Unfold/fold transformations preserving termination properties

Torben Amtoft
Computer Science Department
Aarhus University
Ny Munkegade, building 540
DK-8000 Århus C, Denmark
nternet: tamtoft@daimi.aau.dk

August 19, 1992

Abstract

The unfold/fold framework constitutes the spine of many program transformation strategies. However, by unrestricted use of folding the target program may terminate less often than the source program. Several authors have investigated the problem of setting up conditions of syntactic nature, i.e. not based on some well-founded ordering of the arguments, which guarantee preservation of termination properties. These conditions are typically formulated in a way which makes it hard to grasp the basic intuition why they work, and in a way which makes it hard to give elegant proofs of correctness. The aim of this paper will be to give a more unified treatment by setting up a model which enables us to reason about termination preservation in a cleaner and more algebraic fashion. The model resembles a logic language and is parametrized with respect to evaluation order, but it should not be too difficult to transfer the ideas to other languages.

A summary of this work is reported in [Amt92].

1 Introduction

The unfold/fold framework for program transformation dates bank to (at least) [BD77] and has since been the subject of much interest, primarily aimed at making the process of finding "eureka"-definitions more systematic, e.g. [Wad90], [NN90], [PP91b]. Also supercompilation [Tur86] can be seen as a variant over the concept.

A major problem with the technique is that one, due to "too much folding", may risk that the program resulting from transformation (the *target program*) loops while the original (the *source program*) does not. A classical example is the following, expressed in a logic language: suppose we have a source program containing the clauses

$$p(X) \leftarrow q(X);\ q(a) \leftarrow \Box$$

Here the query $p(X)$ will succeed with answer substitution $\{X \rightarrow a\}$. However, if one *folds* the first clause of the program against itself, the target program will contain the clause

$$p(X) \leftarrow p(X)$$

and now the query $p(X)$ will loop (i.e. neither succeed nor fail).

By innocent abuse of terminology, we will say that a transformation is partially correct iff each time the target program terminates with some result also the source program terminates and with the same result; whereas we will define total correctness to mean partial correctness together with the condition that if the target program does not terminate then neither does the source program. Whether a transformation process itself terminates is beyond the scope of this paper, but e.g. [Wad90], [PP91b] address this problem for certain transformation strategies.

Several ways to guarantee total correctness have been proposed in the literature, e.g. [TS84], [KK90], [Sek91], [Kot85], [GS91], [PP91a]. They all work by putting forward some restrictions on the types of foldings allowed. For a more detailed description (and comparison with our appreach), see section 3.

The purpose of this paper is to present a model for unfold/fold-transformations which enables one to express conditions which are provably sufficient for total correctness. We want the model to include (most of) the results

from the literature as special cases (after the frameworks in question have been encoded into our framework); and we want the model to have a clean algebraic structure.

Our framework is primarily aimed at modeling logic programming - even though the machinery differs from the one usually used when treating logic languages, as done in e.g. [Llo84], [Søn89]. However, we believe that the main ideas can be carried over to other types of languages as well.

The meaning of programs will be defined in terms of a transition semantics (cf. [Plo81]). The reason for this is that we feel this is more appropriate for capturing the essence of unfolding and folding: unfolding corresponds to a transition being made in the "right" direction; folding corresponds to a transition being made in the "wrong" direction. By using a denotational approach, this cannot be expressed directly. We believe that the reason why conditions for unfold/fold transformations to be termination preserving apparently is a more hot topic in the logic programming community than in the functional community is that in the former operational semantics (typically derivation trees) has a more respectable status than in the latter.

1.1 An overview of this paper

The aim of section 2 will be to give the reader a flavor of the main features of our model. In particular.

- in section 2.1 we introduce the concept of *multilevel transition systems*, a way to model the relationship between evaluating the source program, transforming the source program and evaluating the target program.
- in section 2.2 we introduce the concept of *U-mirrors*, a representation of (the control part of) an unfold/fold transformation which facilitates reasoning about preservation of termination properties.
- in section 2.3 we focus upon the data aspect, which is usually modeled by means of substitutions - we will propose an alternative approach.
- in section 2.4 we discuss when it is permissible to fold against a given clause - not wrt. total correctness, but wrt. partial correctness.
- in section 2.5 we discuss how to ensure total correctness. Various evaluation strategies are considered.

- in section 2.6 we discuss how to extend the model such that it is able to represent the whole search tree and not only a single branch.

Section 2 will be rather informal, based on examples and intuition. All concepts introduced will be formally defined and all theorems will be proved in the subsequent sections. Section 3 compares with related work.

In section 4, the basic machinery is set up, e.g. concerning configurations transitions and U-mirrors. In section 5, a multilevel transition system is defined. In section 6, we state and prove various theorems concerning sufficient conditions for total correctness. In section 7 the whole story is repeated, transitions now representing search trees instead of single branches - here some proofs will appeal rather heavily to intuition, but of course these may be formalized at the expense of decreased clarity.

First, however, we give a "realistic" example of the unfold/fold technique:

Example 1.1 Consider the following source program, written in a logic language:

$f([], []) \leftarrow \Box$
$f([N \mid U], [\mathsf{s}(N) \mid V]) \leftarrow f(U, V)$
$g(X, Z) \leftarrow f(X, Y), f(Y, Z)$

Operationally, f adds one to each element in a list of unary numbers. Thus g will traverse its input list X twice. Our aim will be to make a target program where g only traverses its input list once: first consider the configuration $g([], Z)$. This can be unfolded into the configuration $f([], Y), f(Y, Z)$. By unfolding the first f, Y gets bound to [] and we arrive at the configuration $f([], Z)$. Now this f can be unfolded, binding Z to []. We are thus able to let the target program contain the rule

$$g([], []) \leftarrow \Box \tag{1}$$

Next consider the configuration $g([N \mid X], Z)$. This can be unfolded into $f([N \mid X], Y), f(Y, Z)$. By unfolding the first f, Y gets bound to $[\mathsf{s}(N) \mid Y1]$ and we get the configuration $f(X, Y1), f([\mathsf{s}(N) \mid Y1], Z)$. By unfolding the second f, Z gets bound to $[\mathsf{s}(\mathsf{s}(N)) \mid Z1]$ an we arrive at the configuration $f(X, Y1), f(Y1, Z1)$. As $Y1$ is *a new unbound variable*, this can now be folded back into the configuration $g(X, Z1)$ We are thus able to let the target program contain the rule

$$g([N \mid X], [\mathsf{s}(\mathsf{s}(N)) \mid Z1]) \leftarrow g(X, Z1) \tag{2}$$

Now consider the "query" $g([0,0],Z)$. If the target program is used "solve" this query, it is first rewritten into $g([0],Z1)$ binding Z to $[\mathsf{s}(\mathsf{s}(0)) \mid Z1)$; then rewritten into $g([],Z2)$ binding $Z1$ to $[\mathsf{s}(\mathsf{s}(0)) \mid Z2)$ and finally rewritten into the empty configuration, binding $Z2$ to []. Thus the query is solved using three inference steps, and Z has been bound to $[\mathsf{s}(\mathsf{s}(0)),\mathsf{s}(\mathsf{s}(0))]$.

It is easily seen that the same query, $g([0,0],Z)$, also can be solved with the same binding to Z by using the source program - but then seven inference steps are needed. □

2 An outline of the theory

2.1 Modeling the transformation process

The key idea is to model computation as transitions between configurations, and to model a program as a collection of distinguished transitions to be called *rules*. We have a hierarchy as follows:

1. The source program is represented as *rules at level 0*. That t is a rule at level 0 is written $t \in \mathcal{RU}_0$.

2. As soon as the rules at level 0 have been given a series of other entities will be fixed:

   - the set of *level 1 unfolding steps*. That t is a level 1 unfolding step from B to B' intuitively means that B' can be derived by unfolding one of the atoms in B, using a rule in $\mathcal{RU}_0$.
   - the set of *level 1 folding steps*. That t is a level 1 folding step from B to B' intuitively means that B' can be derived from B by performing one folding step, using a rule in $\mathcal{RU}_0$.
   - the set of *level 1 unfoldings*. That t is a level 1 unfolding from B to B' means that B' can be derived by from B by a sequence of unfoldings, using rules in $\mathcal{RU}_0$.
   - the set of *level 1 foldings*. That t is a level 1 folding from B to B' means that B' can be derived from B by a sequence of foldings, using rules in $\mathcal{RU}_0$.
   - the set of *level 1 transitions*. That t is a level 1 transition from B to B' means that B' can be derived by from B by a sequence of unfoldings and foldings, using rules in $\mathcal{RU}_0$.

Level 1 unfoldings model standard evaluation of the source program; whereas level 1 transitions model transformation ("symbolic evaluation") of the source program.

3. Among all level 1 transitions, some are chosen to be *rules at level 1* - that t is a rule at level 1 is written $t \in \mathcal{RU}_1$. These rules represent the target program.

4. As soon as the rules at level 1 have been given, the *level 2 unfolding steps* and the *level 2 unfoldings* are fixed. That t is a level 2 unfolding step (unfolding) from B to B' means that B' can be derived from B by a (sequence of) unfoldings, using rules in $\mathcal{RU}_1$.

   Level 2 unfoldings thus model (standard) evaluation of the target program.

For inference rules determining the set of level 1 unfoldings etc, see section 5. Not surprisingly, it will hold that if t is a level 2 unfolding it also is a level 1 transition.

A key point of our approach is that "standard evaluation" (the level 1 unfoldings) is a special case of "symbolic evaluation" (the level 1 transitions) - this greatly facilitates reasoning about the properties of the target program. This lack of distinction between standard evaluation and symbolic evaluation comes almost for free in a logic language, but also in the functional world one gains from viewing the latter as a generalization of the former [DP88]. However, an important difference between standard and symbolic evaluation is that during symbolic evaluation any atom in the goal sequence may be unfolded, whereas during standard evaluation one for efficiency reasons often chooses a fixed strategy, typically the strategy always to unfold the leftmost goal - this strategy will be denoted $\mathcal{LR}$.

2.2 Modeling control

A transition t from B to B' will be represented by means of *U-mirrors*: intuitively speaking, a U-mirror is a triple (f, f', B'') where f is a *U-forest* describing how to get from B to B'' by means of unfoldings using level 0 rules; and f' is a U-forest describing how to get from B' to B'' by means of unfoldings using level 0 rules. Thus, if t is a level 1 unfolding f' will be trivial; and if t is a level 1 folding f will be trivial. U-mirrors will be treated in depth in section 4.2.

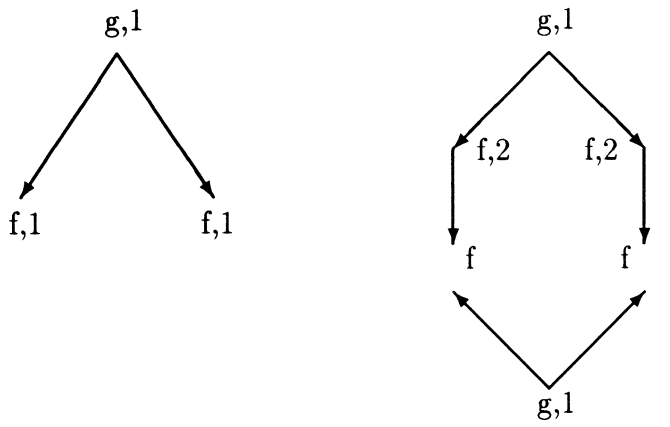

Figure 1: Two U-mirrors

In figure 1 is depicted the U-mirrors corresponding to the level 1 rules (1) and (2) from example 1.1. That the two leaves of the first U-mirror are labeled "$f, 1''$ is because the two occurrences of f were unfolded when deriving the rule, in both cases using the first level 0 rule for f. That two internal nodes in the second U-mirror are labeled "$f, 2''$ is because the two occurrences of f were unfolded when deriving the rule, in both cases using the second level 0 rule for f.

Now suppose the target program loops on some configuration B, i.e. there exists an infinite sequence of level 2 unfolding steps from B. As each such unfolding step is represented by the second U-mirror in figure 1, it is easily seen - as a folding into g is "canceled" by a subsequent unfolding of g - that this means that from B there exists an infinite sequence of level 1 unfolding steps, where the first step unfolds g and the remaining steps unfold f. This informally shows the total correctness of the transformation.

Of course, it is also possible to argue for total correctness by observing that the first argument to g gets "smaller" for each inference step (assuming that g is called with a first argument which is fully instantiated). However, the virtue of the abovementioned way of reasoning is that it only depends on the syntactic structure of the transformation process.

2.3 Modeling data

A configuration is a sequence of goals together with some information about which values the variables in the goals can assume. One usually represents this information as a substitution, cf. [Llo84]. As substitutions 7are hard to reason about from an algebraic point of view (even though e.g. [Søn89] and [Pal89] show that certain sets of substitutions carry some structure), in particular one has to be careful about renaming, we will represent the information as a family of sets of ground values, to be called an *information family*. For a full account of configurations and operations on these, see section 4.1.

We will now briefly sketch how the standard framework translates into our framework. In the following, assume that $\mathcal{D}$ is a universal data domain.[1]

As an example take the goal sequence $(p(X), q(Y), r(Z))$ together with the substitution $\{X \rightarrow f(Y), Z \rightarrow a\}$. This in our framework could be represented as the goal sequence (p, q, r) together with the $\mathcal{D}$-indexed family where the d'th element is the singleton set $\{(f(d), d, a)\}$ - on the other hand, one might also use the family consisting of one element only, namely the set $\{(f(d), d, a) \mid d \in \mathcal{D}\}$. The latter representation will be needed for dealing with "variables occurring on the right hand side but not on the left hand side"; we will elaborate on this issue in section 2.4.

We now return to example 1.1. The level 0 rule $f([], []) \leftarrow \Box$ is represented as a transition from $B_1 = ([f], Q_1)$ to $B_1' = ([], Q_1')$, where Q_1 and Q_1' are $\mathcal{D} \times \mathcal{D}$-indexed families with $Q_1(d_1, d_2) = \{(d_1, d_2)\}$, $Q_1'([], []) = \{()\}$, $Q_1'(d_1, d_2) = \emptyset$ if $d_1 \neq []$ or $d_2 \neq []$.

The level 0 rule $f([N \mid U], [\mathsf{s}(N) \mid V) \leftarrow f(U, V)$ is represented as a transition from $B_2 = ([f], Q_2)$ to $B_2' = ([f], Q_2')$ where the $\mathcal{D} \times \mathcal{D}$-indexed information families Q_2 and Q_2' are given by $Q_2(d_1, d_2) = \{(d_1, d_2)\}$, $Q_2'([d_n \mid d_u], [\mathsf{s}(d_n) \mid d_v]) = \{(d_u, d_v)\}$ and $Q_2'(d_1, d_2) = \emptyset$ if (d_1, d_2) is not of the form above.

In the standard framework, the query $f([0], Z)$ is solved yielding an answer substitution where Z is bound to $[\mathsf{s}(0)]$. Now consider how this works in our framework. There the query $f([0], Z)$ is represented as the configuration $B = ([f], Q)$ where the $\mathcal{D}$-indexed information family Q is given by $Q(d) = \{([0], d)\}$. Now consider the mapping s from $\mathcal{D}$ to $\mathcal{P}(\mathcal{D} \times \mathcal{D})$ given

[1] We will impose no requirements on the structure of this set; in our examples, however, we shall assume the elements of $\mathcal{D}$ to be PROLOG ground terms, i.e. terms built inductively from some set of functors (constants just being zero-arity functors).

by $s(d) = Q(d)$. Then for all d it will trivially hold that

$$Q(d) = \bigcup_{(d_1,d_2)\in s(d)} Q_2(d_1, d_2)$$

The existence of this s shows that B is an instance of B_2, to be written $B = \mathcal{I}_s(B_2)$. Then there will be a level 1 unfolding step from B to $\mathcal{I}_s(B_2')$, to be denoted B'. $B' = ([f], Q')$ where Q' is a $\mathcal{D}$-indexed family given by

$$Q'(d) = \bigcup_{(d_1,d_2)\in s(d)} Q_2'(d_1, d_2)$$

That is, $Q'([\mathsf{s}(0) \mid d_v]) = \{([], d_v)\}$ and $Q'(d) = \emptyset$ otherwise.

Next consider the mapping s' from $\mathcal{D}$ to $\mathcal{P}(\mathcal{D} \times \mathcal{D})$ given by $s'(d) = Q'(d)$. Then for all d it will trivially hold that

$$Q'(d) = \bigcup_{(d_1,d_2)\in s'(d)} Q_1(d_1, d_2)$$

This means that $B' = \mathcal{I}_{s'}(B_1)$. Then there will be a level 1 unfolding step from B' to $\mathcal{I}_{s'}(B_1')$, to be denotes B''. $B'' = ([], Q'')$ where Q'' is a $\mathcal{D}$-indexed family given by

$$Q''(d) = \bigcup_{(d_1,d_2)\in s'(d)} Q_1'(d_1, d_2)$$

That is, $Q''([\mathsf{s}(0) \mid d_v) = Q_1([], d_v)$ and $Q''(d) = \emptyset$ otherwise, i.e. $Q''([\mathsf{s}(0)]) = \{()\}$ and $Q''(d) = \emptyset$ otherwise. As $Q''(d) \neq \emptyset$ iff $d = [\mathsf{s}(0)]$, this corresponds to Z being bound to $[\mathsf{s}(0)]$ in the standard model.

Of course, also $B' = \mathcal{I}_{s'}(B_2)$. So there also is a level 1 unfolding from B' to $\mathcal{I}_{s'}(B_2') = B'''$, where $B''' = ([f], Q''')$. However, it is easily seen that $Q'''(d) = \emptyset$ for all d - we say that B''' is a failure configuration. Thus the transition from B' to B''' represents a failure branch.

