \) there is a finite \( Y \subseteq X \) fulfilling a) and b): Let \( Y = \{ x \in X \mid f(x) \neq \perp_A \} \). To see that \( Y \) is a finite set assume on the contrary it is infinite. Then \( Y \) contains a countable infinite subset \( Z = \{ z_i \}_{i \in \mathbb{N}} \). For each \( i \in \mathbb{N} \) define \( f_i \in \text{ENV}_A \) by

\[
f_i(x) = \begin{cases} f(x) & \text{if } x \notin Z \\ f(z_j) & \text{if } x = z_j \text{ and } j < i \\ \perp_A & \text{otherwise} \end{cases}
\]

Clearly \( D = \{ f_i \}_{i \in \mathbb{N}} \) is a chain \( f_0 \leq_A f_1 \leq_A \ldots \) with lub \( f \). Because \( f \leq f \) and \( f \) is compact there is an \( f_i \in D \) such that \( f \leq f_i \), so \( \forall x \in X. f(x) \leq_A f_i(x) \) and especially \( f(z_i) \leq f_i(z_i) = \perp_A \). Hence \( f(z_i) = \perp_A \) — a contradiction to the definition of \( Z \) \( (z \in Z \subseteq Y \text{ only if } f(z) \neq \perp_A) \).

By definition \( Y \) fulfills a) and to see b) let an \( y \in Y \) be given. We shall show \( f(y) \in \text{Fin}(A) \). To this end let \( D_A \) be a directed set in \( A \) such that \( f(y) \leq_A \bigvee_A D_A \). Then we shall find a \( d \in D_A \) such that \( f(y) \leq_A d \). For every \( a \in D_A \) define \( f_a \) by

\[
f_a(x) = \begin{cases} f(x) & \text{if } x \neq y \\ a & \text{if } x = y \end{cases}
\]

and let \( D \) be \( \{ f_a \mid a \in D_A \} \). \( D \) is directed because \( D_A \) is and \( D \) has lub \( f_D \) in \( \text{ENV}_A \) where

\[
f_D(x) = \begin{cases} f(x) & \text{if } x \neq y \\ \bigvee_A D_A & \text{if } x = y \end{cases}
\]

So \( f \leq f_D \) and because \( f \) is compact there is an \( f_d \in D \) such that \( f \leq f_d \). Then also \( f(y) \leq f_d(y) = d \in D_A \).

That \( f \) is a compact element when there is a finite \( Y \subseteq X \) fulfilling a) and b) is easier to see: Given a directed set \( D \) in \( \text{ENV}_A \) such that \( f \leq f_D \) where \( f_D \) is the lub of \( D \) in \( \text{ENV}_A \). We shall find a \( g \in D \) such that \( f \leq g \). \( f \leq f_D \) implies \( \forall x \in X. f(x) \leq_A \bigvee_A \{ g(x) \mid g \in D \} \). Since \( \forall y \in Y. f(y) \in \text{Fin}(A) \) we then have \( \forall y \in Y \exists g \in D. f(y) \leq_A g(y) \). For each \( y \in Y \) let \( g_y \) be such an \( g \). Denote \( \{ g_y \mid y \in Y \} \) by \( G_Y \). Since \( y \) is finite then so is \( G_Y \) and because \( D \) is directed we can then find a \( g \in D \) such that \( \forall g_y \in G_Y. g_y \leq g \). Hence \( \forall y \in Y. f(y) \leq_A g(y) \). Because \( \forall x \in X \setminus Y. f(x) = \perp_A \) we actually have \( \forall z \in X \setminus Y. f(x) \leq_A g(x) \) and we conclude \( f \leq g \).

Since \( \text{ENV}_A \) is a cpo, meanings of expressions can now be given by means of the function: \( A[\_] : \text{REC}_\Sigma(X) \longrightarrow \text{[ENV}_A \longrightarrow A] \) defined as follows:

\[
A[x] \rho_A = \rho_A(x) \\
A[f(t_1, \ldots, t_k)] \rho_A = f_A(A[t_1] \rho_A, \ldots, A[t_k] \rho_A) \\
A[\text{rec } t \_] \rho_A = Y \lambda a. A[t] \rho_A[a/x]
\]
where $Y$ is a function that yields the least fixpoint of $\lambda a. A[t]_{\rho_A}[a/x]$ in $A$ and $[ENV_A \rightarrow A]$ is the continuous functions of $(ENV_A \rightarrow A)$.

We select some of the results Hennessy displays:

**Proposition 9.1.2** $A[\text{rec} x. t]_{\rho_A} = A[t[\text{rec} x. t/x]]_{\rho_A}$ for all $\rho_A \in \text{ENV}_A$.

With this proposition it is easy to see for $t, u \in \text{REC}_\Sigma(X)$ that:

$t \leq u$ implies $\forall \rho_A \in \text{ENV}_A. A[t]_{\rho_A} \leq_A [u]_{\rho_A}$

I.e., the preorder defined as on the right-hand side extends $\leq$ on $\text{REC}_\Sigma(X)$.

**Theorem 9.1.3** (finite approximations) For every $t \in \text{REC}_\Sigma(X)$ and $\rho_A \in \text{ENV}_A$ the following holds: $A[t]_{\rho_A} = \bigvee_A A[\text{Fin}(t)]_{\rho_A} := \bigvee_A \{A[t']_{\rho_A} | t' \in \text{Fin}(t)\}$.

**Lemma 9.1.4** If $\rho_A, \rho'_A \in \text{ENV}_A$ and $\rho_A \leq_{FV(t)} \rho'_A$ then $A[t]_{\rho_A} \leq_A A[t]_{\rho'_A}$

From this lemma it follows the value of $A[t]_{\rho_A}$ for a $t \in \text{REC}_\Sigma$ (the processes of $\text{REC}_\Sigma(X)$) is independent of $\rho_A$ ($FV(t) = \emptyset$) and this value can be taken as the meaning of $t$. So $A[t]_{\rho_A}$ can be thought of as defining a map $\text{REC}_\Sigma \rightarrow A$ and we will therefore just write $A[t]$ when $t$ is a closed expression.

**Lemma 9.1.5** (Substitution Lemma) $A[t_{\rho}]_{\rho_A} = A[t]_{\rho}(\rho_A \circ \rho)$

where the composition of the $A$-assignment $\rho_A$ and the substitution $\rho$ is the $A$-assignment: $(\rho_A \circ \rho)(x) = A[\rho(x)]_{\rho_A}$.

We are now ready to reflect on extending preorders from closed to open expressions.

A preorder, ambiguously denoted $\leq_A$, over $\text{REC}_\Sigma$ can be induced from the partial order, $\leq_A$, of the $\Sigma$-domain by letting for $t, t' \in \text{REC}_\Sigma$:

$t \leq_A t'$ iff $\forall \rho_A \in \text{ENV}_A. A[t]_{\rho_A} \leq_A A[t']_{\rho_A}$

Since $Y$ and the functions of a $\Sigma$-domain are continuous and especially monotone it is self-evident from the definition of $A[\ ]$ that $\leq_A$ is a $\text{REC}_\Sigma$-precongruence in the sense that for closed expressions:

- for all $f$ in $\Sigma$, $f(t_1, \ldots, t_k) \leq_A f(u_1, \ldots, u_k)$ whenever $t_i \leq_A u_i$ for $i = 1, \ldots, k$
- $t \leq_A u$ implies $\text{rec} x. t \leq_A \text{rec} x. u$

The latter actually does not tell us anything because $A[\text{rec} x. t]_{\rho_A} = (\text{proposition 9.1.2} A[t[\text{rec} x. t/x]]_{\rho_A} = (\text{substitution lemma}) A[t]_{\rho} \circ I[\text{rec} x. t/x]$ which from lemma 9.1.4 and $FV(t) = \emptyset$ is seen to equal $A[t]_{\rho_A}$.

This is the motivation for extending the preorder to $\text{REC}_\Sigma(X)$. There is at least two ways to do this. For $t, u \in \text{REC}_\Sigma(X)$ define:
a) \( t \leq_A u \iff \forall \rho_A \in \text{ENV}_A. A[t] \rho_A \leq_A A[u] \rho_A \)

b) \( t \leq'_A u \iff \) for every closed (syntactic) substitutions \( \rho \), \( A[t \rho] \leq_A A[u \rho] \)

Notice that for closed substitutions \( t \rho \in \text{REC}_\Sigma \).

With similar arguments as above it is now easy to see that \( \leq_A \) is a \( \text{REC}_\Sigma(X) \)-precongruence and one might argue that it is the most natural extension in a denotational set-up whereas the other, \( \leq'_A \), is more natural in an operational set-up. However we will now show that under certain circumstances \( \leq_A \) and \( \leq'_A \) coincide not only on \( \text{REC}_\Sigma \) but also on \( \text{REC}_\Sigma(X) \).

We can then state:

**Proposition 9.1.6** \( \leq_A \) and \( \leq'_A \) defined above coincide over \( \text{REC}_\Sigma(X) \) provided there for every compact element of \( A \) is a \( t \in \text{FREC}_\Sigma \) such that \( A[t] = a \).

**Proof** Let \( t, t' \in \text{REC}_\Sigma(X) \) be given. Suppose that for all closed substitutions \( \rho \), \( A[t \rho] \leq_A A[t' \rho] \). We show this implies \( A[t] \leq_A A[t'] \)—i.e., \( \forall \sigma_A \in \text{ENV}_A. A[t] \sigma_A \leq_A A[t'] \sigma_A \). So let a \( \sigma_A \in \text{ENV}_A \) be given. From the proposition \( 9.1.1 \) \( \text{ENV}_A \) is algebraic because \( A \) is. This means \( \sigma_A = \bigvee F \) where \( F \) is the directed set consisting of the compact elements in \( \text{ENV}_A \) below \( \sigma_A \). From the proposition we know that for a compact element \( \rho_A \in \text{ENV}_A \) we have \( \rho_A(x) \) is compact in \( A \) for all \( x \in X \). By the proviso of the proposition there then is a \( t_x \in \text{FREC}_\Sigma \) for each \( x \in X \) such that \( A[t_x] = \rho_A(x) \).

Letting \( \rho \) be the closed syntactic substitution with \( \rho(x) = t_x \) for all \( x \in X \) we then have \( \rho_A \circ \rho = \rho_A \). I.e., for each \( \rho_A \in F \) there is a closed substitution \( \rho \) with \( \rho_A \circ \rho = \rho_A \). Our assumption was that for all closed substitutions \( \rho' \), \( A[t \rho'] \leq_A A[t' \rho'] \) \( (t \rho' \text{ and } t' \rho' \text{ are closed}) \), so especially \( A[t \rho] \rho_A \leq_A A[t' \rho] \rho_A \) for each \( \rho_A \in F \). By the substitution lemma then \( A[t](\rho_A \circ \rho) \leq_A A[t'](\rho_A \circ \rho) \) for each \( \rho_A \in F \), and since \( \rho_A \circ \rho = \rho_A \) this actually reads:

\[
\forall \rho_A \in F. A[t] \rho_A \leq_A A[t'] \rho_A
\]

From this the result then follows by the deduction

\[
\downarrow \{A[t] \rho_A \mid \rho_A \in F\} \text{ dominated by } \{A[t'] \rho_A \mid \rho_A \in F\}
\downarrow \forall_A \{A[t] \rho_A \mid \rho_A \in F\} \leq_A \forall_A \{A[t'] \rho_A \mid \rho_A \in F\}
\downarrow A[t] \text{ and } A[t'] \text{ continuous } (\in [\text{ENV}_A \rightarrow A])
\downarrow A[t](\bigvee F) \leq_A A[t'](\bigvee F)
\downarrow A[t] \sigma_A \leq_A A[t'] \sigma_A
\]

It remains to show the other direction \( t \leq_A t' \Rightarrow t \leq'_A t' \).

Suppose \( A[t] \rho_A \leq_A A[t'] \rho_A \) for all \( \rho_A \in \text{ENV}_A \). Given a closed substitution \( \rho \) and a \( \rho_A \in \text{ENV}_A \) we show \( A[t] \rho_A \leq_A A[t \rho] \rho_A \). The substitution lemma directly gives us \( A[t \rho] \rho_A = A[t](\rho_A \circ \rho) \) and similar for \( t' \) so because \( \rho_A \circ \rho \) just is another \( A \)-assignment \( (\in \text{ENV}_A) \) we are done. \( \square \)
This result is closely related with the notion of substitutive relations as presented by Hennessy in [Hen83]:

A relation \( R \) over \( \text{REC}_\Sigma(X) \) is substitutive if for all \( t, u \in \text{REC}_\Sigma(X) \):

\[
t R u \iff \text{for all closed syntactic substitutions } \rho, t \rho R u \rho
\]

There Hennessy actually indicate the proposition above referring to [DNH84].

**Proposition 9.1.7** When restricted to \( \text{REC}_\Sigma \) the preorder \( \leq_A \) is algebraic provided for all \( t \in \text{FREC}_\Sigma \), \( A[t] \) is a compact element of \( A \).

**Proof** We show for given \( t, u \in \text{REC}_\Sigma \):

\[
A[t] \leq_A A[u] \iff \forall t' \in \text{Fin}(t) \exists u' \in \text{Fin}(u). A[t'] \leq_A A[u']
\]

Each implication is proven separately.

*if:* If we show that \( A[u] \) is a lub for \( A[\text{Fin}(t)] \) the implication follows because \( A[t] \) by theorem 9.1.3 is a lub for \( A[\text{Fin}(t)] \). Let an arbitrary element \( a \in A[\text{Fin}(t)] \) be given. This means there is a \( t' \in \text{Fin}(t) \) with \( A[t'] = a \). By the antecedent of the implication there is a \( u' \in \text{Fin}(u) \) such that \( A[t'] \leq_A A[u'] \). \( u' \in \text{Fin}(u) \) only if \( u' \leq u \) so \( A[u'] \leq_A A[u] \) and by the transitivity of \( \leq_A \) then \( a \leq_A A[u] \).

*only if:* Assume \( A[t] \leq A[u] \) and let a \( t' \in \text{Fin}(t) \) be given. As above this implies \( A[t'] \leq A[t] \) and therefore also \( A[t'] \leq A[u] \). \( A[u] \) is also the lub for \( A[\text{Fin}(u)] \) so actually \( A[t'] \leq_A \forall_A A[\text{Fin}(u)] \). By the proviso of the proposition \( A[t'] \in \text{Fin}(A) \). I.e., \( A[t'] \) is compact wherefore \( A[t'] \leq_A a \) for some \( a \in A[\text{Fin}(u)] \) or equally \( A[t'] \leq_A A[u'] \) for some \( u' \in \text{Fin}(u) \).

A \( \Sigma \)-domain is finitary if the map \( A[\cdot] : \text{REC}_\Sigma \rightarrow A \) when restricted to \( \text{FREC}_\Sigma \) is surjective onto \( \text{Fin}(A) \).

**Corollary 9.1.8** If a \( \Sigma \)-domain, \( A \), is finitary then the preorder over \( \text{REC}_\Sigma(X) \) is substitutive and when restricted to \( \text{REC}_\Sigma \) it is algebraic.

### 9.1.2 Contexts

When considering a language a context, \( C[\cdot] \), is normally thought of as an expression with zero or more “holes”, to be filled by some other expression of the language. Strictly speaking \( C[\cdot] \) is not an expression of the language, but if we think of a “hole” as a special constant symbol, a context will be an expression of the language extended with this constant. We illustrate the idea on the language of recursive expressions, \( \text{REC}_\Sigma(X) \), built over the signature \( \Sigma \).

We let the special constant symbol \# (assumed not to be in \( \Sigma \)) take the rôle of a “hole”. The set of \( \text{REC}_\Sigma(X) \)-contexts is then simply \( \text{REC}_{\Sigma \cup \{\#\}}(X) \), written \( \text{REC}_{\Sigma\#}(X) \) for short. Notice \( \text{REC}_\Sigma \subseteq \text{REC}_{\Sigma\#} \). The context \( C \in \text{REC}_{\Sigma\#}(X) \) filled with (the context) \( t \in \text{REC}_{\Sigma\#}(X) \), is denoted \( C[t] \) and defined by structural induction:
\[ f[t] = \begin{cases} t & \text{if } f = \# \\ f & \text{otherwise (any other constant)} \end{cases} \]

\[ x[t] = x \]

\[ f(C_1, \ldots, C_k)[t] = f(C_1[t], \ldots, C_k[t]) \text{ for every } f \in \Sigma \text{ of arity } k \geq 1. \]

\[ (\text{rec } x. C)[t] = \text{rec } x. (C[t]) \]

Notice that, as opposed to syntactic substitution, free variables of \( t \) may become bound when filled in to \( C \). Also observe that if \( t \in \text{REC}_\Sigma(X) \) then \( C[t] \in \text{REC}_\Sigma(X) \).

The advantage of considering contexts as ordinary expressions of an enlarged language is that it allows us to use the syntactic precongruence \( \preceq \) on contexts just as we do on ordinary expressions. Recall that \( \preceq \) is defined to be the least \( \Sigma \)-precongruence over \( \text{REC}_\Sigma(X) \) which satisfy

\[ \Omega \preceq t \]

\[ t[\text{rec } x. t/x] \preceq \text{rec } x. t \]

Clearly the least \( \Sigma \)-precongruence over \( \text{REC}_\Sigma#(X) \) which satisfies the two rules above will agree with \( \preceq \) on \( \text{REC}_\Sigma \) so for convenience we shall make no distinction between them.

**Lemma 9.1.9** If \( C \) and \( C' \) are \( \text{FREC}_\Sigma(X) \)-contexts and \( t, u \) are \( \text{REC}_\Sigma(X) \)-contexts then

a) \( t \preceq u \) implies \( C[t] \preceq C[u] \)

b) \( C \preceq C' \) implies \( C[t] \preceq C'[t] \)

Since \( \text{REC}_\Sigma(X) \subseteq \text{REC}_\Sigma#(X) \) the lemma of course applies for \( t, u \in \text{REC}_\Sigma(X) \) (or \( t, u \in \text{FREC}_\Sigma(X) \)) too.

**Proof** a) By induction on the structure of \( C \).

- \( C = f \neq \# \) or \( C = x \): Here we have \( C[t] = C = C[u] \).
- \( C = \# \): Then \( C[t] = t \preceq u = C[u] \).
- \( C = f(C_1, \ldots, C_k) \): \( C[t] = f(C_1[t], \ldots, C_k[t]) \) definition of \( _/[t] \)
  \[ \preceq f(C_1[u], \ldots, C_k[u]) \] hypothesis and definition of \( \preceq \)
  \[ = C[u] \]

b) Induction on the length of the proof of \( C \preceq C' \). For the basis either \( C = \Omega \) or \( C = C' \). The latter case is trivial and in the former we have \( C[t] = \Omega[t] = \Omega \preceq C'[t] \). In the inductive step \( C \preceq C' \) can because \( C' \in \text{FREC}_\Sigma# \) only mean \( C = f(C_1, \ldots, C_k) \preceq f(C'_1, \ldots, C'_k) = C \) where \( C_i \preceq C'_i \) for \( i = 1, \ldots, k \). The result then follows similar as in a). \( \square \)

**Lemma 9.1.10** Suppose \( C \) is a \( \text{FREC}_\Sigma(X) \)-context and \( t \in \text{REC}_\Sigma(X) \). Then \( u \in \text{Fin}(C[t]) \) implies there is a \( \text{FREC}_\Sigma(X) \)-context \( C' \preceq C \) and a \( t' \in \text{Fin}(t) \) such that \( u \preceq C'[t'] \).

**Proof** By induction on the structure of \( C \). Recall \( u \in \text{Fin}(C[t]) \) means \( u \preceq C[t] \) and \( u \in \text{FREC}_\Sigma(X) \).
\[ C = f \neq \# \text{ or } C = x: \] Then \( C[t] = C \) and \( u \) equals \( \Omega \) of \( C \). Letting \( C' = u \) and \( t' = \Omega \in \text{Fin}(t) \) we have \( u = C' = C'[\Omega] = C'[t'] \).

\( C = \# \): I.e., \( u \in \text{Fin}(C[t]) = \text{Fin}(t) \). Choose \( C' = \# \) and \( t' = u \in \text{Fin}(t) \). Then \( u = t' = \#[t'] = C'[t'] \).

\[ C = f(C_1, \ldots, C_k) \text{ for an } f \in \Sigma: \] Here we have \( C[t] = f(C_1[t], \ldots, C_k[t]) \) so inspecting the definition of \( \preceq \) we see \( u \preceq C[t] \) implies \( u = f(u_1, \ldots, u_k) \) where \( u_i \preceq C_i[t] \) for \( i = 1, \ldots, k \). By hypothesis of induction there for each \( i = 1, \ldots, k \) is a \( t'_i \in \text{Fin}(t) \) and a \( \text{FREC\}_\Sigma(X) \)-context \( C'_i \preceq C_i \) such that \( u_i \preceq C'_i[t'_i] \). Since \( \text{Fin}(t) \) is directed there is \( \cup b t' \in \text{Fin}(t) \) for \( t'_1, \ldots, t'_k \). By lemma 9.1.9 then \( u_i \preceq C'_i[t'] \) for each \( i \) and because \( \preceq \) is a \( \Sigma \)-precongruence we then have \( f(u_1, \ldots, u_k) \preceq f(C'_1[t'], \ldots, C'_k[t']) \). Letting \( C' = f(C'_1, \ldots, C'_k) \), \( C' \) is then a \( \text{FREC\}_\Sigma(X) \)-context with \( C' \preceq C \) and \( u \preceq C'[t'] \).

