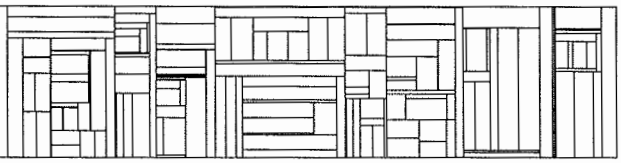
ISSN 0105-8517

On the Compositional Checking of Validity

Glynn Winskel

DAIMI PB – 324
July 1990

COMPUTER SCIENCE DEPARTMENT
AARHUS UNIVERSITY
Ny Munkegade, Building 540
DK-8000 Aarhus C, Denmark



ON THE COMPOSITIONAL CHECKING OF VALIDITY

(Extended Abstract)

Glynn Winskel
Computer Science Department, Aarhus University, Denmark

Introduction

This paper is concerned with deciding whether or not assertions are valid of a parallel process using methods which are directed by the way in which the process has been composed. The assertions are drawn from a modal logic with recursion, capable of expressing a great many properties of interest [EL]. The processes are described by a language inspired by Milner's CCS and Hoare's CSP, though with some modifications. The choice of constructors allows us to handle a range of synchronisation disciplines and ensures that the processes denoted are finite state. The operations are prefixing, a non-deterministic sum, product, restriction, relabelling and a looping construct. Arbitrary parallel compositions are obtained by using a combination of product, restriction and relabelling.

We are interested in deciding whether or not an assertion A is valid of a process t. If it is valid, in the sense that every reachable state of t satisfies A, we write $\models A : t$. Rather than perform the check $\models A : t$ monolithically, on the whole transition system denoted by the term t, we would often rather break the verification down into parts, guided by the composition of t. For instance if t were a sum $t_0 + t_1$ we can ask what assertions A_0 and A_1 should be valid of t_0 and t_1 respectively to ensure that A is valid of $t_0 + t_1$. This amounts to requiring assertions A_0, A_1 such that

$$\models A : t_0 + t_1 \text{ iff } \models A_0 : t_0 \text{ and } \models A_1 : t_1.$$

Once the assertions A_0 and A_1 are found, a validity problem for $t_0 + t_1$ is reduced to a problem to do with t_0 and another with t_1. Further, if the assertions can be found routinely only knowing the top-level operation, that e.g. the process is a sum, we are also told how to construct a process as a sum for which the assertion A is valid: first find components t_0 and t_1 making A_0 and A_1 valid respectively.

This paper investigates the extent to which the composition of t can guide methods for deciding $\models A : t$. It formulates new compositional methods for deciding validity, and exposes some fundamental difficulties. Algorithms are provided to reduce validity problems for prefixing, sum, relabelling, restriction and looping to validity problems for their immediate components—all these reductions depend only on the top-level structure of terms. The existence of these reductions rests on being able to 'embed' the properties of a term in the properties, or products of properties, of its immediate subterms. Because there is not such a simple embedding for the product construction of terms, as might be expected, similar reductions become much more complicated for products; although there are general results, and the reductions can be simple in special cases, the general treatment for products meets with fundamental difficulties. Whereas reductions for products always exist for this finite state language, they demonstrably no longer just depend on the top-level (product structure) of the term; in particular, a simple assertion is exhibited for which the size of the reduction must be quadratic in the number of states of the process. An attempt is thus made to explain what makes product different from the other operations with respect to compositional reasoning, and to delimit the obstacles to automated compositional checking of validity on parallel processes.

1 Transition systems and properties

The syntax, presented formally in the next section, will consist of process terms and assertions.

Process terms will denote labelled transition systems with distinguished initial states. A *labelled transition system* is a structure $(S, i, L, tran)$ where S is a set of *states* containing a distinguished state i, L is a set of *labels*, and $tran \subseteq S \times L \times S$ is a set of *transitions*; as normal, we often write $s \xrightarrow{\alpha} s'$ if $(s, \alpha, s') \in tran$. A state of a labelled transition system is *reachable* iff it can be obtained as the end state of a sequence of transition beginning at the initial state.

A closed assertion is to denote a *property* of a labelled transition system, *i.e.*a subset of its reachable states. We write $P(T)$ for the set of properties

of a labelled transition system T.

We construct labelled transition systems using the constructions of *prefixing, sum, product, restriction, relabelling* and *looping* starting from the *nil* process. These operations form the basis of our syntax for processes. We now describe these constructions. As has been stated, properties of a labelled transition system are identified with subsets of reachable states. The constructions in our language of transition systems are associated with maps. These prove useful in importing properties of immediate components of a term into a property of the term itself. Such mappings between properties are a key to compositional reasoning about processes. We introduce them alongside the constructions with which they are associated.

nil: The *nil* transition system is $(\{i\}, i, \emptyset, \emptyset)$.

Prefixing: For a label α and a labelled transition system $T = (S, i, L, tran)$ the *prefix* αT is obtained by adjoining a new initial state and introducing an α-transition from it to the old initial state. More concretely:

$$\alpha\, T = (S', \emptyset, L \cup \{\alpha\}, tran')$$

where $S' = \{\{s\} \mid s \in S\} \cup \{\emptyset\}$, and

$$\begin{aligned}(s_1, \beta, s_1') \in tran' \text{ iff } & (s_1 = \emptyset \;\&\; \beta = \alpha \;\&\; s_1' = \{i\}) \text{ or} \\ & (s_1 = \{s\} \;\&\; s_1' = \{s'\} \;\&\; (s, \beta, s') \in tran, \\ & \text{for some } s, s'\end{aligned}$$

There is map $S \to S'$ taking $s \in S$ to the corresponding state $\{s\} \in S'$. It extends to a map on properties $P(T) \to P(\alpha T)$. It is convenient to name this map on properties after the prefixing operation and we define

$$\alpha(-) : P(T) \to P(\alpha T)$$

by taking $\alpha U = \{\{s\} \mid s \in U\}$ for $U \in P(T)$.

Sum: Let $T_0 = (S_0, i_0, L_0, tran_0)$ and $T_1 = (S_1, i_1, L_1, tran_1)$ be labelled transition systems. Our nondeterministic sum operation $T_0 + T_1$ is a little different from Milner's. It identifies disjoint copies of the transition systems at their initial states. We define

$$T_0 + T_1 = ((S_0 \times \{i_1\}) \cup (\{i_0\} \times S_1), (i_0, i_1), L_0 \cup L_1, tran')$$

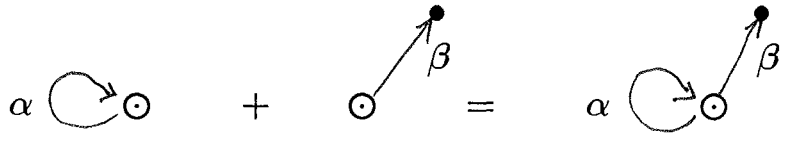
where

$$((s, i_1), \alpha, (s', i_1)) \in tran' \text{ iff } (s, \alpha, s') \in tran_0$$

and

$$((i_0, s), \alpha, (i_0, s')) \in tran' \text{ iff } (s, \alpha, s') \in tran_1.$$

So, the sum construction is obtained by juxtaposing disjoint copies of the transition systems T_0, T_1 but identified at their initial states. The difference with Milner's sum are illustrated by this example:



In the sum it is possible to arbitrarily many α transitions from one component and then do a β transition; this is impossible for Milner's sum where once a transition occurs in one component of a sum then all future transitions must be from the same component. (The introduction of new states demanded by Milner's sum would complicate the reduction.)

A sum $T_0 + T_1$ is associated with two injection functions on states:

$$inj_i : S_i \to (S_0 \times \{i_1\}) \cup (\{i_0\} \times S_1), i = 0, 1$$

with $inj_0(s) = (s, i_1)$ and $inj_1(s) = (i_0, s)$. They induce a map between properties:

$$(- + -) : P(T_0) \times P(T_1) \longrightarrow P(T_0 + T_1)$$

given by $V_0 + V_1 = \{inj_0(s) \mid s \in V_0\} \cup \{inj_1(s) \mid s \in V_1\}$ on $V_0 \in P(T_0), V_1 \in P(T_1)$.

Product: Let $T_0 = (S_0, i_0, L_0, tran_0)$ and $T_1 = (S_1, i_1, L_1, tran_1)$ be labelled transition systems. Their *product* $T_0 \times T_1$ consists of states $S_0 \times S_1$ with initial state (i_0, i_1), labels $L_0 \times_* L_1$ defined to be

$$\{(\alpha_0, *) \mid \alpha \in L_0\} \cup \{(\alpha_0, \alpha_1) \mid \alpha_0 \in L_0, \alpha_1 \in L_1\} \cup \{(*, \alpha_1) \mid \alpha_1 \mid\in L_1\},$$

and transitions $((s_0, s_1), (a_0, a_1), (s_0', s_1'))$ provided these satisfy:

$$a_0 \neq * \Rightarrow (s_0, a_0, s_0') \in tran_0 \text{ and } a_0 = * \Rightarrow s_0 = s_0', \text{ and}$$
$$a_1 \neq * \Rightarrow (s_1, a_1, s_1') \in tran_1 \text{ and } a_1 = * \Rightarrow s_1 = s_1'.$$

Intuitively, the product allows arbitrary synchronisations between pairs of transitions in two components, allowing too for the possibility of a transition in one component proceeding independently of the other.

