FI MU

Faculty of Informatics
Masaryk University

A Fully Abstract Semantics for a Version of Synchronous Concurrent Constraint Programming

by

Jean-Marie Jacquet
Luboš Brim
David Gilbert
Mojmír Křetínský

FI MU Report Series FIMU-RS-99-08

December 1999

A Fully Abstract Semantics for a Version of Synchronous Concurrent Constraint Programming*

Jean-Marie Jacquet
Dept.of Comp.Sci., University of Namur, Belgium
jmj@info.fundp.ac.be

Luboš Brim
Faculty of Informatics, Masaryk University Brno, Czech Republic
brim@fi.muni.cz

David Gilbert
Dept.of Comp.Sci., City University, London, U.K.
drg@soi.city.ac.uk

Mojmír Křetínský
Faculty of Informatics, Masaryk University Brno, Czech Republic
mojmir@fi.muni.cz

Abstract

Concurrent constraint programming is classically based on asynchronous communication via a shared store. In previous work ([1, 2]), we presented a new version of the ask and tell primitives which features synchronicity, our approach being based on the idea of telling new information just in the case that a concurrently running process is asking for it. We turn in this paper to a semantic study of this new framework, called Scc.

It is first shown to be different in nature from classical concurrent constraint programming and from CCS, a classical reference in traditional concurrency theory. This suggests the interest of new semantics for Scc. To that end, an operational semantics reporting the steps of the computations is presented. A denotational semantics is then proposed. It uses monotonic sequences of labelled pairs of input-output states, possibly containing gaps, and ending – according to the logic programming tradition – with marks reporting success or failure. This denotational semantics is proved to be correct with respect to the operational semantics as well as fully abstract.

*L. Brim and M. Křetínský thank the Grant Agency of the Czech Republic, grant 201/97/0456, for supporting their research.

1 Introduction

As a consequence of being a generalisation of previous proposals for concurrent logic programming languages (Concurrent Prolog, Parlog, GHC, etc.), concurrent constraint programming has naturally inherited one of their main features: the *asynchronous* character of the communication. This is obtained by ask primitives blocking when the information on the store is not complete enough to entail the asked constraints. Following these lines, a natural way of obtaining synchronous communication in concurrent constraint programming is to force the reduction of ask and tell primitives to synchronise. Specifically, our approach considers tell primitives as lazy producers of information and views ask primitives as consumers of this information. From this point of view, a tell operation is reduced when an ask operation requires the told information. Moreover, the reduction of the two primitives is performed simultaneously. However, there is no reason to block ask and tell primitives on information which is already present. Consequently, stress is put on the novelty of the information and hence any `tell`(c) and `ask`(c) operations whose constraint argument c is entailed by the current store are reduced without partners.

This framework, called `Scc`, is presented in [1, 2] and its expressiveness has been demonstrated through the coding of a variety of examples. It has been argued that one advantage over related work such as [12, 8, 6], which introduce synchronisation by special operators and not by altering the behaviour of tell and ask primitives, is that `Scc` permits the specification of on *what* information the synchronisation should be made, rather than with *whom*. Synchronisation in `Scc` is thus data-oriented as opposed to process-oriented.

In order to motivate its interest and to substantiate the need for novel treatments, it is worth stressing the behavioural difference of `Scc` with, on the one hand, traditional concurrent constraint programming, as exemplified in the `cc` family of languages ([12]), and, on the other hand, traditional concurrent programming models, as exemplified by CCS ([9]).

It has been argued in [4] that the main difference between CCP and CCS is that complementary actions do not synchronise in CCP. This property is due to the fact that telling a constraint never suspends in cc. In contrast, the action of telling a constraint may suspend until an ask can make use of it. A synchronisation similar to that in CCS is thus produced. However, this synchronisation does not hold in `Scc` in the case that the told or asked constraints are entailed by the current contents of the store. A novel kind of synchronisation is thus achieved.

Major differences appear between the three frameworks. It is to be expected that these differences call for new treatments as well. In order to formalise our reasoning somewhat, let us turn to the example given in [4]. There CCS and CCP are compared by interpreting the action a as telling the constraint $x = a$, and the co-action $\overline{a}$ as asking the constraint $x = a$. To keep our notations consistent, we shall use "$+$" for the non-deterministic choice operator and ";" for the sequential composition operator.

Example 1 (Differentiating CCP and CCS (from [4])) *Let $A_1 = (\overline{a};\overline{b}) + (\overline{a};\overline{c}) + (\overline{a};\overline{d})$ and $A_2 = (\overline{a};\overline{b}) + (\overline{a};(\overline{c}+\overline{d}))$. In any compositional semantics for CCS these two processes must be distinguished. Indeed, they behave differently under the context $A = a;(b+c)$. The process A_1 can deadlock, by choosing the third alternative of the choice, while A_2 cannot. However, in* `cc`*, both A_1 and A_2 have the same behaviour. The process A_2 can deadlock by choosing the second alternative, because A can independently decide to produce $y = b$ (after $x = a$).*

Example 2 (Differentiating CCP and `Scc`**)** *Using the processes A, A_1, A_2 of the above example, the processes A_1 and A_2 are also distinguished by A in* `Scc` *for the same reason as in CCS.*

This example illustrates the difference between `Scc` and cc. Stated in other terms, in the CCP paradigm, since the tell operation is asynchronous, the choice guarded by *tell*(a) is a *local choice* whereas, since the tell operation is synchronous in `Scc`, this choice is *global* in `Scc`.

Nevertheless, synchronisation is only forced in `Scc` in the case that a process tries to tell information which is not already entailed by the store. Otherwise, it can proceed asynchronously. This fact is used subsequently to differentiate CCS and `Scc`.

Example 3 (Differentiating CCS and `Scc`**)** *Using again the above processes A, A_1, A_2, let $B_1 = \overline{b}; A_1$ and $B_2 = \overline{b}; A_2$. In CCS, these two processes can be distinguished by the process $B = b; A$ for the reasons exposed in Example 1. However, in* `Scc`*, both processes have the same behaviour. The process B_2 can now deadlock by choosing the second alternative because A can now independently proceed by the first alternative as $y = b$ is already entailed by the store.*

The distinction between `Scc` and CCS thus appear to be more subtle than the distinction between CCP and CCS. The choice guarded by `tell` is actually a "mixture" of global and local choice. The choice depends upon actions performed and upon the results of the past behaviour of the system, i.e. upon the constraint contained in the store.

The major contribution of this paper consists in a new fully abstract denotational for `Scc`. For the ease of presentation, we shall focus on a restricted version, called `Rscc`, where, as opposed to the full version, synchronous communication is performed only between one tell primitive and one ask primitive. It is worth noting that the above examples rely on this restricted form of communication. Moreover, finite goals are only treated here by noting that our results can be extended to recursion using classical techniques such as the ones discussed in [7] and [10].

The rest of the paper is organised as follows. Section 2 describes informally this restricted version, called `Rscc`. Section 3 presents the basic operational semantics $\mathcal{O}$. Section 4 studies the denotational semantics, which is proved correct in section 5 and fully abstract in section 9 thanks to properties discussed in sections 6, 7, and 8.

2 The Language `Scc`

2.1 The constraint system

As in [14], the constraint system underlying `Rscc` consists of any system of partial information that supports the entailment relation. Precisely, we assume given a set C of assertions. Each assertion, called an elementary constraint, provides partial information about the state of affairs. By adopting the notation $\mathcal{P}_f(C)$ to denote the set of finite subsets of S, we can thus formally define constraint systems as follows.

Definition 1 *A* simple constraint system *is a structure* $(C, \vdash)$ *where* C *is a non-empty denumerable set of tokens, also called* primitive constraints *or more simply* constraints*, and* $\vdash \subseteq \mathcal{P}_f(C) \times C$ *is a relation, called the* entailment relation*, satisfying the following properties: for any* $\rho, \sigma \in \mathcal{P}_f(C)$ *and any* $c \in C$*,*

i) if $c \in \sigma$ *then* $\sigma \vdash c$
ii) if $\sigma \vdash d$ *for any* $d \in \rho$ *and if* $\rho \vdash c$ *then* $\sigma \vdash c$*.*

The relation $\vdash$ *is extended to* $\mathcal{P}_f(C) \times \mathcal{P}_f(C)$ *by stating* $\rho \vdash \sigma$ *iff* $\rho \vdash c$ *for any* $c \in \sigma$*.*

Of course, in practice, the relation $\vdash$ should be decidable and as efficient as possible. Since the issues of how to actually resolve constraints are orthogonal to our concerns, we shall hide the set C and the relation $\vdash$ as parameters of our theory and assume them to be given in the sequel.

Two properties worth observing are that the extended entailment relation is reflexive and transitive.

Proposition 2 *The relation* $\vdash$: $\mathcal{P}_f(C) \times \mathcal{P}_f(C)$ *is reflexive and transitive.*

Proof Immediate. ■

Another property, useful for further developments, consists of a particular application of the reflexivity and transitivity properties of the entailment relation.

Proposition 3 *For any* $\rho, \sigma \in \mathcal{P}_f(C)$*, for any* $c \in C$*, if* $\rho \subseteq \sigma$ *and if* $\rho \vdash c$ *then* $\sigma \vdash c$*.*

Proof By reflexivity of the entailment relation, $\sigma \vdash \rho$. It follows from the transitivity of $\vdash$ that $\sigma \vdash c$. ■

Usually, stores are thought of as entailment closures and are defined as elements of constraint systems as follows.

Definition 4 *The* elements *of a constraint systems* $(C, \vdash)$ *are those subsets* σ *of* C *such that* $c \in \sigma$ *whenever* $\rho \vdash c$ *for some finite subset* ρ *of* σ*. Given* $\sigma \in \mathcal{P}_f(C)$*, the* element generated by σ*, noted* $\tilde{\sigma}$*, is defined as the set* $\{c \in C : \sigma \vdash c\}$*.*

Note that, by reflexivity and transitivity of the entailment relation, the set $\tilde{\sigma}$ is also $\{c : \rho \vdash c, \rho \text{ finite subset of } \sigma\}$. The next property establishes that entailment closures can be constructed incrementally.

Proposition 5 *For any $c \in C$ and any $\sigma \in \mathcal{P}_f(C)$,*

$$\widetilde{\sigma \cup \{c\}} = \{d \in C : \rho \vdash d, \rho \text{ finite subset of } \tilde{\sigma} \cup \{c\}\}.$$

Proof If $e \in \widetilde{\sigma \cup \{c\}}$ then $\sigma \cup \{c\} \vdash e$. As $\sigma \subseteq \tilde{\sigma}$ by reflexivity of $\vdash$, the set $\sigma \cup \{c\}$ is a finite subset of $\widetilde{\sigma \cup \{c\}}$. It follows that $\sigma \cup \{c\}$ qualifies as one of the subsets ρ of the proposition and consequently that $e \in \{d \in C : \rho \vdash d, \rho \text{ finite subset of } \tilde{\sigma} \cup \{c\}\}$.

Conversly, assume $\rho \vdash d$ for some finite subset ρ of $\tilde{\sigma} \cup \{c\}$. Let $\rho \cap \tilde{\sigma} = \{e_1, \cdots, e_p\}$. For $i = 1, \cdots, p$, one has $\sigma \vdash e_i$ by definition of $\tilde{\sigma}$. Therefore, $\sigma \cup \{c\} \vdash \rho$ by reflexivity of $\vdash$. It follows, by transitivity, that $\sigma \cup \{c\} \vdash d$, and, consequently, that $d \in \widetilde{\sigma \cup \{c\}}$. ∎

In the classical concurrent constraint setting, telling a constraint consists of adding that constraint to the current store and of generating the entailment closure of this new set. Moreover, asking a constraint consists of checking whether the constraint belongs to the current store.

As any computation starts with the empty store, any store of any computation is actually of the form $\tilde{\sigma}$ with σ the (finite) set of constraints already told. Asking the constraint c then amounts to checking whether $\sigma \vdash c$ holds. Similarly, the store obtained by updating σ is, by proposition 5, $\widetilde{\sigma \cup \{c\}}$.

In these conditions, we prefer to use finite sets of constraints and the entailment relation instead of entailment closures and set operators. Consequently, we define stores as follows.

Definition 6 *Define the set of stores,* Sstore, *as the set $\mathcal{P}_f(C)$.*

A property will be required in order to build a fully abstract semantics for the `Rscc` language. It is not guided by the synchronous nature of the language but rather is motivated by the need to find, at some point of a computation, constraints which are not entailed by the current store and which do not affect the rest of the computation. Technically speaking, this basically reduces to requiring that the considered constraint system is not degenerated in the sense that a finite subset of C entails all the constraints of C.

This leads to the following assumption. Note that it holds for most practical constraint systems.

Assumption 7 (Separating assumption) *For any finite set of constraints S, there is a constraint c such that*

1) $S \not\vdash c$
2) *for any $s \in S$ and any store ρ, if $\rho \cup \{c\} \vdash s$ then $\rho \vdash s$.*

This assumption will be applied subsequently in the following equivalent form.

Assumption 8 *For any finite set of constraints S and any store σ, there is a constraint c such that*

1) $\sigma \cup S \not\vdash c$.

2) *for any $s \in S$ and any store ρ, if $\rho \cup \{c\} \models s$ then $\rho \models s$.*

2.2 The parametric language

As already said, the language description is parametric with respect to $(C, \vdash)$, and so are the semantic constructions presented.

Definition 9 *Goals $G \in Sgoal$ are defined by the following grammar*

$$\begin{array}{rcl} G & ::= & \triangle \mid NG \\ NG & ::= & \texttt{ask}(c) \mid \texttt{tell}(c) \mid NG; NG \mid NG + NG \mid NG \parallel NG \end{array}$$

where c denotes an elementary constraint.

The constant $\triangle$ denotes the empty goal which is only capable of terminating successfully.

The atomic constructs `ask(c)` and `tell(c)` act on a given store in the following way: as usual, given a constraint `c`, the process `ask(c)` succeeds if `c` is entailed by the store; otherwise it is suspended. However, the process `tell(c)`, of a more lazy nature than the classical one, succeeds only if `c` is (already) entailed by the store; in this case it does not modify the store. Otherwise, it suspends. It is resumed by a concurrently suspended `ask(d)` operation provided that the conjunction of `c` and of the store entails `d`. In that case, both the `tell` and the `ask` are resumed synchronously and at the same time the store is atomically augmented with the constraint `c`.

The sequential composition $G_1; G_2$ is executed by first performing G_1 and, if G_1 terminates successfully, by performing G_2.

The nondeterministic choice $G_1 + G_2$ selects between the execution of G_1 or G_2 respectively provided that the selected component can perform at least one step of a computation (i.e. it is not immediately suspended). It is a *global nondeterministic* choice since the selection of a component can be influenced by the (global) store and by the environment of the process as well.

The parallel composition $G_1 \parallel G_2$ represents both the interleaving (merge) of the computation steps of the components involved (provided they can do these steps independently of each other) and also synchronisation: this is the case of the `tell` and `ask` described above.

Asynchronous tell

$$(\mathrm{T}_a) \qquad <tell(c), \sigma> \xrightarrow{\tau} <\Delta, \sigma>$$

$$\textit{if } \{\sigma \vdash c\}$$

Asynchronous ask

$$(\mathrm{A}_a) \qquad <ask(c), \sigma> \xrightarrow{\tau} <\Delta, \sigma>$$

$$\textit{if } \{\sigma \vdash c\}$$

Synchronous tell

$$(\mathrm{T}_s) \qquad <tell(c), \sigma> \xrightarrow{c} <\Delta, \sigma \cup \{c\}>$$

$$\textit{if } \{\sigma \nvdash c\}$$

Synchronous ask

$$(\mathrm{A}_s) \qquad <ask(c), \sigma> \xrightarrow{\overline{c}} <\Delta, \sigma \cup \{c\}>$$

$$\textit{if } \{\sigma \nvdash c\}$$

Figure 1: `Rscc` transition system: atomic operations

3 The operational semantics $\mathcal{O}$

3.1 The transition system

The operational semantics of `Rscc` is defined in Plotkin's style [11] by means of a labelled transition system. The configurations to be considered are composed here of the goal to be solved coupled to the current store, respectively representing here the statement under consideration and the computed values. The labels are used to indicate communication with the environment. No communication or a silent communication is indicated by τ. Following the notations of section 1, a tell action gives rise to a c label where c is the told constraint whereas a get action gives rise to a $\overline{c}$ label where c is the asked constrained.

Moreover, we define *Seconst* as the set of constraints c and of their complements $\overline{c}$ and take the convention that $\overline{\overline{c}}$ actually denotes c. We also extend the overline notation on sets by defining $\overline{S} = \{\overline{s} : s \in S\}$, for any set $S \subseteq \mathit{Seconst}$.

Definition 10 *Let Slabel be the set composed of a fresh symbol τ and the elements of Secons. Define the transition relation $\rightarrow$ as the smallest relation of $(Sgoal \times Sstore) \times Slabel \times (Sgoal \times Sstore)$ satisfying the rules of Figures 1 and 2. To simplify the presentation, the following relaxation of the syntax is employed in rules (I), (P), and (S): there, it is understood that when G'', G'_1, G'_2 are empty goals, target goals of the form $\triangle \;||\; G^*$, $G^* \;||\; \triangle$, $\triangle; G^*$ with $G^* \neq \triangle$ actually denote G^* and similarly that $\triangle \;||\; \triangle$ actually represent $\triangle$. Simplifications are of course operated accordingly.*

Choice

(C) $$\frac{<G,\sigma> \xrightarrow{l} <G'',\sigma''>}{\begin{array}{l}<G+G',\sigma> \xrightarrow{l} <G'',\sigma''>\\ <G'+G,\sigma> \xrightarrow{l} <G'',\sigma''>\end{array}}$$

Independent parallelism

(I) $$\frac{<G,\sigma> \xrightarrow{l} <G'',\sigma''>}{\begin{array}{l}<G \;||\; G',\sigma> \xrightarrow{l} <G'' \;||\; G',\sigma''>\\ <G' \;||\; G,\sigma> \xrightarrow{l} <G' \;||\; G'',\sigma''>\end{array}}$$

Synchronous communication

(P) $$\frac{<G_1,\sigma> \xrightarrow{c} <G'_1,\sigma'> \qquad <G_2,\sigma> \xrightarrow{\bar{d}} <G'_2,\sigma''>}{\begin{array}{l}<G_1 \;||\; G_2,\sigma> \xrightarrow{\tau} <G'_1 \;||\; G'_2,\sigma'>\\ <G_2 \;||\; G_1,\sigma> \xrightarrow{\tau} <G'_2 \;||\; G'_1,\sigma'>\end{array}}$$

if $\{\sigma' \vdash d\}$

Sequential composition

(S) $$\frac{<G,\sigma> \xrightarrow{l} <G'',\sigma''>}{<G;G',\sigma> \xrightarrow{l} <G'';G',\sigma''>}$$

Figure 2: `Rscc` transition system: composed goals

Notation 11 *To ease the reading, we use the notation*

$$<G,\sigma> \stackrel{l}{\not\longrightarrow}$$

to indicate the fact that there are no G' and σ' such that $<G, \sigma> \xrightarrow{l} <G', \sigma'>$. *Moreover, we employ*

$$<G, \sigma> \xrightarrow{l}$$

to indicate the fact that there are such G' and σ'.

The size of a goal, understood as the number of ask and tell primitives it contains, decreases after each computation step. This fact will be heavily used in subsequent proofs by induction.

Definition 12 *Define the* size *of a goal G as the number of tell and ask primitives it is composed of. This size is subsequently denoted as $size(G)$.*

Proposition 13 *For any goals G, G', any label l, and any stores σ, σ', if $<G, \sigma> \xrightarrow{l} <G', \sigma'>$, then $size(G') < size(G)$.*

Proof By recursive reasoning on the structure of G. ■

The transition is image-finite, as stated by the following proposition.

Proposition 14 *For any goal G and any store σ, the set $\{(G', l, \sigma') : <G, \sigma> \xrightarrow{l} <G', \sigma'>\}$ is finite.*

Proof Simple verification. ■

It is possible to further characterize the transitions in terms of the presence of tell and ask primitives in goals. For instance, a transition $<G, \rho> \xrightarrow{c} <G', \sigma>$ can only occur if G contains a *tell*(c) primitive in a place ready to be executed. The notion of top-context, defined next, specifies those places. Goals under reduction are obtained therefrom by replacing the place holders of top-context by tell and ask primitives. This corresponds to a first notion of application, called *direct application*, and denoted by single square brackets. The goals resulting from the transition step are basically obtained by replacing these primitives by the empty goal and by simplifying the goals as mentioned in definition 10. Choice makes however an exception. In that case, only one branch of the choice is to be considered. This leads to a second notion of application, called *reduced application*, and denoted by double square brackets.

Definition 15 *Top-contexts are functions inductively defined on goals by the following rules.*

i) A nullary top-context is associated with any goal. It is represented by the goal and both applications are defined as the constant mapping from $Sgoal^0$ to this goal with the goal as value.
ii) $\Box$ is a unary top-context that, for both applications, map any goal to itself. Thus, for any goal G, $\Box[G] = \Box[\![G]\!] = G$.

iii) If tc is an n-ary top-context and if G is a goal, then $(tc;G)$, $(tc+G)$, *and* $(G+tc)$ *are n-ary top-contexts. Their applications are defined as follows : for any goals* $G_1, \ldots, G_n$,

$$\begin{array}{rcl}(tc; G)[G_1, \cdots, G_n] & = & tc[G_1, \cdots, G_n]; G \\ (tc; G)[\![G_1, \cdots, G_n]\!] & = & tc[\![G_1, \cdots, G_n]\!]; G \\ (tc + G)[G_1, \cdots, G_n] & = & tc[G_1, \cdots, G_n] \\ (tc + G)[\![G_1, \cdots, G_n]\!] & = & tc[\![G_1, \cdots, G_n]\!] \\ (G + tc)[G_1, \cdots, G_n] & = & tc[G_1, \cdots, G_n] \\ (G + tc)[\![G_1, \cdots, G_n]\!] & = & tc[\![G_1, \cdots, G_n]\!]\end{array}$$

iv) If tc_1 *and* tc_2 *are m-ary and n-ary top-contexts then* $tc_1 \parallel tc_2$ *is a (m+n)-ary top-context. Its applications are defined as follows: for any goals* $G_1, \ldots, G_{m+n}$,

$$\begin{array}{rcl}(tc_1 \parallel tc_2)[G_1, \cdots, G_{m+n}] & = & (tc_1[G_1, \cdots, G_m]) \parallel (tc_2[G_{m+1}, \cdots, G_{m+n}]) \\ (tc_1 \parallel tc_2)[\![G_1, \cdots, G_{m+n}]\!] & = & (tc_1[\![G_1, \cdots, G_m]\!]) \parallel (tc_2[\![G_{m+1}, \cdots, G_{m+n}]\!])\end{array}$$

Proposition 16 *Let* G *and* G' *be two goals,* l *be a label, and* σ *and* ρ *be two stores such that* $\rho \subseteq \sigma$. *Then* $<G, \rho> \xrightarrow{l} <G', \sigma>$ *iff one of the following situations hold for some top-context tc*

1. $l = \tau$, $\rho = \sigma$, $\rho \vdash c$, $G = tc[tell(c)]$, $G' = tc[\![\Delta]\!]$
2. $l = \tau$, $\rho = \sigma$, $\rho \vdash c$, $G = tc[ask(c)]$, $G' = tc[\![\Delta]\!]$
3. $l = \tau$, $\rho \neq \sigma$, $\rho \not\vdash c$, $\rho \not\vdash d$, $\sigma = \rho \cup \{c\}$, $\sigma \vdash d$, $G = tc[tell(c), ask(d)]$, $G' = tc[\![\Delta, \Delta]\!]$
4. $l = \tau$, $\rho \neq \sigma$, $\rho \not\vdash c$, $\rho \not\vdash d$, $\sigma = \rho \cup \{c\}$, $\sigma \vdash d$, $G = tc[ask(d), tell(c)]$, $G' = tc[\![\Delta, \Delta]\!]$
5. $l = c$, $\rho \not\vdash c$, $\sigma = \rho \cup \{c\}$, $\sigma \vdash d$, $G = tc[tell(c)]$, $G' = tc[\![\Delta]\!]$
6. $l = \bar{c}$, $\rho \not\vdash c$, $\sigma = \rho \cup \{c\}$, $\sigma \vdash d$, $G = tc[ask(c)]$, $G' = tc[\![\Delta]\!]$

Proof Assume first that $<G, \rho> \xrightarrow{l} <G', \sigma>$ holds. Let us establish that one of the six cases hold by structural reasoning on G.

EMPTY GOAL. If $G = \Delta$ then $<G, \rho> \not\longrightarrow$ and thus the hypothesis is not verified.

TELL. If $G = tell(c)$ then either rule (T_a) or rule (T_s) has been applied. These applications correspond to cases 1 and 5, respectively.

ASK. If $G = ask(c)$ then either rule (A_a) or rule (A_s) has been applied. These applications correspond to cases 2 and 6, respectively.