\( \square \)

### 9.1.3 \( \Sigma \)-precongruences

Suppose \( \mathcal{L}_{\Sigma'} \) is a language constructed from a signature \( \Sigma' \). Given a preorder, \( \sqsubseteq \), over \( \mathcal{L}_{\Sigma'} \) and a \( \Sigma \subseteq \Sigma' \) we denote the largest \( \Sigma \)-precongruence over \( \mathcal{L}_{\Sigma'} \) contained in \( \sqsubseteq \) by \( \sqsubseteq^\Sigma \). I.e.,

a) \( \sqsubseteq^\Sigma \subseteq \sqsubseteq \)

b) \( \sqsubseteq^\Sigma \) is a \( \Sigma \)-precongruence

c) \( \sqsubseteq' \subseteq \sqsubseteq^\Sigma \) for any other \( \Sigma \)-precongruence, \( \sqsubseteq' \), contained in \( \sqsubseteq \)

Now define \( \sqsubseteq^\Sigma \# \subseteq \mathcal{L}_{\Sigma'} \times \mathcal{L}_{\Sigma'} \) by

\[ t \sqsubseteq^\Sigma \# u \text{ iff } \forall \mathcal{L}_{\Sigma'}-contexts } C. C[t] \sqsubseteq C[u] \]

**Proposition 9.1.11** \( \sqsubseteq^\Sigma \# = \sqsubseteq^\Sigma \), i.e., \( \sqsubseteq^\Sigma \# \) is the largest \( \Sigma \)-precongruence contained in \( \sqsubseteq \).

**Proof** We show that \( \sqsubseteq^\Sigma \# \) fulfills a)—c).

a) Assume \( t \sqsubseteq^\Sigma \# u \). With \( C = \# \) we especially have \( t = C[t] \sqsubseteq C[u] = u \).

b) Suppose \( f \in \Sigma \) of arity \( k \) and assume \( t_i \sqsubseteq^\Sigma \# u_i \) for \( i = 1, \ldots, k \). Let a \( \mathcal{L}_{\Sigma'}-context } C \) be given. We shall show \( C[f(t_1, \ldots, t_k)] \sqsubseteq C[f(u_1, \ldots, u_k)] \). For \( i = 1, \ldots, k \) define \( C_i \) to be the \( \mathcal{L}_{\Sigma'}-context } C[f(u_1, \ldots, u_{i-1}, \#, t_{i+1}, \ldots, t_k)] \). An easy induction on the structure of \( C \) shows \( C_i[t_i] = C[f(t_1, \ldots, t_k)], C_k[u_k] = C[f(u_1, \ldots, u_k)] \) and \( C_i[u_i] = C_{i+1}[t_{i+1}] \) for \( i = 1, \ldots, k - 1 \). By assumption \( C_i[t_i] \sqsubseteq C_i[u_i] \) for \( i = 1, \ldots, k \). The result then follows by the transitivity of \( \sqsubseteq \).

c) Given another \( \Sigma \)-precongruence \( \sqsubseteq' \subseteq \sqsubseteq \) suppose \( t \sqsubseteq' u \). For every \( \mathcal{L}_{\Sigma'}-context } C \) it is easy to show \( C[t] \sqsubseteq' C[u] \) by induction on the structure of \( C \) using the fact that \( \sqsubseteq' \) is a \( \Sigma \)-precongruence. Since \( \sqsubseteq' \subseteq \sqsubseteq \) we by definition of \( \sqsubseteq^\Sigma \# \) then have \( t \sqsubseteq^\Sigma \# u \). \( \square \)
With this lemma we easily get

**Proposition 9.1.12** Let a preorder, \( \sqsubseteq \), over \( \mathcal{L}_\Sigma \) be given together with signatures \( \Sigma^1 \subseteq \Sigma^2 \subseteq \Sigma' \). Assume \( \sqsubseteq^{\Sigma^1} \) agrees on \( \mathcal{L}_\Sigma \) with another preorder, \( \sqsubseteq' \), which is a \( \Sigma^2 \)-pre congruence. Then \( \sqsubseteq^{\Sigma^1} \) equals \( \sqsubseteq^{\Sigma^2} \).

**Proof** \( \sqsubseteq^{\Sigma^2} \subseteq \sqsubseteq^{\Sigma^1} \) by definition. To see the opposite inclusion assume \( t \sqsubseteq^{\Sigma^1} u \). By the previous proposition 9.1.11 is enough to show \( C[t] \sqsubseteq C[u] \) for every \( \mathcal{L}_{\Sigma^2} \)-context \( C \). So let an arbitrary \( \mathcal{L}_{\Sigma^2} \)-context \( C \) be given. By the assumption of the lemma \( t \sqsubseteq^{\Sigma^1} u \) implies \( t \sqsubseteq' u \). Since \( \sqsubseteq' \) is a precongruence w.r.t. to the combinator set \( \Sigma^2 \) we can then by induction on the structure of \( C \) show \( C[t] \sqsubseteq' C[u] \). Then again by the assumption of the lemma \( C[t] \sqsubseteq^{\Sigma^1} C[u] \). Because \( \sqsubseteq^{\Sigma^1} \) by definition is contained in \( \sqsubseteq \) we actually have \( C[t] \sqsubseteq C[u] \) as desired. \( \square \)

Before proceeding with the useful theorem below we need a definition:

**Definition 9.1.13** Given a preorder, \( \sqsubseteq \), over a language \( \mathcal{L} \) and a subset \( A \subseteq \mathcal{L} \), \( \mathcal{L} \) is said to be \( A \)-expressive w.r.t. \( \sqsubseteq \) iff for every \( t \in \mathcal{L} \) there exists a characteristic context \( C_t[\cdot] \) such that

\[
\forall u \in A \ t \sqsubseteq u \mbox{ iff } C_t[t] \sqsubseteq C_t[u]
\]

where \( \sqsubseteq^c \) is the largest precongruence w.r.t. to the combinator set \( \mathcal{L} \) contained in \( \sqsubseteq \). If \( A = \mathcal{L} \) then \( \mathcal{L} \) is simply said to be expressive w.r.t. \( \sqsubseteq \).

**Theorem 9.1.14** Let \( \sqsubseteq \) be an algebraic preorder over \( REC_\Sigma \) containing the syntactic preorder \( \preceq \). If \( FREC_\Sigma \) is Fin\((t)\)-expressive w.r.t. \( \sqsubseteq \) (restricted to \( FREC_\Sigma \)) for every \( t \in REC_\Sigma \), then \( \sqsubseteq \) is algebraic too.

**Proof** Given \( t, u \in FREC_\Sigma \) we show

\[
t \sqsubseteq u \Downarrow\supseteq\forall t' \in \text{Fin}(t) \exists u' \in \text{Fin}(u). t' \sqsubseteq u'
\]

\( \uparrow \): Assume \( \forall t' \in \text{Fin}(t) \exists u' \in \text{Fin}(u). t' \sqsubseteq u' \). In order to have \( t \sqsubseteq u \) it is by proposition 9.1.11 enough to show \( C[t] \sqsubseteq C[u] \) for any given \( FREC_\Sigma \)-context \( C \). So suppose \( C \) is such a context. Let a \( t'' \in \text{Fin}(C[t]) \) be given. By lemma 9.1.10 there is a \( FREC_\Sigma \)-context \( C' \subseteq C \) and a \( t' \in \text{Fin}(t) \) such that \( t'' \preceq C'[t'] \). By assumption there is a \( u' \in \text{Fin}(u) \) with \( t' \sqsubseteq u' \) and so also \( C'[t'] \subseteq C'[u'] \) according to proposition 9.1.11. Clearly \( C'[t'] \in FREC_\Sigma \) and from \( u' \preceq u \) it follows by lemma 9.1.9 that \( C'[u'] \preceq C[u] \preceq C[u] \) so we actually have \( C'[u] \in \text{Fin}(C[u]) \). \( \preceq \subseteq \subseteq \) and the transitivity of \( \sqsubseteq \) gives \( t'' \subseteq C'[u] \). Hence for every \( t'' \in \text{Fin}(C[t]) \) we have found a \( u'' \in \text{Fin}(C[u]) \) such that \( t'' \subseteq u'' \). Because \( \sqsubseteq \) is algebraic this implies \( C[t] \subseteq C[u] \) as we wanted.

\( \Downarrow \): Assume \( t \sqsubseteq u \) and let a \( t' \in \text{Fin}(t) \) be given. We shall find a \( u' \in \text{Fin}(u) \) such that \( t' \sqsubseteq u' \). Since \( t' \in FREC_\Sigma \) and \( FREC_\Sigma \) is Fin\((t)\)-expressive there (for this \( t' \)) is a \( FREC_\Sigma \)-context, \( C \), such that for all \( u' \in \text{Fin}(u) \)

\[
C[t'] \sqsubseteq C[u'] \mbox{ iff } t' \sqsubseteq u'
\]

212
Let $C$ be such a characteristic context for $t'$. We just have to find a $u' \in \text{Fin}(u)$ such that $C[t'] \subseteq C[u']$. Since $t' \leq t$ we by lemma 9.1.9 have $C[t'] \preceq C[t]$ and because $C$ is a $\text{FREC}_\Sigma$-context this implies $C[t'] \in \text{Fin}(C[t])$. From $t \sqsubseteq \Sigma$ we by proposition 9.1.11 especially have $C[t] \sqsubseteq C[u]$ and by the algebraicity of $\sqsubseteq$ we deduce there must be a $u'' \in \text{Fin}(C[u])$ such that $C[t'] \subseteq u''$. Using lemma 9.1.10 we find a $C' \preceq C$ and a $u' \in \text{Fin}(u)$ with $u'' \preceq C'[u']$. By lemma 9.1.9 $u'' \preceq C'[u'] \preceq C[u']$ and from $\leq \subseteq \sqsubseteq$ and transitivity of $\sqsubseteq$ we obtain $C[t'] \sqsubseteq C[u']$ as desired. \hfill $\square$

**Proposition 9.1.15** Given a preorder $\sqsubseteq$ over $\text{REC}_\Sigma$ extended to the open terms of $\text{REC}_\Sigma(X)$ in the substitutive way. Suppose $\Sigma \subseteq \Sigma'$ and $\sqsubseteq$ identifies expressions equal up to rename of bound variables. Then $\sqsubseteq^\Sigma$ is substitutive.

**Proof** Given $t$ and $u$ we show

$$ t \sqsubseteq^\Sigma u \iff \forall \rho \text{ (closed). } t\rho \sqsubseteq^\Sigma u\rho $$

only if: For a particular $\rho$ it will by proposition 9.1.11 do to show that for any closed $\rho'$ and $\Sigma$-context, $C$, we have $(C[t\rho])\rho' \subseteq (C[u\rho])\rho'$ (C might contain free variables). Let such a context and syntactic substitutions be given. Because $FV(t\rho) = FV(u\rho) = \emptyset$ there is a closed context $C'$ such that

$$ C'[t\rho] = (C[t\rho])\rho' \quad \text{and} \quad C'[u\rho] = (C[u\rho])\rho' $$

Further more $C'$ must be incapable of binding variables since it steems from the $\Sigma$-context $C$.

Now $t \sqsubseteq^\Sigma u$ implies $\forall \rho \text{ (closed) } \forall \Sigma$-contexts $C'$. $(C'[t])\rho \subseteq (C'[u])\rho$. Since $C'$ above is closed and $C'$ cannot bind any variables, $(C'[t])\rho$ must equal $C'[t\rho]$ under $\sqsubseteq$; similar for $u$. The result then follows.

if: Assume $t\rho \sqsubseteq^\Sigma u\rho$ for all closed $\rho$. Given a context, $C$, and some closed syntactic substitution, $\rho'$, we show $C[t]\rho' \sqsubseteq C[u]\rho'$. Since $C$ does not bind variables we must have $(C[t\rho'])\rho' = (C[t])\rho'$ and similar for $u$. As a particular case of the assumption we have $(C[t\rho'])\rho' \subseteq (C[u\rho'])\rho'$ and are then done. \hfill $\square$

### 9.1.4 Obtaining Algebraic Complete Partial Orders

The algebraic cpos we are after can be obtained in a uniform way, so the construction of them will be presented here in a more general set-up.

Suppose $(P, \sqsubseteq)$ is a preordered set and $\phi$ is a function $\phi : P \rightarrow \mathcal{P}(P) \setminus \emptyset$ which is extended to $\mathcal{P}(P) \rightarrow \mathcal{P}(P)$ in the natural way: $\phi(s) = \bigcup_{p \in s} \phi(p)$ for every $s \subseteq P$ ($\phi(\emptyset) = \emptyset$).

Also let there be given a collection, $\Pi$, of subsets of $P$—i.e., $\Pi \subseteq \mathcal{P}(P)$. About $\Pi$ it is assumed that

- if $s \in \Pi$ and $p \in s$ then also $\{p\} \in \Pi$
- every nonempty $S \subseteq \Pi$ has lub w.r.t. $\subseteq$: $\bigcup S$ in $\Pi$—i.e $\bigcup S = \bigcup_{s \in S} s$ (\Pi closed under union)

213
there is an element \( s_{\Pi} \in \Pi \) with \( \phi(s_{\Pi}) \subseteq \phi(s) \) for every \( s \in \Pi \)

Furthermore we shall assume that \( \phi \) and \( \sqsubseteq \) are interrelated such that for every \( p \in s \) and \( s \in \Pi \):

a) \( \{ q \in P \mid q \subseteq p \} \) is finite

b) \( q \in \phi(p) \) implies \( q \subseteq p \)

c) \( \exists q \in \phi(p). \ p \sqsubseteq q \)

Finally define:

\[
\Phi \subseteq \mathcal{P}(P) \text{ to be the set } \{ \phi(s) \in \mathcal{P}(P) \mid s \in \Pi \}
\]

In order to facilitate the overview it is intended to make use of symbols such that

\[
p, q, \ldots \in P \quad s, t, \ldots \in \Pi \quad \text{and } S, T, \ldots \subseteq \Pi \quad a, b, \ldots \in \Phi \quad \text{and } A, B, \ldots \subseteq \Phi
\]

The idea is now to make \( \Phi \) into an algebraic cpo by ordering it under inclusion.

**Lemma 9.1.16** \( \langle \Phi, \subseteq \rangle \) is a cpo with least element \( \phi(s_{\Pi}) \) and every nonempty subset \( A \) of \( \Phi \) has a lub: \( \bigcup A \). Furthermore for every nonempty \( S \subseteq \Pi \) we have \( \phi(\bigcup S) = (\bigcup_{s \in S} \phi(s) = ) \bigcup \phi(S) \in \Phi \).

**Proof** Because \( \phi(s_{\Pi}) \subseteq \phi(t) \) for every \( t \in \Pi \) it is a \( \subseteq \)-least element of \( \Phi \). Now let a nonempty subset \( A \) of \( \Phi \) be given. Of course \( \bigcup A \) is a lub for \( A \) if it belongs to \( \Phi \). To see this notice at first that by definition of \( \Phi \) there for each \( a \in A \) exists a \( s_a \in \Pi \) such that \( a = \phi(s_a) \). Since \( \phi \) is a natural extension we then get

\[
\bigcup A = \bigcup_{a \in A} a = \bigcup_{a \in A} \phi(s_a) = \phi(\bigcup_{a \in A} s_a)
\]

and because \( \Pi \) is closed under union and \( A \neq \emptyset \) then \( \bigcup_{a \in A} s_a \in \Pi \), so we conclude \( \bigcup A \in \Phi \). From this it also appears that \( \phi(\bigcup S) = \bigcup \phi(S) \in \Phi \) for every \( \emptyset \neq S \subseteq \Pi \). \( \square \)

We are now concerned with the compact elements of \( \Phi \).

Recall that \( D \) is a directed set if it is nonempty and for all \( d, d' \in D \) there exists an ub in \( D \). An element \( a \in \Phi \) is then compact if for every directed subset \( D \) and \( \Phi \) such that \( a \subseteq D \) there is a \( d \in D \) with \( a \subseteq d \).

**Lemma 9.1.17** The compact elements of \( \langle \Phi, \subseteq \rangle \) is those \( \phi(s) \), where \( s \in \Pi \) and \( s \) is a finite set.

Notice that by a) and b) \( \phi(s) \) must be finite too when \( s \) is.
Proof It is standard that a finite set $\phi(s) = a \in \Phi$ is compact when the partial order is $\subseteq$ and lub is $\cup$: Let $D \subseteq \Phi$ be a directed set such that $a \subseteq \cup D$. Then for each $p \in a$ we can select a $d_p \in D$ with $p \in d_p$. Denote the set of those $d_p$’s by $D_a \subseteq D$. Since $a$ is finite $D_a$ must be finite too and has an ub $d \in D$ because $D$ is directed, so $a \subseteq \cup D_a \subseteq d \in D$. Now $\Phi$ is not an ordinary subset of $P(P)$—it is induced from $\phi$ and $\Pi \subseteq P(P)$, so it is less standard to see that all compact elements $\phi(s) \in \Phi$ are such that $s$ is a finite set—for instance it could be that $\phi$ “collapsed” an infinite set into a finite. But assume on the contrary there is a compact element $\phi(s) = a \in \Phi$ where $s$ is infinite.

The idea will be to construct an increasing infinite chain $D$: $d_0, d_1, \ldots, d_t, \ldots$ with $a = \cup D$. Because $a$ is compact and $D$ is directed there will then be a $d \in D$ such that $a \subseteq d$. Since $D$ is increasing then also $d \subseteq d'$ for $d' \in D$. From $d' \subseteq \cup D = a$ then $a \subseteq a$—a contradiction.

We now construct the infinite increasing chain. Since $s$ is infinite it contains a countable infinite subset $u = \{p_n\}_{n \in \mathbb{N}}$. For each $n \in \mathbb{N}$ define:

$$
\begin{align*}
& t_n = \{p_i \in u \mid i \leq n\} \\
& s_n = \{p \in s \mid \forall j > n. \, p_j \not\subseteq p\} \\
& d_n = \phi(s_n \cup t_n)
\end{align*}
$$

By the assumption of $\Pi$ each element of $s$ is contained in $\Pi$ as singleton sets, and $\Pi$ is closed under union, so $s_n \cup t_n \in \Pi$ and $d_n \in \Phi$ for every $n \in \mathbb{N}$. Clearly $t_n \subseteq t_{n+1}$ and $s_n = \{p \in s \mid p_{n+1} \not\subseteq p\} \cap s_{n+1} \subseteq s_{n+1}$ so because $\phi$ is $\subseteq$-monotone it follows that $D = \{d_n\}_{n \in \mathbb{N}}$ forms a nondecreasing chain in $\Phi$.

To see that $D$ in fact forms an increasing chain it is then enough for each $n \in \mathbb{N}$ to find an $m > n$ such that $d_n \neq d_m$.

Let $n$ be given. By a) $\{q \in P \mid q \not\subseteq p\}$ is finite for every $p \in t_n$. From the finiteness of $t_n$ we see that $\{q \mid \exists p \in t_n. \, q \not\subseteq p\}$ is finite too, so because $u \setminus t_n$ is infinite there must be a $p_m \in u \setminus t_n$ with $\forall p \in t_n. \, p_m \not\subseteq p$. It follows that $m > n$ and by definition of $s_n$ then $\forall p \in s_n. \, p_m \not\subseteq p$, so we actually have

$$
(9.2) \quad \forall p \in s_n \cup t_n. \, p_m \not\subseteq p
$$

Using c) we can now find a $q \in \phi(p_m)$ with $p_m \not\subseteq q$. From (9.2) and the transitivity of $\subseteq$ we conclude $\forall p \in s_n \cup t_n. \, q \not\subseteq p$. By b) we then see that $q \not\subseteq \phi(p)$ for all $p \in s_n \cup t_n$ which, because $d_n = \phi(s_n \cup t_n) = \bigcup_{p \in s_n \cup t_n} \phi(p)$ implies $q \not\subseteq d_n$. Since $q \in \phi(p_m) \subseteq \phi(s_m \cup t_m) = d_m$ we then get $d_n \neq d_m$ as desired.

It remains to show $a = \cup D$. $\supseteq$ follows from $a$ being a ub for $D$ and to see $\subseteq$ let a $q \in a$ be given. Because $a = \phi(s)$ there must be a $p \in s$ with $q \in \phi(p)$. If $p \in u$ then $p = p_n$ for a $n \in \mathbb{N}$ and clearly then $q \in \phi(p_n) \subseteq d_n$. If on the other hand $p \in s \setminus u$ we know from a) that there only is finitely many $p_i \in u$ with $p_i \subseteq p$. Suppose $p_n$ is the last member of $u$ with $p_n \subseteq p$. Then for all $i > n, \, p_i \not\subseteq p$ wherefore $p \in s_n$ and $q \in \phi(p) \subseteq \phi(s_n) \subseteq d_n$. In both cases $q \in d_n$ for some $d_n \in D$, so because $\cup D$ is a ub for $D$ we arrive at $q \in \cup D$. □

Proposition 9.1.18 $\langle \Phi, \subseteq \rangle$ is an algebraic cpo.
Proof From the previous two lemmas we know that \( \langle \Phi, \subseteq \rangle \) is a cpo and how the compact elements look like. So let an element \( a \in \Phi \) be given. We shall show that \( a \) is the lub of the compact elements below \( a \) — i.e., that \( a = \bigcup D_a \) where \( D_a = \{ \phi(s) \in \Phi \mid \phi(s) \subseteq a, s \) is finite\}. \( a \) being a lub for \( D_a \) gives \( \bigcup D_a \subseteq a \) and to see \( a \subseteq \bigcup D_a \) let a \( q \in a \) be given. Then \( q \in \phi(s) \) for some \( p \in s_a \subseteq \Pi \), where \( a = \phi(s_a) \). Hence \( \phi(s) \subseteq a \) and because \( \{ p \} \) is finite, \( \phi(s) \subseteq a \). Therefore \( \phi(s) \subseteq a \) and \( q \in \phi(s) \subseteq a \).