A product $T_0 \times T_1$ is associated this map on properties:

$$(- \times -) : P(T_0) \times P(T_1) \to P(T_0 \times T_1)$$

where $V_0 \times V_1$ is the cartesian product $\{(s_0, s_1) \mid s_0 \in V_0, s_1 \in V_1\}$.

Restriction: Let $T = (S, i, L, tran)$ be a labelled transition system. Given a subset of labels Λ we can restrict the transitions of T to those with labels in Λ. Define the *restriction* $T \restriction \Lambda = (S, i, L \cap \Lambda, tran')$ where $tran' = \{(s, \alpha, s') \in tran \mid \alpha \in \Lambda\}$.

The reachable states of $T \restriction \Lambda$ are cut-down from those of T. There is an associated map on properties:

$$(- \restriction \Lambda) : P(T) \longrightarrow P(T \restriction \Lambda)$$

where $V \restriction \Lambda = \{s \in V \mid s \text{ reachable in } T \restriction \Lambda\}$.

Relabelling: It is often useful to relabel the transitions of a labelled transition system. Let $T = (S, i, L, tran)$. Let Ξ be a relabelling function from L to labels. Define the relabelled transition system $T\{\Xi\}$ to be $(S, i, \Xi L, tran')$ where where $tran' = \{(s, \Xi(\alpha), s) \mid (s, \alpha, s') \in tran\}$.

Relabelling leaves the states unaffected. Consequently any property of T can be regarded as a property of $T\{\Xi\}$. Define

$$(-\{\Xi\}); P(T) \longrightarrow P(T\{\Xi\})$$

by taking $V\{\Xi\} = V$.

Looping: Let $T = (S, i, L, tran)$ be a labelled transition system. Assume U denotes a property of T. Then by $T/\alpha, U$ we mean the transition system obtained from T by introducing a transition (s, α, i) for each s satisfying the property U. More concretely $T/\alpha, U$ is the $(S, i, L \cup \{\alpha\}, tran')$ where $tran' = tran \cup \{(s, \alpha, i) \mid s \in U\}$.

Like relabelling, the looping construct also leaves the states unaffected. Define

$$(-/\alpha, J) : P(T) \to P(T/\alpha, J)$$

by taking $(V/\alpha, J) = V$.

Parallel compositions: We can represent a variety of different parallel composition through a combined use of product, restriction and relabelling. For example, assuming a distinguished atomic label τ and a bijection $\alpha \mapsto \bar{\alpha}$ between non-τ atomic labels such that $\bar{\bar{\alpha}} = \alpha$, we can represent the parallel composition of CCS: take it to be

$$T_0 \mid T_1 =_{def} (T_0 \times T_1 \upharpoonright \Lambda)\{\Xi\}$$

where the restricting set Λ consists of labels $(\alpha, *), (*, \alpha), (\alpha, \bar{\alpha}), (\tau, *), (*, \tau)$ where α ranges over all labels but for the distinguished label τ, and the relabelling Ξ acts so $\Xi(\alpha, *) = \Xi(*, \alpha) = \alpha$ and $\Xi(\tau, *) = \Xi(*, \tau) = \Xi(\alpha, \bar{\alpha}) = \tau$.

2 Languages

2.1 Syntax

Terms t denote labelled transition systems with distinguished initial states. Assertions A denote their properties. In fact an assertion A will only sensibly denote a property of t when a well-formedness judgement $A : t$ holds. The "raw" syntax of terms and assertions, ignoring for the moment their well-formedness, is mutually dependent and given as follows:

Terms:

$$t ::= nil \mid \alpha t \mid t_0 + t_1 \mid t_0 \times t_1 \mid t \upharpoonright \Lambda \mid t\{\Xi\} \mid (t/\alpha, A)$$

Assertions:

$$\begin{aligned} A ::= & I \mid T \mid F \mid A_0 \wedge A_1 \mid \neg A \mid \\ & \langle a \rangle A \mid \overline{\langle a \rangle} A \mid \\ & X \mid \nu X.A \mid \\ & \alpha A \mid A_0 + A_1 \mid A_0 \times A_1 \mid A \upharpoonright \Lambda \mid A\{\Xi\} \mid (A_0/\alpha, A_1) \mid \\ & (\vdash A : t) \end{aligned}$$

where t is a term, α is a label, Λ is a subset of labels, Ξ is a relabelling function, a is a label (possibly the 'idling' label $*$), and X is an assertion

variable. It is convenient to assume assertion variables belong to unique terms and we write $\mathrm{Var}(t)$ for the countably infinite set of assertion variables associated with the term t; so $\mathrm{Var}(t)$ and $\mathrm{Var}(t')$ are disjoint if t and t' are distinct. Process terms t are associated with a set of labels $Labels(t)$ defined by structural induction:

$$Labels(nil) = \emptyset, \qquad Labels(\alpha t) = Labels(t) \cup \{\alpha\},$$
$$Labels(t_0 + t_1) = Labels(t_0) \cup Labels(t_1),$$
$$Labels(t_0 \times t_1) = Labels(t_0) \times_* Labels(t_1),$$
$$Labels(t \restriction \Lambda) = Labels(t) \cap \Lambda, \qquad Labels(t\{\Xi\}) = \Xi Labels(t),$$
$$Labels(t/\alpha, A) = Labels(t) \cup \{\alpha\}.$$

We use $Labels_*(t)$ to mean $Labels(t) \cup \{*\}$.

The assertion language is essentially a modal ν-calculus with recursion. There are 'forwards' and 'backwards' modalities—the latter are useful in obtaining reductions for the product. The assertion I will be used to refer to initial states; $I : t$ will denote the property holding just at the initial state of the transition system denoted by the process term t. In addition, assertions include constructions on properties with meanings described in the last section; these are used to build properties of a term from properties of its immediate components. It has another unusual construction: a *validity assertion* ($\vdash A : t$) which will, in effect, be true or false according to whether or not A is valid in t, with respect to a particular interpretation of its free variables as properties.

We shall employ some standard abbreviations, and write $A_0 \wedge A_1$ for $\neg(\neg A_0 \vee \neg A_1)$, $A_0 \rightarrow A_1$ for $\neg A_0 \vee A_1$, $A_0 \leftrightarrow A_1$ for $(A_0 \rightarrow A_1) \wedge (A_1 \rightarrow A_0)$. The minimum fixed point $\mu X.A$ stands for $\neg\nu X.\neg A[\neg X/X]$. As regards substitution, we assume the usual renaming of bound variables to avoid the capture of free variables. In some reductions we use a nonstandard *directed* conditional

$$B \rightarrow A_0|A_1$$

to abbreviate $(B \wedge A_0) \vee A_1$. This is unusual; one would expect $(B \wedge A_0) \vee (\neg B \wedge A_1)$. The nonstandard choice is taken to avoid problems with monotonicity in the bodies of recursive definitions. Besides the directed conditional is always used in a context where the right arm A_1 is logically stronger than the left A_0; then the directed conditional $B \rightarrow A_0|A_1$ is logically equivalent to $(B \wedge A_0) \vee (\neg B \wedge A_1)$.

The raw syntax allows assertions which are not sensible. For example, in the construct $\nu X.A$ care must be taken that the body A determines a monotonic operator on sets of states. A sufficient condition for this is that all occurrences of the variable X are positive, i.e. under an even number of negations; otherwise the recursive assertion is not well-formed. The judgement $A : t$ says when an assertion A is well-formed as well as when it expresses a sensible property of a term t, once given properties for its free assertion variables. The well-formedness judgement is given by rules which are reminiscient of typing rules. This is consistent with the view that a process term is regarded as a type of properties. Well-formedness of assertions affects well-formedness of terms because the looping construct on terms $(t/\alpha, A)$ involves an assertion which we insist is closed and such that $A : t$.

Well-formedness rules:

$$I : t \qquad T : t \qquad F : t \qquad \frac{A_0 : t \quad A_1 : t}{A_0 \vee A_1 : t} \qquad \frac{A : t}{\neg A : t}$$

$$\frac{A : t \quad a \in Labels_*(t)}{\langle a \rangle A : t} \qquad \frac{A : t \quad a \in Labels_*(t)}{\overline{\langle a \rangle} A : t}$$

$$X : t \quad \text{if } X \in \mathrm{Var}(t) \qquad \frac{X : t \quad A : t \quad X \text{ +ve in } A}{\nu X.A : t}$$

$$\frac{A : t}{\alpha A : \alpha t} \qquad \frac{A_0 : t_0 \quad A_1 : t_1}{A_0 + A_1 : t_0 + t_1} \qquad \frac{A_0 : t_0 \quad A_1 : t_1}{A_0 \times A_1 : t_0 \times t_1}$$

$$\frac{A : t}{A\restriction\Lambda : t\restriction\Lambda} \qquad \frac{A : t}{A\{\Xi\} : t\{\Xi\}} \qquad \frac{A_0 : t \quad A_1 : t \quad A_1 \text{ is closed}}{(A_0/\alpha, A_1) : t/\alpha, A_1}$$

$$\frac{A : t_0}{(\vdash A : t_0) : t}$$

Validity assertions, of the form $\vdash A : t$ will play a transient, though important, role in the reductions. Although the reductions will often introduce validity assertions, they can be removed so that subsequent reductions work on assertions free of them.

Definition: An assertion which does not contain any validity assertions will be called *pure*.

2.2 Semantics

From the previous section, we understand each of the constructions in our language of process terms and so the denotation of a process term by a labelled transition system; in the case of the looping we will need to rely on the semantics of closed assertions as properties, made precise shortly.