SEQUENTIAL COMPOSITION. If $G = G_1; G_2$, then rule (S) has been applied. It follows that $<G_1, \rho> \xrightarrow{l} <G_1', \sigma>$, for some goal G_1'. Since $size(G_1) < size(G)$, the induction hypothesis can be applied, which yields one of the cases 1 to 6 where G and G' are respectively replaced by G_1 and G'_1. The thesis then results from the definition of top-context for sequential goals. For instance, if case 1 applies then $G_1 = tc[tell(c)]$ and $G'_1 = tc[\![\Delta]\!]$. Therefore

$$\begin{aligned} G &= G_1; G_2 \\ &= tc[tell(c)]; G_2 \\ &= (tc; G_2)[tell(c)] \end{aligned}$$

and

$$\begin{aligned} G' &= G'_1; G_2 \\ &= tc[\![\Delta]\!]; G_2 \\ &= (tc; G_2)[\![\Delta]\!] \end{aligned}$$

The other cases are treated similarly.

CHOICE. If $G = G_1 + G_2$ then rule (C) has been applied. It follows that either $<G_1, \rho> \xrightarrow{l} <G'_1, \sigma>$ or $<G_2, \rho> \xrightarrow{l} <G'_2, \sigma>$. Assume the first case hold, the second case being treated similarly. Then $G' = G'_1$. Since $size(G_1) < size(G)$, the induction can be applied to G_1 which yields one of the 6 cases of the thesis but translated for G_1. The thesis then follows from the definition of top-context for choice-goals as above.

PARALLEL COMPOSITION. If $G = G_1 \mid\mid G_2$ then either rule (I) or rule (P) has been applied. Let us examine each of these cases in turn.

Rule (I). If rule (I) has been applied then $<G_i, \rho> \xrightarrow{l} <G'_i, \sigma>$, for $i = 1$ or $i = 2$. Assume $i = 1$, the case where $i = 2$ being treated similarly. Then $G' = G'_1 \mid\mid G_2$ and, as before, the induction hypothesis can be applied to $<G_1, \rho> \xrightarrow{l} <G'_1, \sigma>$. The thesis then follows as before by noting that for any n-ary top-context tc, the structure $tc \mid\mid G_2$ is also an n-ary top-context such that

$$\begin{aligned} (tc \mid\mid G_2)[G^*_1, \cdots, G^*_n] &= (tc[G^*_1, \cdots, G^*_n]) \mid\mid G_2. \\ (tc \mid\mid G_2)[\![G^*_1, \cdots, G^*_n]\!] &= (tc[\![G^*_1, \cdots, G^*_n]\!]) \mid\mid G_2. \end{aligned}$$

Rule (P). Consider now the application of rule (P). Then assuming the first alternative of rule (P) — the other one being treated analogously — there are two constraints c and d such that

$$\begin{aligned} &<G_1, \rho> \xrightarrow{c} <G'_1, \sigma> \\ &<G_2, \rho> \xrightarrow{overlined} <G'_2, \sigma'> \\ &\sigma \vdash d. \end{aligned}$$

Since the sizes of G_1 and G_2 are strictly less than that of G, the induction hypothesis can be applied which yields

$$\begin{array}{llll} G_1 = tc_1[tell(c)] & G'_1 = tc_1[\![\Delta]\!] & \rho \not\vdash c & \sigma = \rho \cup \{c\} \\ G_2 = tc_2[ask(d)] & G'_2 = tc_2[\![\Delta]\!] & \rho \not\vdash d & \sigma' = \rho \cup \{d\} \end{array}$$

Define the binary top-context tc as the parallel composition of the unary contexts tc_1 and tc_2 ie

$$\begin{array}{rcl} tc[G_1^*, G_2^*] & = & tc_1[G_1^*] \parallel tc_2[G_2^*] \\ tc[\![G_1^*, G_2^*]\!] & = & tc_1[\![G_1^*]\!] \parallel tc_2[\![G_2^*]\!] \end{array}$$

Then

$$\begin{array}{rcl} G & = & G_1 \parallel G_2 \\ & = & tc[tell(c), ask(d)] \end{array}$$

$$\begin{array}{rcl} G' & = & G_1' \parallel G_2' \\ & = & tc[\![\Delta, \Delta]\!] \end{array}$$

Moreover, as noted before, $\rho \not\vdash c$, $\rho \not\vdash d$, $\sigma = \rho \cup \{c\}$, and $\sigma \vdash d$. Summing up, case 4 thus holds, which establishes the thesis.

Conversly, assume now that one of the cases 1 to 6 holds and let us structurally on tc and on G.

tc IS UNARY. If tc is unary then cases 1, 2, 5, or 6 can only qualify. Assume case 1 holds; the other cases being treated similarly. Let us reason inductively on the structure of G.

Tell primitive. If $G = tell(c)$ then rule (T_a) can be applied which yields the desired transition.

Ask primitive. If $G = ask(c)$ then rule (A_a) can be applied hich yields the desired transition.

Sequential composition. If $G = G_1; G_2$, then $tc = tc_1; G_2$ and $G_1 = tc_1[tell(c)]$ for some unary top-context tc_1. Since the size of G_1 is strictly less than the size of G, the induction hypothesis can be applied to tc_1 and G_1, which yields the transition

$$<G_1, \rho> \xrightarrow{l} <G_1', \sigma>$$

for some goal G_1'. Rule (S) then establishes the thesis.

Choice. If $G = G_1 + G_2$, then $tc = tc' + G_2$ or $tc = G_1 + tc'$. For both equalities, G_1 and G_2 are of size strictly less than that of G and thus the induction hypothesis can be applied to G_1 or G_2, respectively, and to tc'. The thesis then results from rule (C).

Parallel composition. If $G = G_1 \parallel G_2$, then $tc = tc_1 \parallel G_2$ or $tc = G_1 \parallel tc_2$, for some unary top-context tc_1 or tc_2. In both cases, the induction hypothesis can be applied to G_1 and tc_1, if the first equality holds, or to G_2 and tc_2 if the second equality holds. The thesis then results from rule (I).

tc IS BINARY. If tc is binary then case 3 or 4 is under consideration. Let us focus on case 3, the other one being treated similarly. Let us reason inductively on the size and structure of G.

In the base case, G must be of the form $tell(c) \mid\mid ask(d)$ and the thesis then results from rules (P), (T_s) and (A_s).

The cases of G being a sequentially composed goal or a goal composed by choice is treated first as for tc unary by applying the induction hypothesis and then by concluding thanks to rules (S) and (C) respectively.

The proof proceeds similarly if $G = G_1 \mid\mid G_2$ and $tc = tc_1 \mid\mid G_2$ or $tc = G_1 \mid\mid tc_2$ for binary top-contexts but this time by invoking rule (I).

To conclude, let us assume that $G = G_1 \mid\mid G_2$, $tc = tc_1 \mid\mid tc_2$ for two unary top-contexts such that $G_1 = tc_1[tell(c)]$ and $G_2 = tc_2[ask(d)]$. Then, since the thesis is established for unary top-context, and by considering cases 5 and 6 respectively, one gets

$$<G_1, \rho> \xrightarrow{c} <G'_1, \sigma>$$
$$<G_2, \rho> \xrightarrow{\overline{d}} <G'_2, \rho \cup \{d\}>$$

for $G'_1 = tc_1[\![\triangle]\!]$ and $G'_2 = tc_2[\![\triangle]\!]$. In these conditions, rule (P) can be applied, which yields

$$<G, \rho> \xrightarrow{\tau} <G', \sigma>$$

for $G' = G'_1 \mid\mid G'_2$. The thesis then results from definition 15, in particular from the equalities

$$tc[\![\triangle, \triangle]\!] = tc_1[\![\triangle]\!] \mid\mid tc_2[\![\triangle]\!] = G'_1 \mid\mid G'_2.$$

■

A usual application of proposition 16 will be as follows.

Proposition 17 *For any goals G_1, G_2, any store ρ, σ and any constraint c,*

1) *if $<G_1, \rho> \xrightarrow{c} <G_2, \sigma>$ then $\rho \not\vdash c$ and $\sigma = \rho \cup \{c\}$*
2) *if $<G_1, \rho> \xrightarrow{\overline{c}} <G_2, \sigma>$ then $\rho \not\vdash c$ and $\sigma = \rho \cup \{c\}$.*

Proof Direct consequence of proposition 16. ■

3.2 The Operational Semantics

Following one of the traditions in concurrency theory, the operational semantics specifies the successive non-hypothetical steps of the computation and ends with success if the computation reaches the empty goal and with failure otherwise. Success and failure are respectively indicated by the δ^+ and δ^- marks.

Definition 18 *Define the operational semantics $\mathcal{O}_h$: Sgoal $\rightarrow \mathcal{P}((Slabel \times Sstore)^{<\omega} \times \{\delta^+, \delta^s\})$ as the following function: for any goal G*

$$
\begin{array}{rcll}
\mathcal{O}_h(G) & = & \{\sigma_1.\cdots.\sigma_n.\delta^+ : & <G_0, \sigma_0> \xrightarrow{\tau} <G_1, \sigma_1> \xrightarrow{\tau} \cdots \xrightarrow{\tau} <G_n, \sigma_n>, \\
 & & & G_0 = G, \sigma_0 = \epsilon, G_n = \Delta, n \geq 0\} \\
 & & \cup & \\
 & & \{\sigma_1.\cdots.\sigma_n.\delta^- : & <G_0, \sigma_0> \xrightarrow{\tau} <G_1, \sigma_1> \xrightarrow{\tau} \cdots \xrightarrow{\tau} <G_n, \sigma_n> \not\xrightarrow{\tau}, \\
 & & & G_0 = G, \sigma_0 = \epsilon, G_n \neq \Delta, n \geq 0\}
\end{array}
$$

An alternative recursive definition of the semantics $\mathcal{O}_h$ is as follows.

Definition 19 *Define the semantics $\mathcal{O}_h^r$: Sgoal $\rightarrow$ Sstore $\rightarrow \mathcal{P}(Shist)$ as follows: for any goal $G \neq \Delta$, for any store ρ,*

$$
\begin{array}{rcl}
\mathcal{O}_h^r(\Delta)(\rho) & = & \{\delta^+\} \\
\mathcal{O}_h^r(G)(\rho) & = & \left\{ \begin{array}{l} \{\sigma.h : <G, \rho> \xrightarrow{\tau} <G', \sigma>, h \in \mathcal{O}_h^r(G')(\sigma)\}, \ \textit{if} <G, \rho> \xrightarrow{\tau}, \\ \{\delta^-\}, \ \textit{otherwise} \end{array} \right.
\end{array}
$$

Proposition 20 *For any goal G, $\mathcal{O}_h(G) = \mathcal{O}_h^r(G)(\epsilon)$.*

Proof Simple verification. ∎

It is here worth noting that the operational semantics $\mathcal{O}$ is not compositional. For instance, taking two independent constraints c and d,

$$\mathcal{O}(tell(c)) = \mathcal{O}(tell(d)) = \{\delta^-\}$$

whereas

$$
\begin{array}{rcl}
\mathcal{O}(tell(c) \parallel ask(c)) & = & \{\{c\}.\delta^+\} \\
\mathcal{O}(tell(d) \parallel ask(c)) & = & \{\delta^-\}
\end{array}
$$

The purpose of the next section is precisely to define a compositional semantics for `Rscc` which is correct with respect to the operational semantics but which also contains a "minimal" amount of information to be compositional. In other words, we shall try to define a fully abstract semantics.

4 A denotational semantics

There are two main reasons why the operational semantics $\mathcal{O}$ is not compositional. First, after having made a computation step, a store non necessarily empty is produced. A compositional semantics should therefore be defined in order to account for initial stores of any contents. Second, as deduced from the transition system, the computation of the goal $A \parallel B$ amounts to interleave execution steps of A and B. A compositional semantics should thus allow for the transition steps made by the environment.

Following [7], we shall model transition steps in the form of pairs of input and output stores and take as semantic domain, sets of sequences of such pairs. These sequences possibly contain gaps, accounting for the action of the environment. Moreover, they will start in any store, allowing previous steps resulting in a possibly non-empty store. However, in contrast to [7], they are required to be monotonic in the sense that information cannot be retracted. Finally, following [3], the deadlock mark δ^- is generalized to failures for compositionality and full abstraction purposes.

4.1 Preliminaries

Before going further, a few notations and concepts are in order.

Telling a constraint not entailed by the current store can be viewed as telling any weaker constraint not entailed by the current store since, on the one hand, the update of the store includes the constraint and hence entails weaker constraints, and, on the other hand, tell primitives can synchronize with ask operations for weaker constraints. Conversely, any ask operation can be viewed as asking for stronger constraints. Constraints to be considered as weaker or stronger are formalized as the following sets $c{\downarrow}\rho$ and $c{\uparrow}\rho$, respectively.

Definition 21 *For any goal G and any store σ, define*

$$\begin{aligned} c{\downarrow}\rho &= \{d \in C : \rho \cup \{c\} \models d, \rho \not\models d\} \\ c{\uparrow}\rho &= \{d \in C : \rho \cup \{d\} \models c\} \end{aligned}$$

Proposition 22 *For any constraint c, d, and any store ρ,*

i) if $d \in c{\downarrow}\rho$ then $d{\downarrow}\rho \subseteq c{\downarrow}\rho$
ii) if $d \in c{\uparrow}\rho$ then $d{\uparrow}\rho \subseteq c{\uparrow}\rho$

Proof The proposition results from the transitivity of the entailment relation. ■

For technical reasons, failure sets will be required to be closed in the following way.

Definition 23 *The set $S \subseteq$ Seconst is* closed *with respect to the store σ if it enjoys the following conditions: for any constraint c,*

1. *$c{\uparrow}\sigma \subseteq S$ when $c \in S$*
2. *$\overline{c{\downarrow}\sigma} \subseteq S$ when $\bar{c} \in S$.*

Proposition 24 *For any constraint c and any store σ such that $\sigma \not\vdash c$, the set Seconst$\setminus (c{\downarrow}\sigma)$ is closed wrt σ.*

Proof On the one hand, $e{\uparrow}\sigma \subseteq \mathit{Seconst} \setminus (c{\downarrow}\sigma)$, for any $e \in \mathit{Seconst} \setminus (c{\downarrow}\sigma)$. Indeed, let $f \in e{\uparrow}\sigma$ for such an e. Then, by definition 21, $\sigma \cup \{f\} \vdash e$. Suppose that, by contradiction, that $f \in c{\downarrow}\sigma$. Then, by definition 21, $\sigma \cup \{c\} \vdash f$. It follows that $\sigma \cup \{c\} \vdash e$ and thus that $e \in c{\downarrow}\sigma$, which is impossible.

On the other hand, as $\mathit{Seconst} \setminus (c{\downarrow}\sigma) \supseteq \overline{C}$, it holds that $\overline{e{\downarrow}\sigma} \subseteq \mathit{Seconst} \setminus (c{\downarrow}\sigma)$, for any $\overline{e} \in \mathit{Seconst} \setminus (c{\downarrow}\sigma)$. ∎

Proposition 25 *For any constraint c and any store σ such that $\sigma \nvdash c$, the set $\mathit{Seconst} \setminus (\overline{c{\uparrow}\sigma})$ is closed wrt σ.*

Proof On the one hand, as $\mathit{Seconst} \setminus (\overline{c{\uparrow}\sigma}) \supseteq C$, it holds that $e{\uparrow}\sigma \subseteq \mathit{Seconst} \setminus (\overline{c{\uparrow}\sigma})$, for any $e \in \mathit{Seconst} \setminus (\overline{c{\uparrow}\sigma})$.

On the other hand, consider $\overline{e} \in (\mathit{Seconst} \setminus \overline{c{\uparrow}\rho})$ and $\overline{f} \in \overline{e{\downarrow}\rho}$. If $\overline{f} \in \overline{c{\uparrow}\rho}$ then $\rho \cup \{f\} \vdash c$. Moreover $\rho \cup \{e\} \vdash f$ since $\overline{f} \in \overline{e{\downarrow}\rho}$. It follows that $\rho \cup \{e\} \vdash c$ which contradicts the fact that $e \notin c{\uparrow}\rho$. ∎

Proposition 26 *Let $(S_i)_{i \in I}$ be a family of subsets of Seconst closed wrt the store σ. Then $\bigcap_{i \in I} S_i$ is closed wrt σ.*

Proof Simple verification. ∎

The closure of a set X is defined in a constructive way as follows.

Definition 27 *For any set $X \subseteq \mathit{Seconst}$ and any store σ, the closure of X with respect to σ, noted $\widehat{X}^{\sigma}$, is defined as follows:*

$$\widehat{X}^{\sigma} = \bigcup \{x{\uparrow}\sigma : x \in X\} \cup \bigcup \{\overline{x{\downarrow}\sigma} : \overline{x} \in X\}$$

Proposition 28 *For any subset $X \subseteq \mathit{Seconst}$ and any store σ,*

i) $X \setminus \{\overline{x} \in X : \sigma \vdash x\} \subseteq \widehat{X}^{\sigma}$.
ii) *if X is closed wrt σ, then $\widehat{X}^{\sigma} \subseteq X$.*

Proof Point ii) results directly from definitions 23 and 27.

Point i) is established in two steps as follows. On the one hand, since, for any constraint x, $\{x\} \vdash x$ and thus $\sigma \cup \{x\} \vdash x$ one has $x \in x{\uparrow}\sigma$. It follows that

$$\{x \in X\} \subseteq \bigcup \{x{\uparrow}\sigma : x \in X\} \subseteq \widehat{X}^{\sigma} \tag{1}$$

On the other hand, by similar reasonings $x \in X{\downarrow}\sigma$ whenever $\sigma \nvdash x$. Hence,

$$\{\overline{x} \in X : \sigma \nvdash x\} \subseteq \bigcup \{\overline{x{\downarrow}\sigma} : \overline{x} \in X\} \subseteq \widehat{X}^{\sigma} \tag{2}$$

Point i) then results from the inclusions 1 and 2. ∎

Proposition 29 *Let X be a subset of Seconst, σ be a store and c be a constraint.*

1) *if $c \in X$ then $c \in \widehat{X}^\sigma$*
2) *if $\overline{c} \in X$ and if $\sigma \not\vdash c$ then $\overline{c} \in \widehat{X}^\sigma$.*

Proof This is a direct consequence of proposition 28. ■

Proposition 30 *For any store σ, and any subsets X, Y of Seconst, if $X \subseteq Y$ then $\widehat{X}^\sigma \subseteq \widehat{Y}^\sigma$.*

Proof Indeed, by definition 27 and the inclusion $X \subseteq Y$,

$$\begin{aligned} \widehat{X}^\sigma &= \bigcup\{x{\uparrow}\sigma : x \in X\} \cup \bigcup\{\overline{x{\downarrow}\sigma} : \overline{x} \in X\} \\ &\subseteq \bigcup\{y{\uparrow}\sigma : y \in Y\} \cup \bigcup\{\overline{y{\downarrow}\sigma} : \overline{y} \in Y\} \\ &= \widehat{Y}^\sigma \end{aligned}$$

■

Proposition 31 *For any subsets $X, Y \subseteq$ Seconst such that $X \subseteq Y$, for any store σ, if Y is closed wrt σ then $\widehat{X}^\sigma \subseteq Y$.*

Proof Indeed, in the conditions of the proposition, by proposition 30, $\widehat{X}^\sigma \subseteq \widehat{Y}^\sigma$. By proposition 28, $\widehat{Y}^\sigma \subseteq Y$ and consequently, $\widehat{X}^\sigma \subseteq Y$. ■

Proposition 32 *For any $X \subseteq$ Seconst and any store σ, the set $\widehat{X}^\sigma$ is closed wrt σ.*

Proof According to definition 23, the proposition is established by proving the two following properties:

i) if $c \in \widehat{X}^\sigma$ then $c{\uparrow}\sigma \subseteq \widehat{X}^\sigma$
ii) if $\overline{c} \in \widehat{X}^\sigma$ then $\overline{c{\downarrow}\sigma} \subseteq \widehat{X}^\sigma$

For property i), if $c \in \widehat{X}^\sigma$ then there is $x \in X$ such that $c \in x{\uparrow}\sigma$ and thus such that $\sigma \cup \{c\} \vdash x$. Moreover, for any $y \in c{\uparrow}\sigma$, one has $\sigma \cup \{y\} \vdash c$ and consequently $\sigma \cup \{y\} \vdash x$. It follows that $y \in x{\uparrow}\sigma$ and therefore that $y \in \widehat{X}^\sigma$.

For property ii), if $\overline{c} \in \widehat{X}^\sigma$ then there exist $x \in X$ such that $c \in x{\downarrow}\sigma$ and thus $\sigma \cup \{x\} \vdash c$ with $\sigma \not\vdash c$. Let $\overline{y} \in \overline{c{\downarrow}\sigma}$. One has $\sigma \cup \{c\} \vdash y$ and $\sigma \not\vdash y$. It follows that $\sigma \cup \{x\} \vdash y$ and thus, in view of $\sigma \not\vdash y$ that $y \in x{\downarrow}\sigma$. Hence $\overline{y} \in \overline{x{\downarrow}\sigma} \subseteq \widehat{X}^\sigma$. ■

The notions of steps, ending marks, and sequences of histories are formalized as follows. It is worth noting that, according to the intuition provided by the operational semantics, steps are required to be monotonic in the sense that the output state contains more information than the input state. Denotational histories will be required to be monotonic as well. However, for simplicity purposes, we do not require this property for histories, in general.

Definition 33

1) *Define the set of computational steps Sstep as follows:*

$$Sstep = \{(\rho, l, \sigma) : \rho, \sigma \in Sstore, l \in Slabel, \rho \subseteq \sigma\}$$

2) *Define the set of ending marks Smark as follows:*

$$Smark = \{\delta^+\} \cup (\{\delta^-\} \times \mathcal{P}(Seconst))$$

As usual, for suggestivity purposes, elements of $\{\delta^-\} \times \mathcal{P}(Seconst)$ *are rewritten as* $\delta^-(X)$ *instead of* (δ^-, X).

3) *Define the set of histories Shist as the set*

$$Shist = (Sstep)^{<\omega} \times (Sstore \times Smark)_{cl}$$

where the cl subscript indicates that pairs $(\sigma, \delta^-(X))$ *for X closed with respect to* σ *should only be considered as failing ending marks.*

Notation 34 *Let S be a set of histories of Shist and p be a sequence of* $(Sstep)^{<\omega}$*. Then*

$$\begin{array}{rcl} S[p] & = & \{h : p.h \in S\} \\ S^a & = & \{h : h = (\rho, \tau, \sigma).h' \in S\} \\ S^h & = & \{h : h = (\rho, l, \sigma).h' \in S, l \neq \tau\} \\ S^+ & = & \{h : h = (\sigma, \delta^+) \in S\} \\ S^- & = & \{h : h = (\sigma, \delta^-(X)) \in S\} \end{array}$$

Notation 35

1) *Let h be an history of Shist. Then*

$$init(h) = \begin{cases} \rho & \text{if } h = (\rho, l, \sigma).h' \\ \sigma & \text{if } h = (\sigma, \delta^+) \\ \sigma & \text{if } h = (\sigma, \delta^-(X)) \end{cases}$$

Moreover, for any set S of histories and any store σ*, we denote by* $S \Uparrow \sigma$ *the set* $\{h \in S : \sigma \subseteq init(h)\}$.

2) *Let, for* $n \geq 0$ *and* $\delta \in Smark$*, the history*

$$h = (\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta)$$

be in Shist. Then

$$\begin{array}{rcl} diff(h) & = & \cup_{i=1}^{n} (\sigma_i \setminus \rho_i) \\ const(h) & = & \rho_1 \cup \cdots \cup \rho_n \cup \rho_{n+1} \cup \sigma_1 \cup \cdots \cup \sigma_n \end{array}$$

Abusing notations, we shall lift diff *and* const *to sets of histories in the expected manner: for any set S of histories of Shist,*

$$
\begin{array}{rcl}
diff(S) & = & \bigcup\{diff(h) : h \in S\} \\
const(S) & = & \bigcup\{const(h) : h \in S\}
\end{array}
$$

Monotonic, continuous, and real histories are formalized as follows.

Definition 36 *An history $h \in Shist$ is* monotonic *iff it has the form*

$$(\rho_0, l_0, \sigma_0).\cdots.(\rho_{n-1}, l_{n-1}, \sigma_{n-1}).(\rho_n, \delta)$$

with $\delta \in Smark$ and, $\sigma_{i-1} \subseteq \rho_i$, for $i = 1, \cdots, n$.

Definition 37 *An history $h \in Shist$ is* continuous *iff it has the form*

$$(\sigma_0, l_1, \sigma_1).(\sigma_1, l_2, \sigma_2).\cdots.(\sigma_{n-1}, l_n, \sigma_n).(\sigma_n, \delta)$$

with $\delta \in Smark$. In that case, $\widetilde{h}$ denotes the following sequence of stores

$$\widetilde{h} = \sigma_1.\cdots.\sigma_n.\delta'$$

where

$$\delta' = \begin{cases} \delta^+ & \textit{if } \delta = \delta^+ \\ \delta^- & \textit{if } \delta = \delta^-(X) \textit{ for some set } X \end{cases}$$

Definition 38 *An history $h \in Shist$ is* real *iff it has the form*

$$(\rho_1, \tau, \sigma_1).\cdots.(\rho_n, \tau, \sigma_n).(\sigma_{n+1}, \delta)$$

with $\delta \in Smark$.