Proposition 9.1.19 Let \( \Phi_1 \) and \( \Phi_2 \) be two algebraic cpos constructed as above. Then \( \Phi = \{ \langle \phi_1(s), \phi_2(t) \rangle \mid s \in \Pi_1, t \in \Pi_2 \) and \( s \subseteq t \} \) also is an algebraic cpo ordered under \( \subseteq \) (component wise) with least element \( \langle \phi_1(s_{\Pi_1}), \phi_2(s_{\Pi_2}) \rangle \) and every nonempty \( D \subseteq \Phi \) has a lub \( \bigcup D = \bigcup D_1, \bigcup D_2 \in \Phi \), where \( D_i = \{ d_i \mid \langle d_1, d_2 \rangle \in D \} \). The compact elements of \( \Phi \) are those \( \langle \phi_1(s), \phi_2(t) \rangle \in \Phi \) where \( s \) and \( t \) are finite sets.

Proof \( \Phi \subseteq \Phi_1 \times \Phi_1 \) and the result can be derived from Hennessy [Hen88a, page 123]. At the first glance the result may seem obvious, but it has to be ensured that the lub actually belongs to \( \Phi \). \( \phi_i(\bigcup D_i) = \bigcup \phi_i(D_i) \) is important here. Also the compact elements must be dealt with.

9.2 Denotational Set-up

In this section we present two sets of models. One set will be the extension of the models of the previous two chapters: the models corresponding to the operational \( G \)-semantics \( (\preceq_G) \) and the \( P_{or} \)-model for the precongruence \( (\preceq_G^P) \) over \( RBL \). For convenience we shall denote such a extended model by \( M_\star \) where \( \star \) can be either \( or \) or \( G \) (a set of direct tests). \( M_\star \) on the other hand will denote a model from the other set of models corresponding to the semantics \( (\preceq_G) \) where the experimenter records prefix’s of sequences. The domains of the \( M_\star \) \( (M_\star) \) models will be denoted \( A_\star (A_\star) \) and the operators corresponding to the combinators of the different languages in question will follow the same notational scheme.

9.2.1 The Recursive Languages

In the first section we have seen different pleasant consequences of having domains where the compact elements are denotable/ reachable. The goal will therefore be to extend the domains of the models from the preceding chapters to deal with “infinity” while at the same time enforcing constraints which ensures the reachability. The first subgoal is easely attained simply by considering infinite sets of pomsets instead of finite. Recalling that the different denotational maps were based on the canonical map, \( \varphi \), we get a clue for the second subgoal. At first we look at what pomsets we can get by \( \varphi \). Here we shall lean on a result of Grabowski [Gra81] which essentially states that the sets of pomsets generated from the singleton pomsets and \( \varepsilon \) by sequential and parallel composition exactly are the \( N \)-free pomsets.

Definition 9.2.1 \( P_{N,free} \)-Property for Pomsets
A pomset $p$ is said to have the $P_{N\text{-free}}$-property, $P_{N\text{-free}}(p)$ iff for all $x, x', y, y'$ in $X_p$ we have:

$$
\begin{align*}
&x <_p x' \\
&\text{if } \text{co}_p \text{ co}_p \text{ and } x <_p y' \text{ then } y \leq_p x'
\end{align*}
$$

If a pomset $p$ has the $P_{N\text{-free}}$-property we say that $p$ is $N$-free.

We shall say that a $P(P)$-refinement, $\varphi$, is $N$-free iff $p$ is $N$-free for all $p \in \varphi(a)$ and $a \in \Delta$. Similar a particular refinement for a lpo $p$, $\pi_p$, is $N$-free iff $\pi_p(x)$ is $N$-free for all $x \in X_p$.

Example: $a \xrightarrow{c} b \quad c \xleftarrow{d} d$ and $a \xrightarrow{c} b \quad c \xleftarrow{d} d$ are $N$-free, but $a \xrightarrow{c} b \quad c \xleftarrow{d} d$ is not.

Gischer [Gis88] also calls these pomsets for the series-parallel pomsets and give an alternative and clear proof of the result, which (slightly modified for our set-up) can be formulated:

**Theorem 9.2.2** For all pomsets $p$:

$$
P_{N\text{-free}}(p) \text{ and } p \neq \varepsilon \iff \exists E \in DBL. \varphi(E) = \{p\}
$$

Because $\varphi(E_0 \oplus E_1)$ equals the union of $\varphi(E_0)$ and $\varphi(E_1)$ we immediately get:

**Corollary 9.2.3** If $P$ is a finite and nonempty set of $N$-free pomsets such that $\varepsilon \notin P$ then $\exists E_P \in BL. \varphi(E_P) = P$.

On top of the canonical map the relevant $\delta_\ast$-closure were used top obtain the denotation. This suggests to let the elements of $A_\ast$ be sets of pomsets which are obtained as the $\delta_\ast$-closure of a set of $N$-free nonempty pomsets. As already argued in the introduction to this chapter, information of the $M_\ast$-models must be incorporated when it comes to the $MP_\ast$-models for the semantics concerning prefix. Using the $\pi$-closure of pomsets to capture the idea of prefixes of sequences it appears that elements of $A_\ast^\pi$ should be pairs where the second component is an element of $A_\ast$ and the first component is the $\delta_\ast$- and $\pi$-closure of a nonempty set of $N$-free pomsets with the additional constraint that this set of $N$-free pomsets shall be a superset of the other set which the second component is a $\delta_\ast$-closure of. The additional constraint originates in the fact that if a maximal sequence can be recorded then so can any prefix of it. As we have seen in corollary 7.3.4 and proposition 8.2.3 the $P_G$ and $P_{or}$ properties are both hereditary and dot synthesizable. By proposition 6.4.5 and proposition 6.4.11 $\delta_\ast$ and $\pi$ then commute so it make sense to talk about the $\delta_\ast$-$\pi$-closure of a set. Formally

\begin{align*}
A_\ast &= \{\delta_\ast(t) \mid t \subseteq P_{N\text{-free}}, \varepsilon \not\subseteq t\} \\
A_\ast^\pi &= \{\langle\delta_\ast\pi(s), \delta(t)\rangle \mid s, t \subseteq P_{N\text{-free}}, \varepsilon \not\subseteq t \subseteq s \neq \emptyset\}
\end{align*}
We shall often make use of the observation that \( t \subseteq s \Rightarrow \delta_s(t) \subseteq \delta_t(s) \Rightarrow \delta_s(t) \subseteq \delta_\pi(s) \) because \( \delta_s \) is \( \subseteq \)-monotone and because in general \( p \in \pi(p) \).

We use the results of subsection 9.1.4 of this chapter to show that \( A_s \) and \( A_p^r \) are algebraic cpos. To this end notice that \( A_s \) and \( A_p^r \) can be written such that

\[
A_s = \Phi_2
\]

\[
A_p^r = \{ (\phi_1(s), \phi_2(t)) \mid \phi_1(s) \in \Phi_1, \phi_2(t) \in \Phi_2, t \subseteq s \}
\]

where

\[
\Phi_1 = \{ \phi_1(s) \mid s \in \Pi_1 \} \quad \Phi_2 = \{ \phi_2(t) \mid t \in \Pi_2 \}
\]

\[
\phi_1 = \delta_s \circ \pi \quad \phi_2 = \delta_\pi
\]

\[
\Pi_1 = \{ s \subseteq P_{N,\text{free}} \mid s \neq \emptyset \} \quad \Pi_2 = \{ t \subseteq P_{N,\text{free}} \mid \varepsilon \not\subseteq t \}
\]

Clearly \( s_{\Pi_1} = \{ \varepsilon \} \in \Pi_1 \) and \( s_{\Pi_2} = \emptyset \in \Pi_2 \) are elements such that \( \phi_i(s_{\Pi_i}) \subseteq \phi_i(s) \) for every \( s \in \Pi_i \) and \( i = 0, 1 \).

Recall that we for each pomset, \( p \), have an associated multiplicity function \( m_p \) which for each \( a \in \Delta \) gives the number of elements in \( p \) that are labelled \( a \). We saw that \( p \leq q \) iff \( \forall a \in \Delta, m_p(a) \leq m_q(a) \) defined a partial order on pomsets. Since we only are dealing with finite pomsets a moment's reflection shows that \( \{ p' \in P \mid \exists q, p' \leq q \subseteq p \} \) is finite (given a multiplicity function, \( m' \), which only differs from 0 on finitely many \( a \in \Delta \), there is only finitely many \( m \) below \( m' \), and for each such \( m \) there is only a finite number of pomsets \( p \) with \( m_p = m \)). Obviously \( q \in \phi_i(p) \) implies \( m_q \leq m_p \) for \( i = 0, 1 \). Since \( \delta_\pi(p) \neq \emptyset \) for the pomset properties we are dealing with and \( q \in \delta_\pi(p) \) only differs from \( p \) by the ordering of elements we have \( m_q = m_p \) here. In the case of \( \pi \) we have \( p \in \pi(p) \) so we conclude that there for \( i = 0, 1 \) exists a \( q \in \phi_i(p) \) such that \( m_p \leq m_q \). With the multiplicity preorder on pomsets and the other definitions above we from the results of the first section get:

**Proposition 9.2.4** \( \langle A_s, \subseteq \rangle \) and \( \langle A_p^r, \subseteq \rangle \) (component wise) are algebraic cpos with least elements \( \emptyset \) and \( \langle \{ \varepsilon \}, \emptyset \rangle \) respectively. The compact elements are those \( \delta_s(s) \in A_s \) and \( \langle \delta_\pi(s), \delta_\pi(t) \rangle \in A_p^r \) where \( s \) and \( t \) are finite sets. Every nonempty \( D \subseteq A_s \) has a lub: \( \bigvee D = \bigcup D \in A_s \) and similar every nonempty \( D \subseteq A_p^r \) has a lub \( \bigvee D = \{ \bigcup D_1, \bigcup D_2 \} \in A_p^r \) where \( D_i = \{ d_i \mid \langle d_1, d_2 \rangle \in D \} \) for \( i = 0, 1 \).

The next step is to equip the algebraic cpos with operators corresponding to the different combiners of the languages in question.

In order to be in keeping with the notation we used in the previous chapters we will deviate from Hennessy by letting \( RBL_{\Omega}^{rec}(X) \) denote the set of recursive expressions over \( RBL \). I.e., in terminology of Hennessy \( RBL_{\Omega}^{rec}(X) \) would be \( REC_{\Sigma}(X) \) where the signature \( \Sigma \) is \( a, , \oplus, \| \) and \( [a] \). The set of syntactic finite expressions \( (FREC_{\Sigma}(X)) \) will be denoted \( RBL_{\Omega}(X) \). \( RBL_{\Omega}^{rec} \) is the set of recursive processes over \( RBL \), i.e., the closed expressions of \( RBL_{\Omega}^{rec}(X) \). We use similar notation for the recursive expressions over \( BL \), e.g., \( BL_{\Omega} \) is the set of syntactic finite processes of \( BL_{\Omega}^{rec}(X) \). The recursion combiners will be assumed to have lower precedence than the other combiners of the language in question.

218
**Definition 9.2.5** Assume \( d = (P, Q) \) and \( d_i = (P_i, Q_i) \) for \( i = 0, 1 \) are elements of \( A^p_r \). Then the operators of the \( M^p_r \) models are defined as follows:

\[
\begin{align*}
\Omega^p_r &= \langle \{\varepsilon\}, \emptyset \rangle \\
\alpha^p_r &= \langle \{\varepsilon, a\}, \{a\} \rangle \\
d_0 \sqcap d_1 &= (P_0 \cup Q_0 \cdot P_1, Q_0 \cdot Q_1) \\
d_0 \oplus d_1 &= (P_0 \cup P_1, Q_0 \cup Q_1) \\
d_0 \parallel d_1 &= (\delta_s(P_0 \times P_1), \delta_s(Q_0 \times Q_1)) \\
d[d]_{\text{or}}^p &= (\delta_{or}(P_\langle \varphi(a)\rangle), \delta_{or}(Q_\langle \varphi(a)\rangle))
\end{align*}
\]

The operators of the \( M_s \) models are derived from those of the \( M^p_r \) simply by projecting the second component. I.e., if \( P_0, P_1 \in A \), then \( P_0 ||, P_1 \) equals \( \delta_s(P_0 \times P_1) \).

**Proposition 9.2.6** The operators in the definition above are well-defined.

**Proof** We give a proof for the operators on \( A^p_r \). That the \( A_s \)-operators are well-defined is then easily derived.

\( \Omega^p_r \): This constant equals \( \langle \{\varepsilon\}, \emptyset \rangle = (\delta_s(\{\varepsilon\}), \delta_s(\emptyset)) \) which is a member of \( A^p_r \) because \( \emptyset \subseteq P_{N\text{-free}} \) and \( \varepsilon \not\in \emptyset \). \( \alpha^p_r \): From \( a \in P_{N\text{-free}} \) and \( \varepsilon \not\in \{a\} \not\neq \emptyset \) we see \( \alpha^p_r = \langle \{\varepsilon, a\}, \{a\} \rangle = (\delta_s(\{a\}), \delta_s(\{a\})) \in A^p_r \).

For the binary operators on \( A^p_r \) assume \( d_0, d_1 \in A^p_r \). Then \( d_0 = (\delta_s(p_{s0}), \delta_s(t_{t0})) \) for some \( s_0, t_0 \in P_{N\text{-free}} \) such that \( \varepsilon \not\in t_0 \subseteq s_0 \not\neq \emptyset \). Similar for \( d_1 \).

\( \sqcap^p_r \): From a) of proposition 9.2.7 below and the distributivity of \( \delta_s \) over \( \cdot \) we immediately get: \( d_0 \sqcap d_1 = (\delta_s(p_{s0} \cup t_0 \cdot s_1), \delta_s(t_0 \cdot t_1)) \). By Grabowski \( p \cdot q \) is \( N \)-free when \( p \) and \( q \) are (can also be deduced from lemma 9.2.9 and the observations on page 136). So \( d_0 \sqcap d_1 \in A^p_r \) then follows from \( \varepsilon \not\subseteq t_0 \cdot t_1 \) because \( \varepsilon \not\subseteq t_0, t_1 \subseteq s_0 \) since \( t_1 \subseteq s_1 \not\neq \emptyset \) by \( s_0 \not\neq \emptyset \).

\( \oplus^p_r \): Immediate from the distributivity of \( \delta_s \) over \( \cup \) and proposition 9.2.7.

\( \parallel^p_r \): From proposition 6.4.4 and proposition 9.2.7 we directly get \( d_0 \parallel^p_r d_1 = (\delta_s(p_{s0} \times s_1), \delta_s(t_0 \times t_1)) \). Because the parallel composition of \( N \)-free pomsets are \( N \)-free \( d_0 \parallel^p_r d_1 \in A^p_r \) is then easily deduced from the assumptions of \( s_0, t_0, s_1, t_1 \).

It remains to show that the \([a]_{\text{or}}^p_r \) operator on \( A^p_{or} \) is well-defined. Let a \( d \in A^p_{or} \) be given and assume \( d = (\delta_{or}(s), \delta_{or}(t)) \) where \( s, t \subseteq P_{N\text{-free}} \) and \( \varepsilon \not\subseteq t \subseteq s \not\neq \emptyset \). Using lemma 8.2.6 for the second component and d) of proposition 9.2.7 below for the first we get \( d[a]_{\text{or}}^p = (\delta_{or}(s_\langle \varphi(a)\rangle), \delta_{or}(t_\langle \varphi(a)\rangle))) \).

\( \varphi(a) \in BL \) for every \( a \in \Delta \), so from corollary 9.2.3 \( (\varphi(a))(a) \) is a set of \( N \)-free nonempty pomsets when \( a \in \Delta \). Hence from lemma 9.2.9 we know that \( s_\langle \varphi(a)\rangle \) and \( t_\langle \varphi(a)\rangle \) are sets of \( N \)-free pomsets because \( s \\& t \) are assumed to be \( N \)-free too. \( \varphi(a) \) is \( \varepsilon \)-free so we conclude that \( d[a]_{\text{or}}^p \in A^p_r \).

The following proposition is useful not only for the proof of the proposition above but also for other to come.
**Proposition 9.2.7** Let \( \varrho \) be an \( \varepsilon \)-free \( \mathcal{P}(\mathcal{P}) \)-assignment and suppose \( P, Q \) and \( R \) are sets of pomsets such that \( P \supseteq R \). Then

\[
\begin{align*}
\text{a)} & \quad \delta_s \pi(P) \cup \delta_{s}(R) \cdot \delta_s \pi(Q) = \delta_s \pi(P \cup R \cdot Q) \\
\text{b)} & \quad \delta_s \pi(P) \cup \delta_s \pi(Q) = \delta_s \pi(P \cup Q) \\
\text{c)} & \quad \delta_s(\delta_s \pi(P) \times \delta_s \pi(Q)) = \delta_s \pi(P \times Q) \\
\text{d)} & \quad \delta_{or} \pi((\delta_{or} \pi(P)) \langle \varrho \rangle) = \delta_{or} \pi(P \langle \varrho \rangle)
\end{align*}
\]

**Proof**

\( \text{a)} \) At first we deduce:

\[
(9.3) \quad \pi(p \cdot q) = \pi(p) \cup \{p\} \cdot \pi(q)
\]

from proposition 6.2.6 and the observations direct before that proposition. We then get:

\[
\begin{align*}
& \delta_s \pi(P) \cup \delta_s(R) \cdot \delta_s \pi(Q) \\
& = \delta_s(\pi(P) \cup R \cdot \pi(Q)) \\
& = \delta_s(\pi(P) \cup \pi(R) \cup R \cdot \pi(Q)) \quad \text{if } R \subseteq P \text{ and } \pi \text{ is } \subseteq\text{-monotone} \\
& = \delta_s(\pi(P) \cup R \cdot \pi(Q)) \quad \text{by (9.3)} \\
& = \delta_s(\pi(P \cup R \cdot Q)) \\
& = \pi \text{ distributes over } \cup \\
& \text{b)} \quad \text{Follows from the distributivity of } \delta_s \text{ and } \pi \text{ over } \cup.
\end{align*}
\]

\[
\begin{align*}
\text{c)} & \quad \delta_s(\delta_s \pi(P) \times \delta_s \pi(Q)) \\
& = \delta_s(\pi(P) \times \pi(Q)) \quad \text{proposition 6.4.4} \\
& = \delta_s(\pi(P \times Q)) \quad \pi \text{ distributes over } \times \\
\text{d)} & \quad \delta_{or} \pi((\delta_{or} \pi(P)) \langle \varrho \rangle) \\
& = \pi \delta_{or}((\delta_{or} \pi(P)) \langle \varrho \rangle) \quad \delta_{or} \text{ and } \pi \text{ commutes} \\
& = \pi \delta_{or}(\pi(P)) \langle \varrho \rangle \quad \text{lemma 8.2.6 (} \varrho \text{ is } \varepsilon\text{-free)} \\
& = \delta_{or}(\pi(P)) \langle \varrho \rangle \quad \delta_{or} \text{ and } \pi \text{ commutes} \\
& = \delta_{or}(\pi(P) \langle \varrho \rangle) \quad \text{lemma 9.2.8 below}
\end{align*}
\]

**Lemma 9.2.8** Let \( P \) be a set of pomset and \( \varrho \) a \( \mathcal{P}(\mathcal{P}) \)-refinement. Then

\[
\pi((\pi(P)) \langle \varrho \rangle) = \pi(P \langle \varrho \rangle)
\]

**Proof** \( \pi \) is a natural extension to sets of pomsets so it will do to show:

\[
\pi((\pi(p)) \langle \varrho \rangle) = \pi(p \langle \varrho \rangle)
\]

\( \supseteq \) Immediate from \( p \in \pi(p) \).

\( \subseteq \) Let a \( q \in \pi((\pi(p)) \langle \varrho \rangle) \) be given. Then \( q \subseteq r \) for some \( r \in s \langle \varrho \rangle \) where \( s \subseteq p \). By definition of \( \subseteq \langle \varrho \rangle \), \( r \in s \langle \varrho \rangle \) implies there is a \( \varrho \)-consistent p.ref. \( \pi_s \) for \( s \) with \( r = [s \langle \tau \rangle] \). Since \( s \subseteq p \) we can by the alternative characterization of \( \subseteq \) find a representative \( p' \) of \( p \) such that \( s = p'|X_s \) and \( X_s \) is \( \leq_{p'}\text{-downwards closed} \). Then \( X_s \subseteq X_{p'} \) so we can extend \( \pi_s \) to a \( \varrho \)-consistent p.ref. \( \pi_{p'} \) for \( p' \). Because \( s = p'|X_s \) and \( \pi_{p'} \) equals \( \pi_s \) on \( X_s \) we see

\[
\pi\langle s \rangle = p'\langle \pi_{p'} \rangle|_{X_s \langle \tau \rangle}.
\]

220
We now show that $X_{s<\pi_s}$ is $\leq_{p'<\pi_{p'}}$-downwards closed. Suppose $\langle x, x' \rangle \leq_{p'<\pi_{p'}} \langle y, y' \rangle$ and $\langle y, y' \rangle \in X_{s<\pi_s}$. By construction of $p'<\pi_{p'}$ the former implies $x \leq_{p'} y$. The latter similarly implies $y \in X_s$. Since $X_s$ is $\leq_{p''}$-downwards closed then $x \in X_s$. Now $x' \in X_{\pi_{p'}(x)}$ so because $\pi_{p'}$ equals $\pi_s$ on $X_s$ we also have $x' \in X_{s<\pi_s}$. Hence $\langle x, x' \rangle \in X_{s<\pi_s}$.