Notation: In our subsequent work we will adopt the convention that a term t denotes a labelled transition system

$$(S_t, i_t, L_t, tran_t),$$

and write, for instance, $s \xrightarrow{\alpha}_t s'$ to signify a transition in the transition system denoted by t. We shall write $P(t)$ for the set properties of (the transition system denoted by) t.

We give semantics to assertions A accompanied by a judgement $A : t$. To cope with the possibility of free assertion variables in A, we use environments. Together assertion variables form the set

$$\mathrm{Var} = \bigcup \{\mathrm{Var}(t) \mid t \in \mathrm{Term}\}.$$

An *environment* ρ is a function

$$\rho : \mathrm{Var} \to \bigcup \{P(t) \mid t \in \mathrm{Term}\}$$

such that $\rho(X) \in P(t)$ for $X \in \mathrm{Var}(t)$. Define Env to be the set of environments. The denotation of $A : t$ will be $[\![A : t]\!]$ of type $\mathrm{Env} \to P(t)$. Define:

$$
\begin{array}{lcl}
[\![I : t]\!] & = & \lambda\rho.\{i_t\} \\
[\![T : t]\!] & = & \lambda\rho.S_t \\
[\![F : t]\!] & = & \lambda\rho.\emptyset \\
[\![A_0 \vee A_1 : t]\!] & = & \lambda\rho.[\![A_0 : t]\!]\rho \cup [\![A_1 : t]\!]\rho \\
[\![\neg A : t]\!] & = & \lambda\rho.(S_t \setminus [\![A : t]\!]\rho) \\
[\![\langle\alpha\rangle A : t]\!] & = & \lambda\rho.\{s \in S_t \mid \exists s'.\ s \xrightarrow{\alpha}_t s' \ \&\ s' \in [\![A : t]\!]\rho\} \\
[\![X : t]\!] & = & \lambda\rho.\rho(X) \\
[\![\nu X.A : t]\!] & = & \lambda\rho.(\nu U.[\![A : t]\!]\rho[U/X]) \\
 & & \text{the greatest fixed point of the function} \\
 & & U \mapsto [\![A : t]\!]\rho[U/X] \\
[\![\alpha A : \alpha t]\!] & = & \lambda\rho.\alpha([\![A : t]\!]\rho) \\
[\![A_0 + A_1 : t_0 + t_1]\!] & = & \lambda\rho.[\![A_0 : t_0]\!]\rho + [\![A_1 : t_1]\!]\rho \\
[\![A_0 \times A_1 : t_0 \times t_1]\!] & = & \lambda\rho.[\![A_0 : t_0]\!]\rho \times [\![A_1 : t_1]\!]\rho \\
[\![A \restriction \Lambda : t \restriction \Lambda]\!] & = & \lambda\rho.[\![A : t]\!]\rho\lceil\Lambda \\
[\![A\{\Xi\} : t\{\Xi\}]\!] & = & \lambda\rho.([\![A : t]\!]\rho)\{\Xi\} \\
[\![A_0/\alpha, A_1 : (t/\alpha, A_1)]\!] & = & \lambda\rho.([\![A_0 : t]\!]\rho/\alpha, [\![A_1]\!]\rho) \\
[\![(\vdash A : t_0) : t]\!] & = & \lambda\rho.([\![A : t_0]\!]\rho = S_{t_0} \to S_t \mid \emptyset)
\end{array}
$$

Definition: *(Validity)* Let $A : t$ be an assertion. Define

$$\models A : t \text{ iff } [\![A : t]\!]\rho = S_t \text{ for all environments } \rho.$$

3 How to do reductions

We first motivate the technique by considering the reduction for the looping construct. Let t be a term, $J : t$ a closed assertion. Then $(t/\alpha, J)$ denotes a transition system like that of t but with extra α transitions from all the states satisfying J to the initial state. Suppose A is a closed assertion of the pure ν-calculus (with no mention of validity assertions) so that $A : (t/\alpha, J)$. We describe how to produce an assertion $B : t$ such that

$$\models A : (t/\alpha, J) \text{ iff } \models B : t$$

and in this way reduce the validity problem for a term $(t/\alpha, J)$ to one for t. The assertion B will be defined by structural induction on A.

In the course of the structural induction we will generally encounter assertions which have free variables. To cope with this the reduction is

done with respect to a substitution σ transforming variables $X : (t/\alpha, J)$ to assertions $(Y/\alpha, J) : (t/\alpha, J)$. In order not to introduce spurious dependencies it will be assumed that the free variables Y do not appear free in any assertion being reduced and that σ yields distinct Y for distinct X. With respect to such a change of variables σ, we will consider a few clauses of the reduction, and indicate how it can be proved that if A is pure with $A : t/\alpha, J$ then $red(A : t/\alpha, J; \sigma) = B$ with $B : t$ and

$$\models A[\sigma] \leftrightarrow B/\alpha, J : (t/\alpha, J).$$

This means that for all environments assigning properties to the free assertion variables, the assertions A and $B/\alpha, J$ denote the same property of $t/\alpha, J$. It follows that, whenever $A : t/\alpha, J$ is closed,

$$(\models A : t/\alpha, J) \text{ iff } (\models B : t).$$

In this sense, a validity problem for a term $t/\alpha, J$ is reduced to one for t. We present a few clauses of the reduction:

$$\begin{array}{lcl} red(\langle\alpha\rangle A : t/\alpha, J; \sigma) & = & (\vdash I \to B : t) \to ((\langle\alpha\rangle B) \vee J) \mid \langle\alpha\rangle B \\ & & \text{where } red(A : t/\alpha, J; \sigma) = B \\ red(X : t/\alpha, J; \sigma) & = & Y \text{ when } \sigma(X) = Y/\alpha, J \\ red(\nu X.A : t/\alpha, J; \sigma) & = & \nu Y.red(A : t/\alpha, J; \sigma) \text{ where } \sigma(X) = Y/\alpha, J. \end{array}$$

The second and third clauses express little more than a renaming of free variables. To understand the first reduction, assume inductively that

$$\models A[\sigma] \leftrightarrow B/\alpha, J : t/\alpha, J$$

and argue, for a state s of $t/\alpha, J$ and arbitrary environment ρ, that

$$\begin{array}{l} s \in [\![\langle\alpha\rangle A[\sigma] : (t/\alpha, J)]\!]\rho \\ \iff \exists s'.s \xrightarrow{\alpha}_{t/\alpha,J} s' \;\&\; s' \in [\![A[\sigma] : (t/\alpha, J)]\!]\rho \\ \iff \exists s'.s \xrightarrow{\alpha}_{t/\alpha,J} s' \;\&\; s' \in [\![B : t]\!]\rho \quad \text{by induction} \\ \iff \begin{cases} (\exists s'.s \xrightarrow{\alpha}_t s' \;\&\; s' \in [\![B : t]\!]\rho) \text{ or } s \in [\![J : t]\!]\rho & \text{if } i_t \in [\![B : t]\!]\rho, \\ (\exists s'.s \xrightarrow{\alpha}_t s' \;\&\; s' \in [\![B : t]\!]\rho) & \text{if } i_t \notin [\![B : t]\!]\rho, \end{cases} \\ \quad \text{directly from the looping construction,} \\ \iff \begin{cases} s \in [\![(\langle\alpha\rangle B) \vee J : t]\!]\rho & \text{if } [\![\vdash I \to B : t]\!]\rho = S_t \\ s \in [\![\langle\alpha\rangle B : t]\!]\rho & \text{if } [\![\vdash I \to B : t]\!]\rho = \emptyset \end{cases} \\ \iff s \in [\![(\vdash I \to B : t) \to ((\langle\alpha\rangle B) \vee J) \mid \langle\alpha\rangle B : t]\!]\rho \end{array}$$

There remains however one hitch. The reduction, like that for the other term constructors, works on pure assertions—those which do not contain validity assertions. As is clear from some of the clauses above, reductions can sometimes yield assertions with validity assertions. If we are now to continue the reduction (using the structure of t) we must show how to prepare such validity assertions so they can be handled by these further reductions.