In our examples we will, for ease of exposition, often switch back and forth between the standard model and our model when it is the control aspect which has our primary interest.

2.4 Modeling folding

Let some predicate symbol G be given, and let the level 0 rules for G be of form $\{t_i \mid i \in I\}$, each t_i going from B to B_i. Here B contains goal sequence

G and K-indexed information family Q, and each B_i contains K-indexed information family Q_i.

Given $i \in I$. Now suppose s, a mapping from K' to $\mathcal{P}(K)$, is such that

1. with Q'_i the K'-indexed information family of $\mathcal{I}_s(B_i), Q'_i(k') \neq \emptyset$ for all $k \in K'$
2. $\mathcal{I}_s(B_{i'})$ is failure for $i' \neq i$
3. B_i consists of a non-empty goal sequence.

Then it will be possible to make a level 1 folding step from $\mathcal{I}_s(B_i)$ to $\mathcal{I}_s(B_i)$, cf. the definition in section 5.4.

The rationales for the above requirements are as follows:

1. It must not be possible to make a folding step from a failure configuration into a non-failure configuration. To see whys consider the two program clauses:

   $p(a) \leftarrow q(a);\ q(X) \leftarrow q(X)$

   Starting with the configuration $p(X)$, one may consider unfolding it into $q(a)$, then unfold it once more into q(a), and finally (erroneously!) fold back into $p(X)$ - thus deriving the target program $p(X) \leftarrow p(X)$. As two unfoldings and only one folding is made, the reasoning in section 2.2 may tempt us to believe that this transformation preserves termination properties. However, e.g. the goal $p(b)$ loops at level 2 (i.e. when using the target program); while it fails when evaluated at level 1. This is because the infinite sequence of level 2 unfolding steps $p(b) \rightarrow p(b) \rightarrow \ldots$ corresponds to the sequence of level 1 unfold/fold steps where $p(b)$ is unfolded into failure which then is unfolded into failure which then is folded back to $p(b)$ etc.

   To see why the folding from $q(a)$ to $p(X)$ is not a level 1 folding step in our model, notice that the clause $p(a) \leftarrow q(a)$ is represented as a transition from B to B_1, containing $\mathcal{D}$-indexed information families Q and Q_1 respectively. Here $Q(d) = \{d\}$ for all $d \in \mathcal{D}$, while $Q_1(a) = \{a\}$ and $Q_1(d) = \emptyset$ for $d \neq a$.

2. This in the standard framework is modeled by the requirement that only one clause defining the predicate folded against should match: if we have two program clauses

$p \leftarrow q;\ p \leftarrow r$

it must not be possible to fold r into p and then unfold into q - this would destroy semantics.

3. If there is a source program clause $p \leftarrow \Box$, it should not be possible to fold e.g. q into q, p. Such foldings never occur in practice, and it is convenient to exclude them: otherwise we above could derive the target program $p \leftarrow p$, and then p would loop at level 2 but as the corresponding level 1 transition unfolds p into [] which then is folded back into p etc, p does not loop at level 1.

When folding in a logic language, one has to be careful when folding against a clause containing variables not occurring in the head. This is a problem to which some incorrect solutions have been proposed in the literature (and yet proved correct!), for a survey see [GS91].

In our framework, this problem is solved "for free": consider e.g. the clause from example 1.1 $g(X,Z) \leftarrow f(X,Y), f(Y,Z)$ which is represented as a transition from $B = ([g], \{\{(d_1,d_2)\} \mid d_1, d_2 \in \mathcal{D}\})$ to $B' = ([f,f], Q')$ where the $\mathcal{D} \times \mathcal{D}$-indexed information family Q' is given by $Q'(d_1,d_2) = \{(d_1,d,d,d_2) \mid d \in \mathcal{D}\}$. Now suppose $B'' = \mathcal{I}_s(B')$ for some s, with $B'' = ([f,f], Q'')$. For any k (in the domain of s) we have

$$Q''(k) = \bigcup_{(d_1,d_2) \in s(k)} Q'(d_1,d_2)$$

so if $Q''(k)$ contains an element of the form (d_1,d,d,d_2) then for all $d' \in \mathcal{D}$ also $Q''(k)$ contains (d_1,d',d',d_2). This means - switching back to the standard framework - that if Y is instantiated in $f(X,Y), f(Y,Z)$ then this configuration cannot be written on the form $\mathcal{I}_s(B')$.

On the other hand, if Y is uninstantiated and a "new variable" then it is possible to fold back into g, as done when deriving rule (2) in example 1.1. Let us do so, within our framework: we start with the configuration $B_1 = ([g], Q_1)$ where the $\mathcal{D} \times \mathcal{D} \times \mathcal{D}$-indexed information family Q_1 is given by

$$Q_1(d_1,d_2,d_3) = \{([d_1 \mid d_2], d_3)\}$$

By unfolding g, we get the configuration $B_2 = ([f,f], Q_2)$ where the $\mathcal{D}^3$-indexed information family Q_2 is given by

$$Q_2(d_1,d_2,d_3) = \{([d_1 \mid d_2], d, d, d_3) \mid d \in \mathcal{D}\}$$

Now the first f is unfolded, and we get the configuration[2] $B_3 = ([f, f], Q_2)$ where the $\mathcal{D}^3$-indexed information family Q_3 is given by

$Q_3(d_1, d_2, d_3) = \{(d_2, d', [\mathsf{s}(d_1) \mid d'], d_3) \mid d' \in \mathcal{D}\}$

By unfolding the second f we get the configuration $B_4 = ([f, f], Q_4)$ where the $\mathcal{D}^3$-indexed information family Q_4 is given by

$Q_4(d_1, d_2, [\mathsf{s}(\mathsf{s}(d_1)) \mid d_3']) = \{(d_2, d', d', d_3') \mid d' \in \mathcal{D}\}, Q_4(d_1, d_2, d_3) = \emptyset$otherwise

Now define s, a mapping from $\mathcal{D}^3$ to $\mathcal{P}(\mathcal{D}^2)$, as follows:

$s(d_1, d_2, [\mathsf{s}(\mathsf{s}(d_1)) \mid d_3']) = \{(d_2, d_3')\}, s(d_1, d_2, d_3) = \emptyset$ otherwise

Then we have $B_4 = \mathcal{I}_s(B')$, as

$$Q_4(d_1, d_2, d_3) = \bigcup_{(d_1', d_2') \in s(d_1, d_2, d_3)} Q'(d_1', d_2')$$

So then we can fold B_4 into $B_5 = \mathcal{I}_s(B)$, with $B_5 = ([g], Q_5)$ where the $\mathcal{D}^3$-indexed information family Q_5 is given by $Q_5 = s$, ie.

$Q_5(d_1, d_2, [\mathsf{s}(\mathsf{s}(d_1)) \mid d_3']) = \{(d_2, d_3')\}, Q_5(d_1, d_2, d_3) = \emptyset$ otherwise

The level 1 transition from B_1 to B_5, translated into the standard framework, is just rule (2).

2.5 Conditions for total correctness

Consider the program
$E(a) \leftarrow A$
$E(b) \leftarrow B$
$E(X : Y) \leftarrow E(X), E(Y)$
$A \leftarrow B, B$
...

Starting with $E(a)$, we can unfold this into A and further into B, B. This can be folded bank into $B, E(b)$ into $E(b), E(b)$ and finally folded back into

[2]It will be a good exercise for the reader to check this, after having read section 5.2.

$E(b : b)$, yielding a target program

$E(a) \leftarrow E(b : b)$

As two unfolding steps and three folding steps have been made, the reasoning technique from section 2.2 cannot be used to show total correctness of the transformation. However, we can argue that the clause above represents some progress in the computation process, as A is unfolded into B, B but never folded back. This can be formalized by assigning *weights* (non-negative numbers) to the arcs in the U-mirrors representing a transition, such that the weight of an arc is a function of the predicate symbol being unfolded.[3] We can now define the weight of a path in a U-mirror (f, f') as the sum of the weights encountered when walking along the path, where the weights of arcs in f' are negated before contributing to the summation.

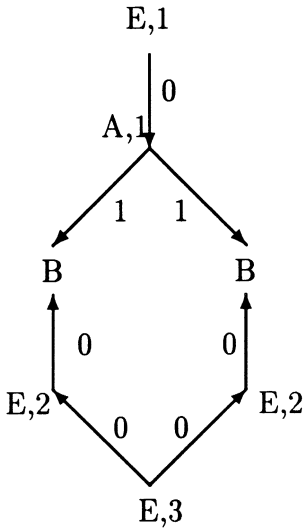

Figure 2: A U-mirror with weights

By assigning arcs from E weight 0 and arcs from A weight 1, the target program clause above is represented by the U-mirror depicted in figure 2. We see that all paths have weights 1 - but if we had assigned E weight 2 all paths would have weight -1.

Informally, a fair strategy sooner or later unfolds any goal. Then we have

[3]Actually, the weight may also depend on which rule is used and which conjunct the arc represents.

Condition 2.1 Suppose it for all level 1 rules holds that all the paths in the corresponding U-mirror have weight ≥ 1. Suppose B loops at level 2 by a fair strategy. Then B loops at level 1 by a fair strategy too. (This is theorem 6.2.)

Thus we have found a condition for a transformation to be total correct wrt. a fair evaluation strategy. Concerning total correctness wrt. the $\mathcal{LR}$ strategy, we have

Condition 2.2 Suppose it for all level 1 rules holds that the leftmost path in the corresponding U-mirror has weight ≥ 1. Suppose B loops at level 2 by the $\mathcal{LR}$ strategy. Then B loops at level 1 by the $\mathcal{LR}$ strategy too. (This is theorem 6.3.)

On the other hand, if the transformation does some non-$\mathcal{LR}$ steps it may happen that the domain of termination is *increased*. To see this, consider the source program

$p(X) \leftarrow q(X), r(X);\ q(a) \leftarrow q(a);\ r(b) \leftarrow \Box$

Starting with the configuration $p(X)$, this can be unfolded into $q(X), r(X)$ and then by a *non-*$\mathcal{LR}$ unfolding into $q(b)$, yielding the target program

$p(b) \leftarrow q(b)$

Now $p(X)$ terminates (and fails) at level 2 by any strategy, while $p(X)$ loops at level 1 by the $\mathcal{LR}$ strategy.

The same source program shows that it may happen that a transformation is total correct wrt. the $\mathcal{LR}$ strategy but not wrt. a fair strategy: again starting with the configuration p(X) we unfold this into $q(X), r(X)$ and then we unfold the leftmost atom yielding $q(a), r(a)$. This can be folded back into $p(a)$, yielding the target program

$p(a) \leftarrow p(a)$

It is easily seen that this transformation is total correct wrt. the $\mathcal{LR}$ strategy - $p(t)$ loops at level 2 (by the $\mathcal{LR}$ strategy) iff t can be unified with a iff $p(t)$ loops at level 1 by the $\mathcal{LR}$ strategy. This is as predicted by condition 2.2, since it is possible to assign weights in a way (e.g. 1 to q and 0 to p) such that the leftmost path of the U-mirror corresponding to this transformation has weight ≥ 1.

On the other hand, $p(X)$ loops at level 2 (by any strategy) but terminates at level 1 by a fair strategy. Thus the transformation is not total correct wrt.

a fair strategy.

Having defined the weight of a U-mirror (f, f') as the sum of the weights occurring in it, the weights occurring in f' negated, we can formulate a - less useful - condition:

Condition 2.3 Suppose it for all level 1 rules holds that the corresponding U-mirror has weight ≥ 1. Suppose B loops at level 2 by some strategy. Then B also loops at level 1, by some strategy. (This is theorem 6.1.)

This condition is not enough to guarantee total correctness (neither wrt. fair nor $\mathcal{LR}$ semantics): consider the source program $p(X) \leftarrow r(X), q(X)$; $q(a) \leftarrow q(a)$; $r(b) \leftarrow \Box$. By unfolding p; unfolding q and finally folding into p we get the level 1 rule $p(a) \leftarrow p(a)$. If q is assigned weight ≥ 1, the corresponding U-mirror will have weight ≥ 1. Now e.g. $p(a)$ loops at level 2 by any strategy, but fails at level 1 by a fair strategy as well as by the $\mathcal{LR}$ strategy.

2.6 Modeling the full search tree

So far a transition – for ease of exposition - only represents a single branch of the search tree, the transition system thus being non-confluent. In order to model the full search tree, configurations have to be *multisets* of "old" configurations (now to be called *basic configwations*). There are two reasons for working with multisets and not with sequences (i.e. not to order the branches), a pragmatic and a mathematical one:

- it is rather easy to implement or-parallelism [Gre87], as no communication has to occur between the branches. On the other hand, and-parallelism [Gre87] is much harder to implement due to the need for sharing of data, hence most implementations employ the $\mathcal{LR}$ strategy.

- If we use sequences, the Church-Rosser property will be lost. To see this, consider the program

  $a \leftarrow b$; $a \leftarrow c$; $d \leftarrow e$; $d \leftarrow f$;

  Now consider the goal (a, d) By first unfolding a and then unfolding d we first get $(b, d); (c, d)$ and then $B_1 = (b, e); (b, f); (c, e); (c, f)$. By first unfolding d and then unfolding a we first get $(a, e); (a, f)$ and then $B_2 = (b, e); (c, e); (b, f); (c, f)$. In [PP91a] one wants to distinguish

between B_1 and B_2, and therefore unfolding of the *leftmost* atom only is allowed (unless extra conditions are satisfied.)

A configuration is said to be in normal form if all the basic configurations belonging to it are non-failure and with an empty goal sequence. Due to the Church-Rosser property, it then for a (basic) configuration B makes sense to define $[\![B]\!]_1$ as follows: if there exists a C in normal form and a level 1 unfolding from B to C, $[\![B]\!]_1 = C$. Otherwise, $[\![B]\!]_1 = \bot$. In a similar vein, one can define $[\![B]\!]_2$. By restricting the level 1 (2) unfoldings in question to be $\mathcal{LR}$, one can define $[\![B]\!]_1^L$ ($[\![B]\!]_2^L$). Now condition 2.1 and 2.2 can be restated (a rule may now be represented by several U-mirrors):

Condition 2.4 Suppose that for all level 1 rules, represented by U-mirrors $m_1 \dots m_k$, it holds for all m_i that all paths in m_i have weight ≥ 1. Then for all B, $[\![B]\!]_2 = [\![B]\!]_1$.

Condition 2.5 Suppose that for all level 1 rules, represented by U-mirrors $m_1 \dots m_k$, it holds for all m_i that the leftmost path in m_i has weight ≥ 1. Then for all B, $[\![B]\!]_2 \geq [\![B]\!]_1$ (notice that the domain of termination may be increased, as shown in section 2.5).

For a more detailed treatment and for proofs, see section 7.

In one way, the expressive power is enhanced by working with the full search tree: we can fold a configuration containing several basic configurations back into a single basic configuration - resembling the process of converting a NFA into a DFA. As an example of this, consider the program

$ab([]) \leftarrow \Box; ab([a \mid X]) \leftarrow ab(X); ab([b \mid X]) \leftarrow ab(X)$

$bc([]) \leftarrow \Box; bc([b \mid X]) \leftarrow bc(X); bc([c \mid X]) \leftarrow bc(X)$

$abc(X) \leftarrow ab(X); abc(X) \leftarrow bc(X)$

Now consider the configuration $abc([])$. This is unfolded into $ab([]); bc([])$ which by two unfoldings yield $\Box; \Box$. The configuration $abc([a \mid X])$ is unfolded into $ab([a \mid X)]; bc([a \mid X])$ which by two unfoldings yield $ab(X)$ (as the second basic configuration is unfolded into failure). In a similar vein, the configuration $abc([c \mid X])$ is unfolded into $bc(X)$.

The interesting case is where we start with the configuration $abc([b \mid X])$. Then we unfold into $ab([b \mid X]); bc([b \mid X])$, two more unfoldings yield $ab(X); bc(X)$ and now this can be *folded back* into $abc(X)$. We have thus derived five new rules for abc:

$abc([]) \leftarrow \Box; abc([]) \leftarrow \Box;$

$abc([a \mid X]) \leftarrow ab(X); abc([c \mid X]) \leftarrow bc(X)$

$abc([b \mid X]) \leftarrow abc(X);$

To the latter rule correspond *two* U-mirrors, depicted in figure 3.

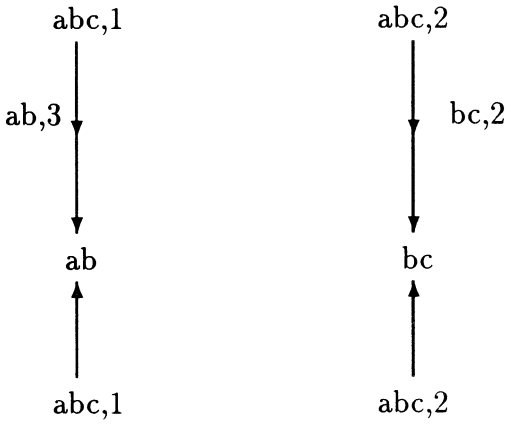

Figure 3: The two U-mirrors for $abc([b \mid X]) \leftarrow abc(X)$

3 Related work

In the literature on unfold/fold transformations in logic languages transformation typically proceeds in a "step by step fashion"; after a goal in the body of a clause has been unfolded the clause is *deleted* from the program and *replaced* by the clause resulting from the unfolding - this is the approach taken in e.g. [GS91], [KK90], [PP91a], [Sek91], [TS84]. As pointed out in [GS91], one by applying this method loses some power - to see this, consider the clause $C = p(f(X)) \leftarrow p(X)$. By our or similar techniques one is able

to derive the clause $C' : p(f(f(f(X)))) \leftarrow p(X)$ but this is impossible by the step-by-step method, since one - after having unfolded C against itself obtaining $p(f(f(X))) \leftarrow p(X)$ - has lost C. Aside from being less powerful, we also think that the step-by-step strategy conceptually is much less clean than our approach - a similar view being held in [Tur86].