Using the alternative characterization of $\subseteq$ again we conclude $[s<\pi_s] \sqsubseteq [p'<\pi_{p'}]$. From the transitivity of $\sqsubseteq$, $q \sqsubseteq r = [s<\pi_s]$ and $[p'<\pi_{p'}] \in p'<q = p<q$ we then get $q \in \pi(p<q)$ as desired. 

\begin{lemma}
Suppose $P$ is a set of $N$-free pomsets and $\varrho$ is a $N$-free $\mathcal{P}(\mathcal{P})$-refinement. Then $P<\varrho$ is a set of $N$-free pomsets too.
\end{lemma}

\textbf{Proof} The lemma follows from $p<\varrho$ being a set of $N$-free pomsets when $p$ is $N$-free. To see this it is clearly enough to show that $p<\pi_p$ is $N$-free for any $\varrho$-consistent p.ref. $\pi_p$ for $p$ (also $N$-free). Of course $\pi_p$ is $N$-free when $\varrho$ is. The proof that $p<\pi_p$ is $N$-free is by contradiction. Assume $p<\pi_p = (X, \leq, \ell)$ is not $N$-free. By construction of $p<\pi_p$ this implies the existence of $\langle x, x' \rangle, \langle y, y' \rangle, \langle z, z' \rangle, \langle v, v' \rangle \in X$ such that

\begin{equation}
\langle x, x' \rangle < \langle z, z' \rangle \quad \text{co} \quad \langle y, y' \rangle < \langle v, v' \rangle
\end{equation}

holds, but $\langle y, y' \rangle \not\leq \langle z, z' \rangle$.

We consider the different cases:

$(x = v, y \neq v)$ or $(x \neq v, y = v)$: If $x \not\leq p y$ then by the construction of $p<\pi_p$ this implies $\langle x, x' \rangle \not\leq \langle y, y' \rangle$—a contradiction to the assumption. To see $x \not\leq p y$ assume w.l.o.g. $(x = v, y \neq v)$ holds. From $y \neq v$ and the construction of $p<\pi_p$ we deduce that $\langle y, y' \rangle < \langle v, v' \rangle$ only can be due to $y \not< p v$. $v = x$ then gives $y \not< p x$ and so $x \not\leq p y$.

$(x = v, x \neq z)$ or $(x \neq v, x = z)$: Similar as above we find $z \not\leq p v$ which leads to a contradiction in the same way.

$x \neq v, y \neq v, x \neq z$: From (9.4) and the construction of $p<\pi_p$ we derive

\begin{align*}
x &< p z \quad \text{co}_p \quad \text{co}_p \quad \text{and} \quad x < p v \\
y &< p v
\end{align*}

Hence $y < p z$ follows from the $N$-freeness of $p$ and then $\langle y, y' \rangle < \langle z, z' \rangle$—a contradiction to $\langle y, y' \rangle \not\leq \langle z, z' \rangle$.

$x = v, y = v, x = z$: I.e., $x = v = y = z$ so this time (9.4) and the construction of $p<\pi_p$ gives:

\begin{align*}
x' &< p_{\pi_p(x)} z' \quad \text{co}_{p_{\pi_p(x)}} \quad \text{co}_{p_{\pi_p(x)}} \quad \text{and} \quad x' < p_{\pi_p(x)} v' \\
y' &< p_{\pi_p(x)} v'
\end{align*}

By the $N$-freeness of $p_{\pi_p}$ then also $y' < p_{\pi_p(x)} z'$. Since $x = y = z$ the construction of $p<\pi_p$ yields $\langle y, y' \rangle < \langle z, z' \rangle$—again a contradiction.
A careful examination of the cases above shows that they actually exhaust all possible combinations of \( x = / \neq v, y = / \neq v \) and \( x = / \neq z \). Each time we reached a contradiction so the assumption, \( p < \pi_p > \) is not \( N \)-free, was wrong. \( \square \)

**Proposition 9.2.10**  The operators of \( A^p_x \) and \( A_* \) are continuous.

**Proof**  The continuity of the \( A_* \)-operators is easily derived from the continuity of the \( A^p_x \)-operators which we now deal with. Constants are continuous. For the binary operators it is enough to show that they are left and right continuous. To this end let \( D' \) be a nonempty subset of \( A^p_x \) and suppose \( \langle P, Q \rangle \) is a member of \( A^p_x \).

\( \therefore \): Right continuous: Let \( D = \langle P, Q \rangle \), \( \therefore P' \), \( Q' \) \( \subset \) \( \{ \langle P \cup Q \cdot P', Q' \cdot Q' \rangle | \langle P', Q' \rangle \in D' \} \). Then \( D_1 = \{ R_1 | \langle R_1, R_2 \rangle \in D \} = \{ P \cup Q \cdot P' \ | \langle P', Q' \rangle \in D' \} = \{ P \cup Q \cdot P' \ | \ P' \in D'_1 \} = P \cup Q \cdot D'_1 \) where the last equation follows from \( D'_1 \neq \emptyset \) which in turn is a consequence of \( D' \neq \emptyset \). Also \( D_2 = \{ Q \cdot Q' | \langle P', Q' \rangle \in D' \} = Q \cdot D'_2 \). We then have: \( V^p_x(\langle P, Q \rangle \cdot P', Q') = \langle \cup D_1, \cup D_2 \rangle = (\cup (P \cup Q \cdot D'_1), \cup (Q \cdot D'_2)) = (P \cup Q \cdot (\cup D'_1), Q \cdot (\cup D'_2)) = \langle P, Q \rangle \cdot P' \).

Left continuous: Here we have: \( V^p_x(D' \cdot P, Q) = \langle \cup \{ P' \cup Q', P \ | \langle P', Q' \rangle \in D' \} \cup \{ Q \cdot Q' \ | \langle P', Q' \rangle \in D' \} \rangle = \langle (\cup \{ P' \cup Q', P \} \cup (\cup \{ Q \cdot Q' \ | \langle P', Q' \rangle \in D' \}) \cdot P, (\cup D'_1 \cdot D'_2) \cdot Q \rangle = \langle \cup D'_1, \cup D'_2 \cdot P, \cup D'_2 \cdot Q \rangle = \langle \cup D'_1 \cdot D'_2 \cdot P, \cup D'_2 \cdot Q \rangle = V^p_x(D') \cdot P, Q). \)

\( \oplus \): Obvious left and right continuous since it just is the union of the respective components.

\( ||p_x|| \): Right continuous: With \( D = \langle P, Q \rangle \), \( ||p_x|| D' \) we have \( D_1 = \delta_x(P \times D'_1) \) and \( D_2 = \delta_x(Q \times D'_2) \). Because \( \delta_x \) is the natural extension to sets we have \( \cup D_1 = \delta_x(\cup (P \times D'_1)) = \delta_x(P \times D'_1) \) and similar for \( \cup D_2 \). It follows that \( V^p_x(\langle P, Q \rangle \cdot ||p_x|| D') = \langle \delta_x(P \times D'_1), \delta_x(Q \times \cup D'_2) \rangle = \langle P, Q \rangle \cdot ||p_x|| D'. \)

Left continuous: Symmetric.

Only the \( [\theta]_{or} \)-operator is missing. \( V^p_x(D'[\theta]_{or}) = V^p_x(\{ \delta_{or} \pi(P' < \varphi(g) >), \delta_{or}(Q' < \varphi(g) >) | \langle P', Q' \rangle \in D' \}) = \langle \cup \delta_{or} \pi(D'_1 < \varphi(g) >), \cup \delta_{or}(D'_2 < \varphi(g) >) \rangle \) since both \( \delta_{or} \) and \( \pi \) as well as \( < \varphi(g) > \) are natural extensions to sets we immediately get: \( V^p_x(D'[\theta]_{or}) = \langle \delta_{or} \pi((\cup D'_1) < \varphi(g) >), \delta_{or}((\cup D'_2) < \varphi(g) >) \rangle = \cup D'_1, D'_1 D'_2 [\theta]_{or} = V^p_x(D'[\theta]_{or}) \).

Now where we have showed that \( A_* \) and \( A^p_x \) are algebraic cpos and that the different operators are continuous on the respective domains, we are in a position to apply the results presented in the previous section.

So for \( B L^{rec}_{\Omega}(X) \) we get the denotational maps:

\[
A_G[] : B L^{rec}_{\Omega}(X) \rightarrow [E N V_{A_G} \rightarrow A_G] \\
A^p_G[] : B L^{rec}_{\Omega}(X) \rightarrow [E N V_{A^p_G} \rightarrow A^p_G]
\]

and for \( R B L^{rec}_{\Omega}(X) \):

\[
A_{or}[] : R B L^{rec}_{\Omega}(X) \rightarrow [E N V_{A_{or}} \rightarrow A_{or}] \\
A^p_{or}[] : R B L^{rec}_{\Omega}(X) \rightarrow [E N V_{A^p_{or}} \rightarrow A^p_{or}]
\]

\( A^p_{or}[] \) and \( A^p_{or}[] \) will be used to refer to the first and second component of \( A^p_{or}[] \) respectively. Notice that if \( E \) is a closed expression then \( A_*[E] = A^p_{or}[E] \subseteq A^p_{or}[E] \).

222
9.2.2 The Syntactic Finite Sublanguages

In this subsection we shall lift some of the results we obtained in the preceding chapters for $BL$ and $RBL$ to the corresponding syntactic finite expressions of $BL_{\Omega}^{rec}$ and $RBL_{\Omega}^{rec}$ respectively, namely $BL_{\Omega}$ and $RBL_{\Omega}$.

When we dealt with $BL$ and $RBL$ the canonical map, $\varphi$, associating sets of pomsets to $BL$-expressions, served as basis for the denotational maps in a natural way. It appears to be difficult to extend this idea (and $\varphi$) to $BL_{\Omega}$, but an extension to $BL_{\Omega}$ seems manageable. How such an extension should be depends on what we are aiming at. If we want to use $\varphi$ as basis for maps concerned with different kinds of maximal pomsets it might be natural to let $\varphi(\Omega)$ be the empty set, $\emptyset$, because $\Omega$ represents the process we know nothing about. If on the other hand we also want information about possible prefixes it might be just as natural to associate $\emptyset$ with $\Omega$ because $\emptyset$ is prefix of any pomset. Our $Mp$-models consists of pairs of pomsets with the "old" denotations from the $M_{\ast}$-models as second component. We therefore arrive at the following:

**Definition 9.2.11** The map $\varphi^p : BL_{\Omega} \rightarrow \mathcal{P}(P) \times \mathcal{P}(P)$ is defined inductively:

- $\varphi^p(\Omega) = \langle \{\varepsilon\}, \emptyset \rangle$
- $\varphi^p(a) = \langle \{\varepsilon, a\}, \{a\} \rangle$
- $\varphi^p(E_0 ; E_1) = \langle \varphi^p_1(E_0) \cup \varphi^p_2(E_0) \cdot \varphi^p_1(E_1), \varphi^p_2(E_0) \cdot \varphi^p_2(E_1) \rangle$
- $\varphi^p(E_0 \oplus E_1) = \langle \varphi^p_1(E_0) \cup \varphi^p_1(E_1), \varphi^p_2(E_0) \cup \varphi^p_2(E_1) \rangle$
- $\varphi^p(E_0 \mid E_1) = \langle \varphi^p_1(E_0) \times \varphi^p_1(E_1), \varphi^p_2(E_0) \times \varphi^p_2(E_1) \rangle$

where $\varphi^p_1(E) = P$ and $\varphi^p_2(E) = Q$ if $\varphi^p(E) = \langle P, Q \rangle$, i.e., $\varphi^p_1$ and $\varphi^p_2$ are the projections of $\varphi^p$ to the first and second component respectively.

The ordinary canonical map $\varphi$ is extended to $BL_{\Omega}$ by $\varphi = \varphi^p_2$.

Observe that $\forall E \in BL_{\Omega} \cdot \varphi^2_2(E) \subseteq \varphi^1_1(E)$.

**Example:** From $\varphi^p(\Omega ; d) = \langle \{\varepsilon\}, \emptyset \rangle$ follows

$\varphi^p((a ; b) ; (\Omega ; d \oplus c)) = \langle \{\varepsilon, a \rightarrow b, a \rightarrow b \rightarrow c\}, \{a \rightarrow b \rightarrow c\} \rangle$

and

$\varphi^p((a ; (\Omega ; d) \oplus b) ; c) = \langle \{\varepsilon, a, b \rightarrow c\}, \{b \rightarrow c\} \rangle$

The following proposition justifies to think of $\varphi^1_1$ as the canonical association of pomset prefixes of an expression.

**Proposition 9.2.12**

a) If $E \in BL$ then $\varphi^1_1(E) = \pi(\varphi(E))$.

b) If $E \in BL_{\Omega}$ then $\varphi^1_1(E) = \pi(\varphi^p_1(E))$.

223
Corollary 9.2.15 \[ A_⋆[E] = (δ_⋆(φ(E)), δ_⋆(ψ(E))) \] for every \( E \in BL \)
With the results obtained so far we are now able to show that the different models are surjective.

**Proposition 9.2.16** Every compact element of \( A_r^p \) and \( A_r \) is the denotation of a syntactic finite expression.

**Proof** The result for \( A_r \) is easily derived from the corresponding proof for \( A_r^p \). To see this let \( a \in A_r^p \) as an expression \( E \in BL_\Omega \subseteq RBL_\Omega \) such that \( A_r^p[E] = a \). So it will actually not be necessary to involve the refinement combinators in order to denote the compact elements of \( A_{or}^p \). Recall at first that \( a \) is an element of \( A_r^p \) in the \( M_r^p \) model when

\[
(9.6) \quad a = \langle \delta_r(s), \delta_r(t) \rangle
\]

where \( s \) and \( t \) are two sets of \( N \)-free pomsets such that \( \varepsilon \not\subseteq t \subseteq s \not\subseteq \emptyset \). Also the compact elements were characterized to be those where \( s \) and \( t \) are finite sets.

If \( u \) is an arbitrary finite and nonempty set of \( N \)-free pomset such that \( \varepsilon \not\subseteq u \) we from the last corollary and corollary 9.2.3 deduce there exists an \( E_u \in BL \) with

\[
(9.7) \quad A_r^p[E_u] = \langle \delta_r(u), \delta_r(u) \rangle
\]

Now let a compact element \( a \) like (9.6) be given. We deal with different cases of \( s \) and \( t \):

\( \varepsilon \not\subseteq s \) and \( t = \emptyset \): Then we can find an \( E_s \in BL \) fulfilling (9.7). Hence \( E = E_s : \Omega \in BL_\Omega \) and we get:

\[
A_r^p[E] = \langle \delta_r(s) \cup \delta_r(s) \cdot \{\varepsilon\}, \delta_r(s) \cdot \emptyset \rangle \quad \text{definition of } A_r^p[f] \\
= \langle \delta_r(s) \cup s, \emptyset \rangle = \langle \delta_r(s), \delta_r(\emptyset) \rangle = \langle \delta_r(s), \delta_r(t) \rangle 
\]

\( \varepsilon \not\subseteq s \) and \( t \neq \emptyset \): Because \( \varepsilon \not\subseteq t \) and \( s \not\subseteq \emptyset \) we can then find \( E_s, E_t \in BL \) fulfilling (9.7). Therefore \( E = (E_s : \Omega) \oplus E_t \in BL_\Omega \) and \( A_r^p[E] = A_r^p[E_s : \Omega] \oplus A_r^p[E_t] = \langle \delta_r(s), \emptyset \rangle \oplus \langle \delta_r(t), \delta_r(t) \rangle = \langle \delta_r(s) \cup \delta_r(t), \emptyset \rangle = \langle \delta_r(s \cup t), \delta_r(t) \rangle \), where the last equation follows from \( t \subseteq s \).

\( s = \{\varepsilon\} \): Because \( t \subseteq s \) and \( \varepsilon \not\subseteq t \) we must have \( t = \emptyset \) in this situation and \( E = \Omega \) will do.

\( \varepsilon \in s \) and \( s \setminus \{\varepsilon\} \not\subseteq \emptyset \): Then no matter whether \( t = \emptyset \) or \( t \neq \emptyset \) we can as above find a \( E' \in BL_\Omega \) such that \( A_r^p[E'] = \langle \delta_r(s \setminus \{\varepsilon\}), \delta_r(t) \rangle \). Letting \( E = \Omega \oplus E' \) we get \( A_r^p[E] = \langle \delta_r(s \cup \{\varepsilon\} \cup (s \setminus \{\varepsilon\})), \delta_r(t) \cup t \rangle = \langle \delta_r(s), \delta_r(t) \rangle \).

Inspecting how \( s \) and \( t \) can be for compact elements like (9.6) of \( A_r^p \) we see that all cases are covered. \( \square \)

As for \( RBL \) and \( BL \) we are able to establish a connection between \( RBL_\Omega \) and \( BL_\Omega \) via the map \( \sigma \) which we together with \( \{\varepsilon\} \) extend to \( RBL_\Omega \) as follows:

\[
\begin{align*}
\Omega \sigma &= \Omega \\
\Omega \{\varepsilon\} &= \Omega \\
as &= a \\
(E_0 : E_1) \sigma &= E_0 \sigma : E_1 \sigma \\
(E_0 : E_1) \{\varepsilon\} &= E_0 \{\varepsilon\} : E_1 \{\varepsilon\} \\
(E_0 \oplus E_1) \sigma &= E_0 \sigma \oplus E_1 \sigma \\
(E_0 \oplus E_1) \{\varepsilon\} &= E_0 \{\varepsilon\} \oplus E_1 \{\varepsilon\} \\
(E_0 \parallel E_1) \sigma &= E_0 \sigma \parallel E_1 \sigma \\
(E_0 \parallel E_1) \{\varepsilon\} &= E_0 \{\varepsilon\} \parallel E_1 \{\varepsilon\} \\
E[\varepsilon] \sigma &= (E \sigma) \{\varepsilon\}
\end{align*}
\]
**Proposition 9.2.17** For every $E \in RBL_{\Omega}$ we have:

a) $A_{or}[E] = A_{or}[E\sigma]$

b) $A_{or}^p[E] = A_{or}^p[E\sigma]$

**Proof**  

a) Since $A_{or}^p[]_2$ equals $A_{or}[]$ this is just a simple consequence of b).

b) The proof is by induction on the structure of $E$. The basis is immediate because $\Omega\sigma = \Omega$ and $a\sigma = a$. In the inductive step there are four cases:

$E = E_0 ; E_1$; Then:

\[
A_{or}^p[E] = A_{or}^p[E_0] ;_{or} A_{or}^p[E_1]  \quad \text{definition of } A_{or}^p[]
\]

\[
= A_{or}^p[E_0\sigma] ;_{or} A_{or}^p[E_1\sigma]  \quad \text{induction}
\]

\[
= A_{or}^p[E_0\sigma ; E_1\sigma]  \quad \text{definition of } A_{or}^p[]
\]

\[
= A_{or}^p[(E_0 ; E_1)\sigma] = A_{or}^p[E\sigma]  \quad \text{definition of } \sigma
\]

$E = E_0 \oplus E_1$ and $E = E_0 \parallel E_1$; Similar

$E = F[q]$; In this case we have:

\[
A_{or}^p[E] = (A_{or}^p[F])[q]_{or}
\]

\[
= (A_{or}^p[F\sigma])[q]_{or}
\]

\[
= (A_{or}^p[F\sigma\{q\}])
\]

\[
= (A_{or}^p[F[q\sigma]]) = A_{or}^p[E]
\]

 Lemma 9.2.18 If $E \in BL_{\Omega}$ then

a) $A_{or}[E\{q\}] = (A_{or}[E])[q]_{or}

b) $A_{or}^p[E\{q\}] = (A_{or}^p[E])[q]_{or}$

**Proof**  

a) The proof is like lemma 8.2.7 for $RBL$ but with the additional case $\Omega$ (see also b)).

b) Since $A_{or}[E]$ equals $A_{or}[E]_2$ we from a) and the definition of $[q]_{or}$ deduce

\[
A_{or}^p[E\{q\}]_2 = \delta_{or}(A_{or}^p[E]_2 <q\{q\}>)
\]

With this we then by induction on the structure of $E \in BL_{\Omega}$ prove

\[
A_{or}^p[E\{q\}]_1 = \delta_{or}\pi(A_{or}^p[E]_1 <q\{q\}>)
\]

from which b) then follows using (9.8).

$E = \Omega$; $A_{or}^p[\Omega\{q\}]_1 = A_{or}^p[\Omega]_1 = \{\varepsilon\} = \delta_{or}\pi(\varepsilon <q\{q\}>) = \delta_{or}\pi(A_{or}^p[\Omega]_1 <q\{q\}>)$.

$E = a$; Then:

\[
A_{or}^p[a\{q\}]_1 = A_{or}^p[q(a)]_1  \quad \text{definition of } \{q\}
\]

\[
= \delta_{or}(q^p(q(a)))  \quad q(a) \in BL \text{ and proposition 9.2.14}
\]

\[
= \delta_{or}\pi(q(a))  \quad q^p = \pi \circ q \text{— proposition 9.2.12}
\]

\[
= \delta_{or}\pi(\varepsilon,a <q\{q\}>)  \quad \text{definition of } \varepsilon
\]

\[
= \delta_{or}\pi(A_{or}^p[a]_1 <q\{q\}>)  \quad \text{proposition 6.3.3}
\]

226
By proposition 9.2.13 it then follows that

\[ \text{Corollary 9.2.20} \]
\[ \text{The different} \]
\[ \text{From this proposition and proposition 9.2.16 we immediately have:} \]

**Proof**

The proof for the \( M^p \) models is exemplary for the corresponding for the \( M_* \) models. Suppose \( E \in BL_\Omega \). Then

\[
A^p_\text{or}[E] = \langle \delta_\text{or}(\varphi^p(E)), \delta_\text{or}(\varphi^p_2(E)) \rangle \quad \text{proposition 9.2.14}
\]

\[
= \langle \delta_\text{or}(\varphi^p(E)), \delta_\text{or}(\varphi^p_2(E)) \rangle \quad \text{proposition 9.2.12}
\]

By proposition 9.2.13 it then follows that \( A^p_\text{or}[E] \in \text{Fin}(A^p) \). Now if \( E \in RBL_\Omega \), that is \( * = \text{or} \), then by proposition 9.2.17 \( A^p_\text{or}[E] = A^p[E] \) and because \( E_\sigma \in BL_\Omega \) it follows that \( A^p_\text{or}[E] \) denotes a compact element in \( A^p_\text{or} \).

