TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

On the asynchronous nature of communication in concurrent logic languages : a fully abstract model based on sequences

Citation for published version (APA):
Boer, de, F. S., & Palamidessi, C. (1990). *On the asynchronous nature of communication in concurrent logic languages : a fully abstract model based on sequences*. (Computing science notes; Vol. 9017). Technische Universiteit Eindhoven.

Document status and date:
Published: 01/01/1990

Document Version:
Publisher's PDF, also known as Version of Record (includes final page, issue and volume numbers)

Please check the document version of this publication:

• A submitted manuscript is the version of the article upon submission and before peer-review. There can be important differences between the submitted version and the official published version of record. People interested in the research are advised to contact the author for the final version of the publication, or visit the DOI to the publisher's website.
• The final author version and the galley proof are versions of the publication after peer review.
• The final published version features the final layout of the paper including the volume, issue and page numbers.

Link to publication

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
• You may freely distribute the URL identifying the publication in the public portal.

If the publication is distributed under the terms of Article 25fa of the Dutch Copyright Act, indicated by the "Taverne" license above, please follow below link for the End User Agreement:

www.tue.nl/taverne

Take down policy
If you believe that this document breaches copyright please contact us at:

openaccess@tue.nl

providing details and we will investigate your claim.

Download date: 05. Sep. 2026

On the asynchronous nature of communication in concurrent logic languages: a fully abstract model based on sequences

by

F.S. de Boer C. Palamidessi

90/17

October , 1990

COMPUTING SCIENCE NOTES

This is a series of notes of the Computing Science Section of the Department of Mathematics and Computing Science Eindhoven University of Technology.
Since many of these notes are preliminary versions or may be published elsewhere, they have a limited distribution only and are not for review.
Copies of these notes are available from the author or the editor.

Eindhoven University of Technology
Department of Mathematics and Computing Science
P.O. Box 513
5600 MB EINDHOVEN
The Netherlands
All rights reserved
Editors: prof.dr.M.Rem
prof.dr.K.M. van Hee

On the Asynchronous Nature of Communication in Concurrent Logic Languages: a Fully Abstract Model based on Sequences *

Frank S. de Boer[1] and Catuscia Palamidessi[2]

[1]Technische Universiteit Eindhoven,
P.O. Box 513, 5600 MB Eindhoven, The Netherlands

[2]Dipartimento di Informatica, Università di Pisa,
Corso Italia 40, 56125 Pisa, Italy

October 10, 1990

Abstract

The main contribution of this paper is to show that the nature of the communication mechanism of concurrent logic languages is essentially different from imperative concurrent languages. We show this by defining a compositional model based on sequences of input-output substitutions. This is to be contrasted with the compositionality in languages like CCS and TCSP, which requires more complicated structures, like trees and failure sets. Moreover, we prove that this model is fully abstract, namely that the information encoded by these sequences is necessary.

Regarding fully abstractness, our observation criterium consists of all the possible finite results, namely the computed answer substitution together with the termination mode (success, failure, or suspension). The operations we consider are parallel composition of goals and disjoint union of programs. We define a compositional operational semantics delivering sequences of input-output substitutions. Starting from this we obtain a fully abstract denotational semantics by requiring some closure conditions on sequences, that essentially model the monotonic nature of communication in concurrent logic languages. The correctness of this model is proved by refining the operational semantics in order to embody these closure conditions.

Key words and phrases: operational semantics, denotational semantics, concurrent logic languages, substitutions, sequences, compositionality fully abstractness.
1985 Mathematics Subject Classification: 68Q55, 68Q10.
1987 Computing Reviews Categories: D.1.3, D.3.1, F.1.2, F.3.2.

*Part of this work was supported by the ESPRIT BRA project "Intergration"

1 Introduction

Compositionality is considered one of the most desirable characteristics of a formal semantics, since it provides a foundation of program verification and modular design. The difficulty in obtaining this property depends upon the *operators* for composing programs, the behaviour we want to describe (*observables*), and the degree of *abstractness* we want to reach. A compositional model is called *fully abstract* (with respect to some operators and observables) if it identifies programs that behave in the same way under all the possible contexts. A fully abstract model can be considered to be *the* semantics of a language: all the other compositional semantics can be reduced to it by abstracting from the redundant information. Fully abstractness is important, for instance, for deciding correctness of program transformation techniques. If a fully abstract model distinguishes the transformed program from the original one then the transformation is not correct (in the sense that it does not preserve the same behaviour under composition).

In the field of logic languages there are basically two operators for composing programs: the conjunction of goals and the union of clauses. The observables are usually related to the finite result: success, failure, and computed answer substitutions. For Concurrent Logic Languages compositionality has been studied mainly with respect to the conjunction of goals, whilst union of clauses has been considered only in simple cases (union of *disjoint* programs [GCLS88], and union of *nicely intersecting* programs [GMS89]). This is rather natural since in a concurrent framework the main operation is the parallel composition of processes. On the other hand, the class of observables has to be enriched by *suspension* (or *deadlock*).

The main problem of compositionality in concurrent languages is the description of deadlock behaviour. For languages like CCS and TCSP it is well-known that sequences are not sufficient, and that, on the other hand, trees contain too much control information to be fully abstract. In order to abstract from redundant branching information encoded by the tree structure different approaches have been proposed. The most well known consist of defining an appropriate equivalence relation on trees (see, for instance, bisimulation [Mil80]), and of grouping the *branching information* in *failure sets* [BHR84]. In general, failure set semantics is more abstract than bisimulation and it is proved to be fully abstract in the case of TCSP.

Until now, with respect to compositionality, Concurrent Logic Languages have been regarded just as a particular case of the classic paradigms. Therefore, the problem has been approached by the standard methods. De Bakker and Kok ([dBK88], [Kok88]) and De Boer et. al. ([dBKPR89a], [dBKPR89b]) use tree-like structures labeled with functions on substitutions. More simple tree-like structures, labeled by constraints, are used by Gabbrielli and Levi ([GL90]) and by Saraswat and Rinard ([SR89]). In [GCLS88] and in [GMS89] the authors approach the problem of fully abstractness by refining the failure set semantics of TCSP.

Let us try to argue why we think that Concurrent Logic Languages require a completely different approach. The communication mechanism in Concurrent Logic Languages is based on the *production* and *consumption* of bindings (substitutions) on shared variables. We can translate a CCS process by interpreting the action a as the production of a binding on a variable x_a and the complementary action $\bar{a}$ (in CCS parallel processes synchronize on complementary actions) as the consumption of this binding. The main difference is that the behaviour of complementary actions is not symmetrical as it is in CCS and in TCSP. Indeed, the production of a binding can always proceed, whilst the consumption has to wait. In other terms, the communication mechanism of concurrent logic languages is intrinsically *asynchronous*. The following example shows that this leads to an essentially different deadlock behaviour [1].

Example 1.1 *Let* $p_1 ::= \bar{a}\bar{b}+\bar{a}\bar{c}+\bar{a}\bar{d}$ *and* $p_2 ::= \bar{a}\bar{b}+\bar{a}(\bar{c}+\bar{d})$. *The failure set semantics distinguishes these two processes. Indeed, in case of synchronous communication, they behave differently under the context* $p ::= a(b + c)$. *The process* p_1 *can deadlock, by choosing the third branch, whilst* p_2 *cannot. In the case of asynchronous communication however, both processes have the same*

[1] The term *deadlock* is used here with its classical meaning in the theory of concurrency. In concurrent logic programming, this kind of *deadlock* can correspond both to *failure* or to *suspension* (whilst, in the current terminology, it is associated only to *suspension*). In example 1.1 it correspond to *suspension*.

deadlock behaviour. The process p_2 can now deadlock by choosing the second branch, because p can independently decide to produce b (after a). In the formalism of concurrent logic languages, this example can be translated as follows (we use here the syntax of [GMS89]. $ask(t = u)$ represents the consumption of a substitution satisfying the equation $t = u$, whilst $tell(t = u)$ represents the production of the most general substitution satisfying $t = u$, and $|$ is the commit operator). Figure 1 illustrates this example.

$$\{\ p_1(x_a, x_b, x_c, x_d) : -ask(x_a = a) \mid ask(x_b = b)$$
$$\quad p_1(x_a, x_b, x_c, x_d) : -ask(x_a = a) \mid ask(x_c = c)$$
$$\quad p_1(x_a, x_b, x_c, x_d) : -ask(x_a = a) \mid ask(x_d = d)\ \}$$

$$\{\ p_2(x_a, x_b, x_c, x_d) : -ask(x_a = a) \mid ask(x_b = b)$$
$$\quad p_2(x_a, x_b, x_c, x_d) : -ask(x_a = a) \mid p_3(x_c, x_d)$$
$$\quad p_3(x_c, x_d) : -ask(x_c = c) \mid$$
$$\quad p_3(x_c, x_d) : -ask(x_d = d) \mid \qquad \}$$

$$\{\ p(x_a, x_b, x_c, x_d) : -tell(x_a = a) \mid p'(x_b, x_c)$$
$$\quad p'(x_b, x_c) : -tell(x_b = b) \mid$$
$$\quad p'(x_b, x_c) : -tell(x_c = c) \mid \qquad \}$$

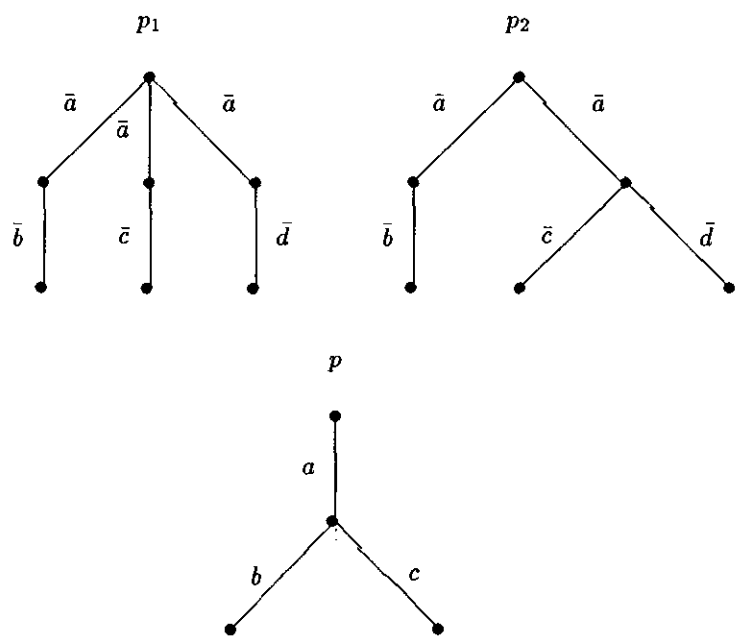

Figure 1: In logic programming p_1 and p_2 cannot be distinguished by p.

This example indicates that in the asynchronous case the failure set semantics (at least as it is defined in [BHR84]) is not abstract enough. The essence of this redundancy relies in the following observation:

Example 1.2 *In the asynchronous case, $p_1 ::= a(b+c)$ is equivalent to $p_2 ::= ab + ac$ under every context. This is due to the fact that the choice present in p_1 does not depend upon the environment. After the production of a, p_1 can proceed to produce either b or c in the same way as p_2.*

This example may induce one to believe that simple sequences of bindings are sufficient for obtaining compositionality. However this is not true in general, because of the different behaviour of the complementary (consuming) case. Consider the following example:

Example 1.3 *The process* $p_1 ::= \bar{a}(\bar{b}+\bar{c})$ *is not equivalent to* $p_2 ::= \bar{a}\bar{b}+\bar{a}\bar{c}$. *They are distinguished by the context* $p := ab$ *(*p_2 *can deadlock whilst* p_1 *cannot). This is due to the fact that the choice present in* p_1 *does depend upon the environment. This dependency is, however, of a different nature from the synchronous case. Here it is related only to the past behaviour of the environment, i.e., to the bindings that have already been produced.*

This last remark indicates a possible way to solve the problem. Given a sequence of substitutions representing the computation of a process with respect to an arbitrary environment, we add the information about who is the producer of each substitution, the process or its environment. If the substitution obtained from such a sequence does not provide the process with the necessary information to proceed then it will deadlock *assuming* that the environment does not produce any bindings anymore. The composition of different processes then simply amounts to verifying that the assumptions made by one process about its environment are indeed validated by the other processes.

We will show that we can describe compositionally the behaviour of a process by means of a Plotkin-like transition system, labeled by input-output substitutions. The behaviour of the possible environments are modeled by transitions labeled with input substitutions. This kind of transition step does not occur in the usual transition system for CCS, and it allows here to obtain directly a compositional operational semantics based on (sets of) sequences. These sequences are essentially different from the *scenarios* of [Sar85], where input substitutions correspond to assumptions about the environment which are necessary for the process to proceed. As a consequence, compositionality is there obtained only for the success set. The input-output sequences we use have been introduced in [GCLS88] as one component of the domain of the denotational semantics, the other ingredient being the suspension set. Because of what is stated above, this first component would have already been sufficient to define there a compositional (hence denotational) semantics.