Definition 39 *To any history $h = (\rho_1, l_1, \sigma_1).\cdots.(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta)$ (possibly non real), we associate a goal, called the* complementor, *defined as $\triangle$ if all the l_i's are τ or otherwise from $g_1; \cdots; g_n$ by removing all the occurrences of the empty goal. There,*

$$g_i = \begin{cases} \triangle & \textit{if } l_i = \tau \\ ask(c) & \textit{if } l_i = c \\ tell(c) & \textit{if } l_i = \overline{c}. \end{cases}$$

This goal is subsequently denoted by $Co(h)$. Moreover, the associated real history

$$(\rho_1, \tau, \sigma_1).\cdots.(\rho_n, \tau, \sigma_n).(\rho_{n+1}, \delta)$$

is called the completed *history of h whereas the history*

$$(\rho_1, \overline{l_1}, \sigma_1).\cdots.(\rho_n, \overline{l_n}, \sigma_n).(\rho_{n+1}, \delta)$$

obtained by complementing the labels, is called the negative *of h. These histories are respectively denoted as h^c and h^n. Note that we extend, here and in the sequel of the paper, the convention $\overline{\overline{c}} = c$ for any constraint c by the convention that $\overline{\tau}$ actually stands for τ.*

4.2 Semantic domain

Definition 40 *The semantic domain is defined as the set Sdhist of monotonic histories. Its elements are subsequently called* denotational histories *or, more often,* histories, *when the context allows this abuse of language.*

4.3 Denotational semantics

The semantic domain being specified, defining a compositional semantics consists in, on the one hand, specifying the meaning of elementary statements and, on the other hand, providing an operator at the semantic level for each syntatic operator. We start by this last task in the following subsection. A compositional semantics is defined next. It is called denotational in view of its compositionality property and the fact that it is defined on denotations only without reference to a transition system. It is also proved correct with respect to the history operational semantics $\mathcal{O}_h$. Finally it is established to be fully abstract.

4.3.1 Semantic operators

There are three operators to combine elementary goals: sequential composition, parallel composition, and choice. Let us examine each of them in turn.

Sequential composition. Since the semantic histories may include gaps and start on any input store, composing the meaning of two subgoals amounts to concatenating their histories. This is achieved by the following operator, where further care is taken, in the expected manner, for monotonicity and the termination marks.

Definition 41 *Define* $\tilde{;}$: $\mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \to \mathcal{P}(Sdhist)$ *as the following function: for any subsets* S_1, S_2 *of Sdhist,*

$$\begin{aligned} S_1 \,\tilde{;}\, S_2 = \; & \{h_1.h_2 : h_1.(\rho_1, \delta^+) \in S_1, h_2 \in S_2, \rho_1 \subseteq init(h_2)\} \\ & \cup \{h_1.(\rho, \delta^-(X)) : h_1.(\rho, \delta^-(X)) \in S_1\} \end{aligned}$$

For subsequent results, it is worth providing an equivalent recursive definition of the operator $\tilde{;}$.

Definition 42 *Define* $\tilde{;}_r$: *Sdhist* $\times$ *Sdhist* $\to \mathcal{P}(Sdhist)$ *as the following function: for any histories* h_1, h_2 *of Sdhist,*

$$h_1 \,\tilde{;}_r\, h_2 = \begin{cases} (\rho_1, l_1, \sigma_1).(h_r \,\tilde{;}_r\, h_2) \text{ if } h_1 = (\rho_1, l_1, \sigma_1).h_r \\ h_2 \text{ if } h_1 = (\sigma_1, \delta^+) \text{ and } \sigma_1 \subseteq init(h_2) \\ h_1 \text{ if } h_1 = (\sigma_1, \delta^-(X)) \end{cases}$$

Proposition 43 *For any subsets* S_1, S_2 *of Sdhist,*

$$S_1 \,\tilde{;}\, S_2 = \{h_1 \,\tilde{;}_r\, h_2 : h_1 \in S_1, h_2 \in S_2\}.$$

Proof Simple verification. ■

Parallel composition. Parallel composition is modelled in an interleaving fashion. Consequently, composing in parallel two semantic histories amounts to take their merge or to synchronize their steps. Again care has to be taken to termination marks, as formalized below.

Definition 44 *Define the parallel composition of two histories as the function* $\widetilde{\|}_h$: *Sdhist* $\times$ *Sdhist* $\rightarrow \mathcal{P}($*Sdhist*$)$ *according to their form by means of the following equalities. There,* δ *stands either for* δ^+ *or* $\delta^-(X)$*, for some set* X*.*

$$
\begin{array}{l}
(\rho_1, l_1, \sigma_1).h_1 \;\widetilde{\|}_h\; (\rho_2, l_2, \sigma_2).h_2 \quad = \\[1ex]
\qquad\qquad \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \;\widetilde{\|}_h\; (\rho_2, l_2, \sigma_2).h_2, \sigma_1 \subseteq \rho_2\} \\
\qquad\qquad \cup \{(\rho_2, l_2, \sigma_2).h : h \in (\rho_1, l_1, \sigma_1).h_1 \;\widetilde{\|}_h\; h_2, \sigma_2 \subseteq \rho_1\} \\
\qquad\qquad \cup \{(\rho, \tau, \sigma).h : \rho_1 = \rho_2 = \rho, l_1 = c_1, l_2 = \overline{c_2}, \\
\qquad\qquad\qquad\qquad \sigma = \sigma_1 = \rho \cup \{c_1\}, \sigma_2 = \rho \cup \{c_2\}, \sigma \vdash c_2, \\
\qquad\qquad\qquad\qquad h \in h_1 \;\widetilde{\|}_h\; h_2\} \\
\qquad\qquad \cup \{(\rho, \tau, \sigma).h : \rho_1 = \rho_2 = \rho, l_1 = \overline{c_1}, l_2 = c_2, \\
\qquad\qquad\qquad\qquad \sigma = \sigma_2 = \rho \cup \{c_2\}, \sigma_1 = \rho \cup \{c_1\}, \sigma \vdash c_1, \\
\qquad\qquad\qquad\qquad h \in h_1 \;\widetilde{\|}_h\; h_2\}
\end{array}
$$

$$
\begin{array}{rcl}
(\rho_1, l_1, \sigma_1).h_1 \;\widetilde{\|}_h\; (\rho_2, \delta_2) & = & \\[2ex]
(\rho_2, \delta_2) \;\widetilde{\|}_h\; (\rho_1, l_1, \sigma_1).h_1 & = & \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \;\widetilde{\|}_h\; (\rho_2, \delta_2)\}
\end{array}
$$

$$
(\rho_1, \delta_1) \;\widetilde{\|}_h\; (\rho_2, \delta_2) \quad = \quad \left\{
\begin{array}{l}
\{(\rho_1, \delta^+)\}, \ \textit{if } \rho_1 = \rho_2 \textit{ and } \delta_1 = \delta_2 = \delta^+, \\
\{(\rho_1, \delta^-(X))\}, \ \textit{if } \rho_1 = \rho_2, \delta_1 = \delta^+, \delta_2 = \delta^-(X), \\
\{(\rho_1, \delta^-(X))\}, \ \textit{if } \rho_1 = \rho_2, \delta_1 = \delta^-(X), \delta_2 = \delta^+, \\
\{(\rho_1, \delta^-(X))\}, \ \textit{if } \rho_1 = \rho_2, \delta_1 = \delta^-(X_1), \delta_2 = \delta^-(X_2), \\
\qquad\qquad\qquad\qquad X \subseteq X_1 \cap X_2, X \textit{ closed wrt } \rho_1 \\
\qquad\qquad\qquad\qquad (\textit{Seconst} \setminus X_1) \cap \overline{(\textit{Seconst} \setminus X_2)} = \emptyset, \\
\emptyset, \ \textit{otherwise.}
\end{array}
\right.
$$

A word on this definition is in order. The first equality involves four sets. The first two correspond to making, as first step, the first step of the histories under consideration. The last two correspond to synchronizing the first steps of these histories.

The second equality basically asserts that ending marks should be combined only and to postpone this combination after all the steps of the two histories have been treated. This is suggested by the operational semantics: when one of the goals ends, the other concurrent goal has to continue.

The last equality deals with combining ending marks. As a general rule, marks may be combined only if they agree on their input store. Note that this is actually no problem since denotational histories may include gaps.

All the cases should be clear, except the fourth one. The intuition behind it relies on the standard meaning of failures. A mark of the form $\delta^{-}(X)$ expresses that the considered goal is suspended and that, whatever its environment is, it is unable to perform the actions of X. Note that X is not required to contain all the actions that the goal cannot do; it is simply a subset of the set of these actions. Therefore, for two goals G_1 and G_2 suspended and unable to perform the actions of X_1 and X_2, respectively, the parallel composition $G_1 \mid\mid G_2$ is unable to perform the actions of $X_1 \cap X_2$ and is suspended provided G_1 and G_2 cannot synchronize. A sufficient condition for that is obtained by noting that the actions that G_i $(i = 1, 2)$ can do are in the complement of X_i, namely in $Seconst \setminus X_i$. Consequently, if telling a constraint is understood as telling all the new constraints it subsumes, then G_1 and G_2 synchronize only if there is an action of $Seconst \setminus X_1$ which is the complement of an action of $Seconst \setminus X_2$ ie only if

$$(Seconst \setminus X_1) \cap \overline{(Seconst \setminus X_2)} \neq \emptyset.$$

More generally, if X_i^* $(i = 1, 2)$ is exactly the set of all actions that G_i cannot do (and not simply a subset), then, following the same reasoning, it is easy to establish that G_1 and G_2 do not synchronize if and only if

$$(Seconst \setminus X_1^*) \cap \overline{(Seconst \setminus X_2^*)} \neq \emptyset.$$

Moreover, the set of actions that $G_1 \mid\mid G_2$ cannot perform is included in $X_1^* \cap X_2^*$.

Definition 45 *Define the parallel composition of two sets of histories as the natural lifting of function* $\widetilde{||}_h$ *, namely as the function* $\widetilde{||}$ $: \mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \rightarrow \mathcal{P}(Sdhist)$ *defined as follows: for any subset* S_1*,* S_2 *of Sdhist,*

$$S_1 \,\widetilde{||}\, S_2 = \bigcup \{h_1 \,\widetilde{||}_h\, h_2 : h_1 \in S_1, h_2 \in S_2\}$$

As we shall see in section 5, the following notations provide a convenient means to rewrite the parallel composition of histories.

Definition 46 *Define the operators*

$$\begin{array}{l} \underline{||} : \mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \rightarrow \mathcal{P}(Sdhist) \\ \otimes : \mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \rightarrow \mathcal{P}(Sdhist) \\ \odot : \mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \rightarrow \mathcal{P}(Sdhist) \end{array}$$

as follows: for any subsets S_1*,* S_2 *of Sdhist,*

$$\begin{array}{rcl} S_1 \,\underline{||}\, S_2 & = & \{(\rho_1, l_1, \sigma_1).h : \; (\rho_1, l_1, \sigma_1).h_1 \in S_1, h_2 \in S_2, h \in h_1 \,\widetilde{||}_h\, h_2, \\ & & \qquad \sigma_1 \subseteq init(h)\} \\ & & \\ S_1 \otimes S_2 & = & \{(\rho, \tau, \sigma).h : \; (\rho, c, \sigma).h_1 \in S_1, (\rho, \overline{d}, \sigma').h_2 \in S_2, h \in h_1 \,\widetilde{||}_h\, h_2, \\ & & \qquad c, d \in C, \sigma = \rho \cup \{c\}, \sigma \vdash d, \sigma' = \rho \cup \{d\}\} \\ & & \cup \{(\rho, \tau, \sigma).h : \; (\rho, c, \sigma).h_2 \in S_2, (\rho, \overline{d}, \sigma').h_1 \in S_1, h \in h_1 \,\widetilde{||}_h\, h_2, \\ & & \qquad c, d \in C, \sigma = \rho \cup \{c\}, \sigma \vdash d, \sigma' = \rho \cup \{d\}\} \\ & & \\ S_1 \odot S_2 & = & \bigcup \{(\rho_1, \delta_1) \,\widetilde{||}_h\, (\rho_2, \delta_2) : (\rho_1, \delta_1) \in S_1, (\rho_2, \delta_2) \in S_2\} \end{array}$$

Proposition 47 *For any subsets S_1, S_2 of Sdhist,*

$$S_1 \,\widetilde{\|}\, S_2 = S_1 \lfloor\!\lfloor S_2 \cup S_2 \lfloor\!\lfloor S_1 \cup S_1 \otimes S_2 \cup S_1 \odot S_2$$

Proof The proposition results from definitions 44, 45, 46 as well as from the monotonic property of denotational histories: for any subsets $S_1, S_2 \subseteq$ *Sdhist*,

$$
\begin{aligned}
S_1 \,\widetilde{\|}\, S_2 \;=\;& \bigcup\{h_1 \,\widetilde{\|}_h\, h_2 : h_1 \in S_1, h_2 \in S_2\} \\
=\;& \bigcup\{(\rho_1, l_1, \sigma_1).h_1 \,\widetilde{\|}_h\, (\rho_2, l_2, \sigma_2).h_2 : (\rho_1, l_1, \sigma_1).h_1 \in S_1, \\
& \qquad\qquad (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \bigcup\{(\rho_1, l_1, \sigma_1).h_1 \,\widetilde{\|}\, (\rho_2, \delta_2) : (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, \delta_2) \in S_2\} \\
& \cup \bigcup\{(\rho_1, \delta_1) \,\widetilde{\|}_h\, (\rho_2, l_2, \sigma_2).h_2 : (\rho_1, \delta_1) \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \bigcup\{(\rho_1, \delta_1) \,\widetilde{\|}_h\, (\rho_2, \delta_2) : (\rho_1, \delta_1) \in S_1, (\rho_2, \delta_2) \in S_2\} \\
=\;& \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \,\widetilde{\|}_h\, (\rho_2, l_2, \sigma_2).h_2, \\
& \qquad \sigma_1 \subseteq \rho_2, (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho_2, l_2, \sigma_2).h : h \in (\rho_1, l_1, \sigma_1).h_1 \,\widetilde{\|}_h\, h_2, \\
& \qquad \sigma_2 \subseteq \rho_1, \\
& \qquad (\rho_1, l_1, \sigma_1).h_1 \in S_1, \\
& \qquad (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho, \tau, \sigma).h : \rho_1 = \rho_2 = \rho, l_1 = c_1, l_2 = \overline{c_2}, \\
& \qquad \sigma = \sigma_1 = \rho \cup \{c_1\}, \sigma_2 = \rho \cup \{c_2\}, \sigma \vdash c_2, h \in h_1 \,\widetilde{\|}_h\, h_2, \\
& \qquad (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho, \tau, \sigma).h : \rho_1 = \rho_2 = \rho, l_1 = \overline{c_1}, l_2 = c_2, \\
& \qquad \sigma = \sigma_2 = \rho \cup \{c_2\}, \sigma_1 = \rho \cup \{c_1\}, \sigma \vdash c_1, h \in h_1 \,\widetilde{\|}_h\, h_2, \\
& \qquad (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \,\widetilde{\|}_h\, (\rho_2, \delta_2), (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, \delta_2) \in S_2\} \\
& \cup \{(\rho_2, l_2, \sigma_2).h : h \in (\rho_1, \delta_1) \,\widetilde{\|}_h\, h_2, (\rho_1, \delta_1) \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \bigcup\{(\rho_1, \delta_1) \,\widetilde{\|}_h\, (\rho_2, \delta_2)(\rho_1, \delta_1) \in S_1, (\rho_2, \delta_2) \in S_2\} \\
=\;& \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \,\widetilde{\|}_h\, (\rho_2, l_2, \sigma_2).h_2, \sigma_1 \subseteq \rho_2, (\rho_1, l_1, \sigma_1).h_1 \in S_1, \\
& \qquad (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho_1, l_1, \sigma_1).h : h \in h_1 \,\widetilde{\|}_h\, (\rho_2, \delta_2), (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, \delta_2) \in S_2, \\
& \qquad \sigma_1 \subseteq \rho_2\} \\
& \cup \{(\rho_2, l_2, \sigma_2).h : h \in (\rho_1, l_1, \sigma_1).h_1 \,\widetilde{\|}_h\, h_2, \sigma_2 \subseteq \rho_1, (\rho_1, l_1, \sigma_1).h_1 \in S_1, \\
& \qquad (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \{(\rho_2, l_2, \sigma_2).h : h \in (\rho_1, \delta_1) \,\widetilde{\|}_h\, h_2, (\rho_1, \delta_1) \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2, \\
& \qquad \sigma_2 \subseteq \rho_1\} \\
& \cup \{(\rho, \tau, \sigma).h : \rho_1 = \rho_2 = \rho, l_1 = c_1, l_2 = \overline{c_2}, \\
& \qquad \sigma = \sigma_1 = \rho \cup \{c_1\}, \sigma_2 = \rho \cup \{c_2\}, \sigma \vdash c_2, h \in h_1 \,\widetilde{\|}_h\, h_2, \\
& \qquad (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\}
\end{aligned}
$$

$$
\begin{aligned}
& \cup \{(\rho,\tau,\sigma).h : \rho_1 = \rho_2 = \rho, l_1 = \overline{c_1}, l_2 = c_2, \\
& \qquad \sigma = \sigma_2 = \rho \cup \{c_2\}, \sigma_1 = \rho \cup \{c_1\}, \\
& \qquad \sigma \vdash c_1, h \in h_1 \mathbin{\widetilde{\|}_h} h_2, \\
& \qquad (\rho_1, l_1, \sigma_1).h_1 \in S_1, (\rho_2, l_2, \sigma_2).h_2 \in S_2\} \\
& \cup \bigcup \{(\rho_1,\delta_1) \mathbin{\widetilde{\|}_h} (\rho_2,\delta_2) (\rho_1,\delta_1) \in S_1, (\rho_2,\delta_2) \in S_2\} \\
= \; & \{(\rho_1, l_1, \sigma_1).h : (\rho_1, l_1, \sigma_1).h_1 \in S_1, h_2 \in S_2, h \in h_1 \mathbin{\widetilde{\|}_h} h_2, \\
& \qquad \sigma_1 \subseteq init(h)\} \\
& \{(\rho_2, l_2, \sigma_2).h : (\rho_2, l_2, \sigma_2).h_2 \in S_2, h_1 \in S_1, h \in h_1 \mathbin{\widetilde{\|}_h} h_2, \\
& \qquad \sigma_2 \subseteq init(h)\} \\
& \cup \{(\rho,\tau,\sigma).h : (\rho, c_1, \sigma).h_1 \in S_1, (\rho, \overline{c_2}, \sigma').h_2 \in S_2, h \in h_1 \mathbin{\widetilde{\|}_h} h_2, \\
& \qquad \sigma = \rho \cup \{c_1\}, \sigma' = \rho \cup \{c_2\}, \sigma \vdash c_2\} \\
& \cup \{(\rho,\tau,\sigma).h : (\rho, \overline{c_1}, \sigma').h_1 \in S_1, (\rho, c_2, \sigma).h_2 \in S_2, h \in h_1 \mathbin{\widetilde{\|}_h} h_2, \\
& \qquad \sigma = \rho \cup \{c_2\}, \sigma' = \rho \cup \{c_1\}, \sigma \vdash c_1, \} \\
& \cup \bigcup \{(\rho_1,\delta_1) \mathbin{\widetilde{\|}_h} (\rho_2,\delta_2) (\rho_1,\delta_1) \in S_1, (\rho_2,\delta_2) \in S_2\} \\
= \; & S_1 \mathbin{\|\!\lfloor} S_2 \cup S_2 \mathbin{\|\!\lfloor} S_1 \cup S_1 \otimes S_2 \cup S_1 \odot S_2
\end{aligned}
$$

■

Choice. Choice is modelled as a global choice, namely a goal formed from the choice of two goals can proceed as any of its components. As before care has to be taken to termination marks. The composed goal fails if the two components do so; it succeeds if at least one of the two components does.

Definition 48 *Define* $\tilde{+} : \mathcal{P}(Sdhist) \times \mathcal{P}(Sdhist) \to \mathcal{P}(Sdhist)$ *as the following function: for any subset* S_1, S_2 *of Sdhist,*

$$S_1 \mathbin{\tilde{+}} S_2 = S_1^a \cup S_1^h \cup S_2^a \cup S_1^h \cup S_1^+ \cup S_2^+ \cup (S_1^- \cap S_2^-)$$

4.3.2 Definition

Given the operators $\tilde{;}$, $\widetilde{\|}$, and $\tilde{+}$, defining the denotational semantics amounts to specifying the semantics of the basic constructs *tell* and *ask*, and of the empty goal. This is achieved according to the intuition given by their operational behavior.

Consequently, a $tell(c)$ operation makes a silent step (ρ, τ, ρ) if the input store ρ entails c. It makes an hypothetical step $(\rho, c, \rho \cup \{c\})$ in case $\rho \not\vdash c$. Moreover, in that case, a failure mark $\delta^-(X)$ is registered for any subset X of actions that $tell(c)$ cannot perform i.e. any set disjoint from $\{d \in C : \rho \cup \{c\} \models d, \rho \not\models d\}$.

An $ask(c)$ operation has a dual behaviour. It makes a silent step (ρ, τ, ρ) if the input store ρ entails c. Otherwise, it makes an hypothetical step $(\rho, \overline{c}, \rho \cup \{c\})$ hoping for the presence of a concurrent $tell(d)$ operation with $\rho \cup \{d\} \vdash$

c. Moreover, as before, a mark $\delta^-(X)$ is enclosed for those actions that $ask(c)$ cannot perform i.e. for all the actions except $\overline{c{\uparrow}\rho}$.

All these silent and hypothetical steps are followed by a successfully ending mark (σ, δ^+) however for all the stores $\sigma \sqsupseteq \rho$ only in order to maintain the monotonicity property of denotational histories.

Definition 49 *Define the denotational semantics as the following function $\mathcal{D}_h : Sgoal \to \mathcal{P}(Sdhist)$: for any constraint c, for any goals G_1, G_2,*

$$\begin{array}{rcl}
\mathcal{D}_h(tell(c)) & = & \{(\rho, \tau, \rho).(\sigma, \delta^+) : \rho, \sigma \in Sstore, \rho \vdash c, \rho \subseteq \sigma\} \\
& & \cup \{(\rho, c, \rho \cup \{c\}).(\sigma, \delta^+) : \rho, \sigma \in Sstore, \rho \not\vdash c, \rho \cup \{c\} \subseteq \sigma\} \\
& & \cup \{(\rho, \delta^-(X)) : \rho \in Sstore, \rho \not\vdash c, X \subseteq (Seconst \setminus c{\downarrow}\rho), \\
& & \hfill X \text{ closed wrt } \rho\} \\
\mathcal{D}_h(ask(c)) & = & \{(\rho, \tau, \rho).(\sigma, \delta^+) : \rho, \sigma \in Sstore, \rho \vdash c, \rho \subseteq \sigma\} \\
& & \cup \{(\rho, \overline{c}, \rho \cup \{c\}).(\sigma, \delta^+) : \rho, \sigma \in Sstore, \rho \not\vdash c, \rho \cup \{c\} \subseteq \sigma\} \\
& & \cup \{(\rho, \delta^-(X)) : \rho \in Sstore, \rho \not\vdash c, X \subseteq (Seconst \setminus \overline{c{\uparrow}\rho}), \\
& & \hfill X \text{ closed wrt } \rho\} \\
\mathcal{D}_h(\Delta) & = & \{(\sigma, \delta^+) : \sigma \in Sstore\} \\
\mathcal{D}_h(G_1; G_2) & = & \mathcal{D}_h(G_1) \,\tilde{;}\, \mathcal{D}_h(G_2) \\
\mathcal{D}_h(G_1 \mid\mid G_2) & = & \mathcal{D}_h(G_1) \,\tilde{\mid\mid}\, \mathcal{D}_h(G_2) \\
\mathcal{D}_h(G_1 + G_2) & = & \mathcal{D}_h(G_1) \,\tilde{+}\, \mathcal{D}_h(G_2)
\end{array}$$

4.3.3 Semantics of concurrent tell and ask operations

As an example, the denotational semantics of $tell(c) \mid\mid ask(c)$ is given in the following proposition. It is composed of four kinds of histories

1. those which start in a store ρ entailing c, in which case the $tell(c)$ and $ask(c)$ operations proceed asynchronously and hence are interleaved in the histories;
2. those which start in a store ρ which does not entail c, in which case
   (a) either the $ask(c)$ operation synchronizes with another tell operation and then the $tell(c)$ operation proceeds asynchronously,
   (b) or the $tell(c)$ operation synchronizes with another ask operation, and then the $tell(c)$ operation proceeds asynchronously,
   (c) or the $tell(c)$ and $ask(c)$ operations synchronize together.