Look at one clause where validity assertions are introduced:

$$red(\langle\alpha\rangle A : t/\alpha, J; \sigma) = (\vdash I \rightarrow B : t) \rightarrow ((\langle\alpha\rangle B) \vee J) \mid \langle\alpha\rangle B$$

where $red(A : t/\alpha, J; \sigma) = B$. If B is closed there are no difficulties: we check for the smaller term t whether or not $I \rightarrow B$ is valid and if it is return the left, and otherwise the right branch of the conditional as the appropriate reduction. Validity assertions $\vdash B : t$ cause no difficulties when B is closed. But in general B will contain free assertion variables. However, ultimately we are concerned with reducing a closed assertion, which will mean that all free variables in validity assertions are bound by an enclosing recursive definition. The following fact means that, for a closed assertion denoting a property of t, the internal validity assertions introduced by its reduction can be made closed, and so benign because they refer to proper subterms of t:

Lemma 1 *(The closure lemma)*
Let $\mathcal{C}[\ ]$ be a context such that $\mathcal{C}[Y] : t$ and Y occurs positively in $\mathcal{C}[Y] : t$, for any variable $Y : t_0$. Suppose $B : t_0$ and $[\![B : t_0]\!]\rho = \emptyset$ or $[\![B : t_0]\!]\rho = S_{t_0}$, for any environment ρ. Let X be a variable such that $\nu X.\ \mathcal{C}[B] : t$. Then

$$\models \nu X.\ \mathcal{C}[B] \leftrightarrow \nu X.\ \mathcal{C}[B[\nu X.\ \mathcal{C}[T]/X]] : t.$$

As an illustration, consider the reduction of $\nu X.\ \langle\alpha\rangle X : t/\alpha, J$. This should yield an assertion of t true at those reachable states of t which become able to do arbitrarily many α-transitions once the loops of the construction $t/\alpha, J$ are introduced. Assume the change of variables takes X to $Y/\alpha, J$. Then, following the reductions above, we get

$$red(\nu X.\ \langle\alpha\rangle X : t/\alpha, J; \sigma) = \nu Y.\ ((\vdash I \rightarrow Y : t) \rightarrow ((\langle\alpha\rangle Y) \vee J) | \langle\alpha\rangle Y).$$

By the closure lemma we can close the validity assertion, to obtain the equivalent

$$\nu Y.(\vdash I \rightarrow \nu Y.\ (((\langle\alpha\rangle Y) \vee J)) : t) \rightarrow ((\langle\alpha\rangle Y) \vee J | \langle\alpha\rangle Y.$$

Thus $red(\nu X.\ \langle\alpha\rangle X : t/\alpha, J; \sigma)$ is equivalent to

1. $\nu Y.\ \langle\alpha\rangle Y$ if $\nvdash I \rightarrow \nu Y.(((\langle\alpha\rangle Y) \vee J)$, and to
2. $\nu Y.\ (((\langle\alpha\rangle Y) \vee J)$ if $\vdash I \rightarrow \nu Y.\ (((\langle\alpha\rangle Y) \vee J)$.

In other words, a state in $t/\alpha, J$ can do arbitrarily many α transitions

1. if the corresponding state in t can, or
2. it can reach a state in J through α-transitions and the initial state of t can either do unboundedly many α-transitions or itself reach a state in J via α-transitions.

This is the kind of result one could write down informally, except one might forget a case in 2. The informal argument is helped enormously through there being a simple reading of the recursive assertion. The reductions work for all manner of recursive assertions. I hope this indicates how the reductions perform rather complicated inference steps.

To illustrate more fully the issues involved in performing reductions we will consider the case of sums. Provided $A_0 : t_0$ and $A_1 : t_1$ then $A_0 + A_1 : t_0 + t_1$. This sum constructor on assertions reflects the operation we have seen for obtaining a property of a sum from properties of its components. For a closed assertion $A : t_0 + t_1$ we are interested in how to produce assertions $A_0 : t_0, A_1 : t_1$ so that

$$\models A \leftrightarrow A_0 + A_1 : t_0 + t_1.$$

Provided we can ensure in addition that the pair $A_0 : t_0$, $A_1 : t_1$ is *balanced* in the sense that

$$i_{t_0} \in [\![A_0 : t_0]\!]\rho \quad \text{iff} \quad i_{t_1} \in [\![A_1 : t_1]\!]\rho,$$

for all environments ρ, then

$$\models A : t_0 + t_1 \quad \text{iff} \quad (\models A_0 : t_0 \text{ and } \models A_1 : t_1).$$

(Without the additional requirement of the assertions being balanced the "only if" direction of the statement can fail because only one of A_0 and A_1 can be true at the initial state.) The method for producing A_0, A_1 will work by induction on the structure of A, in the course of which we cannot hope to always deal with closed assertions. In particular, how are we to reduce a variable $X : t_0 + t_1$? The answer rests on the fact that the reduction will take place relative to a change of variables, in which variables like X are replaced by $Y_0 + Y_1$ for distinct variables $Y_0 : t_0$, $Y_1 : t_1$.

To illustrate the mechanism of the reduction it is shown how, for a closed assertion $\nu X.A : t_0 + t_1$, a balanced pair of assertions $B_0 : t_0$ and $B_1 : t_1$ can be found such that

$$\models \nu\, X.A \leftrightarrow B_0 + B_1 : t_0 + t_1.$$

Of key importance are the maps between properties $P(t_0+t_1)$ and $P(t_0)\times P(t_1)$. The change of variables is associated with the map

$$in : P(t_0) \times P(t_1) \to P(t_0 + t_1) \text{ where } in(V_0, V_1) = U_0 + U_1.$$

On the other hand, the reduction is associated with a map *out* in the converse direction

$$out : P(t_0 + t_1) \to P(t_0) \times P(t_1) \text{ where } out(U) = (out_0(U), out_1(U))$$

which projects a property U of $t_0 + t_1$ to a pair of properties

$$out_0(U) = \{s \in S_{t_0} \mid inj_0(s) \in U\},$$
$$out_1(U) = \{s \in S_{t_1} \mid inj_1(s) \in U\}.$$

It is easy to see that the maps are monotonic with respect to inclusion and that *out* is an *embedding* of the properties $P(t_0 + t_1)$ in $P(t_0) \times P(t_1)$ in the sense that $in \circ out = 1_{P(t_0+t_1)}$. These facts are important because they fit into a general pattern for transforming fixed points:

Lemma 2 *(The embedding lemma)*
Suppose D and E are complete lattices for which $in : D \to E$ and $out : E \to D$ are monotonic, with $in \circ out = 1_E$. Suppose $\phi : E \to E$ is monotonic. Defining

$$\psi = out \circ \phi \circ in$$

we obtain a monotonic function $\psi : D \to D$ for which $\nu\phi = in(\nu\psi)$.

An assertion $A : t_0 + t_1$ with a single free variable $X : t_0 + t_1$ determines a function $\phi : P(t_0 + t_1) \to P(t_0 + t_1)$ from a property U of $t_0 + t_1$ to a property $[\![A : t_0 + t_1]\!]\rho[U/X]$ of $t_0 + t_1$. Suppose we have already obtained a reduction of A to a balanced pair of assertions A_0, A_1 with respect to a change of variables taking X to $Y_0 + Y_1$, i.e.

$$\models A[Y_0 + Y_1/X] \leftrightarrow A_0 + A_1 : t_0 + t_1.$$

Then, equivalently, we can see the reduction as giving a syntactic expression of the embedding:

$$[\![A_0 : t_0]\!]\rho = out_0([\![A[Y_0 + Y_1/X] : t_0 + t_1]\!]\rho)$$
$$[\![A_1 : t_1]\!]\rho = out_1([\![A[Y_0 + Y_1/X] : t_0 + t_1]\!]\rho).$$

Writing $\psi_0(V_0, V_1) = [\![A_0 : t_0]\!]\rho[V_0/Y_0, V_1/Y_1]$ and $\psi_1(V_0, V_1) = [\![A_1 : t_1]\!]\rho[V_0/Y_0, V_1/Y_1]$, yields

$$\psi_0(V_0, V_1) = out_0(\phi(V_0 + V_1)) \quad \text{and} \quad \psi_1(V_0, V_1) = out_1(\phi(V_0 + V_1))$$

for all $V_0 \in P(t_0), V_1 \in P(t_1)$. Now, defining ψ by

$$\psi(V_0, V_1) = (\psi_0(V_0, V_1), \psi_1(V_0, V_1)),$$

for $V_0 \in P(t_0), V_1 \in P(t_1)$, we can see that $\psi = out \circ \phi \circ in$. The well-formedness of assertions will ensure monotonicity of ϕ and ψ so we can apply the embedding lemma to obtain:

$$\nu U.\phi(U) = in(\nu(V_0, V_1).(\psi_0(V_0, V_1), \psi_1(V_0, V_1))).$$

By Bekic's theorem

$$\nu U.\phi(U) = in(\nu V_0.\psi_0(V_0, \nu V_1.\psi_1(V_0, V_1)),\ \nu V_1.\psi_1(\nu V_0.\psi_0(V_0, V_1), V_1)).$$

With an eye back to syntax, this means:

$$\models \nu X.A \leftrightarrow (\nu Y_0.A_0[\nu Y_1.A_1/Y_1] + \nu Y_0.A_1[\nu Y_0.A_0/Y_0]) : t_0 + t_1.$$

We have thus succeeded in producing a pair of assertions $B_0 : t_0$, $B_1 : t_1$ such that

$$\models \nu X.A \leftrightarrow (B_0 + B_1) : t_0 + t_1.$$

A small additional argument, based on the assumption that $A_0 : t_0, A_1 : t_1$ are balanced, shows that $B_0 : t_0$, $B_1 : t_1$ form a balanced pair. Because

$B_0 : t_0$, $B_1 : t_1$ denote fixed points of ψ we see, for an arbitrary environment ρ, that

$$[\![B_0 : t_0]\!]\rho = [\![A_0 : t_0]\!]\rho' \quad \text{and} \quad [\![B_1 : t_1]\!]\rho = [\![A_1 : t_1]\!]\rho'$$

where $\rho' = \rho[[\![B_0 : t_0]\!]\rho/Y_0, [\![B_1 : t_1]\!]\rho/Y_1]$. Now we observe that because the pair $A_0 : t_0, A_1 : t_1$ is balanced, so is $B_0 : t_0$, $B_1 : t_1$.

The reductions for the other constructions follow similar lines. Reductions will express embeddings of properties of a term in the properties, or products of properties, of its immediate components. They will be defined with respect to a change of variables associated with the left inverse to the embedding.