The language described in [GCLS88] contains non-monotonic test predicates. However the real intricacies of the asynchronous and declarative nature of communication in logic languages come to surface in the monotonic case. Here even the sequences contain too much information about the particular order of production of bindings, information that cannot be sensed by monotonic contexts. This is mainly due to the fact that monotonic contexts cannot be specified to ask (only) a specific amount of bindings, they can always proceed when more bindings are provided. We will define therefore a refinement of the transition system, based on some additional steps that allow to abstract from the particular order of productions by buffering them in a kind of *active* store. Active is meant here in the sense that it can produce bindings itself. The monotonic case has also been investigated in [GMS89]. However the model presented in that paper is based on a refinement of failure set semantics, via some equivalence relation based on simulation of sequences.

In our paper we give a fully abstract semantics for a class of Flat Concurrent Logic Languages. This class will be denoted by HC($\mathcal{A},\mathcal{T}$), where HC stands for Horn Clauses and $\mathcal{A},\mathcal{T}$ are parameters which denote the set of primitives that can occur in the guards. We consider the following possibilities: $\mathcal{A} = \emptyset$ or $\mathcal{A} = Ask$, where

$$Ask = \{ask(E) \mid E \textit{ is a set of equations in the Herbrand universe}\}$$

and $\mathcal{T} = \emptyset$ or $\mathcal{T} = Tell$, where

$$Tell = \{tell(E) \mid E \textit{ is a set of equations in the Herbrand universe}\}.$$

This class includes, for instance, Oc [Hir87] (HC($\emptyset,\emptyset$)), Flat GHC [Ued88] (HC($Ask,\emptyset$)), ccH [GL90] and the language of [GMS89](HC(Ask, $Tell$)), that is a refined version of Flat CP [Sha89]. HC($\mathcal{A},\mathcal{T}$) can be seen as a particular instance of the cc/Herbrand framework [SR89]. For example, HC($Ask,\emptyset$) corresponds to Eventual Herbrand and HC(Ask, $Tell$) corresponds to atomic Herbrand.

We are mainly concerned with the above three instances of HC($\mathcal{A},\mathcal{T}$). Observe that HC($\emptyset,\emptyset$) is included in HC($Ask,\emptyset$) and HC($Ask,\emptyset$) is included in HC(Ask, $Tell$). A model that is compositional for a language is compositional also for the sublanguages, but not vice-versa. On the other side, with respect to fully abstractness the situation is just the reverse. Therefore each case must be

considered separately. However we show that the fully abstract compositional model is the same for all these languages.

The plan of the paper is the following: In the next section we introduce some basic notions. In section 3 we give the syntax and an informal operational semantics of the class of languages we study. Then we present a compositional semantics based on a Plotkin-like labeled transition system. In the last section we present a refinement of this semantics which is fully abstract.

2 Preliminaries

In this section we briefly recall some basic notions and results about substitutions and unification. We will use (mainly) the terminology of [Apt88], [Llo87], [LMM88], and [Ede85], to which we refer for a more detailed presentation of these topics.

2.1 Substitutions

Let *Var* be a non empty set of *variables*, with typical elements $x, y, z, \ldots$. Let *Cons* be a set of *constructors*, with typical elements $a, b, c, \ldots$ (constructors with 0 arguments, or *constants*), and $f, g, h, \ldots$ (constructors with one or more arguments). We assume the presence of at least one constant.

The set *Term*, with typical elements $t, u, \ldots$, is the set of all the *terms* built on *Var* and on *Cons*. Examples of terms are $f(a), f(x), g(f(a), f(x)), \ldots$. The set of variables occurring in t is denoted by $var(t)$.

The set of *substitutions*, *Subst*, with typical elements $\vartheta, \sigma, \ldots$, consists of all the mappings ϑ from *Var* into *Term* such that the domain of ϑ, $dom(\vartheta) = \{x \in Var \mid \vartheta(x) \neq x\}$, is finite. We will use the set-theoretic notation $\{x/t \mid x \in dom(\vartheta) \wedge \vartheta(x) \equiv t\}$ to represent ϑ.

The *application* $t\vartheta$ of ϑ to t is defined as the term obtained by replacing each variable x in t by $\vartheta(x)$. The set $ran(\vartheta)$ (range of ϑ) is defined as $\bigcup_{x \in dom(\vartheta)} var(\vartheta(x))$. The substitution ϑ is *ground* iff $ran(\vartheta) = \emptyset$.

The *composition* $\vartheta\vartheta'$ of ϑ and ϑ' is defined by $(\vartheta\vartheta')(x) \equiv (\vartheta(x))\vartheta'$. We recall that the composition is associative, the empty substitution ϵ is the neutral element, and, for each term t, $t(\vartheta\vartheta') \equiv (t\vartheta)\vartheta'$ holds.

The restriction $\vartheta_{|V}$ of ϑ to a set of variables V is the substitution $\vartheta_{|V}(x) = \vartheta(x)$ for $x \in V$ and $\vartheta_{|V}(x) = x$ otherwise. We will abbreviate $\vartheta_{|var(\bar{A})}$ to $\vartheta_{|\bar{A}}$.

2.2 Unification

The notion of *unification* can be given, equivalently, either with respect to a set of sets of terms [Ede85, Llo87], or with respect to a set of equations [LMM88, Apt88]. We choose the second approach.

An *equation* is an expression of the form $t = u$, where t and u are terms. The set of sets of equations, with typical element E, will be denoted by *Eqn*. A set of equations E is *unifiable* iff there exists ϑ such that for all $t = u$ in E, $t\vartheta \equiv u\vartheta$ holds, where $\equiv$ is the syntactic identity on terms. ϑ is then called an *unifier* of E. The set of all the unifiers of E will be denoted by $unif(E)$. The set of all the *most general unifiers* of E is given by $mgu(E) = \{\vartheta \in unif(E) \mid \forall \vartheta' \in unif(E).\ \vartheta \leq \vartheta'\}$. It is well known that all the most general unifiers of a set E are equivalent, and this explains why, in the literature, we often find the terminology "*the* mgu of E".

Given a substitution ϑ the associated set of equations will be denoted by $\mathcal{E}(\vartheta)$.

2.3 Constraints on the Herbrand Universe

A constraint system is essentially a system of partial information that supports the notion of consistency and entailment. The notion of *constraint* was introduced in Logic Programming by Jaffar and Lassez [JL87]. Maher [Mah87] suggested the use of constraints to model logically the synchronisation and communication mechanisms of concurrent logic languages. We restrict here to a particular class of constraints: the *ask* and *tell* constraints on the Herbrand universe [Sar89]. They are of the form $ask(E)$ and $tell(E)$ respectively, where E is a set of equations. A constraint of the form $ask(E)$ or $tell(E)$ is *solved* by a substitution ϑ if $\vartheta = mgu(E)$.

We use the following notations: $Ask = \{ask(E) \mid E \in Eqn\}$ and $Tell = \{tell(E) \mid E \in Eqn\}$.

3 The class HC($\mathcal{A}, \mathcal{T}$). Syntax and informal operational semantics.

The (parametric) sets $\mathcal{A}$ and $\mathcal{T}$ specify the set of *primitives* used in the guards. We restrict here to the following cases: $\mathcal{A} = \emptyset$ or $\mathcal{A} = Ask$, and $\mathcal{T} = \emptyset$ or $\mathcal{T} = Tell$.

Let *Pred* be a set of *predicates*, with typical elements $p, q, r, \ldots$. The set *Atom*, with typical elements $A, B, H \ldots$, is the set of all the *atomic formulas* built on *Pred* and on *Term*, and of the primitives of the form $tell(E)$. In the following, we use the abbreviations $\bar{A}, \bar{x}$ etc. to denote conjunctions of atoms, variables etc.

A *clause* C of HC($\mathcal{A}, \mathcal{T}$) is a formula of the form $H \leftarrow g_1 : g_2 \mid \bar{B}$, where $g_1 \in \mathcal{A}$ and $g_2 \in \mathcal{T}$. The atom H is the head, $g_1 : g_2$ is the guard and $\bar{B}$, a multiset of atoms, is the body of the clause. A *goal statement* G is a multiset of atoms $\bar{A}$. The empty goal statement is denoted by $\Box$. A HC($\mathcal{A}, \mathcal{T}$) program P is a (finite) set of clauses. An atom A in a goal G is seen as an (AND-) process. If A is of the form $tell(E)$ then:

- if E is unifiable then A *succeeds* producing the substitution $\vartheta = mgu(E)$, and the remaining goal $G \setminus A$ is instantiated by ϑ.
- if E is not unifiable then A and the whole goal *fails*

If A is an atomic formula, its computation proceeds by looking for a candidate clause in W.

Definition 3.1 *Let A be an atomic formula and let $C \equiv H \leftarrow g_1 : g_2 \mid \bar{B}$ be a clause in P, renamed in order to have no variables in common with A. Then the clause C*

- is candidate *if*
  1. $\exists mgu(H = A) = \vartheta$,
  2. $\vartheta_{|A} = \epsilon$
  3. $\exists \vartheta_1$ *such that* $g_1\vartheta$ *is solved by* ϑ_1,
  4. $\vartheta_{1|A} = \epsilon$,
  5. $\exists \vartheta_2$ *such that* $g_2\sigma$ *is solved by* ϑ_2, *where* $\sigma = \vartheta\vartheta_1$.
- suspends *if (1) is satisfied but (2) is not satisfied, or (1), (2) and (3) are satisfied but (4) is not satisfied.*
- fails *in all the other cases.*

If there are candidate clauses, then the computation of A *commits* to one of them (i.e. no backtracking will take place), A is replaced by $\bar{B}\sigma\vartheta_2$, and the rest of the goal is instantiated by ϑ_2. If all the clauses for A fail, then A and the whole goal fail. If no clauses are candidate and at least one clause suspends, then A *suspends*. A can be *resumed* when its arguments get bound by other processes in the goal. If all the processes in the goal suspend, then the goal suspends.

4 A compositional operational semantics for HC($\mathcal{A}, \mathcal{T}$).

To define the meaning of a goal in terms of its subgoals, we describe the behaviour of a goal as a sequence of *interactions* with its environment. Interactions are modeled as *input/output* substitutions. An input substitution is provided by the environment, whereas an output substitution is produced by the goal itself.

Definition 4.1

- *The set of input substitutions is* $Subst_I = \{\vartheta^I \mid \vartheta \in Subst\}$.
- *The set of output substitutions is* $Subst_O = \{\vartheta^O \mid \vartheta \in Subst\}$.
- *The set of input/output substitutions, with typical element* ϑ^ι, *is* $Subst_{IO} = Subst_I \cup Subst_O$.

The operational semantics we define is based on a transition system T labeled on $Subst_{IO}$. The *configurations* of T are pairs consisting of a goal and an infinite set of fresh variables (representing the variables still available for the renaming mechanism). To obtain a compositional operational semantics we need a compositional renaming mechanism. We propose the following (formal) solution. Let $\mathcal{P}_i(Var)$ be the set of all the infinite sets of variables. We assume the existence of a *partitioning* function $Part : \mathcal{P}_i(Var) \rightarrow \mathcal{P}_i(Var) \times \mathcal{P}_i(Var)$ such that

$$\forall V \in \mathcal{P}_i(Var)\ [\langle V_1, V_2\rangle = Part(V) \Rightarrow (V_1 \cup V_2 \subseteq V\ \wedge\ V_1 \cap V_2 = \emptyset)]$$

and, moreover, we assume $Part(V \setminus Z) = (V_1 \setminus Z, V_2 \setminus Z)$, where $Part(V) = (V_1, V_2)$. In this way we can split the "available variables" among the subgoals so to avoid clash of variables among the subcomputations.

Table 1 gives the rules for T describing the "successful" computation steps. We call them *computation rules.*

Table 1: The Transition System T. Computation Rules

C1	$\langle A, V\rangle \xrightarrow{\vartheta_2^O} \langle \bar{B}\sigma\vartheta_2, W\rangle$ **if** $\exists C \in P_V$ [C is candidate] $\wedge\ W = V \setminus var(\bar{B}\sigma\vartheta_2)$
C2	$\langle tell(E), V\rangle \xrightarrow{\vartheta^O} \langle \Box, V\rangle$ **if** $\exists\vartheta = mgu(E)$
C3	$\langle A, V\rangle \xrightarrow{\vartheta^I} \langle A\vartheta, V\rangle$ **where** A is either an atomic formula or $tell(E)$
C4	$\dfrac{\langle \bar{A}_1, V_1\rangle \xrightarrow{\vartheta^\iota} \langle \bar{B}_1, W_1\rangle \quad \langle \bar{A}_2, V_2\rangle \xrightarrow{\vartheta^\iota} \langle \bar{B}_2, W_2\rangle}{\langle \bar{A}_1, \bar{A}_2, V\rangle \xrightarrow{\vartheta^\iota} \langle \bar{B}_1, \bar{B}_2, W_1 \cup W_2\rangle}$ **where** $\langle V_1, V_2\rangle = Part(V)$

The rule **C1** describes the normal atomic reduction step. The symbols $C, \bar{B}, \sigma$, and ϑ_2 are the ones of definition 3.1, whilst P_V denotes the program P renamed with the variables in V.