Proposition 50 *For any constraint c,*

$\mathcal{D}_h(tell(c) \mid\mid ask(c))$

$$
\begin{aligned}
= \quad & \{(\rho,\tau,\rho).(\rho',\tau,\rho').(\sigma,\delta^+) : \rho,\rho',\sigma \in Sstore, \rho \vdash c, \rho \subseteq \rho', \rho' \subset \sigma\} \\
& \cup \{(\rho,\overline{c},\rho\cup\{c\}).(\rho',\tau,\rho').(\sigma,\delta^+) : \quad \rho,\rho',\sigma \in Sstore, \rho \not\vdash c, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \rho\cup\{c\} \subseteq \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho,c,\rho\cup\{c\}).(\rho',\tau,\rho').(\sigma,\delta^+) : \quad \rho,\rho',\sigma \in Sstore, \rho \not\vdash c, \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \rho\cup\{c\} \subseteq \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho,\tau,\rho\cup\{c\}).(\sigma,\delta^+) : \rho,\sigma \in Sstore, \rho \not\vdash c, \rho\cup\{c\} \subseteq \sigma\}
\end{aligned}
$$

Proof Indeed, by definition 49, $\mathcal{D}_h(tell(c) \mid\mid ask(c))$ collects all the histories obtained by composing the histories of $\mathcal{D}_h(tell(c))$ and $\mathcal{D}_h(ask(c))$ according to the operator $\widetilde{||}$. Nine cases need to be considered for combining each of the three forms of histories of the two denotational semantics.

1. For $\rho,\rho',\sigma,\sigma' \in Sstore$ such that $\rho \vdash c, \rho' \vdash c, \rho \subseteq \sigma, \rho' \subseteq \sigma'$

$$
(\rho,\tau,\rho).(\sigma,\delta^+) \ \widetilde{||}_h \ (\rho',\tau,\rho').(\sigma',\delta^+) =
\begin{cases}
\{(\rho,\tau,\rho).(\rho',\tau,\rho').(\sigma,\delta^+)\}, \text{ if } \sigma=\sigma', \rho \subset \rho' \\
\{(\rho',\tau,\rho').(\rho,\tau,\rho).(\sigma,\delta^+)\}, \text{ if } \sigma=\sigma', \rho' \subset \rho \\
\{(\rho,\tau,\rho).(\rho,\tau,\rho).(\sigma,\delta^+)\} \text{ if } \sigma=\sigma', \rho=\rho' \\
\emptyset, \text{ otherwise}
\end{cases}
$$

2. For $\rho,\rho',\sigma,\sigma' \in Sstore$ such that $\rho \vdash c, \rho' \not\vdash c, \rho \subseteq \sigma, \rho' \subseteq \sigma'$

$$
(\rho,\tau,\rho).(\sigma,\delta^+) \ \widetilde{||}_h \ (\rho',\overline{c},\rho'\cup\{c\}).(\sigma',\delta^+) =
\begin{cases}
\{(\rho',\overline{c},\rho'\cup\{c\}).(\rho,\tau,\rho).(\sigma,\delta^+)\}, \text{ if } \rho'\cup\{c\} \subseteq \rho, \sigma=\sigma' \\
\emptyset, \text{ otherwise}
\end{cases}
$$

Note that $(\rho,\tau,\rho).(\rho',\overline{c},\rho'\cup\{d\}).(\sigma,\delta^+)$ does not appear in the above composition since this would require that $\rho \subseteq \rho'$ and consequently that $\rho' \vdash c$.

3. For $\rho,\rho',\sigma \in Sstore$ such that $\rho \vdash c, \rho' \not\vdash c, \rho \subseteq \sigma$

$$
(\rho,\tau,\rho).(\sigma,\delta^+) \ \widetilde{||}_h \ (\rho',\delta^-(X)) \quad = \quad \emptyset
$$

since the presence of $(\rho,\tau,\rho).(\sigma,\delta^-(X))$ would require that $\sigma = \rho'$, consequently, that $\rho \subseteq \rho'$, and thus that $\rho' \vdash c$.

4. For $\rho,\rho',\sigma,\sigma' \in Sstore$ such that $\rho \not\vdash c, \rho' \vdash c, \rho \subseteq \sigma, \rho' \subseteq \sigma'$,

$$
(\rho,c,\rho\cup\{c\}).(\sigma,\delta^+) \ \widetilde{||} \ (\rho',\tau,\rho').(\sigma',\delta^+) =
\begin{cases}
\{(\rho,c,\rho\cup\{c\}).(\rho',\tau,\rho').(\sigma,\delta^+)\}, \text{ if } \rho\cup\{c\} \subseteq \rho', \sigma=\sigma' \\
\emptyset, \text{otherwise.}
\end{cases}
$$

Note that $(\rho',\tau,\rho').(\rho,c,\rho\cup\{c\}).(\sigma,\delta^+)$ does not appear above since this would require that $\rho \vdash c$.

5. For $\rho, \rho', \sigma, \sigma' \in \mathit{Sstore}$ such that $\rho \not\vdash c$, $\rho' \not\vdash c$, $\rho \subseteq \sigma$, $\rho' \subseteq \sigma'$,

$$(\rho, c, \rho \cup \{c\}).(\sigma, \delta^+) \,\widetilde{|||}_h\, (\rho', \bar{c}, \rho' \cup \{c\}).(\sigma', \delta^+) = \\ \begin{cases} \{(\rho, \tau, \rho \cup \{c\}).(\sigma, \delta^+)\} \text{ if } \rho = \rho', \sigma = \sigma', \\ \emptyset, \text{ otherwise} \end{cases}$$

Note that, an interleaving of the form $(\rho, c, \rho \cup \{c\}).(\rho', \bar{c}, \rho' \cup \{c\}).(\sigma', \delta^+)$ is not possible since this would imply that $\rho \cup \{c\} \subseteq \rho'$ and therefore that $\rho' \vdash c$. Similarly, $(\rho', \bar{c}, \rho' \cup \{c\}).(\rho, c, \rho \cup \{c\}).(\sigma, \delta^+)$ cannot appear.

6. For $\rho, \rho', \sigma \in \mathit{Sstore}$ such that $\rho \not\vdash c$, $\rho \cup \{c\} \subseteq \sigma$, $\rho' \not\vdash c$, $\rho \subseteq \sigma$

$$(\rho, c, \rho \cup \{c\}).(\sigma, \delta^+) \,\widetilde{|||}_h\, (\rho', \delta^-(X')) \quad = \quad \emptyset$$

since getting a non-empty parallel composition requires $\rho' = \sigma$ and therefore that $\rho' \vdash c$.

7. For $\rho, \rho', \sigma' \in \mathit{Sstore}$ such that $\rho \not\vdash c$, $\rho' \vdash c$, $\rho' \subseteq \sigma'$

$$(\rho, \delta^-(X)) \,\widetilde{|||}_h\, (\rho', \tau, \rho').(\sigma', \delta^-(X)) \quad = \quad \emptyset$$

since getting a non-empty set requires $\rho = \sigma'$ and therefore that $\rho \vdash c$.

8. For $\rho, \rho', \sigma' \in \mathit{Sstore}$ such that $\rho \not\vdash c$, $\rho' \not\vdash c$, $\rho' \cup \{c\} \subseteq \sigma'$,

$$(\rho, \delta^-(X)) \,\widetilde{|||}\, (\rho', \bar{c}, \rho' \cup \{c\}).(\sigma', \delta^+) \quad = \quad \emptyset$$

since getting a non-empty set requires $\rho = \sigma'$ and therefore $\rho \vdash c$.

9. For $\rho, \rho' \in \mathit{Sstore}$ such that $\rho \not\vdash c$, $\rho' \not\vdash c$, $X \subseteq (\mathit{Seconst} \setminus c{\downarrow}\rho)$, $X' \subseteq (\mathit{Seconst} \setminus \overline{c{\uparrow}\rho})$,

$$(\rho, \delta^-(X)) \,\widetilde{|||}\, (\rho', \delta^-(X')) = \emptyset$$

since getting a non-empty set requires that $(\mathit{Seconst} \setminus X) \cap \overline{(\mathit{Seconst} \setminus X')} = \emptyset$. However,

$$\begin{aligned} (\mathit{Seconst} \setminus X) \cap \overline{(\mathit{Seconst} \setminus X')} &\supseteq c{\downarrow}\rho \cap \overline{c{\uparrow}\rho} \\ &\supseteq \{c\} \end{aligned}$$

by definitions 21, 49, and the fact that $\rho \not\vdash c$.

Summing up, assuming for the ease of reading that all the occurences of ρ, ρ', and σ are constrained to belong to *Sstore*,

$$
\begin{array}{ll}
\multicolumn{2}{l}{\mathcal{D}_h(tell(c)) \mathbin{\widetilde{\|}} \mathcal{D}_h(ask(c))} \\
= & \{(\rho,\tau,\rho).(\rho',\tau,\rho').(\sigma,\delta^+) : \rho \vdash c, \rho \subset \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho',\tau,\rho').(\rho,\tau,\rho).(\sigma,\delta^+) : \rho' \vdash c, \rho' \subset \rho, \rho' \subseteq \sigma\}, \\
& \cup \{(\rho,\tau,\rho).(\rho,\tau,\rho).(\sigma,\delta^+) : \rho \vdash c, \rho \subseteq \sigma\} \\
& \cup \{(\rho',\overline{c},\rho' \cup \{c\}).(\rho,\tau,\rho).(\sigma,\delta^+) : \rho' \not\vdash c, \rho' \cup \{c\} \subseteq \rho, \rho \subseteq \sigma\} \\
& \cup\, \emptyset \\
& \cup \{(\rho,c,\rho \cup \{c\}).(\rho',\tau,\rho').(\sigma,\delta^+) : \rho \not\vdash c, \rho \cup \{c\} \subseteq \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho,\tau,\rho \cup \{c\}).(\sigma,\delta^+) : \rho \not\vdash c, \rho \cup \{c\} \subseteq \sigma\} \\
& \cup\, \emptyset \\
& \cup\, \emptyset \\
& \cup\, \emptyset \\
& \cup\, \emptyset \\
= & \{(\rho,\tau,\rho).(\rho',\tau,\rho').(\sigma,\delta^+) : \rho \vdash c, \rho \subseteq \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho',\overline{c},\rho' \cup \{c\}).(\rho,\tau,\rho).(\sigma,\delta^+) : \rho' \not\vdash c, \rho' \cup \{c\} \subseteq \rho, \rho \subseteq \sigma\} \\
& \cup \{(\rho,c,\rho \cup \{c\}).(\rho',\tau,\rho').(\sigma,\delta^+) : \rho \not\vdash c, \rho \cup \{c\} \subseteq \rho', \rho' \subseteq \sigma\} \\
& \cup \{(\rho,\tau,\rho \cup \{c\}).(\sigma,\delta^+) : \rho \not\vdash c, \rho \cup \{c\} \subseteq \sigma\}
\end{array}
$$

■

5 Correctness

The semantics $\mathcal{D}_h$ is compositional by construction. It is also correct with respect to the semantics $\mathcal{O}_h$ in the sense that this operational semantics can be obtained from it. In fact, it is sufficient to take from $\mathcal{D}_h$ the real and continuous histories starting in the empty store to get those produced by $\mathcal{O}_h$.

To establish this result, an intermediate model is first introduced. It has the flavour of $\mathcal{D}_h$ in that it manipulates denotational histories. It has also an operational flavour in that its definition uses the transition system of section 3. It is proved to be identical to the denotational semantics $\mathcal{D}_h$. It is then proved to yield $\mathcal{O}_h$ after application of a suitable operator. The expected connection between $\mathcal{O}_h$ and $\mathcal{D}_h$ is derived from these two results.

5.1 The intermediate model $\widetilde{\mathcal{O}}_h$

A few auxiliary concepts are first required. The set $Actions(G,\rho)$ is composed of all the tell and ask operations that G can perform whereas the set $Refusal(G,\rho)$ is composed of all the tell and ask operations that G cannot perform.

Definition 51 *Let G be a goal and ρ be a store. Then define*

$$
\begin{array}{rcl}
Actions(G,\rho) & = & \bigcup \{c{\downarrow}\rho : <G,\rho> \xrightarrow{c} <G',\sigma>, G' \in Sgoal, \sigma \in Sstore, c \in C\} \\
& & \cup \bigcup \{\overline{c{\uparrow}\rho} : <G,\rho> \xrightarrow{\overline{c}} <G',\sigma>, G' \in Sgoal, \sigma \in Sstore, c \in C\} \\
Refusal(G,\rho) & = & Seconst \setminus Actions(G,\rho)
\end{array}
$$

Proposition 52 *For any goal G, for any store ρ, any constraints c_1 and c_2, if $c_1, \overline{c_2} \in Actions(G, \rho)$ then $\rho \not\vdash c_1$ and $\rho \not\vdash c_2$.*

Proof On the one hand, if $c_1 \in Actions(G, \rho)$ then there is a constraint c such that $c_1 \in c{\downarrow}\rho$. By definition, this requires that $\rho \not\vdash c_1$.

On the other hand, if $\overline{c_2} \in Actions(G, \rho)$ then there is a constraint c such that $c_2 \in c{\uparrow}\rho$ and $<G, \rho> \xrightarrow{\bar{c}} <G', \sigma>$, for some goal G' and some store σ. It then follows that $\rho \cup \{c_2\} \vdash c$ and, by proposition 17, that $\rho \not\vdash c$. In these conditions, $\rho \not\vdash c_2$. Otherwise, from $\rho \vdash c_2$ one would derive $\rho \vdash \rho \cup \{c_2\}$ and consequently, from the transitivity of the entailment relation and from $\rho \cup \{c_2\} \vdash c$, one would then derive $\rho \vdash c$. ∎

Corollary 53 *For any goal G, for any store ρ, any constraint c such that $\rho \vdash c$ verifies $c \in Refusal(G, \rho)$ and $\bar{c} \in Refusal(G, \rho)$.*

Proof This is an immediate consequence of proposition 52. ∎

Proposition 54 *For any goal G and any store ρ, the set $Refusal(G, \rho)$ is closed wrt ρ.*

Proof Indeed, by definition 51,

$$\begin{array}{rcll} Refusal(G, \rho) & = & \bigcap \{Seconst \setminus c{\downarrow}\rho : & <G, \rho> \xrightarrow{c} <G', \sigma>, G' \in Sgoal, \\ & & & \sigma \in Sstore, c \in C\} \\ & & \cap \bigcap \{Seconst \setminus \overline{c{\uparrow}\rho} : & <G, \rho> \xrightarrow{\bar{c}} <G', \sigma>, G' \in Sgoal, \\ & & & \sigma \in Sstore, c \in C\} \end{array}$$

Therefore, by propositions 24, 25, and 52, for each constraint c involved in $Refusal(G, \rho)$, the sets $Seconst \setminus (c{\downarrow}\rho)$ and $Seconst \setminus \overline{(c{\uparrow}\rho)}$ are closed wrt ρ. It follows that $Refusal(G, \rho)$ is the intersection of a family of closed sets wrt ρ and is thus, by proposition 26, closed wrt ρ.

∎

Proposition 55 *For any non-empty goals G_1, G_2,*

$$\begin{array}{rcl} Refusal(G_1; G_2, \rho) & = & Refusal(G_1, \rho) \\ Refusal(G_1 + G_2, \rho) & = & Refusal(G_1, \rho) \cap Refusal(G_2, \rho) \\ Refusal(G_1 \parallel G_2, \rho) & = & Refusal(G_1, \rho) \cap Refusal(G_2, \rho) \end{array}$$

Proof Indeed, definition 51 and the transition system of figures 1 and 2 induces the following equalities.

$$
\begin{aligned}
Actions(G_1;G_2,\rho) &= \bigcup\{c{\downarrow}\rho : \; <G_1;G_2,\rho> \xrightarrow{c} <G,\sigma>, G \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&\quad \cup \bigcup\{\overline{c{\uparrow}\rho} : \; <G_1;G_2,\rho> \xrightarrow{\bar{c}} <G,\sigma>, G \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&= \bigcup\{c{\downarrow}\rho : \; <G_1,\rho> \xrightarrow{c} <G^*,\sigma>, G^* \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&\quad \cup \bigcup\{\overline{c{\uparrow}\rho} : \; <G_1,\rho> \xrightarrow{\bar{c}} <G^*,\sigma>, G^* \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&= Actions(G_1,\rho).
\end{aligned}
$$

Moreover, if $\#$ stands either for $+$ or $||$, then

$$
\begin{aligned}
Actions(G_1\# G_2,\rho) &= \bigcup\{c{\downarrow}\rho : \; <G_1\# G_2,\rho> \xrightarrow{c} <G,\sigma>, G \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&\quad \cup \bigcup\{\overline{c{\uparrow}\rho} : \; <G_1\# G_2,\rho> \xrightarrow{\bar{c}} <G,\sigma>, G \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&= \bigcup\{c{\downarrow}\rho : \; <G_i,\rho> \xrightarrow{c} <G^*,\sigma>, G^* \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C, i = 1,2\} \\
&\quad \cup \bigcup\{\overline{c{\uparrow}\rho} : \; <G_i,\rho> \xrightarrow{\bar{c}} <G^*,\sigma>, G^* \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C, i = 1,2\} \\
&= Actions(G_1,\rho) \cup Actions(G_2,\rho).
\end{aligned}
$$

■

Definition 56 *Define the intermediate model* $\widetilde{\mathcal{O}}_h : Sgoal \to \mathcal{P}(Sdhist)$ *as the following function: for any goal* $G \neq \triangle$*,*

$$
\begin{aligned}
\widetilde{\mathcal{O}}_h(\triangle) &= \{(\sigma,\delta^+) : \sigma \in Sstore\} \\
\widetilde{\mathcal{O}}_h(G) &= \{(\rho,l,\sigma).h : <G,\rho> \xrightarrow{l} <G',\sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\} \\
&\quad \cup \{(\rho,\delta^-(X)) : <G,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G,\rho), X \; closed \; wrt \; \rho\}
\end{aligned}
$$

5.2 Relating $\widetilde{\mathcal{O}}_h$ and $\mathcal{D}_h$

$\widetilde{\mathcal{O}}_h$ and $\mathcal{D}_h$ are straightforward to relate for elementary goals. As regards composed goals, a compositional characterization of $\widetilde{\mathcal{O}}_h$ is required. These two properties are the subject of the following propositions.

Proposition 57 *For any constraint c,*

$$\begin{array}{rcl}\tilde{\mathcal{O}}_h(\Delta) & = & \mathcal{D}_h(\Delta)\\ \tilde{\mathcal{O}}_h(tell(c)) & = & \mathcal{D}_h(tell(c))\\ \tilde{\mathcal{O}}_h(ask(c)) & = & \mathcal{D}_h(ask(c))\end{array}$$

Proof Simple verification. ∎

Proposition 58 *For any non-empty goals G_1 and G_2,*

$$\begin{array}{rcl}\tilde{\mathcal{O}}_h(G_1;G_2) & = & \tilde{\mathcal{O}}_h(G_1)\,\tilde{;}\,\tilde{\mathcal{O}}_h(G_2)\\ \tilde{\mathcal{O}}_h(G_1+G_2) & = & \tilde{\mathcal{O}}_h(G_1)\,\tilde{+}\,\tilde{\mathcal{O}}_h(G_2)\\ \tilde{\mathcal{O}}_h(G_1 \parallel G_2) & = & \tilde{\mathcal{O}}_h(G_1)\,\tilde{\parallel}\,\tilde{\mathcal{O}}_h(G_2)\end{array}$$

Proof

SEQUENTIAL COMPOSITION. In this case, the proof proceeds by induction on the size of G_1 and by a slight generalization allowing G_1 to be empty, in which case $G_1;G_2$ is understood to be G_2. For the base case where $size(G_1) = 0$, we observe that obviously $\tilde{\mathcal{O}}_h(G_2) = \tilde{\mathcal{O}}_h(\Delta)\,\tilde{;}\,\tilde{\mathcal{O}}_h(G_2)$. Let us now study the induction case. By definition, $\tilde{\mathcal{O}}_h(G_1;G_2)$ is composed of two sets, which we consider successively.

On the one hand, the transition rules, the induction hypothesis, and proposition 43 successively lead to the following equalities.

$$\begin{array}{l}\{(\rho,l,\sigma).h : <G_1;G_2,\rho> \xrightarrow{l} <G',\sigma>, h \in \tilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\}\\ \quad = \quad \{(\rho,l,\sigma).h : <G_1,\rho> \xrightarrow{l} <G_1',\sigma>, h \in \tilde{\mathcal{O}}_h(G_1';G_2), \sigma \subseteq init(h)\}\\ \quad = \quad \{(\rho,l,\sigma).h : <G_1,\rho> \xrightarrow{l} <G_1',\sigma>, h \in \tilde{\mathcal{O}}_h(G_1')\,\tilde{;}\,\tilde{\mathcal{O}}_h(G_2), \sigma \subseteq init(h)\}\\ \quad = \quad \{(\rho,l,\sigma).h : \begin{array}[t]{l}<G_1,\rho> \xrightarrow{l} <G_1',\sigma>, h \in h_1 \,\tilde{;}_r\, h_2, h_1 \in \tilde{\mathcal{O}}_h(G_1'),\\ h_2 \in \tilde{\mathcal{O}}_h(G_2), \sigma \subseteq init(h)\}\end{array}\\ \quad = \quad \{(\rho,l,\sigma).h_1 : <G_1,\rho> \xrightarrow{l} <G_1',\sigma>, h_1 \in \tilde{\mathcal{O}}_h(G_1'), \sigma \subseteq init(h)\}\,\tilde{;}\,\tilde{\mathcal{O}}_h(G_2)\end{array}$$

On the other hand, the transition rules and proposition 43 justify the following equalities.

$$\begin{array}{l}\{(\rho,\delta^-(X)) : <G_1;G_2,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1;G_2,\rho)\}\\ \quad = \quad \{(\rho,\delta^-(X)) : <G_1,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1,\rho)\}\\ \quad = \quad \{(\rho,\delta^-(X)) : <G_1,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1,\rho)\}\,\tilde{;}\,\tilde{\mathcal{O}}_h(G_2)\end{array}$$

Summing up,

$$
\begin{aligned}
\widetilde{\mathcal{O}}_h(G_1;G_2) &= \{(\rho,l,\sigma).h_1 : \begin{array}[t]{l} <G_1,\rho> \xrightarrow{l} <G'_1,\sigma>, h_1 \in \widetilde{\mathcal{O}}_h(G'_1), \\ \sigma \subseteq init(h)\} \end{array} \tilde{;}\, \widetilde{\mathcal{O}}_h(G_2) \\
&\quad \cup \{(\rho,\delta^-(X)) : <G_1,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1,\rho)\} \,\tilde{;}\, \widetilde{\mathcal{O}}_h(G_2) \\
&= \widetilde{\mathcal{O}}_h(G_1) \,\tilde{;}\, \widetilde{\mathcal{O}}_h(G_2)
\end{aligned}
$$

CHOICE. The proof for the choice operator is conducted by inspecting again the two sets composing $\widetilde{\mathcal{O}}_h(G)$ for a non-empty goal G. On the one hand, the transition rules induce the following equalities.

$$
\begin{aligned}
&\{(\rho,l,\sigma).h : <G_1+G_2,\rho> \xrightarrow{l} <G',\sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\} \\
&\quad = \cup_{i=1}^{2}\{(\rho,l,\sigma).h : <G_i,\rho> \xrightarrow{l} <G',\sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\} \\
&\quad = (\widetilde{\mathcal{O}}_h(G_1))^a \cup (\widetilde{\mathcal{O}}_h(G_1))^h \cup (\widetilde{\mathcal{O}}_h(G_2))^a \cup (\widetilde{\mathcal{O}}_h(G_2))^h
\end{aligned}
$$

On the other hand,

$$
\begin{aligned}
&\{(\rho,\delta^-(X)) : <G_1+G_2,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1+G_2,\rho)\} \\
&\quad = \{(\rho,\delta^-(X)) : \begin{array}[t]{l} <G_1,\rho> \stackrel{\tau}{\not\longrightarrow}, <G_2,\rho> \stackrel{\tau}{\not\longrightarrow}, \\ X \subseteq Refusal(G_1,\rho) \cap Refusal(G_2,\rho)\} \end{array} \\
&\quad = \{(\rho,\delta^-(X)) : <G_1,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1,\rho)\} \\
&\qquad \cap \{(\rho,\delta^-(X)) : <G_2,\rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_2,\rho)\} \\
&\quad = (\widetilde{\mathcal{O}}_h(G_1))^- \cap (\widetilde{\mathcal{O}}_h(G_2))^-
\end{aligned}
$$

To conclude, we observe that, since G_1 and G_2 are assumed to be non-empty, then $\widetilde{\mathcal{O}}_h(G_1)$ and $\widetilde{\mathcal{O}}_h(G_2)$ contain no element of the form (σ,δ^+) (see definition 56) and consequently,

$$
(\widetilde{\mathcal{O}}_h(G_1))^+ = (\widetilde{\mathcal{O}}_h(G_2))^+ = \emptyset.
$$

The following equalities follow therefrom.

$$
\begin{aligned}
\widetilde{\mathcal{O}}_h(G_1+G_2) &= (\widetilde{\mathcal{O}}_h(G_1))^a \cup (\widetilde{\mathcal{O}}_h(G_1))^h \cup (\widetilde{\mathcal{O}}_h(G_2))^a \cup (\widetilde{\mathcal{O}}_h(G_2))^h \\
&\quad \cup [(\widetilde{\mathcal{O}}_h(G_1))^- \cap (\widetilde{\mathcal{O}}_h(G_2))^-] \\
&= (\widetilde{\mathcal{O}}_h(G_1))^a \cup (\widetilde{\mathcal{O}}_h(G_1))^h \cup (\widetilde{\mathcal{O}}_h(G_2))^a \cup (\widetilde{\mathcal{O}}_h(G_2))^h \\
&\quad \cup (\widetilde{\mathcal{O}}_h(G_1))^+ \cup (\widetilde{\mathcal{O}}_h(G_2))^+ \cup [(\widetilde{\mathcal{O}}_h(G_1))^- \cap (\widetilde{\mathcal{O}}_h(G_2))^-] \\
&= \widetilde{\mathcal{O}}_h(G_1) \,\tilde{+}\, \widetilde{\mathcal{O}}_h(G_2)
\end{aligned}
$$

PARALLEL COMPOSITION. The proof proceeds by induction on the size of the composed goal $G_1 \parallel G_2$ and by a slight generalization allowing G_1 and G_2 to be empty, in which case, $\triangle \parallel \triangle$ is understood to be $\triangle$ and, for G non-empty, $\triangle \parallel G$ and $G \parallel \triangle$ are understood to be G. . The base case where G_1 or G_2 are empty is proved by the following equalities, which are straightforward to established: $\widetilde{\mathcal{O}}_h(G) = \widetilde{\mathcal{O}}_h(\triangle) \,\widetilde{\|}\, \widetilde{\mathcal{O}}_h(G) = \widetilde{\mathcal{O}}_h(G) \,\widetilde{\|}\, \widetilde{\mathcal{O}}_h(\triangle)$.