In the literature, one is typically (contrary to our framework) not allowed to fold against a (direct or indirect) recursive predicate [KK90], [PP91a], [Sek91], [TS84]. This mirrors the view that folding corresponds to abbreviation, a view also held in [Han91].

[TS84] and [KK90] divide the predicates into two classes: the *new* (corresponding to "eureka-definitions") and *old*, where folding is allowed against new predicates only. In the body of new predicates as well as in the body of old predicates, only old predicates can occur. Folding is valid in two cases:

- Starting with the definition of an old predicate, $O \leftarrow O_1 \ldots O_n$, one can do zero or more unfoldings of some of the O_i's and then fold some of these back into a new predicate.

- Starting with the definition of a new predicate, $N \leftarrow O_1 \ldots O_n$, one has to do *at least one* unfolding of some of the O_i's before folding back into a new predicate.[4]

If new predicates are assigned weight 0 and old predicates are assigned weight 1, this translates into our condition 2.3. As we have seen in section 2.5 this condition is (too) weak, since failing branches may convert to loops.

[Sek91] improves on the above, essentially by coming up with condition 2.1 (still when new predicates have been assigned weight 0 and old predicates weight 1). As now not only the success set but also the failure set is preserved, negation can be handled as well.

[Kot85] treats a functional language (where there apparently is no branching), thus his results are not immediately compatible to ours. The situation is that first a number of unfoldings are made, then some laws are applied (not catered for by our framework), then some foldings are made. It is claimed that folding is safe if the number of unfoldings is greater than the number of

[4] Actually, in [TS84] one is allowed to fold even if no unfolding of an O_i is made, provided not all the O_i's disappear by the folding. By assigning new predicates a weight equal the number of goals on the right hand side of their definition, and by assigning old predicates a "very large integer" as weight, this translates into our condition 2.3.

foldings. In some sense, this corresponds to assigning all predicates weight 1 in our framework.

[GS91] allows folding against *existing* clauses (recall clauses are deleted after having been unfolded) only (not allowing a clause to be folded against itself). This greatly limits the applications, since it seems impossible to arrive at recursive definitions of eureka-predicates. On the other hand, it becomes possible to give a relatively simple proof of termination preservation.

In contrast to the authors mentioned so far, [PP91a] impose an order on a sequence of goals, i.e. consider PROLOG's $\mathcal{LR}$ strategy. The crucial condition on folding is that the leftmost atom has been unfolded. Again by assigning the predicates folded against weight 0 and the others 1, the essence of this translates into our condition 2.2. A version of condition 2.2 is also stated in [Han91].

4 Fundamental concepts

4.1 Basic configurations

Assume a finite universe of predicate symbols U.

Definition 4.1 A *goal sequence* (J, H) consists of a totally ordered set J, together with a mapping H which to each $j \in J$ assigns a member of U.

Often we drop J and just write H. $j < j'$ models that $H(j)$ is "to the left" of $H(j')$.

Definition 4.2 A *basic configuration* (over K) is a quadruple (J, H, K, Q) where (J, H) is a goal sequence, K is a set and Q is a mapping which to each $k \in K$ assigns a member of $\mathcal{P}(\prod_{j \in J} \mathcal{D})$ (for simplicity, we assume that all predicates have arity 1).

A basic configuration is *failure* if $Q(k) = \emptyset$ for all $k \in K$; and is *empty* if $J = \emptyset$.

Definition 4.3 Given goal sequence (J, H), we define the canonical basic configuration over (J, H) as follows: $Ca_{(J,H)} = (J, H, \prod_{j \in J} \mathcal{D}, Q)$ where $Q(\vec{d}) = \{\vec{d}\}$.

Specializations

Definition 4.4 Given basic configurations B and B', with $B = (J, H, K, Q)$ and $B' = (J, H, K', Q')$. A *specialization* from B to B'[5] is a mapping s from K to $\mathcal{P}(K')$ such that for all $k \in K$

$$Q(k) = \bigcup_{k' \in s(k)} Q'(k')$$

We say that $B = \mathcal{I}_s(B')$.

Fact 4.5 Given basic configuration $B = (J, H, K, Q)$. Now there exists one and only one specialization s from B to $Ca_{(J,H)}$.

PROOF: s will be a specialization iff

$$Q(k) = \bigcup_{\vec{d} \in s(k)} \{\vec{d}\} = s(k)$$

□

Operators on configurations and specializations

Definition 4.6 If J_1 and J_2 are two ordered sets (ordered by $<_1$ and $<_2$), we define $J = J_1 \& J_2$ (ordered by $<$) by letting J be the disjoint union of J_1 and J_2 by letting $in_1(j) < in_1(j')$ iff $j <_1 j'$ and $in_2(j) < in_2(j')$ iff $j <_2 j'$; and by letting $in_1(j) < in_2(j')$ for all $j \in J_1, j' \in J_2$.

Definition 4.7 Let (J_1, H_1) and (J_2, H_2) be two goal sequences. We define $(J_1, H_1)\&(J_2, H_2)$ $(= (J, H))$ as follows: $J = J_1 \& J_2$; $H(in_1(j)) = H_1(j_1)$ and $H(in_2(j)) = H_2(j)$.

Definition 4.8 Let $B_1 = (J_1, H_1, K_1, Q_1)$ and $B_2 = (J_2, H_2, K_2, Q_2)$ be basic configurations. Then we define $B_1 \& B_2 = (J, H, K, Q)$ as follows:

- $(J, H) = (J_1, H_1)\&(J_2, H_2)$.
- $K = K_1 \times K_2$

[5] B' is "more general" than B.

- $Q(k_1, k_2) = \{\vec{d_1} \times \vec{d_2} \mid \vec{d_1} \in Q_1(k_1), \vec{d_2} \in Q_2(K_2)\}$ where $(\vec{d_1} \times \vec{d_2})(in_1(j)) = \vec{d_1}(j), (\vec{d_1} \times \vec{d_2})(in_2(j)) = \vec{d_2}(j)$

Fact 4.9 $B_1 \& B_2$ is failure iff B_1 is failure or B_2 is failure.

Definition 4.10 Given specializations s_1 from B_1 to B_1' and s_2 from B_2 to B_2'. Let $B_1 = (J_1, H_1, K_1, Q_1), B_2 = (J_2, H_2, K_2, Q_2), B_1' = (J_1, H_1, k_1', Q_1')$ and $B_2' = (J_2, H_2, K_2', Q_2')$. Then define $s = s_1 \& s_2$, a specialization from $B_1 \& B_2$ to $B_1' \& B_2'$, by

$$(s_1 \& s_2)(k_1, k_2) = \{(k_1', k_2') \mid k_1' \in s_1(k_1), k_2' \in s_2(k_2)\}$$

We have to check that this actually *is* a specialization: but with $B_1 \& B_2 = (J, H, K, Q)$ and $B_1' \& B_2' = (J, H, K', Q')$ we have

$$\begin{aligned} Q(k_1, k_2) &= \{\vec{d_1} \times \vec{d_2} \mid \vec{d_1} \in Q_1(k_1), \vec{d_2} \in Q_2(k_2)\} \\ &= \{\vec{d_1} \times \vec{d_2} \mid \exists k_1' \in s_1(k_1), \exists k_2' \in s_2(k_2) : \vec{d_1} \in Q_1'(k_1'), \vec{d_2} \in Q_2'(k_2')\} \\ &= \{\vec{d} \mid \exists k_1' \in s_1(k_1), \exists k_2' \in s_2(k_2) : \vec{d} \in Q'(k_1', k_2')\} \\ &= \{\vec{d} \mid \exists k' \in s(k_1, k_2) : \vec{d} \in Q'(k')\} \\ &= \bigcup_{k' \in s(k_1, k_2)} Q'(k') \end{aligned}$$

Definition 4.11 Given specialization s from B to B', and specialization s' from B' to B'', we define $s \star s'$, a specialization from B to $''$, by (here $B = (J, H, K, Q), B' = (J, H, K', Q'), B'' = (J, H, K'', Q'')$)

$$(s \star s')(k) = \bigcup_{k' \in s(k)} s'(k')$$

We have to check that this actually *is* a specialization:

$$Q(k) = \bigcup_{k' \in s(k)} Q'(k') = \bigcup_{k' \in s(k)} \bigcup_{k'' \in s'(k')} Q''(k'') = \bigcup_{k'' \in (s \star s')(k)} Q''(k'')$$

Definition 4.12 Given basic configuration $B = (J, H, K, Q)$ we define Id_B, a specialization from B to B, by

$$Id_B(k) = \{k\}$$

Algebraic identities

When writing "=", we always mean "module isomorphism". It should be obvious what it means for two basic configurations to be isomorphic.

Fact 4.13 By letting the objects be basic configurations and by letting the morphisms be specializations, we obtain a category. That is, $\star$ is associative and Id_B is a neutral element for all B.

Moreover, & is a functor in this category - i.e. $Id_{B_1}\&Id_{B_2} = Id_{B_1\&B_2}$, and $(s_1 \star s_1')\&(s_2 \star s_2') = (s_1\&s_2) \star (s_1'\&s_2')$.

Finally, & is associative and $Ca_{H_1}\&Ca_{H_2} = Ca_{H_1\&H_2}$.

PROOF: The only nontrivial part is the relation between & and $\star$:

$$
\begin{array}{ll}
 & (k_1'', k_2'') \in ((s_1 \star s_1')\&(s_2 \star s_2'))(k_1, k_2) \\
\Leftrightarrow & k_1'' \in (s_1 \star s_1')(k_1), k_2'' \in (s_2 \star s_2')(k_2) \\
\Leftrightarrow & \exists k_1', k_2' : k_1' \in s_1(k_1), k_1'' \in s_1'(k_1'), k_2' \in s_2(k_2), k_2'' \in s_2'(k_2') \\
\Leftrightarrow & \exists (k_1', k_2') : (k_1', k_2') \in (s_1\&s_2)(k_1, k_2), (k_1'', k_2'') \in (s_1'\&s_2')(k_1', k_2') \\
\Leftrightarrow & (k_1'', k_2'' \in ((s_1\&s_2) \star (s_1'\&s_2'))(k_1, k_2)
\end{array}
$$

□

4.2 U-mirrors

Given a function OI which for each $G \in U$ returns a non-empty and finite index set $OI(G)$.

Given a function AI which for each $G \in U$ and each $i \in OI(G)$ returns a finite index set $AI(G)$, equipped with a total order $<$.

Given a function P which for each $G \in U, i \in OI(G)$ and $j \in AI(G, i)$ returns $P(G, i, j) \in U$.

Given a function W which for each $G \in U, i \in OI(G)$ and $j \in AI(G, i)$ returns $W(G, i, j)$, a non-negative integer.

Returning to example 1.1, there e.g. $OI(f) = \{1, 2\}$ (or any two-element set); $OI(g) = \{1\}$ (or any one-element set), $AI(f, 1) = \emptyset$, $AI(g) = \{1, 2\}$ with $1 < 2$, $P(g, 1, 1) = P(g, 1, 2) = f$.

U-forests

Definition 4.14 A *U-forest* from goal sequence (J, H) to goal sequence (J', H') is a J-indexed family of trees where

1. Nodes are labeled by a *goal label* G, $G \in U$. Some nodes are also equipped with an *or-direction* label i with $i \in OI(G)$. All nodes not being leaves, and possibly also some leaves, have an or-direction label.

2. Arcs are labeled by an *and-direction* label j and a *weight label* w. Distinct arcs going from the same node are labeled by distinct and-direction labels.

3. For all $j \in J$, the root of the j'th tree has goal label $H(j)$.

4. Let N be a node which has an or-direction label i, and which has goal label G. Then j will be the and-direction label of an arc going from N iff $j \in AI(G, i)$. The arcs from N inherit the ordering of $AI(G, i)$.

5. Let a be an arc from a node N, with goal label G, to N'. Then G will contain an or-direction label i. Further, with j the and-direction label and w the weight label of a, the goal label of N' is $P(G, i, j)$ and $w = W(G, i, j)$.

6. There is a total ordering among the leaves - and thus also among the paths, where a path starts at a root and ends at a leaf - determined in the "natural way" by the ordering on J and the ordering on the arcs leaving each node.

7. The sequence of leaves *not having an or-direction label*, together with their goal labels, is isomorphic to (J', H').

- A path ending in a leaf not having an or-direction label is termed *working*.

- A U-forest is *working* iff all paths are working.

- The weight of a path p, $W(p)$, is the sum of the weight labels encountered when walking along p.

- The weight of a U-forest f, $W(f)$, is the sum of the weight labels in f.

To be more formal, a working path p in a U-forest from (J, H) to (J', H') is a sequence of the form

$$jG_0(i_1, j_1, w_1)G_1 \cdots (i_n, j_n, w_n)j'G_n(n \geq 0)$$

where $j \in J, G_0 = H(j), j' \in J', G_n = H'(j'), G_k = P(G_{k-1}, i_k, j_k)$ and $w_k = W(G_{k-1}i_k, j_k)$ for $k = 1 \ldots n$.

A non-working path in a U-forest from (J, H) to (J', H') is a sequence of the form

$$jG_0(i_1, j_1, w_1G_1 \cdots (i_n, j_n, w_n)G_ni(n \geq 0)$$

where $j \in J, G_0 = H(j), AI(G_n, i) = \emptyset, G_k = P(G_{k-1}, i_k, j_k)$ and. $w_k = W(G_{k-1}, i_k, j_k)$ for $k = 1 \ldots n$.

In both cases $W(p) = \sum_{k=1}^{n} w_k$.

Definition 4.15 If p is a working path in a U-forest from (J, H) to (J', H') of form $jGqj'G'$, and p' is a path in a U-forest from (J', H') to (J'', H'') of form $j'G'q'$, then we define $p \star p' = jGqj'G'q'$.

Definition 4.16 Given U-forest f from (J, H) to (J', H') and U-forest f' from (J', H') to (J'', H''). We can now define $f \star f'$, a U-forest from (J, H) to (J'', H''), by "gluing" the two forests together in the obvious way.

Observation 4.17 Given a path p'' in $f \star f'$. Two possibilities:

- p'' is a non-working path in f. Then p'' will be non-working in $f \star f'$ as well.
- There exists working path p in f and path p' in f' such that $p'' = p \star p'$. p'' will be working iff p' is. These p and p' are unique.

Conversely, if p' is a path in f' there exists exactly one (working) path p in f such that $p \star p'$ forms a path in $f \star f'$. If p is a working path in f, there exists at least one path p' in f' such that $p \star p'$ forms a path in $f \star f'$.

Definition 4.18 Given goal sequence (J, H). $Id_{(J,H)}$ is now defined as the U-forest from (J, H) to (J, H), where all paths ire of the form jG.

Definition 4.19 Given U-forest f_1 from H_1 to H_1' and U-forest f_2 from H_2 to H_2'. Now define $f_1 \& f_2$, a U-forest from $H_1 \& H_2$ to $H_1' \& H_2'$, in the obvious way - i.e. the paths in $f_1 \& f_2$ will be the "disjoint union" of the paths in f_1 and the paths in f_2.

Fact 4.20 By letting the objects be goal sequences and the morphisms be U-forests, one gets a category (with $\star$ as composition and $Id_{\_}$ as identities). & is a functor in this category, and & is associative.

Definition 4.21 Given a U-forest f_1 from H to H_1, and a U-forest f_2 from H to H_2. We say that (f_1', f_2', H') is a *completion* of (f_1, f_2) if f_1' is a U-forest from H_1 to H', f_2' is a U-forest from H_2 to H', and $f_1 \star f_1' = f_2 \star f_2'$.

Pushouts

Observation 4.22 Given a U-forest f_1 from H to H_1, and a U-forest f_2 from H to H_2. Suppose (f_1, f_2) has a completion. Then there exists a completion (f_1', f_2', H') such that for all completions (f_1'', f_2'', H'') there exists a U-forest f from H' to H'' with $f_1' \star f = f_1''$, $f_2' \star f = f_2''$. Consequently, this completion is unique - we term (f_1', f_2', H') the *pushout* of (f_1, f_2).

Moreover, if f_1 is working then f_2' will be working.

Fact 4.23 Taking pushouts is commutative in the following sense: given (f_1, f_1') and (f_2, f_2'), such that f_1' and f_2 are U-forests to the same goal sequence. Suppose (f_1, f_1') has pushout (f_3, f_3'), and suppose $(f_2 \star f_3', f_2')$ has pushout (f_4, f_4'). Now the situation is as in the left part of figure 4.

