BRICS

Basic Research in Computer Science

BRICS RS-03-15 A. Ingólfsdóttir: A Semantic Theory for Value–Passing Processes Based on the Late Approach

A Semantic Theory for Value–Passing Processes Based on the Late Approach

Anna Ingólfsdóttir

BRICS Report Series **RS-03-15**

ISSN 0909-0878 **March 2003**

See back inner page for a list of recent BRICS Report Series publications. Copies may be obtained by contacting:

BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK–8000 Aarhus C
Denmark

Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk

BRICS publications are in general accessible through the World Wide Web and anonymous FTP through these URLs:

`http://www.brics.dk`
`ftp://ftp.brics.dk`
This document in subdirectory `RS/03/15/`

A Semantic Theory for Value–Passing Processes Based on the Late Approach

Anna Ingólfsdóttir

BRICS[*]
Department of Computer Science
Aalborg University, Denmark

Abstract

A general class of languages for value-passing calculi based on the late semantic approach is defined and a concrete instantiation of the general syntax is given. This is a modification of the standard CCS according to the late approach. Three kinds of semantics are given for this language. First a Plotkin style operational semantics by means of an applicative labelled transition system is introduced. This is a modification of the standard labelled transition system that caters for value-passing according to the late approach. As an abstraction, late bisimulation preorder is given. Then a general class of denotational models for the late semantics is defined. A denotational model for the concrete language is given as an instantiation of the general class. Two equationally based proof systems are defined. The first one, which is value-finitary, i. e. only reasons about a finite number of values at each time, is shown to be sound and complete with respect to this model. The second proof system, a value-infinitary one, is shown to be sound with respect to the model, whereas the completeness is proven later. The operational and the denotational semantics are compared and it is shown that the bisimulation preorder is finer than the preorder induced by the denotational model. We also show that in general the ω-bisimulation preorder is strictly included in the model induced preorder. Finally a value-finitary version of the bisimulation preorder is defined and the full abstractness of the denotational model with respect to it is shown. It is also shown that for CCS_L the ω-bisimulation preorder coincides with the preorder induced by the model. From this we can conclude that if we allow for parameterized recursion in our language, we may express processes which coincide in any algebraic domain but are distinguished by the ω-bisimulation. This shows that if we extend CCS_L in this way we obtain a strictly more expressive language.

1 Introduction

In the original work of Milner on CCS [Mil80] and Hoare on CSP [Hoa78], processes are allowed to exchange data in communications. In these original calculi the value-passing calculus is interpreted in terms of the pure calculus in which communication is pure synchronization. A process which is ready to input a value on a channel c (e. g. a prefixing with an input action, $c?x.p$) is interpreted as a non-deterministic choice between pure terms of the form

[*]Basic Research in Computer Science, Centre of the Danish National Research Foundation.

$c_v.p[v/x]$, where v ranges over the set of possible values, which in many cases is infinite. In this approach, two processes that synchronize are both supposed to know each other's *channel* and *value*, i.e. the data variable is instantiated by the potential input values already when the process reports the willingness or ability to communicate on the channel c.

In more recent work on the π-calculus [MPW92] this semantic approach is referred to as *early semantics* due to the early instantiation of the data variables as described above. Its counterpart, the *late semantics*, is also introduced in the same reference. Here the idea is that the processes *only* synchronize on the channel name and that the input process has to accept whatever value the output process has to offer. This may be interpreted as if the result of the instantiation of the data variable is *delayed* until the process has received the value. The input process reports the willingness to receive a value on a channel c by performing an action of the form $c?$ and thereby evolving to a function which waits for the value the output counterpart in the communication provides. Symmetrically the result of reporting the willingness to output an uninterpreted value on the channel c is given by the action $c!$. By performing this action the process evolves to a term which basically consists of a data expression, i.e. the expression whose value the sender wants to output, and a process expression, i.e. what remains to be executed of the sender.

In a more recent version of the π-calculus, the Polyadic π-calculus presented in [Mil91], the outcomes of input and output actions are modelled by extending the syntax with the new constructions *abstractions* and *concretions*. The semantics for Thomsen's *plain CHOCS* in [Tho89] is based on the late approach although the author does not use that terminology.

In the literature the late semantic approach has been investigated in different ways, both in connection with the π-calculus and higher order calculi (see e.g. [MPW91, Hen94, San93]) and also with the main focus on the simpler case where only first order values are allowed (see e.g. [HL93a, HL93b, Ing93]). In this paper we will aim at contributing to the studies of the late semantics of communicating processes. We will concentrate on processes which allow for transmission of simple values only. Of course studying value-passing processes is interesting in itself, but we also believe that it may give some insight into the nature of the semantics of value passing processes, in particular the late semantic approach, which may be useful in future studies of the semantics of the more complicated calculi of higher order or mobile processes (such as the π-calculus).

In order to make our studies more complete, rather than giving only one type of semantics, we follow the line of [Hen88a] and [HI93] and introduce a trinity of semantic descriptions for a CCS like process language and show how they relate to one another. More precisely, first we put forward an operational or behavioural semantics in terms of an extended version of labelled transition systems and corresponding bisimulation based relations. Then we give a denotational semantics following the Scott-Strachey approach, and axiomatic semantics by means of equationally based proof systems. Like many researchers in the area of process algebra we believe that the operational or the behavioural semantic model is the most natural and intuitive one, but that different kinds of semantic descriptions give important alternative views of the nature of the interpretation of process languages. For instance the interpretation of an infinite process, modelled by an algebraic cpo, is fully specified by the interpretation of its finitely computable approximations. This is not the case for many behaviourally based semantics as will be explained in more detail later.

We start the study of the late semantic approach by defining a general class of syntax for languages that support this approach. To this end we extend the standard notion of a *signature* Σ, a set of syntactic operators, to that of *applicative signature* (Σ, C) where Σ is a signature in the original sense and C is a set of *channel names*. Then we define a concrete language *Late-CCS* (CCS_L) by instantiating the general applicative signature (Σ, C). This language is a slight modification of the standard *CCS* where the syntax is basically the same as for the Polyadic π-calculus although we use a slightly different notation and only allow for the transmission of simple values in communications. Then we will give a behavioural semantics to the language CCS_L in terms of a Plotkin style operational semantics and a bisimulation based preorder. Our main aim is to relate the behavioural view of processes we present here to the domain-theoretical one. In the Scott-Strachey approach an infinite process is obtained as a chain of finite and possibly partially specified processes. The completely unspecified process is modelled by the bottom element of the domain and is syntactically denoted by the constant Ω. An operational interpretation of this approach is to take divergence into account and give the behavioural semantics in terms of a prebisimulation or bisimulation preorder [Hen81, Wal90] rather than by the standard bisimulation equivalence [Par81, Mil83].

To reflect the late approach, the operational semantics is given in terms of an applicative transition system, a concept that is a modification of that defined in [Abr90]. We generalize the notion of bisimulation [Par81, Mil83] to be applied to applicative transition systems and introduce a preorder motivated by Abramsky's applicative bisimulation [Abr90]. To this end we first introduce the notion of strong applicative prebisimulation and the corresponding strong applicative bisimulation preorder. Following standard practice this preorder is obtained as the largest fixed point of a suitably defined monotonic functional.

Next we define a general framework for denotational semantics for value-passing processes using the late principle. For this purpose we introduce the general class of applicative (Σ, C)-domains to model the semantics of the (Σ, C)-terms. These are a direct generalization of Σ-domains originally introduced in [GTWW77] and used for instance in [Hen88a] to model a pure calculus. In the denotational interpretation of a language in terms of a (Σ, C)-domain, the idea of the late semantic approach is made explicit; the outcome of an input action is modelled as a function which takes a value as an argument and returns an element of the model, i.e. a process, whereas the outcome of an output action is modelled by a pair consisting of the output value and the resulting process.

After having defined our general class of models, we will modify the definition of evaluation mapping, i.e. the unique mapping from the process algebra into the domain known from the theory for pure processes. As we want to be able to reason about a subset of the process algebra, we extend the definition slightly. For this purpose we introduce the notion of *recursively closed subsets* of a process algebra. This extension of the definition allows us to reason about the compact elements of an algebraic *cpo* at the syntactic level by means of structural induction. This enables us to take advantage of the notion of algebraicity when comparing the semantics defined by the model to other kinds of semantics such as behavioural or axiomatic semantics.

Then we define a concrete denotational model for CCS_L, the domain of *Applicative Communication Trees* (ACT) as an instantiation of the general class of (Σ, C)-domains, where Σ is the signature consisting of the operators of CCS_L. The model ACT is obtained by defining a preorder K, which acts as a representation of the compact elements of the complete model. Then we define the operators of Σ and C as monotonic functions over this preorder. Finally we apply a general result (that, for instance, can be found in [Hen88a]) that says that a preorder with monotonic operators induces a unique structure consisting of the algebraic *cpo* obtained as completion by ideals of the preorder and the corresponding unique continuous extension of the operators.

By defining the operators in this way, i.e. first as monotonic endofunctions on the preorder that represents the compact elements and then extending them in a continuous way to the whole domain, we ensure that they *preserve compactness*. By this we mean that the result of applying an operator to a compact element is again a compact element. From an intuitive point of view this is an important property; the compact elements represent the finitely computable elements of the domain so if we expect an operator *op* to be finitely computable, applying it to something finitely computable should again result in something finitely computable. Note that this property is not automatically satisfied in an applicative (Σ, C)-domain, or even a Σ-domain, as a continuous function does not necessarily map a compact element into a compact element.