The induction case is established as follows. On the one hand, applying successively the transition rules, the induction hypothesis, and definitions 46 and 56 yield the following equalities.

$$
\begin{aligned}
&\{(\rho, l, \sigma).h : <G_1 \parallel G_2, \rho> \xrightarrow{l} <G', \sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\} \\
&\quad = \quad \{(\rho, l, \sigma).h : <G_1, \rho> \xrightarrow{l} <G'_1, \sigma>, h \in \widetilde{\mathcal{O}}_h(G'_1 \parallel G_2), \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, l, \sigma).h : <G_2, \rho> \xrightarrow{l} <G'_2, \sigma>, h \in \widetilde{\mathcal{O}}_h(G_1 \parallel G'_2), \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, \tau, \sigma).h : <G_1, \rho> \xrightarrow{c} <G'_1, \sigma>, <G_2, \rho> \xrightarrow{\overline{d}} <G'_2, \sigma'>, \\
&\qquad\qquad\qquad h \in \widetilde{\mathcal{O}}_h(G'_1 \parallel G'_2), c, d \in C, \sigma \vdash d, \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, \tau, \sigma).h : <G_2, \rho> \xrightarrow{c} <G'_2, \sigma>, <G_1, \rho> \xrightarrow{\overline{d}} <G'_1, \sigma'>, \\
&\qquad\qquad\qquad h \in \widetilde{\mathcal{O}}_h(G'_1 \parallel G'_2), c, d \in C, \sigma \vdash d, \sigma \subseteq init(h)\} \\
&\quad = \quad \{(\rho, l, \sigma).h : <G_1, \rho> \xrightarrow{l} <G'_1, \sigma>, h \in h_1 \,\widetilde{\|}_h\, h_2, h_1 \in \widetilde{\mathcal{O}}_h(G'_1), \\
&\qquad\qquad h_2 \in \widetilde{\mathcal{O}}_h(G_2), \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, l, \sigma).h : <G_2, \rho> \xrightarrow{l} <G'_2, \sigma>, h \in h_1 \,\widetilde{\|}_h\, h_2, h_1 \in \widetilde{\mathcal{O}}_h(G_1), \\
&\qquad\qquad h_2 \in \widetilde{\mathcal{O}}_h(G'_2), \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, \tau, \sigma).h : <G_1, \rho> \xrightarrow{c} <G'_1, \sigma>, <G_2, \rho> \xrightarrow{\overline{d}} <G'_2, \sigma'>, \\
&\qquad\qquad\qquad h \in h_1 \,\widetilde{\|}_h\, h_2, h_1 \in \widetilde{\mathcal{O}}_h(G'_1), h_2 \in \widetilde{\mathcal{O}}_h(G'_2), \\
&\qquad\qquad\qquad c, d \in C, \sigma \vdash d, \sigma \subseteq init(h)\} \\
&\qquad\quad \cup \{(\rho, \tau, \sigma).h : <G_2, \rho> \xrightarrow{c} <G'_2, \sigma>, <G_1, \rho> \xrightarrow{\overline{d}} <G'_1, \sigma'>, \\
&\qquad\qquad\qquad h \in h_1 \,\widetilde{\|}_h\, h_2, h_1 \in \widetilde{\mathcal{O}}_h(G'_1), h_2 \in \widetilde{\mathcal{O}}_h(G'_2), \\
&\qquad\qquad\qquad c, d \in C, \sigma \vdash d, \sigma \subseteq init(h)\} \\
&\quad = \quad \widetilde{\mathcal{O}}_h(G_1) \,\|\!\!\_\, \widetilde{\mathcal{O}}_h(G_2) \cup \widetilde{\mathcal{O}}_h(G_2) \,\|\!\!\_\, \widetilde{\mathcal{O}}_h(G_1) \cup \widetilde{\mathcal{O}}_h(G_1) \otimes \widetilde{\mathcal{O}}_h(G_2)
\end{aligned} \tag{3}
$$

On the other hand, let us establish that

$$
\begin{aligned}
\{(\rho, \delta^-(X)) : <G_1 \parallel G_2, \rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1 \parallel G_2, \rho)\} \\
= \widetilde{\mathcal{O}}_h(G_1) \odot \widetilde{\mathcal{O}}_h(G_2)
\end{aligned} \tag{4}
$$

To that end, let us first observe that G_1 and G_2 being non-empty, by definition, $\widetilde{\mathcal{O}}_h(G_1)$ and $\widetilde{\mathcal{O}}_h(G_2)$ just contain elements of the form $(\sigma, \delta^-(Y))$ as ending constructs (ρ, δ). We thus have to prove that

$$
\begin{aligned}
&\{(\rho, \delta^-(X)) : <G_1 \parallel G_2, \rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1 \parallel G_2, \rho)\} \\
&\quad = \bigcup \{(\sigma_1, \delta^-(Y_1)) \,\widetilde{\|}_h\, (\sigma_2, \delta^-(Y_2)) : <G_1, \sigma_1> \stackrel{\tau}{\not\longrightarrow}, Y_1 \subseteq Refusal(G_1, \sigma_1), \\
&\qquad\qquad <G_2, \sigma_2> \stackrel{\tau}{\not\longrightarrow}, Y_2 \subseteq Refusal(G_2, \sigma_2)\}
\end{aligned}
$$

In view of definition 44, this amounts to establishing that $S_1 = S_2$ where

$$\begin{array}{rcl} S_1 & = & \{(\rho, \delta^-(X)) : <G_1 \mid\mid G_2, \rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G_1 \mid\mid G_2, \rho)\} \\ S_2 & = & \{(\sigma, \delta^-(Y)) : Y \subseteq Y_1 \cap Y_2, \\ & & \qquad <G_1, \sigma> \stackrel{\tau}{\not\longrightarrow}, Y_1 \subseteq Refusal(G_1, \sigma), \\ & & \qquad <G_2, \sigma> \stackrel{\tau}{\not\longrightarrow}, Y_2 \subseteq Refusal(G_2, \sigma), \\ & & \qquad (Seconst \setminus Y_1) \cap \overline{(Seconst \setminus Y_2)} = \emptyset\} \end{array}$$

Indeed, let $(\rho, \delta^-(X))$ be in S_1. The following properties thus hold:

$$<G_1 \mid\mid G_2, \rho> \stackrel{\tau}{\not\longrightarrow} \text{ and } X \subseteq Refusal(G_1 \mid\mid G_2, \rho)$$

The first property and the transition rules induce that

$$<G_1, \sigma> \stackrel{\tau}{\not\longrightarrow} \text{ and } <G_2, \sigma> \stackrel{\tau}{\not\longrightarrow}$$

The second property and proposition 55 imply that

$$X \subseteq Refusal(G_1, \rho) \cap Refusal(G_2, \rho)$$

To establish that $(\rho, \delta^-(X))$ belongs to S_2, it thus remains to be shown that

$$(Seconst \setminus Refusal(G_1, \rho)) \cap \overline{(Seconst \setminus Refusal(G_2, \rho))} = \emptyset$$

If this is not the case, then one of the two following cases hold:

1. there is $e \in Seconst \setminus Refusal(G_1, \rho)$ such that $\overline{e} \in Seconst \setminus Refusal(G_2, \rho)$
2. there is $e \in Seconst \setminus Refusal(G_2, \rho)$ such that $\overline{e} \in Seconst \setminus Refusal(G_1, \rho)$.

Let us consider the first case, the other being treated similarly. Then, by definition 51, there are constraints c_1, c_2, goals G_1', G_2', and stores σ, σ' such that

- $\rho \cup \{c_1\} \vdash e$, $\rho \not\vdash e$, and $<G_1, \rho> \xrightarrow{c_1} <G_1', \sigma>$;
- $\rho \cup \{e\} \vdash c_2$ and $<G_2, \rho> \xrightarrow{\overline{c_2}} <G_2', \sigma'>$.

Consequently, $\rho \cup \{c_1\} \vdash c_2$ and, by transition rule (P),

$$<G_1 \mid\mid G_2, \rho> \xrightarrow{\tau} <G_1' \mid\mid G_2', \sigma>,$$

which is not possible.

Let us now consider $(\sigma, \delta^-(Y))$ in S_2. Then there are Y_1 and Y_2 such that $Y \subseteq Y_1 \cap Y_2$ and $Y_i \subseteq Refusal(G_i, \sigma)$ for $i = 1, 2$. It follows therefrom and from proposition 55 that

$$\begin{array}{rcl} Y & \subseteq & Y_1 \cap Y_2 \\ & \subseteq & Refusal(G_1, \sigma) \cap Refusal(G_2, \sigma) \\ & = & Refusal(G_1 \mid\mid G_2, \sigma) \end{array}$$

Let us now prove that $<G_1 \mid\mid G_2, \sigma> \stackrel{\tau}{\not\longrightarrow}$. Otherwise, as $(\sigma, \delta^-(Y)) \in S_2$ and thus $<G_i, \sigma> \stackrel{\tau}{\not\longrightarrow}$ for $i = 1, 2$, there should exist goals G'_1, G'_2, constraints c, d, and stores σ', σ'' such that one of the two following situations hold:

1. $<G_1, \sigma> \stackrel{c}{\longrightarrow} <G'_1, \sigma'>$ and $<G_2, \sigma> \stackrel{\overline{d}}{\longrightarrow} <G'_2, \sigma''>$ and $\sigma' \vdash d$
2. $<G_2, \sigma> \stackrel{c}{\longrightarrow} <G'_2, \sigma'>$ and $<G_1, \sigma> \stackrel{\overline{d}}{\longrightarrow} <G'_1, \sigma''>$ and $\sigma' \vdash d$.

Assume the first situation hold; the other one is treated similarly. Then, by proposition 17, $\sigma' = \sigma \cup \{c\}$. Moreover, $d \in \mathit{Seconst} \setminus \mathit{Refusal}(G_1, \sigma)$ and $\overline{d} \in \mathit{Seconst} \setminus \mathit{Refusal}(G_2, \sigma)$. Hence, thanks to the inclusions $Y_i \subseteq \mathit{Refusal}(G_i, \sigma)$, $i = 1, 2$, one has $d \in (\mathit{Seconst} \setminus Y_1) \cap \overline{(\mathit{Seconst} \setminus Y_2)}$. This is impossible since, by hypothesis, this intersection is empty.

The thesis then follows from equalities 3, 4 and from proposition 47:

$$
\begin{aligned}
\widetilde{\mathcal{O}}_h(G_1 \mid\mid G_2) &= \{(\rho, l, \sigma).h : <G_1 \mid\mid G_2, \rho> \stackrel{l}{\longrightarrow} <G', \sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \\
&\qquad\qquad \sigma \subseteq \mathit{init}(h)\} \\
&\quad \cup \{(\rho, \delta^-(X)) : <G_1 \mid\mid G_2, \rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq \mathit{Refusal}(G_1 \mid\mid G_2, \rho)\} \\
&= \widetilde{\mathcal{O}}_h(G_1) \,\underline{\|}\, \widetilde{\mathcal{O}}_h(G_2) \cup \widetilde{\mathcal{O}}_h(G_2) \,\underline{\|}\, \widetilde{\mathcal{O}}_h(G_1) \cup \widetilde{\mathcal{O}}_h(G_1) \otimes \widetilde{\mathcal{O}}_h(G_2) \\
&\quad \cup \widetilde{\mathcal{O}}_h(G_1) \odot \widetilde{\mathcal{O}}_h(G_2) \\
&= \widetilde{\mathcal{O}}_h(G_1) \,\widetilde{\|}\, \widetilde{\mathcal{O}}_h(G_2)
\end{aligned}
$$

■

We are now in a position to relate the semantics $\widetilde{\mathcal{O}}_h$ and $\mathcal{D}_h$.

Theorem 59 *For any goal G, $\widetilde{\mathcal{O}}_h(G) = \mathcal{D}_h(G)$.*

Proof The proof proceeds by induction on the size of G. The base cases for $G = \triangle$, $G = \mathit{tell}(c)$, and $G = \mathit{ask}(c)$ are established in proposition 57. The induction case follows from the following reasoning. Let # denote one of the sequential, parallel, and choice operators. Then, applying successively proposition 58, the induction hypothesis, and definition 49 yield

$$
\begin{aligned}
\widetilde{\mathcal{O}}_h(G_1 \# G_2) &= \widetilde{\mathcal{O}}_h(G_1) \widetilde{\#} \widetilde{\mathcal{O}}_h(G_2) \\
&= \mathcal{D}_h(G_1) \widetilde{\#} \mathcal{D}_h(G_2) \\
&= \mathcal{D}_h(G_1 \# G_2)
\end{aligned}
$$

■

5.3 Relating $\widetilde{\mathcal{O}}_h$ and $\mathcal{O}_h$

As already announced, the abstraction function relating the operational and denotational semantics basically consists of taking the real and continuous histories starting in the empty store and of selecting some elements in the steps. It turns out that this function allows also to relate the intermediate model $\widetilde{\mathcal{O}}_h$ with the operational semantics $\mathcal{O}_h$. It is specified in the following definition. A recursive characterization is then provided to ease further reasonings.

Definition 60 *Define $\alpha : \mathcal{P}(Shist) \to \mathcal{P}(Shist)$ as follows: for any subset $S \subseteq Shist$,*

$$\alpha(S) = \{\widetilde{h} : h \in S, h \textit{ real and continuous}, init(h) = \epsilon\}.$$

Definition 61 *Define $\alpha_{hist} : Sstore \to Shist \to \mathcal{P}(Shist)$ as follows: for any $\rho, \sigma, \nu \in Sstore$, any $l \in Slabel$, any $X \subseteq Seconst$, any $h \in Shist$,*

$$\begin{aligned}
\alpha_{hist}(\nu)((\rho, \delta^+)) &= \begin{cases} \{\delta^+\}, & \textit{if } \rho = \nu \\ \emptyset, & \textit{otherwise} \end{cases} \\
\alpha_{hist}(\nu)((\rho, \delta^-(X))) &= \begin{cases} \{\delta^-\}, & \textit{if } \rho = \nu \\ \emptyset, & \textit{otherwise} \end{cases} \\
\alpha_{hist}(\nu)((\rho, l, \sigma).h) &= \begin{cases} \{\sigma.h' : h' \in \alpha_{hist}(\sigma)(h)\}, & \textit{if } \rho = \nu \textit{ and } l = \tau \\ \emptyset, & \textit{otherwise} \end{cases}
\end{aligned}$$

Definition 62 *Define $\alpha_r : Sstore \to \mathcal{P}(Shist) \to \mathcal{P}(Shist)$ as the natural lifting of α_{hist} to sets: for any $\sigma \in Sstore$, any $S \subseteq Shist$,*

$$\alpha_r(\sigma)(S) = \cup_{h \in S}\ \alpha_{hist}(\sigma)(h)$$

Proposition 63 $\alpha_r(\epsilon) = \alpha$

Proof Simple verification. ∎

The relationship between the semantics $\widetilde{\mathcal{O}}_h$ and $\mathcal{O}_h$ by means of the operator α is first established on the basis of their recursive characterizations.

Proposition 64 *For any store σ and any goal G, $\alpha_r(\sigma)(\widetilde{\mathcal{O}}_h(G)) = \mathcal{O}_h^r(G)(\sigma)$*

Proof The proof is conducted by induction on the size of G. If $G = \triangle$, then

$$\widetilde{\mathcal{O}}_h(G) = \{(\sigma, \delta^+) : \sigma \in Sstore\}$$

and consequently for any store ν,

$$\alpha_r(\nu)(\widetilde{\mathcal{O}}_h(G)) = \{\delta^+\}.$$

The thesis then results in this case from the fact that, by definition 19,

$$\mathcal{O}_h^r(\triangle)(\nu) = \{\delta^+\}.$$

If $G \neq \emptyset$, then let us first observe that, for a given $\nu \in Sstore$, successively applying definition 61 and the induction hypothesis and taking into account the monotonicity of elements of *Sstep* yield

$$\begin{array}{ll} & \alpha_r(\nu)(\{(\rho, l, \sigma).h : <G, \rho> \stackrel{l}{\longrightarrow} <G', \sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\}) \\ = & \{\sigma.h' : <G, \nu> \stackrel{\tau}{\longrightarrow} <G', \sigma>, h' \in \alpha_r(\sigma)(\widetilde{\mathcal{O}}_h(G'))\} \\ = & \{\sigma.h' : <G, \nu> \stackrel{\tau}{\longrightarrow} <G', \sigma>, h' \in \mathcal{O}_h^r(G')(\sigma)\} \\ = & \left\{ \begin{array}{l} \mathcal{O}_h^r(G)(\nu), \text{ if there are } G', \nu' \text{ such that } <G, \nu> \stackrel{\tau}{\longrightarrow} <G', \sigma> \\ \emptyset, \text{ otherwise} \end{array} \right. \end{array}$$

Moreover,

$$\alpha_r(\nu)(\{(\rho, \delta^-(X)) : <G, \rho> \stackrel{\tau}{\not\longrightarrow}, X \subseteq Refusal(G, \rho)\}) \quad = \quad \left\{ \begin{array}{l} \{\delta^-\}, \text{ if } <G, \nu> \stackrel{\tau}{\not\longrightarrow} \\ \emptyset, \text{ otherwise.} \end{array} \right.$$

Two cases need now to be considered. On the one hand, if $<G, \nu> \stackrel{\tau}{\not\longrightarrow}$ then

$$\alpha_r(\nu)(\widetilde{\mathcal{O}}_h(G)) = \emptyset \cup \{\delta^-\} = \mathcal{O}_h^r(G)(\nu)$$

On the other hand, if there are G', ν' such that $<G, \nu> \stackrel{\tau}{\longrightarrow} <G', \nu'>$ then

$$\alpha_r(\nu)(\widetilde{\mathcal{O}}_h(G)) = \mathcal{O}_h^r(G)(\nu) \cup \emptyset = \mathcal{O}_h^r(G)(\nu)$$

Summing up, in the two cases, $\alpha_r(\nu)(\widetilde{\mathcal{O}}_h(G)) = \mathcal{O}_h^r(G)(\nu)$, which establishes the thesis. ■

$\widetilde{\mathcal{O}}_h$ and $\mathcal{O}_h$ can now be related directly by means of α.

Theorem 65 *For any goal G, $\alpha(\widetilde{\mathcal{O}}_h(G)) = \mathcal{O}_h(G)$*

Proof In fact, successively combining propositions 63, 64, and 20 yields

$$\begin{array}{rcl} \alpha(\widetilde{\mathcal{O}}_h(G)) & = & \alpha_r(\epsilon)(\widetilde{\mathcal{O}}_h(G)) \\ & = & \mathcal{O}_h^r(G)(\epsilon) \\ & = & \mathcal{O}_h(G) \end{array}$$

■

5.4 Relating $\mathcal{D}_h$ and $\mathcal{O}_h$

We are now in a position to relate the semantics $\mathcal{O}_h$ and $\mathcal{D}_h$.

Theorem 66 *For any goal G, $\mathcal{O}_h(G) = \alpha(\mathcal{D}_h(G))$.*

Proof This result follows from theorems 59 and 65. ■

Note that, as a corollary of this proposition an operational history $\sigma_1.\cdots.\sigma_n.\delta$ corresponds to a continuous and real denotational history starting in the empty store: $(\epsilon, \tau, \sigma_1).(\sigma_1, \tau, \sigma_2).\cdots.(\sigma_n, \delta')$. This correspondance is biunivoque if $\delta = \delta^+$. In that case, $\delta' = \delta^+$. It is essentially biunivoque if $\delta = \delta^-$. In that case, the prefix up to the last element (σ_n, δ') is fixed and the only varying part is δ' which takes the form $\delta^-(X)$ for some set X.

6 First steps characterization

The denotational semantics can also be characterized in terms of the operational semantics as follows.

Proposition 67 *Let A be a goal, l be a label, and ρ, τ be stores such that $\mathcal{D}_h(A)[(\rho, l, \sigma)] \neq \emptyset$. Moreover, let $B_1, \ldots, B_m$ be all the goals such that*

$$<A, \rho> \xrightarrow{l} <B, \sigma>$$

Then,

$$\mathcal{D}_h(A)[(\rho, l, \sigma)] = \mathcal{D}_h(B_1)\Uparrow\sigma \cup \cdots \cup \mathcal{D}_h(B_m)\Uparrow\sigma.$$

Proof Indeed, theorem 59 and definition 56 justify the following equalities:

$$\begin{array}{rcl}
\mathcal{D}_h(A)[(\rho, l, \sigma)] & = & \widetilde{\mathcal{O}}_h(A)[(\rho, l, \sigma)] \\
& = & \{h : <A, \rho> \xrightarrow{l} <B, \sigma>, h \in \widetilde{\mathcal{O}}_h(B), \sigma \subseteq init(h)\} \\
& = & (\widetilde{\mathcal{O}}_h(B_1) \cup \cdots \cup \widetilde{\mathcal{O}}_h(B_m)) \cap \{h \in Sdhist : \sigma \subseteq init(h)\} \\
& = & \widetilde{\mathcal{O}}_h(B_1)\Uparrow\sigma \cup \cdots \cup \widetilde{\mathcal{O}}_h(B_m)\Uparrow\sigma \\
& = & \mathcal{D}_h(B_1)\Uparrow\sigma \cup \cdots \cup \mathcal{D}_h(B_m)\Uparrow\sigma
\end{array}$$

■

Note that $\mathcal{D}_h(B_1) \cup \cdots \cup \mathcal{D}_h(B_m)$ is almost $\mathcal{D}_h(B_1 + \cdots + B_m)$. They actually differ by the treatment of immediately failing computations: all of them are registered in $\mathcal{D}_h(B_1) \cup \cdots \cup \mathcal{D}_h(B_m)$ while only those common to B_1, …, B_m appear in the denotational semantics of $B_1 + \cdots + B_m$.

It is possible to further characterize the first steps of denotational histories in terms of the presence of tell and ask primitives in goals. For instance, an history $(\rho, c, \sigma).h$ in $\mathcal{D}_h(G)$ can only occur if G contains a $tell(c)$ primitive in a place ready to be executed.

The notion of top-context, already introduced in definition 15, specifies those places. The following proposition provides a few characterizations, to be intuitively read as follows.

- (ρ, τ, σ) as a first step requires either the synchronization of a tell primitive with an ask primitive or the asynchronous reduction of one of these primitives. Furthermore, the two cases can be distinguished as follows. Asynchronous reductions occur when the constraint under consideration is entailed by the input store ρ. Otherwise synchronous reduction has to occur.

- (ρ, c, σ) as a first step requires a *tell*(c) primitive to be reduced first.

- $(\rho, \overline{c}, \sigma)$ as a first step requires an *ask*(c) primitive to be reduced first.