The rule **C2** describes the output of the substitution satisfying an atom of the form $tell(E)$. Note that in the case of HC($\mathcal{A}, \emptyset$) this is the only rule that allows to produce bindings on the (shared) variables of the goal.

The rule **C3** describes the input of a substitution produced by the external environment. Note that ϑ is any substitution, namely it represent a *free assumption* on the environment. In other words, it does not depend upon what the goal needs to proceed, and whether or not it will fail after this input. This is the point in which our transition system essentially differs from the semantics

described in [Sar85] and in [GMS89], and this is why our sequences contain information enough to allow for compositionality.

Finally, **C4** describes the transition of a goal in terms of the transitions of the subgoals. Note that this is the rule that allows to check that the input assumptions really correspond to the outputs done by the environment.

Table 2 and table 3 illustrate the rules for failure and suspension respectively. We need to introduce in our configurations the symbols *fail* and *susp*, with the obvious meaning.

Table 2: The Transition System T. Failure Rules

F1	$\langle A, V\rangle \xrightarrow{\epsilon^O} \langle fail, V\rangle$ **if** $\forall C \in P_V$ [C fails]
F2	$\langle tell(E), V\rangle \xrightarrow{\epsilon^O} \langle fail, V\rangle$ **if** $\nexists mgu(E)$
F3	$\dfrac{\langle A, V\rangle \xrightarrow{\epsilon^O} \langle fail, V\rangle}{\langle A, \bar{B}, V \xrightarrow{\epsilon^O} \langle fail, V\rangle}$ **where** A is either an atomic formula or $tell(E)$

Table 3: The Transition System T. Suspension Rules

S1	$\langle A, V\rangle \xrightarrow{\epsilon^O} \langle susp, V\rangle$ **if** $\exists C \in P_V$ [C suspends] $\wedge$ $\forall C \in P_V$ [C is not candidate]
S2	$\dfrac{\langle \bar{A}_1, V_1\rangle \xrightarrow{\epsilon^O} \langle susp, V_1\rangle \quad \langle \bar{A}_2, V_2\rangle \xrightarrow{\epsilon^O} \langle susp, V_2\rangle}{\langle \bar{A}_1, \bar{A}_2, V\rangle \xrightarrow{\epsilon^O} \langle susp, V_1 \cup V_2\rangle}$ **where** $\langle V_1, V_2\rangle = Part(V)$

The operational semantics $\mathcal{O}$ based on this transition system T delivers sets of sequences s of input/output substitutions, ended by a *termination mode*. We denote the set of these sequences as $Seq = Subst_{IO}{}^*.\{ss, ff, dd, \bot\}$. The set $Subst_{IO}{}^*$, with typical element c, denotes the sequences of substitutions generated during the computation, whilst the symbols ss, ff, and dd represent the possible ways in which a process can terminate: *success*, *failure* and *suspension* (or *deadlock*) respectively. Sequences ending in $\bot$ will denote *unfinished* computations. The symbol α will denote an element ranging over the set $\{ss, ff, dd, \bot\}$.

We need first to introduce a technical notion: the *restriction operator*. Intuitively, this operator restricts the substitutions in the sequences generated by a goal A to only those bindings which affect the variables of A. This is not really necessary to achieve compositionality here, but it is a step towards fully abstractness, since, intuitively, we must identify all those processes that only differ for the assignments to the local variables (no contexts can distinguish them). Moreover, if V is the set of local variables available for the computation of A, we must abstract from all those sequences that are of the form $c.\vartheta^I.s$ such that ϑ introduces a variable in V that has not yet been introduced by c.

Definition 4.2 (Restriction operator) *Let $\bar{A}$ be a goal and let $V \in \mathcal{P}_i(Var)$. Then $r_{\bar{A},V} : \mathcal{P}(Seq) \to \mathcal{P}(Seq)$ is defined by*

$$r_{\bar{A},V}(S) \quad = \quad \{Restrict_{\bar{A}}(s) \mid s \in S \ \wedge \ [s = c.\vartheta^I.s' \Rightarrow var(\vartheta) \cap V \subseteq var(A\vartheta_c)]\}$$

where ϑ_c denotes the substitution obtained by composing all the elements of c: $\vartheta_\lambda = \epsilon$ and $\vartheta_{\sigma.c} = \sigma\vartheta_c$.

and

$$\begin{array}{lcl} Restrict_{\bar{A}}(\alpha) & = & \alpha \\ Restrict_{\bar{A}}(\vartheta^I.s) & = & \vartheta^I.Restrict_{\bar{A}\vartheta}(s) \\ Restrict_{\bar{A}}(\vartheta^O.s) & = & \begin{cases} \vartheta^O_{|\bar{A}}.Restrict_{\bar{A}\vartheta}(s) & \text{if } \vartheta_{|\bar{A}} \neq \epsilon \\ Restrict_{\bar{A}}(s) & \text{otherwise} \end{cases} \end{array}$$

We can now define the operational semantics.

Definition 4.3 (Operational Semantics) *The operational semantics $\mathcal{O} : Goal \rightarrow [\mathcal{P}_i(Var) \rightarrow Seq]$ is given by*

$$\begin{array}{lcl} \mathcal{O}[\![\Box]\!](V) & = & \{c.\alpha : \ \alpha = \bot, ss, \text{and } c \text{ contains only input substitutions}\} \\ \mathcal{O}[\![fail]\!](V) & = & \{\bot, ff\} \\ \mathcal{O}[\![susp]\!](V) & = & \{\bot, dd\} \\ \mathcal{O}[\![\bar{A}]\!](V) & = & r_{\bar{A},V}(\{\vartheta^t.\mathcal{O}[\![\bar{B}]\!](W) \mid (\bar{A}, V) \xrightarrow{\vartheta^t} (\bar{B}, W)\} \cup \{\bot\}) \end{array}$$

Note that this definition is recursive. We consider $\mathcal{O}$ as the least fixpoint of the transformation (continous on the domain $\mathcal{P}_\bot(Seq)$, the set of subsets of Seq containing $\bot$, ordered with respect to set inclusion) associated with its defining equations.

From this operational semantics we obtain our observation criterium as follows:

Definition 4.4 (Observables)

$$Obs(\bar{A}) = Result_{\bar{A}}(\mathcal{O}[\![\bar{A}]\!](V))$$

where V is the set $Var \setminus var(A)$, and

$$Result_{\bar{A}}(S) = \{\langle \vartheta_{c|\bar{A}}, \alpha \rangle \mid \ c.\alpha \in S \ \wedge \ c \in Subst^*_O \ \wedge \ \alpha \in \{ss, ff, dd\}\}.$$

In the following, when $V = Var \setminus var(A)$, we will write simply $\mathcal{O}[\![\bar{A}]\!]$ instead of $\mathcal{O}[\![\bar{A}]\!](V)$.

Note that in this definition we pick up the sequences entirely composed by output substitutions. This amounts to require that each substitution we observe has been really produced. It is easy to see that this notion of observables, based on T, correspond to the set of *finite results* (computed answeer substitution, termination mode) that can be derived by the classical operational semantics described in section 3.

To show the compositionality of the operational semantics we define the *parallel composition* $||$. This operator, first introduced in [GCLS88], allows to combine sequences of input/output substitutions that are equal at each point, apart from the labels, so modeling the interaction of a process with its environment. It corresponds to apply iteratively the rule **C4** of the transition system T.

Definition 4.5 (Parallel composition operator) *The* partial *operator $||$: $Seq \times Seq \rightarrow Seq$ is defined by*

- $s_1 \,||\, s_2 = s_2 \,||\, s_1$
- $\alpha \,||\, ss = \alpha$
- $\alpha \,||\, ff = ff$
- $dd \,||\, dd = dd, \ \bot || \ dd = \bot$

- $\vartheta^l.s_1 \parallel \vartheta^\ell.s_2 = \vartheta^\ell.(s_1 \parallel s_2)$

The natural extension of $\parallel$ *on sets will be still denoted by* $\parallel$.

The following result shows the compositionality of our operational semantics with respect to goal conjunction. The compositionality with respect to the union of disjoint programs (or nicely intersecting programs) is obviously given by the set union.

Theorem 4.6 (Compositionality of $\mathcal{O}$)

$$\mathcal{O}[\![\bar{A}_1, \bar{A}_2]\!](V) = \mathcal{O}[\![\bar{A}_1]\!](V_1) \parallel \mathcal{O}[\![\bar{A}_2]\!](V_2), \quad \textit{where } \langle V_1, V_2\rangle = \textit{Part}(V).$$

Proof
Notice that the parallel composition operator models consecutive applications of the rule **C4** of the transition system.

We end this section by noticing that $\mathcal{O}$ is also a *denotational semantics*, since it is compositional and it is defined as the fixpoint of an (higher order) operator.

5 Fully abstract semantics for the languages of the class HC($\mathcal{A}, \mathcal{T}$).

The operational semantics defined in the previous section is not fully abstract, this is due mainly to the fact that it enforces synchronization between the producer and the consumer. This synchronization consists in that a substitution produced by an atom has to be consumed at the same time by the other atoms of the goal. To abstract from this phenomenon we will define some closure conditions on sets of sequences of *elementary* substitutions. The fully abstract denotational semantics $\mathcal{D}$ then will be based on the transition system T plus these closure conditions.

We first introduce the notion of an elementary substitution.

Definition 5.1 *We denote by ESubst the set of all the elementary substitutions, namely the ones of the form* $\{x/t\}$. *Moreover we define*

- $ESubst_I = \{\vartheta^I \mid \vartheta \in ESubst\}$
- $ESubst_O = \{\vartheta^O \mid \vartheta \in ESubst\}$
- $ESubst_{IO} = ESubst_I \cup ESubst_O$
- $ESeq = ESubst^*_{IO}.\{ss, ff, dd, \bot\}$

To decompose a substitution into elementary ones we define

Definition 5.2 *The function* decomposition, $Dec : Subst_{IO} \rightarrow ESubst^*_{IO}$, *is defined as follows*

$$Dec(\{x_1/t_1, \ldots, x_n/t_n\}^\ell) = \{x_1/t_1\}^\ell \ldots \{x_n/t_n\}^\ell$$

We next give the definition of the closure conditions.

Definition 5.3 *Given an initial goal* $\bar{A}$, *a set of local variables* V, *we define* $Closure_{\bar{A},V}(S)$ *to be the minimal set that contains* S *and that is closed with respect to the conditions* **P1**-**P6** *of table 4.*

Table 4: The closure conditions with respect to an initial goal $\bar{A}$ and set of local variables V

P1	$c_1.\vartheta^O.s \in S \Rightarrow c_1.\vartheta^I.s \in S$
P2	$c_1.\vartheta^O_1.\vartheta_2[\vartheta_1]^I.s \in S \Rightarrow c_1.\vartheta^I_2.\vartheta_1[\vartheta_2]^O.s \in S$
P3	$c_1.\vartheta^O_1.Dec(\vartheta^I_2).s \in S \Rightarrow c_1.\vartheta_1[mgu(\mathcal{E}(\vartheta_2))]^I.Dec(\vartheta^O_2).s \in S$
P4	$c_1.\vartheta_1[mgu(\mathcal{E}(\vartheta_2))]^O.Dec(\vartheta^I_2).s \in S \Rightarrow c_1.\vartheta^I_1.Dec(\vartheta^O_2).s \in S$
P5	$c \equiv c_1.\vartheta^\ell_1 \ldots \vartheta^\ell_m.s \in S \Rightarrow c_1.\sigma^\ell_1 \ldots \sigma^\ell_n.s \in S$, where $Z_1 \cap var(c_1) = \emptyset$, $Z_2 \cap var(c) = \emptyset$, $(\vartheta_1 \cdots \vartheta_m)_{Var \setminus Z_1} = (\sigma_1 \cdots \sigma_n)_{Var \setminus Z_2}$, for $\ell = I$: Z_1 and Z_2 have no variables in common with V or $\bar{A}$, for $\ell = O$: $Z_1, Z_2 \subseteq V$
P6	$c.\alpha \in S \Rightarrow c.\ \bot \in S$ where $\alpha \in \{ss, ff, dd\}$

Here $\vartheta_1[\vartheta_2]$ denotes the substitution $\{x/t\vartheta_2 : x/t \in \vartheta_1\}$. With respect to the rules **P3** and **P4** we require that $dom(mgu(\mathcal{E}(\vartheta_2))) \subseteq ran(\vartheta_1)$. The first condition expresses that there is no

direction involved in the communication of substitutions, that is, a substitution produced can also be consumed. The condition **P2** expresses that a input substitution received after the production of a substitution can also be received before. The following condition states that a input substitution preceded by a output substitution can be produced when the environment provides the instantiated output substitution. On the other hand the condition **P4** expresses that when the environment provides less information than the process is able to produce the additional information then can be produced by the process itself. Condition **P5** states that successive input (output) substitutions can be replaced by a equivalent sequence of substitutions. This equivalence is defined in such a way that local variables can be added and deleted. Here is the definition of the denotational semantics $\mathcal{D}$.