The definition of our model is motivated by the following models that have been studied in the literature. In 1979, Milne and Milner [MM79] gave a domain theoretical definition of the concept of *communicating processes*. This definition reflects the late semantic approach described above. Each process has a collection of typed ports through which it may communicate with other processes. There are two types of communications: input and output. If we abstract away from the types of the possible values then the input capability of a process p along a channel c is modelled as an element of the domain $V \longrightarrow P$ labelled by the channel name c, where the domain of processes is denoted by the *cpo* P and the domain of values by V. An output capability of p on c, on the other hand, is modelled as an element of $V \times P$ labelled by c. A process is modelled as a set of communication capabilities or more precisely as an element of the Smyth Power Domain [Smy78] over the domain of communication capabilities. The empty set is embedded into the domain in such a way that it becomes the top element of the domain. This leads to a recursive domain equation over a suitable class of domains. The domain of processes is then defined as the initial solution to this equation.

In [Abr91] Abramsky pointed out a disadvantage of this model: The use of the Smyth Power Domain to model communicating processes rules out the possibility of any correspondence with bisimulation as it only compares the processes in one direction, i. e. if the process p is smaller than the process q in the preorder then q simulates p, given that p is convergent, but not necessarily the other way around. Also the embedding of the empty set, which corresponds to the inactive and convergent process *nil* as the top element of the model, is intuitively incorrect as in the bisimulation based semantics, this process is not related to anything but itself and the inactive divergent process Ω. In the same reference the author defined a model to describe the semantics of pure processes. This model is similar to the model of [MM79] and is also obtained as the initial solution to a recursive domain equation. The main difference is that Abramsky, instead of the Smyth Power Domain, defined his model in terms of the Plotkin Power Domain which is based on comparison both ways; p is smaller than q if q simulates p and, in the case when p converges, p simulates q too. He added the empty set to the model as an isolated element only comparable with itself and the bottom element of the model in the obvious way. This also corresponds to the bisimulation interpretation of the process *nil*. Abramsky then interpreted the calculus $SCCS$ in the model and showed the full abstractness of this interpretation with respect to a bisimulation based preorder.

In the paper we show that the model we define is basically the one presented in [MM79] where the modifications of Abramsky's are adopted. Thus we define a model which describes value-passing based on the late approach as a solution to a recursive domain equation using the Plotkin Power Domain but with the empty set adjoined as an isolated element. Then we give an explicit representation of the compact elements of the solution by unfolding the recursive definition and show that it coincides with the preorder K. This in turn implies that the two domains we have defined so-far in two different ways are isomorphic.

The definition of the denotational model supports in a natural way systems of equations and inference rules. We define two such proof systems and prove their soundness and completeness with respect to the model. The two system are based on the same set of equations and only differ in the way the infinite terms are dealt with. In the first one, the value-finitary one, we make an extensive use of the ω-algebraicity of the model and say that t is provably smaller than u, written $t \sqsubseteq u$, if $t^{(n)} \sqsubseteq u$ can be proven for all n where $t^{(n)}$ is a *compact* approximation of t, i. e. $t^{(n)}$ is a syntactically finite term that that only reasons about the first n values of the value space (which we take to be countable) and is interpreted as a compact element in the model. Then the ω-algebraicity of the model, together with the fact that the operators preserve compactness, enables us to reduce the proof of completeness and soundness of this proof system to a proof of the same property for the sublanguage which denotes exactly the compact elements of the model.

The second proof system, the value-infinitary one, is maybe more standard and is obtained from the first one by replacing the compact approximations $t^{(n)}$ by the usual syntactic approximations t^n which may involve reasoning about an infinite number of values and are therefore not necessarily interpreted as compact elements in the model. The soundness of this system is obvious as it is weaker than the previous one. To prove the completeness, on the other hand, turns out to be more complicated and is in the paper postponed until the operational semantics is investigated. This will be explained later in this introduction.

Our next task is to compare the behavioural and the denotational semantics. One of the results in the pure case presented in [Abr91] is that the denotational model given in that reference is not fully abstract with respect to the bisimulation preorder which turns out to be too fine. Intuitively this is due to the algebraicity of the model and the fact that the finite elements in the model are denotable by syntactically finite terms. The algebraicity implies that the denotational semantics of a process is completely decided by the semantics of its syntactically finite approximations, whereas the same cannot be said about the bisimulation preorder. In fact we need experiments of infinite depth and width to investigate bisimulation while this is not the case for the preorder induced by the model as explained above. An obvious consequence of this observation is that in general, a bisimulation preorder cannot be expected to be modelled by an algebraic cpo given that the compact elements are denotable by syntactically finite elements.

In [Hen81] Hennessy defined a term model for $SCCS$. This model is ω-algebraic and, as expected, fails to be fully abstract with respect to the strong bisimulation preorder. In the same reference the author introduces the notion of "the finitary part of a relation" and "a finitary relation". The finitary part of a relation $\mathcal{R}$ over processes, denoted by $\mathcal{R}^F$, is defined by

$$p\mathcal{R}^F q \text{ iff } \forall d.d\mathcal{R}p \Rightarrow d\mathcal{R}q$$

where d ranges over the set of syntactically finite processes. A relation $\mathcal{R}$ is finitary if $\mathcal{R}^F = \mathcal{R}$. Intuitively this property may be interpreted as algebraicity at the behavioural level provided that syntactically finite terms are interpreted as compact elements in the denotational model; if a relation is finitary then it is completely decided by the syntactically finite elements. The "finitary part" of the bisimulation preorder is in [Abr91] referred to as the "finitely observable part" of the preorder. In both [Hen81] and [Abr91] the full abstractness of the respective denotational semantics with respect to $\precsim^F$ are shown. In [Abr91] it is also shown that if the language is sort finite and satisfies a kind of finite branching condition, then $\precsim^F = \precsim_\omega$ where $\precsim_\omega$

is the strong bisimulation preorder induced by experiments of finite depth, i.e. the preorder is obtained by iterated application of the functional that defines the bisimulation. Note that in general the preorder $\precsim$ is strictly finer than the preorder $\precsim_\omega$. However if the transition system is image finite, i.e. if the number of arcs leading from a fixed state and labelled with a fixed action is finite, then these two preorders coincide.

We show by an example that the bisimulation preorder is not finitary in the sense described above and is strictly finer than the preorder induced by the model. Next we define the strong applicative ω-bisimulation preorder for applicative labelled transition systems in the standard way by iterative application of the functional that induces the bisimulation preorder. This gives as a result a $\precsim_\omega$ preorder which in general is still too fine to match the preorder induced by the denotational model. This we demonstrate by an example given in a language which is a slight extension of our example language as it allows parameterized recursive definitions. Intuitively the reason for this mismatch is that we still need infinite experiments to decide the operational preorder, now because of an infinite breadth as an infinite number of values might need to be observed. In other words, the preorder of the model is value-finitary whereas the preorder $\precsim_\omega$ is not.

By an example, given in the language CCS_L extended with parameterized recursion, we show that, considered on the class of applicative labelled transition systems, the behavioural preorder $\sqsubseteq_\omega$ and a preorder induced by any algebraic cpo cannot coincide.

Motivated by the observations above we give a suitable definition of the notion of the "finitary part", which we refer to as the emvalue-finitary part, of the bisimulation preorder to meet the preorder induced by the denotational model. We define the so-called *compact* terms as the syntactically finite terms which only test and use a finite number of values. We also show that these terms correspond exactly to the compact elements in the denotational model in the sense that an element in the model is compact if and only if it can be denoted by a compact term. This motivates a definition of the value-finitary part $\precsim^F$ of the bisimulation preorder $\precsim$ by

$$p \precsim^F q \text{ iff } \forall c.\, c \precsim p \Rightarrow c \precsim q$$

where c ranges over the set of syntactically compact terms. We also define yet another preorder $\precsim^f_\omega$, a coarser version of $\precsim_\omega$ in which we only consider a finite number of values at each level in the iterative definition of the preorder. Here it is vital that the set of values is countable and can be enumerated as $Val = \{v_1, v_2, \cdots\}$. Thus in the definition of $\precsim^f_1$ we only test whether the defining constraints of the preorder hold when the only possible input and output value is v_1, and in general in the definition of $\precsim^f_n$ we test the constraints for the first n values only. (Here we would like to point out that a similar idea originally appears in [HP80].) It turns out that $\precsim^f_\omega$ is the finitary part of $\precsim$ in our new sense and that the model is fully abstract with respect to $\precsim^f_\omega$. We will prove both these results in this paper using techniques which are similar to those used by Hennessy in the aforementioned reference [Hen81]. We also prove that the value-infinitary proof system, i. e. the one based on the syntactically finite approximations is complete with respect to the preorder $\precsim_\omega$. Finally we compare the value-finitary and the value-infinitary semantics over our example language CCS_L by proving that the ω-bisimulation preorder $\precsim_\omega$ coincides with the preorder from the model. Thus we have proven that all the semantic preorders we have considered so far, apart from the bisimulation preorder $\precsim$, coincide over this language.

As pointed out above, the example that shows that the preorder $\precsim_\omega$ in general does not coincide with the preorder induced by the denotational model, is given in the language CCS_L

extended in such a way that recursive definitions can be parameterized over value expressions. The reasoning above shows that this extended language is strictly more expressive than the original one.