Proposition 68 *Let G be a goal and ρ, σ be two different stores such that $\rho \subseteq \sigma$. Let h be an history.*

i) If $(\rho, \tau, \rho).h \in \mathcal{D}_h(G)$, then $G = tc[tell(c)]$ or $G = tc[ask(c)]$ for a constraint c such that $\rho \vdash c$.
ii) If $(\rho, c, \sigma).h \in \mathcal{D}_h(G)$, then $G = tc[tell(c)]$ for a constraint c such that $\rho \not\vdash c$ and $\sigma = \rho \cup \{c\}$.
iii) If $(\rho, \overline{c}, \sigma).h \in \mathcal{D}_h(G)$, then $G = tc[ask(c)]$ for a constraint c such that $\rho \not\vdash c$ and $\sigma = \rho \cup \{c\}$.
iv) If $(\rho, \tau, \sigma).h \in \mathcal{D}_h(G)$ with $\rho \neq \sigma$, then $G = tc[tell(c), ask(d)]$ for some constraints c and d such that $\rho \not\vdash c$, $\rho \not\vdash d$, $\sigma = \rho \cup \{c\}$, $\sigma \vdash d$.

Proof The proposition is a direct consequence of proposition 16 and of theorem 59. ■

7 Completion

Pursuing further the operational characterization of the denotational steps leads to the following property.

Proposition 69 *Let G be a goal and let h be an history of $\mathcal{D}_h(G)$. Then, h^c is a real history of $\mathcal{D}_h(G \mid\mid Co(h))$.*

Proof Using theorem 59, the proof consists in a simple recursive reasoning on the size of goals. ■

8 Coherence properties

We now turn to a few properties grouped under the name of coherence. They are all satisfied by the denotational semantics and will be required later for sets of denotational histories.

8.1 Finiteness

The impact of a goal on stores can be shown finite in the following sense.

Proposition 70 *For any goal G, the set* $diff(\mathcal{D}_h(G))$ *is finite.*

Proof Using theorem 59, the proof consists in a simple recursive reasoning on the size of goals. ■

The image-finiteness property (see proposition 14) can be rephrased at the denotational level as follows.

Definition 71 *For any* $S \subseteq Sdhist$*, any store* ρ*, define* $Act(S, \rho) = \{l : S[(\rho, l, \sigma)] \neq \emptyset\}$.

Proposition 72 *For any goal G, any store* σ*, the set* $Act(\mathcal{D}_h(G), \sigma)$ *is finite.*

Proof The proposition results from theorem 59 and proposition 14. ■

Definition 73 *The set* $S \subseteq Sdhist$ *is* action-finite at level n *if for any prefix* $p \in (Sstore \times Slabel \times Sstore)^n$ *of length n, for any store* σ*, the set* $Act(S[p], \sigma)$ *is finite. It is* uniformly action-finite *if it is action-finite at any level.*

Proposition 74 *For any goal G, the semantics* $\mathcal{D}_h(G)$ *is uniformly action-finite.*

Proof The proposition results from theorem 59 and proposition 14. ■

Proposition 75 *For any prefix* $p \in (Sstore \times Slabel \times Sstore)^n$*, for any set* $S \subseteq Sdhist$*, if S is uniformly action-finite then so does* $S[p]$.

Proof Simple verification. ■

8.2 Extensibility

The denotational semantics $\mathcal{D}_h(G)$ of a goal G is extensible in the following sense.

Proposition 76 *For any goal G, any stores* ρ*,* $\rho_1, \ldots, \rho_n$*,* $\sigma_1, \ldots, \sigma_n$*, for any labels* $l_1, \ldots, l_n$*,*

1) *there is a continuous history in* $\mathcal{D}_h(G)$ *starting in* ρ
2) *if* $\mathcal{D}_h(G)[(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)] \neq \emptyset$ *and if* $\sigma_n \subseteq \rho$*, then there is a continuous history in* $\mathcal{D}_h(G)[(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)]$ *starting in* ρ.

Proof Indeed, the proposition is easily verified for $\widetilde{\mathcal{O}}_h(G)$. The thesis then results from theorem 59. ■

Definition 77 *A set $S \subseteq Sdhist$ is* extensible from a store ν *if for any stores ρ, ρ_1, ..., ρ_n, σ_1, ..., $\sigma_n \supseteq \nu$, for any labels $l_1, \ldots, l_n$,*

1) *there is a continuous history in S starting in ρ*
2) *if $S[(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)] \neq \emptyset$ and if $\sigma_n \subseteq \rho$ then there is a continuous history in $S[(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)]$ starting in ρ.*

Proposition 78 *For any goal G, for any store σ, the semantics $\mathcal{D}_h(G)$ is extensible from σ.*

Proof This is a direct consequence of proposition 76. ∎

Proposition 79 *For any set $S \subseteq Sdhist$ of histories extensible from a store ν, for any stores ρ_1, ..., ρ_n, σ_1, ..., σ_n, for any labels l_1, ..., l_n, then, if it is not empty, the set $S[(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)]$ is extensible from any store μ such that $\mu \supseteq \nu$ and $\mu \supseteq \sigma_n$.*

Proof Simple verification. ∎

8.3 Consistency

This section aims at establishing that one may always add or substract auxiliary constraints in histories in the case these constraints do not influence the entailment of constraints told or asked by the considered goal.

Definition 80 *Let S_1 and S_2 be two sets of constraints. S_1 is* independent *from S_2 if the two following conditions hold:*

1. *for any constraint $c \in S_1$ and any store σ, if $\sigma \cup S_2 \vdash c$ then $\sigma \vdash c$;*
2. *for any store $\rho \subseteq S_1$, for any constraint c, if $\rho \cup S_2 \vdash c$ then $\rho \vdash c$ or exclusively $S_2 \vdash c$.*

Alternatively, S_2 is said not to influence *S_1.*

Lemma 81 *Let G be a goal and Sc and Su be two sets of constraints. Assume Su contains all the tell and ask primitives of G and Sc does not influence Su. Then,*

$$<G, \rho> \xrightarrow{l} <G', \sigma>$$

iff

$$<G, \rho \cup Sc> \xrightarrow{l} <G', \sigma \cup Sc>$$

Proof The lemma results directly from proposition 16 by noting that, for any store ρ, and any constraint c of Su, one has $\rho \vdash c$ iff $\rho \cup Sc \vdash c$ by the independence of Su wrt to Sc. ∎

Lemma 82 *Let Sc be a set of constraints which does not influence the set of constraints Su. Let $\rho \subseteq Su$ be a store and $c \in Su$ be a constraint. Then,*

$$\begin{array}{rcl} c\uparrow(\rho \cup Sc) & = & c\uparrow\rho \\ c\downarrow(\rho \cup Sc) & = & c\downarrow\rho \end{array}$$

Proof The first equation is established by the following equalities:

$$\begin{array}{rcl} c\uparrow(\rho \cup Sc) & = & \{e : \rho \cup Sc \cup \{e\} \vdash c\} \\ & = & \{e : \rho \cup \{e\} \vdash c\} \\ & = & c\uparrow\rho. \end{array}$$

The first equality and the third one results immediately from definition 21. The second one requires a few explanations. On the one hand, if $\rho \cup \{e\} \vdash c$ then $\rho \cup Sc \cup \{e\} \vdash c$. On the other hand, by definition 80, if $\rho \cup Sc \cup \{e\} \vdash c$ then $\rho \cup \{e\} \vdash c$.

The second equation is established by the following ones:

$$\begin{array}{rcl} c\downarrow(\rho \cup Sc) & = & \{e : \rho \cup Sc \cup \{c\} \vdash e, \rho \cup Sc \not\vdash e\} \\ & = & \{e : \rho \cup \{c\} \vdash e, \rho \not\vdash e\} \\ & = & c\downarrow\rho \end{array}$$

Again the first and third equalities results directly from definition 21. The second equality is proved by demonstrating the two corresponding inclusions. On the one hand, to prove

$$\{e : \rho \cup Sc \cup \{c\} \vdash e, \rho \cup Sc \not\vdash e\} \subseteq \{e : \rho \cup \{c\} \vdash e, \rho \not\vdash e\}$$

let us first note that $\rho \cup Sc \not\vdash e$ implies $\rho \not\vdash e$. Moreover, since $\rho \cup \{c\} \subseteq Su$, then by definition 80, for any constraint e such that $\rho \cup Sc \cup \{c\} \vdash e$, either $\rho \cup \{c\} \vdash e$ or (exclusively) $Sc \vdash e$. However, if additionnally e is such that $\rho \cup Sc \not\vdash e$ then $Sc \not\vdash e$.

On the other hand, to establish the converse inclusion

$$\{e : \rho \cup \{c\} \vdash e, \rho \not\vdash e\} \subseteq \{e : \rho \cup Sc \cup \{c\} \vdash e, \rho \cup Sc \not\vdash e\}$$

let us observe that if $\rho \cup \{c\} \vdash e$ then $\rho \cup \{c\} \cup Sc \vdash e$. Moreover, by definition 80, it follows that $Sc \not\vdash e$. To conclude, it remains to be shown that $\rho \cup Sc \not\vdash e$. Indeed, otherwise, by definition 80, either $\rho \vdash e$ or exclusively $Sc \vdash e$. However, by hypothesis, $\rho \not\vdash e$ and, as just established, $Sc \not\vdash e$.

■

Lemma 83 *Let G be a goal and ρ be a store. Assume the set Su contains the constraints occurring in the tell and ask primitives of G and the set Sc does not influence Su. Assume moreover that $\rho \subseteq Su$. Then*

1) $Actions(G, \rho \cup Sc) = Actions(G, \rho)$
2) $Refusal(G, \rho \cup Sc) = Refusal(G, \rho)$.

Proof The second part of the proposition follows directly from the first. This part is in turn proved as follows by successively using definition 51 and lemma 82 – with for this lemma, the fact that the constraints c mentioned are in one of the tell and ask primitives of G by proposition 16:

$$\begin{aligned}
Actions(G, \rho \cup Sc) &= \bigcup \{c{\downarrow}(\rho \cup Sc) : \ <G, \rho \cup Sc> \xrightarrow{c} <G', \sigma>, G' \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&\quad \cup \bigcup \{\overline{c{\uparrow}(\rho \cup Sc)} : \ <G, \rho \cup Sc> \xrightarrow{\bar{c}} <G', \sigma>, G' \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&= \bigcup \{c{\downarrow}\rho : \ <G, \rho> \xrightarrow{c} <G', \sigma>, G' \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&\quad \cup \bigcup \{\overline{c{\uparrow}\rho} : \ <G, \rho> \xrightarrow{\bar{c}} <G', \sigma>, G' \in Sgoal, \\
&\qquad\qquad \sigma \in Sstore, c \in C\} \\
&= Actions(G, \rho).
\end{aligned}$$

■

Definition 84 *A set of histories $S \subseteq Shist$ is* consistent wrt to the store γ *if for any superset Su of $diff(S)$, for any set of constraints Sc not influencing Su, the following properties hold:*

1. *for any labels $l_1, \ldots, l_n$ and for any stores $\rho_1, \ldots, \rho_{n+1}, \sigma_1, \ldots, \sigma_n$ included in Su and such that $\rho_i \subseteq \sigma_i \subseteq \rho_{i+1}$ for $i = 1, \cdots, n$, and $\gamma \subseteq \rho_i$, $\gamma \subseteq \sigma_j$, for $i = 1, \cdots, n+1$ and $j = 1, \cdots, n$,*

$$\begin{aligned}
&(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta^+) \in S \text{ iff} \\
&\quad (\rho_1 \cup Sc, l_1, \sigma_1 \cup Sc). \cdots .(\rho_n \cup Sc, l_n, \sigma_n \cup Sc).(\rho_{n+1} \cup Sc, \delta^+) \in S
\end{aligned}$$

2. *for any labels $l_1, \ldots, l_n$, for any stores $\rho_1, \ldots, \rho_{n+1}, \sigma_1, \ldots, \sigma_n$ included in Su, and such that $\rho_i \subseteq \sigma_i \subseteq \rho_{i+1}$ for $i = 1, \cdots, n$, and $\gamma \subseteq \rho_i$, $\gamma \subseteq \sigma_j$, for $i = 1, \cdots, n+1$ and $j = 1, \cdots, n$, for any set X closed wrt ρ_{n+1},*

$$\begin{aligned}
&(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta^-(X)) \in S \text{ iff} \\
&\quad (\rho_1 \cup Sc, l_1, \sigma_1 \cup Sc). \cdots .(\rho_n \cup Sc, l_n, \sigma_n \cup Sc). \\
&\qquad (\rho_{n+1} \cup Sc, \delta^-(\widehat{X}^{(\rho_{n+1} \cup Sc)})) \in S
\end{aligned}$$

3. *for any labels $l_1, \ldots, l_n$, for any stores $\rho_1, \ldots, \rho_{n+1}, \sigma_1, \ldots, \sigma_n$ included in Su, and such that $\rho_i \subseteq \sigma_i \subseteq \rho_{i+1}$ for $i = 1, \cdots, n$, and $\gamma \subseteq \rho_i$, $\gamma \subseteq \sigma_j$, for $i = 1, \cdots, n+1$ and $j = 1, \cdots, n$, for any set X closed wrt $\rho_{n+1} \cup Sc$,*

$$\begin{aligned}
&(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta^-(\widehat{X}^{\rho_{n+1}})) \in S \text{ iff} \\
&\quad (\rho_1 \cup Sc, l_1, \sigma_1 \cup Sc). \cdots .(\rho_n \cup Sc, l_n, \sigma_n \cup Sc).(\rho_{n+1} \cup Sc, \delta^-(X)) \in S
\end{aligned}$$

4. *for any store ρ, σ, any label l, any history h such that $\sigma \cup Sc \subseteq init(h)$,*

$$(\rho, l, \sigma).h \in S \textit{ iff } (\rho \cup Sc, l, \sigma \cup Sc).h \in S.$$

It is uniformly consistent wrt γ *if it is consistent wrt γ and if, for any stores ρ, σ, γ' such that $\gamma \subseteq \rho$, $\sigma \subseteq \gamma'$, and for any label l if, it is not empty, the set $S[(\rho, l, \sigma)]$ is uniformly consistent wrt γ'.*

Proposition 85 *For any goal G, the set $\mathcal{D}_h(G)$ is consistent wrt to the empty store ϵ.*

Proof In view of theorem 59, the proposition is established for $\widetilde{\mathcal{O}}_h(G)$ instead of $\mathcal{D}_h(G)$ by an inductive reasoning of the size of G and by noting, for ask and tell primitives, say $ask(c)$ and $tell(c)$ respectively, that what determines their semantics is the verification of $\rho \vdash c$ where ρ is the current store. However, by hypothesis, this is independent from the set Sc. More precisely, the proof is conducted as follows.

BASE CASE. If $size(G) = 0$, then $G = \Delta$ and thus, by definition 56,

$$\widetilde{\mathcal{O}}_h(G) = \{(\sigma, \delta^+) : \sigma \in Sstore\}.$$

Therefore, points 2, 3, and 4 of definition 80 are straightfowardly verified in this case. Moreover, the following equivalence trivially holds:

$$(\sigma, \delta^+) \in \widetilde{\mathcal{O}}_h(G) \text{ iff } (\sigma \cup Sc, \delta^+) \in \widetilde{\mathcal{O}}_h(G).$$

INDUCTIVE CASE. If $size(G) > 0$ then $G \neq \Delta$ and by definition 56,

$$\begin{array}{rcl}\widetilde{\mathcal{O}}_h(G) & = & \{(\rho, l, \sigma).h : <G, \rho> \xrightarrow{l} <G', \sigma>, h \in \widetilde{\mathcal{O}}_h(G'), \sigma \subseteq init(h)\} \\ & & \cup \{(\rho, \delta^-(X)) : <G, \rho> \overset{\tau}{\not\longrightarrow}, X \subseteq Refusal(G, \rho), X \text{ closed wrt } \rho\}\end{array}$$

Proof of point 1. The case where $n = 0$ does not occur by definition of $\widetilde{\mathcal{O}}_h(G)$. For $n > 0$, the thesis results from the following equivalences:

$(\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta^+) \in \widetilde{\mathcal{O}}_h(G)$

iff by definition 56, for some goal G'

$<G, \rho_1> \xrightarrow{l_1} <G', \sigma_1>$
and
$(\rho_2, l_2, \sigma_2). \cdots .(\rho_n, l_n, \sigma_n).(\rho_{n+1}, \delta^+) \in \widetilde{\mathcal{O}}_h(G')$

iff by lemma 81 and the induction hypothesis

$<G, \rho_1 \cup Sc> \xrightarrow{l_1} <G', \sigma_1 \cup Sc>$
and
$(\rho_2 \cup Sc, l_2, \sigma_2 \cup Sc). \cdots .(\rho_n \cup Sc, l_n, \sigma_n \cup Sc).(\rho_{n+1} \cup Sc, \delta^+) \in \widetilde{\mathcal{O}}_h(G')$

iff by definition 56

$$(\rho_1 \cup Sc, l_1, \sigma_1 \cup Sc). \cdots .(\rho_n \cup Sc, l_n, \sigma_n \cup Sc).(\rho_{n+1} \cup Sc, \delta^+) \in \widetilde{\mathcal{O}}_h(G)$$

Proof of point 2. The case where $n > 0$ is treated as for point 1. For $n = 0$, if $(\rho, \delta^-(X)) \in \widetilde{\mathcal{O}}_h(G)$ then, the following properties hold:

1. $<G, \rho> \overset{\tau}{\not\longrightarrow}$,
2. $X \subseteq Refusal(G, \rho)$,
3. X closed wrt ρ.

In these conditions, by lemma 81, $<G, \rho \cup Sc> \overset{\tau}{\not\longrightarrow}$ and, by proposition 32, $\widehat{X}^{\rho \cup Sc}$ is closed wrt $\rho \cup Sc$. To conclude, it remains to be established that

$$\widehat{X}^{\rho \cup Sc} \subseteq Refusal(G, \rho \cup Sc).$$

Indeed, by lemma 83, $Refusal(G, \rho \cup Sc) = Refusal(G, \rho)$. Moreover, by proposition 54, $Refusal(G, \rho \cup Sc)$ is closed wrt $\rho \cup Sc$. Applying proposition 31 then leads to the desired inclusion.

Conversly, assume the following properties hold:

1. $<G, \rho \cup Sc> \overset{\tau}{\not\longrightarrow}$,
2. $\widehat{X}^{\rho \cup Sc} \subseteq Refusal(G, \rho \cup Sc)$,
3. X closed wrt ρ.

Then, by lemma 81, $<G, \rho> \overset{\tau}{\not\longrightarrow}$. Moreover, as X is closed wrt ρ, it remains to be established that $X \subseteq Refusal(G, \rho)$. Indeed, by lemma 83, $Refusal(G, \rho \cup Sc) = Refusal(G, \rho)$. The thesis then amounts to establishing that $X \subseteq Refusal(G, \rho \cup Sc)$. By proposition 28,

$$X \setminus \{\overline{x} \in X : \rho \cup Sc \vdash x\} \subseteq \widehat{X}^{\rho \cup Sc}$$

and thus

$$X \setminus \{\overline{x} \in X : \rho \cup Sc \vdash x\} \subseteq Refusal(G, \rho \cup Sc).$$

Moreover, by corollary 53,

$$\{\overline{x} \in X : \rho \cup Sc \vdash x\} \subseteq Refusal(G, \rho \cup Sc).$$

Summing up, $X \subseteq Refusal(G, \rho \cup Sc)$, as required.

Proof of point 3. The proof for the case where $n > 0$ proceeds as for point 1. For $n = 0$, if $(\rho \cup Sc, \delta^-(X)) \in \widetilde{\mathcal{O}}_h(G)$ then the following properties hold:

1. $<G, \rho \cup Sc> \stackrel{\tau}{\not\longrightarrow}$,

2. $X \subseteq Refusal(G, \rho \cup Sc)$,

3. X closed wrt $\rho \cup Sc$.

In these conditions, by lemma 81, $<G, \rho> \stackrel{\tau}{\not\longrightarrow}$ and, by proposition 32, $\widehat{X}^{\rho}$ is closed wrt ρ. To conclude, we note that, by lemma 83, $Refusal(G, \rho \cup Sc) = Refusal(G, \rho)$ and that, by proposition 54, $Refusal(G, \rho)$ is closed wrt ρ. Consequentely, by proposition 31, $\widehat{X}^{\rho} \subseteq Refusal(G, \rho)$. If follows from definition 56 that $(\rho, \delta^{-}(\widehat{X}^{\rho})) \in \widetilde{\mathcal{O}}_h(G)$.

The converse implication is established as for point 2.

Proof of point 4. Applying successively definition 56, lemma 81, and definition 56 again leads to the following equivalences, which establish the thesis:

$$(\rho, l, \sigma).h \in \widetilde{\mathcal{O}}_h(G)$$

iff

for some goal G',
$$<G, \rho> \stackrel{l}{\longrightarrow} <G', \sigma> \text{ and } h \in \widetilde{\mathcal{O}}_h(G')$$

iff

for some goal G',
$$<G \cup Sc, \rho> \stackrel{l}{\longrightarrow} <G' \cup Sc, \sigma> \text{ and } h \in \widetilde{\mathcal{O}}_h(G')$$

iff

$$(\rho \cup Sc, l, \sigma \cup Sc).h \in \widetilde{\mathcal{O}}_h(G)$$

■

Lemma 86 *The union of (uniformly) consistent sets is (uniformly) consistent.*

Proof Simple verification. ■

Lemma 87 *For any store γ_1, γ_2, γ_3 such that $\gamma_1 \subseteq \gamma_2 \subseteq \gamma_3$, if S is (uniformly) consistent wrt γ_1 then $S \Uparrow \gamma_2$ is (uniformly) consistent wrt γ_3.*

Proof Simple verification. ■

Proposition 88 *For any goal G, $\mathcal{D}_h(G)$ is uniformly consistent.*

Proof The proof is conducted by an inductive reasoning on the size of G.

I. BASE CASE. If $G = \triangle$, then $\mathcal{D}_h(G) = \{(\sigma, \delta^+) : \sigma \in Sstore\}$. Consequently, for any stores ρ, σ, γ such that $\gamma \subseteq \sigma$ and for any label l, the set $\mathcal{D}_h(G)[(\rho, l, \sigma)]$ is empty. Moreover, by proposition 85, $\mathcal{D}_h(G)$ is consistent wrt to ϵ. The denotation $\mathcal{D}_h(G)$ is thus uniformly consistent.

II. INDUCTIVE CASE. By proposition 85, $\mathcal{D}_h(G)$ is consistent wrt ϵ. Moreover, for any stores ρ, σ, γ such that $\gamma \supseteq \sigma \supseteq \rho$ and for any label l, by proposition 67,

$$\mathcal{D}_h(G)[(\rho, l, \sigma)] = \mathcal{D}_h(B_1)\Uparrow\sigma \cup \cdots \cup \mathcal{D}_h(B_m)\Uparrow\sigma.$$

where $B_1, \ldots, B_m$ are all the goals B such that

$$<G, \rho> \xrightarrow{l} <B, \sigma>$$

By proposition 13, the size of the B_i's is strictly less than that of G. The induction hypothesis can thus be applied, which leads to the fact that the sets $\mathcal{D}_h(B_i)$ are uniformly consistent wrt ϵ. It follows from lemma 87 that the sets $\mathcal{D}_h(B_i)\Uparrow\sigma$ are uniformly consistent wrt γ. Hence so is $\mathcal{D}_h(G)[(\rho, l, \sigma)]$ by lemma 86. ∎

8.4 Deadlock disjointness

Failure sets of denotational semantics may be characterized as follows.

Proposition 89 *For any goal G, for any store σ, for any set $X \subseteq Seconst$ closed wrt σ, one has $(\sigma, \delta^-(X)) \in \mathcal{D}_h(G)$ if and only if $<G, \sigma> \stackrel{\tau}{\not\longrightarrow}$ and $X \cap Actions(G, \sigma) = \emptyset$.*

Proof The proposition directly results from theorem 59 and definition 56. ∎

This characterization of failure sets may be generalized to sets of histories as follows.

Definition 90 *The set $S \subseteq Sdhist$ satisfies the* disjointness deadlock condition *at level n iff for any prefix $p \in (Sstore \times Slabel \times Sstore)^n$ of length n and for any store σ such that $S[p] \neq \emptyset$ and $(\sigma, \delta^-(X)) \in S[p]$ for some set X closed wrt σ, there are non-empty goals $G_1, \ldots, G_m$ ($m > 0$) such that*

1. *$<G_i, \sigma> \stackrel{\tau}{\not\longrightarrow}$*
2. *for any $Y \subseteq Seconst$ closed wrt σ, one has $(\sigma, \delta^-(Y)) \in S[p]$ iff $Y \cap Actions(G_i, \sigma) = \emptyset$ for some $i = 1, \cdots, m$.*

The set S satisfies the uniform disjointness deadlock condition *if it satisfies the disjointness deadlock condition at any level.*

Proposition 91 *For any goal G, the denotational semantics $\mathcal{D}_h(G)$ satisfies the uniform disjointness deadlock condition.*

Proof The case of the empty prefix is treated in proposition 89. Let $p = (\rho_1, l_1, \sigma_1). \cdots .(\rho_n, l_n, \sigma_n)$ be a non-empty prefix of length n such that $\mathcal{D}_h(G)[p] \neq \emptyset$. Then by theorem 59 and definition 56, there are goals $G_1^*, \ldots, G_n^*$ such that

$$<G, \rho_1> \xrightarrow{l_1} <G_1^*, \sigma_1>$$
$$\cdots$$
$$<G_{n-1}^*, \rho_n> \xrightarrow{l_n} <G_n^*, \sigma_n>$$

Moreover, by proposition 14, the set of sequences $(G_1^*, \cdots, G_n^*)$ of such goals is finite. Take as goals $G_1, \ldots, G_m$ all the possible non-empty target goals G_n^* such that $<G_n^*, \sigma> \overset{\tau}{\not\longrightarrow}$. They verify the disjointness deadlock condition. Indeed, on the one hand, if $(\sigma, \delta^-(X)) \in \mathcal{D}_h(G)[p]$ for some $X \subseteq Seconst$, then as $\mathcal{D}_h(G)[p] = \widetilde{\mathcal{O}}_h(G)[p]$ by theorem 59, there are goals $G_1', \ldots, G_n'$ such that

$$<G, \rho_1> \xrightarrow{l_1} <G_1', \sigma_1>$$
$$\cdots$$
$$<G_{n-1}', \rho_n> \xrightarrow{l_n} <G_n', \sigma_n>$$
$$(\sigma, \delta^-(X)) \in \widetilde{\mathcal{O}}_h(G_n')$$

By definition 56, this implies that $G_n' \neq \Delta$, that $<G_n', \sigma> \overset{\tau}{\not\longrightarrow}$, and that $X \cap Actions(G_n', \sigma) = \emptyset$. The goal G_n' is thus one of the goals G_i's for which the disjointness deadlock condition holds. Moreover, note that this reasoning proves in addition that the set of G_i's is non-empty ie $m > 0$.