Definition 5.4 (The denotational semantics)

$$
\begin{array}{lcl}
\mathcal{D}[\![\Box]\!](V) & = & \{c.\alpha : \ \alpha = \perp, ss, \textit{and } c \textit{ contains only input substitutions}\} \\
\mathcal{D}[\![\mathit{fail}]\!](V) & = & \{\perp, ff\} \\
\mathcal{D}[\![\mathit{susp}]\!](V) & = & \{\perp, dd\} \\
\mathcal{D}[\![A]\!](V) & = & r_{A,V}(\mathit{Closure}_{A,V}(\{\mathit{Dec}(\vartheta^t).\mathcal{D}[\![\bar{B}]\!](W) \mid \langle A, V\rangle \xrightarrow{\vartheta^t} \langle \bar{B}, W\rangle\} \cup \{\perp\})) \\
\mathcal{D}[\![\bar{A}_1, \bar{A}_2]\!](V) & = & r_{\bar{A}_1,\bar{A}_2,V}(\mathit{Closure}_{\bar{A}_1,\bar{A}_2,V}(\mathcal{D}[\![\bar{A}_1]\!](V_1) \parallel \mathcal{D}[\![\bar{A}_2]\!](V_2))), \\
 & & \textit{where } \langle V_1, V_2\rangle = \mathit{Part}(V)
\end{array}
$$

6 The correctness of $\mathcal{D}$

The first question is now the correctness of $\mathcal{D}$, i.e., whether it distinguishes *enough*. In other words, if two processes that have the same denotational semantics give also the same observables. We show this by defining a transition system T', which adds to T some rules that model the closure conditions. Essentially T' describes the asynchronous communication by the use of a *store of bindings*. Substitutions produced are communicated to this store. The consumption of a substitution then consists of retrieving this substitution from the store. The store is modeled by adding to a goal constructs of the form $store(E)$, which is to be interpreted as that the equation E is present in the store.

Table 5: The Transition System T'. Additional computation Rules

C1	$\langle A, V\rangle \xrightarrow{\epsilon^O} \langle \bar{B}\sigma, store(E_2\sigma), W\rangle$	**if** $\exists C \in P_V[C \text{ is candidate}] \wedge W = V \setminus var(P_V)$
C2	$\langle tell(E), V\rangle \xrightarrow{\epsilon^O} \langle store(E), V\rangle$	
C5	$\langle p(t), V\rangle \xrightarrow{\epsilon} \langle p(z), store(z = t), V \setminus \{z\}\rangle$	**where** $z \in V$
C6	$\langle store(E_1), \ldots store(E_m), V\rangle \xrightarrow{\epsilon^O} \langle store(E'_1), \ldots store(E'_n), V \setminus Z\rangle$	**if** $[(E_1 \wedge \ldots E_m) \Leftrightarrow \exists Z\ (E'_1 \wedge \ldots E'_n)] \wedge Z \subseteq V$
C7	$\langle store(x = t), V\rangle \xrightarrow{\vartheta^O} \langle \Box, V\rangle$	**if** $\vartheta = mgu(x = t)$
C8	$\dfrac{\langle \bar{A}, V\rangle \xrightarrow{\vartheta^O} \langle \bar{B}, W\rangle}{\langle \bar{A}, V\rangle \xrightarrow{\epsilon^O} \langle store(\mathcal{E}(\vartheta)), \bar{B}, W\rangle}$	

With respect to the transition system T' we modify the rule **C1** so that the output substitution of the guard is added to the store. Rule **C2** is modified such that instead of producing the *mgu* of a set of equations of a *tell* construct these equations are added to the store. In the rule **C3** A can now also be of the form $store(E)$. It is important to note that the failure rules remain the same, as a consequence inconsistencies in the store do not lead to failure. The rule **C5** allows to add information about the terms occurring in the goal to the store. The rule **C6** describe how a store can resolve itself into a equivalent one. The rule **C7** describes the retrieval of a substitution from the store. The last rule describes the *partial consumption* of substitutions, i.e., it allows the retrieval of some information from the store by a single atom of a goal. We illustrate this by a simple example.

Example 6.1 *We show how to derive* $p(a), q(x), store(x = a)$ *from the goal* $p(x), q(x), tell(x = a)$. *(As it plays no role in this example we forget about the set of fresh variables.)*

1. $tell(x = a) \xrightarrow{\epsilon^O} store(x = a)$, *by* **C2**
2. $p(x), q(x), tell(x = a) \xrightarrow{\epsilon^O} p(x), q(x), store(x = a)$, *from 1 by* **C3** *and* **C4**
3. $store(x = a) \xrightarrow{\{x/a\}^O} \Box$, *by* **C7**

4. $p(x), store(x = a) \overset{\{x/a\}^O}{\rightarrow} p(a)$, *from 3 by* **C3** *and* **C4**

5. $p(x), store(x = a) \overset{\epsilon^O}{\rightarrow} p(a), store(x = a)$, *from 4 by* **C8**

6. $p(x), q(x), store(x = a) \overset{\epsilon^O}{\rightarrow} p(a), q(x), store(x = a)$, *from 5 by* **C3** *and* **C4**.

This partial consumption may lead to additional failures, but these occur as inconsistencies in the store, and are as such not visible in the final result.

Given this transition system T' we define $\mathcal{O}'$ a follows:

Definition 6.2 *We define*

$$\mathcal{O}'[\![\bar{A}]\!](V) = r_{\bar{A},V}(\{\vartheta^\ell.\mathcal{O}'[\![\bar{B}]\!](W) : \langle \bar{A}, V\rangle \overset{\vartheta^\ell}{\rightarrow} \langle \bar{B}, W\rangle\} \cup \{\bot : \bar{A} \subseteq Atoms \cup Tell\})$$

where the transtition relation is defined with respect to T'.

To prove the correctness of D we show that

$$Dec(\mathcal{O}[\![\bar{A}]\!](V)) \subseteq \mathcal{D}[\![\bar{A}]\!](V) \subseteq \mathcal{O}'[\![\bar{A}]\!](V)$$

and

$$Result(\mathcal{O}[\![\bar{A}]\!](V)) = Result(\mathcal{O}'[\![\bar{A}]\!](V))$$

Therefore we obtain (by monotonicity of the operator *Result*)

Theorem 6.3 (Correctness of $\mathcal{D}$) $Result_{\bar{A}}(\mathcal{D}[\![\bar{A}]\!](V)) = Obs(\bar{A})$, *with* $V = Var \setminus var(\bar{A})$

We start with the proposition

Proposition 6.4 *For every goal* $\bar{A}$ *and set* V *we have*

$$Dec(\mathcal{O}[\![\bar{A}]\!](V)) \subseteq \mathcal{D}[\![\bar{A}]\!](V)$$

Proof

Let Φ be a continous operator with its least fixed point $Dec \circ \mathcal{O}$ ($Dec \circ \mathcal{O}[\![\bar{A}]\!](V) = Dec(\mathcal{O}[\![\bar{A}]\!](V))$). It suffices to prove that $\Phi(\mathcal{D}) \subseteq \mathcal{D}$. We proceed by induction on the complexity of the goal $\bar{A}$.

$$\Phi(\mathcal{D})[\![A]\!](V) =$$

$$Dec(r_{A,V}(\{\vartheta^\ell.\mathcal{D}[\![\bar{B}]\!](W) : \langle A, V\rangle \overset{\vartheta^\ell}{\rightarrow} \langle \bar{B}, W\rangle\} \cup \{\bot\})) =$$

$$r_{A,V}(\{Dec(\vartheta^\ell).\mathcal{D}[\![\bar{B}]\!](W) : \langle A, V\rangle \overset{\vartheta^\ell}{\rightarrow} \langle \bar{B}, W\rangle\} \cup \{\bot\}) \subseteq$$

$$r_{A,V}(Closure_{A,V}(\{Dec(\vartheta^\ell).\mathcal{D}[\![\bar{B}]\!](W) : \langle A, V\rangle \overset{\vartheta^\ell}{\rightarrow} \langle \bar{B}, W\}\} \cup \{\bot\})) =$$

$$\mathcal{D}[\![A]\!](V)$$

And, for $\bar{A} = \bar{A}_1, \bar{A}_2$:

$$\Phi(\mathcal{D})[\![\bar{A}]\!](V) =$$

$$Dec(r_{\bar{A},V}(\{\vartheta^\ell.\mathcal{D}[\![\bar{B}]\!](W) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\})) =$$

$$r_{\bar{A},V}(\{Dec(\vartheta^\ell).\mathcal{D}[\![\bar{B}]\!](W) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\}) =$$

$$r_{\bar{A},V}(\{Dec(\vartheta^\ell).r_{\bar{B},W}(Closure_{\bar{B},W}(\mathcal{D}[\![\bar{B}_1]\!](W_1) \parallel \mathcal{D}[\![\bar{B}_2]\!](W_2))) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\}) =$$

$$r_{\bar{A},V}(\{Dec(\vartheta^\ell).Closure_{\bar{B},W}(\mathcal{D}[\![\bar{B}_1]\!](W_1) \parallel \mathcal{D}[\![\bar{B}_2]\!](W_2)) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\}) \subseteq$$

$$r_{\bar{A},V}(Closure_{\bar{A},V}(\{Dec(\vartheta^\ell).\mathcal{D}[\![\bar{B}_1]\!](W_1) \parallel \mathcal{D}[\![\bar{B}_2]\!](W_2) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\})) =$$

$$r_{\bar{A},V}(Closure_{\bar{A},V}(r_{\bar{A},V}(\{Dec(\vartheta^\ell).\mathcal{D}[\![\bar{B}_1]\!](W_1) \parallel \mathcal{D}[\![\bar{B}_2]\!](W_2) : \langle \bar{A}, V\rangle \xrightarrow{\vartheta^\ell} \langle \bar{B}, W\rangle\} \cup \{\bot\}))) =$$

$$r_{\bar{A},V}(Closure_{\bar{A},V}\begin{pmatrix} r_{\bar{A}_1,V_1}(\{Dec(\vartheta^{\ell_1}).\mathcal{D}[\![\bar{B}_1]\!](W_1) : \langle \bar{A}_1, V_1\rangle \xrightarrow{\vartheta^{\ell_1}} \langle \bar{B}_1, W_1\rangle\} \cup \{\bot\}) \\ \parallel \\ r_{\bar{A}_2,V_2}(\{Dec(\vartheta^{\ell_2}.\mathcal{D}[\![\bar{B}_2]\!](W_2) : \langle \bar{A}_2, V_2\rangle \xrightarrow{\vartheta^{\ell_2}} \langle \bar{B}_2, W_2\rangle\} \cup \{\bot\}) \end{pmatrix}) \subseteq$$

$$r_{\bar{A},V}(Closure_{\bar{A},V}(\mathcal{D}[\![\bar{A}_1]\!](V_1) \parallel \mathcal{D}[\![\bar{A}_2]\!](V_2))) =$$

$$\mathcal{D}[\![\bar{A}]\!](V)$$

Here $\bar{B}_1, \bar{B}_2 = \bar{B}$ and $(W_1, W_2) = Part(W)$. Some remarks are in order here:

1. The fourth equality holds because the operation $r_{\bar{B},W}$ occurs in the scope of $r_{\bar{A},V}$ and as such it is not so difficult to see that it can be deleted.
2. The first inclusion holds because $W \subseteq V$ and $V \setminus W = var(\bar{B}) \setminus var(\bar{A})$.
3. The fifth equality holds because of the property $r_{\bar{A},V}(Closure_{\bar{A},V}(X)) = r_{\bar{A},V}(Closure_{\bar{A},V}(r_{\bar{A},V}(X)))$, X arbitrary, which we leave the reader to verify.
4. The sixth equality holds because $Part(V \setminus Z) = (V_1 \setminus Z, V_2 \setminus Z)$, with $(V_1, V_2) = Part(V)$. Furthermore we note that $r_{\bar{A}_1,\bar{A}_2,V}$ can be splitted into $r_{\bar{A},V_1}$ and $r_{\bar{A}_2,V_2}$ due to the renaming mechanism.

To prove that for every goal $\bar{A}$ and set V we have that $\mathcal{D}[\![\bar{A}]\!](V) \subseteq \mathcal{O}'[\![\bar{A}]\!](V)$ we have first to verify that $\mathcal{O}'$ satisfies the closure conditions.