The structure of the paper is as follows: In Section 2 we define a general syntax for value-passing languages that support the late semantic approach. The definition of the operational semantics and the notion of applicative bisimulation are the subject of Section 3. In Section 4 we define the general class of (Σ, C)-domains and our concrete denotational model. In Section 5 we give a equationally based value-finitary proof system and prove its soundness and completeness with respect to the model. We also define a value-infinitary version of the system and prove that it is sound with respect to the model. Section 6 contains two examples that distinguishes the preorder $\sqsubseteq_{\overline{ACT}}$ from $\lesssim$ and $\lesssim_\omega$ respectively. These examples are followed up by the definition of the value-finitary preorder $\lesssim_\omega^f$. In Section 7 we give a definition of the notion of the value-finitary part of a relation and a value-finitary relation over processes. In the same section we prove that the preorder $\lesssim_\omega^f$ is value-finitary and that it coincides with the value-finitary part of the preorder $\lesssim$. Then we prove the soundness and the completeness of the value-finitary proof system with respect to the value-finitary bisimulation preorder $\lesssim_\omega^f$ and of the value-infinitary system with respect to $\lesssim_\omega$. Finally we prove that the model is fully abstract with respect to $\lesssim_\omega$, i. e. that the preorder from the model coincides with $\lesssim_\omega$. From this we can conclude that all five preorders mentioned above coincide. In Section 8 we give some concluding remarks.

2 Syntax

In this section we will extend the standard notion of a signature Σ and that of Σ-terms used for the pure calculus in order to model processes with value-passing based on the late approach. To this end we introduce the notion of *applicative signature* as a pair (Σ, C) where Σ is a signature and C is a set (of channel names), and that of (Σ, C)-terms.

The general syntax is based on predefined expression languages for value expressions and boolean expressions. Thus we assume some predefined syntactic category of expression Exp, ranged over by e, including a countable, unordered set of values Val, ranged over by v, and a set of value variables Var, ranged over by x. We also assume a predefined syntactic category $BExp$ of boolean expressions, ranged over by be, with the only values T (*true*) and F (*false*). $BExp$ should at least include a test for equality between the elements of Exp. From such a predicate a test for membership of a finite set can easily be derived. Value expressions are supposed to be equipped with a notion of substitution of an expression for a value variable, denoted by $e[e'/x]$, and an evaluation function $[\![\_]\!] : Exp \times VEnv \longrightarrow Val$, where $VEnv$ is the set of value environments $\sigma : Var \longrightarrow Val$. For closed expression we write $[\![e]\!]$ instead of $[\![e]\!]\sigma$. Furthermore we preassume an infinite set of process names PN, ranged over by P, Q, etc, to be used in recursively defined terms. The set of (Σ, C)-*terms* is given as the triplet

$$T_{(\Sigma,C)} = (T^{proc}_{(\Sigma,C)}, T^{fun}_{(\Sigma,C)}, T^{pair}_{(\Sigma,C)})$$

of the sets generated by Σ and C according to the following syntax:

$$\begin{array}{ll} T^{proc}_{(\Sigma,C)}: & p ::= op(\underline{p}), op \in \Sigma \mid c?.f \mid c!.o \mid \tau.p \mid be \to p, p', \\ T^{fun}_{(\Sigma,C)}: & f ::= [x]p, \\ T^{pair}_{(\Sigma,C)}: & o ::= (e, p), \end{array}$$

where we use the notation $\underline{p}$ to denote a vector of terms in $T^{proc}_{(\Sigma,C)}$ of a suitable length. We let t range over the union of these three categories and f and o over $T^{fun}_{(\Sigma,C)}$ and $T^{pair}_{(\Sigma,C)}$ respectively. To express recursive or infinite processes we add the process names in PN, ranged over by P, and the recursive binding $recP.\_$, to the syntax and write $T^{rec}_{(\Sigma,C)}(PN)$ for the resulting triplet of (Σ, C)-terms.

We have three kinds of actions, input actions of the form $c?$, $c \in C$, output actions of the form $c!$, $c \in C$ and the silent action τ. We write $C?$ for $\{c?|c \in C\}$ and $C!$ for $\{c!|c \in C\}$. The set $Act = C! \cup C?$ is ranged over by a whereas $Act_\tau = C! \cup C? \cup \{\tau\}$ is ranged over by μ. The structure of this syntax is basically the same as the one suggested by Milner in [Mil91] although the notation is slightly different. The action of inputting on channel c is given by $c?$ whereas the action of outputting on that channel is given by $c!$. The *function terms* are of the form $[x]p$, where x is a data variable and p a process term. These correspond to the *abstractions* in the aforementioned reference. The input prefixing becomes $c?.[x]p$. The *pair terms* are of the form (e, p), where e is a data expression and p a process term. These correspond to the *concretions* in [Mil91]. The output prefixing becomes $c!.(e, p)$. We also assume that we have a set of operators Σ, which is supposed to contain at least the symbol Ω to model the divergent or completely unspecified process. The processes are obtained by the input and output prefixing just described, prefixing with the silent action τ and by applying the operators in Σ. We use the notation $be \longrightarrow p, p'$ to denote the standard conditional choice usually written as "If be then p else p' ".

Prefixing by $[x]$ binds the data variable x and the $recP.\_$ construct is a binding construct for the process name P. A value variable x is free if it is not in the scope of a prefix $[x]$ and a process name P is free if it is not in the scope of a recursion construct $recP.\_$. We shall mainly be concerned with expressions which contain no free occurrences of value variables. We denote the set of all value closed terms, process terms, functions terms and pair terms by $\mathbf{T}_{(\Sigma,C)}(PN)$, $\mathbf{T}^{proc}_{(\Sigma,C)}(PN)$, $\mathbf{T}^{fun}_{(\Sigma,C)}(PN)$ and $\mathbf{T}^{pair}_{(\Sigma,C)}(PN)$ respectively. These will be referred to as processes, functions and pairs ranged over by $\mathbf{p}$, $\mathbf{f}$ and $\mathbf{o}$. We assume a notion of substitution for both data variables and process names in terms defined in the usual way.

The language $CCS_L = (CCS_L^{proc}, CCS_L^{fun}, CCS_L^{pair})$ (*Late-CCS*), ranged over by t, p, f, o respectively, is obtained by taking Σ as $\{nil, \Omega, +, |\} \cup \{\_R | R \in Ren\} \cup \{- \backslash c | c \in Chan\}$ where Ren is the set of finite[1] permutations of $Chan$ (the set PN is not indicated and is implicitly assumed to be known). The process nil is the convergent, inactive process, Ω is the completely unspecified or divergent one, $p+q$ is a nondeterministic choice between p and q, $p|q$ is a parallel composition of p, q, $p[R]$, p renamed by R, stands for the process p with its channels renamed by R and $p \backslash c$, p restricted on c, behaves like p apart from not being allowed to communicate on channel c. The corresponding closed terms $\mathbf{C}CS_L = (\mathbf{C}CS_L^{proc}, \mathbf{C}CS_L^{fun}, \mathbf{C}CS_L^{pair})$ are again ranged over by $\mathbf{t}, \mathbf{p}, \mathbf{o}, \mathbf{f}$ respectively. We let $\mathbf{d}$ range over syntactically finite or recursion free closed terms. Note that $f = [x]p$ can be considered as a function by using the convention $f(v) = ([x]p)(v) = p[v/x]$ where $v \in Val$.

In the theory to follow we will make an extensive use of the fact that the value domain Val is countable and can therefore be written as $Val = \{v_1, v_2, v_3, \ldots, \}$. By defining $V_n = \{v_1, \ldots, v_n\}$ we get that $Val = \bigcup_n V_n$. From now on V_n will have this meaning.

[1]This restriction is of technical reasons which are not going to be explained further here.

3 Operational Semantics

The operational semantics is given in terms of an applicative transition system, a slight modification of a notion originally suggested by Abramsky [Abr90]. An applicative transition system models the idea of looking at an input term as a prefixing of a function which is ready to receive values along the prefixing channel. Furthermore it reflects the idea of looking at an output term as a prefixing of a pair consisting of the value and the resulting process.

Definition 3.1 An *applicative labelled transition system* (*ALTS*) is a five tuple $AT = \langle Con, Val, Act, \longrightarrow, \downarrow \rangle$ where

- Con is a set of configurations,
- Val is a set of Values,
- $Act = Act_{Con} \uplus Act_{Pair} \uplus Act_{Fun}$ is a set of actions,
- $\longrightarrow$ is a transition relation

$$\begin{array}{rl} \longrightarrow \subseteq & (Con \times Act_{Con} \times Con) \cup \\ & (Con \times Act_{Pair} \times (Val \times Con)) \cup \\ & (Con \times Act_{Fun} \times (Val \longrightarrow Con)) \text{ and} \end{array}$$

- $\downarrow \subseteq Con$ is a convergence predicate.

We refer to $States = Con \cup (Val \times Con) \cup (Val \longrightarrow Con)$ as the set of *states*. □

Now we will define the so-called *strong applicative prebisimulation* (sa-prebisimulation) as a further abstraction on the applicative transition system. More precisely we define it as the greatest fixed point to a monotonic endofunction on the complete lattice $\langle \mathcal{P}(Con \times Con), \subseteq \rangle$. For this purpose we extend our notion of relations over configuration so they apply to states. Given a binary relation over Con we extend it pointwise to $Val \times Con$ by

for all $c_1, c_2 \in Con$ and $v_1, v_2 \in Val$, $(v_1, c_1)\mathcal{R}^{pair}(v_2, c_2)$ iff $c_1 \mathcal{R} c_2$ and $v_1 = v_2$

and to $Val \longrightarrow Con$ by

for all $f_1, f_2 \in Val \longrightarrow Con$, $f_1 \mathcal{R}_{fun} f_2$ iff $f_1(v)\mathcal{R}f_2(v)$ for all $v \in Val$.

For any $s, s' \in States$ we write $s\mathcal{R}s'$ if $s\mathcal{R}s'$ or $s\mathcal{R}^{pair}s'$ or $s\mathcal{R}^{fun}s'$ depending on the types of s and s'.