Then (f_2, f_2') will have a pushout, to be written (f_5, f_5'); and also $(f_1, f_1' \star f_5)$ will have a pushout, to be written (f_6, f_6') - as depicted in the right part of figure 4. Moreover,

$f_3 \star f_4 = f_6$, $f_5' \star f_6' = f_4'$

PROOF: Standard categorical reasoning is applied: As $f_2 \star f_3' \star f_4 = f_2' \star f_4'$ we see that (f_2, f_2') in fact has a pushout (f_5, f_5'), and there exists unique f such that $f_5 \star f = f_3' \star f_4$, $f_5' \star f = f_4'$. Now

$f_1 \star f_3 \star f_4 = f_1' \star f_3' \star f_4 = f_1' \star f_5 \star f$

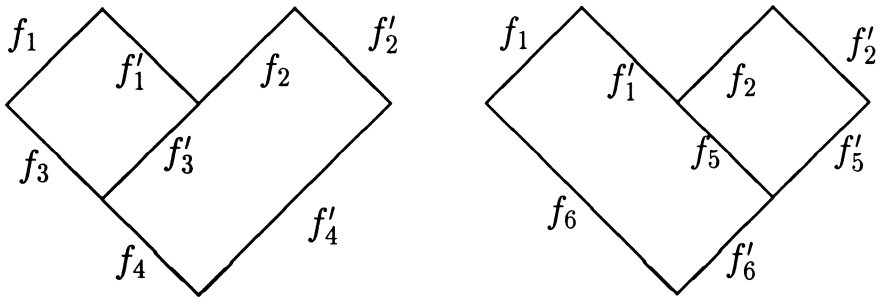

Figure 4: Pushout commutes

which shows that $(f_1, f_1' \star f_5)$ has a pushout (f_6, f_6'), and there exists f' such that $f_6 \star f' = f_3 \star f_4$, $f_6' \star f' = f$.

Also we have $f_5' \star f_6' \star f' = f_5' \star f = f_4'$. For reasons of symmetry (f_6 plays the same role as f_4', f_5' plays the same role as f_3 and f_6' plays the same role as f_4) we can find f'' such that $f_4' \star f'' = f_5' \star f_6'$ $f_3 \star f_4 \star f'' = f_6$.

From $f_6 = f_3 \star f_4 \star f'' = f_6 \star f' \star f''$ we conclude $f' = Id_{\_}$ - hence the claim.

□

U-mirrors

Definition 4.24 A *U-mirror* m from goal sequence H to goal sequence H' is a triple (f, f', H'')[6] where f is a U-forest from H to H''; and where f' is a U-forest from H' to H''. We require f' to be working (as we do not allow folding using a rule of form $g \leftarrow \Box$).

- One can in the natural way define the paths of a U-mirror. A path p'' is *either* a non-working path in f *or* of the form (p, p') where p is a working path in f of form $qj''G''$ and p' a (working) path in f' of form $q'j''G''$ - we say p and p' are *connected*. Paths of the former kind are termed non-working; paths of the latter kind are termed working.

- We say that m is working iff all paths are working - i.e. iff f is working.

- The weight of a non-working path p, $W(p)$, is the weight of p in f; the

[6]We often just write (f, f') and omit H''.

weight of a working path (p, p'), $W(p, p')$, is the difference between the weight of p in f and the weight of p' in f'.

- The weight of the U-mirror, $W(m)$, is $W(f) - W(f')$.

Observe that given working path p in f there exists exactly one path p' in f' connected to p; and given (working) path p' in f' there exists exactly one path p in f connected to p'. Also, there will exist a working path in m iff H' is not empty.

We will not distinguish between a U-forest f from H to H' and the U-mirror $(f, Id_{H'}, H')$.

Definition 4.25 Let $m = (f, f', H'')$ be a U-mirror from H to H', and suppose m is working (i.e. f is working). Then we can define $\mathcal{R}(m)$, a (working) U-mirror from H' to H, by $\mathcal{R}(m) = (f', f, H'')$.

Definition 4.26 Let $m_1 = (f_1, f_1', H_1'')$ be a U-mirror from H_1 to H_1', and let $m_2 = (f_2, f_2', H_2'')$ be a U-mirror from H_2 to H_2'. Now we can define $m_1 \& m_2$, a U-mirror from $H_1 \& H_2$ to $H_1' \& H_2'$, by letting

$$m_1 \& m_2 = (f_1 \& f_2, f_1' \& f_2', H_1'' \& H_2'')$$

Definition 4.27 Given goal sequence H. Then we can define Id_H as (Id_H, Id_H, H).

Definition 4.28 Given a U-mirror $m_1 = (f_{11}, f_{21}, H_1')$ from H_1 to H_2, and given a U-mirror $m_2 = (f_{22}, f_{32}, H_2')$ from H_2 to H_3.

First suppose the pushout of (f_{21}, f_{22}) exists. Let it be (f_1', f_2', H''). Then

$$m_1 \star m_2 = (f_{11} \star f_1', f_{32} \star f_2', H'')$$

(This is a well-defined U-mirror: m_1 is a U-mirror, so f_{21} is working. By observation 4.22, f_2' is working. As m_2 is a U-mirror, f_{32} is working. Hence, $f_{32} \star f_2'$ is working.)

If the pushout does not exist, $m_1 \star m_2 = \perp$ (i.e. undefined).

Lemma 4.29 Suppose $m_1 \star m_2 \neq \perp$. Suppose (p, p') is a path in $m_1 \star m_2$. Then there exists paths (p_1, p_1') in m_1, (p_2, p_2') in m_2 and paths p'', p''' such that $p = p_1 \star p''$, $p' = p_2' \star p'''$, and $p_1' \star p'' = p_2 \star p'''$.

PROOF: Let $m_1 = (f_1, f_1'(J_1', H_1'))$ be a U-mirror from (J_1, H_1) to (J_2, H_2), and let $m_2 = (f_2, f_2'(J_2', H_2'))$ b e a U-mirror from (J_2, H_2) to (J_3, H_3). Let $(f'', f''', (J'', H''))$ be the pushout of (f_1', f_2). Then

$$m_1 \star m_2 = (f_1 \star f'', f_2' \star f''')$$

We have that p is a working path in $f_1 \star f''$, and p' is a (working) path in $f_2' \star f'''$. There thus exist working paths p_1 in f_1, p'' in f'', p_2' in f_2' and p''' in f''' such that $p = p_1 \star p''$, $p' = p_2' \star p'''$. Now let p_2 in f_2 and p_1' in f_1' be the unique paths connected to p_2' and p_1 respectively. As there exists $j_1' \in J_1'$ such that p_1 and p_1' both end with $j_1' H_1'(j_1')$, $p_1' \star p''$ will be a path in $f_1' \star f''$. Similarly, $p_2 \star p'''$ will be a path in $f_2 \star f'''$. Since $f_1' \star f'' = f_2 \star f'''$, and since $p_2 \star p'''$ ends with the same element in J'' as p''' does as p' does as p does as p'' does as $p_1' \star p''$ does, we conclude that actually $p_1' \star p'' = p_2 \star p'''$. □

Fact 4.30

1. Operating on U-mirrors, $\star$ is associative[7] with $Id_{\_}$ as neutral element.
2. $\&$ is a functor; i.e. $(m_1 \star m_1')\&(m_2 \star m_2') = (m_1 \& m_2) \star (m_1' \& m_2')$; and $Id_H \& Id_{H'} = Id_{H \& H'}$. Also, $\&$ is associative.
3. The property of being working is closed under all operations in question.
4. Given working f from H to H'. Considered as a U-mirror, $\mathcal{R}(f) \star f = Id_{H'}$.

PROOF: First for the last claim: we must show that $(Id_{H'}, f) \star (f, Id_{H'}) = Id_{H'}$. Now the pushout of (f, f) is $(Id_{H'}, Id_{H'})$, hence the claim follows.

That $Id_{\_}$ is neutral element can be seen as follows: Let $m = (f, f', H'')$ be a U-mirror from H to H'. The pushout of $(f', Id_{H'})$ clearly is $(Id_{H''}, f', H'')$. So

$$m \star Id_{H'} = (f \star Id_{H''}, Id_{H'} \star f', H'') = (f, f', H'') = m$$

That $Id_{\_}$ is left neutral element is seen similarly.

[7]Equality, in the presence of $\perp$, means that either both sides are $\perp$ or both sides are $\neq \perp$ and equal.

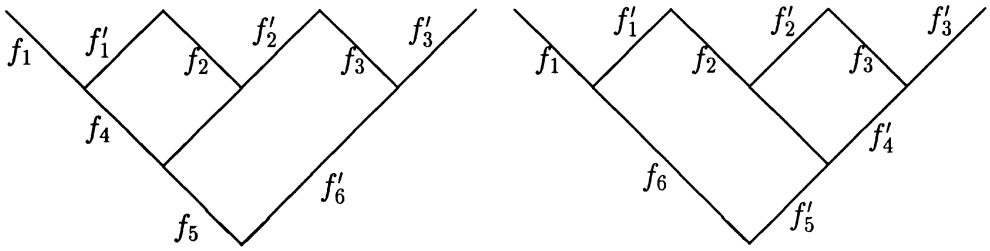

Figure 5: Associativity of $\star$ on U-mirrors

Next for the associativity: let $m_1 = (f_1, f_1')$, $m_2 = (f_2, f_2')$, $m_3 = (f_3, f_3')$. Consider figure 5, where the relevant pushouts have been drawn (assuming they exist). Then

$(m_1 \star m_2) \star m_3 = (f_1 \star f_4 \star f_5, f_3' \star f_6')$
$m_1 \star (m_2 \star m_3) = (f_1 \star f_6, f_3' \star f_4' \star f_5')$

Now we can apply fact 4.23 to see that $(m_1 \star m_2) \star m_3$ will be defined iff $m_1 \star (m_2 \star m_3)$ is, and then they will be equal.

The remaining claims are trivial. □

4.3 Properties of U-mirrors

Definition 4.31 Given n, we say that a U-mirror m satisfies $\mathcal{F}(n)$ iff for all working paths p we have $W(p) \geq n$; and there exists a working path.

Lemma 4.32 If m_1 satisfies $\mathcal{F}(n_1)$ and m_2 satisfies $\mathcal{F}(n_2)$ then $m_1 \star m_2$ - if defined - satisfies $\mathcal{F}(n_1 + n_2)$.

PROOF: Let $m_1 = (f_1, f_1', H_1')$ from H_1 to H_2; and let $m_2 = (f_2, f_2', H_2')$ from H_2 to H_3.

Since there exists a working path in m_2, H_3 is not empty, and hence there exists a working path in $m_1 \star m_2$.

Let (p, p') be a working path in $m_1 \star m_2$. By lemma 4.29, there exists (p_1, p_1') in m_1, (p_2, p_2') in m_2 and paths p'', p''' such that $p = p_1 \star p''$, $p' = p_2' \star p'''$, and $p_1' \star p'' = p_2 \star p'''$. Now, by applying the assumption, we have

$$W(p, p') \;=\; W(p) - W(p') = W(p_1) + W(p'') - W(p_2') - W(p''')$$

$$\begin{aligned} &\geq W(p_1') + n_1 + W(p'') - W(p_2) + n_2 - W(p''') \\ &= W(p_1' \star p'') - W(p_2 \star p''') + n_1 + n_2 \\ &= n_1 + n_2 \end{aligned}$$

□

Definition 4.33 Given n, we say that a U-mirror m satisfies $\mathcal{A}(n)$ iff $W(m) \geq n$.

Lemma 4.34 If m_1 satisfies $\mathcal{A}(n_1)$, and m_2 satisfies $\mathcal{A}(n_2)$, then $m_1 \star m_2$ - if defined - satisfies $\mathcal{A}(n_1) + m_2)$.

PROOF: Let $m_1 = (f_1, f_1', H_1')$ from H_1 to H_2; and let $m_2 = (f_2, f_2', H_2')$ from H_2 to H_3. Let (f'', f''') be the pushout of (f_1', f_2). Now

$$\begin{aligned} W(m_1 \star m_2) &= W(f_1 \star f'') - W(f_2' \star f''') \\ &= W(f_1) + W(f'') - W(f_2') - W(f''') \\ &\geq W(f_1') + n_1 + W(f'') - W(f_2) + n_2 - W(f''') \\ &= W(f_1' \star f'') - W(f_2 \star f''') + n_1 + n_2 \\ &= n_1 + n_2 \end{aligned}$$

□

Definition 4.35 Given n, we say that a U-mirror m satisfies $\mathcal{L}(n)$ iff there exists a path in m, and the leftmost path has weight $\geq n$.

Definition 4.36 Given a U-mirror m, we let

- $E(m)$ denote the number of non-working paths to the left, i.e. $E(m) \geq n$ iff the n leftmost paths in m are non-working.
- $L(m)$ denote the weight of the leftmost working path (if no such exists, 0).

Then we can define an ordering $\prec$ on the set of U-mirrors by letting $m_1 \prec m_2$ iff $E(m_1) < E(m_2)$ or $E(m_1) = E(m_2)$ and $L(m_1) < L(m_2)$.

Lemma 4.37 Suppose m_2 satisfies $\mathcal{L}(1)$, and $m = m_1 \star m_2$ is defined. Then $m_1 \prec m$.

PROOF: First some notation: we say that a path is *left-directed* if the following holds for all its arcs: let the arc have and-direction label j and go from a node with goal label G and or-direction label i. Then j is the least element in $AI(G, i)$.

Let $m_1 = (f_1, f_1')$, $m_2 = (f_2, f_2')$. Let (f'', f''') be the pushout of (f_1', f_2). Now $m = (f_1 \star f'', f_2' \star f''')$. The $E(m_1)$ leftmost paths of m_1 will be non-working; so the $E(m_1)$ leftmost paths of f_1 will be non-working; so the $E(m_1)$ leftmost paths of $f_1 \star f''$ will be non-working; so the $E(m_1)$ leftmost paths of m will be non-working. This shows that $E(m_1) \leq E(m)$.

Now consider the leftmost path in m_2. Two possibilities:

- It is of form p_2 with p_2 non-working in f_2. As $f_2 \star f''' = f_1' \star f''$, there will exist p_1' in f_1' and left-directed p'' in f'' such that $p_2 = p_1' \star p''$, and such that p_1' is the leftmost path in $f' - 1$. Let p_1' be connected to p_1; p_1 will be the leftmost working path in f_1. Now $p_1 \star p''$ will be non-working in $f_1 \star f''$ and hence also in m. This shows that $E(m_1) < E(m)$ and hence $m_1 \prec m$.

- It is of form (p_2, p_2') with p_2 working in f_2. As $f_2 \star f''' = f_1' \star f''$, there will exist p_1' in f_1', left-directed p'' in f'' and left-directed p''' in f''' such that $p_2 \star p''' = p_1' \star p''$, and such that p_1' is the leftmost path in f_1'. Let p_1' be connected to p_1; p_1 will be the leftmost working path in f_1. Now $(p_1 \star p'', p_2' \star p''')$ will belong to m, and be the leftmost working path in m. Moreover,

$$\begin{aligned} L(m) &= W(p_1 \star p'') - W(p_2' \star p''') \\ &= W(p_1) + W(p'') - W(p_2') - W(p''') \\ &> W(p_1) + W(p'') - W(p_2) - W(p''') \\ &= W(p_1) + W(p'') - W(p_2 \star p''') \\ &= W(p_1) + W(p'') - W(p_1' \star p'') \\ &= W(p_1) - W(p_1') \\ &= L(m_1) \end{aligned}$$

  This shows $m_1 \prec m$.

□

4.4 Transitions

Definition 4.38 A *transition* t from basic configuration $B = (J, H, K, Q)$ to $B' = (J', H', K', Q')$ (notice the K-sets are identical) is a set of U-mirrors from (J, H) to (J', H') which is either empty or a singleton.[8] We will demand that (Q, Q') is a *non-increasing* pair, i.e. that for all $k \in K$, $Q(k) = \emptyset \Rightarrow Q'(k) = \emptyset$ (so it will not be possible to make a transition from a failure basic configuration into a non-failure, cf. the discussion in section 2.4, (1)).

We say that t is *stable* iff also (Q', Q) is a non-increasing pair, i.e. $Q(k) = \emptyset \Leftrightarrow Q'(k) = \emptyset$.

Definition 4.39 Let t be a transition from B to B', and let t' be a transition from B' to B''. Now define $t \star t'$, a transition from B to B'', by

$$m'' \in t \star t' \Leftrightarrow \exists m \in t, m' \in t' : m'' = m \star m'$$

Clearly (B, B'') is a non-increasing pair, and $t \star t'$ will be stable if t and t' are stable.

Definition 4.40 Let t_1 be a transition from B_1 to B_1', and let t_2 be a transition from B_2 to B_2'. Now define $t_1 \& t_2$, a transition from $B_1 \& B_2$ to $B_1' \& B_2'$, by

$$m \in t_1 \& t_2 \Leftrightarrow \exists m_1 \in t_1, m_2 \in t_2 : m = m_1 \& m_2$$

Clearly non-increasingness will be preserved, and $t_1 \& t_2$ will be stable if t_1 and t_2 are stable.

Definition 4.41 Given $B = (J, H, K, Q)$, define Id_B, a transition from B to B, by letting Id_B be the singleton set containing $Id_{J,H}$. This is clearly stable.

Definition 4.42 Given a transition t from B to B'. Let s be a specialization from $\mathcal{I}_s(B)$ to B. Now we define $\mathcal{I}_s(t)$, a transition from $\mathcal{I}_s(B)$ to $\mathcal{I}_s(B')$, by letting m be in $\mathcal{I}_s(t)$ iff m is in t. Clearly $(\mathcal{I}_s(B), \mathcal{I}_s(B'))$ is a non-increasing pair, and $\mathcal{I}_s(t)$ will be stable if t is.

Definition 4.43 Given a transition t from B to B'. Suppose t is stable, and suppose it for all $m \in t$ holds that m is working. Then we say that t is

[8] We will often identify t with its element.

reversible and we can define $\mathcal{R}(t)$, a transition from B' to B, by stipulating that m is in $\mathcal{R}(t)$ iff $\mathcal{R}(m)$ is in t. Clearly $\mathcal{R}(t)$ is reversible.

Fact 4.44 By letting the objects be basic configurations and by letting the morphisms be transitions, we obtain a category. & is a functor in this category.