4 Summary of results

We now describe how to perform reductions for all the operations. As with the reduction for the looping construct, we shall need to change variables, so as to transform properties of a term to corresponding properties of its immediate subcomponents. We shall call such transformations *changes of variables*. All such transformations will be achieved through substitutions which introduce only fresh variables over properties.

Definition: A substitution σ is said to be *fresh* for an assertion A if it has the properties:

(i) for all variables X at which σ is defined the free variables in $\sigma(X)$ are disjoint from those in A, and

(ii) for distinct variables X and X', at which σ is defined, the free variables in $\sigma(X)$ and $\sigma(X')$ are disjoint.

Many of the reductions will introduce validity assertions. These are harmless however. They will always be validity assertions with respect to a smaller term than that of immediate interest, and, through the use of the closure lemma, lemma 1, they can be made closed whenever they arise as the reduction of a closed assertion; as such they can be checked, replaced by T or F as appropriate, and hence eliminated.

4.1 The reduction for *nil*

Given a closed, pure assertion $A : nil$, it is a simple matter to see whether or not it is valid at *nil*. The following function yields true in case it is valid, and false otherwise:

$$\begin{array}{lcl}
red(I : nil) & = & \text{true} \\
red(T : nil) & = & \text{true} \\
red(F : nil) & = & \text{false} \\
red(A_0 \vee A_1 : nil) & = & red(A_0 : nil) \text{ or } (red(A_1 : nil) \\
red(\neg A : nil) & = & \text{not } red(A : nil) \\
red(\langle * \rangle A : nil) & = & red(A : nil) \\
red(\overline{\langle * \rangle} A : nil) & = & red(A : nil) \\
red(\nu X.A : nil) & = & red(A[T/X] : nil).
\end{array}$$

Theorem 3 *For $A : nil$ a closed, pure assertion, $red(A : nil)$ iff $\models A : nil$.*

4.2 Reduction for prefixing

The reduction for prefixing is based on an embedding of $P(\alpha t)$ into $P(t) \times P(t)$, for a term t, meeting the requirements of the embedding lemma 2. Define $down : P(\alpha t) \to P(t)$ by taking

$$down(U) = \{s \in S_t \mid \{s\} \in U\}.$$

Define $contI : P(\alpha t) \to P(t)$ by taking

$$contI(U) = \{i_t \mid i_{\alpha t} \in U\}.$$

Now we take the embedding to be

$$out = (down, contI) : P(\alpha t) \to P(t)^2$$

so that $out(U) = (down(U), contI(U))$. The converse map arises by taking

$$in : P(t)^2 \to P(\alpha t)$$

where

$$in(V_0, V_1) = \alpha(V_0) \cup \{i_{\alpha t} \mid i_t \in V_1\}.$$

It is easy to check that both *in* and *out* are monotonic and $in \circ out = 1_{P(\alpha t)}$.

The map *in* corresponds to a change of variables, with respect to which we'll define a reduction whose two components correspond to the two components of the embedding *out*.

Definition: Assume $A : \alpha t$. A *change of variables* of $A : \alpha t$ is a substitution σ, with domain $Var(\alpha t)$, which is fresh for A, and such that for all variables $X : \alpha t$ there are distinct variables $Y_0 : t$ and $Y_1 : t$ with

$$\sigma(X) = \alpha(Y_0) \vee ((\vdash I \to Y_1 : t) \wedge I).$$

This change of variables expresses the map *in*:

Proposition 4 *Let* $X : \alpha t$, $Y_0 : t$ *and* $Y_1 : t$ *be distinct variables. Suppose*

$$\sigma(X) = \alpha(Y_0) \vee ((\vdash I \to Y_1 : t) \wedge I).$$

Let ρ *be an environment. Then, for* $V_0, V_1 \in P(t)$

$$in(V_0, V_1) = [\![\sigma(X) : \alpha t]\!]\rho[V_0/Y_0, V_1/Y_1].$$

Given a pure assertion $A : \alpha t$ and σ a change of variables for it we define two functions
$red^0(A : \alpha t; \sigma)$ and $red^1(A : \alpha t; \sigma)$, such that

$$[\![red^0(A : \alpha t; \sigma) : t]\!]\rho = down([\![A[\sigma] : \alpha t]\!]\rho)$$
$$[\![red^1(A : \alpha t; \sigma) : t]\!]\rho \cap \{i_t\} = contI([\![A[\sigma] : \alpha t]\!]\rho)$$

for any environment ρ.

By structural induction on $A : \alpha t$, with respect to a change of variables for it, define

$$
\begin{array}{lcl}
red^0(I : \alpha t; \sigma) & = & F \\
red^0(T : \alpha t; \sigma) & = & T \\
red^0(F : \alpha t; \sigma) & = & F \\
red^0(A_0 \vee A_1 : \alpha t; \sigma) & = & red^0(A_0 : \alpha t; \sigma) \vee red^0(A_1 : \alpha t; \sigma) \\
red^0(\neg A : \alpha t; \sigma) & = & \neg red^0(A : \alpha t; \sigma) \\
red^0(\langle a \rangle A : \alpha t; \sigma) & = & \langle a \rangle red^0(A : \alpha t, \sigma) \\
red^0(\overline{\langle a \rangle} A : \alpha t; \sigma) & = & \overline{\langle a \rangle} red^0(A : \alpha t; \sigma) \text{ if } a \neq \alpha \\
red^0(\overline{\langle \alpha \rangle} A : \alpha t; \sigma) & = & \overline{\langle \alpha \rangle} red^0(A : \alpha t; \sigma) \vee (red^1(A : \alpha t; \sigma) \wedge I) \\
red^0(X : \alpha t; \sigma) & = & Y_0 \text{ where} \\
 & & \sigma(X) = \alpha(Y_0) \vee ((\vdash I \rightarrow Y_1 : t) \wedge I.) \\
red^0(\nu X.A : \alpha t; \sigma) & = & \nu Y_0.\ red^0(A : \alpha t; \sigma)[\nu Y_1.red^1(A : \alpha t; \sigma)/Y_1]) \text{ where} \\
 & & \sigma(X) = \alpha(Y_0) \vee ((\vdash I \rightarrow Y_1 : t) \wedge I) \\
red^0(\alpha A : \alpha t; \sigma) & = & A
\end{array}
$$

$$
\begin{array}{lcl}
red^1(I : \alpha t; \sigma) & = & T \\
red^1(T : \alpha t; \sigma) & = & T \\
red^1(F : \alpha t; \sigma) & = & F \\
red^1(A_0 \vee A_1 : \alpha t; \sigma) & = & red^1(A_0 : \alpha t; \sigma) \vee red^1(A_1 : \alpha t; \sigma) \\
red^1(\neg A : \alpha t; \sigma) & = & \neg red^1(A : \alpha t; \sigma) \\
red^1(\langle * \rangle A : \alpha t; \sigma) & = & red^1(A : \alpha t; \sigma) \\
red^1(\langle a \rangle A : \alpha t; \sigma) & = & F \text{ if } a \neq * \ \& \ a \neq \alpha \\
red^1(\langle \alpha \rangle A : \alpha t; \sigma) & = & red^0(A : \alpha t; \sigma) \\
red^1(\overline{\langle * \rangle} A : \alpha t; \sigma) & = & red^1(A : \alpha t, \sigma) \\
red^1(\overline{\langle a \rangle} A : \alpha t; \sigma) & = & F \text{ if } a \neq * \\
red^1(X : \alpha t; \sigma) & = & Y_1 \text{ where } \sigma(X) = \alpha(Y_0) \vee ((\vdash I \rightarrow Y_1 : t) \wedge I) \\
red^1(\nu X.A : \alpha t; \sigma) & = & \nu Y_1.(red^1(A : \alpha t; \sigma)[\nu Y_0.red^0(A : \alpha t; \sigma)/Y_0]) \\
red^1(\alpha A : \alpha t; \sigma) & = & F
\end{array}
$$

Theorem 5 *Assume $A : \alpha t$ with A pure. Suppose σ is a change of variables of $A : \alpha t$. Let $red^0(A : \alpha t; \sigma) = A_0$ and $red^1(A : \alpha t; \sigma) = A_1$. Then $A_0 : t$ and $A_1 : t$. Moreover*

$$
\begin{array}{c}
[\![A_0 : t]\!]\rho = down([\![A[\sigma] : \alpha t]\!]\rho) \text{ and} \\
i_t \in [\![A_1 : t]\!]\rho \quad \textit{iff} \quad i_{\alpha t} \in [\![A[\sigma] : \alpha t]\!]\rho
\end{array}
$$

for any environment ρ.
If $A : \alpha t$ is closed then $(\models A : \alpha t)$ iff $(\models A_0 \wedge (I \rightarrow A_1) : t)$.

4.3 Reduction for sum

The reduction will be based on an embedding of $P(t_0+t_1)$ in $P(t_0)\times P(t_1)$, for terms t_0, t_1. Define

$$out_0 : P(t_0 + t_1) \to P(t_0),$$
$$out_1 : P(t_0 + t_1) \to P(t_1)$$

by taking $out_0(U) = \{s \in S_{t_0} \mid inj_0(s) \in U\}$ and $out_1(U) = \{s \in S_{t_1} \mid inj_1(s) \in U\}$ for all $U \in P(t_0 + t_1)$. Define the embedding

$$out : P(t_0 + t_1) \to P(t_0) \times P(t_1)$$

by taking $out(U) = (out_0(U), out_1(U))$ for all $U \in P(t_0 + t_1)$. Define its left-inverse

$$in : P(t_0) \times P(t_1) \to P(t_0 + t_1)$$

by taking $in(V_0, V_1) = V_0 + V_1$. Both *in* and *out* are monotonic, and it is easily seen that $in \circ out = 1_{P(t_0+t_1)}$.