On the other hand, suppose that $X \subseteq Seconst$ is such that $X \cap Actions(G_i, \sigma) = \emptyset$, for some $i = 1, \cdots, m$. Then, by definition 56, $(\sigma, \delta^-(X)) \in \widetilde{\mathcal{O}}_h(G_i)$ and thus, by construction, $(\sigma, \delta^-(X)) \in \mathcal{D}_h(G)[p]$. ■

Proposition 92 *For any prefix $p \in (Sstore \times Slabel \times Sstore)^n$, for any set $S \subseteq Sdhist$, if S enjoys the uniform disjointness condition, then so does $S[p]$.*

Proof The proposition directly results from definition 90. ■

8.5 Coherence

The above properties are summarized in the following notion of coherence.

Definition 93 *A set of denotational histories is called* coherent *if its set $diff(S)$ is finite, if it is extensible, if it is uniformly action-finite, if it is uniformly consistent, and if it enjoys the uniform disjointness deadlock condition.*

9 Full abstraction

The next property to ask is whether $\mathcal{D}_h$ contains the least information necessary to be compositional and correct. That corresponds to a full abstraction result. This result is so involved that it deserves a complete section, which is done in this section.

9.1 Definitions

Rephrased in other terms, full abstraction consists in requiring that the denotational semantics of two goals are identical iff, from the operational point of view, the two goals behave identically even when they are composed with other goals in all the possible manners. This form of composition is provided by the classical notion of context, recalled in the following definition.

Definition 94 *Let $\square$ be a fresh symbol. Define the set of contexts Scontext by the following rules, where G is a goal.*

$$C \quad ::= \quad \square \mid G \mid C \,;\, G \mid G \,;\, C \mid C \parallel G \mid G \parallel C \mid C + G \mid G + C$$

The application of a context C to a goal G is defined as the new goal obtained by replacing the place holder $\square$ in C, if any, by G. This is subsequently denoted as $C[G]$.

Definition 95 *The semantics $\mathcal{D}_h$ is fully abstract with respect to the semantics $\mathcal{O}_h$ iff the following property holds: for any goals G_1, G_2, the following assertions are equivalent*

i) for any context C, $\mathcal{O}_h(C[G_1]) = \mathcal{O}_h(C[G_2])$;

ii) $\mathcal{D}_h(G_1) = \mathcal{D}_h(G_2)$.

The semantics $\mathcal{O}_h$ and $\mathcal{D}_h$ being connected by the operator α we introduce the following notation as the support of subsequent reasonings.

Notation 96 *For any set $S \subseteq Sdhist$, define $\lfloor S \rfloor$ as the largest set S' such that $\alpha(S') = \alpha(S)$.*

Note that the difference between histories of S and $\lfloor S \rfloor$ consists in failure marks only according to the remark under theorem 66.

9.2 Intuition

The compositional property of $\mathcal{D}_h$ together with theorem 66 establish the implication $(ii) \Rightarrow (i)$ of definition 95. It thus remains to prove the converse $(i) \Rightarrow (ii)$. To that end, we shall proceed by contraposition. Given two goals G_1, G_2 such that

$$\mathcal{D}_h(G_1) \quad \neq \quad \mathcal{D}_h(G_2) \tag{5}$$

we shall construct a context C such that

$$\mathcal{O}_h(C[G_1]) \quad \neq \mathcal{O}_h(C[G_2])$$

The two semantics reporting sets, the construction amounts to constructing from a denotational history h of one goal, say G_1, which is not in the denotation of the other G_2, a context C and an operational history of $C[G_1]$ not of $C[G_2]$. In view of the relation between $\mathcal{O}_h$ and $\mathcal{D}_h$ as shown by α in theorem 66, this amounts to establishing the existence of a real and continuous denotational history, starting in ϵ, which is in $\mathcal{D}_h(C[G_1])$ and not in $\lfloor \mathcal{D}_h(C[G_2]) \rfloor$. To that end, following [7], we shall construct from h a new history h' and an goal T such that h' is in the denotational semantics of $G_1 \mid\mid T$ and not in $\lfloor \mathcal{D}_h(G_2 \mid\mid T) \rfloor$.

The proof basically proceeds by induction on the length of h.

In the base case, h takes the form (σ, δ) with δ being either δ^+ or $\delta^-(X)$. The tester T then basically construct a real and continuous sequence yielding σ from the initial store ϵ in a way that, on the one hand, prevents G_1 and G_2 to do any intermediary step, and, on the other hand, forces G_1 and G_2 to do the last step (σ, δ). By hypothesis, this is possible for G_1 and not for G_2 if $\delta = \delta^+$ or if $\delta = \delta^-$ but $\mathcal{D}_h(G_2)$ contains no ending steps $(\sigma, \delta^-(Y))$ whatever Y is. In the case where $(\sigma, \delta^-(Y)) \in \mathcal{D}_h(G_2)$, then a test can be appended to T which forbids G_2 to do the last step but yet allow G_1 to do it.

In the non basic case, h takes the form $(\rho, l, \sigma).h^*$ for some history h^*. Two cases are possible: either there is no history starting by (ρ, l, σ) in $\mathcal{D}_h(G_2)$ or those which start by (ρ, l, σ) cannot end by h^*. In the first case, the proof proceeds as in the base case. In the second case, the proof uses induction. However, the induction should be applied for h^* in $\mathcal{D}_h(G_1)[(\rho, l, \sigma)]$ and not in $\mathcal{D}_h(G_2)[(\rho, l, \sigma)]$. As stated by proposition 67, these sets turned out to be basically but not exactly the denotations $\mathcal{D}_h(G_1')$ and $\mathcal{D}_h(G_2')$, of some goals G_1' and G_2'. We shall consequently generalize slightly the induction to sets of denotational histories. This extension being discarded here for the sake of simplicity, we thus apply the induction hypothesis for h^*, G_1' and G_2'. It points out a tester T' and an history h'' which is in $\mathcal{D}_h(G_1' \mid\mid T')$ and not in $\lfloor \mathcal{D}_h(G_2' \mid\mid T') \rfloor$. From there we should construct a tester T and an history h''' in $\mathcal{D}_h(G_1 \mid\mid T)$ and not in $\lfloor \mathcal{D}_h(G_2 \mid\mid T) \rfloor$. Basically, the step (ρ, l, σ) has to be done before h'' and since h''' needs to be continuous, h'' has to start in a possibly non empty store. Hence, we have to generalize the theorem and construct in general from h an history h' which start in any initial store. Given this generalization, the tester T basically consists of first making the steps necessary to produce ρ from the given initial store, then of making an auxiliary transition from σ to some σ' chosen so as to ensure that G_1 and G_2 have to do the step (ρ, l, σ), and finally consists of T'.

A slight extension is needed when $l \neq \tau$. In that case, in order to guarantee h' to be real, the tester T is requested to perform the complementary step $(\rho, \bar{l}, \sigma)$ before making the transition from σ to σ'.

9.3 Auxiliary concepts

The above intuition points out an auxiliary task which consists of making by an auxiliary goal the steps necessary to produce a given target store σ from a given initial store ρ. These steps are subsequently achieved by means of the following goal $G^{S;Sc}_{\rho\to\sigma}$.

Notation 97 *Let $c_1, \ldots, c_m$ be contraints and let ρ and σ be two stores such that $\rho \subseteq \sigma$. Then, we denote by $\sigma \setminus \rho \asymp \{c_1, \cdots, c_m\}$ the following properties*

i) $\sigma = \rho \cup \{c_1, \cdots, c_m\}$

ii) $\rho \cup \{c_{\nu_1}, \cdots, c_{\nu_k}\} \not\vdash c_i$, *for any (possibly empty) subset $\{c_{\nu_1}, \cdots, c_{\nu_k}\}$ of the c_j's and for any $c_i \notin \{c_{\nu_1}, \cdots, c_{\nu_k}\}$*

Note that, as a consequence of point *ii* above, $\rho \not\vdash c_i$, for $i = 1, \cdots, m$

Definition 98 *Let S be a finite set of constraints, and ρ and σ be two stores, such that $\rho \subseteq \sigma$. Consider*

- *$c_1, \ldots, c_m$ constraints such that $\sigma \setminus \rho \asymp \{c_1, \cdots, c_m\}$*
- *$a_1, \ldots, a_m$ constraints such that*
  1. *$\sigma \cup S \cup \{a_1, \cdots, a_i\} \not\vdash a_{i+1}$, for $i = 1, \cdots, m$*
  2. *for any store α and any constraint $c \in S$, if $\alpha \cup \{a_1, \cdots, a_m\} \vdash c$ then $\alpha \vdash c$,*
  3. *for any store $\gamma \subseteq S$, for any constraint c, if $\gamma \cup \{a_1, \cdots, a_m\} \vdash c$ then $\gamma \vdash c$ or exclusively $\{a_1, \cdots, a_m\} \vdash c$.*

Then, abusing language by forgetting about the constraints c_i's in the notation, we denote by $G^{S;\{a_1,\cdots,a_m\}}_{\rho\to\sigma}$ the following goal

$$\begin{gathered}(tell(c_1) \mid\mid ask(c_1)); (tell(a_1) \mid\mid ask(a_1));\\ \cdots\\ (tell(c_m) \mid\mid ask(c_m)); (tell(a_m) \mid\mid ask(a_m));\end{gathered}$$

Moreover, we note by $\Sigma^{S;\{a_1,\cdots,a_m\}}_{\rho\to\sigma}$ the associated sequence of states

$$\begin{gathered}(\rho_0, \tau, \gamma_1).(\gamma_1, \tau, \rho_1).\\ \cdots\\ (\rho_{i-1}, \tau, \gamma_i).(\gamma_i, \rho_i).\\ \cdots\\ (\rho_{m-1}, \tau, \gamma_m).(\gamma_m, \rho_m)\end{gathered}$$

where

$$\begin{aligned}&\rho_0 = \rho\\ &\gamma_i = \rho_{i-1} \cup \{c_i\},\ i = 1, \cdots, m\\ &\rho_i = \gamma_i \cup \{a_i\},\ i = 1, \cdots, m\end{aligned}$$

Note, in particular, that $\rho_m = \sigma \cup \{a_1, \cdots, a_m\}$. Furthermore, in the case where $\rho = \sigma$, the goal $G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}$ reduces to the empty goal and the sequence $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}$ is the empty sequence.

Finally, in the remainder of this paper, the use of the above notations implicitly assume the above hypotheses on ρ, σ, the c_i's, and the a_i's.

Obviously, $G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}$ can perform the history $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.(\rho_m, \delta^+)$. If S is suitably chosen, it also has the property of being responsible for making the steps of $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}$ when placed in parallel with another goal.

Proposition 99 *Let ρ and σ be two stores such that $\rho \subset \sigma$. Let A be a goal and let S be the set of constraints present in the tell and ask primitives of A.*

1) *Any history $h = \Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h'$ of $\mathcal{D}_h(G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}} \mid\mid A)$ is from the set $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.(\gamma, \delta^+) \,\widetilde{\|}_h\, h_a$ for some store γ and some history $h_a \in \mathcal{D}_h(A)$.*
2) *For any goal B, any history $h = \Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h'$ of $\mathcal{D}_h((G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}; B) \mid\mid A)$ is from the set $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h_b \,\widetilde{\|}_h\, h_a$ for some histories $h_a \in \mathcal{D}_h(A)$ and $h_b \in \mathcal{D}_h(B)$.*

Proof Let us establish the first part of the proposition, the proof of the other part being similar.

By definition 49, if h is in $\mathcal{D}_h(G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}} \mid\mid A)$, there are $h_1 \in \mathcal{D}_h(G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}})$ and $h_2 \in \mathcal{D}_h(A)$ such that $h \in h_1 \,\widetilde{\|}_h\, h_2$. Let us first progressively establish that $h_1 = \Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.(\gamma, \delta^+)$ for some store γ.

STEP 1: $h_1 = (\rho_0, \tau, \gamma_1).h_1',\ h = (\rho_0, \tau, \gamma_1).h_r,\ h_r \in h_1' \,\widetilde{\|}_h\, h_2$.

To start, let us first observe that, employing the notations of definition 98, h rewrites as $h = (\rho_0, \tau, \gamma_1).h_r$ with $h_r = (\gamma_1, \tau, \rho_1).h_r'$. Recalling definition 44 and proposition 50, we now observe that only four situations are possible:

1. $h_1 = (\rho_0, \tau, \gamma_1).h_1'$ and $h_r \in h_1' \,\widetilde{\|}_h\, h_2$
2. $h_2 = (\rho_0, \tau, \gamma_1).h_2'$ and $h_r \in h_1 \,\widetilde{\|}_h\, h_2'$
3. $h_1 = (\rho_0, c_1, \gamma_1).h_1'$, $h_2 = (\rho_0, \overline{d_1}, \gamma_1).h_2'$, $h_r \in h_1' \,\widetilde{\|}_h\, h_2'$, with $\rho \cup \{c_1\} \vdash d_1$
4. $h_1 = (\rho_0, \overline{c_1}, \gamma_1).h_1'$, $h_2 = (\rho_0, d_1, \gamma_1).h_2'$, $h_r \in h_1' \,\widetilde{\|}_h\, h_2'$, with $\rho \cup \{d_1\} \vdash c_1$

Let us prove that the last three cases cannot occur.

Case 2. In that case, the history h_2' cannot be of the form $h_2' = (\gamma_1, \tau, \rho_1).h_2''$ or $h_2' = (\gamma_1, a_1, \rho_1).h_2''$. Indeed, by propositions 67 and 68, the goal A should then contain *tell*(a_1), which is impossible in view of the choice of S and a_1.

Moreover, h_2' cannot be of the form $h_2' = (\gamma_1, \overline{d}, \rho_1).h_2''$. Indeed, in view of h_r, this would require that $(\gamma_1, \overline{d}, \rho_1).h_2''$ is combined with h_1 in a synchronized way by the operator $\widetilde{||}_h$. In view of proposition 50, h_1 would then be of the form $h_1 = (\gamma_1, c_1, \rho_1).h'_1$ and the equality $\rho_1 = \gamma_1 \cup \{c_1\}$ would hold. It would follow from the equality $\rho_1 = \gamma_1 \cup \{a_1\}$ that $\sigma \cup S \vdash a_1$, which contradicts the choice of a_1.

Consequently, h_1 must be of the form $(\gamma_1, l, \rho_1).h'_1$. However, as $\gamma_1 \vdash c_1$, proposition 50 then implies that $h_1 = (\gamma_1, \tau, \gamma_1).h'_1$ and thus that $\gamma_1 = \rho_1$ which is impossible since then $\sigma \cup S \vdash a_1$.

Summing up, case 2 cannot occur.

Case 3. In that case, by proposition 50, if h'_1 starts in γ_1, then $h'_1 = (\gamma_1, \tau, \gamma_1).h''_1$. Since $\gamma_1 \neq \rho_1$, to get $h'_r = (\gamma_1, \tau, \rho_1) \in h'_1 \widetilde{||}_h h'_2$, the history h'_2 should thus be of the form $h'_2 = (\gamma_1, \tau, \rho_1).h''_2$. By proposition 68, this implies the presence of a $tell(c)$ operation in A such that $\gamma_1 \cup \{c\} = \rho_1$ and consequently such that $\gamma_1 \cup \{c\} \vdash a_1$. However, this contradicts the choice of a_1.

Case 4. In that case, by proposition 50, if h'_1 starts in γ_1, then $h'_1 = (\gamma_1, \tau, \gamma_1).h''_1$. Since $\gamma_1 \neq \rho_1$, to get $h'_r = (\gamma_1, \tau, \rho_1) \in h'_1 \widetilde{||}_h h'_2$, the history h'_2 should thus be of the form $h'_2 = (\gamma_1, \tau, \rho_1).h''_2$. However, this is impossible as just established.

Summing up, case 1 holds, namely $h_1 = (\rho_0, \tau, \gamma_1).h'_1$ and $h_r \in h'_1 \widetilde{||}_h h_2$. Moreover, it is worth observing that thanks to proposition 50,

$$h'_1 \in \mathcal{D}_h((tell(a_1) \mid\mid ask(a_1)); ; \cdots ; (tell(c_m) \mid\mid ask(c_m)); (tell(a_m) \mid\mid ask(a_m))).$$

STEP 2: $h'_1 = (\gamma_1, \tau, \rho_1).h''_1,\ h_r = (\gamma_1, \tau, \rho_1).h'_r,\ h'_r \in h''_1 \widetilde{||}_h h_2$.

Let us establish that h_2 cannot be of one of the three forms $h_2 = (\gamma_1, \tau, \rho_1).h'_2$, $h_2 = (\gamma_1, c, \rho_1).h'_2$, or $h_2 = (\gamma_1, \overline{d}, \rho'_1).h'_2$. If so, then, by definition of $\widetilde{||}_h$, h'_1, h_r, and h'_r must then be of the required form.

By proposition 68, $h_2 = (\gamma_1, \tau, \rho_1).h'_2$ or $h_2 = (\gamma_1, c, \rho_1).h'_2$ implies that A contains a $tell(c)$ operation such that $\gamma_1 \cup \{c\} = \rho_1$ and consequently such that $\gamma_1 \cup \{c\} \vdash a_1$. However, this contradicts the choice of a_1.

Moreover, h'_2 of the form $h'_2 = (\gamma_1, \overline{d}, \rho'_1).h''_2$ requires, in view of h_r, that $(\gamma_1, \overline{d}, \rho'_1).h''_2$ is combined with h'_1 in a synchronized way by the operator $\widetilde{||}_h$. In view of proposition 50, h'_1 must then be of the form $h_1 = (\gamma_1, a_1, \rho_1).h"_1$. Moreover, for the triple $(\gamma_1, \overline{d}, \rho'_1)$ to appear, $\gamma_1 \not\vdash d$ and $\rho_1 \vdash d$. Taking γ_1 for α and d for c, this contradicts the property of the a_i's that, for any constraint $c \in S$, $\alpha \cup \{a_1, \cdots, a_m\} \vdash c \Rightarrow \alpha \vdash c$.

FOLLOWING STEPS. By similar reasonings, h_1 can be proved of the form $h_1 = \Sigma^{S;\{a_1,\cdots,a_m\}}_{\rho\rightarrow\sigma}.h'''_1$. The goal $G^{S;\{a_1,\cdots,a_m\}}_{\rho\rightarrow\sigma}$ has thus made the first $2 \times m$ steps. Consequently, it has reached completion successfully and $h'''_1 = (\gamma, \delta^+)$. for some store γ.

■

Proposition 99 can be extended to more general sets of denotational histories.

Proposition 100 *Let ρ and σ be two stores such that $\rho \subset \sigma$. Let Sh be a subset of Shhist such that diff(S) is finite. Let S be a finite set containing diff(Sh).*

1) *Any history $h = \Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h'$ of $\mathcal{D}_h(G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}})\ \widetilde{||}\ Sh$ is from the set $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.(\gamma,\delta^+)\ \widetilde{||}_h\ h_s$ for some store γ and some history $h_s \in Sh$.*
2) *For any goal B, any history $h = \Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h'$ of $\mathcal{D}_h(G_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}};B)\ \widetilde{||}\ Sh$ is from the set $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}.h_b\ \widetilde{||}_h\ h_s$ for some histories $h_s \in Sh$ and $h_b \in \mathcal{D}_h(B)$.*

Proof Similar to that of proposition 99. ■

A point to observe with the sequence $\Sigma_{\rho\to\sigma}^{S;\{a_1,\cdots,a_m\}}$ is that it starts with ρ but does not end exactly in σ. In fact, it ends in $\sigma \cup \{a_1,\cdots,a_m\}$. Fortunately, this is not a problem since, as stated in proposition 85, the a_i's can be discarded if they are suitably chosen.

9.4 Key proposition

Theorem 101 *Let S_1, S_2 be two coherent subsets of Sdhist. Then, for any store α such that $(S_1{\Uparrow}\alpha) \setminus (S_2{\Uparrow}\alpha) \neq \emptyset$, there is a goal T and a continuous and real history $h \in (S_1\ \widetilde{||}\ \mathcal{D}_h(T)) \setminus \lfloor(S_2\ \widetilde{||}\ \mathcal{D}_h(T))\rfloor$ which starts in α.*

Proof The proof is conducted by induction on the minimum Lg of the length of the histories which are in $S_1{\Uparrow}\alpha$ and not in $S_2{\Uparrow}\alpha$.

CASE I: $Lg = 1$. Then there is an history $h \in S_1 \setminus S_2$ which is of the form (σ,δ^+) or of the form $(\sigma,\delta^-(X))$, for some set X.

Subcase i: $h = (\sigma,\delta^+)$. Let us first examine the case where $h = (\sigma,\delta^+)$. If $\alpha = \sigma$ then $T = \Delta$ satisfies the theorem. Otherwise, by hypothesis, $(\sigma,\delta^+) \notin S_2$. Let V be the set $diff(S_1) \cup diff(S_2)$. Consider $T = G_{\alpha\to\sigma}^{V;Sc}$, for some set of constraints Sc satisfying definition 98. Obviously, $h = \Sigma_{\alpha\to\sigma}^{V;Sc}.(\sigma \cup Sc,\delta^+)$ is a real and continuous history. It belongs to $S_1\ \widetilde{||}\ \mathcal{D}_h(T)$ since S_1 is consistent. To conclude in this case, let us prove that it does not belong to $\lfloor S_2\ \widetilde{||}\ \mathcal{D}_h(T)\rfloor$. To that end, since h ends successfully, it is sufficient to establish that it does not belong to $S_2\ \widetilde{||}\ \mathcal{D}_h(T)$. Indeed, if so, by proposition 100, h should come from the following merge:

$$h \in \Sigma_{\alpha\to\sigma}^{V;Sc}.(\gamma,\delta^+)\ \widetilde{||}_h\ h_s$$

for some store γ and some history $h_s \in S_2$. Moreover, since h ends after $\Sigma_{\alpha\to\sigma}^{V;Sc}$ by $(\sigma \cup Sc,\delta^+)$, one should have, by definition of the merge (see definition 44), $\gamma = \sigma \cup Sc$ and $h_s = (\sigma \cup Sc,\delta^+)$. Therefore, $(\sigma \cup Sc,\delta^+)$ should belong to S_2 and so should (σ,δ^+), since S_2 is consistent. However, this contradicts the hypothesis on S_2.

Subcase ii: $h = (\sigma,\delta^-(X))$. The case where $h = (\sigma,\delta^-(X))$ can be treated similarly if $(\sigma,\delta^-(Y)) \notin S_2$ for every subset $Y \subseteq Seconst$. If this is not the case, then, since S_2 satisfies the uniform disjointness deadlock condition, there are a finite number of non-empty goals $G_1, \ldots, G_m$ ($m > 0$) such that $<G_i, \sigma> \stackrel{\tau}{\not\longrightarrow}$ for $i = 1, \cdots, m$, and such that, for any $Z \subseteq Sdhist$, $(\sigma, \delta^-(Z)) \in S_2$ if and only if $Z \cap Actions(G_i, \sigma) = \emptyset$ for some $i = 1, \cdots, m$. In these conditions, the fact that $(\sigma, \delta^-(X)) \notin S_2$, implies that $X \cap Actions(G_i, \sigma) \neq \emptyset$, for every $i = 1, \cdots, m$. Select then for each i, $x_i \in X \cap Actions(G_i, \sigma)$ and rewrite $\{x_1, \cdots, x_m\}$ as $\{e_1, \cdots, e_p, \overline{f_1}, \cdots, \overline{f_q}\}$, with $e_1, \cdots, e_p, f_1, \cdots, f_q \in C$, $p \geq 0$, $q \geq 0$, according to the fact that x_i is an overlined constraint or not. Note that, by construction $\{e_1, \cdots, e_p, \overline{f_1}, \cdots, \overline{f_q}\}$ is finite and non-empty.