Definition 3.2 Let $AT = \langle Con, Val, Act, \longrightarrow, \downarrow \rangle$ be an $ALTS$. We define $\mathcal{F} : \mathcal{P}(Con \times Con) \longrightarrow \mathcal{P}(Con \times Con)$ by:

if $\mathcal{R} \subseteq Con \times Con$ then $c_1 \mathcal{F}(\mathcal{R}) c_2$ iff for all $\mu \in Act$

(i) $c_1 \xrightarrow{\mu} s_1$ implies $c_2 \xrightarrow{\mu} s_2$ for some s_2 such that $s_1 \mathcal{R} s_2$,

(ii) $c_1 \downarrow$ implies ($c_2 \downarrow$ and whenever $c_2 \xrightarrow{\mu} s_2$ then $c_1 \xrightarrow{\mu} s_1$ for some s_1 such that $s_1 \mathcal{R} s_2$),

where $s_1, s_2 \in States$.

□

Obviously $\mathcal{F}$ defined in this way is a monotonic endofunction over the complete lattice $\langle \mathcal{P}(Con \times Con), \subseteq \rangle$. Thus the Knaster-Tarski fixed point theorem [Tar55] applies and the greatest fixed point to $\mathcal{F}$ exists. We may therefore give the following definition:

Definition 3.3 (Strong Applicative Prebisimulation)
Let $AT = \langle Con, Val, Act, \longrightarrow, \downarrow \rangle$ be an applicative labelled transition system and $\mathcal{F}$ be defined as in Definition 3.2. Then $\mathcal{R} \subseteq \mathcal{P}(Con \times Con)$ is called a prebisimulation if it is a post-fixed point to $\mathcal{F}$, i.e. if $\mathcal{R} \subseteq \mathcal{F}(\mathcal{R})$. We define the *strong applicative bisimulation preorder* $\precsim$ as the greatest fixed point to $\mathcal{F}$, i.e.

$$\precsim = \bigcup \{\mathcal{R} | \mathcal{R} \subseteq \mathcal{F}(\mathcal{R})\}.$$

We define the *strong applicative bisimulation equivalence* as $\sim = \precsim \cap \precsim^{-1}$.

□

Similar results as for the pure case also hold here and are simply restated in the following lemma.

Lemma 3.4

1. *$\precsim$ is a preorder,*
2. *$\sim$ is an equivalence relation.*

So far we have given a definition of $\precsim$ on an abstract $ALTS$. Now we define a concrete $ALTS$ by taking Con to be $\mathbf{C}CS_L^{proc}$ as generated by the syntax in Section 2, where, as pointed out before, $\mathbf{C}CS_L^{fun}$ may be considered as a subset of $Val \longrightarrow \mathbf{C}CS_L^{proc}$. We let $\longrightarrow$ be the least transition relation closed under the rules of Figure 1 and the convergence predicate $\downarrow$ to be the least relation on $\mathbf{C}CS_L^{proc}$ satisfying the rules in Figure 2. As usual the divergence predicate $\uparrow$ is defined as the complement of $\downarrow$.

The basic rule for input has the form $c?.[x]p \xrightarrow{c?} [x]p$, the one for output is $c!.(v, \mathbf{q}) \xrightarrow{c!} (v, \mathbf{q})$ and that for communication expresses the fact that synchronization takes the form of functions application:

$$\frac{\mathbf{p} \xrightarrow{c?} \mathbf{f}, \mathbf{q} \xrightarrow{c!} (v, \mathbf{q}')}{\mathbf{p}|\mathbf{q} \xrightarrow{\tau} \mathbf{f}(v)|\mathbf{q}'}\ .$$

As an example of an application of the inference rules, let us have a look at the processes given by Ω and $recP.P$. By inspection of the rules it is not difficult to see that neither of them can be proven to be convergent which means that they both are divergent. Furthermore we can also see that neither of them can perform any action and therefore they must be equivalent according to the late strong bisimulation semantics.

The bisimulation preorder $\precsim$ defined on the $ALTS$ as described above satisfies:

Theorem 3.5

1. *$\precsim$ is a pre-congruence with respect to the operators in Σ.*
2. (a) *For all $\mathbf{p}_1, \mathbf{p}_2$, $\mathbf{p}_1 \precsim \mathbf{p}_2$ implies $\tau.\mathbf{p}_1 \precsim \tau.\mathbf{p}_2$.*

$$(input) \quad c?.\mathbf{f} \xrightarrow{c?} \mathbf{f}$$

$$(output) \quad c!.(e, \mathbf{p}) \xrightarrow{c!} (v, \mathbf{p}), \ [\![e]\!] = v$$

$$(tau) \quad \tau.\mathbf{p} \xrightarrow{\tau} \mathbf{p}$$

$$(ren) \quad \frac{\mathbf{p} \xrightarrow{c?} [x]p'}{\mathbf{p}[R] \xrightarrow{R(c)?} [x](p'[R])}$$

$$\frac{\mathbf{p} \xrightarrow{c!} (v, \mathbf{p}')}{\mathbf{p}[R] \xrightarrow{R(c)!} (v, \mathbf{p}'[R])}$$

$$\frac{\mathbf{p} \xrightarrow{\tau} \mathbf{p}'}{\mathbf{p}[R] \xrightarrow{\tau} \mathbf{p}'[R]}$$

$$(res) \quad \frac{\mathbf{p} \xrightarrow{c?} [x]p'}{\mathbf{p}\backslash c' \xrightarrow{c?} [x](p'\backslash c')}, \ c \neq c'$$

$$\frac{\mathbf{p} \xrightarrow{c!} (v, \mathbf{p}')}{\mathbf{p}\backslash c' \xrightarrow{c!} (v, \mathbf{p}'\backslash c')}, \ c \neq c'$$

$$\frac{\mathbf{p} \xrightarrow{\tau} \mathbf{p}'}{\mathbf{p}\backslash c' \xrightarrow{\tau} \mathbf{p}'\backslash c'}$$

$$(choice) \quad \frac{\mathbf{p} \xrightarrow{\mu} ct}{\mathbf{p} + \mathbf{q} \xrightarrow{\mu} ct}$$

$$(par) \quad \frac{\mathbf{p} \xrightarrow{c?} [x]p'}{\mathbf{p} \mid \mathbf{q} \xrightarrow{c?} [x](p' \mid \mathbf{q})}$$

$$\frac{\mathbf{p} \xrightarrow{c!} (v, \mathbf{p}')}{\mathbf{p} \mid \mathbf{q} \xrightarrow{c!} (v, \mathbf{p}' \mid \mathbf{q})}$$

$$\frac{\mathbf{p} \xrightarrow{\tau} \mathbf{p}'}{\mathbf{p} \mid \mathbf{q} \xrightarrow{\tau} \mathbf{p}' \mid \mathbf{q}}$$

$$(com) \quad \frac{\mathbf{p} \xrightarrow{c?} [x]p', \ \mathbf{q} \xrightarrow{c!} (v, \mathbf{q}')}{\mathbf{p} \mid \mathbf{q} \xrightarrow{\tau} p'[v/x] \mid \mathbf{q}'}$$

$$(cond) \quad \frac{\mathbf{p} \xrightarrow{\mu} ct}{(be \longrightarrow \mathbf{p}, \mathbf{q}) \xrightarrow{\mu} ct} [\![be]\!] = T$$

$$\frac{\mathbf{q} \xrightarrow{\mu} ct}{(be \longrightarrow \mathbf{p}, \mathbf{q}) \xrightarrow{\mu} ct} [\![be]\!] = F$$

$$(rec) \quad \frac{\mathbf{p}[recP.\mathbf{p}/P] \xrightarrow{\mu} \mathbf{p}'}{recP.\mathbf{p} \xrightarrow{\mu} \mathbf{p}'}$$

Figure 1: Operational semantics for $\mathbf{C}CS_L$; (*choice*), (*par*) and (*com*) have symmetric counterparts.

$$\frac{}{nil \downarrow} \qquad \frac{\mathbf{p} \downarrow, \mathbf{p}' \downarrow}{\mathbf{p} + \mathbf{p}' \downarrow} \qquad \frac{\mathbf{p} \downarrow, \mathbf{p}' \downarrow}{\mathbf{p} \mid \mathbf{p}' \downarrow} \qquad \frac{\mathbf{p} \downarrow}{\mathbf{p}\backslash c \downarrow} \qquad \frac{\mathbf{p} \downarrow}{\mathbf{p}[R] \downarrow} \qquad \frac{\mathbf{p}[\Omega/P] \downarrow}{recP.\mathbf{p} \downarrow}$$

$$\frac{[\![be]\!] = T, \mathbf{p}_1 \downarrow}{be \longrightarrow \mathbf{p}_1, \mathbf{p}_2 \downarrow} \qquad \frac{[\![be]\!] = F, \mathbf{p}_2 \downarrow}{be \longrightarrow \mathbf{p}_1, \mathbf{p}_2 \downarrow}$$

Figure 2: The convergence predicate

(b) *For all $c \in Chan$ and $\mathbf{o}_1, \mathbf{o}_2$, $\mathbf{o}_1 \precapprox \mathbf{o}_2$ implies $c!.\mathbf{o}_1 \precapprox c!.\mathbf{o}_2$.*

(c) *For all $c \in Chan$ and $\mathbf{f}_1, \mathbf{f}_2$, $\mathbf{f}_1 \precapprox \mathbf{f}_2$ implies $c?.\mathbf{f}_1 \precapprox c?.\mathbf{f}_2$.*

3. (a) *For all $p_1, p_2 \in CCS_L^{proc}$ with $\{y | y$ free value-variable in $p_1, p_2\} \subseteq \{x\}$, whenever $p_1[v/x] \precapprox p_2[v/x]$ for every $v \in Val$ then $[x]p_1 \precapprox [x]p_2$.*

   (b) *For all $\mathbf{p}_1, \mathbf{p}_2$ and v, $\mathbf{p}_1 \precapprox \mathbf{p}_2$ implies $(v, \mathbf{p}_1) \precapprox (v, \mathbf{p}_2)$.*

Proof

1. We have to prove that for any operator $op \in \Sigma$, $\underline{\mathbf{p}} \precapprox \underline{\mathbf{q}} \;\; \Rightarrow \;\; op(\underline{\mathbf{p}}) \precapprox op(\underline{\mathbf{q}})$. We will only prove the statement for the case $op = \_|\_$, leaving the remaining cases to the interested reader to check.