The map *in* accompanies a change of variables:

Definition: Let $A : t_0 + t_1$. A *change of variables of* $A : t_0 + t_1$ is a substitution σ with domain $Var(t_0 + t_1)$, which is fresh for A, and such that for any variable $X : t_0 + t_1$ we have $\sigma(X) = Y_0 + Y_1$ for distinct variables $Y_0 : t_0$ and $Y_1 : t_1$.

Proposition 6 *Let* $X : t_0 + t_1$, $Y_0 : t_0$ *and* $Y_1 : t_1$ *be distinct variables. Suppose* $\sigma(X) = Y_0 + Y_1$. *Let* ρ *be an environment. Then, for* $V_0 \in P(t_0)$, $V_1 \in P(t_1)$

$$in(V_0, V_1) = [\![\sigma(X) : \alpha t]\!]\rho[V_0/Y_0, V_1/Y_1].$$

With respect to a change of variables σ, we can transform an assertion $A : t_0 + t_1$ to the sum of a pair of assertions $A_0 : t_0$ and $A_1 : t_1$ which realise the components of the embedding *out*, i.e.

$$[\![A_0 : t_0]\!]\rho = out_0([\![A[\sigma] : t_0{+}t_1]\!]\rho) \quad \text{and} \quad [\![A_1 : t_1]\!]\rho = out_1([\![A[\sigma] : t_0{+}t_1]\!]\rho)$$

for any environment ρ.

The reduction is carried out by the pair of functions $red^0(A : t_0 + t_1; \sigma)$, $red^1(A : t_0 + t_1; \sigma)$, acting on an assertion A for which $A : t_0 + t_1$ and a change of variables for it. They are defined by the following structural induction (we omit the clauses for red^1 as they reflect those for red^0):

$$
\begin{array}{lcl}
red^0(I : t_0 + t_1; \sigma) & = & I \\
red^0(T : t_0 + t_1; \sigma) & = & T \\
red^0(F : t_0 + t_1; \sigma) & = & F \\
red^0(A \vee B : t_0 + t_1; \sigma) & = & A_0 \vee B_0 \\
 & & \text{where } A_0 = red^0(A : t_0 + t_1; \sigma) \text{ and} \\
 & & \qquad B_0 = red^0(B : t_0 + t_1; \sigma) \\
red^0(\neg A : t_0 + t_1; \sigma) & = & \neg red^0(A : t_0 + t_1; \sigma) \\
red^0(\langle * \rangle A : t_0 + t_1; \sigma) & = & red^0(A : t_0 + t_1; \sigma) \\
red^0(\langle \alpha \rangle A : t_0 + t_1; \sigma) & = & (\vdash I \to \langle \alpha \rangle A_1 : t_1) \to (\langle \alpha \rangle A_0) \vee I \mid \langle \alpha \rangle A_0 \text{ where} \\
 & & red^0(A : t_0 + t_1; \sigma) = A_0 \text{ and} \\
 & & red^1(A : t_0 + t_1; \sigma) = A_1, \\
 & & \text{if } \alpha \neq * \\
red^0(\overline{\langle * \rangle} A : t_0 + t_1; \sigma) & = & red^0(A : t_0 + t_1; \sigma) \\
red^0(\overline{\langle \alpha \rangle} A : t_0 + t_1; \sigma) & = & ((\vdash I \to \overline{\langle \alpha \rangle} A_1 : t_1) \to (\overline{\langle \alpha \rangle} A_0) \vee I \mid \overline{\langle \alpha \rangle} A_0 \text{ where} \\
 & & red^0(A : t_0 + t_1; \sigma) = A_0 \text{ and} \\
 & & red^1(A : t_0 + t_1; \sigma) = A_1, \\
 & & \text{if } \alpha \neq * \\
red^0(X : t_0 + t_1; \sigma) & = & (\vdash I \to Y_1 : t_1) \to Y_0 \vee I | Y_0 \quad \text{where } \sigma(X) = Y_0 + Y_1 \\
red^0(\nu X.A : t_0 + t_1; \sigma) & = & \nu Y_0.A_0[\nu Y_1.A_1/Y_1] \text{ where} \\
 & & A_0 = red^0(A : t_0 + t_1; \sigma), \\
 & & A_1 = red^1(A : t_0 + t_1; \sigma) \\
 & & \text{and } \sigma(X) = Y_0 + Y_1 \\
red^0(A_0 + A_1 : t_0 + t_1; \sigma) & = & (\vdash I \to A_1 : t_1) \to (A_0 \vee I) | A_0
\end{array}
$$

Theorem 7 *Let $A : t_0 + t_1$ be pure. Let $A_0 = red^0(A : t_0 + t_1; \sigma)$ and $A_1 = red^1(A : t_0 + t_1; \sigma)$, for σ a change of variables for A. Then*

$$[\![A_0 : t_0]\!]\rho = out_0([\![A[\sigma] : t_0{+}t_1]\!]\rho) \quad \textit{and} \quad [\![A_1 : t_1]\!]\rho = out_1([\![A[\sigma] : t_0{+}t_1]\!]\rho)$$

for any environment ρ.
If $A : t_0{+}t_1$ is closed then $(\models A : t_0{+}t_1)$ iff $[(\models A_0 : t_0)$ and $(\models A_1 : t_1)]$.

4.4 Reduction for looping.

The properties $P(t/\alpha, J)$ and $P(t)$ are the same, and this time the embedding and its inverse with respect to which the reduction is performed are both the identity map. The inverse is realised by a change of variables, which essentially just renames them:

Definition: Assume $A : t/\alpha, J$ with A pure. A *change of variables* of $A : t/\alpha, J$ is a substitution σ, with domain $Var(t/\alpha, J)$, which is fresh for A, and of the form $\sigma(X) = Y/\alpha, J$ for variables $X : t/\alpha, J$ and $Y : t$.

The effort goes into finding an assertion to an assertion $red(A : t/\alpha, J; \sigma)$, such that for a pure $A : (t/\alpha, J)$ and a change of variables σ for it,

$$\models A[\sigma] \leftrightarrow red(A : t/\alpha, J; \sigma) : t/\alpha, J.$$

The reduction of a pure assertion $A : t/\alpha, J$, with respect to a change of variables for it, is defined by structural induction:

$$\begin{array}{lcl}
red(I : t/\alpha, J; \sigma) & = & I \\
red(T : t/\alpha, J; \sigma) & = & T \\
red(F : t/\alpha, J; \sigma) & = & F \\
red(A_0 \vee A_1 : t/\alpha, J; \sigma) & = & red(A_0 : t/\alpha, J; \sigma) \vee red(A_1 : t/\alpha, J; \sigma) \\
red(\neg A : t/\alpha, J; \sigma) & = & \neg red(A : t/\alpha, J; \sigma) \\
red(\langle b\rangle A : t/\alpha, J; \sigma) & = & \langle b\rangle red(A : t/\alpha, J; \sigma) \text{ if } b \neq \alpha \\
red(\langle \alpha\rangle A : t/\alpha, J; \sigma) & = & (\vdash I \rightarrow B : t) \rightarrow ((\langle\alpha\rangle B) \vee J) \mid \langle\alpha\rangle B \\
 & & \text{where } red(A : t/\alpha, J; \sigma) = B \\
red(\overline{\langle b\rangle} A : t/\alpha, J; \sigma) & = & \overline{\langle b\rangle} red(A : t/\alpha, J; \sigma) \text{ if } b \neq \alpha \\
red(\overline{\langle \alpha\rangle} A : t/\alpha, J; \sigma) & = & \neg(\vdash J \rightarrow \neg B : t) \rightarrow ((\overline{\langle\alpha\rangle} B) \vee I) \mid \overline{\langle\alpha\rangle} B \\
 & & \text{where } red(A : t/\alpha, J; \sigma) = B \\
red(X : t/\alpha, J; \sigma) & = & Y \text{ when } \sigma(X) = Y/\alpha, J \\
red(\nu X.A : t/\alpha, J; \sigma) & = & \nu Y.red(A : t/\alpha, J; \sigma) \text{ where } \sigma(X) = Y/\alpha, J \\
red(A/\alpha, J : t/\alpha, J; \sigma) & = & A.
\end{array}$$

Theorem 8 *Let A be pure with $A : t/\alpha, J$. Let σ be a change of variables of $A : t/\alpha, J$. Let $red(A : t/\alpha, J; \sigma) = B$. Then $B : t$ and moreover*

$$\models A[\sigma] \leftrightarrow B/\alpha, J : t/\alpha, J.$$

If $A : t/\alpha, J$ is closed then $(\models A : t/\alpha, J)$ iff $(\models B : t)$.

4.5 Reduction for restriction

Any property of a restriction $t \restriction \Lambda$ can be regarded as a property of the component t; define

$$out : P(t \restriction \Lambda) \to P(t)$$

by taking $out(U) = U$. The inverse map takes account of the fact that properties of $t \restriction \Lambda$ consist of states which are reachable via transitions within Λ. It is

$$in : P(t) \to P(t \restriction \Lambda)$$

defined by $in(V) = V \cap S_{t\restriction\Lambda}$—note this means $in(V) = V \restriction \Lambda$. Both maps are monotonic and together satisfy $in \circ out = 1_{P(t\restriction\Lambda)}$.