5 Two level transition system

5.1 The level 0 rules

Assume that for all $G \in U$ and $i \in OI(G)$ there exists a transition $t(G,i)$ from Ca_G to $c(G,i)$, where $c(G,i)$ takes the form
$(AI(G,i), \lambda j.P(G,i,j), \mathcal{D})$

and where $t(G,i)$ contains the U-mirror (U-forest) determined by the paths

$\{G(i,j,\_)G' \mid j \in AI(G,i)\}$

where $G' = P(G,i,j)$. However, if $AI(G,i) = \emptyset$ the U-mirror will be determined by the singleton path Gi. Clearly, this in non-increasing. Now

$$\mathcal{RU}_0 = \{t(G,i) \mid G \in U, i \in OI(G)\} \tag{3}$$

In example 1.1, e.g. $c(g,1) = ([f, fj], \mathcal{D} \times \mathcal{D}, Q)$ with

$Q(d_1,d_2) = \{\{((d_1,d),(d,d_2))\} \mid d \in \mathcal{D}\}$

5.2 Unfolding at level 1

We now define what it means for a transition t from B to B' to be a level 1 unfolding step, to be written $1 \vdash_u^S t : B \to B'$[9] (here H_1 and H_2 are goal sequences, and s is a specialization):

[9]When we are not interested in the configurations, we may simply write $1 \vdash_u^S t$.

$$\frac{t(G,i) \in \mathcal{RU}_0}{1 \vdash_u^S \mathcal{I}_s(Id_{Ca_{H_1}} \& t(G,i) \& Id_{Ca_{H_2}})} \tag{4}$$

Next we define what it means for a transition t from B to B' to be a level 1 unfolding, to be written $1 \vdash_u T : B \to B'$

$$\frac{1 \vdash_u^S t : B \to B'}{1 \vdash_u t : B \to B'} \tag{5}$$

$$\frac{}{1 \vdash_u Id_B : B \to B} \tag{6}$$

$$\frac{1 \vdash_u t : B \to B', 1 \vdash_u t' : B' \to B''}{1 \vdash_u t \star t' : B \to B''} \tag{7}$$

If $1 \vdash_u t$, we can write $t = t_1 \star \cdots t_n$ $(n \geq 0)$ with $1 \vdash_u^S t_i$ for $i \in \{1 \ldots n\}$. The least n which can be used is called the *length* of t.

Fact 5.1

1. If $1 \vdash_u^S t$, then $1 \vdash_u^S t \& Id_B$ and $1 \vdash_u^S Id_B \& t$ for basic configuration B.
2. If $1 \vdash_u^S t$, then $1 \vdash_u^S \mathcal{I}_s(t)$ for specialization s.
3. If $1 \vdash_u t$, then $1 \vdash_u t \& Id_B$ and $1 \vdash_u Id_B \& t$ for basic configuration B.
4. If $1 \vdash_u t_1$ and $1 \vdash_u t_2$, then $1 \vdash_u t_1 \& t_2$.
5. If $1 \vdash_u t$, then $1 \vdash_u \mathcal{I}_s(t)$ for specialization s.
6. If $1 \vdash_u t$, then t is a singleton - i.e. of form $\{m\}$.

PROOF: For (1), notice that fact 4.5 tells us that we can write $B = \mathcal{I}_{s'}(Ca_H)$. Then

$$\begin{aligned} & \mathcal{I}_s(Id_{Ca_{H_1}} \& t(G,i) \& Id_{Ca_{H_2}}) \& Id_B \\ = \; & \mathcal{I}_s(Id_{Ca_{H_1}} \& t(G,i) \& Id_{Ca_{H_2}}) \& \mathcal{I}_{s'}(Id_{Ca_H}) \\ = \; & \mathcal{I}_{s \& s'}(Id_{Ca_{H_1}} \& t(G,i) \& Id_{Ca_{H_2}} \& Id_{Ca_H}) \\ = \; & \mathcal{I}_{s \& s'}(Id_{Ca_{H_1}} \& t(G,i) \& Id_{Ca_{H_2 \& H}}) \end{aligned}$$

(2) is trivial. (3) follows by induction in the derivation tree for $1 \vdash_u t$: the case where rule (5) has b een applied follows from what has just been

shown; the case where rule (6) has been applied is trivial; and the case where rule (7) has been applied follows from the calculation (cf. fact 4.44)

$(t\&Id_B) \star (t'\&Id_B) = (t \star t')\&(Id_B \star Id_B) = (t \star t')\&Id_B$

(4) follows from what has been just shown and fact 4.44:

$t_1\&t_2 = (t_1 \star Id\_)\&(Id\_ \star t_2) = (t_1\&Id\_) \star (Id\_\&t_2)$

(5) is a trivial induction in the derivation tree. (6) follows from the fact that all U-mirrors involved in fact are U-forests, thus $\star$ can never be undefined.

□

The diamond lemma

Lemma 5.2 Suppose $1 \vdash_u^S t_1 : B \to B_1$ and $1 \vdash_u^S t_2 : B \to B_2$. Suppose t_1 and t_2 have a completion (viewed as U-forests). Then there exists B', transition t_1' from B_1 to B_1' and transition t_2 from B_2 to B_2' such that (t_1', t_2') (viewed as a U-forest) is the pushout of (t_1, t_2). Moreover, one of two holds:

- $B_1 = B_2 = B', t_1 = t_2, t_1' = t_2' = Id_{B'}$.
- $1 \vdash_u^S t_1' : B_1 \to B'$ and $1 \vdash_u^S t_2' : B_2 \to B'$.

Proof: We can assume that the transitions involved are of the following form:

$$\begin{aligned} t_1 &= \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}}\&t(G_1, i_1)\&Id_{Ca_{H_{12}}}) \\ t_2 &= \mathcal{I}_{s_2}(Id_{Ca_{H_{21}}}\&t(G_2, i_2)\&Id_{Ca_{H_{22}}}) \end{aligned}$$

As t_1 and t_2 both are transitions from B, we get

$\mathcal{I}_{s_1}(Ca_{H_{11}}\&G_1\&H_{12}) = B = \mathcal{I}_{s_2}(Ca_{H_{21}}\&G_2\&H_{22})$

From this we infer $H_{11}\&G_1\&H_{12} = H_{21}\&G_2\&H_{22}$ and (by fact 45) that $s_1 = s_2$. Now two possibilities:

- $H_{11} = H_{21}$. Then $G_1 = G_2$, and $H_{12} = H_{22}$. As (t_1, t_2) has a completion, $i_1 = i_2$. This shows $t_1 = t_2$ and $B_1 = B_2$. Thus we can choose $t'_1 = t'_2 = Id_{B_1}$, and clearly (t'_1, t'_2) is the pushout.

- $H_{11} \neq H_{21}$. We can wlog. assume that H_{11} is shorter than H_{21}. Then there exists H such that $H_{21} = H\&G_2\&H_{22}$, $H_{21} = H_{11}\&G_1\&H$. Now define

$$\begin{aligned} t'_1 &= \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}}\&Id_{c(G_1,i_1)}\&Id_{Ca_H}\&t(G_2,i_2)\&Id_{Ca_{H_{22}}}) \\ t_2 &= \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}}\&t(G_1,i_1)\&Id_{Ca_H}\&Id_{c(G_2,i_2},\&Id_{Ca_{H_{22}}}) \end{aligned}$$

We can easily calculate

$$\begin{aligned} t_1 \star t'_1 &= \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}}\&t(G_1,i_1)\&Id_{Ca_H}\&t(G_2,i_2)\&Id_{Ca_{H_{22}}} \\ &= t_2 \star t'_2 \end{aligned}$$

To show that $1 \vdash_u^S t'_1$, notice that we can write $c(G_1, i_1)$ on the form $\mathcal{I}_{s'}(Ca_{H'})$ for some s', H' - similarly we have $1 \vdash_u^S t'_2$

Again, viewed as U-forests (t'_1, t'_2) is clearly the pushout.

□

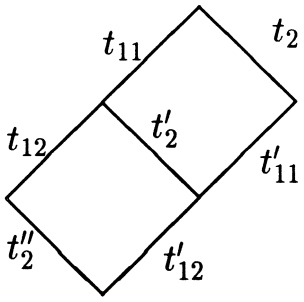

Figure 6: The Church-Rosser property

The Church-Rosser lemma

Lemma 5.3 Suppose $1 \vdash_u t_1 : B \to B_1$ and $1 \vdash_u t_2 : B \to B_2$. Suppose t_1 and t_2 have a completion, viewed as U-forests. Then there exists B' and transitions t'_1, t'_2 such that $1 \vdash_u t'_1 : B_1 \to B'$, $1 \vdash_u t'_2 : B_2 \to B'$, and such that (t'_1, t'_2) - viewed as U-forests - is the pushout of (t_1, t_2).

Moreover, the length of t_1' is less than or equal the length of t_2, and the length of t_2' is less than or equal the length of t_1.

PROOF: Induction in the length of t_1 plus the lenght of t_2. If t_1 or t_2 is the identity the claim is clear. If $1 \vdash_u^S t_1$ and $1 \vdash_u^S t_2$, the claim follows from lemma 5.2.

Otherwise, we can wlog. assume that $t_1 = t_{11} \star t_{12}$ with $1 \vdash_u t_{11}$, $1 \vdash_u t_{12}$. The situation is as depicted in figure 6. By induction, there exists t_2', t_{11}' such that $1 \vdash_u t_{11}'$, $1 \vdash_u t_2'$ and such that (t_2', t_{11}') is the pushout of (t_{11}, t_2). Moreover, the length of t_2' is less than or equal the length of t_2 - this enables us to use the induction hypothesis once more and arrive at t_2'', t_{12}' such that $1 \vdash_u t_{12}'$, $1 \vdash_u t_2''$.

Now we can choose $(t_2'', t_{11}' \star t_{12}')$ as the desired transition pair. That it actually is the pushout of (t_1, t_2) follows from fact 4.23. □

Lemma 5.3 shows that given a basic configuration $B = (J, H, K, Q)$ and an U-forest f from (J, H), it makes sense to define $\mathcal{U}_f(B)$ as the basic configuration B' such that $1 \vdash_u f : B \to B'$.

5.3 Evaluation strategies and Looping at level 1

Definition 5.4 We say that B loops at level 1 *by some strategy* if for all $n \geq 1$ third exists a t_n and a B_n not failure such that $1 \vdash_u^S t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

Lemma 5.5 The following are sufficient conditions for B to loop at level 1 by some strategy:

1. For all $N \geq 0$, there for all n with $1 \leq n \leq N$ exists t_n and B_n not failure such that $1 \vdash_u^S t_n : B_{n-1} \to B_n$ (agin, $B_0 = B$).

2. For all $n \geq 0$, there exists B_n not failure and t_n such that $1 \vdash_u t_n : B \to B_n$, where t_n viewed as a U-forest has n nodes with an or-direction label.

3. For all $n \geq 0$, there exists B_n not failure and t_n such that $1 \vdash_u t_n : B \to B_n$, where t_n viewed as a U-forest has weight $\geq n$ - i.e. satisfies $\mathcal{A}(n)$.

PROOF: That (1) implies that B loops at level 1 by some strategy is a consequence of Königs lemma (as there from a given B is a finite number of

level 1 unfolding steps, since $OI(G)$ is finite). That (2) implies (1) is obvious. That (3) implies (2) is a consequence of the weights being upper bounded (as U, $OI(G)$ and $AI(G,i)$ are finite sets). □

Fair strategy

Definition 5.6 We say that t is a *fair level 1 unfolding step* if t is of form $\mathcal{I}_s(r_1\& \ldots \& r_k)$, $k \geq 1$, where $r_1 \ldots r_k$ are level 0 rules.

For a fair level 1 unfolding step t, we clearly have $1 \vdash_u t$.

Definition 5.7 We say that B loops at level 1 *by a fair strategy* if for all $n \geq 1$ there exists a fair level 1 unfolding step t_n and a B_n not failure such $1 \vdash_u t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

Similar to lemma 5.5, we may prove

Lemma 5.8 The following are sufficient conditions for B to loop at level 1 by a fair strategy:

1. For all $N \geq 0$, there for all n with $1 \leq n \leq N$ exists a fair level 1 unfolding step t_n and B_n not failure such that $1 \vdash_u t_n : B_{n-1} \to B_n$ (again, $B_0 = B$).

2. For all $n \geq 0$, there exists B_n not failure and t_n such that $1 \vdash_u t_n : B \to B_n$, where each working path in t_n (viewed as a U-forest) has length $\geq n$ and there exists a working path.

3. For all $n \geq 0$, there exists B_n not failure and t_n such that $1 \vdash_u t_n : B \to B_n$, where t_n satisfies $\mathcal{F}(n)$.

$\mathcal{LR}$ strategy

Definition 5.9 We say that t is a $\mathcal{LR}$ *level 1 unfolding step* if t is of form $\mathcal{I}_s(r\&Id\_)$ with r a level 0 rule.

We say that t is a $\mathcal{LR}$ *level 1 unfolding* if t is of form $t_1 \star \ldots \star t_n$ with each t_i being a $\mathcal{LR}$ level 1 unfolding step.

The U-forest corresponding to a $\mathcal{LR}$ level 1 unfolding is termed a $\mathcal{LR}$ *U-forest*.

Definition 5.10 We say that B loops at level 1 *by a* $\mathcal{LR}$ *strategy* if for all $n \geq 1$ there exists a $\mathcal{LR}$ level 1 unfolding step t_n and a B_n not failure such that $1 \vdash_u t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

Recall the functions $E(m)$ and $L(m)$ introduced in section 4.3.

Lemma 5.11 The following are sufficient conditions for B to loop at level 1 by a $\mathcal{LR}$ strategy:

1. For all $N \geq 0$, there for all n with $1 \leq n \leq N$ exists a $\mathcal{LR}$ level 1 unfolding step t_n and B_n not failure such that $1 \vdash_u^S t_n : B_{n-1} \to B_n$ (again, $B_0 = B$).

2. For all $n \geq 0$, there exists a transition t_n and B_n not failure, with $1 \vdash_u t_n : B \to B_n$, such that $E(t_n) \geq n$.

3. For all $n \geq 0$, there exists a transition t_n and B_n not failure, with $1 \vdash_u t_n : B \to B_n$, such that the length of the leftmost working path $\geq n$.

4. For all $n \geq 0$, there exists a transition t_n and B_n not failure, with $1 \vdash_u t_n : B \to B_n$, such that $L(t_n) \geq n$.

PROOF: First consider (1); then (2) $\Rightarrow$ (1); then (3) $\Rightarrow$ (1) and finally (4) $\Rightarrow$ (3). □

5.4 Folding at level 1

We now define what it means for a transition t from B to B' to be a level 1 folding step, to be written $1 \vdash_f^S t : B \to B'$.

We will assume the existence of a partial function $s(G, i)$ such that $\mathcal{I}_{s(G,i)}$ $(t(G, i))$ is reversible, and such that $\mathcal{I}_{s(G,i)}(c(G, i'))$ is failure for all $i' \neq i$.

$$\frac{t(G,i) \in \mathcal{RU}_0, s(G,i) \text{ defined}}{1 \vdash_f^S \mathcal{I}_s(Id\, Ca_{H_1} \& \mathcal{R}(\mathcal{I}_{s(G,i)}(t(G,i))) \& Id\, Ca_{H_2})} \tag{8}$$

Next we define what it means for a transition t from B to B' to be a level 1 folding, to be written $1 \vdash_f t : B \to B'$:

$$\frac{1 \vdash_f^S t : B \to B'}{1 \vdash_f t : B \to B'} \tag{9}$$

$$\frac{}{1 \vdash_f Id_B : B \to B} \tag{10}$$

$$\frac{1 \vdash_f t : B \to B', 1 \vdash_f t' : B' \to B''}{1 \vdash_f t \star t' : B \to B''} \tag{11}$$

Fact 5.12

1. If $1 \vdash_f^S t$, then $1 \vdash_f^S t\&Id_B$ and $1 \vdash_f^S Id_B\&t$ for basic configuration B.
2. If $1 \vdash_f t_1$ and $1 \vdash_f t_2$, then $1 \vdash_f t_1\&t_2$.
3. If $1 \vdash_f^S t$ ($1 \vdash_f t$), then $1 \vdash_f^S \mathcal{I}_s(t)$ ($1 \vdash_f \mathcal{I}_s(t)$) for specialization s.
4. If $1 \vdash_f^S t : B \to B'$ (or $1 \vdash_f t : B \to B'$), then t is reversible.
5. If $1 \vdash_f^S t : B \to B'$ then $1 \vdash_u^S \mathcal{R}(t) : B' \to B$. If $1 \vdash_f t : B \to B'$) then $1 \vdash_u \mathcal{R}(t) : B' \to B$.

Proof: Mostly as in the proof of fact 5.1. The last point follows from

$$\begin{array}{ll} & \mathcal{R}(\mathcal{I}_s(Id_{Ca_{H_1}}\&\mathcal{R}(\mathcal{I}_{s(G,i)}(t(G,i)))\&Id_{Ca_{H_2}})) \\ = & \mathcal{I}_s(Id_{Ca_{H_1}}\&\mathcal{I}_{s(G,i)}(t(G,i))\&Id_{Ca_{H_2}}) \\ = & \mathcal{I}_s(\mathcal{I}_{Id\_}(Id_{Ca_{H_1}})\&\mathcal{I}_{s(G,i)}(t(G,i)\&\mathcal{I}_{Id\_}(Id_{Ca_{H_2}})) \\ = & \mathcal{I}_s(\mathcal{I}_{Id\_\&_{s(G,i)}\&Id\_}(Id_{Ca_{H_1}}\&t(G,i)\&Id_{Ca_{H_2}})) \\ = & \mathcal{I}_{s\star(Id\_\&_{s(G,i)}\&Id\_)}(Id_{Ca_{H_1}}\&t(G,i)\&Id_{Ca_{H_2}}) \end{array}$$

□

5.5 Unfold/fold at level 1

We now define what it means for a transition t from B to B' to be a level 1 transition, to be written $1 \vdash t : B \to B'$.

$$\frac{1 \vdash_u^S B \to B'}{1 \vdash t : B \to B'} \tag{12}$$

$$\frac{1 \vdash_f^S t : B \to B'}{1 \vdash t : B \to B'} \tag{13}$$

$$\frac{}{1 \vdash Id_B : B \to B'} \tag{14}$$

$$\frac{1 \vdash t : B \to B', 1 \vdash t' : B' \to B''}{1 \vdash t \star t' : B \to B''} \tag{15}$$

Fact 5.13

1. If $1 \vdash t_1$ and $1 \vdash t_2$, then $1 \vdash t_1 \& t_2$.
2. If $1 \vdash t$, then $1 \vdash \mathcal{I}_s(t)$ for specialization s.