   So assume $\mathbf{p}_1 \precapprox \mathbf{p}_2$ and $\mathbf{q}_1 \precapprox \mathbf{q}_2$. This means that there are sa-prebisimulations $\mathcal{R}_p$ and $\mathcal{R}_q$ such that $(\mathbf{p}_1, \mathbf{p}_2) \in \mathcal{R}_p$ and $(\mathbf{q}_1, \mathbf{q}_2) \in \mathcal{R}_q$. We define

   $$\mathcal{R}_p | \mathcal{R}_q = \{(\mathbf{p}'_1 | \mathbf{q}'_1, \mathbf{p}'_2 | \mathbf{q}'_2) \,\big|\, (\mathbf{p}'_1, \mathbf{p}'_2) \in \mathcal{R}_p, (\mathbf{q}'_1, \mathbf{q}'_2) \in \mathcal{R}_q\}.$$

   As $(\mathbf{p}_1 | \mathbf{q}_1, \mathbf{p}_2 | \mathbf{q}_2) \in \mathcal{R}_p | \mathcal{R}_q$ it is sufficient to show that $\mathcal{R}_p | \mathcal{R}_q$ is a sa-prebisimulation. To prove this we proceed as follows:

   (a) Assume that $(\mathbf{p}'_1 | \mathbf{q}'_1, \mathbf{p}'_2 | \mathbf{q}'_2) \in \mathcal{R}_p | \mathcal{R}_q$ and that $\mathbf{p}'_1 | \mathbf{q}'_1 \xrightarrow{\mu} \mathbf{r}_1$. We only consider the following cases:

      i. $\mu = c?$, $\mathbf{p}'_1 \xrightarrow{c?} [x]p''_1$ and $\mathbf{r}_1 = [x](p''_1 | \mathbf{q}'_1)$. Now there is a $[y]p''_2$ where $\mathbf{p}'_2 \xrightarrow{c?} [y]p''_2$ and $([x]p''_1, [y]p''_2) \in \mathcal{R}^{fun}$, i.e. for all $v \in Val$, $(p''_1[v/x], p''_2[v/y]) \in \mathcal{R}_p$. As $\mathbf{q}'_1$ and $\mathbf{q}'_2$ do not contain free value-variables this implies that for all $v \in Val$

      $$((p''_1 | \mathbf{q}'_1)[v/x], (p''_2 | \mathbf{q}'_2)[v/y]) = ((p''_1[v/x] | \mathbf{q}'_1), (p''_2[v/y] | \mathbf{q}'_2)) \in \mathcal{R}_p | \mathcal{R}_q.$$

      This shows that $([x](p''_1 | \mathbf{q}'_1), [y](p''_2 | \mathbf{q}'_2) \in (\mathcal{R}_p | \mathcal{R}_q)^{fun}$. Furthermore $\mathbf{p}'_2 | \mathbf{q}'_2 \xrightarrow{c?} [y](p''_2 | \mathbf{q}'_2)$.

      ii. $\mu = \tau$, $\mathbf{p}'_1 \xrightarrow{c?} [x]p''_1$, $\mathbf{q}'_1 \xrightarrow{c!} (v, \mathbf{q}''_1)$ and $\mathbf{r}_1 = p''_1[v/x] | \mathbf{q}''_1$. Then $\mathbf{p}'_2 \xrightarrow{c?} [y]p''_2$ where $([x]p''_1, [y]p''_2) \in \mathcal{R}_p^{fun}$, i.e for all $v \in Val$, $(p''_1[v/x], p''_2[v/y]) \in \mathcal{R}_p$. Furthermore $\mathbf{q}'_2 \xrightarrow{c!} (v', \mathbf{q}''_2)$ where $((v, \mathbf{q}''_1), (v', \mathbf{q}''_2) \in \mathcal{R}_q^{pair}$, i.e. where $v = v'$ and $(\mathbf{q}''_1, \mathbf{q}''_2) \in \mathcal{R}_q$. This implies that $(p''_1[v/x] | \mathbf{q}''_1, p''_2[v/y] | \mathbf{q}''_2) \in \mathcal{R}_p | \mathcal{R}_q$. Furthermore $\mathbf{p}'_2 | \mathbf{q}'_2 \xrightarrow{\tau} p''_2[v/y] | \mathbf{q}''_2$.

   (b) Next assume that $\mathbf{p}'_1 | \mathbf{q}'_1 \downarrow$. This implies that $\mathbf{p}'_1 \downarrow$ and $\mathbf{q}'_1 \downarrow$ and therefore that $\mathbf{p}'_2 \downarrow$ and $\mathbf{q}'_2 \downarrow$. This in turn implies that $\mathbf{p}'_2 | \mathbf{q}'_2 \downarrow$. Now assume that $\mathbf{p}'_1 | \mathbf{q}'_1 \downarrow$, $\mathbf{p}'_2 | \mathbf{q}'_2 \downarrow$ and $\mathbf{p}'_2 | \mathbf{q}'_2 \xrightarrow{\mu} \mathbf{r}_2$. In the same way as in (a) we may show that $\mathbf{p}'_1 | \mathbf{q}'_1 \xrightarrow{\mu} \mathbf{r}_1$ for some $\mathbf{r}_1$ such that $(\mathbf{r}_1, \mathbf{r}_2) \in \mathcal{R}_p | \mathcal{R}_q$.

2. Here we will only prove the last case, i.e. that $\mathbf{f}_1 \precapprox \mathbf{f}_2$ implies $c?.\mathbf{f}_1 \precapprox c?.\mathbf{f}_2$. So assume that $\mathbf{f}_1 = [x]p$ and $\mathbf{f}_2 = [y]q$ and that $[x]p \precapprox [y]q$. This implies that $([x]p, [y]q) \in \mathcal{R}^{fun}$ for some sa-prebisimulation $\mathcal{R}$. We define $c?.\mathcal{R} = \mathcal{R} \cup \{(c?.[x]p, c?.[y]q)\}$. Obviously $(c?.[x]p, c?.[y]q) \in c?.\mathcal{R}$. It is also easy to see that $c?.\mathcal{R}$ is an sa-prebisimulation.

3. This is just a rephrasing of the definition of the extension of the relations from Con to $Val \longrightarrow Con$ and $Val \times Con$ in the case when $Con = CCS_L^{proc}$.

$\square$

4 Denotational Semantics

In the previous section we extended the standard notion of a signature Σ to that of applicative signature (Σ, C). In this section we define the general class of (Σ, C)-domains which is a direct generalization of the standard Σ-domains introduced in [GTWW77]. In fact the (Σ, C)-domains are only a slight modification of the Natural Interpretations introduced in [HP80] and applied in [HI93]. We also introduce the notion of *recursively closed* subsets of a process algebra.

After having presented the general models we show how we may obtain a (Σ, C)-domain by defining a preordered set that represents the compact elements, defining the operators as monotonic functions over this set and then taking the domain to be the unique extension of the kernel of the preorder and the induced monotonic functions to such a structure. We also study the relationship between the evaluation mappings from our generic process language into two arbitrary (Σ, C)-preorders. In what follows we abstract away from possible structures or properties of the value domain Val and simply view it as being partially ordered by the discrete order.

4.1 (Σ, C)-Orders and (Σ, C)-Domains

In this subsection we define the notion of applicative orders and applicative ordered (Σ, C)-algebras. We borrow the notation from [Hen88a] and use the abbreviations *pro* for preorder, *po* for partial order and *cpo* for complete partial order together with their domains. We assume that the reader is familiar with basic domain theory and algebraic semantics. (See e.g. [Plo81, Hen88a] for details.)

Definition 4.1 (Applicative Orders) A pair $\langle \overline{A}, <_{\overline{A}} \rangle$ is an applicative *pro/po/cpo* if $\overline{A} = (A_{proc}, A_{fun}, A_{pair})$ and $<_{\overline{A}} = (<_{A_{proc}}, <_{A_{fun}}, <_{A_{pair}})$ are such that:

1. $\langle A_{proc}, <_{A_{proc}} \rangle$ is a *pro/po/cpo* and
2. $A_{fun} \subseteq Val \longrightarrow A_{proc}$ and $A_{pair} \subseteq Val \times A_{proc}$ are *pro/po/cpo*s with the standard induced ordering, i.e. $<_{A_{fun}}$ is the pointwise ordering and $<_{A_{pair}}$ is defined by:

$$(v_1, p_1) <_{A_{pair}} (v_2, p_2) \text{ if } v_1 = v_2 \text{ and } p_1 <_{A_{proc}} p_2.$$

$\overline{A}$ is said to be *fully applicative* if $A_{proc} = A$, $A_{fun} = Val \longrightarrow A$ and $A_{pair} = Val \times A$ for some A. In that case we refer to $\overline{A}$ as A. $\overline{A}$ is said to be *finitely applicative* if $A_{pair} = Val \times A$ for some A and

$$A_{fun} = Val \longrightarrow_{fin} A = \{f \in Val \longrightarrow A \,|\, \{a | f(a) \neq \bot\} \text{ is finite}\}.$$

In this case we refer to $\overline{A}$ as A^{fin}. An applicative *cpo* is said to be algebraic/ω-algebraic if A_{proc}, A_{fun} and A_{pair} are algebraic/ω-algebraic *cpo*s. □

Here we want to point out that the partial order of compact elements of an applicative algebraic *cpo* is in general not fully applicative but finitely applicative. We often write a-pro/po/cpo as a shorthand for applicative pro/po/cpo.