Proposition 6.5 *For every goal $\bar{A}$ and set V we have*

$$r_{\bar{A},V}(Closure_{\bar{A},V}(\mathcal{O}'[\![\bar{A}]\!](V))) = \mathcal{O}'[\![\bar{A}]\!](V)$$

Proof
It suffices to verify that the closure conditions are satisfied by the transition system T'. However with respect to condition **P5** we have to require for $\ell = O$ that $(\vartheta_1 \ldots \vartheta_m) = (\sigma_1 \ldots \sigma_n)_{Var \setminus Z}$, with $Z \cap var(c) = \emptyset$. So in this case we only allow the introduction of local variables. Deletion of local variables is then taken care of by the restriction operator. Here we go.

P1 This case follows from the observation that if

$$\langle store(E), V\rangle \xrightarrow{\vartheta^O} \langle \Box, V\rangle$$

with $\vartheta = mgu(E)$, then, identifying $store(t = t)$ with $\Box$, we have

$$\langle store(E), V\rangle \xrightarrow{\vartheta^I} \langle \Box, V\rangle$$

P2 We have to consider the following cases corresponding to the computation rules:

C1 Let

$$\langle A, V\rangle \xrightarrow{\epsilon^O} \langle \bar{B}\vartheta_0\vartheta_1, store(E_2\vartheta_0\vartheta_1), W\rangle \xrightarrow{\vartheta^I} \langle \bar{B}\vartheta_0\vartheta_1\vartheta, store(E_2\vartheta_0\vartheta_1\vartheta), W\rangle$$

where for $H \leftarrow ask(E_1) : tell(E_2)|\bar{B}$ we have

- $\vartheta_0 = mgu(A, H)$, $\vartheta_{0|A} = \epsilon$
- $\vartheta_1 = mgu(E_1\vartheta_0)$, $\vartheta_{1|A} = \epsilon$

It follows that $\vartheta_0[\vartheta] = mgu(A\vartheta, H)$ and $\vartheta_1[\vartheta] = mgu(E_1\vartheta_0[\vartheta])$. Furthermore we $(\vartheta_0[\vartheta])(\vartheta_1[\vartheta]) = (\vartheta_0\vartheta_1)[\vartheta]$ (we may assume $dom(\vartheta) \cap dom(\vartheta_1) = \emptyset$ because sequences generated by such transitions do not occur in the operational semantics, as defined by $\mathcal{O}'$, of any goal). Furthermore we have that $E_2(\vartheta_0\vartheta_1)[\vartheta] = E_2\vartheta_0\vartheta_1\vartheta$ and $\bar{B}(\vartheta_0\vartheta_1)[\vartheta] = \bar{B}\vartheta_0\vartheta_1\vartheta$, so we conclude

$$\langle A, V\rangle \xrightarrow{\vartheta^I} \langle A\vartheta, V\rangle \xrightarrow{\epsilon^O} \langle \bar{B}\vartheta_0\vartheta_1\vartheta, store(E_2\vartheta_0\vartheta_1\vartheta), W\rangle$$

C2 Let

$$\langle tell(E), V\rangle \xrightarrow{\epsilon^O} \langle store(E), V\rangle \xrightarrow{\vartheta^I} \langle store(E\vartheta), V\rangle$$

Obviously we then have

$$\langle tell(E), V\rangle \xrightarrow{\vartheta^I} \langle tell(E\vartheta), V\rangle \xrightarrow{\epsilon^O} \langle store(E\vartheta), V\rangle$$

C5 Let

$$\langle p(t), V\rangle \xrightarrow{\epsilon^O} \langle p(z), store(z = t), V \setminus \{z\}\rangle \xrightarrow{\vartheta^I} \langle p(z)\vartheta, store(z = t\vartheta), V \setminus \{z\}\rangle$$

It then follows that

$$\langle p(t), V\rangle \xrightarrow{\vartheta^I} \langle p(t)\vartheta, V\rangle \xrightarrow{\epsilon^O} \langle p(z)\vartheta, store(z = t\vartheta), V \setminus \{z\}\rangle$$

C6 Let

$$\langle store(E_1, \ldots, E_n), V\rangle \xrightarrow{\epsilon^O} \langle store(E'_1, \ldots, E'_m), V \setminus Z\rangle \xrightarrow{\vartheta^I} \langle store(E'_1\vartheta, \ldots, E'_m\vartheta), V \setminus Z\rangle$$

where $(E_1 \wedge \ldots \wedge E_n) \Leftrightarrow \exists Z(E'_1 \wedge \ldots \wedge E'_m)$. It follows that $(E_1 \wedge \ldots \wedge E_n)\vartheta \Leftrightarrow \exists Z(E'_1 \wedge \ldots \wedge E'_m)\vartheta$. (Here we assume that ϑ does not affect the local variables Z, this assumption being justified because sequences generated by such transitions do not occur in the semantics, as defined by $\mathcal{O}'$, of any goal.) So we have

$$\langle store(E_1, \ldots, E_n), V\rangle \xrightarrow{\vartheta^I} \langle store(E_1\vartheta, \ldots, E_n\vartheta), V\rangle \xrightarrow{\epsilon^O} \langle store(E'_1\vartheta, \ldots, E'_m\vartheta), V \setminus Z\rangle$$

C7 Next we consider:

$$\langle \bar{A}, store(x = t), V\rangle \xrightarrow{\vartheta_1{}^O} \langle \bar{A}\vartheta_1, V\rangle \xrightarrow{\vartheta_2[\vartheta_1]^I} \langle \bar{A}\vartheta_1(\vartheta_2[\vartheta_1]), V\rangle$$

where $\vartheta_1 = mgu(x = t)$. Without loss of generality we may assume that $\vartheta_1 = \{x/t\}$, furthermore, let $\vartheta_2 = \{y/t'\}$. As $y \in var(t)$ implies $x \notin var(t')$ and $x \in var(t')$ implies $y \notin var(t)$ we have $\vartheta_1(\vartheta_2[\vartheta_1]) = \vartheta_2(\vartheta_1[\vartheta_2])$. Furthermore we have $mgu(x\vartheta_2 = t\vartheta_2) = \vartheta_1[\vartheta_2]$. So we conclude

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta_2{}^I}{\rightarrow} \langle \bar{A}\vartheta_2, store(x\vartheta_2 = t\vartheta_2), V\rangle \overset{\vartheta_1[\vartheta_2]^O}{\rightarrow} \langle \bar{A}\vartheta_1(\vartheta_2[\vartheta_1]), V\rangle$$

C8 Finally, the last case:

$$\langle \bar{A}, store(x = t), V\rangle \overset{\epsilon^O}{\rightarrow} \langle \bar{A}mgu(x = t), store(x = t), V\rangle \overset{\vartheta^I}{\rightarrow} \langle \bar{A}mgu(x = t)\vartheta, store(x\vartheta = t\vartheta), V\rangle$$

Let $\vartheta = \{y/t'\}$ and $z \in V$ correspond to y. We have

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta^I}{\rightarrow} \langle \bar{A}\vartheta, store(x\vartheta = t\vartheta), V\rangle \overset{\epsilon^O}{\rightarrow}$$

$$\langle \bar{A}[z/y], store(z = t', x[z/y] = t[z/y]), V \setminus \{z\}\rangle \overset{\epsilon^O}{\rightarrow}$$

$$\langle \bar{A}[z/y]mgu(x[z/y] = t[z/y]), store(z = t', x[z/y] = t[z/y]), V \setminus \{z\}\rangle =$$

$$\langle \bar{A}mgu(x, t)[z/y], store(z = t', x[z/y] = t[z/y]), V \setminus \{z\}\rangle \overset{\{z/t'\}^O}{\rightarrow}$$

$$\langle \bar{A}mgu(x = t)\vartheta, store(x\vartheta = t\vartheta), V \setminus \{z\}\rangle$$

Note that in this case we have generated additionally a binding to a local variable. This however does not matter as this binding can be deleted by the restriction operator. The main point is that an input substitution made after an application of **C8** can also be made before it.

This concludes the case of **P2**.

P3 Let

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta_1^O}{\rightarrow} \langle \bar{A}\vartheta_1, V\rangle \overset{Dec(\vartheta_2^I)}{\rightarrow} \langle \bar{A}\vartheta_1\vartheta_2, V\rangle$$

where $\vartheta_1 = mgu(x = t)$. Let $\vartheta = mgu(\mathcal{E}(\vartheta_2))$. Without loss of generality we may assume that $\vartheta_1 = \{x/t\}$. First we note that $\vartheta_1\vartheta_2 = (\vartheta_1[\vartheta])\vartheta_2$ ($\vartheta\vartheta_2 = \vartheta_2$). Furthermore we have that $\vartheta_2 \in mgu(t\vartheta = t) = mgu(x\vartheta_1[\vartheta] = t\vartheta_1[\vartheta])$. So we conclude

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta_1[\vartheta]^I}{\rightarrow} \langle \bar{A}\vartheta_1[\vartheta], store(t\vartheta = t), V\rangle \overset{Dec(\vartheta_2{}^O)}{\rightarrow} \langle \bar{A}\vartheta_1\vartheta_2, V\rangle$$

P4 Let

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta_1[\vartheta]^O}{\rightarrow} \langle \bar{A}\vartheta_1[\vartheta], V\rangle \overset{Dec(\vartheta_2{}^I)}{\rightarrow} \langle \bar{A}\vartheta_1[\vartheta]\vartheta_2, V\rangle$$

where $\vartheta = mgu(\mathcal{E}(\vartheta_2))$. First we note that $(\vartheta_1[\vartheta])\vartheta_2 = \vartheta_1\vartheta_2$ ($\vartheta\vartheta_2 = \vartheta$). Again, without loss of generality we assume that $\vartheta_1[\vartheta] = mgu(x = t) = \{x/t\}$. So for some t' we have $\vartheta_1 = \{x/t'\}$,with $t'\vartheta = t$. From this we infer that $\vartheta_2 \in mgu(t' = t'\vartheta) = mgu(x\vartheta_1 = t\vartheta_1)$. We conclude

$$\langle \bar{A}, store(x = t), V\rangle \overset{\vartheta_1{}^I}{\rightarrow} \langle \bar{A}\vartheta_1, store(x\vartheta_1 = t\vartheta_1), V\rangle \overset{\vartheta_2{}^O}{\rightarrow} \langle \bar{A}\vartheta_1\vartheta_2, V\rangle$$

P5 First we treat the case $\ell = I$. Let

$$\langle \bar{A}_1, V_1\rangle \overset{\vartheta_1{}^I}{\rightarrow} \ldots \overset{\vartheta_m{}^I}{\rightarrow} \langle \bar{A}_{m+1}, V_{m+1}\rangle$$

We have $\bar{A}_1(\vartheta_1 \ldots \vartheta_m) = \bar{A}_{m+1}$ (note that $V_1 = \ldots = V_m$). Furthermore by the restrictions on the variables Z_1 and Z_2 we have $(Z_1 \cup Z_2) \cap var(\bar{A}_1) = \emptyset$. Thus we have $\bar{A}_{m+1} = \bar{A}_1(\vartheta_1 \ldots \vartheta_m) = \bar{A}_1(\vartheta_1 \ldots \vartheta_m)_{Var \setminus Z_1} = \bar{A}_1(\sigma_1, \ldots \sigma_n)_{Var \setminus Z_2} = \bar{A}_1(\sigma_1 \ldots \sigma_n)$. Now let for $1 \leq i < n$: $\bar{A}'_i = \bar{A}_1(\sigma_1 \ldots \sigma_i)$. We then have

$$\langle \bar{A}_1, V_1 \rangle \xrightarrow{\sigma_1{}^I} \langle \bar{A}'_1, V_1 \rangle \ldots \xrightarrow{\sigma_n{}^I} \langle \bar{A}_{m+1}, V_{m+1} \rangle$$

Next we consider the case $\ell = O$. Let

$$\langle \bar{A}_1, V_1 \rangle \xrightarrow{\vartheta_1{}^O} \ldots \xrightarrow{\vartheta_m{}^O} \langle \bar{A}_{m+1}, V_{m+1} \rangle$$

Applying rule **C6** and **C8** we have

$$\langle \bar{A}_1, V_1 \rangle \xrightarrow{\epsilon^O} \langle \bar{A}_{m+1}, store(\mathcal{E}(\vartheta_1), \ldots, \mathcal{E}(\vartheta_m)), V_{m+1} \rangle \xrightarrow{\epsilon^O} \langle \bar{A}_{m+1}, store(\mathcal{E}(\sigma_1), \ldots, \mathcal{E}(\sigma_n)), V_{m+1} \rangle$$

By the restriction on the variables Z we may assume that $Z \cap var(\bar{A}_{m+1}) = \emptyset$. Furthermore we have $dom(\vartheta_1 \ldots \vartheta_m) \cap var(\bar{A}_{m+1}) = \emptyset$. So we infer $\bar{A}_{m+1}(\sigma_1 \ldots \sigma_n) = \bar{A}_{m+1}(\sigma_1 \ldots \sigma_n)_{Var \setminus Z} = \bar{A}_{m+1}(\vartheta_1 \ldots \vartheta_m) = \bar{A}_{m+1}$. So applying rule **C7** gives us

$$\langle \bar{A}_{m+1}, store(\mathcal{E}(\sigma_1), \ldots, \mathcal{E}(\sigma_n)), V_{m+1} \rangle \xrightarrow{c} \langle \bar{A}_{m+1}, V_{m+1} \rangle$$

where $c = \sigma_1^O \ldots \sigma_n^O$.