5.6 Fundamental properties of level 1 transitions

The switching lemma

Lemma 5.14 Suppose $1 \vdash^S_f t_1 : B_1 \to B$, $1 \vdash^S_u t_2 : B \to B_2$, with B_2 not failure. Then one of two holds:

- $B_1 = B_2$, $t_1 \star t_2 = Id_{B_1}$
- There exists B', t_1', t_2' with $1 \vdash^S_u t_1' : B_1 \to B'$, $1 \vdash^S_f t_2' : B' \to B_2$ such that

  $t_1 \star t_2 = t_1' \star t_2'$

PROOF: This lemma, as well as its proof, is very similar to lemma 5.2.

We can assume that the transitions involved are of the following form:

$$\begin{array}{rcl} t_1 & = & \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}} \& \mathcal{R}(\mathcal{I}_{s(G_1,i_1)}(t(G_1,i_1))) \& Id_{Ca_{H_{12}}}) \\ t_2 & = & \mathcal{I}_{s_2}(Id_{Ca_{H_{21}}} \& t(G_2,i_2) \& Id_{Ca_{H_{22}}}) \end{array}$$

With $s = s_1 \star (Id\_ \& s(G_1, i_1) \& Id\_)$, we from the fact that t_1 is a transition to B and t_2 is a transition from B get

$$\mathcal{I}_s(Ca_{H_{11} \& G_1 \& H_{12}}) = B = \mathcal{I}_{s_2}(Ca_{H_{21} \& G_2 \& H_{22}})$$

From this we infer $H_{11} \& G_1 \& H_{12} = H_{21} \& G_2 \& H_{22}$ and that $s = s_2$. Now two posse bilities:

- $H_{11} = H_{21}$. Then $G_1 = G_2$, and $H_{12} = H_{22}$. Again two possibilities:

– $i_1 \neq i_2$. Then we have
$B_2 = \mathcal{I}_{s_1}(Ca_{H_{21}} \& \mathcal{I}_{s(G_1,i_1)}(c(G_1,i_2)) \& Ca_{H_{22}})$ but $\mathcal{I}_{s(G_1,i_1)}(c(G_1,i_2))$ is failure by the definition of $s(G,i)$, hence B_2 is failure contradicting our assumption.

– $i_1 = i_2$. Then it is obvious that $B_1 = B_2$. And

$$\begin{aligned} t_1 \star t_2 &= \mathcal{I}_s(Id_{Ca_{H_{11}}} \& (\mathcal{R}(t(G_1,i_1)) \star t(G_1,i_1)) \& Id_{Ca_{H_{12}}}) \\ &= \mathcal{I}_s(Id_{Ca_{H_{11}}} \& Id_{c(G_1,i_1)} \& Id_{Ca_{H_{12}}}) \\ &= Id_{\mathcal{I}_s(Ca_{H_{11}} \& c(G_1,i_1) \& Ca_{H_{12}})} = Id_{B_1} \end{aligned}$$

where we have used fact 4.30,(4).

- $H_{11} \neq H_{21}$. We can wlog. assume that H_{11} is shorter than H_{21}. Then there exists H such that $H_{12} = H \& G_2 \& H_{22}$, $H_{21} = H_{11} \& G_1 \& H$. Now define

$$\begin{aligned} t'_1 &= \mathcal{I}_s(Id_{Ca_{H_{11}}} \& Id_{c(G_1,i_1)} \& Id_{Ca_H} \& t(G_2,i_2) \& Id_{Ca_{H_{22}}}) \\ t'_2 &= \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}} \& \mathcal{R}(\mathcal{I}_{s(G_1,i_1)}(t(G_1,i_1))) \& Id_{Ca_H} \& Id_{c(G_2,i_2)} \& Id_{Ca_{H_{22}}}) \end{aligned}$$

We can easily calculate

$$\begin{aligned} & t'_1 \star t'_2 \\ = \; & \mathcal{I}_{s_1}(Id_{Ca_{H_{11}}} \& \mathcal{R}(\mathcal{I}_{s(G_1,i_1)}(t(G_1,i_1))) \& Id_{Ca_H} \& t(G_1,i_1) \& Id_{Ca_{H_{22}}}) \\ = \; & t_1 \star t_2 \end{aligned}$$

To show that $1 \vdash^S_u t'_1$, notice that we can write $c(G_1,i_1)$ on the form $\mathcal{I}_{s'}(Ca_{H'})$ for some s', H'. To show that $1 \vdash^S_f t'_2$, apply the same observation to $c(G_2,i_2)$.

□

The normalization lemma

Lemma 5.15 Suppose $1 \vdash t : B \to B'$, with B' not failure. Then there exists B'', t_1 and t_2 such that $1 \vdash_u t_1 : B \to B''$, $1 \vdash_f t_2 : B'' \to B'$, $t_1 \star t_2 = t$.

From transitions being non-increasing we conclude that B'' is not failure, and from t_1, t_2 and $t_1 \star t_2$ trivially being $\neq \bot$ we conclude $t \neq \bot$.

PROOF: There exists $t_1, \ldots, t_n$ such that $t = t_1 \star \cdots \star t_n$ and such that for each $i \in \{1 \ldots n\}$ either $1 \vdash^S_u t_i$ or $1 \vdash^S_f t_i$. We will use induction in

the number of times an application of the fold-rule precedes (not necessarily immediately) an application of the unfold-rule. If this number is zero, we are through. Otherwise there exists i, $1 \leq i \leq n$, such that $1 \vdash_f^S t_i$, $1 \vdash_u^S t_{i+1}$. Now apply lemma 5.14. Two possibilities:

- $t_i \star t_{i+1} = Id\_$. Then

  $t = t_1 \star \cdots \star t_{i-1} \star t_{i+2} \star \cdots \star t_n$

  with a strictly smaller number of "inversions".

- There exists t'_i, t'_{i+1} such that $t'_i \star t'_{i+1} = t_i \star t_{i+1}$, and such that $1 \vdash_u^S t'_i$, $1 \vdash_u^S t'_{i+1}$. Now

  $t = t_1 \star \cdots \star t_{i-1} \star t'_i \star t'_{i+1} \star t_{i+2} \star \cdots \star t_n$

  and again the number of inversions has decreased.

□

5.7 Unfolding at level 2

Now assume that we have defined $\mathcal{RU}_1$, a finite set of rules at level 1, such that $t \in \mathcal{RU}_1$ implies that $1 \vdash t$.

We now define what it means for a transition t from B to B' to be a level 2 unfolding step, to be written $2 \vdash^S t : B \to B'$:

$$\frac{t \in \mathcal{RU}_1}{2 \vdash^S \mathcal{I}_s(Id_{Ca_{H_1}} \& t \& Id_{Ca_{H_2}})} \tag{16}$$

Next we define what it means for a transition t from B to B' to be a level 2 unfolding, to be written $2 \vdash t : B \to B'$:

$$\frac{2 \vdash^S t : B \to B'}{2 \vdash t : B \to B'} \tag{17}$$

$$\frac{}{2 \vdash Id_B : B \to B'} \tag{18}$$

$$\frac{2 \vdash t : B \to B', 2 \vdash t' : B' \to B''}{2 \vdash t \star t' : B \to B''} \tag{19}$$

Fact 5.16 Suppose $2 \vdash^S t$ or $2 \vdash t$. Then $1 \vdash t$.

Proof: For $2 \vdash^S t$, exploit e.g. fact 5.13. For $2 \vdash t$, a straight forward induction in the derivation tree. □

Definition 5.17 We say that B loops at level 2 *by some strategy* if for all $n \geq 1$ there exists a t_n and a B_n not failure such that $2 \vdash^S t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

Definition 5.18 We say that t is a *fair level 2 step* if t is of form $\mathcal{I}_s(r_1 \& \cdots \& r_k)$, $k \geq 1$, where $r_1 \ldots r_k$ are level 1 rules.

For a fair level 2 step t, we clearly have $2 \vdash t$.

Definition 5.19 We say that B loops at level 2 *by a fair strategy* if for all $n \geq 1$ there exists a fair level 2 step t_n and a B_n not failure such that $2 \vdash t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

Definition 5.20 We say that t is a $\mathcal{LR}$ *level 2 step* if t is of form $\mathcal{I}_s(r \& Id\_)$ with r a level 1 rule.

Definition 5.21 We say that B loops at level 2 *by a* $\mathcal{LR}$ *strategy* if for all $n \geq 1$ there exists a $\mathcal{LR}$ level 2 step t_n and a B_n not failure such that $2 \vdash^S t_n : B_{n-1} \to B_n$ (here $B_0 = B$).

6 Conditions for termination preservation

Theorem 6.1 Assume all rules in $\mathcal{RU}_1$ satisfies $\mathcal{A}(1)$. Then if B loops at level 2 by some strategy, it also loops at level 1 by some strategy.

PROOF: Let, for all $n \geq 1$, be given t_n and B_n not failure such that $2 \vdash^S t_n : B_{n-1} \to B_n$. Define $t'_n = t_1 \star \cdots \star t_n$. Now $2 \vdash t'_n : B \to B_n$, and by fact 5.16 also $1 \vdash t'_n : B \to B_n$. By lemma 5.15, there exists t''_n, t'''_n and B'_n such that $1 \vdash_u t'''_n : B \to B'_n$, $1 \vdash_f t'''_n : B'_n \to B_n$, $t'_n = t''_n \star t'''_n$ and B'_n not failure.

Due to the assumption of the theorem, each t_i will satisfy $\mathcal{A}(1)$. Then, by lemma 4.34, each t'_n will satisfy $\mathcal{A}(n)$. But then also t''_n will satisfy $\mathcal{A}(n)$. By lemma 5.5, this shows that B loops at level 1 by some strategy. □

Theorem 6.2 Assume all rules in $\mathcal{RU}_1$ satisfies $\mathcal{F}(1)$. Then if B loops at level 2 by a fair strategy, it also loops at level 1 by a fair strategy.

PROOF: Let, for all $n \geq 1$, be given fair level 2 step t_n and B_n not failure such that $2 \vdash^S t_n : B_{n-1} \to B_n$. Define $t'_n = t_1 \star \cdots \star t_n$. Now $2 \vdash t'_n : B \to B_n$, and by fact 5.16 also $1 \vdash t'_n : B \to B_n$. By lemma 5.15. there exists t''_n, t'''_n and B'_n such that $1 \vdash_u t''_n : B \to B'_n$, $1 \vdash_f t'''_n : B' - n \to B_n$, $t'_n = t''_n \star t'''_n$ and B'_n not failure.

Due to the assumption of the theorem, each t_i will satisfy $\mathcal{F}(1)$. Then, by lemma 4.32, each t'_n will satisfy $\mathcal{F}(n)$. But then also t''_n will satisfy $\mathcal{F}(n)$. By lemma 5.8, this shows that B loops at level 1 by a fair strategy. □

Theorem 6.3 Assume all rules in $\mathcal{RU}_1$ satisfies $\mathcal{L}(1)$. Then if B loops at level 2 by a $\mathcal{LR}$ strategy, it also loops at level 1 by a $\mathcal{LR}$ strategy.

PROOF: Let, for all $n \geq 1$, be given $\mathcal{LR}$ level 2 step t_n and B_n not failure such that $2 \vdash^S t_n : B_{n-1} \to B_n$. Define $t'_n = t_1 \star \cdots \star t_n$. Now $2 \vdash t'_n : B \to B_n$, and by fact 5.16 also $1 \vdash t'_n B \to B_n$. By lemma 5.15, there exists t''_n, t'''_n and B'_n such that $1 \vdash_u t''_n : B \to B'_n$, $1 \vdash_f t'''_n : B'_n \to B_n$, $t'_n = t''_n \star t'''_n$ and B'_n not failure.

Due to the assumption of the theorem, each t_i will satisfy $\mathcal{L}(1)$. Then, by lemma 4.37, we will have an increasing sequence

$$t'_1 \prec t'_2 \prec t'_3 \prec \cdots$$

Now two possibilities:

- $E(t'_n)$ is not bounded. Then neither $E(t''_n)$ is bounded, so by lemma 5.11 B loops at level 1 by a $\mathcal{LR}$ strategy.
- $E(t'_n)$ is bounded. Then $L(t'_n)$ is unbounded, so also $L(t''_n)$ is unbounded. Then again lemma 5.11 tells us that B loops at level 1 by a $\mathcal{LR}$ strategy.

□

7 Working with the full search tree

Definition 7.1 A *configuration* C (over K) is a family of basic configurations over K, i.e. consists of an index set I and a mapping B which to each $i \in I$ assigns a basic configuration over K.

Two new operators will be defined, + and $\mathsf{P}(\_)$:

Definition 7.2 Given configurations $C_1 = (I_1, B_1)$ and $C_2 = (I_2, B_2)$, over the same K. Now we define $C_1 + C_2 = (I, B)$ by letting $I = I_1 + I_2$ (where + denotes disjoint union); and by letting$B(in_1(i)) = B_1(i)$, $B(in_2(i)) = B_2(i)$.

Definition 7.3 Given configuration $C = (I, B)$. Let

$I' = \{i \in I \mid B(i) \text{ is not failure}\}$

Now we define $\mathsf{P}(C) = (I', B')$ where $B'(i') = B(i')$. We say that C is *pruned* if $C = \mathsf{P}(C)$.

& will be extended to configurations such that & distributes over +. That is, if $C = (I, B)$ and $C' = (I', B')$ then $C\&C' = (I \times I', B'')$ where $B''(i, i') = B(i)\&B'(i')$. Notice that this is possible only because configurations are multisets with + as multiset union; if configurations had been sequences with + as concatenation it would be impossible to make & left-distributive as well as right-distributive.

$\mathcal{I}\_(\_)$ will be extended to configurations such that

$\mathcal{I}_s(C_1 + C_2) = \mathcal{I}_s(C_1) + \mathcal{I}_s(C_2)$

A lot of new identities hold, not to be stated explicitly here – most are very trivial.

7.1 Transitions

Definition 7.4 A transition t from $C = (I, B)$ to $C' = (I', B')$ now to each $(i, i') \in I \times I'$ assigns a set $t(i, i')$ of U-mirrors from $B(i)$ to $B'(i')$. Moreover, we will demand t to be non-increasing: if $t(i, i')$ is non-empty, $(B(i), B'(i'))$ must be a non-increasing pair.

Definition 7.5 Given transition t from $C = (I, B)$ to $C' = (I', B')$ and transition t' from C' to $C'' = (I'', B'')$. $t \star t'$, a transition from C to C'', is now defined as follows:

$(t \star t')(i, i'') = \{m \star m' \mid \exists i' \in I' : m \in t(i, i'), m' \in t(i', i'')\}$

& on transitions is defined in a similar vein:

Definition 7.6 Given transition t_1 from $C_1 = (I_1, B_1)$ to $C_1' = (I_1', B_1')$, and given transition t_2 from $C_2 = (I_2, B_2)$ to $C_2' = (I_2', B_2')$. Then $t_1 \& t_2$, a transition from $C_1 \& C_2$ to $C_1' \& C_2'$, is defined by

$(t_1 \& t_2)((i_1, i_2), (i_1', i_2')) = \{m_1 \& m_2 \mid m_1 \in t_1(i_1, i_1'), m_2 \in t_2(i_2, i_2')\}$

Definition 7.7 Given transition t from $C = (I, B)$ to $C' = (I', B')$ Then $\mathcal{I}_s(t)$, a transition from $\mathcal{I}_s(C)$ to $\mathcal{I}_s(C')$, is given by

$\mathcal{I}_s(t)(i, i') = t(i, i')$

Definition 7.8 Given configuration $C = (I, B)$, we define Id_C by

$Id_C(i, i) = \{Id_{B(i)}\}, Id_C(i, i') = \emptyset$ for $i \neq i'$

Definition 7.9 Given transition t_1 from $C_1 = (I_1, B_1)$ to $C_1' = (I_1', B_1')$, and given transition t_2 from $C_2 = (I_2, B_2)$ to $C_2' = (I_2', B_2')$ (suppose C_1 and C_2 configurations over the same K). Then $t_1 + t_2$, a transition from $C_1 + C_2$ to $C_1' + C_2'$, is defined by

$$\begin{array}{lcl} (t_1 + t_2)(in_1(i_1,\ in_1(i_1')) & = & t_1(i_1, i_1') \\ (t_1 + t_2)(in_2(i_2,\ in_2(i_2')) & = & t_2(i_2, i_2') \\ (t_1 + t_2)(in_1(i_1,\ in_2(i_2')) & = & \emptyset \\ (t_1 + t_2)(in_2(i_2,\ in_1(i_1')) & = & \emptyset \end{array}$$

Definition 7.10 Given transition t from $C_1 = (I_1, B_1)$ to $C_2 = (I_2, B_2)$. Let $\mathsf{P}(C_1) = (I_1', B_1')$, $\mathsf{P}(C_2) = (I_2', B_2')$. Now define $\mathsf{P}(t)$, a transition from $\mathsf{P}(C_1)$ to $\mathsf{P}(C_2)$ by

$\mathsf{P}(t)(i_1', i_2') = t(i_1', i_2')$

Definition 7.11 Given transition t from $C = (I, B)$ to $C' = (I', B')$, we say that t is reversible iff the following holds for all (i, i') with $t(i, i') \neq \emptyset$: $(B'(i'), B(i))$ is a non-increasing pair, and for all $m \in t(i, i')$ m is a working U-mirror.