Definition 4.2 ((Σ, C)-Orders) A four tuple $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ is an applicative $(\Sigma, C)-$ *pro/po/cpo* if $\overline{A} = (A_{proc}, A_{fun}, A_{pair})$ is such that

1. $\langle \overline{A}, <_{\overline{A}} \rangle$ is an a-*pro*/*po*/*cpo*,
2. $\Sigma_{\overline{A}}$ is a set of monotonic/monotonic/continuous functions $op_{\overline{A}} : A_{proc} \longrightarrow A_{proc}$,
3. $C_{\overline{A}} = C!_{\overline{A}} \cup C?_{\overline{A}}$ where:
   (a) $C!_{\overline{A}}$ is a set of monotonic/monotonic/continuous functions $c!._{\overline{A}} : A_{pair} \longrightarrow A_{proc}$ and
   (b) $C?_{\overline{A}}$ is a set of monotonic/monotonic/continuous functions $c?_{\overline{A}} : A_{fun} \longrightarrow A_{proc}$.

An ω-algebraic applicative $(\Sigma, C) - cpo$ is called a (Σ, C)-*domain*. For an algebraic cpo A we use $Comp(A)$ to denote the set of compact elements of A. Usually we use $<$, $\prec$, $\sqsubset$ etc. to denote preorders but $\leq$, $\preceq$, $\sqsubseteq$ to denote partial orders, including the complete ones. □

Definition 4.3 A function $f : A_1 \longrightarrow A_2$, where $\langle A_1, \leq_1 \rangle$ and $\langle A_2, \leq_2 \rangle$ are algebraic *cpo*s, is said to be *compact* if it maps compact elements of A_1 into compact elements of A_2, i.e. if $f(Comp(A_1)) \subseteq Comp(A_2)$. □

Next we extend the standard notion of homomorphisms for applicative orders.

Definition 4.4 A a-*pro*/*po*/*cpo* homomorphism from $\langle \overline{A}, <_{\overline{A}} \rangle$ to $\langle \overline{B}, <_{\overline{B}} \rangle$ is a monotonic function $h : A_{proc} \longrightarrow B_{proc}$ that satisfies

1. $h(op_{\overline{A}}(\overline{a})) = op_{\overline{B}}(h(\overline{a}))$,
2. $h(c?_{\overline{A}}.F) = c?_{\overline{B}}.(h \circ F)$ and
3. $h(c!_{\overline{A}}.(v, \overline{a})) = c!_{\overline{B}}.(v, h(\overline{a}))$.

We define $h_{proc} = h$, $h_{fun}(F) = h \circ F$ for all $F \in A_{fun}$ and $h_{pair}(v, a) = (v, h(a))$ for all $(v, a) \in A_{pair}$. Sometimes we refer to the triplet $\overline{h} = (h_{proc}, h_{fun}, h_{pair})$ as such a homomorphism. □

At times it is useful to be able to apply structural induction on a sublanguage of the full language defined by an a-signature (Σ, C) and a set of process names PN. In particular we want to be able to give recursive definitions on certain sublanguages (e.g. the language that denotes the compact elements of the model). This motivates the following definition of a *recursively closed* subset of a language.

Definition 4.5 $\overline{S} = (S_{proc}, S_{fun}, S_{pair}) \subseteq T_{(\Sigma,C)}(PN)$ is said to be *recursively closed* if the following hold:

1. $p = op(p_1, \ldots, p_n) \in S_{proc}$ implies $p_i \in S_{proc}$ for $i = 1, \ldots, n$,
2. $[x]p \in S_{proc}$ implies $p[v/x] \in S_{proc}$ for all $v \in Val$,
3. $(e, p) \in S_{proc}$ implies $p \in S_{proc}$,
4. $c?.f \in S_{proc}$ implies $f \in S_{fun}$,
5. $c!.o \in S_{proc}$ implies $o \in S_{pair}$,
6. $be \longrightarrow p_1, p_2 \in S_{proc}$ implies $p_1, p_2 \in S_{proc}$,

7. $recP.p \in S_{proc}$ implies $P, p \in S_{proc}$.

In this case we write $\overline{S} \subseteq_{rec} T_{(\Sigma,C)}(PN)$. $\square$

Note that if $\Sigma' \subseteq \Sigma$ and $C' \subseteq C$ then $T_{(\Sigma',C')} \subseteq_{rec} T_{(\Sigma,C)}(PN)$.

Definition 4.6 Let $\overline{S} \subseteq_{rec} T_{(\Sigma,C)}(PN)$, where $\overline{S}$ only contains value closed terms, $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ be an applicative $(\Sigma, C) - pro$ and $PEnv_{\overline{A}}$ be the set of process environments $\rho : PN \longrightarrow A_{proc}$. A function

$$\overline{A}[\![\_]\!] : S_{proc} \longrightarrow (PEnv_{\overline{A}} \longrightarrow A_{proc}))$$

is an *evaluation* function if for each $\rho \in PEnv_{\overline{A}}$, $\overline{A}[\![\_]\!]\rho : S_{proc} \longrightarrow A_{proc}$ is a (Σ, C)-pro homomorphism, where $\overline{S}$ is the (Σ, C)-pro obtained by taking $\overline{S}$ with the discrete order and the operators to be the syntactic ones from Σ and C, and if the following holds:

$$\begin{array}{l} \overline{A}[\![P]\!]\rho = \rho(P) \text{ and} \\ \overline{A}[\![\ be \rightarrow p_1, p_2]\!]\rho = \begin{cases} \overline{A}[\![p_1]\!]\rho & \text{if } [\![be]\!] = T, \\ \overline{A}[\![p_2]\!]\rho & \text{if } [\![be]\!] = F. \end{cases} \end{array}$$

If $\overline{A}$ is a *cpo* then, following standard practice, we may define

$$\overline{A}[\![recP.\mathbf{p}]\!]\rho \quad = \quad Y\ \lambda d.\overline{A}[\![\mathbf{p}]\!]\rho[d/P]$$

where Y is the least fixed point operator. $\square$

For closed terms the environments do not have any influence on the definition of the semantics. For terms without process names, a mapping $\overline{A}[\![\mathbf{t}]\!] = \overline{A}[\![\mathbf{t}]\!]\rho$ may be derived from the above definition omitting the last clause of the definition and the occurrence of ρ in the others. Now we show that recursively closed subsets of $T_{(\Sigma,C)}(PN)$ have at most one interpretation in an a-(Σ, C)-pro. This is the subject of the next theorem.

Theorem 4.7 *Let $\overline{S} = (S_{proc}, S_{fun}, S_{pair}) \subseteq_{rec} T_{(\Sigma,C)}(PN)$ and $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ be an applicative $(\Sigma, C) - pro$. Then there is at most one evaluation mapping*

$$\overline{A}[\![\_]\!] : S_{proc} \longrightarrow (PEnv_{\overline{A}} \longrightarrow \overline{A}).$$

If $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ is fully applicative then such an evaluation mapping exists.

Proof May be proved by structural induction and is left to the reader. $\square$

Note that if $\overline{A}$ is not fully applicative then a function term of the form $[x]p$, where p only has x as a free variable, may fail to have an interpretation in $\overline{A}$. For instance, if $\overline{A}$ is the a-po of compact elements of some (Σ, C)-domain $\overline{A}$ and p is a process term denoting a compact element of A_{proc} different from $\perp$, then the function term $[x]p$ fails to have an interpretation in A_{fun}.

The following result turns out to be useful in the next section.

Corollary 4.8 *Assume that*

$$\overline{S} = (S^{proc}, S^{fun}, S^{pair}) \subseteq_{rec} (CCS_L^{proc}, CCS_L^{fun}, CCS_L^{pair}),$$

that $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ *and* $\langle \overline{B}, <_{\overline{B}}, \Sigma_{\overline{B}}, C_{\overline{B}} \rangle$ *are* (Σ, C)*-pros and that*

$$\psi : \langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle \longrightarrow \langle \overline{B}, <_{\overline{B}}, \Sigma_{\overline{B}}, C_{\overline{B}} \rangle$$

is a (Σ, C)*-pro homomorphism. If* $\overline{A}[\![\_]\!] : S_{proc} \longrightarrow \langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ *and* $\overline{B}[\![\_]\!] : S_{proc} \longrightarrow \langle \overline{B}, <_{\overline{B}}, \Sigma_{\overline{B}}, C_{\overline{B}} \rangle$ *are evaluation mappings, then* $\overline{B}[\![\_]\!] = \psi \circ \overline{A}[\![\_]\!]$.