Now we can prove the following theorem.

Theorem 6.6 *For every goal $\bar{A}$ and set V we have*

$$\mathcal{D}[\![\bar{A}]\!](V) \subseteq \mathcal{O}'[\![\bar{A}]\!](V)$$

Proof

Let Φ be a continous operator with its least fixed point $\mathcal{D}$. It suffices to show that for every goal $\bar{A}$ and set V we have $\Phi(\mathcal{O}')[\![\bar{A}]\!](V) \subseteq \mathcal{O}'[\![\bar{A}]\!](V)$. Using proposition 6.5 this is proved by a straightforward induction on the goal $\bar{A}$.

Finally we have to establish that for every goal $\bar{A}$ and set V we have $Result_{\bar{A}}(\mathcal{O}[\![\bar{A}]\!](V)) = Result_{\bar{A}}(\mathcal{O}'[\![\bar{A}]\!](V))$. By proposition 6.4 and theorem 6.6 we have $Result_{\bar{A}}(\mathcal{O}[\![\bar{A}]\!](V)) \subseteq Result_{\bar{A}}(\mathcal{O}'[\![\bar{A}]\!](V))$. To relate T and T' we define a transition system $\bar{T}$, which is obtained from T by introducing the constructs of the form $store(E)$ and adding the rule

$$\langle store(E), V \rangle \xrightarrow{\vartheta^O} \langle \Box, V \rangle$$

$\vartheta = mgu(E)$. It is important to note that inconsistencies in the store do not lead to failures in this extension of T. We have the following proposition about $\bar{T}$.

Proposition 6.7 *With respect $\bar{T}$ we have that if*

$$\langle \bar{A}, store(E), V \rangle \xrightarrow{\vartheta}{}^* \langle \bar{B}, W \rangle$$

with $\langle \bar{B}, W \rangle \not\rightarrow \langle fail, W \rangle \mid \langle susp, W \rangle \mid \langle \Box, W \rangle$ and $\bar{B} \subseteq Atoms \cup Tell$ ($\bar{B}$ does not contain constructs of the form $store(E)$), then there exists $\vartheta_0 = mgu(E)$ and ϑ_1 such that

$$\langle \bar{A}\vartheta_0, V \rangle \xrightarrow{\vartheta_1}{}^* \langle \bar{B}, W \rangle$$

with $\vartheta = \vartheta_0\vartheta_1$. (Here $\langle \bar{A}, V \rangle \xrightarrow{\vartheta}{}^* \langle \bar{B}, W \rangle$ is to be interpreted as there exists $\vartheta_1, \ldots, \vartheta_n$ with $\vartheta = \vartheta_1 \ldots \vartheta_n$ and $\langle \bar{A}, V \rangle \xrightarrow{\vartheta_1{}^O} \ldots \xrightarrow{\vartheta_n{}^O} \langle \bar{B}, W \rangle$.)

Proof
We proceed by induction on the length of the derivation in $\bar{T}$. Let

$$\langle \bar{A}, store(E), V\rangle \overset{\vartheta' \circ}{\rightarrow} \langle \bar{A}'\vartheta', store(E\vartheta'), V'\rangle \overset{\vartheta''}{\rightarrow}{}^{*} \langle \bar{B}, W\rangle$$

First we consider the case that for some $A \in \bar{A}$ of the form $tell(E')$ or $store(E')$ we have $\vartheta' = mgu(E')$, and $\bar{A}' = \bar{A} \setminus A$. By the induction hypothesis we have for some ϑ'_1

$$\langle \bar{A}'\vartheta' mgu(E\vartheta'), V'\rangle \overset{\vartheta'_1}{\rightarrow}{}^{*} \langle \bar{B}, W\rangle$$

with $mgu(E\vartheta')\vartheta'_1 = \vartheta''$. But $\vartheta' mgu(E\vartheta') = mgu(E)mgu(E' mgu(E))$. This we can prove as follows: first note that for idempotant substitutions ϑ_1 and ϑ_2 such that $dom(\vartheta_1) \cap ran(\vartheta_2) = \emptyset$ we have $\vartheta_1\vartheta_2 = mgu(\mathcal{E}(\vartheta_1) \cup \mathcal{E}(\vartheta_2))$. We then have

$$\begin{array}{ll}
mgu(E')mgu(E mgu(E')) & = \\
mgu(\mathcal{E}(mgu(E')) \cup \mathcal{E}(mgu(E mgu(E')))) & = \\
mgu(E' \cup E mgu(E')) & = \\
mgu(E' \cup E) & = \\
mgu(E \cup E' mgu(E)) & = \\
mgu(\mathcal{E}(mgu(E)) \cup \mathcal{E}(mgu(E' mgu(E)))) & = \\
mgu(E)mgu(E' mgu(E)) &
\end{array}$$

From the above we infer

$$\langle \bar{A} mgu(E), V'\rangle \overset{\vartheta_1}{\rightarrow}{}^{*} \langle \bar{B}, W\rangle$$

where $\vartheta_1 = mgu(E' mgu(E))\vartheta'_1$, and $mgu(E)\vartheta_1 = \vartheta'\vartheta''$.

Next we have to consider the case that for some $A \in \bar{A}$ there exists a candidate clause $H \leftarrow ask(E_1) : tell(E_2)|\bar{B}'$ such that $\vartheta' = mgu(E_2\sigma_0\sigma_1)$, where $\sigma_0 = mgu(A, H)$ and $\sigma_1 = mgu(E_1\sigma_0)$. Given the following observations this case can be dealt with in a similar way as above: As σ_0 and σ_1 do not instantiate the variables of A we have for $\vartheta_0 = mgu(E)$

$$\sigma_0[\vartheta_0] = mgu(A\vartheta_0, H) \text{ and } \sigma_1[\vartheta_0] = mgu(E_1\sigma_0[\vartheta_0]).$$

Furthermore as $dom(\vartheta_0) \cap dom(\sigma_1) = \emptyset$ we have $\sigma_0[\vartheta_0](\sigma_1[\vartheta_0]) = (\sigma_0\sigma_1)[\vartheta_0]$ and $E_2\sigma_0\sigma_1\vartheta_0 = E_2(\sigma_0\sigma_1)[\vartheta_0])$ (due to the renaming mechanism ϑ_0 does not affect the variables of E_2).

Given the above proposition about the system $\bar{T}$ we can prove the following relation between the sytem $\bar{T}$ and T'.

Proposition 6.8 *If*

$$\langle \bar{A}, V\rangle \overset{\vartheta}{\rightarrow}{}^{*} \langle \bar{B}, W\rangle \text{ in } T'$$

with $\langle \bar{B}, W\rangle \rightarrow \langle fail, W\rangle \mid \langle susp, W\rangle \mid \langle \Box, W\rangle$ *and* $\bar{B} \subseteq Atoms$, *then there exists a* ϑ' *such that*

$$\langle \bar{A}, V\rangle \overset{\vartheta'}{\rightarrow}{}^{*} \langle \bar{B}, W\rangle \text{ in } \bar{T}$$

with $\vartheta'_{|\bar{A}} = \vartheta_{|\bar{A}}$. *We remark that* $\bar{A}$ *may contain constructs of the form* $store(E)$.

Proof We proceed by induction on the length of the derivation in T'. We have to consider the following cases:

C1 Let with respect to T'

$$\langle \bar{A}, V\rangle \overset{\epsilon^O}{\rightarrow} \langle A_1, \ldots, \bar{B}'\sigma, \ldots, A_n, store(E_2\sigma), V'\rangle \overset{\vartheta^O}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

where for some $A \in \bar{A} = A_1, \ldots, A_n$ there exists a candidate clause $H \leftarrow ask(E_1) : tell(E_2)|\bar{B}'$ such that $\sigma = \vartheta_0\vartheta_1$, with

- $\vartheta_0 = mgu(A, H)$, $\vartheta_{0|A} = \epsilon$
- $\vartheta_1 = mgu(E_1\vartheta_0)$, $\vartheta_{1|A} = \epsilon$
- $\vartheta_2 = mgu(E_2\sigma)$

By the induction hypothesis we have in $\bar{T}$

$$\langle A_1, \ldots, \bar{B}'\sigma, \ldots, A_n, store(E_2\sigma), V'\rangle \overset{\vartheta'}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}'} = \vartheta'_{|\bar{A}'}$, where $\bar{A}' = A_1, \ldots, \bar{B}'\sigma, \ldots, A_n, store(E_2\sigma)$. Applying proposition 6.7 we have in $\bar{T}$

$$\langle A_1\vartheta_2, \ldots, \bar{B}'\sigma\vartheta_2, \ldots, A_n\vartheta_2, V'\rangle \overset{\vartheta''}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta' = \vartheta_2\vartheta''$. From this it is easy to obtain (in $\bar{T}$)

$$\langle \bar{A}, V\rangle \overset{\vartheta_2\vartheta''}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $(\vartheta_2\vartheta'')_{|\bar{A}} = \vartheta_{|\bar{A}}$.

C2 Let (in T')

$$\langle \bar{A}, V\rangle \overset{\epsilon^O}{\rightarrow} \langle A_1, \ldots, A_n, store(E), V\rangle \overset{\vartheta}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

where $\bar{A} = A_1, \ldots, tell(E), \ldots, A_n$. By the induction hypothesis we have (in $\bar{T}$)

$$\langle A_1, \ldots, store(E), \ldots, A_n, V\rangle \overset{\vartheta'}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}} = \vartheta'_{|\bar{A}}$. Applying propostion 6.7 we have in $\bar{T}$

$$\langle A_1\vartheta_0, \ldots, A_n\vartheta_0, V\rangle \overset{\vartheta''}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta' = \vartheta_0\vartheta''$ and $\vartheta_0 = mgu(E)$. From this we obtain in $\bar{T}$

$$\langle \bar{A}, V\rangle \overset{\vartheta_0\vartheta''}{\rightarrow} \langle \bar{B}, W\rangle$$

C5 Let (in T')

$$\langle \bar{A}, V\rangle \overset{\epsilon^O}{\rightarrow} \langle A_1, \ldots, p(z), \ldots, A_n, store(z = t), V \setminus \{z\}\rangle \overset{\vartheta}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

where $p(t) \in \bar{A} = A_1, \ldots, A_n$. By the induction hypothesis we have (in $\bar{T}$)

$$\langle A_1, \ldots, p(z), \ldots, store(z = t), V \setminus \{z\}\rangle \overset{\vartheta'}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}'} = \vartheta'_{|\bar{A}'}$, where $\bar{A}' = A_1, \ldots, p(z), \ldots, A_n, store(z = t)$. Applying again propostion 6.7 we have in $\bar{T}$

$$\langle \bar{A}, V \setminus \{z\}\rangle \overset{\vartheta''}{\rightarrow}{}^* \langle \bar{B}, W\rangle$$

with $\vartheta' = \{z/t\}\vartheta''$. From this we infer

$$\langle \bar{A}, V\rangle \stackrel{\vartheta''}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}} = \vartheta''_{|\bar{A}}$.

C6 Let (in T')

$$\langle \bar{A}, store(E_1), \ldots, store(E_m), V\rangle \stackrel{\epsilon^O}{\rightarrow} \langle \bar{A}, store(E'_1), \ldots, store(E'_n), V \setminus Z\rangle \stackrel{\vartheta}{\rightarrow}^* \langle \bar{B}, W\rangle$$

where $(E_1 \wedge \ldots \wedge E_m) \Leftrightarrow \exists Z(E'_1 \wedge \ldots \wedge E'_n)$, $Z \subseteq V$. By the induction hypothesis we have (in $\bar{T}$)

$$\langle \bar{A}, store(E'_1), \ldots, store(E'_n), V \setminus Z\rangle \stackrel{\vartheta'}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}'} = \vartheta'_{|\bar{A}'}$ $(\bar{A}' = \bar{A}, store(E'_1), \ldots, store(E'_n))$. Applying propostion 6.7 (repeatedly) we have in $\bar{T}$

$$\langle \bar{A}\vartheta_0\rangle \stackrel{\vartheta''}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta' = \vartheta_0\vartheta''$ and $\vartheta_0 = mgu(E'_1) \ldots mgu(E'_n)$. Now, as $(E_1 \wedge \ldots \wedge E_m) \Leftrightarrow \exists Z(E'_1 \wedge \ldots \wedge E'_n)$, we have $mgu(E_1) \ldots mgu(E_n) = (mgu(E'_1) \ldots mgu(E'_n))_{Var \setminus Z}$. Furthermore we have that $var(\bar{A}) \cap Z = \emptyset$, so we obtain in $\bar{T}$

$$\langle \bar{A}, store(E_1), \ldots, store(E_m), V\rangle \stackrel{\vartheta_1\vartheta''}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta_1\vartheta''_{|\bar{A}} = \vartheta_{|\bar{A}}$, where $\vartheta_1 = mgu(E_1) \ldots mgu(E_m)$.