If t is reversible we can define $\mathcal{R}(t)$, a transition from C' to C, by stipulating

$$\mathcal{R}(t)(i', i) = \{\mathcal{R}(m) \mid m \in t(i, i')\}$$

Again, a lot of algebraic identities hold – most are quite trivial. Let us just show that

$$(t_1 \& t_2) \star (t_1' \& t_2') = (t_1 \star t_1') \& (t_2 \star t_2') \tag{20}$$

where t_1 is a transition from $C_1 = (I_1, B_1)$ to $C_1' = (I_1', B_1')$, t_1' is a transition from C_1' to $C_1'' = (I_1'', B_1'')$, t_2 is a transition from $C_2 = (I_2, B_2)$ to $C_2' = (I_2', B_2')$ and t_2' is a transition from C_2' to $C_2'' = (I_2'', B_2'')$. Now the left hand side of (20) as well as the right hand side will be a transition from $C_1 \& C_2$ to $C_1'' \& C_2''$. And for $i_1 \in I_1$, $i_2 \in I_2$, $i_1'' \in I_1''$ and $i_2'' \in I_2''$ we have

$$\begin{aligned}
& ((t_1 \& t_2) \star (t_1' \& t_2'))((i_1, i_2), (i_1'', i_2'')) \\
= \; & \{m'' \mid \exists (i_1', i_2'), \exists m \in (t_1 \& t_2)((i_1, i_2), (i_1', i_2')), \\
& \exists m' \in (t_1' \& t_2')((i_1', i_2'), (i_1'', i_2'')) \; : \; m'' = m \star m'\} \\
= \; & \{m'' \mid \exists (i_1', i_2'), \exists m_1 \in t_1(i_1, i_1'), \exists m_2 \in t_2(i_2, i_2'), \\
& \exists m_1' \in t_1'(i_1', i_1''), \exists m_2' \in t_2'(i_2', i_2'') \; : \; m'' = (m_1 \& m_2) \star (m_1' \& m_2')\} \\
= \; & \{m'' \mid \exists (i_1', i_2'), \exists m_1 \in t_1(i_1, i_1'), exists m_2 \in t_2(i_2, i_2'), \\
& \exists m_1' \in t_1'(i_1', i_1''), \exists m_2' \in t_2'(i_2', i_2'') \; : \; m'' = (m_1 \star m_1') \& (m_2 \star m_2')\} \\
= \; & ((t_1 \star t_1') \& (t_2 \star t_2'))((i_1, i_2), (i_1'', i_2''))
\end{aligned}$$

where we have used fact 4.30, (2).

Also we have that

$$\mathsf{P}(t_1 \star t_2) = \mathsf{P}(t_1) \star \mathsf{P}(t_2)$$

This holds only because we demand transitions to be non-increasing.

7.2 The level 0 rules

For each $G \in U$, there exists a rule $t(G) \in \mathcal{RU}_0$ from Ca_G to $c(G)$, where $c(G) = \{c(G,i) \mid i \in OI(G)\}$. Here $m \in t(G)(i)$ iff $m \in t(G,i)$.

7.3 Unfolding at level 1

We now define what it means for a transition t from C to C' to be a *level 1c unfolding step*[10], to be written $1\ c \vdash_u^S t : C \to C'$.

$$\frac{t(G) \in \mathcal{RU}_0}{1 \vdash_u^S \mathcal{I}_s(Id_{Ca_{H_1}} \& t(G) \& Id_{Ca_{H_2}})} \tag{21}$$

$$\frac{}{1\ c \vdash_u^S Id_C} \tag{22}$$

$$\frac{1\ c \vdash_u^S t_1, 1\ c \vdash_u^S t_2}{1\ c \vdash_u^S t_1 + t_2} \tag{23}$$

Fact 7.12 If $1\ c \vdash_u^S t$, then $1\ c \vdash_u^S t \& Id_\_$ and $1\ c \vdash_u^S Id_\_ \& t$. Also, $1\ c \vdash_u^S \mathcal{I}_s t$.

Next we define what it means for a transition t from C to C' to be a level $1c$ unfolding, to be written $1\ c \vdash_u t : C \to C'$:

$$\frac{1\ c \vdash_u^S t : C \to C'}{1\ c \vdash_u \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')} \tag{24}$$

$$\frac{1\ c \vdash_u t : C \to C', 1\ c \vdash_u t' : C' \to C''}{1\ c \vdash_u t \star t' : C \to C''} \tag{25}$$

Observe that if $1\ c \vdash_u t : C \to C'$, then $t = \mathsf{P}(t)$, $C = \mathsf{P}(C)$ and $C' = \mathsf{P}(C')$.

Fact 7.13 If $1\ c \vdash_u t_1$ and $1\ c \vdash_u t_2$, then $1\ c \vdash_u t_1 \& t_2$, $1\ c \vdash_u t_1 + t_2$ and $1\ c \vdash_u \mathsf{P}(\mathcal{I}_s(t_1))$.

Proof: Inductions in the derivation tree:

- Concerning &, it will be enough to show that $1\ c \vdash_u t$ implies $1\ c \vdash_u t \& Id_C$ (and $1\ c \vdash_u Id_C \& t$) for C with $\mathsf{P}(C) = C$, as

  $$t_1 \& t_2 = (t_1 \star Id_\_) \& (Id_\_ \star t_2) = (t_1 \& Id_\_) \star (Id_\_ \& t_2)$$

[10] The "c" to denote that the complete search tree is modeled.

If $t = t_1 \star t_2$ with $1\ c \vdash_u t_1, 1\ c \vdash_u t_2$ this follows from

$t\&Id\_ = (t_1\&Id\_) \star (t_2\&Id\_)$

If $t = \mathsf{P}(t')$ with $1\ c \vdash_u^S t'$, first note that by fact 7.12 $1\ c \vdash_u^S t'\&Id\_$. Thus $1\ c \vdash_u \mathsf{P}(t'\&Id\_)$, i.e. $1\ c \vdash_u t\&Id\_$.

- Concerning $+$, it will be enough to show that $1\ c \vdash_u t'$ implies $1\ c \vdash_u t + Id_C$ with C such that $C = \mathsf{P}(C)$ - as

  $t_1 + t_2 = (t_1 + Id\_) \star (Id\_ + t_2)$

  If $t = t_1 \star t_2$, this follows from

  $t + Id\_ = (t_1 + Id\_) \star (t_2 + Id\_)$

  If $t = \mathsf{P}(t')$ with $1\ c \vdash_u^S t'$, first note that $1\ c \vdash_u^S t' + Id\_$.
  Thus $1\ c \vdash_u \mathsf{P}(t' + Id\_)$, i.e. $1\ c \vdash_u t + Id\_$.

- Concerning $\mathcal{I}\_(\_)$, first suppose $1\ c \vdash_u t$ because $t = \mathsf{P}(t')$, $1\ c \vdash_u^S t'$. Now also $1\ c \vdash_u^S \mathcal{I}_s(t')$, and thus $1\ c \vdash_u \mathsf{P}(\mathcal{I}_s(t'))$, i.e. also $1\ c \vdash_u \mathsf{P}(\mathcal{I}_s(t))$.

  Next suppose $t = t_1 \star t_2$. By induction, $1\ c \vdash_u \mathsf{P}(\mathcal{I}_s(t_1))$ and $1\ c \vdash_u \mathsf{P}(\mathcal{I}_s(t_2))$. The claim now follows from

  $\mathsf{P}(\mathcal{I}_s(t_1)) \star \mathsf{P}(\mathcal{I}_s(t_2)) = \mathsf{P}(\mathcal{I}_s(t_1 \star t_2))$

□

If $1\ c \vdash_u t$, there exists $t_1 \ldots t_n$ such that $t = \mathsf{P}(t_1) \star \ldots \star \mathsf{P}(t_n)$ with $1\ c \vdash_u^S t_i$ for $i = 1 \ldots n$. Again, we can define the length of a transition t as the minimal n which can be used ($n \geq 1$).

Observation 7.14 Suppose $1\ c \vdash_u t : B \to C$, with $C = (I, B)$. Then for all $i \in I$ there exists t_i such that $1\ \vdash_u t_i : B \to B(i)$. We say that $t_i = \pi_i(t)$.

The diamond lemma, revisited

Lemma 7.15 Suppose $1\ c \vdash_u^S t_1 : C \to C_1$ and $1\ c \vdash_u^S t_2 : C \to C_2$. Then there exists C', transition t_1' with $1\ c \vdash_u^S t_1' : C_1 \to C'$ and transition t_2' with $1\ c \vdash_u^S t_2' : C_2 \to C'$ such that $t_1 \star t_1' = t_2 \star t_2'$.

PROOF: Much as the proof of lemma 5.2. A brief sketch: we can assume C to consist of a single basic configuration (if C is the union of several basic configurations each of these can be "treated" separately).

If $t_1 = Id_C$, choose $t_1' = t_2$, $t_2' = Id_{C_2}$. Similarly if $t_2 = Id_C$. So in the following we can assume that t_1 as well as t_2 are derived by means of (21). If "the same G" is unfolded, we take $t_1' = t_2' = Id_{C_1}$. Otherwise, we can - dispensing with the Id_{Ca_H}-parts - write $t_1 = \mathcal{I}_s(Id_{Ca_{G_1}} \& t(G_2))$, $t_2 = \mathcal{I}_s(t(G_1) \& Id_{Ca_{G_2}})$. Then C_1 and C_2 take the form

$$C_1 = \{\mathcal{I}_s(Ca_{G_1} \& c(G_2, i)) \mid i \in OI(G_2)\}$$
$$C_2 = \{\mathcal{I}_s(c(G_1, i) \& Ca_{G_1}) \mid i \in OI(G_1)\}$$

Now define t_1', t_2' as follows:

$$t_1' = \mathcal{I}_s(t(G_1) \& Id_{c(G_2)})$$
$$t_2' = \mathcal{I}_s(Id_{c(G_1)} \& t(G_2))\}$$

That e.g. $1\ c \vdash_u^S t_1'$ follows from the fact that t_1' can be written on the form

$$t_1' = \sum_{i \in OI_{(G_2)}} \mathcal{I}_s(t(G_1) \& Id_{c(G_2,i)})$$

where again $c(G_2, i)$ can be written on the form $\mathcal{I}_{s'}(Ca_{H'})$.

That $t_1 \star t_1' = t_2 \star t_2'$ follows from the fact that both equal $\mathcal{I}_s(t(G_1) \& t(G_2))$.

□

The Church-Rosser lemma, revisited

Lemma 7.16 Suppose $1\ c \vdash_u t_1 : C \to C_1$ and $1\ c \vdash_u t_{12} : C \to C_2$. Then there exists C_3 and transitions t_3, t_4 such that $1\ c \vdash_u t_3 : C_1 \to C_3$, $1\ c \vdash_u t_4 : C_2 \to C_3$, and such that $t_1 \star t_3 = t_2 \star t_4$.

Moreover, the length of t_3 is less than or equal the length of t_2, and the length of t_4 is less than or equal the length of t_1.

PROOF: We use induction in the length of t_1 plus the length of t_2. For the induction step, proceed as in the proof of lemma 5.3. So assume that both transitions have length 1. The situation is as follows: we have $1\ c \vdash_u^S t_1' : C' \to C_1'$ and $1\ c \vdash_u^S t_2' : C'' \to C_2'$, with $t_1 = \mathsf{P}(t_1'), t_2 = \mathsf{P}(t_2')$, and $C = \mathsf{P}(C') = \mathsf{P}(C'')$. We can assume that $C' = C''$, as we can "expand" t_1' and t_2'. Now apply lemma 7.15, to find t_3', t_4' and C_3' with $1\ c \vdash_u^S t_3' : C_1' \to C_3'$, $1\ c \vdash_u^S t_4' : C_2' \to C_3'$, $t_1' \star t_3' = t_2' \star t_4'$. Then define $C_3 = \mathsf{P}(C_3'), t_3 = \mathsf{P}(t_3'), t_4 = \mathsf{P}(t_4')$. Then $1\ c \vdash_u t_3, 1\ c \vdash_u t_4$, and

$t_1 \star t_3 = \mathsf{P}(t_1') \star \mathsf{P}(t_3') = \mathsf{P}(t_1' \star t_3') = \mathsf{P}(t_2' \star t_4') = t_2 \star t_4$
□

7.4 Level 1 semantics

We say that a configuration $C = (I, B)$ is in *normal form* iff C is pruned and for all $i \in I$ $B(i)$ is empty.

Lemma 7.16, together with the observation that if C is in normal form and $1\ c \vdash_u t : C \to C'$ then $C = C'$, shows that the following is welldefined:

Definition 7.17 Given basic configuration B (not failure). Suppose $1\ c \vdash_u t : B \to C$ with C in normal form. Then $[\![B]\!]_1 = C$.

If no such t and C exists, $[\![B]\!]_1 = \bot$.

Fact 7.18 $[\![B]\!]_1 = \bot$ iff B loops at level 1 by a fair strategy (as defined in definition 5.7)

PROOF: Suppose $[\![B]\!]_1 = \bot$. Given $n \geq 0$. It is easily seen that there will exist t_n and $C' = (I', B')$ such that $1\ c \vdash_u t_n : B \to C'$, and such that for all $i \in I'$ we have $1 \vdash_u \pi_i(t_n) : B \to B'(i)$ where either $B'(i)$ is empty or $\pi_i(t_n)$ is composed of n fair level 1 unfolding steps. Now, there exists at least one $i \in I'$ where $B'(i)$ is not empty (otherwise C' would be in normal form). Hence we conclude that B loops at level 1 by a fair strategy.

Conversely, suppose there exists C in normal form such that $1\ c \vdash_u t : B \to C$. Then there exists U-forests $f_1 \ldots f_k$ such that $\mathcal{U}_{f_i}(B)$ is either empty or failure for all i. Moreover, there exists a n such that for all U-forests f where

the shortest working path is longer than n, there exists an i such that f can be written as $f_i \star f'$ for some f'. Hence for such f also $\mathcal{U}_f(B)$ is empty or failure, showing that B does not loop at level 1 by a fair strategy. □

$\mathcal{LR}$ semantics

We say that t is a $\mathcal{LR}$ level 1c single step if t takes the form $t = \mathcal{I}_s(t(G)\&Id_B)$, with $t(G) \in \mathcal{RU}_0$. We say that t is a $\mathcal{LR}$ level 1c step if t takes the form $t = t_1 + \cdots + t_k$, at least one t_i being a $\mathcal{LR}$ level 1c single step and the rest being of form Id_B, B empty. We say that t is a $\mathcal{LR}$ level 1c unfolding if t takes the form $t = \mathsf{P}(t_1) \star \cdots \star \mathsf{P}(t_i)$, each t_i being a $\mathcal{LR}$ level 1c step.

Definition 7.19 Given basic configuration B. Suppose $1\ c \vdash_u t : B \to C$ with C in normal form, where t is a $\mathcal{LR}$ level 1c unfolding. Then $[\![B]\!]^L_1 = C$.

If no such t and C exists, $[\![B]\!]_1 = \bot$.

Fact 7.20 $[\![B]\!]^L_1 = \bot$ iff B loops at level 1 by a $\mathcal{LR}$ strategy (as defined in definition 5.10).

Proof: Suppose $[\![B]\!]^L_1 = \bot$. Given $n \geq 0$. It is easily seen that there will exist t_n and $C' = (I', B')$ such that $1\ c \vdash_u t_n : B \to C'$, and such that for all $i \in I'$ we have $1 \vdash_u \pi_i(t_n) : B \to B'(i)$ where either B'_i is empty or $\pi_i(t_n)$ is composed of n $\mathcal{LR}$ level 1 unfolding steps. Now, there exists at least one $i \in I'$ where $B'(i)$ is not empty (otherwise C' would be in normal form). Hence we conclude that B loops at level 1 by a $\mathcal{LR}$ strategy.

Conversely, suppose there exists C in normal form such that $1\ c \vdash_u t : B \to C$, with t a $\mathcal{LR}$ level 1c unfolding. Then there exists U-forests $f_1 \ldots f_k$ such that $\mathcal{U}_{f_i}(B)$ is either empty or failure for all i. Moreover, there exists a n such that for all $\mathcal{LR}$ U-forests f of size greater than n, there exists an i such that f can be written as $f_i \star f'$ for some f'. Hence for such f also $\mathcal{U}_f(B)$ is empty or failure, showing that B does not loop at level 1 by a $\mathcal{LR}$ strategy. □

□

7.5 Folding at level 1

We now define what it means for a transition t from C to C' to be a level 1c transition, to be written $1 \vdash^S_f t : C \to C'$.