Proof It is easy to check that the mapping $\overline{B}[\![\_]\!]$ defined by $\overline{B}[\![\_]\!] = \overline{\psi} \circ A[\![\_]\!]$ is an evaluation mapping from $\overline{S}$ to $\langle \overline{B}, <_{\overline{B}}, \Sigma_{\overline{B}}, C_{\overline{B}} \rangle$. By Theorem 4.7 such an evaluation mapping is unique and the equality follows. □

There is a standard way of extending a *preorder* with a least element to an algebraic *cpo*, often called *completion by ideals* ([Hen88a, §3.3], [Win85]). If $\langle A, <_A \rangle$ is a preorder, a set $X \subseteq A$ is *downwards closed* if whenever $x \in X$ and $y \sqsubseteq_A x$ then $y \in X$ and *directed* if whenever $x, y \in X$ then, for some $z \in X$, $x < z$ and $y < z$. An *ideal* in A is a non-empty, directed and downwards closed subset of A. Let $\mathcal{I}(A)$ denote the set of all ideals in A. If A has a least element then $\langle \mathcal{I}(A), \subseteq \rangle$ is an algebraic *cpo*. The compact elements of $\mathcal{I}(A)$ are $Comp(\mathcal{I}(A)) = \{\downarrow a | a \in A\}$ where $\downarrow a = \{x | x \sqsubseteq_A a\}$. $\mathcal{I}(A)$ is the unique algebraic *cpo* (up to isomorphism) whose partial order of compact elements consists of the kernel of $\langle A, \sqsubseteq_A \rangle$, i.e. $\langle A/ =_A, \sqsubseteq_{A/=_A} \rangle$ where $=_A$ is the equivalence induced by $\sqsubseteq_A$. This is referred to as the ideal completion of $\langle A, \sqsubseteq_A \rangle$. Note that if $A/ =_A$ is countable then $\mathcal{I}(A)$ is ω-algebraic.

We have the following standard theorem (see e.g. [Hen88a]).

Theorem 4.9 *Let* $\langle M, \sqsubseteq_M \rangle$ *be a partial order,* $\langle A, \sqsubseteq_A \rangle$ *a cpo and* $f : M \longrightarrow A$ *be monotonic. Then there is a unique continuous extension* $\tilde{f}$ *of* f *to* $\mathcal{I}(M)$.

These results generalize to the applicative orders as follows. Let $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$ be a (Σ, C)-pro. We may define a continuous (Σ, C) structure on $\mathcal{I}(\overline{A}) = (\mathcal{I}(A_{proc}), \mathcal{I}(A_{fun}), \mathcal{I}(A_{pair}))$ as follows: Let $\langle \overline{A}/\approx, \sqsubseteq_{\overline{A}/\approx} \rangle$ denote the a-po induced by $\langle \overline{A}, <_{\overline{A}} \rangle$ and $[\_]_\approx : \overline{A} \longrightarrow \overline{A}/\approx$ denote the quotient mapping. For $op_{\overline{A}} \in \Sigma_{\overline{A}}$, we define $op_{\overline{A}/\approx}$ on $\overline{A}/\approx$ by $op_{\overline{A}/\approx}([x]_\approx) = [op_{\overline{A}}(x)]_\approx$. It is easy to see that this defines an operator in $\overline{A}/\approx$ that preserves the order. Now we define $op_{\mathcal{I}(\overline{A})}$ to be the unique continuous extension of $op_{\overline{A}/\approx}$ as described in Theorem 4.9. This gives continuous (Σ, C) structure that turns $\mathcal{I}(\overline{A})$ into a (Σ, C)-domain. We refer to this domain as the *domain induced by* $\langle \overline{A}, <_{\overline{A}}, \Sigma_{\overline{A}}, C_{\overline{A}} \rangle$. Furthermore, by Corollary 4.8, if $\overline{S} \subseteq_{rec} T_{(C,\Sigma)}$ and $\overline{A}[\![\_]\!] : \overline{S} \longrightarrow \overline{A}$ is an evaluation mapping then $\mathcal{I}(\overline{A})[\![\_]\!]|_{\overline{S}} = [\_]_\approx \circ \overline{A}[\![\_]\!]$ where $\mathcal{I}(\overline{A})[\![\_]\!]|_{\overline{S}}$ means the restriction of the function $\mathcal{I}(\overline{A})[\![\_]\!]$ to $\overline{S}$. In particular this implies that for all $s_1, s_2 \in \overline{S}$

$$\mathcal{I}(\overline{A})[\![s_1]\!] \sqsubseteq_{\mathcal{I}(\overline{A})} \mathcal{I}(\overline{A})[\![s_2]\!] \text{ iff } \overline{A}[\![s_1]\!] <_{\overline{A}} \overline{A}[\![s_2]\!].$$

4.2 Definition of the Model

This section is devoted to defining the concrete model $\overline{ACT}$. First we give a description of a representation of the compact elements.

Definition 4.10 We define K as the least set which satisfies:

1. $\emptyset,\ \{\perp\} \in K$,
2. $c \in C, V \subseteq_{fin} Val$ and $\forall v \in V.\, k_v \in K$ implies $\{\langle c?, \lambda v.x \in V \longrightarrow k_v, \Omega\rangle\} \in K$,
3. $c \in C, v \in Val$ and $k \in K$ implies $\{\langle c!, (v,k)\rangle\} \in K$,
4. $k \in K$ implies $\{\langle \tau, k\rangle\} \in K$,
5. $k_1, k_2 \in K$ implies $k_1 \cup k_2 \in K$.

The preorder $\prec$ is defined as the least preorder on K which satisfies

1. $\{\perp\} \prec \emptyset$
2. $k_1 \prec k_2$ if $\forall a \in k_1 \exists b \in k_2.\, a < b$ and $\forall b \in k_2 \exists a \in k_1.\, a < b$
   where $<$ is defined on the elements of the sets in K by
   (a) $\forall a.\, \perp < a$,
   (b) $\langle \tau, k\rangle < \langle \tau, k'\rangle$ iff $k \prec k'$,
   (c) $\langle c?, f\rangle < \langle c?, g\rangle$ iff $\forall v \in Val. f(v) \prec g(v)$,
   (d) $\langle c!, (v,k)\rangle \ < \langle c!, (v,k')\rangle$ iff $k \prec k'$.

We let $\approx = \prec \cap \prec^{-1}$. □

Definition 4.11 We define $\langle \overline{K}, \prec_{\overline{K}}\rangle$ to be $\langle K^{fin}, \prec_{fin}\rangle$, i.e. the finitely applicative preorder induced by K. □

Next we define the operators on the applicative preorder $\langle \overline{K}, \prec_{\overline{K}}\rangle$ and make sure that they are monotonic.

Definition 4.12 We define $\Sigma_{\overline{K}}$ as follows:

Constants:

$$\begin{aligned} nil_{\overline{K}} &= \emptyset, \\ \Omega_{\overline{K}} &= \{\perp\}. \end{aligned}$$

Prefixing:

$$\begin{aligned} c?_{\overline{K}}.\_ &= \lambda f.\{\langle c?, f\rangle\}, \\ c!_{\overline{K}}.\_ &= \lambda(v,k).\{\langle c!, (v,k)\rangle\}, \\ \tau_{\overline{K}}.\_ &= \lambda k.\{\langle \tau, k\rangle\}. \end{aligned}$$

Nondeterminism:

$$+_{\overline{K}} \ = \ \cup.$$

Restriction:

$$\_\backslash c_{\overline{K}} = F_c$$

where $F_c : K_{proc} \to K_{proc}$ is defined by

$$\begin{array}{rcl}
F_c\{\bot\} & = & \{\bot\}, \\
F_c\emptyset & = & \emptyset, \\
F_c\{\langle b?, f\rangle\} & = & \begin{cases} \{\langle b?, F_c \circ f\rangle\} & \text{if } b \neq c, \\ \emptyset & \text{otherwise}, \end{cases} \\
F_c\{\langle b!, (v, k)\rangle\} & = & \begin{cases} \{\langle b!, (v, F_c k)\rangle\} & \text{if } b \neq c, \\ \emptyset & \text{otherwise}, \end{cases} \\
F_c\{\langle \tau, k\rangle\} & = & \{\langle \tau, F_c k\rangle\}, \\
F_c(k_1 \cup k_2) & = & (F_c k_1) \cup (F_c k_2).
\end{array}$$

Renaming:

$$\_[R]_{\overline{K}} = G_R$$

where $G_R : K_{proc} \to K_{proc}$ is defined by

$$\begin{array}{rcl}
G_R\{\bot\} & = & \{\bot\}, \\
G_R\emptyset & = & \emptyset, \\
G_R\{\langle c?, f\rangle\} & = & \{\langle R(c)?, G_R \circ f\rangle\}, \\
G_R\{\langle c!, (v, k)\rangle\} & = & \{\langle R(c)!, (v, G_R k)\rangle\}, \\
G_R\{\langle \tau, k\rangle\} & = & \{\langle \tau, G_R k\rangle\}, \\
G_R(k_1 \cup k_2) & = & (G_R k_1) \cup (G_R k_2).
\end{array}$$