From this proposition and proposition 9.2.16 we immediately have:

**Corollary 9.2.20** The different \( M_* \) and \( M^p \) models are finitary.

We end this section with a proposition corresponding to proposition 8.3.1 which gives a connection between the denotations of the \( M^p_\text{or} \)-model and the \( M^p_\text{w} \)-model.

**Proposition 9.2.21** Suppose \( E \in RBL_\Omega \). Then

**a)** \( A^p_\text{w}[E_\sigma] = \delta_\text{w}(A^p_\text{or}[E]) \)

**b)** \( A^p_\text{w}[E_\sigma]_i = \delta_\text{w}(A^p_\text{or}[E])_i \) for \( i = 1, 2 \)

**Proof** As in the preceding proofs a) is just the special case of b) with \( i = 2 \).

b) The case \( i = 2 \) follows exactly as the case \( i = 1 \):

\[
A^p_\text{w}[E_\sigma]_1 = \delta_\text{w}(\varphi^p_1(E_\sigma)) \quad \text{proposition 9.2.14}
\]

\[
= \delta_\text{w}(\delta_\text{or}(\varphi^p_1(E_\sigma))) \quad \text{\( \delta_\text{or} \circ \delta_\text{w} = \delta_\text{w} \) (from \( P^p_\text{w} \Rightarrow P^p_\text{or} \))}
\]

\[
= \delta_\text{w}(A^p_\text{or}[E_\sigma]_1) \quad \text{proposition 9.2.14}
\]

\[
= \delta_\text{w}(A^p_\text{or}[E]_1) \quad \text{proposition 9.2.17}
\]
9.3 Operational Set-up

9.3.1 The Recursive Languages

The configuration languages are as earlier obtained by adding $\hat{\cdot}$. With refinement the set
of recursive configuration expressions, $RCL_{\Omega}^{rec}(X)$, is then in the usual way defined as the
least set $C$ which satisfies:

$$\hat{\cdot} \in C$$

$$RBL_{\Omega}^{rec}(X) \subseteq C$$

$$E_0 ; E_1 \in C \text{ if } E_0 \in C \text{ and } E_1 \in RBL_{\Omega}^{rec}(X)$$

$$E_0 \parallel E_1 \in C \text{ if } E_0, E_1 \in C$$

$RCL_{\Omega}^{rec}$ is the set of recursive process configurations, i.e., the closed configuration ex-
pressions of $RCL_{\Omega}^{rec}(X)$. $RCL_{\Omega}(X)$, $RCL_{\Omega}$, $CL_{\Omega}^{rec}(X)$ etc. can then be considered as
$RCL_{\Omega}^{rec}(X)$ restricted to the appropriate sublanguage.

The different extended labelled transition systems are all changed in the same way to
cope with the new situation and further the change only affects the definition of internal
steps. The following rules are added:

$$\Omega \longrightarrow \Omega$$

$$\Omega[\varphi] \longrightarrow \Omega$$

$$recx. E \longrightarrow E[recx. E/x]$$

The intuition behind the first and last rule is explained by Hennessy [Hen88a, pages 202–
203] and the rule in the middle is there mainly for proof technical reasons. It can easily
be shown to have no operational effect. It is worth to notice that (modulo this rule) no extra rules are needed for refinement to cope with recursion. From the next example one
can see how it works.

**Example:** Suppose $E = a \oplus a ; x$ and $\varphi$ is a BL-refinement with $\varphi(a) = b$ and $\varphi(b) = a$. Then the following scenario shows a possible evolvement of $F = (recx. E)[\varphi]$:

$$F \longrightarrow (a \oplus a ; recx. E)[\varphi] \longrightarrow a[\varphi]$$

$$\longrightarrow (a ; recx. E)[\varphi] \longrightarrow b$$

$$\longrightarrow a[\varphi] ; F$$

$$\longrightarrow b ; F$$

$$\quad b \Rightarrow F \ldots$$

With a slight change $F = recx. (E[\varphi])$ we instead have:

$$F \longrightarrow^* (a ; F)[\varphi]$$

$$\longrightarrow^* (b ; F[\varphi])$$

$$\Rightarrow F[\varphi]$$

$$\longrightarrow^* (b ; F[\varphi])[\varphi]$$

$$\longrightarrow^* (a ; F[\varphi])[\varphi]$$

$$\Rightarrow (F[\varphi])[\varphi] \ldots$$

228
The definition of \( \preceq_G \) remains the same, but of course it is now defined over \( RBL_{\Omega}^{\text{rec}} \) and \( BL_{\Omega}^{\text{rec}} \) respectively. The new operational preorder, \( \preceq_G \), from the introduction to the chapter can be formulated:

**Definition 9.3.1** \( \preceq_G \subseteq RBL_{\Omega}^{\text{rec}} \times RBL_{\Omega}^{\text{rec}} \) is defined

\[
E_0 \preceq_G E_1 \iff \forall s \in G^*. E_0 \xrightarrow{\Delta} \implies E_1 \xrightarrow{\Delta}
\]

Restricting \( \preceq_G \) appropriately to the different sublanguages gives the remaining preorders.

Up till now we have got along with mainly structural induction. When it comes to recursion it will be convenient with the notion of the size of a step. For an internal step \( E \xrightarrow{>\cdot} E' \) the size, \( m \), will be indicated by a relation \( \xrightarrow{>\cdot}_m \), i.e., \( E \xrightarrow{>\cdot}_m E' \). \( \xrightarrow{>\cdot}_0 \) is the empty relation. Similar for external steps. \( m \) in \( E \xrightarrow{>\cdot}_m E' \) can be thought of as stating that there is a proof of \( E \xrightarrow{>\cdot} E' \) from the rules of \( \xrightarrow{>\cdot} \) with no more than \( m \) stages. E.g., if \( E_0 \xrightarrow{>\cdot}_m E_0' \) then \( E_0 \parallel E_1 \xrightarrow{>\cdot}_{m+1} E_0' \parallel E_1' \). See [Win85] for more details.

### 9.3.2 The Syntactic Finite Sublanguages

Not all notions can be carried over directly to the extended languages with recursion. For instance it is difficult to make sense in talking about the behaviours of a process of \( BL_{\Omega}^{\text{rec}} / RBL_{\Omega}^{\text{rec}} \) since a process now may continue infinitely. At least it is hard to see how the map \( \text{Beh} \) should be extended to say \( BL_{\Omega}^{\text{rec}} \) and we will not find any use for it. It will later turn out that the different operational preorders are determined by their restriction to the sublanguages of syntactic finite expression, i.e., \( BL_{\Omega} \) and \( RBL_{\Omega} \). We will therefore now look at how the previous obtained results for \( BL \) and \( RBL \) can be lifted to \( BL_{\Omega} \) and \( RBL_{\Omega} \). If a proposition need no reformulation (except e.g., \( BL \) should be replaced with \( BL_{\Omega} \)) will in the sequel simply be referred by writing it as: proposition\( \Omega \).

Both proposition 7.2.3 and proposition 8.1.1 extends directly to proposition\( \Omega \) 7.2.3 and proposition\( \Omega \) 8.1.1.

As we also saw for the denotational set-up, the syntactic map, \( \sigma \), removing refinements will be useful to establish connections from \( RBL_{\Omega} \) to \( BL_{\Omega} \) or when it comes to configurations, from \( RCL_{\Omega} \) to \( CL_{\Omega} \). We extend \( \sigma \) from \( RCL \) to \( RCL_{\Omega} \) in the same way as \( \sigma \) was extended from \( BL \) to \( RBL_{\Omega} \) namely by letting \( \Omega \sigma = \Omega \) and similar for \( \{ \emptyset \} \) we let \( \Omega \{ \emptyset \} = \Omega \). It should be clear that the maps \( \sigma : RCL_{\Omega} \longrightarrow CL_{\Omega} \) and \( \{ \emptyset \} : CL_{\Omega} \longrightarrow CL_{\Omega} \) when restricted to \( RCL \) and \( BL \) respectively gives the old maps.

The first important result we shall extend to \( RBL_{\Omega} \) is proposition 8.1.2:

\[
\forall E \in RBL_{\Omega}. E \xrightarrow{\Delta} \uparrow \iff E\sigma \xrightarrow{\Delta} \uparrow
\]

According to our convention we shall refer to it as proposition\( \Omega \) 8.1.2. Of course some care has to be taken because we now have to deal with one more extra case in the proofs: \( \Omega \), but because of the additional rule \( \Omega[\emptyset] \xrightarrow{>\cdot} \Omega \), this only give rise to minor changes. We briefly comment on selected parts of the results used to prove proposition\( \Omega \) 8.1.2. In
lemma\_\Omega 8.1.4 the case \( E = \Omega \) is trivially true and the same case is also easy in lemma\_\Omega 8.1.3. Here however there is also the case \( E = F[\varrho] \) where \( F = \Omega \). It is exactly here the rule \( \Omega[\varrho] \rightarrow \Omega \) is considered: Since \( F[\varrho]\sigma = \Omega\sigma\{\varrho\} = \Omega\{\varrho\} = \Omega \rightarrow \Omega \) this goes through smoothly and in (8.3)\_\Omega one just use \( \Omega\{\varrho\} = \Omega \). In lemma\_\Omega 8.1.7 \( E = \Omega \) is similar to \( E = a \) and in the following lemma it is even simpler.

If we go back and look at the proof of proposition\_\Omega 8.1.2 we see that the proposition just is a special case of a more general result which also implies:

**Proposition 9.3.2** For every \( E \in RBL\_\Omega \) and \( s \in G^* \):

\[
E \xrightarrow{\sigma} \iff E\sigma \xrightarrow{\sigma}
\]

The only remaining purely operational results from the preceding chapters are lemma 7.4.5 and lemma 7.4.6 which carry over totally unchanged, because \( \Omega \) is not directly involved in the rules for expressions of the form \( E_0 ; E_1 \) and \( E_0 \parallel E_1 \).

## 9.4 Full Abstractness

In this section we connect the denotational semantics with the operational through full abstractness results which are obtained by lifting via algebraicity of the involved preorders the corresponding results for the (syntactic) finite sublanguages.

### 9.4.1 The Recursive Languages

As mentioned in the beginning of the chapter we are after the largest precongruence contained in the relevant preorder. There we were just concerned with the ordinary combinators of the language in question, but of course we want the obtained preorder to be a precongruence w.r.t. to the recursive combinators too. If this shall make sense the operational preorders have to be extended to open expressions. This is usually done in what might be called the substitutive way:

\[
E_0 \leq_G E_1 \quad \text{iff} \quad \text{for every closed syntactic substitution } \rho, \ E_0\rho \leq_G E_1\rho
\]

\[
E_0 \preceq_G E_1 \quad \text{iff} \quad \text{for every closed syntactic substitution } \rho, \ E_0\rho \preceq_G E_1\rho
\]

The largest precongruence over \( BL\_\Omega^{rec}(X) \) contained in \( \leq_G \) will as usual be denoted \( \leq_G^c \). Similar for the other preorders. We can now formulate:

<table>
<thead>
<tr>
<th>Theorem 9.4.1</th>
<th>The following denotations are fully abstract:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) ( A_G[\emptyset] ) on ( BL_\Omega^{rec}(X) ) w.r.t. ( \leq_G )</td>
<td></td>
</tr>
<tr>
<td>b) ( A_{or}[\emptyset] ) on ( RBL_\Omega^{rec}(X) ) w.r.t. ( \leq_w )</td>
<td></td>
</tr>
<tr>
<td>c) ( A^{p}_{G}[\emptyset] ) on ( BL_\Omega^{rec}(X) ) w.r.t. ( \preceq_G )</td>
<td></td>
</tr>
<tr>
<td>d) ( A^{p}_{or}[\emptyset] ) on ( RBL_\Omega^{rec}(X) ) w.r.t. ( \preceq_w )</td>
<td></td>
</tr>
</tbody>
</table>
Proof. The denotational preorders $\mathcal{A}_s$ and $\mathcal{A}_p$ are qua induced by the denotational maps, precongruences w.r.t. all the combinators—the recursion combinators inclusive. By proposition 9.1.12 it is then enough to show the theorem to hold where the operational precongruences now are understood to be the largest w.r.t. the ordinary combinators.

By corollary 9.1.8 the associated (denotational) induced preorders are then substitutive as well as algebraic. The different operational preorders are by definition substitutive and by proposition 9.1.15 then so are the associated precongruences. Hence if we can manage to show that the involved operational precongruences are algebraic and agrees with the denotational preorders on the syntactic finite sublanguages (closed expressions) the theorem then follows.

From theorem 9.4.18 we know that $\preceq_G$ and $\preceq_p$ are algebraic over $RBL_{\Omega}^{rec}$ and therefore also over $BL_{\Omega}^{rec}$. Since theorem 9.4.19 gives the corresponding results full abstractness for the syntactic finite sublanguages it only remains to show the operational precongruences (w.r.t. the ordinary combinators) are algebraic:

a) $\preceq_G$ is algebraic on $BL_{\Omega}^{rec}$ and agrees on the finite expressions, $BL_\Omega$, with $\mathcal{A}_G$ so $\preceq_G$ is also a precongruence w.r.t. the ordinary combinators.

b) and d) Both $\preceq_w$ and $\preceq_p$ are algebraic and by theorem 9.4.22 $RBL_\Omega$ is expressive w.r.t. both preorders. Theorem 9.1.14 then gives us that $\preceq_w$ and $\preceq_p$ are algebraic over $RBL_{\Omega}^{rec}$.

c) $\preceq_w$ is algebraic on $BL_{\Omega}^{rec}$ and by theorem 9.4.22 $BL_\Omega$ is $\{E' \in BL_\Omega \mid L(E') \subseteq A\}$-expressive w.r.t. $\preceq_G$ for every finite subset $A$ of $\Delta$. For every $E \in BL_{\Omega}^{rec}$, $L(E)$ is finite and if $E' \in \text{Fin}(E)$ then $L(E') \subseteq L(E)$. Hence $BL_\Omega$ is $\text{Fin}(E)$-expressive for all $E \in BL_{\Omega}^{rec}$ and from theorem 9.1.14 it then follows that $\preceq_w$ is algebraic over $BL_{\Omega}^{rec}$ and this case is done.

Using the same considerations as in section 8.4 also:

Corollary 9.4.2. For $E_0, E_1 \in RBL_{\Omega}^{rec}(X)$ we have

$$(E_0 \preceq_w E_1) \Rightarrow (E_0 \preceq_G E_1) \Rightarrow (E_0 \preceq_w E_1)$$

and

$$(E_0 \preceq_w E_1) \Rightarrow (E_0 \preceq_G E_1) \Rightarrow (E_0 \preceq_w E_1)$$

Using the same considerations as in section 8.4 also:

Corollary 9.4.3. For all $E_0, E_1 \in RBL_{\Omega}^{rec}(X)$:

$$E_0 \preceq_G E_1 \iff E_0 \preceq_w E_1$$
$$E_0 \preceq_G E_1 \iff A_{or}[E_0] = A_{or}[E_1]$$
$$E_0 \preceq_G E_1 \iff A_{or}[E_0] = A_{or}[E_1]$$
$$E_0 \preceq_G E_1 \iff A_{or}[E_0] = A_{or}[E_1]$$
Since $A_{or}[E] = A_{or}^p[E]_2$ we from this corollary and the expressions of (9.1) in the introduction to the chapter get:

**Corollary 9.4.4** For all $E_0, E_1 \in RBL_{\Omega}^{rec}(X)$:

$$E_0 \preceq G E_1 \Rightarrow E_0 \preceq G E_1$$

and in general the implication does not hold in the other direction.

We shall now prove all the propositions we used to get the different full abstractness results.

In connection with the denotational set-up for recursion we have already meet the syntactic preorder $\preceq$. There it was used a relation telling what processes, $E$, that might be thought of as approximations to a process, $F$, possibly with recursion constructors, i.e., $E \preceq F$. We saw that the denotation of $F$ was the limit of all its syntactic finite approximations. When it comes to the operational set-up here, $\preceq$ will play a similar rôle. Recall that $\preceq$ was defined to be the least relation over $RBL_{\Omega}^{rec}(X)$ satisfying:

<table>
<thead>
<tr>
<th>$E \preceq E$</th>
<th>$\Omega \preceq E$</th>
<th>$E[\text{rec x. } E/x] \preceq E$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$E \preceq F, F \preceq G$</td>
<td>$E_0 \preceq F_0, E_1 \preceq F_1$</td>
<td>$E \preceq F_0 \implies \exists F : E \preceq F_0$</td>
</tr>
<tr>
<td>$E \preceq G$</td>
<td>$E_0 \preceq F_0 \implies F_0 \preceq F_0$</td>
<td>$E \preceq F \implies E \preceq F$</td>
</tr>
<tr>
<td>$E_0 \parallel E_1 \preceq F_0 \parallel F_1$</td>
<td>$E_0 \parallel E_1 \preceq F_0 \parallel F_1$</td>
<td>$E \parallel \Omega \preceq F \parallel \Omega$</td>
</tr>
</tbody>
</table>

$\preceq$ is extended to $RCL_{\Omega}^{rec}(X)$ simply by letting $\preceq$ be the least relation over $RCL_{\Omega}^{rec}(X)$ which satisfies the rules above. Notice that in this way we may only have $E \preceq F[\varrho]$ if $E$ and $F$ comes from $RBL_{\Omega}^{rec}(X)$. It is also important to notice that $\uparrow \preceq E$ implies $E = \uparrow$ and that $\preceq$ contains the old precongruence over $RBL_{\Omega}^{rec}(X)$.

Having extended $\preceq$ to $RCL_{\Omega}^{rec}(X)$ we at first show that if $E$ is an approximation of $F$ then $F$ can do all the sequences $E$ can. A stronger formulation of this is

**Lemma 9.4.5** Suppose $E, E' \in RCL_{\Omega}^{rec}$. Then

$$E \preceq E' \Rightarrow F' \implies \exists F : E \preceq F \preceq F'$$

**Proof** As usual by induction on the size of $\Rightarrow$ using the analogous lemma 9.4.9 for single steps.  

Before proving lemma 9.4.9 consider the situation where $E \preceq E' \Rightarrow F'$. We cannot expect that $E$ immediately can do an internal step and evolve into $F$ with $F \preceq F'$. This is because $E' \preceq E$ can imply that some of the recursive subexpressions of $E$ have been “unwound” by $\preceq$ in order to obtain an expression equal to $E'$ (up to $\Omega$ at some places in $E'$). By the recursion rule for $\Rightarrow$ it is possible to do one unwinding, so given $E' \preceq E$ we
would ideally like to unwind $E$ by internal steps to a $E''$ which equals $E'$ up to $\Omega$. Then we could be sure that whatever internal (or external for that matter) step $E'$ could do, $E''$ would be able to do a similar. There is however the snag about it that the definition of $\rightarrow$ does not open up for unwinding in the right hand argument of the $;\text{-combinator}$ and neither in the arguments of the $\oplus\text{-combinator}$. The situation is closely related with the one in chapter 8 where we wished to “perform” the substitutions of the refinement combinator of an expression. Thought by the experience there we define a subpreorder, $\preceq^u$, of $\preceq$ as the least relation over $RCL^\text{rec}_\Omega(X)$ which can be inferred from the rules:

\[
\begin{align*}
E \preceq^u E & \quad \frac{E \preceq^u F, F \preceq^u G}{E \preceq^u G} \\
\Omega \preceq^u E & \\
E_0 \preceq^u F_0, E_1 \preceq^u F_1 & \quad E_0 \preceq F_0, E_1 \preceq F_1 \quad E_0 \oplus E_1 \preceq^u F_0 \oplus F_1 \\
E_0 \parallel E_1 \preceq^u F_0 \parallel F_1 & \\
E_0 \preceq^u F_0, E_1 \preceq^u F_1 & \quad E \preceq^u F \\
E[\bar{\theta}] \preceq^u F[\bar{\theta}]
\end{align*}
\]

**Example:** $(\text{rec} y, E) ; (a \parallel \text{rec} x. (a \parallel x)) \preceq^u (\text{rec} y, E) ; \text{rec} x. a \parallel x$ but $(a \parallel \text{rec} x. (a \parallel x)) ; \text{rec} y. E \not\preceq^u (\text{rec} x. a \parallel x) ; \text{rec} y. E$

This definition of $\preceq^u$ deserves several remarks:

- The requirement that $\preceq^u$ shall be over $RCL^\text{rec}_\Omega(X)$ has the implication that an inference rule only may be used when for the consequent, $E \preceq^u F$, it is ensured that $E, F \in RCL^\text{rec}_\Omega(X)$
- $\preceq^u \subseteq \preceq$
- The preorder $\preceq$ is used in the premisses of the $;\text{- and } \oplus\text{-inference rule just in order to capture the unwindings which cannot be done by internal steps.}$
- There is no rule for $\text{rec} x.$ . This reflects that the expressions are equal up to $\Omega$ (except of course in connection with $;\text{ and } \oplus$)

By the last remark the following useful lemma is reasonable:

**Lemma 9.4.6**

- a) $\dagger \preceq^u F$ ($a \preceq^u F$) implies $F = \dagger (F = a)$
- b) $E_0 ; E_1 \preceq^u F$ implies $F = F_0 ; F_1, E_0 \preceq^u F_0$ and $E_1 \preceq F_1$
- c) $E_0 \oplus E_1 \preceq^u F$ implies $F = F_0 \oplus F_1$, $E_0 \preceq F_0$ and $E_1 \preceq F_1$

233
d) $E_0 \parallel E_1 \preceq^u F$ implies $F = F_0 \parallel F_1$, $E_0 \preceq^u F_0$ and $E_1 \preceq^u F_1$

e) $\text{rec } x. E \preceq^u F$ implies $F = \text{rec } x. E$

**Proof** Using structural arguments each implication is proven by a simple induction on the number of rules used to prove the left hand side of the implication.