As usual the inverse map is associated with a change of variables:

Definition: Assume $A : t \restriction \Lambda$. A *change of variables* of $A : t \restriction \Lambda$ is a substitution σ, with domain $Var(t \restriction \Lambda)$, which is fresh for A, and of the form $\sigma(X) = Y \restriction \Lambda$ for variables $X : t \restriction \Lambda$ with $Y : t$ a variable.

With respect to a change of variables σ we define a reduction of pure assertions $A : t \restriction \Lambda$ to assertions $red(A : t \restriction \Lambda; \sigma) : t$ such that

$$\models A[\sigma] \leftrightarrow red(A : t \restriction \Lambda; \sigma) \restriction \Lambda : t \restriction \Lambda$$

The reduction is related to the embedding *out* in the sense that

$$[\![red(A : t \restriction \Lambda; \sigma) : t]\!]\rho \cap S_{t\restriction\Lambda} = out([\![A[\sigma] : t \restriction \Lambda]\!]\rho).$$

It is defined on pure assertion A for which $A : t \restriction \Lambda$ and reductions σ, on variables of them by the following structural induction:

$$
\begin{array}{lcl}
red(I : t \restriction \Lambda; \sigma) & = & I \\
red(T : t \restriction \Lambda; \sigma) & = & T \\
red(F : t \restriction \Lambda; \sigma) & = & F \\
red(A_0 \vee A_1 : t \restriction \Lambda; \sigma) & = & red(A_0 : t \restriction \Lambda, \sigma) \vee red(A_1 : t \restriction \Lambda; \sigma) \\
red(\neg A : t \restriction \Lambda; \sigma) & = & \neg red(A : t \restriction \Lambda; \sigma) \\
red(\langle\alpha\rangle A : t \restriction \Lambda; \sigma) & = & \begin{cases} F & \text{if } \alpha \notin \Lambda \\ \langle\alpha\rangle red(A : t \restriction \Lambda; \sigma) & \text{if } \alpha \in \Lambda \end{cases} \\
red(\overline{\langle\alpha\rangle} A : t \restriction \Lambda; \sigma) & = & \begin{cases} F & \text{if } \alpha \notin \Lambda \\ \overline{\langle\alpha\rangle}(R_\Lambda \wedge red(A : t \restriction \Lambda; \sigma)) & \text{if } \alpha \in \Lambda \end{cases} \\
 & & \text{where } R_\Lambda \equiv \mu X.I \vee \bigvee_{\beta \in \Lambda} \overline{\langle\beta\rangle} X \\
red(X : t \restriction \Lambda; \sigma) & = & Y \text{ where } \sigma(X) = Y \restriction \Lambda \\
red(\nu X.A : t \restriction \Lambda; \sigma) & = & \nu Y.red(A : t \restriction \Lambda; \sigma) \text{ where } \sigma(X) = Y \restriction \Lambda \\
red(A \restriction \Lambda : t \restriction \Lambda; \sigma) & = & A.
\end{array}
$$

Theorem 9 *Let A be a pure assertion such that $A : t \restriction \Lambda$, for which σ is a change of variables. Let $red(A : t\restriction\Lambda; \sigma) = B$. Then $B : t$ and*

$$\models A[\sigma] \leftrightarrow red(A : t \restriction \Lambda; \sigma) \restriction \Lambda : t \restriction \Lambda$$

If $A : t\restriction\Lambda$ is closed then $(\models A : t\restriction\Lambda)$ iff $(\models R_\Lambda \rightarrow B : t)$.

4.6 Reduction for relabelling

Because the properties $P(t)$ and $P(t\{\Xi\})$ are the same, the reductions and change of variables correspond to the identity map and are relatively straightforward.

Definition: Assume $A : t\{\Xi\}$. A *change of variables* of $A : t\{\Xi\}$ is a substitution σ, with domain $Var(t\{\Xi\})$, which is fresh for A, and of the form $\sigma(X) = Y\{\Xi\}$ for variables $X : t\{\Xi\}$ and $Y : t$.

With respect to a change of variables σ, we define a reduction of a pure assertion $A : t\{\Xi\}$ to an assertion $red(A : t\{\Xi\}; \sigma)$ by structural induction:

$$
\begin{array}{lcl}
red(I : t\{\Xi\};\sigma) & = & I \\
red(T : t\{\Xi\};\sigma) & = & T \\
red(F : t\{\Xi\};\sigma) & = & F \\
red(A_0 \vee A_1 : t\{\Xi\};\sigma) & = & red(A_0 : t\{\Xi\};\sigma) \vee red(A_1 : t\{\Xi\};\sigma) \\
red(\neg A : t\{\Xi\};\sigma) & = & \neg red(A : t\{\Xi\};\sigma) \\
red(\langle * \rangle A : t\{\Xi\};\sigma) & = & red(A : t\{\Xi\};\sigma) \\
red(\langle \alpha \rangle A : t\{\Xi\};\sigma) & = & \mathbb{W}_{\beta \in \Xi^{-1}\{\alpha\}} \langle \beta \rangle red(A : t\{\Xi\};\sigma) \\
red(\langle \overline{*} \rangle A : t\{\Xi\};\sigma) & = & red(A : t\{\Xi\};\sigma) \\
red(\langle \overline{\alpha} \rangle A : t\{\Xi\};\sigma) & = & \mathbb{W}_{\beta \in \Xi^{-1}\{\alpha\}} \overline{\langle \beta \rangle} red(A : t\{\Xi\};\sigma) \\
red(X : t\{\Xi\};\sigma) & = & Y \text{ where } \sigma(X) = Y\{\Xi\} \\
red(\nu X.A : t\{\Xi\};\sigma) & = & \nu Y.red(A : t\{\Xi\};\sigma) \text{ where } \sigma(X) = Y\{\Xi\} \\
red(A\{\Xi\} : t\{\Xi\};\sigma) & = & A.
\end{array}
$$

Theorem 10 *Let A be a pure assertion for such that $A : t\{\Xi\}$ for which σ is a change of variables. Let $red(A : t\{\Xi\};\sigma) = B$. Then $B : t$ and*

$$\models A[\sigma] \leftrightarrow B\{\Xi\} : t\{\Xi\}.$$

If $A : t\{\Xi\}$ is closed then $(\models A : t\{\Xi\})$ iff $(\models B : t)$.

4.7 Reduction for product

Looking back at the constructions so far, we see they share a common property, the presence of an embedding from properties of a constructed term to properties of its immediate components which are realised by the reductions on assertions of that term. Indeed, this reduction can be performed without looking at the composition of the immediate components; for instance, the reduction for $t_0 + t_1$ proceeds independently of the composition of t_0 and t_1.

The difficulty in obtaining analogous reductions for parallel compositions stems from there not being such an embedding from properties of products to properties of their components. While there is the map

$$(- \times -) : P(t_0) \times P(t_1) \longrightarrow P(t_0 \times t_1)$$

There is no 1-1 map in the converse direction if one of t_0, t_1 has more than one and the other more than two reachable states—a little arithmetic

shows that then the set $P(t_0 \times t_1)$ has more states than $P(t_0) \times P(t_1)$. Reduction for assertions of a product, in general, cannot follow the same scheme as that of the other constructions. We are obliged to look for a different method of embedding and reduction, or at special kinds of properties in $P(t_0 \times t_1)$ such as those which can embed in $P(t_0) \times P(t_1)$.

Properties having the shape $V_0 \times V_1$, a cartesian product of $V_0 \in P(t_0), V_1 \in P(t_1)$, are in correspondence with, and so embed in, properties $P(t_0) \times P(t_1)$. By cutting down the properties of a product to those denoted by the following assertions, we can obtain a reduction:

$$A ::= I \mid T \mid B \times C \mid \langle (a,b) \rangle A \mid \overline{\langle (a,b) \rangle} A \mid ((a,b))A \mid \overline{((a,b))}A \mid A \wedge A' \mid X \mid \nu X.\, A$$

where we use $(a)A$ to abbreviate $[a]A \wedge \langle \alpha \rangle T$ and $\overline{(a)}A$ for $\overline{[a]}A \wedge \overline{\langle a \rangle} T$. Any closed assertion A in this class, for which $A : t_0 \times t_1$, has the property that

$$\models A \leftrightarrow A_0 \times A_1 : t_0 \times t_1$$

for simply found $A_0 : t_0$, $A_1 : t_1$. These assertions for the components are obtained with respect to a change of variables for $t_0 \times t_1$ which is a substitution, fresh for A, sending variables $X : t_0 \times t_1$ to $Y_0 \times Y_1$, for distinct variables $Y_0 : t_0, Y_1 : t_1$. Define:

$$\begin{aligned}
red(I : t_0 \times t_1; \sigma) &= (I,\ I) \\
red(T : t_0 \times t_1; \sigma) &= (T,\ T) \\
red(\langle (a,b) \rangle A : t_0 \times t_1; \sigma) &= (\langle a \rangle A_0,\ \langle b \rangle A_1) \\
red(\overline{\langle (a,b) \rangle} A : t_0 \times t_1; \sigma) &= (\overline{\langle a \rangle} A_0,\ \overline{\langle b \rangle} A_1) \\
red(((a,b))A : t_0 \times t_1; \sigma) &= ((a)A_0,\ (b)A_1) \\
red(\overline{((a,b))}A : t_0 \times t_1; \sigma) &= (\overline{(a)}A_0,\ \overline{(b)}A_1) \text{ where } red(A : t_0 \times t_1; \sigma) = (A_0,\ A_1) \\
red(A \wedge A' : t_0 \times t_1; \sigma) &= ((A_0 \wedge A'_0),\ (A_1 \wedge A'_1)) \text{ where} \\
red(A : t_0 \times t_1; \sigma) &= (A_0,\ A_1),\ red(A' : t_0 \times t_1; \sigma) = (A'_0,\ A'_1) \\
red(X : t_0 \times t_1; \sigma) &= (Y_0,\ Y_1) \text{ where } \sigma(X) = Y_0 \times Y_1 \\
red(\nu X.\, A : t_0 \times t_1; \sigma) &= ((\nu Y_0.\, A_0),\ (\nu Y_1.\, A_1)) \\
\text{where } \sigma(X) &= Y_0 \times Y_1 \text{ and } red(A : t_0 \times t_1; \sigma) = (A_0,\ A_1)
\end{aligned}$$