C7 Let (in T')

$$\langle \bar{A}, store(x = t), V\rangle \stackrel{\vartheta_0^O}{\rightarrow} \langle \bar{A}, , V\rangle \stackrel{\vartheta}{\rightarrow}^* \langle \bar{B}, W\rangle$$

where $\vartheta_0 = mgu(x = t)$. Follows immediately from the induction hypothesis.

C8 Let (in T')

$$\langle \bar{A}, store(x = t), V\rangle \stackrel{\epsilon^O}{\rightarrow} \langle \bar{A}', store(x = t), V\rangle \stackrel{\vartheta}{\rightarrow}^* \langle \bar{B}, W\rangle$$

where $\vartheta_0 = mgu(x = t)$ and $\bar{A}' = A'_1, \ldots, A'_n$, with $A'_i = A_i$ or $A'_i = A_i\vartheta_0$ $(\bar{A} = A_1, \ldots, A_n)$. By the induction hypothesis we have in $\bar{T}$

$$\langle \bar{A}', store(x = t), V\rangle \stackrel{\vartheta'}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta_{|\bar{A}''} = \vartheta'_{|\bar{A}''}$ $(\bar{A}'' = \bar{A}', store(x = t))$. Applying proposition 6.7 we have in $\bar{T}$

$$\langle \bar{A}\vartheta_0, V\rangle \stackrel{\vartheta''}{\rightarrow}^* \langle \bar{B}, W\rangle$$

with $\vartheta' = \vartheta_0\vartheta''$. From this we obtain in $\bar{T}$

$$\langle \bar{A}, store(x = t), V\rangle \stackrel{\vartheta'}{\rightarrow}^* \langle \bar{B}, W\rangle$$

Now we can state the proposition

Proposition 6.9 *We have*

$$Result_{\bar{A}}(\mathcal{O}'[\![\bar{A}]\!]) \subseteq Result_{\bar{A}}([\![\mathcal{O}[\![\bar{A}]\!])$$

Proof Straightforward, given proposition 6.8.

7 The full abstractness of $\mathcal{D}$

In this section we prove the full abstractness of $\mathcal{D}$ with respect to our observation criterium. To formulate the full abstractness of $\mathcal{D}$ we introduce the notion of an *initialized* program $P;\bar{A}$. We will write $\mathcal{D}[\![P;\bar{A}]\!]$ to make explicit that we consider $\mathcal{D}[\![\bar{A}]\!]$, the meaning of the goal $\bar{A}$, with respect to the program P.

In the following definition we give, given a sequence s and set of variables V, the construction of a *context*, that is, an initialized program $C(s,V) \in \mathrm{HC}(\emptyset,\emptyset)$, which will recognize the sequence s. The set of variables V are supposed to denote the set of variables of the initial goal for which s is a computation plus the varuables introduced by input substitutions of s.

Definition 7.1

- $C(ss,V) = C(ff,V) = C(dd,V) = \{\};\Box$
  $C(\bot,V) = \{\};p$
- $C(\{x_i/t\}^I s,\{x_1,\ldots,x_n\}) =$
  $\{p_0(x_1,\ldots,x_n) \leftarrow x_i = t, r(x_1,\ldots,x_i,\ldots,x_n)$
  $r(x_1,\ldots,t,\ldots,x_n) \leftarrow p_1(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)\} \cup P;p_0(x_1,\ldots,x_n)$
  where $C(s,\{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n\}) = P;p_1(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n)$ *and* p_0 *is a new predicate variable, not occurring in* $C(s,\{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n\})$. *(Note that we assume* $var(t) \subseteq \{x_1,\ldots,x_n\}$.*)*
- $C(\{x_i/t\}^O s,\{x_1,\ldots,x_n\}) =$
  $\{p_0(x_1,\ldots,t,\ldots,x_n) \leftarrow p_1(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,\bar{y})\} \cup P;p_0(x_1,\ldots,x_n)$
  where $\bar{y} = var(t) \setminus \{x_1,\ldots,x_n\}$,
  $C(c,\{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,\bar{y}\}) = P;p_1(x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,\bar{y})$, *and* p_0 *is a new predicate variable, not occurring in* $C(c,\{x_1,\ldots,x_{i-1},x_{i+1},\ldots,x_n,\bar{y}\})$.

The following proposition essentially states that a context $C(s,V)$ indeed recognizes the sequence s. However, to state this proposition we need first to introduce the notion $Visible(V,c)$ which gives the set of variables which can be "reached" from the set of variables V by the sequence c.

Definition 7.2 *We define* $Visible(V,c)$ *by induction on the length of* c*:*

$$\begin{array}{lcl} Visible(V,\lambda) & = & V \\ Visible(V,\vartheta^\ell.c) & = & Visible(V\vartheta,c) \end{array}$$

where $V\vartheta = \{var(\vartheta(x)) : x \in V\}$.

Proposition 7.3 *For* $c'.\alpha' \in \mathcal{D}[\![C(c.\alpha,V)]\!](W)$ *such that*

- $\vartheta_{c'|V} = \vartheta_{\bar{c}|V}$,
- $Visible(V,c') = dom(\vartheta_{c'})$, *the variables introduced by output substitutions of* c *do not occur in* W, *and*
- *the variables introduced by input substitutions of* c' *which are not visible in the resulting substitution (they are not contained in* $var(V\vartheta_{c'})$*) do not occur in* $\bar{c}$,

we have $c' \in Closure_{V,W}(\bar{c})$. *(Here* $\bar{c}$ *denotes the "mirror" of* c*:* $\overline{\vartheta^I c} = \vartheta^O \bar{c}$ *and* $\overline{\vartheta^O c} = \vartheta^I \bar{c}$.*)*

Proof
We proceed by induction on the length of $\bar{c}$. Let's go.

$\bar{c} = \lambda$: From $\vartheta_{c'|V} = \rho(\vartheta_{\bar{c}|V})$ and $Visible(V, c') = dom(\vartheta_{c'})$ it follows that $c' = \bar{c}$.

$\bar{c} = \{x/t\}^{\ell}\bar{c}_1$: Let $P_1; p_1 = C(c_1.\alpha, V \setminus \{x\} \cup var(t))$. We consider the cases $\ell = I$ and $\ell = O$ separately. (Note that for $\ell = I$ we have $var(t) \subseteq V$.)

$\bar{c} = \{x/t\}^{I}\bar{c}_1$: By the definition of the denotational semantics $\mathcal{D}$ we have

$$c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_k^I.c_1')$$

with $\vartheta_1 \ldots \vartheta_k(x) = t\vartheta'$, $\vartheta'_{|V} = (\vartheta_1 \ldots \vartheta_k)_{|V}$, and $c_1'.\alpha' \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_k\vartheta']\!](W')$, for some $W' \subseteq W$. First note that actually we have $W' = W$ because all the variables of p_1 are instantiated. Now let $\sigma_1, \ldots, \sigma_m$ such that $\sigma_1 = \{x/t\}$ and $(\sigma_1 \ldots \sigma_m)_{Var \setminus Z_1} = (\vartheta_1 \ldots \vartheta_m)_{Var \setminus Z_2}$, where $Z_1 \subseteq var(t) \setminus V$ contains those variables bound by ϑ' and $Z_2 = dom(\vartheta_1 \ldots \vartheta_m) \setminus V$. Thus we have by **P5**

$$\vartheta_1^I \ldots \vartheta_k^I.c_1' \in Closure_{V,W}(\sigma_1^I \ldots \sigma_m^I.c_1').$$

Note that the sequence $\sigma_1^I \ldots \sigma_m^I.c_1'$ is well-defined because we may assume that $Z_1 \cap var(c_1') = \emptyset$. This we can justify as follows: suppose first that $z \in Z_1$ is introduced by an output substitution of c_1'. So we then would have $z \in W$ which contradicts the assumption that the variables introduced by output substitutions of c do not occur in W. Suppose then that $z \in Z_1$ is introduced by a input substitution of c_1'. As $\vartheta'_{c|V} = \vartheta_{\bar{c}|V}$, $\vartheta_1 \ldots \vartheta_k(x) = t\vartheta'$, and $c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_k^I.c_1')$ we have that $z \notin var(V\vartheta'_c)$. But variables introduced by input subsitutions of c' (and z is such a variable as $c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_k^I.c_1')$) which are not contained in $var(V\vartheta'_c)$ do not occur in $\bar{c}$ (as z does).

Furthermore we note that the application of **P5** is justified because the variables of Z_1 do not occur in W as the variables introduced by output substitutions of c (thus introduced by input substitutions of $\bar{c}$) do not occur in W.

We have $p_1\vartheta_1 \ldots \vartheta_k\vartheta' = p_1\sigma_2 \ldots \sigma_m$ (note that $x \notin var(p_1)$). From this and $c_1'.\alpha' \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_k\vartheta']\!](W)$ we infer that

$$\sigma_2^I. \ldots .\sigma_m^I.c_1'.\alpha' \in \mathcal{D}[\![P_1; p_1]\!](W).$$

Now we apply the induction hypothesis yielding

$$\sigma_2^I. \ldots .\sigma_m^I.c_1' \in Closure_{V,W}(\bar{c}_1)$$

(it is not difficult to check that the conditions for applying the induction hypothesis are indeed satisfied). We conclude

$$c' \in Closure_{V,W}(\sigma_1^I. \ldots .\sigma_m^I.c_1') \subseteq Closure_{V,W}(\bar{c}).$$

$\bar{c} = \{x/t\}^{O}\bar{c}_1$: we distinguish the following three cases:

1. We have

   $$c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_k^I.c_1')$$

   such that $\vartheta_1 \ldots \vartheta_k(x) = t\vartheta'$, with $\vartheta' = (\vartheta_1 \ldots \vartheta_k)_{|dom(\vartheta')}$, and $c_1'.\alpha \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_k]\!](W')$, for some $W' \subseteq W$. This case is treated in a similar way as above, making additionally use of **P1**.

2. We have

   $$c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_{k-1}^I\vartheta_k^O.c_1')$$

with $\vartheta_k(x) = t\vartheta_1 \ldots \vartheta_{k-1}$ and $c'_1.\alpha' \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_k]\!](W')$, for some $W' \subseteq W$, as above we actually have $W' = W$. Let $\vartheta' = \{x/t\}$. It then follows by successive applications of **P2**

$$\vartheta_1^I \ldots \vartheta_{k-1}^I.\vartheta_k^O.c'_1 \in Closure_{V,W}(\vartheta'^O.\vartheta_1[\vartheta']^I.\vartheta_2[\vartheta'\vartheta_1]^I \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]^I.c'_1).$$

Furthermore we have $p_1\vartheta_1 \ldots \vartheta_k = p_1\vartheta_1[\vartheta'] \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]$. This we can prove as follows:

$$\begin{array}{l}(\vartheta_1 \ldots \vartheta_k)_{|\,Var\setminus\{x\}} = \\ \vartheta_1[\vartheta_k] \ldots \vartheta_{k-1}[\vartheta_k] = \\ \vartheta_1[\vartheta'\vartheta_1] \ldots \vartheta_i[\vartheta'\vartheta_1 \ldots \vartheta_i] \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-1}] = \\ \vartheta_1[\vartheta'] \ldots \vartheta_i[\vartheta'\vartheta_1 \ldots \vartheta_{i-1}] \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]\end{array}$$

The last equation follows from the following observation: Let $\vartheta_i = \{y/t'\}$. Now, if $x \in var(t')$ then $y \notin var(t\vartheta_1 \ldots \vartheta_{i-1})$. From $p_1\vartheta_1 \ldots \vartheta_k = p_1\vartheta_1[\vartheta'] \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]$ and $c'_1.\alpha' \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_k]\!](W)$ we derive

$$\vartheta_1[\vartheta']^I \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]^I.c'_1.\alpha' \in \mathcal{D}[\![P_1; p_1]\!](W).$$

Now we apply the induction hypothesis (again, it is not difficult to verify that the conditions for applying the induction hypothesis are satisfied) yielding

$$\vartheta_1[\vartheta']^I \ldots . \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-2}]^I.c'_1 \in Closure_{V,W'}(\bar{c}_1).$$

From which we derive

$$c' \in Closure_{V,W}(\vartheta'^O.\vartheta_1[\vartheta']^I \ldots \vartheta_{k-1}[\vartheta'\vartheta_1 \ldots \vartheta_{k-1}]^I.c'_1) \subseteq Closure_{V,W}(\bar{c}).$$