In the following we need to be able to see that an internal step solely originate in an unwinding of a recursive subexpression. We write this as $E \rightarrow^{u\rightarrow} F$.

Formally $\rightarrow^{u\rightarrow} \subseteq \rightarrow$ is defined to be the least relation over $\text{RCL}^{\text{rec}}_\Omega$ which can be deduced from $\text{rec } x. E \rightarrow^{u\rightarrow} E[\text{rec } x. E/x]$ and the $\rightarrow^{u\rightarrow}$ equivalent versions of the $\rightarrow$ inference rules.

The lemma now states:

**Lemma 9.4.7** Given $E, E' \in \text{RCL}^{\text{rec}}_\Omega$ then

$$E \preceq E' \implies \exists F. E \rightarrow^{u\rightarrow} F \preceq^u E'$$

Before proving the lemma observe that there is an “unwind version” of proposition $^{\text{rec}}_{\Omega}$ 8.1.1.

**Proof** By induction on the number of rules used in the proof of $E' \preceq E$. There are three case in the basis:

$E = E'$: Let $F = E$ and $E \rightarrow^{u\rightarrow} F \preceq^u E = E'$.

$E' = \Omega \preceq E$: Then also $\Omega \preceq^u E$ and we can choose $F = E$ as above.

$E' = G[\text{rec } x. G/x] \preceq \text{rec } x. G = E$: By the recursion rule for $\rightarrow^{u\rightarrow}$ it is seen that $E \rightarrow^{u\rightarrow} G[\text{rec } x. G/x] = E' \preceq^u E'$ so we can choose $F = E'$.

Now for the inductive step there are five ways $E' \preceq E$ could have been obtained.

$E' \preceq E'', E'' \preceq E$: By hypothesis of induction there are $F'$ and $F''$ such that $E'' \rightarrow^{u\rightarrow} F' \preceq^u E'$ and $E \rightarrow^{u\rightarrow} F'' \preceq^u E''$. From lemma 9.4.8 below we know that $E'' \preceq^u E'' \rightarrow^{u\rightarrow} F'' F \preceq^u F'$. Then we actually have $E \rightarrow^{u\rightarrow} F'' \rightarrow^{u\rightarrow} F \preceq^u F' \preceq^u E'$ as we want.

$E' = E_0' \parallel E_1', E = E_0 \parallel E_1$ and $E_0' \preceq E_0, E_1' \preceq E_1$: Using the inductive hypothesis on $E_0 \preceq E_0'$ we find a $F_0$ such that $E_0 \rightarrow^{u\rightarrow} F_0 \preceq^u E_0'$. The unwind version of proposition $^{\text{rec}}_{\Omega}$ 8.1.1 then gives $E = E_0 \parallel E_1 \rightarrow^{u\rightarrow} F_0 \parallel E_1$. Since $E_1' \preceq E_1$ we by definition of $\preceq^u$ actually have $E' = E_0'; E_1' \preceq^u F_0; E_1$ and we can let $F = F_0 \parallel E_1$.

$E' = E_0' \oplus E_1', E = E_0 \parallel E_1$ and $E_0' \preceq E_0, E_1' \preceq E_1$: Then also $E \preceq^u E'$ so we can choose $F = E$ because $E \rightarrow^{u\rightarrow} F = E \preceq^u E'$.

$E' = E_0' \parallel E_1', E = E_0 \parallel E_1$ and $E_0' \preceq E_0, E_1' \preceq E_1$: By induction there for $i = 0, 1$ exists a $F_i$ such that $E_i \rightarrow^{u\rightarrow} F_i \preceq^u E_i'$, so if we use the unwind version of proposition $^{\text{rec}}_{\Omega}$ 8.1.1 we get $E = E_0 \parallel E_1 \rightarrow^{u\rightarrow} F_0 \parallel F_1$. Letting $F = F_0 \parallel E_1$ we have $F \preceq^u E_0' \parallel E_1' = E'$.

$E' = G'[\alpha], E = G[\alpha]$ and $G' \preceq G$ (and $G, G' \in \text{RBL}^{\text{rec}}_\Omega$): As above we find a $H$ such that $G \rightarrow^{u\rightarrow} H \preceq^u G'$. By definition of $\preceq^u$ we then have $F := H[\alpha] \preceq^u G'[\alpha] = E'$ and of course $E \rightarrow^{u\rightarrow} F$. 234
Lemma 9.4.8 If \( E, E' \in RCL_{\Omega}^{rec} \) then
\[
E \succeq^u E' \succ^u F' \implies \exists F. E \succ^u F \succeq^u F'
\]

Proof By induction on the number of unwinding steps using:
\[
(9.9) \quad E \succeq^u E' \succ^u F' \Rightarrow \exists F. E \succ^u F \succeq^u F'
\]
which in turn is proven by induction on the size, \( m \), of \( E' \succ^u F' \).
Since \( \succ^u_0 = \emptyset \) the basic case is trivial and in the inductive step we can assume (9.9) holds for \( m \) when proving (9.9) for \( m+1 \). The different rules are handled one by one:
\[
E' = \textit{rec}. G \succ^u_{m+1} G[\textit{rec}. G/x] = F': \text{ By lemma } 9.4.6 E \succeq^u \textit{rec}. G \text{ implies } E = \textit{rec}. G. \text{ Let } F = F' \text{ and we use the same rule to get } E \succ^u F \succeq^u F = F'.
\]
\[
E' = E_0'; E_1' \succ^u_{m+1} F_0'; E_1' = F' \text{ where } E_0' \succ^u_m F_0'; E_1' \succeq^u E \text{ implies } E = E_0; E_1 \text{ where } E_0' \succeq^u E_0 \text{ and } E_1' \succeq^u E_1. \text{ We can then use the hypothesis of induction to get an } F_0 \text{ with } E_0 \succ^u F_0 \succeq^u F_0'. \text{ Then also } E = E_0; E_1 \succ^u F_0'; E_1 \succeq^u F'_0; E_1' = F' \text{ and we can choose } F = F_0; E_1.
\]
\[
E' = E_0' || E_1' \succ^u_{m+1} F': \text{ There are two subcases which are handled similar/ symmetric as the rule for } ;.
\]
\[
E' = G'[\alpha] \succ^u_{m+1} H'[\alpha] = F' \text{ where } G' \succ^u_m H': \text{ By lemma } 9.4.6 E \succeq^u G'[\alpha] \text{ only if } E = G'[\alpha] \text{ and } G' \succeq^u G. \text{ Then by induction } G \succ^u H \succeq^u H' \text{ for some } H \text{ and we get } E \succ^u H[\alpha] \succeq^u H'[\alpha] = F' \text{ as desired.}
\]

We are now ready to prove the equivalent lemma of 9.4.5 for single steps.

Lemma 9.4.9 Given \( E, E' \in RCL_{\Omega}^{rec} \) and \( A \in G \). Then:
- \( E \succeq E' \longrightarrow F' \) implies \( \exists F. E \longrightarrow^* F \succeq F' \)
- \( E \succeq E' \xrightarrow{A} F' \) implies \( \exists F. E \xrightarrow{A} F \succeq F' \)

Proof Immediate form the preceding lemma 9.4.7 and the two following lemmas.

Lemma 9.4.10 If \( E, E' \in RCL_{\Omega}^{rec} \) then
\[
E \succeq^u E' \longrightarrow F' \text{ implies } \exists F. E \longrightarrow^* F \succeq F'
\]

Proof By induction on the size, \( m \), of the internal step \( E' \longrightarrow_m F' \).
The basic case is trivial and in the inductive case the lemma can be assumed to be true for all internal steps of size \( m \). We now investigate all the rules.
Using the fact that \( \succeq^u \subseteq \succeq \) and proposition 8.1.1 the inference rules are handled exactly as in the proof of lemma 9.4.8. E.g., \( E' = G'[\alpha] \longrightarrow_{m+1} H'[\alpha] = F' \) where \( G' \longrightarrow_m H' \). By lemma 9.4.6 \( E \succeq^u G'[\alpha] \) implies \( E = G[\alpha] \) where \( G' \succeq^u G \), so by hypothesis of induction then \( G \longrightarrow^* H \) for some \( H \succeq H' \). By definition of \( \succeq \) we have \( F := H[\alpha] \succeq H'[\alpha] = F' \) and by proposition 8.1.1 also \( E = G[\alpha] \longrightarrow^* H[\alpha] = F \). We will therefore just look at the ordinary rules for \( \longrightarrow \).
Lemma 9.4.11 Suppose \( E, E' \in RCL_\Omega^{\text{rec}} \) and \( A \in G \). Then \( E \preceq u E' \xrightarrow{A} F' \) implies \( \exists F. E \xrightarrow{A} F \succeq F' \)

Proof By induction on the size of the step \( E' \xrightarrow{A} F' \). The proof follows exactly the line of the previous lemma, except that we do not have to use proposition \( \text{rec} \) 8.1.1.

Up til now we have showed that if \( E \) is the approximation of \( F \) then \( F \) can do all the sequences \( E \) can. Now we take the opposite angel and show that if a (possible recursive) process is able to perform a sequence, then there is a syntactic finite approximation which also can do this sequence.

Lemma 9.4.12 Suppose \( E \in RCL_\Omega^{\text{rec}} \). Then
\[
E \xrightarrow{\hat{\Rightarrow}} F \succeq F'' \in RCL_\Omega \text{ implies } \exists E', F' \in RCL_\Omega. E \succeq E' \xrightarrow{\hat{\Rightarrow}} F' \succeq F''
\]

Proof By induction on the size of \( \xrightarrow{\hat{\Rightarrow}} \). In the basic case we have \( E = F \) and can choose \( E' = F' = F'' \). In the inductive step there as usual are two maincases:

- \( E \rightarrow G \xrightarrow{\hat{\Rightarrow}} F \succeq F'' \), (where \( \rightarrow = \rightarrow^* \) and the length of \( \rightarrow^* \) is less than that of \( \hat{\Rightarrow} \))
- By hypothesis of induction there are \( G', H \in RCL_\Omega \) such that \( G \succeq G' \xrightarrow{\hat{\Rightarrow}} H \succeq F'' \).

Now \( E \rightarrow G \succeq G'' \xrightarrow{\hat{\Rightarrow}} H \) implies by lemma 9.4.14 below the existence of \( E', G'' \in RCL_\Omega \) with \( E \succeq E' \rightarrow^* G'' \succeq G' \). We can then use lemma 9.4.5 on \( G'' \succeq G' \xrightarrow{\hat{\Rightarrow}} H \) to find an \( F' \) which fulfills \( G'' \xrightarrow{\hat{\Rightarrow}} F' \succeq H \). Collecting the facts so far we have \( E \succeq E' \rightarrow^* G'' \xrightarrow{\hat{\Rightarrow}} F' \succeq H \succeq F'' \) and so \( E \succeq E' \xrightarrow{\hat{\Rightarrow}} F' \succeq F'' \). For \( E' \in RCL_\Omega \) we easily prove \( E' \xrightarrow{\hat{\Rightarrow}} F' \) implies \( F' \in RCL_\Omega \) so this case is settled.
Lemma 9.4.13 For $E \in RCL^{rec}_\Omega(X)$ we have:

a) $\vdash \leq E$ iff $E = \vdash$

b) If $E \neq \Omega$ then for $\odot \in \{; \oplus, \|\}$, $E \leq F_0 \odot F_1$ implies $E = E_0 \odot E_1$ where $E_0 \leq F_0$ and $E_1 \leq F_1$

c) If $E \neq \Omega$ then $E \preceq F'[\varrho]$ implies $E = E'[\varrho]$ where $E' \preceq F'$

Proof Each implication is proven by a simple induction on the number of rules (from the definition of $\leq$) used to prove the left hand side of the implication.

We should mention that e.g., $E \preceq F'[\varrho]$ implies $F'[\varrho] \in RBL^{rec}_\Omega(X)$, because $\preceq$ only is defined on $RCL^{rec}_\Omega(X)$ and expressions only can be of this form when $F' \in RBL^{rec}_\Omega(X)$. Also notice that the opposite implications of b) – c) does not hold in general. E.g., from $E_0 \parallel E_1 \leq F$ one cannot deduce that $F$ is of the form $F = F_0 \parallel F_1$ where $E_0 \leq F_0$ and $E_1 \leq F_1$ because $F$ might equal $rec x. G$ where $G[rec x. G/x] = E_0 \parallel E_1$. The only reason we can deduce something about $E$ in a) when $\vdash \leq E$ is because all recursive expressions are from $RBL^{rec}_\Omega(X)$ and because $\vdash \not\in RBL^{rec}_\Omega(X)$.

Lemma 9.4.14 If $E \in RCL^{rec}_\Omega$ then

$E \rightarrow F \succeq F'' \in RCL_\Omega$ implies $\exists E', F' \in RCL_\Omega. E \succeq E' \rightarrow^* F' \preceq F''$

Proof If $F'' = \Omega$ the lemma follows by choosing $E' = F' = \Omega \in RCL_\Omega$. Hence we will assume $F'' \neq \Omega$ when proving the lemma by induction on the size, $m$, of $E \rightarrow^m F$.

We assume the lemma holds for $m$ when proving it for $m + 1$ by considering the different rules.

$E = \Omega \rightarrow_{m+1} \Omega = F \succeq F'':$ This can only mean $F'' = \Omega$ so we can choose $E' = F' = \Omega$

$E = E_0 \n E_1 \rightarrow_{m+1} F \succeq F'':$ There are two subcases:

$E_0 = \vdash$ and $F = E_1$: Let $E' = \vdash$; $F'' \in RCL_\Omega$ and $F' = F''$.

$F = F_0 \n E_1$ where $E_0 \rightarrow_m F_0$: We assume $F'' \neq \Omega$ so by lemma 9.4.13 we know that $F'' \leq F_0$; $E_1$ implies $F'' = F_0''$; $E_1''$ for some $F_0'' \leq F_0$ and $E_1'' \leq E_1$. By hypothesis of induction there are $E_0', F_0' \in RCL_\Omega$ with $E_0' \geq E_0 \rightarrow^* F_0' \geq F_0''$. Because $F'' \in RCL_\Omega$ implies $F'' \in RBL_\Omega$ we then have $E' := E_0'; E_1'' \in RCL_\Omega$ and $F' := F_0'; E_1'' \in RCL_\Omega$; $E'' = F_0'; E_1'' \leq F_0'; E_1'' \leq E_1; E_1' \leq E_0'; E_1' \leq E_0; E_1' \leq E_0; E_1' \leq E_0; E_1' \leq E_0; E_1' \leq E_0; E_1' \leq E_0.$ So as $F'' = F_0''; E_1'' \leq F_0''; E_1'' = F'$. Therefore $E = E_0 \n E_1 \rightarrow_{m+1} F \succeq F'':$ W.l.o.g. we just consider the case where $F = E_0$. Then let $E' = F'' \n \Omega \in RCL_\Omega$ and $F = F''$. Clearly $E' = F'' \n \Omega \succeq F \n E_1 = E$ and $E' \rightarrow F'' \succeq F''$. Let $F' = F''$. 

$E \rightarrow F \succeq F'':$ Similar but using lemma 9.4.15 in place of lemma 9.4.14.
Lemma 9.4.15 If $E \in RCL_\Omega^r$ and $A \in G$ then

$$E \xrightarrow{A} F \succeq F'' \in RCL_\Omega \text{ implies } \exists E', F' \in RCL_\Omega. E \succeq E' \xrightarrow{A} F' \succeq F''$$

Proof  At first the lemma is proven for the case $F'' \neq \Omega$. This will be done by induction on the size, $m$, of $E \xrightarrow{A}^m F$. As usual only the inductive step needs attention. We consider each rule in turn under the assumption $F'' \neq \Omega$ and that the lemma holds for $m$.

$$E = E_0 \parallel E_1 \xrightarrow{A} F \succeq F'' \parallel F_1$$

Clearly $A = a$ and $F'' = \parallel$. Choose $E' = a \in RCL_\Omega$ and $F' = \parallel \in RCL_\Omega$.

$$E = E_0 \parallel E_1 \xrightarrow{A}^m F_0 \parallel F_1$$

where $E_0 \xrightarrow{A}^m F_0 \parallel \Omega \neq F'' \succeq F_0 \parallel F_1$, implies by lemma 9.4.13 $F'' = F_0'' \parallel F_1''$ where $F_0 \succeq F_0'' \in RCL_\Omega$ and $E_1 \succeq E_1'' \in RCL_\Omega$. By induction then $\exists F_0'' \in RCL_\Omega, E_0 \succeq E_0'' \xrightarrow{A} F_0'' \succeq F_0''$. Letting $E' = E_0'' \parallel E_1''$ we have $E' \in RCL_\Omega$ and $E' \succeq E_0'' \parallel E_1 \succeq E$ and using the same inference rule finally $E' = E_0'' \parallel E_1'' \xrightarrow{A} F_0'' \parallel F_1'' = F''$ and also $F'' = F_0'' \parallel F_1'' \succeq F''$.

$$E = E_0 \parallel E_1 \xrightarrow{A}^m F_0 \parallel F_1 \succeq F'' \parallel F_1$$

The two rules where only one of the components of $E$ is involved are handled similar/ symmetric as above. If $E \xrightarrow{A}^m F$ steams from the remaining inference rule we must have $A = A_0 \times A_1$ and $F'' \succeq F_0'' \parallel F_1''$.
where for \( i = 0, 1 \), \( E_i \xrightarrow{A_i} F_i \geq F''_i \) and \( A_i \in G \). Like above we can apply the hypothesis of induction on each component and since \( A_0 \times A_1 = A \in G \) we can use the same rule again to obtain the result in a similar fashion.

Now from the rules of \( \xrightarrow{A} \) obviously \( E \xrightarrow{A} F \) only if \( \dagger \) occurs in \( F \). By structural induction on \( F \) an \( F'' \in RCL_\Omega \) can then be found such that \( F \geq F'' \neq \Omega \). As above appropriate \( E', F' \in RCL_\Omega \) are found for \( F'' \). When \( F'' = \Omega \) we have \( F'' \geq F'' \) so this case is dealt with too.

The two key lemmas 9.4.12 and 9.4.5 enables us to establish the important properties which shall bring the different preorders in connection with our denotational models.

**Proposition 9.4.16** \( \preceq_G \) and \( \succeq_G \) extends \( \preceq \) on \( RBL^{rec}_\Omega \).

**Proof** We shall show that when \( \preceq \) is restricted to \( RBL^{rec}_\Omega \) then \( \preceq \subseteq \preceq_G \) and \( \preceq \subseteq \succeq_G \). So let \( E, F \in RBL^{rec}_\Omega \) be given such that \( E \preceq F \).

\( \preceq_G \): Assume \( E \Rightarrow \dagger \). By lemma 9.4.5 there is an \( F' \) such that \( F \Rightarrow F' \geq \dagger \). From lemma 9.4.13 \( \dagger \preceq F' \) only if \( F' = \dagger \).

\( \succeq_G \): Immediate from lemma 9.4.5.

**Proposition 9.4.17** Given \( E \in RBL^{rec}_\Omega \) then

a) \( E \Rightarrow \dagger \) iff \( \exists E' \in \text{Fin}(E). E' \Rightarrow \dagger \)

b) \( E \Rightarrow \dagger \) iff \( \exists E' \in \text{Fin}(E). E' \Rightarrow \dagger \)

**Proof** By definition \( E' \in \text{Fin}(E) \) means \( E' \preceq E \) and \( E' \in RBL_\Omega \), so the if part of a) and b) are just special cases of the previous proposition. Only if:

a) Suppose \( E \Rightarrow \dagger \). Because \( \dagger \preceq \dagger \) we can use lemma 9.4.12 to find \( E', F' \in RCL_\Omega \) such that \( E \preceq E' \Rightarrow F' \geq \dagger \). \( \dagger \preceq F' \) only if \( F' = \dagger \) so this means \( E \preceq E' \Rightarrow \dagger \). Now \( E' \preceq E \in RBL^{rec}_\Omega \) clearly implies \( E' \in RBL^{rec}_\Omega \) wherefore we from \( E' \in RCL_\Omega \) deduce \( E' \in RBL_\Omega \) and thus \( E' \in \text{Fin}(E) \).

b) Suppose \( E \Rightarrow \dagger \). This means \( E \Rightarrow F \) for some \( F \in RCL^{rec}_\Omega \). Using \( F \preceq \Omega \) the rest goes as under a).