While this reduction works for a nontrivial class of assertions the class is limited. In particular, it does not include the 'reachability' assertions $R_\Lambda \equiv \mu X.I \vee \bigvee_{\beta \in \Lambda} \overline{\langle \beta \rangle} X$, true of those states which are reachable from

the initial state purely by transitions with labels in Λ. However, in the special case where

$$(\lambda_0, \lambda_1) \in \Lambda \;\&\; \lambda_0 \neq * \Rightarrow (\lambda_0, *) \in \Lambda, \quad \text{and}$$
$$(\lambda_0, \lambda_1) \in \Lambda \;\&\; \lambda_1 \neq * \Rightarrow (*, \lambda_1) \in \Lambda \qquad (+)$$

we have that

$$\models R_\Lambda \leftrightarrow R_{\Lambda_0} \times R_{\Lambda_1} : t_0 \times t_1$$

where $\Lambda_0 = \{\lambda_0 \mid \exists \lambda_1.\ (\lambda_0, \lambda_1) \in \Lambda\}$, $\Lambda_1 = \{\lambda_1 \mid \exists \lambda_0.\ (\lambda_0, \lambda_1) \in \Lambda\}$. As we shall see, this has implications for reductions with respect to the parallel composition of Milner's CCS.

Any recursion-free assertion of a product can be routinely transformed into a finite disjunction $\mathbb{W}_{i \in I} B_i \times C_i$ (though conjunctions cause a quadratic 'blow-up' in size and negations an exponential 'blow-up'). As we have seen, there are special cases of recursive assertions where this can be achieved too. Once we have a property of a product $t_0 \times t_1$ expressed in such a form, the following result provides a method for reducing its validity to validities in the components t_0, t_1. Note the result is independent of the composition of t_0 and t_1.

Proposition 11 *Suppose* $\mathbb{W}_{i \in I}\ B_i \times C_i : t_0 \times t_1$ *is a finite disjunction.*

$$\models \mathbb{W}_{i \in I}\ B_i \times C_i : t_0 \times t_1$$

iff

$$(\models \mathbb{W}_{j \in J}\ B_j : t_0) \text{ or} (\models \mathbb{W}_{k \in K}\ C_k : t_1)$$

for all partitions $J \dot{\cup} K = I$.

It is useful to generalise the above proposition a little, so we can establish a property of a product relative to assumptions on the states of each component. This paper has concentrated on 'backwards proof'; given a goal for a compound term it has addressed how to reduce this to subgoals for its immediate components. One use of the following proposition is when asking the converse: if $\models B : t_0$ and $\models C : t_1$ does it follow that $\models A : t_0 \times t_1$? The proposition provides a partial answer—it depends on A having been expressed as a finite disjunction $\models \mathbb{W}_{i \in I}\ B_i \times C_i : t_0 \times t_1$. Then:

Proposition 12 *Suppose* $\mathbb{W}_{i\in I}\ B_i \times C_i : t_0 \times t_1$ *is a finite disjunction. Let* $B : t_0$, $C : t_1$. *Then*

$$\models B \times C \to \mathbb{W}_{i\in I}\ B_i \times C_i : t_0 \times t_1$$

iff

$$(\models B \to \mathbb{W}_{j\in J}\ B_j : t_0)\ \text{or}(\models C \to \mathbb{W}_{k\in K}\ C_k : t_1)$$

for all partitions $J \dot{\cup} K = I$.

As a corollary of this proposition, we obtain reduction results for certain parallel compositions, including that of CCS. A parallel composition of t_0, t_1 has the form

$$t_0 \parallel t_1 \equiv ((t_0 \times t_1)\restriction\Lambda)\{\Xi\}$$

A problem $\models A : t_0 \parallel t_1$ reduces to

$$\models R_\Lambda \to A' : t_0 \times t_1$$

where A' is obtained by carrying out the reductions for relabelling, then restriction. (Note the reduction for restriction, as expressed by theorem 9, introduces the reachability assertion R_Λ.) Now, in the case of $t_0|t_1$, for CCS, the appropriate restriction is with respect to a subset Λ satisfying (+). It follows that

$$\models R_\Lambda \leftrightarrow R_{\Lambda_0} \times R_{\Lambda_1} : t_0 \times t_1$$

for reachability assertions R_{Λ_0} and R_{Λ_1}. Hence if A' can be expressed as a manageable disjunction $\mathbb{W}_{i\in I}\ B_i \times C_i$ we have a reductive way of checking $\models A : t_0|t_1$:

$$\models A : t_0|t_1$$

iff

$$(\models R_{\Lambda_0} \to \mathbb{W}_{j\in J}\ B_j : t_0)\ \text{or}(\models R_{\Lambda_1} \to \mathbb{W}_{k\in K}\ C_k : t_1)$$

for all partitions $J \dot{\cup} K = I$. Certainly, A' can be so decomposed when the original assertion A contains no variables. Note that even for assertions A without recursion, the statement $\models A : t_0 \times t_1$, being one of validity, can express a nontrivial invariant of $t_0 \times t_1$. It is emphasised that again this reduction does not depend on the structure of t_0 and t_1.

Unfortunately, the important use of restriction in CCS to internalise communication along a channel does not use a restricting set satisfying (+).

The problem with checking validity for terms which force internal communication along a channel centres on the difficulty of expressing

$$R_{\{(a,b)\}} \equiv \mu X.\ (I \vee \overline{\langle (a,b) \rangle} X),$$

true of those states in a product which are reachable from the initial state via a sequence of (a,b)-transitions, as a finite, and manageable, disjunction $\bigvee_{i \in I} B_i \times C_i : t_0 \times t_1$. Of course, once we know the size of the transition system $t_0 \times t_1$ to be k, we have

$$\models R_{\{(a,b)\}} \leftrightarrow \bigvee_{n<k} \overline{\langle a \rangle}^n I \times \overline{\langle b \rangle}^n I : t_0 \times t_1.$$

With luck, the recursion might become stationary at an earlier point, but to be a valid equivalence for *all* transition systems of size k all the k disjuncts have to be included. This reduction is thus quadratic in the size of the transition system. Certainly in this case the reduction can no longer be independent of t_0, t_1, with the assertion language as it stands presently.

Conclusion

General methods have been provided for reasoning compositionally with a modal ν-calculus. These methods are presently being implemented by Henrik Andersen at Aarhus. Henrik has also extended the reductions to cope with a more traditional recursive definition of processes, which could be used in place of looping. The introduction of process variables which this entails could be useful for other reasons. Because all the reductions are directed only by the top-level operation on terms they might well be helpful in synthesising a process satisfying a specification, using process variables to leave parts of terms unspecified.

There remain important properties of products which do not seem directly amenable to the techniques outlined here. It is notable though that some nontrivial assertions have reductions which are independent of the structure of the components of a product. It is hoped that the techniques and limitations exposed here will help guide the search for methods of reasoning about parallel processes. Promising leads may be found in [CLM] and [LX].

The approach here can be understood as running the compositional proof system of [W] backwards, and relates to the more modest compositional proof systems of [St] and [W1], and more superficially to [GS]. The reductions however have a fuller treatment of assertion variables than the proof system of [W]; the latter should be redone so that it supports the reasoning given by the reductions in a forwards direction and makes plain the sense in which the reductions correspond to running the proof system backwards.

Acknowledgements

I would like to acknowledge the support of the Esprit Basic Research Actions CEDISYS and CLICS. I've had valuable conversations with Henrik Andersen. Thanks to the referees for their comments.

References

[CLM] Clarke, E., Long, D., and McMillan, K., Compositional model checking. LICS 1989.

[EL] Emerson, A. and Lei, C., Efficient model checking in fragments of the propositional mu-calculus. Proc.LICS, 1986.

[GS] Graf, I., and Sifakis, J., A logic for the description of nondeterministic programs and their properties. Report IMAG RR 551, Grenoble, France, 1985.

[LX] Larsen,K.G. and Xiuxin, L., Compositionality through an operational semantics of contexts. Proc. ICALP 90, 1990.

[St] Stirling, C, A complete modal proof system for a subset of SCCS. LNCS 185, 1985.

[W1] Winskel, G., A complete proof system for SCCS with modal assertions. In the proceedings of Foundations of Software Technology (1985).

[W] Winskel, G., A compositional proof system on a category of labelled transition systems. To appear in Information and Computation, 1990.