Consider now

$$\begin{aligned} TS &= ask(e_1) + \cdots + ask(e_p) + tell(f_1) + \cdots + tell(f_q) \\ T &= G^{V;Sc}_{\alpha \to \rho}; TS \end{aligned}$$

for the set $V = diff(S_1) \cup diff(S_2) \cup \sigma \cup \{e_1, \cdots, e_p\} \cup \{f_1, \cdots, f_q\}$. The thesis is established if the following properties hold:

1. $\Sigma^{V;Sc}_{\alpha \to \rho}.(\sigma \cup Sc, \delta^-(U)) \in S_1 \,\widetilde{\|}\, \mathcal{D}_h(T)$, for some $U \subseteq Sdhist$;
2. $\Sigma^{V;Sc}_{\alpha \to \rho}.(\sigma \cup Sc, \delta^-(U)) \notin S_2 \,\widetilde{\|}\, \mathcal{D}_h(T)$, for any $U \subseteq Sdhist$.

i.e. in view of proposition 100, if the following properties hold:

(P_1) $(\sigma \cup Sc, \delta^-(U)) \in S_1 \,\widetilde{\|}\, \mathcal{D}_h(TS)$, for some $U \subseteq Sdhist$;

(P_2) $(\sigma \cup Sc, \delta^-(U)) \notin S_2 \,\widetilde{\|}\, \mathcal{D}_h(TS)$, for any $U \subseteq Sdhist$.

PROOF OF P_1. To prove (P_1), let us first observe that, by definition 49

$$(\sigma \cup Sc, \delta^-(Seconst \setminus (\cup_{i=1}^{p} \overline{e_i\uparrow(\sigma \cup Sc)} \cup \cup_{j=1}^{q} f_j\downarrow(\sigma \cup Sc)))) \in \mathcal{D}_h(TS).$$

As $(\sigma, \delta^-(X)) \in S_1$ by hypothesis and since S_1 is consistent, $(\sigma \cup Sc, \delta^-(\widehat{X}^{(\sigma \cup Sc)})) \in S_1$. In view of definition 44, it thus remains to be shown that

$$(Seconst \setminus \widehat{X}^{(\sigma \cup Sc)}) \cap \overline{(Seconst \setminus (Seconst \setminus (\cup_{i=1}^{p} \overline{e_i\uparrow(\sigma \cup Sc)} \cup \cup_{j=1}^{q} f_j\downarrow(\sigma \cup Sc))))} = \emptyset$$

i.e. for any $i = 1, \cdots, p$, any $j = 1, \cdots, q$,

$$(Seconst \setminus \widehat{X}^{(\sigma \cup Sc)}) \cap e_i\uparrow(\sigma \cup Sc) = \emptyset$$

and

$$(Seconst \setminus \widehat{X}^{(\sigma \cup Sc)}) \cap \overline{f_j\downarrow(\sigma \cup Sc)} = \emptyset.$$

Since $\widehat{X}^{(\sigma \cup Sc)}$ is closed wrt $\sigma \cup Sc$, proving these equalities amounts to establishing that, on the one hand, $e_i \in \widehat{X}^{(\sigma \cup Sc)}$, for $i = 1, \cdots, p$, and $\overline{f_j} \in \widehat{X}^{(\sigma \cup Sc)}$, for $j = 1, \cdots, q$. The first set of memberships results directly from proposition 29 and the fact that, by construction, $e_i \in X$, for $i = 1, \cdots, p$. To prove the second set of memberships, the same reasoning can be followed but the application of proposition 29 requires in addition that $\sigma \cup Sc \not\vdash f_j$, for any $j = 1, \cdots, q$. This is indeed the case. Otherwise, since Sc does not influence γ, $\sigma \cup Sc \vdash f_j$ implies that $\sigma \vdash f_j$. However, by proposition 52, $\sigma \not\vdash f_j$.

PROOF OF (P_2). Assume that

$$(\sigma \cup Sc, \delta^-(U)) \in S_2 \mathbin{\widetilde{\|}} \mathcal{D}_h(TS)$$

for some set U. In that case, by definition 44, there are W_2 and W_T such that

$$(\sigma \cup Sc, \delta^-(W_2)) \in S_2$$

$$W_T \subseteq Seconst \setminus (\cup_{i=1}^{p} \overline{e_i\uparrow(\sigma \cup Sc)} \cup \cup_{j=1}^{q} f_j\downarrow(\sigma \cup Sc)) \tag{6}$$

$$(Seconst \setminus W_2) \cap \overline{Seconst \setminus W_T} = \emptyset \tag{7}$$

As S_2 is consistent, it follows that $(\sigma, \delta^-(\widehat{W_2}^{\sigma})) \in S_2$ and thus that there is $k \in \{1, \cdots, m\}$ such that

$$\widehat{W_2}^{\sigma} \cap Actions(G_k, \sigma) = \emptyset$$

i.e. such that

$$Seconst \setminus \widehat{W_2}^{\sigma} \supseteq Actions(G_k, \sigma).$$

Moreover, from (6), one derives that

$$Seconst \setminus WT \supseteq (\cup_{i=1}^{p} \overline{e_i\uparrow(\sigma \cup Sc)} \cup \cup_{j=1}^{q} f_j\downarrow(\sigma \cup Sc))$$

and consequently that

$$\overline{Seconst \setminus WT} \supseteq (\cup_{i=1}^{p} e_i\uparrow(\sigma \cup Sc) \cup \cup_{j=1}^{q} \overline{f_j\downarrow(\sigma \cup Sc)}).$$

Summing up,

$$(Seconst \setminus \widehat{W_2}^{\sigma}) \cap \overline{Seconst \setminus WT} \supseteq Actions(G_k, \sigma) \cap (\cup_{i=1}^{p} e_i\uparrow(\sigma \cup Sc) \cup \cup_{j=1}^{q} \overline{f_j\downarrow(\sigma \cup Sc)})$$

To conclude, let us reason on x_k. On the one hand, if $x_k = e_i$ for some $i \in \{1, \cdots, p\}$ then $e_i \in Actions(G_k, \sigma)$ by definition of x_k, and thus, as $e_i \in e_i\uparrow(\sigma \cup Sc)$ – see definition 21 –

$$e_i \in (Seconst \setminus \widehat{W_2}^{\sigma}) \cap \overline{Seconst \setminus WT}.$$

By contraposing proposition 29 applied to e_i and W_2, if $e_i \notin \widehat{W_2}^{\sigma}$ then $e_i \notin W_2$. As a result,

$$(Seconst \setminus W_2) \cap \overline{Seconst \setminus W_T} \neq \emptyset$$

which contradicts (7). On the other hand, if $x_k = \overline{f_j}$, for some $j \in \{1, \cdots, q\}$, then $\overline{f_j} \in Actions(G_k, \sigma)$, by definition of x_k. Therefore, since, as established in the proof of (P_1), $\sigma \cup Sc \not\vdash f_j$, then $\overline{f_j} \in f_j{\downarrow}(\sigma \cup Sc)$ and consequently

$$\overline{f_j} \in (Seconst \setminus \widehat{W_2}^{\sigma}) \cap \overline{Seconst \setminus WT}.$$

Contraposing proposition 29 applied to f_j and W_2 then leads to $\overline{f_j} \notin \widehat{W_2}^{\sigma}$ implies $f_j \notin W_2$. Therefore

$$(Seconst \setminus W_2) \cap \overline{Seconst \setminus W_T} \neq \emptyset,$$

which contradicts (7).

CASE II: $Lg > 1$. Let us now consider the case where the minimum of the lengths of the histories of $(S_1{\Uparrow}\alpha) \setminus (S_2{\Uparrow}\alpha)$ is greater than 1. In that case, let h be such an history of minimum length. It is thus of the form $h = (\rho, l, \sigma).h'$ for some stores σ, τ, some label l, and some history h'. There are two cases to be considered: either $S_2[(\rho, l, \sigma)] = \emptyset$ or $S_2[(\rho, l, \sigma)] \neq \emptyset$ but $h' \notin S_2[(\rho, l, \sigma)]$.

Subcase i: $S_2[(\rho, l, \sigma)] = \emptyset$. In the case $S_2[(\rho, l, \sigma)] = \emptyset$, the intuition behind the proof is as follows. We first observe, by proposition 76, that, for any σ' there is a continuous history $h_r \in S_1[(\rho, l, \sigma)]$ starting in σ'. It can be made real by being put in parallel with its complementor $Co(h_r)$. This determines the tail of the distinguishing history. The start of this history is obtained by the goal $T_s = G^{V;Sc}_{\alpha\to\rho}$, with V and Sc as above, which is able to perform the steps of $\Sigma^{V;Sc}_{\alpha\to\rho}$. To conclude, it remains to leave and force S_1 and S_2 perform the step (ρ, l, σ). To that end, we need a trick which consists of adding after (ρ, l, σ) a step (σ, τ, σ') which can only be made by the tester.

More formally, take V as a finite superset of $diff(S_1) \cup diff(S_2)$ which also contains the constraint appearing in l, if any, and Sc as one of the possible sets of constraints $a_1, \ldots, a_m$ verifying the hypothesis of definition 98. Let c be a constraint such that $\sigma \cup diff(S_1) \cup diff(S_2) \cup Sc \not\vdash c$ and let $\sigma' = \sigma \cup \{c\} \cup Sc$. Define the goal G_l as follows:

$$G_l = \begin{cases} \triangle & \text{if } l = \tau \\ ask(d) & \text{if } l = d \\ tell(d) & \text{if } l = \overline{d} \end{cases}$$

Consider the continuous history h_r mentioned above, and define T as the goal[1]

$$T = G^{V;Sc}_{\alpha\to\rho}; G_l; (tell(c) \mid\mid ask(c)); Co(h_r).$$

Then, the history

$$h^* = \Sigma^{V;Sc}_{\alpha\to\rho}.(\rho \cup Sc, \tau, \sigma \cup Sc).(\sigma \cup Sc, \tau, \sigma').(h_r)^c$$

enjoys the following properties:

[1]Strictly speaking, G_l and $G^{V;Sc}_{\alpha\to\rho}$ are dropped from T in the case they are empty.

- it is real, continuous, and starts in α,
- it belongs to $S_1 \,\widetilde{||}\, \mathcal{D}_h(T)$,
- it is not of $\lfloor S_2 \,\widetilde{||}\, \mathcal{D}_h(T)\rfloor$.

The first property should be clear. The second is evidenced by noting the three following facts:

1. h_T defined as follows is in $\mathcal{D}_h(T)$,

$$h_T = \begin{cases} \Sigma^{V;Sc}_{\alpha\to\rho}.(\sigma \cup Sc, \tau, \sigma').(h_r)^n & \text{if } l = \tau \\ \Sigma^{V;Sc}_{\alpha\to\rho}.(\rho \cup Sc, \bar{l}, \sigma \cup Sc).(\sigma \cup Sc, \tau, \sigma').(h_r)^n & \text{if } l \neq \tau \end{cases}$$

2. since S_1 is consistent,

$$h_1 = (\rho \cup Sc, \bar{l}, \sigma \cup Sc).h_r \in S_1$$

3. $h^* \in h_1 \,\widetilde{||}_h\, h_T$.

Finally the third property is established by contradiction as follows. If it does not hold then, following proposition 100, a variant of h^* obtained by possibly modifying its ending mark should come from the merge of two histories of the form $\Sigma^{V;Sc}_{\alpha\to\sigma}.h_g$ and h_s, with $h_g \in \mathcal{D}_h(G_l; (tell(c) \mid\mid ask(c)); Co(h_r))$ and $h_s \in S_2$. However, h_g cannot be of the form $(\rho \cup Sc, \tau, \sigma \cup Sc).h'_g$. Indeed, on the one hand, if $l = \tau$, then, $G_l = \Delta$ and $h_g \in \mathcal{D}_h((tell(c) \mid\mid ask(c)); Co(h_r))$. By proposition 50, $\sigma \cup Sc \vdash c$, which contradicts the choice of c. On the other hand, if $l = d$ or $l = \overline{d}$, then, by proposition 67 and definition of Sc, $\rho \cup Sc \not\vdash d$. Hence, if it starts in $\rho \cup Sc$, then h_g is necessarily of the form $h_g = (\rho \cup Sc, \bar{l}, \sigma \cup Sc).h'_g$ with $\bar{l} \neq \tau$.

It follows that h_s must participate to the step $(\rho \cup Sc, \tau, \sigma \cup Sc)$ either alone if $l = \tau$ or in synchronization with h_g if $l \neq \tau$. In other terms, $h_s = (\rho \cup Sc, l, \sigma \cup Sc).h'_s$. Therefore, $S_2[(\rho \cup Sc, l, \sigma \cup Sc)]$ is not empty and, so is $S_2[(\rho, l, \sigma)]$ since S_2 is consistent. However, this contradicts the hypothesis on S_2.

Subcase ii: $S_2[(\rho, l, \sigma)] \neq \emptyset$ but $h' \notin S_2[(\rho, l, \sigma)]$. In that case, by hypothesis $h' \in S_1[(\rho, l, \sigma)] \setminus S_2[(\rho, l, \sigma)]$ and the minimum of the length of those histories which are in $S_1[(\rho, l, \sigma)]$ and not in $S_2[(\rho, l, \sigma)]$ is strictly less than Lg. We are thus in the position of applying the induction hypothesis. For an arbitrarily given store α' such that

$$S_1[(\rho, l, \sigma)]\Uparrow\alpha' \setminus S_2[(\rho, l, \sigma)]\Uparrow\alpha' \neq \emptyset$$

it delivers a tester T' and a real and continuous history h^*_r starting in α' and which is in $(S_1[(\rho, l, \sigma)] \,\widetilde{||}\, \mathcal{D}_h(T')) \setminus \lfloor S_2[(\rho, l, \sigma)] \,\widetilde{||}\, \mathcal{D}_h(T')\rfloor$. The proof then consists of prefixing T' by some actions, yielding T, and h^*_r by a suitable sequence, yielding h^*, such as h^* starts, as required, in α, is real and continuous, and is in $S_1 \tilde{||} \mathcal{D}_h(T)$ and not in $\lfloor S_2 \tilde{||} \mathcal{D}_h(T) \rfloor$. Applying the technique of subcase i), T should start by $G^{V;Sc}_{\alpha\to\rho}$ to bring the store α to $\rho \cup Sc$, then leave S_1 and S_2 do the step (ρ, l, σ) and then resume by doing T'. In order to force the S_i's to do so, we shall use again an additional step that can only be made by T. Hence, let c be a constraint not influencing $\sigma \cup diff(S_1) \cup diff(S_2) \cup Sc$ and let T be defined as

$$T = G^{V;Sc}_{\alpha\to\rho}; G_l; (tell(c) \mid\mid ask(c)); T'.$$

Moreover, take α' as $\alpha' = \sigma \cup \{c\} \cup Sc$. Note that since S_1 and S_2 are uniformly consistent,

$$S_1[(\rho, l, \sigma)] \Uparrow \alpha' \setminus S_2[(\rho, l, \sigma)] \Uparrow \alpha' \neq \emptyset$$

can be deduced from

$$S_1[(\rho, l, \sigma)] \Uparrow \sigma \setminus S_2[(\rho, l, \sigma)] \Uparrow \sigma \neq \emptyset,$$

which holds by hypothesis of subcase ii).

In these conditions, the history

$$h^* = \Sigma^{V;Sc}_{\alpha\to\rho}.(\rho \cup Sc, \tau, \sigma \cup Sc).(\sigma \cup Sc, \tau, \alpha').h^*_r$$

enjoys the following properties:

- it is real, continuous, and starts in α,
- it belongs to $S_1 \tilde{||} \mathcal{D}_h(T)$,
- it is not of $\lfloor S_2 \tilde{||} \mathcal{D}_h(T) \rfloor$.

The first property should be clear. To establish the second property, let us first note that, since $h^*_r \in S_1[(\rho, l, \sigma)] \tilde{||} \mathcal{D}_h(T')$ there is $h^*_1 \in S_1[(\rho, l, \sigma)]$ and $h^*_t \in \mathcal{D}_h(T')$ such that $h^*_r \in h^*_1 \tilde{||}_h h^*_t$. Therefore,

$$h_T = \begin{cases} \Sigma^{V;Sc}_{\alpha\to\rho}.(\sigma \cup Sc, \tau, \alpha').h^*_t & \text{if } l = \tau \\ \Sigma^{V;Sc}_{\alpha\to\rho}.(\rho \cup Sc, \bar{l}, \sigma \cup Sc).(\sigma \cup Sc, \tau, \alpha').h^*_t & \text{if } l \neq \tau \end{cases}$$

is in $\mathcal{D}_h(T)$ and, since S_1 is uniformly consistent,

$$h_1 = (\rho \cup Sc, l, \sigma \cup Sc).h^*_1 \in S_1.$$

Hence, $h^* \in h_T \tilde{||}_h h_1$ and thus $h^* \in S_1 \tilde{||} \mathcal{D}_h(T)$.

To conclude, let us prove the third property by contradiction. Otherwise, in view of proposition 100, a variant of h^* obtained by possibly modifying its ending mark should be in the set $\Sigma^{V;Sc}_{\alpha\to\rho}.h_t \tilde{||}_h h_s$ for some histories $h_t \in \mathcal{D}_h(G_l; (tell(c) \mid\mid ask(c)); T')$ and $h_s \in S_2$.

Moreover, using the same argument as in the previous subcase i), T cannot be responsible for the step $(\rho \cup Sc, l, \sigma \cup Sc)$ ie, restated in formal terms, h_t cannot be of the form $h_t = (\rho \cup Sc, \tau, \sigma \cup Sc).h'_t$. Hence, $h_s = (\rho \cup Sc, l, \sigma \cup Sc).h'_s$, for some history h'_s, and,

$$h_t = \begin{cases} h'_t & \text{if } l = \tau \\ (\rho \cup Sc, \bar{l}, \sigma \cup Sc).h'_t & \text{if } l \neq \tau, \end{cases}$$

for some history $h'_t \in \mathcal{D}_h((tell(c) \mid\mid ask(c)); T')$. Note, in particular, that $h'_s \in S_2[(\rho, l, \sigma)]$ since S_2 is uniformly consistent and since $init(h_s) \supseteq \sigma \cup Sc$. Furthermore, thanks to the choice of the constraint c, S_2 cannot participate to the step $(\sigma \cup Sc, \tau, \alpha')$ ie h'_s cannot rewrite as $h'_s = (\sigma \cup Sc, \tau, \alpha').h''_s$. Indeed, if this was the case, then, by definition (see notation 35), $c \in diff(S_2)$. It follows that $diff(S_2) \vdash c$, which contradicts the choice of c.

Therefore, $h'_t = (\sigma \cup Sc, \tau, \alpha').h''_t$, for some history $h''_t \in \mathcal{D}_h(T')$.

Summing up, $h^*_r \in h''_t \,\widetilde{||}_h\, h'_s$ for some histories $h''_t \in \mathcal{D}_h(T')$ and $h'_s \in S_2[(\rho, l, \sigma)]$ and consequently, $h^*_r \in S_2[(\rho, l, \sigma)] \,\widetilde{||}\, \mathcal{D}_h(T')$, which contradicts the fact that, by construction, h^*_r is in $(S_1[(\rho, l, \sigma)] \,\widetilde{||}\, \mathcal{D}_h(T')) \setminus \lfloor S_2[(\rho, l, \sigma)] \,\widetilde{||}\, \mathcal{D}_h(T') \rfloor$.

■

9.5 Proof of the full abstraction property

We are now in a position to establish the full abstraction property.

Theorem 102 *The semantics $\mathcal{D}_h$ is fully abstract with respect to the semantics $\mathcal{O}_h$.*

Proof Following definition 95, the two following properties should be established equivalent:

i) for any context C, $\mathcal{O}_h(C[G_1]) = \mathcal{O}_h(C[G_2])$;
ii) $\mathcal{D}_h(G_1) = \mathcal{D}_h(G_2)$.

The implication ii) $\Rightarrow$ i) follows directly from theorem 66.

The other implication i) $\Rightarrow$ ii) is proved by contraposition. Assume $\mathcal{D}_h(G_1) \neq \mathcal{D}_h(G_2)$. Then, since both $\mathcal{D}_h(G_1)$ and $\mathcal{D}_h(G_2)$ are sets, there is an history h which is in one set and not in the other one. Without lost of generality, we may assume that $h \in \mathcal{D}_h(G_1)$ and $h \notin \mathcal{D}_h(G_2)$. Then $\mathcal{D}_h(G_1) \setminus \mathcal{D}_h(G_2) \neq \emptyset$ and, consequently taking as coherent sets $S_1 = \mathcal{D}_h(G_1)$, $S_2 = \mathcal{D}_h(G_2)$, and ϵ as initial store α, theorem 101 establishes that there is a goal T and a real and continuous history $h \in (\mathcal{D}_h(G_1) \,\widetilde{||}\, \mathcal{D}_h(T)) \setminus \lfloor(\mathcal{D}_h(G_2) \,\widetilde{||}\, \mathcal{D}_h(T))\rfloor$ which starts in ϵ. Note that, by definition 49, $h \in \mathcal{D}_h(G_1 \mid\mid T) \setminus \lfloor \mathcal{D}_h(G_2 \mid\mid T) \rfloor$. Therefore, by theorem 66, $\widetilde{h}$ is an operational history of $\mathcal{O}_h(G_1 \mid\mid T)$ which is not in $\mathcal{O}_h(G_2 \mid\mid T)$. There is thus a context $C = \square \mid\mid T$, such that $\mathcal{O}_h(C[G_1]) \neq \mathcal{O}_h(C[G_2])$, which concludes the proof.

■

10 Acknowledgment

The authors thank K. Apt, M. Bonsangue, A. Brogi, J.W. de Bakker, F. de Boer, E. de Vinck, M. Gabbrielli, E. Horita, J. Kok, B. Le Charlier, U. Montanari, C. Palamidessi, J. Rutten, V. Saraswat, P.-Y. Schobbens, and P. Wegner for stimulating discussions on the semantics of concurrent logic languages.

References

[1] L. Brim, D. Gilbert, J.-M. Jacquet, and M. Křetínský. Synchronisation in Scc. In *Proc. Int. Symp. on Logic Programming*, 1995.

[2] L. Brim, D. Gilbert, J.-M. Jacquet, and M. Křetínský. A Process Algebra for Synchronous Concurrent Constraint Programming. In M. Hanus and M. Rodriguez-Artalejo, editors, *Proceedings of the 5th Conference on Algebraic and Logic Programming*, volume 1139 of *Lecture Notes in Computer Science*, pages 165–178. Springer-Verlag, September 1996.

[3] S.D. Brookes, C.A.R. Hoare, and W. Roscoe. A Theory of Communicating Sequential Processes. *Journal of the ACM*, 31:499–560, 1984.

[4] F. S. de Boer and C. Palamidessi. A Fully Abstract Model for Concurrent Constraint Programming. In S. Abramsky and T.S.E. Maibaum, editors, *Proc. of TAPSOFT/CAAP91*, Lecture Notes in Computer Science, pages 296–319. Springer-Verlag, 1991.

[5] F.S. de Boer, J. W. Klop, and C. Palamidessi. Asynchronous Communication in Process Algebra. In *Proceedings of the Seventh Annual IEEE Symposium on Logic in Computer Science*, pages 137–147. The IEEE Computer Society Press, 1992.

[6] M. Falaschi, G. Levi, and C. Palamidessi. A Synchronization Logic: Axiomatics and Formal Semantics of Generalized Horn Clauses. *Information and Control*, 60:36–69, 1994.

[7] E. Horita, J.W. de Bakker, and J.J.M.M. Rutten. Fully Abstract Denotational Models for Nonuiform Concurrent Languages. *Information and computation*, 115(1):125–178, 1994.

[8] J.-M. Jacquet and L. Monteiro. Communicating Clauses: Towards Synchronous Communication in Contextual Logic Programming. In Krzysztof Apt, editor, *Proceedings of the Joint International Conference and Symposium on Logic Programming*, pages 98–112, Washington, USA, 1992. The MIT Press.

[9] R. Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[10] Michael W. Mislove and Frank J. Oles. Full Abstraction and Recursion. *Theoretical Computer Science*, 151:207–256, 1995.

[11] G. Plotkin. A Structured Approach to Operational Semantics. Technical report, DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[12] V. Saraswat. *Concurrent Constraint Programming*. The MIT Press, 1993.

[13] V. Saraswat and M. Rinard. Concurrent Constraint Programming. In *Proc. of 17th POPL*, pages 232–245, 1990.

[14] V. Saraswat, M. Rinard, and P. Panangaden. Semantic Foundations of Concurrent Constrant Programming. In *Proc. of 18th POPL*. ACM, 1991.

Publications in the FI MU Report Series are in general accessible via WWW and anonymous FTP:

`http://www.fi.muni.cz/informatics/reports/`
`ftp  ftp.fi.muni.cz (cd pub/reports)`

Copies may be also obtained by contacting:

Faculty of Informatics
Masaryk University
Botanická 68a
602 00 Brno
Czech Republic