With the last to propositions it is easy to prove the main result of this section:

**Theorem 9.4.18** The preorders \( \preceq_G \) and \( \succeq_G \) over \( RBL^{rec}_\Omega \) are algebraic.

**Proof** The preorder \( \preceq_G \) is proved algebraic in exactly the same way as we now will prove \( \succeq_G \) algebraic. For \( \succeq_G \) we shall prove

\[ E \preceq_G F \text{ iff } \forall E' \in \text{Fin}(E) \exists F' \in \text{Fin}(F). E' \preceq_G F' \]

239
if: Assume the right hand side holds and let an \( s \in G^* \) be given such that \( E \xrightarrow{\delta} \). We prove \( F \xrightarrow{\delta} \). By proposition 9.4.17 above there is an \( E' \in \text{Fin}(E) \) with \( E' \xrightarrow{\delta} \). By assumption there is also an \( F' \in \text{Fin}(F) \) such that \( E' \subseteq G F' \). Hence \( F' \xrightarrow{\delta} \) and using the same proposition again then \( F \xrightarrow{\delta} \).

only if: Assume \( E \not\subseteq G F \) and let a \( E' \in \text{Fin}(E) \) be given.
At first we show that for each \( s \in G^* \) such that \( E' \xrightarrow{\delta} \) we can pick an \( F_s \in \text{Fin}(F) \) with \( F_s \xrightarrow{\delta} \). Suppose \( E' \xrightarrow{\delta} \). Applying the previous proposition we see that \( E \xrightarrow{\delta} \) and from the assumption then also \( F \xrightarrow{\delta} \). Using the proposition once more brings us an \( F' \in \text{Fin}(F) \) such that \( F' \xrightarrow{\delta} \). Let \( F_s \) be one of these \( F' \)’s.

Now for any \( H \in BL_\Omega \) it is an easy matter to prove by induction on the structure of \( H \) that \( \{ s \in G^* \mid H \xrightarrow{\delta} \} \) is finite. By proposition 9.3.2 we have \( \{ s \in G^* \mid E' \xrightarrow{\delta} \} = \{ s \in G^* \mid E\sigma \xrightarrow{\delta} \} \), so because \( E\sigma \in BL_\Omega \) we conclude \( \{ F_s \in \text{Fin}(F) \mid E' \xrightarrow{\delta} \} \) is finite too.

\( \text{Fin}(F) \) is directed so there is an ub \( F' \in \text{Fin}(F) \) for \( \{ F_s \mid E' \xrightarrow{\delta} \} \). By proposition 9.4.16 \( \subseteq \subseteq \subseteq_G \) this therefore means that for every \( F_s \), \( F' \) can perform \( s \). But there is exactly one \( F_s \) for each \( E' \xrightarrow{\delta} \) wherefore we conclude \( E' \subseteq_G F' \) as desired.

9.4.2 The Syntactic Finite Sublanguages

In this subsection look at how the full abstractness results for \( BL \) and \( RBL \) can be carried over to \( BL_\Omega \) and \( RBL_\Omega \).

### Theorem 9.4.19

The following denotatons are fully abstract:

- a) \( A_G[\_] \) on \( BL_\Omega \) w.r.t. \( \subseteq_G \)
- b) \( A_{or}[\_] \) on \( RBL_\Omega \) w.r.t. \( \subseteq_w \)
- c) \( A_G^c[\_] \) on \( BL_\Omega \) w.r.t. \( \subseteq_G \)
- d) \( A_{or}^c[\_] \) on \( RBL_\Omega \) w.r.t. \( \subseteq_w \)

**Proof**

- a) Since \( A_G[\_] \) is defined compositionally and the operators are monotone, \( \triangledown_G \) is a precongruence w.r.t. \( BL_\Omega \). a) then follows from proposition 9.4.23 below.

- b) By definition \( \subseteq_w \subseteq RBL_\Omega \times RBL_\Omega \) is a precongruence w.r.t. the combinators of \( RBL_\Omega \). We then just have to show \( \triangledown_{or} = \subseteq_w^c \). By proposition 9.1.11 this follows if we can prove for all \( E_0, E_1 \in RBL_\Omega \)

\[
E_0 \triangledown_{or} E_1 \text{ iff } \forall \text{RBL}_\Omega\text{-contexts } C. C[E_0] \subseteq_w C[E_1]
\]

*only if:* Assume \( E_0 \triangledown_{or} E_1 \) and let a \( \text{RBL}_\Omega \) context, \( C \), be given. By the compositional nature of \( A_{or}[\_] \) and the monotonicity of the \( A_{or} \) operators it follows that \( \triangledown_{or} \) is a precongruence w.r.t. the combinators of \( RBL_\Omega \). Hence also \( C[E_0] \triangledown_{or} C[E_1] \) or equally \( A_{or}[C[E_0]] \subseteq A_{or}[C[E_1]] \). From the \( \subseteq \)-monotonicity of \( \delta_w \) then \( \delta_w(A_{or}[C[E_0]]) \subseteq \delta_w(A_{or}[C[E_1]]) \) which by proposition 9.4.23 implies \( C[E_0] \subseteq_w C[E_1] \).

240
if: Assume $E_0 \not\preceq A_\sigma E_1$ or equally $A_\sigma[E_0] \not\subseteq A_\sigma[E_1]$. From lemma 9.4.27 we see there is a $RBL_\Omega$-context, $\mathcal{C}$, such that $\delta_\mathcal{C}(A_\sigma[\mathcal{C}[E_0]]) \not\subseteq \delta_\mathcal{C}(A_\sigma[\mathcal{C}[E_1]])$. Then also $\mathcal{C}[E_0] \not\preceq \mathcal{C}[E_1]$ by proposition 9.4.25.

c) Similar as b) with $\mathcal{A}_\sigma^p$ instead of $\mathcal{A}_\sigma$ and only $BL_\Omega$ contexts are concerned. Proposition 9.4.23 is used in stead of proposition 9.4.25 ($\delta_\mathcal{C}$ does not appear). With $A = L(E_1)$ the $BL_\Omega$-context for the $if$ part is found from lemma 9.4.26.

d) Similar to b) using lemma 9.4.28 to find the $RBL_\Omega$-context in the $if$ part.

\[\square\]

From proposition 9.4.23, proposition\(\Omega\) 8.1.2 and proposition 9.3.2 so as proposition 9.2.17 and the theorem above we deduce the $RBL_\Omega$ equivalent of corollary 8.4.1 of section 8.4:

**Corollary 9.4.20** For all $E_0, E_1 \in RBL_\Omega$:

\[
\begin{align*}
E_0 \preceq_w E_1 & \iff A_\sigma^w[E_0] = A_\sigma^w[E_1] & E_0 \preceq_G E_1 & \iff A_\sigma^p[E_0] = A_\sigma^p[E_1] \\
E_0 \preceq_\omega E_1 & \iff A_\sigma^G[E_0] = A_\sigma^G[E_1] & E_0 \preceq_\omega E_1 & \iff A_\sigma^p[E_0] = A_\sigma^p[E_1]
\end{align*}
\]

With the same argumentation as in section 8.4 then also:

**Corollary 9.4.21** For $E_0, E_1 \in RBL_\Omega$ we have

\[(E_0 \preceq^e_w E_1) \Rightarrow (E_0 \preceq G E_1) \Rightarrow (E_0 \preceq_w E_1)\]

and

\[(E_0 \preceq^e_w E_1) \Rightarrow (E_0 \preceq G E_1) \Rightarrow (E_0 \preceq_w E_1)\]

**Theorem 9.4.22**

a) $BL_\Omega$ is $\{E \in BL_\Omega \mid L(E) \subseteq A\}$-expressive w.r.t. $\preceq_G$ for every finite subset $A$ of $\Delta$.

b) $RBL_\Omega$ is expressive w.r.t. both $\preceq_w$ and $\preceq^e_w$.

**Proof**

a) Suppose $A \subseteq \Delta$ is finite and $E_0 \in BL_\Omega$. Let $\mathcal{C}$ be the $BL_\Omega$-context, $\# ; e$, from lemma 9.4.26. Given an $E_1$ with $L(E_1) \subseteq A$ we show

\[E_0 \preceq_G E_1 \iff \mathcal{C}[E_0] \preceq_G \mathcal{C}[E_1]\]

only if: Since $\preceq_G$ by definition is a precongruence it follows that $\mathcal{C}[E_0] \preceq_G \mathcal{C}[E_1]$.

Again by definition of $\preceq_G$ also $\preceq_G \subseteq \preceq_G$.

if: $\mathcal{C}[E_0] \preceq_G \mathcal{C}[E_1]$ \[\Rightarrow A_\sigma^p[\mathcal{C}[E_0]] \subseteq A_\sigma^p[\mathcal{C}[E_1]]\] \[\Rightarrow A_\sigma^p[E_0] \subseteq A_\sigma^p[E_1]\] \[\Rightarrow E_0 \preceq_G E_1\]

by proposition 9.4.23

\[\Rightarrow A_\sigma^p[E_0] \subseteq A_\sigma^p[E_1]\]

by choice of $\mathcal{C}$

\[\Rightarrow E_0 \preceq_G E_1\]

definition of $\mathcal{A}_\sigma^p$

\[\Rightarrow E_0 \preceq_G E_1\]

by the theorem above

\[241\]
b) Similar as a) but using the $RBL_{\Omega}$-context $C = \#[\hat{e}]$ found by lemma 9.4.27 when dealing with $\leq_{w}$ and $C = \#[\hat{e}]$; $e$ from lemma 9.4.28 when concerned with $\leq_{w}$. In both cases proposition 9.4.25 is used in place of proposition 9.4.23.

\[\square\]

**Proposition 9.4.23** For every $E_0, E_1 \in BL_{\Omega}$:

a) $A_G[E_0] \subseteq A_G[E_1]$ iff $E_0 \leq_G E_1$

b) $A_G^p[E_0]_1 \subseteq A_G^p[E_1]_1$ iff $E_0 \leq_G E_1$

**Proof**

a) This is nothing more than the extension of proposition 7.4.3 to $BL_{\Omega}$. So of course a) holds if we can manage to extend lemma 7.4.4 to $BL_{\Omega}$ obtaining lemma$_{\Omega}$ 7.4.4. We will just comment on the main spots where the proof change:

*only if:* One additional case: $E = \Omega$: The implication holds trivially because $\Omega \Downarrow \ldots \Downarrow F$ implies $F = \Omega \neq \uparrow$.

*if:* If $E = \Omega$ then $\varphi(E) = \emptyset$ and we cannot have have $p \in \varphi(E)$. As above we can also here take over the corresponding proof for the other cases if we use lemma$_{\Omega}$ 7.4.5 and lemma$_{\Omega}$ 7.4.6.

b) Along the lines of the proof of proposition 7.4.3 in chapter 7 one from lemma 9.4.24 below for any $E \in BL_{\Omega}$ see:

\[\delta_G(\varphi^p_1(E)) = \{s \in G^* \mid E \Downarrow s\}\]

From proposition 9.2.14 $A_G^p[E]_1 = \delta_G(\varphi^p_1(E))$ so the proposition then follows by the definition of $\leq_G$.

\[\square\]

As the extended canonical map $\varphi$ (by definition) agrees with $\varphi^p$ we can use lemma$_{\Omega}$ 7.4.4 directly in the the proof of the next lemma. The same notation will also be used.

**Lemma 9.4.24** Given $E \in BL_{\Omega}$ and $A_1, \ldots, A_n \in G_{\varepsilon}$ ($n \geq 1$). Then

\[E \Downarrow A_1 \ldots \Downarrow A_n \text{ iff } \exists p \in \varphi^p_1(E). A_1 \ldots A_n \leq p\]

**Proof** Here we also start out by observing that any subexpression of a $BL_{\Omega}$ expression itself is from $BL_{\Omega}$.

*if:* By induction on the structure of $E$.

$E = \Omega$: $\varphi^p_1(\Omega) = \{\varepsilon\}$ and we must have $p = \varepsilon$ and all $A_1, \ldots, A_n$ equal to $\emptyset$. Since for for every $E$, $E \rightarrow^0 E$ clearly $E \Downarrow \ldots \Downarrow \emptyset$. 

242
\( E = a: \varphi^\Omega_1(a) = \{a, \varepsilon\}. \) There are two possibilities for \( p \)—either \( p = \varepsilon \) or \( p = a \). The former case goes as above and the latter as in the corresponding case of lemma \( \Omega 7.4.4. \)

\( E = E_0; E_1: \varphi^\Omega_1(E) = \varphi^\Omega_1(E_0) \cup \varphi^\Omega_2(E_0) \cdot \varphi^\Omega_1(E_1). \) If \( p \in \varphi^\Omega_1(E_0) \) the result follows from hypothesis of induction and proposition \( \Omega 7.2.3. \) Otherwise \( p \) must equal \( p_0 \cdot p_1 \) where \( p_0 \in \varphi^\Omega_2(E_0) \) and \( p_1 \in \varphi^\Omega_1(E_1). \) If \( p_1 = \varepsilon \) then \( p = p_0 \cdot \varepsilon = p_0 \in \varphi^\Omega_2(E_0) \) and the rest follows from lemma \( \Omega 7.4.4. \) We can then assume to be in the situation where \( A_1 \ldots A_n \preceq p_0 \cdot p_1 \) and \( p_1 \neq \varepsilon. \) \( p_0 \neq \varepsilon \) because \( p_0 \in \varphi^\Omega_2(E_0). \) By lemma \( \Omega 7.4.7 \) then \( n \geq 2 \) and there exists a \( 1 \leq j < n \) such that \( A_1 \ldots A_j \preceq p_0 \) and \( A_{j+1} \ldots A_n \preceq p_1. \) Since \( p_0 \in \varphi^\Omega_2(E_0) \) we can use lemma \( \Omega 7.4.4. \) to get \( E_0 \xrightarrow{\text{A}_j} \ldots \xrightarrow{\text{A}_1} \xrightarrow{\text{A}_j} \xrightarrow{\text{A}_1}. \) From \( A_{j+1} \ldots A_n \preceq p_1 \in \varphi^\Omega_1(E_1) \) we by hypothesis of induction also have \( E_1 \xrightarrow{\text{A}_{j+1}} \ldots \xrightarrow{\text{A}_1} \). Applying proposition \( \Omega 7.2.3 \) we finally get \( E_0; E_1 \xrightarrow{\text{A}_1} \ldots \xrightarrow{\text{A}_j} \xrightarrow{\text{A}_1} \xrightarrow{\text{A}_j}. \)

\( E = E_0 \oplus E_1 \) and \( E = E_0 \parallel E_1: \) On expressions of this form \( \varphi^\Omega_1 \) is defined like \( \varphi^\Omega_2 \) so the arguments are identical to those of lemma \( \Omega 7.4.4. \)

**only if:** Also by induction on the structure of \( E. \)

\( E = \Omega: \) \( \Omega \) can only perform internal steps wherefore all \( A_1 \) through \( A_n \) must equal \( \emptyset \) or by the alternative notation equal \( \varepsilon. \) But \( \varepsilon \cdot \ldots \cdot \varepsilon \preceq \varepsilon \in \{\varepsilon\} = \varphi^\Omega_1(\Omega). \)

\( E = a: \) \( a \) cannot do any internal steps and if \( a \xrightarrow{\text{A}} F \) then \( A = a \) and \( F = \top. \) \( \top \) can do no steps at all so we conclude all \( A_1, \ldots, A_n \) must equal \( \varepsilon (= \emptyset) \) except for at most one which then only can be \( a. \) If all equals \( \varepsilon \) then \( A_1 \cdot \ldots \cdot A_n = \varepsilon \preceq \varepsilon \in \{\varepsilon, a\} = \varphi^\Omega_1(a). \) Ohterwise \( A_1 \cdot \ldots \cdot A_n = a \in \varphi^\Omega_1(a) \).

\( E = E_0; E_1: \) Assume \( E_0; E_1 \xrightarrow{\text{A}_1} \ldots \xrightarrow{\text{A}_j} F. \) We want to use lemma \( \Omega 7.4.5 \) which mentions four ways such a sequence could be obtained. The first presuppose \( E_0 \xrightarrow{\text{A}_1} \ldots \xrightarrow{\text{A}_j} \top \) so it can be excluded because \( E_0 \in BL_\Omega. \) The remaining three can for our purpose be summarized in two:

\[
E_0 \xrightarrow{\text{A}_1} \ldots \xrightarrow{\text{A}_j} \xrightarrow{\text{A}_1} F \quad \text{for a } 1 \leq j < n
\]

\[
E_0 \xrightarrow{\text{A}_1} \ldots \xrightarrow{\text{A}_j} F^* \quad \text{for a } 1 \leq j < n
\]

In the latter case we can apply the hypothesis of induction to find a \( p \in \varphi^\Omega_1(E_0) \) such that \( A_1 \cdot \ldots \cdot A_n \preceq p. \) As \( \varphi^\Omega_1(E_0) \subseteq \varphi^\Omega_2(E_0; E_1) \) this case is settled. In the former case we can use lemma \( \Omega 7.4.4. \) to find a \( p_0 \in \varphi^\Omega_2(E_0) \) with \( A_1 \cdot \ldots \cdot A_j \preceq p_0 \) and by induction there is a \( p_1 \in \varphi^\Omega_1(E_1) \) such that \( A_{j+1} \cdot \ldots \cdot A_n \preceq p_1. \) From the \( \preceq \)-monotonicity of \( \cdot \) we then deduce \( A_1 \cdot \ldots \cdot A_n \preceq p_0 \cdot p_1 \in \varphi^\Omega_2(E_0) \cdot \varphi^\Omega_1(E_0) \subseteq \varphi^\Omega_2(E_0; E_1) \) as we want.

\( E = E_0 \oplus E_1 \) and \( E = E_0 \parallel E_1: \) Similar arguments as in lemma \( \Omega 7.4.4. \)

\( \square \)

**Proposition 9.4.25** For all \( E_0, E_1 \in RBL_\Omega: \)

a) \( \delta_{w}(A_{or}[E_0]) \subseteq \delta_{w}(A_{or}[E_1]) \) iff \( E_0 \preceq_{w} E_1 \)
\[ \delta_w(A^p_{or}[E_0]) \subseteq \delta_w(A^p_{or}[E_1]) \iff E_0 \subseteq_w E_1 \]

**Proof** a) follows with exactly the same arguments as we now show b). b) follows from the definition of \( \subseteq_w \) and the general deduction (\( E \in RBL_\Omega \)).

\[ \delta_w(A^p_{or}[E_1]) = A^p_{or}[E\sigma] \]

\[ = \delta_w(\rho^p(E\sigma)) \]

\[ = \{ w \in W \mid E\sigma \Rightarrow \} \]

\[ = \{ w \in W \mid E \Rightarrow \} \]

\[ \text{by (9.10)} \]

\[ \text{proposition 9.3.2} \]

\[ \Box \]

**Lemma 9.4.26** Given an expression \( E_0 \in BL_\Omega \) and a finite subset \( A \) of \( \Delta \). Then there is an \( e \in \Delta \) such that for all \( E_1 \in BL_\Omega \) with \( L(E_1) \subseteq A \) we have

\[ A^p_G[E_0] \not\subset A^p_G[E_1] \Rightarrow A^p_G[E_0 ; e] \not\subset A^p_G[E_1 ; e] \]

**Proof** Let a \( E_0 \in BL_\Omega \) be given. \( L(E_0) \) is finite and contains \( L(A^p_G[E_0]) = \cup \{ L(p) \mid p \in A^p_G[E_0]_1 \cup A^p_G[E_0]_2 \} \)

\[ a \in L(p) \iff m_p(a) \neq 0 \]

So since \( A \) is finite too, but \( \Delta \) infinite, we can choose an \( e \in \Delta \) that does occur in \( L(A^p_G[E_0]) \) or \( A \). Before we start out proving that this \( e \) meets the requirement observe that for any \( E \in BL_\Omega \) we have:

\[ A^p_G[E ; e]_1 = A^p_G[E]_1 \cup A^p_G[E]_2 \cdot \{ e, e \} \]

by definition of \( p^e_G \) and \( e_G \)

\[ = A^p_G[E]_1 \cup A^p_G[E]_2 \cup A^p_G[E]_2 \cdot \{ e \} \]

\[ = A^p_G[E]_1 \cup A^p_G[E]_2 \cdot \{ e \} \]

\[ \subseteq A^p_G[E] \]

Now let an \( E_1 \in BL_\Omega \) be given such that \( L(E_1) \subseteq A \) and \( A^p_G[E_0] \not\subset A^p_G[E_1] \). There are two ways how this can be:

\[ A^p_G[E_0]_2 \not\subset A^p_G[E_1]_2 \]

I.e., there is a \( p \in A^p_G[E_0]_2 \) not in \( A^p_G[E_1]_2 \). By the observations above then \( p \cdot e \not\in A^p_G[E_0]_1 \). \( p \cdot e \not\in A^p_G[E_1]_1 \) because \( L(A^p_G[E_1]_1) \subseteq L(E_1) \subseteq A \) and \( e \not\in A \). \( p \not\in A^p_G[E_1]_2 \) implies \( p \cdot e \not\in A^p_G[E_1]_2 \). \( \{ e \} \) so from the observations above we conclude \( p \cdot e \not\in A^p_G[E_1 ; e]_1 \).

\[ A^p_G[E_0]_1 \not\subset A^p_G[E_1]_1 \]

Then let a \( p \in A^p_G[E_0]_1 \) be given such that \( p \not\in A^p_G[E_1]_1 \). By the observations \( p \in A^p_G[E_0 ; e]_1 \). We have \( p \not\in A^p_G[E_1 ; e] \). \( \{ e \} \) because \( p \in A^p_G[E_1 ; e] \) would imply \( e \in L(p) \subseteq L(A^p_G[E_0]) \) contradicting the way \( e \) is chosen. Hence \( p \not\in A^p_G[E_1 ; e]_1 \).

\[ \Box \]

**Lemma 9.4.27** Given an expression \( E_0 \in RBL_\Omega \). Then there is a refinement combinator, \([\varphi] \), such that

\[ \forall E_1 \in RBL_\Omega \quad A_{or}[E_0] \not\subset A_{or}[E_1] \Rightarrow \delta_w(A_{or}[E_0[\varphi]]) \not\subset \delta_w(A_{or}[E_1[\varphi]]) \]

**Proof** From chapter 8 we already get the corresponding result for the \( M_{or} \)-model, but for the language \( RBL \). All what can happen when \( \Omega \) is added to the language is that \( A_{or}[E_0] \) might be empty in which case the implication holds vacuously. \( \Box \)
Lemma 9.4.28 Given an expression $E_0 \in RBL_\Omega$. Then there is a refinement combinator, $\varrho$, and an action $e \in \Delta$ such that
\[
\forall E_1 \in RBL_\Omega. \ A_{or}^p[E_0] \not\subseteq A_{or}^p[E_1] \Rightarrow \delta_w(A_{or}^p[E_0[\varrho] ; e_1]) \not\subseteq \delta_w(A_{or}^p[E_1[\varrho] ; e_1])
\]

**Proof** Let $E_0 \in RBL_\Omega$ be given. As for $A_{or}[]$ we are after a fission refinement, $\varrho$, such that any pomset, $p$, associated with the denotation of $E_0$ can be reflected in a linearization of $q \in p <\varrho>$, but this time with the additional requirement that $e$ does not occur in any pomset which stems from a $<\varrho>$-refinement of a pomset associated with the denotation of an arbitrary $E_1 \in RBL_\Omega$. Since $E_1$ can be any syntactic finite expression there are practical no limitations on what singleton pomsets there may be in a pomset from its denotation. We can therefore just as well pick an arbitrary $e \in \Delta$ and seek a fission refinement $\varrho$ for $E_0$ such that

\[
(9.11) \quad \forall a \in \Delta. \ e \not\in L(\varrho(a))
\]

Let $m$ be the lub of the multiplicity functions associated with the pomsets of $A_{or}^p[E_0]$, i.e.,
\[
m = \bigvee \{m_p | p \in A_{or}^p[E_0]_1 \cup A_{or}^p[E_0]_2\} \text{ (finite because } E_0 \in RBL_\Omega\}.
\]

$\Delta \setminus \{e\}$ is (countable) infinite because $\Delta$ is, so from the arguments about the existence of fission refinements it should be clear we also can find a $m$-fission refinement $\varrho$ with desired property (9.11). Remember when dealing with fission refinements we use the same symbol for the $BL$-fission refinement and the $P(\mathcal{P})$-fission refinement.