3. We have

$$c' \in Closure_{V,W}(\vartheta_1^I \ldots \vartheta_m^I.Dec(\sigma^O).c'_1)$$

with $\vartheta_1 \ldots \vartheta_m(x) = t\vartheta'\sigma$ for some ϑ' such that $\vartheta' = (\vartheta_1 \ldots \vartheta_m)_{dom(\vartheta')}$ (note that $dom(\vartheta_1 \ldots \vartheta_m) \cap dom(\sigma) = \emptyset$). Furthermore, we have $c'_1.\alpha \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_m\sigma]\!](W')$, for some $W' \subseteq W$ (again, we have actually $W' = W$). Let $\sigma_1, \ldots, \sigma_k$ such that $\sigma_1 = \{x/t\}$ and $\sigma_1 \ldots \sigma_k\sigma = \vartheta_1 \ldots \vartheta_m\sigma$. We first show that

$$\vartheta_1^I \ldots \vartheta_m^I.Dec(\sigma^O).c'_1 \in Closure_{V,W}(\sigma_1^O.\sigma_2^I \ldots \sigma_k^I.Dec(\sigma^I).c'_1)$$

By **P2** we have

$$\sigma_2^I \ldots \sigma_k^I.\sigma_1[\sigma_2 \ldots \sigma_k]^O.Dec(\sigma^I).c'_1 \in Closure_{V,W}(\sigma_1^O.\sigma_2^I \ldots \sigma_k^I.Dec(\sigma^I).c'_1)$$

Note that $\sigma_1[\sigma_2 \ldots \sigma_k] = \{x/t\vartheta'\}$. Applying **P3** gives us

$$\sigma_2^I \ldots \sigma_k^I.\{x/t\vartheta'\sigma\}^I.Dec(\sigma^O).c'_1 \in Closure_{V,W}(\sigma_2^I \ldots \sigma_k^I.\sigma_1[\sigma_2 \ldots \sigma_k]^O.Dec(\sigma^I).c'_1)$$

Finally an application of **P5** gives us

$$\vartheta_1^I \ldots \vartheta_m^I.Dec(\sigma^O).c'_1 \in Closure_{V,W}(\sigma_2^I \ldots \sigma_k^I.\{x/t\vartheta'\sigma\}^I.Dec(\sigma^O).c'_1)$$

Next we observe that $p_1\vartheta_1 \ldots \vartheta_m\sigma = p_1\sigma_2 \ldots \sigma_m\sigma$, from this and $c'_1.\alpha \in \mathcal{D}[\![P_1; p_1\vartheta_1 \ldots \vartheta_m\sigma\vartheta']\!](W')$ we obtain

$$\sigma_2^I \ldots \sigma_k^I.Dec(\sigma^I).c'_1.\alpha \in \mathcal{D}[\![P_1; p_1]\!](W).$$

Applying the induction hypothesis (again, it is not difficult to verify that the conditions for applying the induction hypothesis are indeed satisfied) gives us

$$\sigma_2^I \ldots \sigma_k^I.Dec(\sigma^I).c_1' \in Closure_{V,W'}(\bar{c}_1).$$

From which we derive

$$c' \in Closure_{V,W}(\sigma_1^O.\sigma_2^I \ldots \sigma_k^I.Dec(\sigma^I).c_1') \subseteq Closure(\bar{c}).$$

Proposition 7.4 *If* $c \in Closure_{\bar{A}_1,V_1}(c')$ *then* $\bar{c}' \in Closure_{\bar{A}_2,V_2}(\bar{c})$, *where* $V_1 \cap V_2 = \emptyset$ *and* $V_1 \cup V_2 = Var$.

Proof Straightforward induction by the number of applications of the closure conditions, making use of the fact that the conditions mirror themselves in the following sense: if c is derived from c' by an application of, say, **P1** then $\bar{c}'$ can be derived from $\bar{c}$ using **P1** again. In this way it is easy to see that **P1**, **P2** and **P5** mirror themselves. On the other hand, **P3** and **P4** mirror each other.

Finally, we can prove the main theorem:

Theorem 7.5 (Full abstractness of $\mathcal{D}$) *For arbitrary initialized programs* $P_1;\bar{A}_1$ *and* $P_2;\bar{A}_2$ *if*

$$\mathcal{D}[\![P_1;\bar{A}_1]\!] \neq \mathcal{D}[\![P_2;\bar{A}_2]\!]$$

then there exists a $P;\bar{A}$, a distinguishing context, *with no predicates in common with* $P_1;\bar{A}_1$ *and* $P_2;\bar{A}_2$, *such that*

$$Obs(P \cup P_1;\bar{A},\bar{A}_1) \neq Obs(P \cup P_2;\bar{A},\bar{A}_2).$$

Proof Suppose that $\mathcal{D}[\![P_1;\bar{A}_1]\!](V) \neq \mathcal{D}[\![P_2;\bar{A}_2]\!](V)$, for some V. Without loss of generality we may assume that $var(\bar{A}_1) = var(\bar{A}_2)$. Let $c.\alpha \in \mathcal{D}[\![P_1;\bar{A}_1]\!](V) \setminus \mathcal{D}[\![P_2;\bar{A}_2]\!](V)$. Let W consist of the variables of $\bar{A}_1$ and the variables introduced by input substitutions of c. Furthermore, let $P;\bar{A} = C(c.\alpha, W)$ and $(W_1, W_2) = Part(Var \setminus W)$. For arbitrary renaming $\rho \in V \to W_2$ we have $\rho(\bar{c}).\alpha' \in \mathcal{D}[\![P;\bar{A}]\!](W_1)$ and $\rho(c).\alpha \in \mathcal{D}[\![P_1;\bar{A}_1]\!](W_2)$, where

$$\begin{aligned} \alpha' &= ss \quad \text{if } \alpha \in \{ss, ff, dd\} \\ &= ff \quad \text{otherwise} \end{aligned}$$

So we have

$$\begin{aligned} \rho(Result_W(c.\alpha \parallel \bar{c}.\alpha')) &\in Result_W(\mathcal{D}[\![P;\bar{A}]\!](W_1) \parallel \mathcal{D}[\![P_1;\bar{A}_1]\!](W_2)) \\ &= Result_W(\mathcal{D}[\![P \cup P_1;\bar{A},\bar{A}_1]\!]) \\ &= Obs(P \cup P_1;\bar{A},\bar{A}_1) \end{aligned}$$

Now suppose that for some $\rho : V \to W_2$ we have $\rho(Result_W(c.\alpha \parallel \bar{c}.\alpha')) \in Obs(P \cup P_2;\bar{A},\bar{A}_2)$. So there exist $c'.\alpha \in \mathcal{D}[\![P_2;\bar{A}_2]\!](W_2)$ and $\bar{c}'.\alpha' \in \mathcal{D}[\![P;\bar{A}]\!](W_1)$ such that $\rho(Result_W(c.\alpha \parallel \bar{c}.\alpha')) = Result_W(c'.\alpha \parallel \bar{c}'.\alpha')$. Furthermore we may assume without loss of generality that the variables introduced by input substitutions of $\bar{c}'$ which are not visible in the resulting substitution (thatis, they do not occur in $\vartheta_{\bar{c}'|W}$) do not occur in $\rho(\bar{c})$. It follows that $\vartheta_{\bar{c}'|W} = \vartheta_{\rho(\bar{c})|W}$ and $Visible(W,\bar{c}') = dom(\vartheta_{\bar{c}'})$. By proposition 7.3 (note that $\bar{c}'.\alpha' \in \mathcal{D}[\![C(\rho(c).\alpha, W]\!](W_1)$) we then have that $\bar{c}' \in Closure_{W,W_1}(\rho(\bar{c}))$. So by proposition 7.4 we derive $\rho(c) \in Closure_{W,W_2}(c')$. So we have $\rho(c).\alpha \in \mathcal{D}[\![P_2;\bar{A}_2]\!](W_2)$. From which it is not difficult to deduce that $c.\alpha \in \mathcal{D}[\![P_2;\bar{A}_2]\!](V)$, contradicting our initial assumption.

8 Conclusion

We have studied in this paper the asynchronous nature of the communication in Concurrent Logic Languages. We have shown that to obtain a fully abstract semantics for these languages a quite different approach is required than for the imperative concurrent languages like CCS. One of the main differences consist in the description of deadlock behaviour. In Concurrent Logic Languages deadlock depends upon the *past* behaviour of a process, whereas in languages like CCS deadlock essentially depends upon the *current* state of the system as described by the failure sets.

A future research topic consist of generalizing the result to Concurrent Constraint Programming Languages. Another interesting line of research is to define a framework to study asynchronous communication in a more abstract setting.

References

[Apt88] K.R. Apt. Introduction to logic programming (revised and extended version. Technical Report CS-R8826, Centre for Mathematics and Computer Science, Amsterdam, 1988. To appear as a chapter in *Handbook of Theoretical Computer Science*, North-Holland (J. van Leeuwen, editor).

[BHR84] S.D. Brookes, C.A.R. Hoare, and W. Roscoe. A theory of communicating sequential processes. *JACM*, 31:499–560, 1984.

[dBK88] J.W. de Bakker and J.N. Kok. Uniform abstraction, atomicity and contractions in the comparative semantics of concurrent prolog. In *Proc. Fifth Generation Computer Systems*, pages 347–355, Tokyo, Japan, 1988. Extended Abstract, full version available as CWI report CS-8834. To appear on Theoretical Computer Science.

[dBKPR89a] F.S. de Boer, J.N. Kok, C. Palamidessi, and J.J.M.M. Rutten. Control flow versus logic: a denotational and a declarative model for guarded horn clauses. In *Proc. of the Symposium on Mathematical Foundations of Computer Science*, LNCS, pages 165–176, 1989.

[dBKPR89b] F.S. de Boer, J.N. Kok, C. Palamidessi, and J.J.M.M. Rutten. Semantic models for a version of parlog. In G. Levi and M. Martelli, editors, *Proc. of the Sixt International Conference on Logic Programming*, pages 621–636, Lisboa, 1989. MIT Press. Extended version to appear in Theoretical Computer Science.

[Ede85] E. Eder. Properties of substitutions and unifications. *Journal Symbolic Computation*, 1:31–46, 1985.

[GCLS88] R. Gerth, M. Codish, Y. Lichtenstein, and E. Shapiro. Fully abstract denotational semantics for concurrent prolog. In *Proc. of the Third IEEE Symposium on Logic In Computer Science*, pages 320–335, 1988.

[GL90] M. Gabbrielli and G. Levi. An unfolding reactive semantics for concurrent constraint programming. Technical Report TR ../90, Dipartimento di Informatica, Pisa, 1990.

[GMS89] H. Gaifman, M.J. Maher, and E. Shapiro. Rehactive behaviour semantics for concurrent constraint logic languages. In *Proc. of the North American Conference on Logic Programming*, 1989.

[Hir87] M. Hirata. Parallel list processing language oc and its self-description. *Computing Software*, 4(3):41–64, 1987. In Japanese.

[JL87] J. Jaffar and J.-L. Lassez. Constraint logic programming. In *Proc. ACM Symp. on Principles of Programming Languages*, pages 111–119, 1987.

[Kok88] J.N. Kok. A compositional semantics for concurrent prolog. In R. Cori and M. Wirsing, editors, *Proc. 5th Theoretical Aspects of Computer Science*, number 294 in LNCS, pages 373–388. Springer Verlag, 1988.

[Llo87] J.W. Lloyd. *Foundations of Logic Programming.* Springer Verlag, 1987. Second edition.

[LMM88] J.-L. Lassez, M.J. Maher, and K. Marriot. Unification revisited. In J. Minker, editor, *Foundations of deductive databases and logic programming*, Los Altos, 1988. Morgan Kaufmann.

[Mah87] M. J. Maher. Logic semantics for a class of committed choice programs. In J.-L. Lassez, editor, *Proc. of the Fourth Int. Conference on Logic Programming*, pages 877–893, Melbourne, 1987. MTI Press.

[Mil80] R. Milner. *A Calculus of Communicating Systems.* Number 92 in LNCS. Springer Verlag, New York, 1980.

[Sar85] V.A. Saraswat. Partial correctness semantics for cp($\downarrow, |, \&$). In *Proc. of the Conf. on Foundations of Software Computing and Theoretical Computer Science*, number 206 in LNCS, pages 347–368, 1985.

[Sar89] V.A. Saraswat. *Concurrent Constraint Programming Languages.* PhD thesis, january 1989. To be published by MTI Press.

[Sha89] E.Y. Shapiro. The family of concurrent logic languages. *ACM Computing Surveys*, 21(3):412–510, 1989.

[SR89] V.A. Saraswat and M. Rinard. Concurrent constraint programming. Technical report, Carnegie-Mellon University, 1989.

[Ued88] K. Ueda. Guarded horn clauses, a parallel logic programming language with the concept of a guard. In M. Nivat and K. Fuchi, editors, *Programming of Future Generation Computers*, pages 441–456, Amsterdam, 1988. North Holland.