We will assume the existence of a partial (perhaps multivalued) function $s(G)$ such that $\mathsf{P}(\mathcal{I}_{s(G)}(t(G)))$ is reversible.

$$\frac{t(G) \in \mathcal{RU}_0}{1\ c \vdash_f^S \mathcal{I}_s(Id_{Ca_{H_1}} \& \mathcal{R}(\mathsf{P}(\mathcal{I}_{s(G)}(t(G)))) \& Id_{Ca_{H_2}})} \tag{26}$$

$$\frac{}{1\ c \vdash_f^S Id_C} \tag{27}$$

$$\frac{1\ c \vdash_f^S t_1, 1\ c \vdash_f^S t_2}{1\ c \vdash_f^S t_1 + t_2} \tag{28}$$

Next we define what it means for a transition t from C to C' to be a level $1c$ folding, to be written $1\ c \vdash_f t : C \to C'$:

$$\frac{1\ c \vdash_f^S t : C \to C'}{1\ c \vdash_f \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')} \tag{29}$$

$$\frac{1\ c \vdash_f t : C \to C', 1\ c \vdash_f t' : C' \to C''}{1\ c \vdash_f t \star t' : C \to C''} \tag{30}$$

7.6 Unfold/fold at level 1

Next we now define what it means for a transition t from C to C' to be a level $1c$ folding, to be written $1\ c \vdash t : B \to B'$.

$$\frac{1\ c \vdash_u^S t : C \to C'}{1\ c \vdash \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')} \tag{31}$$

$$\frac{1\ c \vdash_f^S t : C \to C'}{1\ c \vdash \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')} \tag{32}$$

$$\frac{1\ c \vdash t : C \to C', 1\ c \vdash t' : C' \to C''}{1\ c \vdash t \star t' : C \to C'} \tag{33}$$

Fact 7.21 If $1\ c \vdash t_1$ and $1\ c \vdash t_2$, then also $1\ c \vdash t_1 \& t_2$ and $1\ c \vdash \mathsf{P}(\mathcal{I}_s(t_1))$.

The switching lemma, revisited

Lemma 7.22 Suppose $1\ c \vdash t_1 : C_1 \to C$ is a folding step, i.e. is derived by means of rule (32), and suppose $1\ c \vdash t_2 : C \to C_2$ is an unfolding step, i.e. is derived by means of rule (31). Then there exists t_3, t_4 and C_3 such that

$1\ c \vdash t_{31} : C_1 \to C_3$ by an unfolding step; $1\ c \vdash t_4 : C_3 \to C_2$ by a folding step; and $t_1 \star t_2 = t_3 \star t_4$.

PROOF: (A sketch only) There exists $t'_1, t'_2, C'_1, C'_2, C'$ and C'' such that $1\ c \vdash^S_f t'_1 : C'_1 \to C'$. $1\ c \vdash^S_u t'_2 : C'' \to C'_2$, $t_1 = \mathsf{P}(t'_1)$, $t_2 = \mathsf{P}(t'_2)$ and $C = \mathsf{P}(C') = \mathsf{P}(C'')$. It is not hard to see that we can assume that t'_1 and t'_2 are derived by means of (26) and (21) respectively, and that we can assume C to be a singleton (i.e. not $\emptyset$).

There are two cases (where we dispense with writing the Id_{Ca_H}-parts):

1. t'_1, t'_2 takes the form

   $$t'_1 = \mathcal{I}_{s_1}(\mathcal{R}(\mathsf{P}(\mathcal{I}_{s(G)}(t(G)))))$$
   $$t'_2 = \mathcal{I}_{s_2}(t(G))$$

   Thus $t_1 = \mathcal{R}(\mathsf{P}(\mathcal{I}_{s_1 \star s(G)}(t(G)))), t_2 = \mathsf{P}(\mathcal{I}_{s_2}(t(G)))$. Then

   $$\mathcal{I}_{s_1 \star s(G)}(Ca_G) = C = \mathcal{I}_{s_2}(Ca_G)$$

   enabling us to conclude that $s_1 \star s(G) = s_2$ and hence also $C_1 = C_2$. Now we can use $t_3 = t_4 = Id_{C_1}$. That $t_1 \star t_2 = Id_{C_1}$ is an easy consequence of fact 4.30,(4).

2. t'_1, t'_2 takes the form

   $$t'_1 = \mathcal{I}_{s_1}(\mathcal{R}(\mathsf{P}(\mathcal{I}_{s(G_1)}(t(G_1))))\&Id_{Ca_{G_2}})$$
   $$t'_2 = \mathcal{I}_{s_2}(Id_{Ca_{G_1}}\&t(G_2))$$

   Now apply the usual technique: we infer that $s_1 \star (s(G_1)\&Id_{\_}) = s_2$, and define

   $$t'_4 = \mathcal{I}_{s_1}(\mathcal{R}(\mathsf{P}(\mathcal{I}_{s(G_1)}(t(G_1))))\&Id_{c(G_2)})$$
   $$t'_3 = \mathcal{I}_{s_2}(Id_{c(G_1)}\&t(G_2))$$

   We now define $t_3 = \mathsf{P}(t'_3)$, $t_4 = \mathsf{P}(t'_4)$. Clearly $1\ c \vdash^S_u t'_3$, $1\ c \vdash^S_f t'_4$ and

   $$t_3 \star t_4 = \mathsf{P}(t'_3 \star t'_4) = \mathsf{P}(t'_1 \star t'_2) = t_1 \star t_2$$

□

The normalization lemma, revisited

Lemma 7.23 Suppose $1\ c \vdash t : C' \to C'$. Then there exists t_1, t_2, C'' such that $1\ c \vdash_u t_1 : C \to C''$, $1\ c \vdash_f t_2 : C'' \to C'$, $t = t_1 \star t_2$.

PROOF: As the proof of lemma 5.15, now exploiting lemma 7.22. □

7.7 Unfolding at level 2

Now assume that we have defined $\mathcal{RU}_1$, a finite set of rules at level 1. Assume that there is a bijective correspondence between $\mathcal{RU}_1$ and U, such that the rule corresponding to G is a transition from Ca_G. Then there is no risk of a configuration being stuck (i.e. not in normal form but cannot be unfolded further).

We now define what it means for a transition t from C to C' to be a level $2c$ unfolding step, to be written $2\ c \vdash^S t : C \to C'$.

$$\frac{t \in \mathcal{RU}_1}{2\ c \vdash^S \mathcal{I}_s(id_{Ca_{H_1}} \& t \& id_{Ca_{H_2}})} \tag{34}$$

$$\frac{}{2\ c \vdash^S Id_C} \tag{35}$$

$$\frac{2\ c \vdash^S t_1, 2\ c \vdash^S t_2}{2\ c \vdash^S t_1 + t_2} \tag{36}$$

Next we define what it means for a transition t from C to C' to be a level $2c$ unfolding, to be written $2\ c \vdash t : C \to C'$:

$$\frac{2\ c \vdash^S t : C \to C'}{2\ c \vdash \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')} \tag{37}$$

$$\frac{2\ c \vdash t : C \to C', 21\ c \vdash t' : C' \to C''}{2\ c \vdash t \star t' : C \to C'} \tag{38}$$

Fact 7.24 If $2\ c \vdash t : C \to C'$, also $1\ c \vdash t : C \to C'$.
If $2\ c \vdash^S t : C \to C'$, also $1\ c \vdash \mathsf{P}(t) : \mathsf{P}(C) \to \mathsf{P}(C')$.

PROOF: Induction in the derivation tree: the only interesting case is where (34) has been applied. We must show that

$1\ c \vdash \mathsf{P}(\mathcal{I}_s(Id_{Ca_{H_1}} \& t \& Id_{Ca_{H_2}}))$

But this is a consequence of fact 7.21. □

By combining lemma 7.23 and fact 7.24 we get

Fact 7.25 If $2\ c \vdash t : C \to C'$, there exists t_1, t_2, C'' such that

$1\ c \vdash_u t_1 : C \to C''$, $1c \vdash_f t_2 : C'' \to C'$, $t = t_1 \star t_2$.

If C' is in normal form, $C' = C''$.

7.8 Level 2 semantics

Definition 7.26 Given basic configuration B (not failure). Suppose $2\ c \vdash t : B \to C$ with C in normal form. Then $[\![B]\!]_2 = C$.

If no such t and C exists, $[\![B]\!]_2 = \bot$.

By fact 7.25, this is well-defined.

We say that t is a fair level $2c$ single step if t takes the form $t = \mathcal{I}_s(t_1 \& \cdots \& t_n), n \geq 1$, each $t_i \in \mathcal{RU}_1$. We say that t is a fair level $2c$ step if t takes the form $t = t_1 + \cdots + t_k$, at least one t_i being a fair level $2c$ single step and the rest of form Id_B with B empty. We say that t is a fair level $2c$ unfolding if t takes the form $t = \mathsf{P}(t_1) \star \cdots \star \mathsf{P}(t_k)$, each t_i being a fair level $2c$ step. We say that B loops at level $2c$ by a fair strategy if for all $i > 0$ there exists C_i not in normal form and fair level $2c$ step t_i from C_{i-1} to C_i (here $C_0 = B$).

We say that t is a $\mathcal{LR}$ level $2c$ single step if t takes the form $t = \mathcal{I}_s(t_1 \& Id_B)$, with $t_1 \in \mathcal{RU}_1$. We say that t is a $\mathcal{LR}$ level $2c$ step if t takes the form $t = t_1 + \cdots + t_k$, at least one t_i being a $\mathcal{LR}$ level 2 single step and the rest of form Id_B with B empty. We say that t is a $\mathcal{LR}$ level $2c$ unfolding if t takes the form $t = \mathsf{P}(t_1) \star \cdots \star \mathsf{P}(t_k)$, each t_i being a $\mathcal{LR}$ level $2c$ step. We say that B loops at level $2c$ by the $\mathcal{LR}$ strategy if for all $i > 0$ there exists C_i not in normal form and $\mathcal{LR}$ level $2c$ step t_i from C_{i-1} to C_i (here $C_0 = B$).

Definition 7.27 Given basic configuration B (not failure) Suppose $2\ c \vdash t : B \to C$ with C in normal form, where t is a $\mathcal{LR}$ level $2c$ unfolding. Then $[\![B]\!]_2^L = C$.

If no such t and C exists, $[\![B]\!]_2^L = \bot$.

Clearly, $[\![B]\!]_2^L = \bot$ iff B loops at level $2c$ by the $\mathcal{LR}$ strategy.

7.9 Total correctness

Theorem 7.28 Assume all U-mirrors occurring in rules in $\mathcal{RU}_1$ satisfy $\mathcal{F}(1)$. Then for all B, $[\![B]\!]_2 = [\![B]\!]_1$.

PROOF: First suppose $[\![B]\!]_2 = C \neq \bot$. By fact 7.25, also $[\![B]\!]_1 = C$.

Now suppose $[\![B]\!]_2 = \bot$. Then for all $n \geq 1$ there will exist fair level $2c$ step t_n and C_n not in normal form such that $2\ c \vdash \mathsf{P}(t_n) : C_{n-1} \to C_n$ ($C_0 = B$). Let $t'_n = \mathsf{P}(t_1) \star \cdots \star \mathsf{P}(t_k)$. $2\ c \vdash t'_n : B \to C_n$, and by fact 7.25 there exists t''_n, t'''_n and C'_n such that $1\ c \vdash_u t''_n : B \to C'_n$, $1\ c \vdash_f t'''_n : C'_n \to C_n$ and $t'_n = t''_n \star t'''_n$.

As C_n contains a non-empty basic configuration, this shows that t'_n for all n contains at least one mirror from B to a non-empty basic configuration. Then it will be possible (by Königs lemma) for all n to find $m_n \in t_n$ such that $m'_n = m_1 \star \cdots \star m_n$ is a mirror in t'_n from B to a non-empty basic configuration. Also there will exist mirrors $m''_n \in t''_n$ and $m'''_n \in t'''_n$ such that $m'_n = m''_n \star m'''_n$. It is easily seen that $1 \vdash_u m''_n : B \to B'_n$, with B'_n not failure.

Due to the assumption of the theorem, each m_i will satisfy $\mathcal{F}(1)$. Then, by lemma 4.32, each m'_n will satisfy $\mathcal{F}(n)$. But then also m''_n will satisfy $\mathcal{F}(n)$. By lemma 5.8, this shows that B loops at level 1 by a fair strategy, and by fact 7.18 $[\![B]\!]_1 = \bot$. □

Theorem 7.29 Assume all U-mirrors occurring in rules in $\mathcal{RU}_1$ satisfy $\mathcal{L}(1)$. Then for all B, $[\![B]\!]_2^L \geq [\![B]\!]_1^L$.

PROOF: First suppose $[\![B]\!]_2^L = C \neq \bot$. Then also $[\![B]\!]_2 = C$, so by fact 7.25 $[\![B]\!]_1 = C$. Now either $[\![B]\!]_1^L = C$ or $[\![B]\!]_1^L = \bot$.

Now suppose $[\![B]\!]_2^L = \bot$. Then for all $n \geq 1$ there will exist $\mathcal{LR}$ level $2c$ step t_n and C_n not in normal form such that $2\ c \vdash \mathsf{P}(t_n) : C_{n-1} \to C_n$ ($C_0 = B$). Let $t'_n = \mathsf{P}(t_1) \star \cdots \star \mathsf{P}(t_n)$. $2\ c \vdash t'_n : B \to C_n$, and by fact 7.25 there exists t''_n, t'''_n and C'_n such that $1\ c \vdash_u t''_n : B \to C'_n$, $1\ c \vdash_f t''''_n : C'_n \to C_n$ and $t'_n = t''_n \star t'''_n$.

As C_n contains a non-empty basic configuration, this shows that t'_n for all n contains at least one mirror from B to a non-empty basic configuration.

Then it will be possible for all n to find $m_n \in t_n$ such that $m'_n = m_1 \star \cdots m_n$ is a mirror in t'_n from B to a non-empty basic configuration. Also there will exist mirrors $m''_n \in t''_n$ and $m'''_n \in t'''_n$ such that $m'_n = m''_n \star m'''_n$. It is easily seen that $1 \vdash_u m''_n : B \to B'_n$, with B'_n not failure.

Due to the assumption of the theorem, each m_i satisfies $\mathcal{L}(1)$. By lemma 4.37, there exists an increasing sequence

$$m'_1 \prec m'_2 \prec m'_3 \prec \cdots$$

Now two possibilities:

1. $E(m'_n)$ is not bounded. Then neither $E(m''_n)$ is bounded, so by lemma 5.11 B loops at level 1 by a $\mathcal{LR}$ strategy.

2. $E(m'_n)$ is bounded. Then $L(m'_n)$ is unbounded, so also $L(m''_n)$ is unbounded. Then again lemma 5.11 tells us that B loops at level 1 by a $\mathcal{LR}$ strategy.

In both cases, fact 7.20 tells us that $[\![B]\!]^L_1 = \bot$. □

References

[Amt92] Torben Amtoft. Unfold/fold transformations preserving termination properties. To appear in the proceedings of PLILP 92, 1992.

[BD77] R.M. Burstall and John Darlington. A transformation system for developing recursive programs. *Journal of the ACM*, 24(1):44–67, January 1977.

[DP88] John Darlington and Helen Pull. A program development methodology based on a unified approach to execution and transformation. In D. Bjørner, A.P. Ershov, and N.D. Jones, editors, *Partial Evaluation and Mixed Computation*, pages 117–131. North-Holland, 1988.

[Gre87] Steve Gregory. *Parallel Logic Programming in PARLOG - the language and its implementation.* Addison-Wesley, 1987.

[GS91] P. A. Gardner and J. C. Shepherdson. Unfold/fold transformations of logic programs. In *Computational Proofs: Essays in honour of Alan Robinson.* 1991.

[Han91] Torben Amtoft Hansen. Properties of unfolding-based meta-level systems. In *Partial Evaluation and Semantics-Based Program manipulation, New Haven, Connecticut. (Sigplan Notices, vol. 26, no. 9)*, 1991.

[KK90] Tadashi Kawamura and Tadashi Kanamori. Preservation of stronger equivalence in unfold/fold logic program transformation. *Theoretical Computer Science,* 75:139–156, 1990.

[Kot85] Laurent Kott. Unfold/fold program transformations. In Maurice Nivat and John C. Reynolds, editors, *Algebraic methods in Semantics*, chapter 12. Cambridge University Press, 1985.

[Llo84] J. W. Lloyd. *Foundations of Logic Programming.* Springer-Verlag, 1984.

[NN90] Hanne Riis Nielson and Flemming Nielson. Eureka definitions for free! or disagreement points for fold/unfold transformations. In Neil D. Jones, editor, *ESOP 90, Copenhagen, Denmark. LNCS 432*, pages 291–305, May 1990.

[Pal89] Catuscia Palamidessi. Algebraic properties of idempotent substitutions. Technical Report TR-33/89, University of Piss, 1989.

[Plo81] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report FN-19, DAIMI, University of Aarhus, Denmark, September 1981.

[PP91a] Maurizio Proietti and Alberto Pettorossi. Semantics preserving transformation rules for Prolog. In *Partial Evaluation and Semantics-Based Program Manipulation, New Haven, Connecticut. (Sigplan Notices, vol. 26, no. 9)*, 1991.

[PP91b] Maurizio Proietti and Alberto Pettorossi. Unfolding - Definition - Folding, in this order, for avoiding unnecessary variables in logic programs. In *Proceedings of PLILP 91, Passau, Germany (LNCS 528)*, August 1991.

[Sek91] Hirohisa Seki. Unfold/fold transformations of stratified programs. *Theoretical Computer Science*, 86(1):107–139, 1991.

[Søn89] Harald Søndergaard. Semantics-based analysis and transformation of logic programs. Technical Report 89/22, DIKU, University of Copenhagen, Denmark, 1989.

[TS84] Hisao Tamaki and Taisuke Sate. Unfold/fold transformation of logic programs. In *Proceedings of 2nd International Logic Programming Conference, Uppsala*, pages 127–138, 1984.

[Tur86] Valentin F. Turchin. The concept of a supercompiler. *ACM Transactions on Programming Languages and Systems*, 8(3):292–325, July 1986.

[Wad90] Philip Wadler. Deforestation: Transforming programs to eliminate trees. *Theoretical Computer Science*, 73:231–248, 1990.