Before we continue notice as in lemma 9.4.26 for any $E \in RBL_\Omega$
\[
A_{or}^p[E[\varrho] ; e_1] = A_{or}^p[E[\varrho]_1] \cup A_{or}^p[E[\varrho]_2] \cdot e
\]

Now let any $E_1 \in RBL_\Omega$ be given and suppose $A_{or}^p[E_0] \not\subseteq A_{or}^p[E_1]$. Assume on the contrary $\delta_w(A_{or}^p[E_0[\varrho] ; e_1]) \subseteq \delta_w(A_{or}^p[E_1[\varrho] ; e_1])$. There are two ways how $A_{or}^p[E_0] \not\subseteq A_{or}^p[E_1]$ can be:

$A_{or}^p[E_0]_2 \not\subseteq A_{or}^p[E_1]_2$: Then there is a $p \in A_{or}^p[E_0]_2$ with $p \not\in A_{or}^p[E_1]_2$. Since $A_{or}^p[E_0]_2$ is $\delta_{or}$-closed $p$ must have the $P_{or}$-property. Because $\varrho$ is $m$-fission refinement and $m_p \leq m$ we can use lemma 8.3.6 to find a $w \in \delta_w(p <\varrho>)$ which is $p$-reflecting. We then have:

\[
w \cdot e \in \delta_w(A_{or}^p[E_0[\varrho]_2 <\varrho>]) \cdot \{e\} = \delta_w(A_{or}^p[E_0[\varrho]_2 <\varrho>]) \cdot \{e\} \quad \delta_w \circ \delta_{or} = \delta_w
\]

\[
= \delta_w(A_{or}^p[E_0[\varrho]) \cdot \{e\} \quad \text{definition of } [\varrho]^p
\]

\[
= \delta_w(A_{or}^p[E_0[\varrho]) \cdot \{e\} \quad \delta_w \text{ distributes over } \cdot, \delta_w(\{e\}) = \{e\}
\]

\[
\subseteq \delta_w(A_{or}^p[E_0[\varrho]) \subseteq \delta_w(A_{or}^p[E_0[\varrho]) \subseteq \delta_w(A_{or}^p[E_1[\varrho]) \cdot \{e\} \quad \text{assumption}
\]

\[
\subseteq \delta_w(A_{or}^p[E_1[\varrho]) \cdot \{e\}
\]

\[
= \delta_w(A_{or}^p[E_1[\varrho]) \cup A_{or}^p[E_1[\varrho]) \cdot \{e\} \quad \text{from notice}
\]

\[
\delta_w(A_{or}^p[E_1[\varrho]) \cup \delta_w(A_{or}^p[E_1[\varrho]) \cdot \{e\}
\]

Because $A_{or}^p[E_1[\varrho])_1 = \delta_{or}(A_{or}^p[E_1[\varrho])_1)$ we see from (9.11) that $e \not\in L(\delta_w(A_{or}^p[E_1[\varrho])_1)$.

Hence also $w \cdot e \not\in \delta_w(A_{or}^p[E_1[\varrho])_1)$ and we are left with $w \cdot e \in \delta_w(A_{or}^p[E_1[\varrho])_2 \cdot \{e\}$. But then $w \in \delta_w(A_{or}^p[E_1[\varrho])_2 = \delta_w(\delta_{or}(A_{or}^p[E_1[\varrho])_2 <\varrho>))$. This means there is a $p_1 \in A_{or}^p[E_1]_2$ and $q \in p_1 <\varrho>$ such that $w \preceq q$. Since $w$ is $p$-reflecting we by lemma 8.3.5 get $p \preceq p_1$. Because $P_{or}(p)$ and $A_{or}^p[E_1]$ is $\delta_{or}$-closed this implies $p \in A_{or}^p[E_1]_2$—a contradiction.

$A_{or}^p[E_0]_1 \not\subseteq A_{or}^p[E_1]_1$: We see there exists a $p \in A_{or}^p[E_0]_1$ such that $p \not\in A_{or}^p[E_1]_1$ and $P_{or}(p)$ because $A_{or}^p[E_0]_1$ is $\delta_{or}$-closed (as well as $\pi$-closed). We can also here find a $p$-reflecting linearization $w \in \delta_w(p <\varrho>)$. Notice that because of (9.11) we have $e \not\in L(w)$. We infer:
\( w \in \delta_w(A^p_{or}[E_0]_1\langle \varrho \rangle) \subseteq \delta_w(\pi(A^p_{or}[E_0]_1\langle \varrho \rangle)) \)
\( = \delta_w(\delta_{or}\pi(A^p_{or}[E_0]_1\langle \varrho \rangle)) \)
\( \subseteq \delta_w(A^p_{or}[E_0[\varrho];e]_1) \)
\( \subseteq \delta_w(A^p_{or}[E_1[\varrho];e]_1) \)
\( = \delta_w(A^p_{or}[E_1[\varrho]]_1) \cup \delta_w((A^p_{or}[E_1]_1)<\varrho>) \)
\( \subseteq \delta_w(A^p_{or}[E_1[\varrho]]_1) \cup \delta_w((A^p_{or}[E_1]_1)<\varrho>) \cdot \{e\} \) as above
\( e \not\in L(w) \) excludes \( w \in \delta_w(A^p_{or}[E_1[\varrho]]_1) \cdot \{e\} \) and we are left with \( w \in \delta_w(A^p_{or}[E_1[\varrho]]_1) = \delta_w(\delta_{or}\pi((A^p_{or}[E_1]_1)<\varrho>)) = \delta_w(\pi((A^p_{or}[E_1]_1)<\varrho>). Then there must be pomsets such that
\( w \leq q \sqsubseteq q' \in p_1<\varrho> \) where \( p_1 \in A^p_{or}[E_1]_1 \)

\( w \) is the linearization of some pomset refined by \( <\varrho> \) and therefore must be balanced w.r.t. to the fission pairs of \( \varrho \). Because \( w \not\leq q \) they have the same labels and so \( q \) must be balanced w.r.t. to the fission pairs. With \( q \sqsubseteq q' \in p_1<\varrho> \) we can then use the lemma below to conclude there is a pomset \( p_1' \sqsubseteq p_1 \) such that \( q \in p_1'<\varrho> \). Because \( w \leq q \in p_1'<\varrho> \) and \( w \) is \( p \)-reflecting we can as in the case above conclude \( p \leq p_1' \).

Lemma 9.4.29 Let a finite multiplicity function \( m \) over \( \Delta \) be given together with a \( \varepsilon \)-free \( m \)-fission refinement \( \varrho \). Suppose \( p, q \) and \( r \) are pomsets such that \( p \sqsubseteq q \in r<\varrho> \). If \( p \) is balanced w.r.t. to the fission pairs of \( \varrho \) in the sense:
\[ \forall a \in \Delta \forall k \in n(m). m_p(a_{S_k}) = m_p(a_{S_k}) \]
then there is a pomset \( s \sqsubseteq r \) such that \( p \in s<\varrho> \).

Proof By definition of the refinement operator, \( q \in r<\varrho> \) means there is a \( \varrho \)-consistent p.ref., \( \pi_r \), for \( r \) such that \( q = [r<\pi_r>] \). Then also \( p \subseteq [r<\pi_r>] \).

We illustrate the situation by an example. Suppose \( r \) is the representative of the pomset
\[ a \rightarrow a \\
\]
\[ b \rightarrow a \]

Then \( [r<\pi_r>] \) typically may look like:
\[ a_{S_2} \rightarrow a_{F_2} \rightarrow a_{S_1} \rightarrow a_{F_1} \]
\[ b_{S_4} \rightarrow b_{F_4} \rightarrow a_{S_2} \rightarrow a_{F_2} \]

Evidently no matter how \( p \) is a \( (\leq r<\pi_r> \)-downwards closed) prefix of \( [r<\pi_r>] \) then for the fission pair \( a_{S_2}, a_{F_2} \) the number of times \( a_{S_2} \) occur in \( p \) must be greater than or equal the number of times \( a_{F_2} \) occur in \( p \). Similar for the other fission pairs. Clearly also if these numbers balance for every fission pair then there can be no element of \( p \) labelled say \( a_{S_1} \) without an immediate following element labelled \( a_{F_1} \). By the nature of fission refinement these two elements must originate from the same element in \( r \) and then \( p \) must be the refinement of a prefix of \( r \).
Now formally $p \sqsubseteq [r < \pi_r, s]$ by the alternative characterization of $\sqsubseteq$ implies that we can find a representative $p'$ of $p$ such that

$$p' = r < \pi_r |_{X_{p'}} \text{ and } X_{p'} \text{ is } \leq_{r < \pi_r} \text{-downwards closed}$$

Notice this implies $X_{p'} \subseteq X_{r < \pi_r}$. It then gives sense to define $Y = \{ x \in X_r | \langle x, x' \rangle \in X_{p'} \}$ and $s = r |_Y$.

At first we show $Y$ to be $\leq_r$-downwards closed thus gaining $s \sqsubseteq r$. Given an $x \in X_r$ and $y \in Y$ such that $x \leq_r y$. We shall show $x \in Y$. Now $y \in Y$ means there is some $y'$ with $\langle y, y' \rangle \in X_{p'}$. Because $\varrho$ is $\varepsilon$-free there must be an $x' \in X_{\pi_r(x)}$. By construction of $r < \pi_r$ then $\langle x, x' \rangle \in X_{r < \pi_r}$ and from $x \leq_r y$ also $\langle x, x' \rangle \leq_{r < \pi_r} \langle y, y' \rangle$. Because $X_{p'}$ is $\leq_{r < \pi_r}$-downwards closed this implies $\langle x, x' \rangle \in X_{p'}$ and by definition of $Y$ then $x \in Y$ as we want.

Now $Y \subseteq X_r$ so we let $\pi_s$ be $\pi_r |_Y$ which clearly is a $\varrho$-consistent particular fission refinement for $s$. We then wish to show

$$r < \pi_r |_{X_{p'}} = s < \pi_s$$

To do this we show at first that $X_{p'} = X_{s < \pi_s}$.

$\subseteq$: $\langle x, x' \rangle \in X_{p'} \subseteq X_{r < \pi_r}$ implies $x \in Y$ and $x' \in X_{\pi_r(x)}$. Since $\pi_s = \pi_r |_Y$ and $x \in Y$ we have $x' \in X_{\pi_s(x)}$ and by construction of $s < \pi_s$ also $\langle x, x' \rangle \in X_{s < \pi_s}$.

$\supseteq$: Calling in mind the observation we did on page 185 about $\varrho$-consistent particular fission refinements, we can especially for $\pi_r$ make the following deductions: If $x \in X_r$ then there are exactly two elements in $X_{r < \pi_r}$ with first component $x$, namely $x_S^x$ and $x_F^x$. Recall that $x_S^x$ is the $(x, x') \in X_{r < \pi_r}$ where $x'$ is that element of $X_{\pi_r(x)}$ with label $\ell_{\pi_r(x)}(x') = a_{S_h}$ where $a$ equals $\ell_r(x)$ and $k \in n(m)$. Similar for $x_F^x$ and we have seen $x_S^x < r < \pi_r, x_F^x$. So $x_F^x \in X_p$ implies $x_S^x \in X_{p'}$ because $X_{p'}$ is $\leq_{r < \pi_r}$-downwards closed. It follows that if there is one $x_S^x$ in $X_{p'}$ for some $x \in X_r$ without the corresponding $x_F^x$ in $X_{p'}$ then the number of $a_{S_h}$’s in $r < \pi_r |_{X_{p'}}$ must be at least one less than the number of $a_{F_h}$’s where $\ell_{r < \pi_r}(x_{F_r}) = a_{S_h}$. Hence $r < \pi_r |_{X_{p'}}$ cannot be ballanced w.r.t. the fission pairs of $\varrho$ if there is one $x_S^x$ in $X_{p'}$ without the corresponding $x_F^x$. By the proviso of the lemma $p' = r < \pi_r |_{X_{p'}}$ is balanced so we conclude there can be no such $x_S^x$’s.

We can now return to the question of $X_{s < \pi_s} \subseteq X_{p'}$. Suppose $\langle x, x' \rangle \in X_{s < \pi_s}$. This means $x \in Y$ and $x' \in X_{\pi_s(x)} = X_{\pi_r |_Y(x)} = X_{\pi_r(x)}$. $x \in Y$ implies by definition of $Y$ the existence of an $x''$ such that $\langle x, x'' \rangle \in X_{p'}$. If $x' = x''$ we are done immediately. Now $\langle x, x'' \rangle \in X_{p'} \subseteq X_{r < \pi_r}$ only if $x \in X_r$ and $x'' \in X_{\pi_r(x)}$, so when $x' \neq x''$ we get from $x \in X_r$ and $x' \in X_{\pi_r(x)}$, that either $\langle x, x' \rangle = x_S^x, \langle x, x'' \rangle = x_F^x$ or $\langle x, x' \rangle = x_F^x, \langle x, x'' \rangle = x_S^x$. As argued above we must have $\langle x, x' \rangle \in X_{p'}$ in the former case because $X_{p'}$ is $\leq_{r < \pi_r}$-downwards closed and in the latter also $\langle x, x' \rangle \in X_{p'}$, but this time because $X_{p'}$ is balanced w.r.t. to the fission pairs of $\varrho$.

Having proved $X_{p'} = X_{s < \pi_s}$, it is an easy matter to show $r < \pi_r |_{X_{p'}} = s < \pi_s$ from the definition of $s$, $\pi_s$ and $Y$. For instance w.r.t. labels we have: $\langle x, x' \rangle \in X_{p'}$ implies $x \in Y$ so $\ell_{r < \pi_r, s |_{X_{p'}}}(\langle x, x' \rangle) = \ell_{\pi_r |_Y}(x') = \ell_{\pi_r(x)}(x') = \ell_{\pi_s(x)}(x') = \ell_{s < \pi_s}(\langle x, x' \rangle)$—the last equality from $\langle x, x' \rangle \in X_{p'} = X_{s < \pi_s}$.
Collecting the facts we have $p' = s<\pi_s>$ so of course $p = p' = [s<\pi_s>]$. Since $\pi_s$ is $\rho$-consistent then also $p = [s<\pi_s>] \in s<\rho>$. We have already shown $s \sqsubseteq r$ wherefore the proof is completed. \qed
Index

Constants

#, 209
⊥, see least element
†, see extinct action
Δ, 168
∇, 168
ff, 168
tt, 168
Ω, 204

Functions

$A[]$, 206
$A_G[]$, $A^p_G[]$, 222
$A_{or}[]$, $A^p_{or}[]$, 222
$\llbracket G, 158$
$\llbracket_{or}, 181$
{$g}$, 174

Operations

$[g], 172$
$\|, 149$
⊕, 149
⊕-free, 153
; , 149
$A, 168$
$\triangleright, 168$
$\blacktriangleright, 191$
$\trianglerighteq, 191$
$<\pi_p>, see refinement, particular$
$<\omega>, see pomset, refinement of$
* , see pomset, sequential composition
×, see pomset, parallel composition

Relations

$\xrightarrow{A} G, 151, 152$
$\xrightarrow{\rightarrow} G, 151$
$\xrightarrow{a} G, 152$
$\xrightarrow{\leftarrow}, 131$
$\cong, see isomorphism of lpos$
$x_G, 153$
x_G, 152$
x_G, 173$
$\not\in_G, 229$
$\in_G, 203$
$\subseteq_G, 203$
$\trianglerighteq_G, 158$
$\trianglerighteq_{or}, 181$
$\subseteq, see pomset, prefix of$
\leq, see smoother than$
\preceq, see syntactic preorder$
\preceq^u, 233$
$\models_G, 168, 191$
$\models^w_G, 194$

Alphabetical

$A$, $A_G$, $A_{or}, 216, 217$
$A$-expressive, 212
algebraic (a relation being ∼), 205
approximation, 205
assignment, 204, 205, 207, 220
$\alpha^n, see pomset, multisingleton$
auto-parallelism, 190

Beh, 153
behaviours, 153
$BL, 149$
$BL$-refinement, 172
$BL^\text{rec}(X), BL^\text{rec}, BL^\Omega, 218$

$C[], see contexts$
characteristic context, 212
$CL, 151$
co, 131
compact elements, 205
configurations, 151
contexts, 209
co-set, 131
cpo, 205

DBL, 153
DCL, 153
Δ, 130, 149
δ, 137
δn, 137
dual of a formula, 168

ENV, see environment
evironment, 205
ε, see pomset, empty
experimenting, 149
expressions
closed, 204
syntactically finite, 204
expressive (a relation being ∼), 212
external computation step, 151
extinct action, 151

Fin, see syntactic finite approximations
Fin, see compact elements
finitary Σ-domain, 209
firing steps, 150
fission pairs, 186, 246
FV, see free variables

G, see tests, direct
ground set, 130

image finite, 141
internal computation step, 151
isomorphism of lpos, 131

k, see standard set

L, 190
ℓ, 130
L, 168, 191
Lg, 168
LG, 167
lr, 191
LG, 192
labelled poset, 130

labelling function, 130
least element, 205
least fixpoint, 207
logic, 167
linear time, 167
LPO, 130
lp, 130

M, see multiplicity function
M, see pomset, multiset
m, see multiplicity function
M, M, M, 216
M, M, M, 216
M, see minimal elements
m, see pomset, multiplicity function of
maximal sequence of direct tests, 149, 152
minimal elements, 188
morphism of lpos, 130
multiplicity function, 132

n, see standard set
N, 130
N, see pomset, multisigelton
N-free pomsets, 216
observer, 149
Ω, 204
overlapping, 186

P, see pomset
P( ), 141
P(P)-refinement, 141
P and, see pomset, P-and-property of
PG, 157
PM, see pomset, property, multiset in-
duced
PN∩, see pomset, PN∩-property of
P, see pomset, property
Pw, 166, 190
Pw, 166
φ, 158
φ(g), 182
φp, 223
p-reflecting, 186
partial order, 130
π, 138
pomset, 131
P_{N-free}-property of, 216
P_{and}-property of, 157
P_{or}-property of, 181
balanced w.r.t. to fission pairs, 246
canonic representative of, 167, 190
canonical association of, 158
empty, 132
finite, 132
multiplicity function of, 132
multiset, 132
multisingleton, 132
parallel composition, 134
prefix of, 138
property, 133
dot synthesizable, 146
hereditary, 144
multiset induced, 133
refinement of, 141
representative of, 131
sequential composition, 134
set, 132
sets of \sim s, 141
singleton, 132
sub\sim, 131
precedence (of combinators), 149
proposition_{\Omega}, 229

RBL, 172
\overline{RBL}, 176
RBL_{\Omega}^{rec}(X), RBL_{\Omega}^{rec}, RBL_{\Omega}, 218
RCL, 173
\overline{RCL}, 176
RCL_{\Omega}^{rec}(X), 228
rec_{x_{-} \sim}, 203
refinement, 135, 141
combinator, 172
fission, 185
m-fission, 185
operator, 141
particular, 136
consistent, 141
syntactic substitution of, 174
restriction of lpo, 131
\sigma, see BL-refinement
\sigma, see P(P)-refinement
\rho, see syntactic substitution
S, see pomset, set

semiword, 166, 190
complement of, 193
\Sigma, see signature
\sigma, 174
\Sigma-domain, 205
\Sigma-precongruence, 211
signature, 204
smoother than, 137
standard set, 185
substitution lemma, 207
substitutive (a relation being \sim), 209
subsumed by, 137
syntactic finite approximations, 205
syntactic preorder, 205
syntactic substitution, 204
tests, 149
direct, 150
trace models, 155
unwinding, 232
variables, 204
free, 204
W, 133
word, 133, 147, 155
X, see variables