Parallel Composition:

$$\_ \,|_{\overline{K}}\, \_ = F$$

where $F = \mathit{int} \cup \mathit{comm} \cup \mathit{div}$ where $\mathit{int} = \mathit{int}_{in} \cup \mathit{int}_{out} \cup \mathit{int}_\tau$ and

$$\begin{array}{rcl}
\mathit{int}_{in}(x, y) & = & \{\langle c_x?, \lambda v.F(f_x(v), y)\rangle | \langle c_x?, f_x\rangle \in x\} \\
& \cup & \{\langle c_y?, \lambda v.F(x, f_y(v))\rangle | \langle c_y?, f_y\rangle \in y\}, \\
\mathit{int}_{out}(x, y) & = & \{\langle c_x!, (v, F(x', y))\rangle | \langle c_x!, (v, x')\rangle \in x\} \\
& \cup & \{\langle c_y!, (v, F(x, y'))\rangle | \langle c_y!, (v, y')\rangle \in y\}, \\
\mathit{int}_\tau & = & \{\langle \tau, F(x', y)\rangle | \langle \tau, x'\rangle \in x\} \\
& \cup & \{\langle \tau, F(x, y')\rangle | \langle \tau, y'\rangle \in y\},
\end{array}$$

$$\begin{array}{rcl}
\mathit{comm}(x, y) & = & \{\langle \tau, F(f(v), y')\rangle \,|\, \exists c, v.\langle c?, f\rangle \in x \text{ and } \langle c!, (v, y')\rangle \in y\} \\
& \cup & \{\langle \tau, F(x', g(v))\rangle \,|\, \exists c, v.\langle c?, g\rangle \in y \text{ and } \langle c!, (v, x')\rangle \in x\}
\end{array}$$

and

$$div(x,y) = \begin{cases} \{\perp\} & \text{if } \perp \in x \cup y, \\ \emptyset & \text{otherwise.} \end{cases}$$

□

The reader may notice the close connection between the definition of the parallel operator and the interleaving law presented later in the paper. We have the following result:

Lemma 4.13 *$\langle \overline{K}, \prec_{\overline{K}}, \Sigma_{\overline{K}}, C_{\overline{K}} \rangle$ is a (Σ, C)-pro.*

Proof We leave it to the reader to check that the operators defined by Definition 4.12 are well-defined. The monotonicity of the operators $nil_{\overline{K}}$, $\Omega_{\overline{K}}$, $c?_{\overline{K}}$, $c!_{\overline{K}}$, $\tau_{\overline{K}}$ and $+_{\overline{K}}$ is obvious. To prove the monotonicity of the remaining operators we use the depth, $d(\_)$, of the elements of K defined in the proof of Proposition 4.18. To prove the monotonicity of the restriction and the renaming operators we prove by induction on $d(k)$ that $k \prec k'$ implies $F_c k \prec F_c k'$ and $k \prec k'$ implies $G_R k \prec G_R k'$. To prove the monotonicity of the parallel operator with respect to the induced ordering on $K \times K$, we extend d to $K \times K$ by $d(k_1, k_2) = d(k_1) + d(k_2)$. Then we may prove that

$$(k_1, k_2) \prec (k_1', k_2') \text{ implies } F(k_1, k_2) \prec F(k_1', k_2')$$

by induction on $d(k_1, k_2)$. We leave the straightforward details of the proof to the reader. □

Now we let $\langle \overline{ACT}, \sqsubseteq_{\overline{ACT}}, \Sigma_{\overline{ACT}}, C_{\overline{ACT}} \rangle$ be the (unique) fully applicative (Σ, C)-domain induced by $\langle \overline{K}, \prec_{\overline{K}}, \Sigma_{\overline{K}}, C_{\overline{K}} \rangle$. The operators in $\Sigma_{\overline{ACT}}$ and $C_{\overline{ACT}}$ are compact.

4.3 Syntactically Compact Elements

Next we will show that the compact elements of the model $\overline{ACT}$ may be denoted in our syntax by a recursively closed subset of the whole language. For this purpose we introduce the so-called *syntactically compact terms*, $\mathbf{co}CCS_L = (\mathbf{co}CCS_L^{proc}, \mathbf{co}CCS_L^{fun}, \mathbf{co}CCS_L^{pair})$.

As usual, syntactically finite terms are those without occurrences of recursion. We define syntactically compact terms as the syntactically finite ones which only use a finite number of values. Note that, as we are dealing with recursion free terms, the number of channels used by the term is automatically finite. We start by introducing some notation.

Notation 4.14 *Let $\underline{w} = (w_1, \ldots, w_n)$ and $\underline{p} = (p_1, \ldots, p_n)$ be vectors of values and processes respectively. We write $x : \underline{w} \longrightarrow \underline{p}$ for*

$$x = w_1 \longrightarrow p_1, (x = w_2 \longrightarrow p_2, (\ldots x = w_n \longrightarrow p_n, \Omega) \ldots).$$

(Intuitively $x : \underline{w} \longrightarrow \underline{p}$ stands for the mapping that maps w_i to p_i for $i = 1, \ldots, n$ and all the other values $w \in Val$ into Ω.) Furthermore we let $\{\underline{w}\} = \{w_i | \underline{w_n} = (w_1, \ldots, w_n), i \leq n\}$ and similarly for $\{\underline{p}\}$.

Definition 4.15 **[Syntactically Compact Terms]** The set of syntactically compact terms is the triplet $\mathbf{co}CCS_L = (\mathbf{co}CCS_L^{proc}, \mathbf{co}CCS_L^{fun}, \mathbf{co}CCS_L^{pair})$, where $\mathbf{co}CCS_L^{proc}$, $\mathbf{co}CCS_L^{fun}$ and $\mathbf{co}CCS_L^{pair}$ are the least sets satisfying:

1. $nil, \Omega \in \mathbf{c}oCCS_L^{proc}$,

2. $\overline{\mathbf{p}} \in \mathbf{c}oCCS_L^{proc}$ implies $op(\overline{\mathbf{p}}) \in \mathbf{c}oCCS_L^{proc}$, $op = |, +, \_\backslash, \_[R], \tau.\_$,

3. $\mathbf{o} \in \mathbf{c}oCCS_L^{pair}$, $c \in C$ implies $c!.\mathbf{o} \in \mathbf{c}oCCS_L^{proc}$,

4. $\mathbf{f} \in \mathbf{c}oCCS_L^{fun}$ and $c \in C$ implies $c?.\mathbf{f} \in \mathbf{c}oCCS_L^{proc}$,

5. $\mathbf{p} \in \mathbf{c}oCCS_L^{proc}$ and $e \in Exp$ implies $(e, \mathbf{p}) \in \mathbf{c}oCCS_L^{pair}$,

6. $\mathbf{p}_1, \ldots, \mathbf{p}_n \in \mathbf{c}oCCS_L^{proc}$, $v_1, \ldots, v_n \in Val$ and $x \in Var$ implies $[x].\, x : (v_1 \ldots, v_n) \longrightarrow (\mathbf{p}_1 \ldots, \mathbf{p}_n) \in \mathbf{c}oCCS_L^{fun}$.

We let $\mathbf{c}oCCS_L$, $\mathbf{c}oCCS_L^{proc}$, $\mathbf{c}oCCS_L^{fun}$ and $\mathbf{c}oCCS_L^{pair}$ be ranged over by $\mathbf{c}t$, $\mathbf{c}p$, $\mathbf{c}f$ and $\mathbf{c}o$ respectively (Note that by definition these terms are value closed.). We say that a term is *compact* if it belongs to $\mathbf{c}oCCS_L$. □

Note that $\mathbf{c}oCCS_L \subseteq_{rec} \mathbf{C}CS_L$. We have the following:

Theorem 4.16

1. *There are unique evaluation mappings* $\overline{ACT}[\![\_]\!] : \mathbf{C}CS_L \longrightarrow \overline{ACT}$ *and* $\overline{K}[\![\_]\!] : \mathbf{c}oCCS_L \longrightarrow \overline{K}$.

2. $\overline{ACT}[\![\_]\!]\,|_{\mathbf{c}oCCS_L} = [\_]_\approx \circ \overline{K}[\![\_]\!]$ *where*
   - $f|_A$ *means the restriction of the function* f *to the set* A,
   - $[\_]_\approx$ *is the quotient mapping with respect to the preorder* $\prec$.

3. *For any* $\mathbf{c}t \in \mathbf{c}oCCS_L$, $\overline{ACT}[\![\mathbf{c}t]\!] \in Comp(\overline{ACT})$.

4. *For all* $\mathbf{c}t_1, \mathbf{c}t_2 \in \mathbf{c}oCCS_L$, $\overline{ACT}[\![\mathbf{c}t_1]\!] \sqsubseteq \overline{ACT}[\![\mathbf{c}t_2]\!]$ *if and only if* $\overline{K}[\![\mathbf{c}t_1]\!] \prec_{\overline{K}} \overline{K}[\![\mathbf{c}t_2]\!]$.

5. *For any* $c \in Comp(\overline{ACT})$ *there is a* $\mathbf{c}t \in \mathbf{c}oCCS_L$ *such that* $\overline{ACT}[\![\mathbf{c}t]\!] = c$.

Proof

1. As $\overline{ACT}$ is a fully applicative (Σ, C)-cpo then, by Theorem 4.7,
$$\overline{ACT}[\![\_]\!] : \mathbf{C}CS_L \longrightarrow \overline{ACT}$$
is well-defined and unique. The existence and uniqueness of $\overline{K}[\![\_]\!]$ follows by a simple structural induction on $\mathbf{c}oCCS_L$.

2. As $\mathbf{c}oCCS_L$ is a recursively closed subset of $\mathbf{C}CS_L$, we get that $\overline{ACT}[\![\_]\!]|_{\mathbf{c}oCCS_L}$ is an evaluation mapping on $\mathbf{c}oCCS_L$. By construction of $\langle \overline{ACT}, \sqsubseteq_{\overline{ACT}}, \Sigma_{\overline{ACT}}, C_{\overline{ACT}} \rangle$, $[\_]_\approx : \overline{K} \longrightarrow \overline{ACT}$ is a (Σ, C)-po homomorphism. It then follows from Corollary 4.8 that $\overline{ACT}[\![\_]\!]\,|_{\mathbf{c}oCCS_L} = [\_]_\approx \circ \overline{K}[\![\_]\!]$.

3. and 4. follow immediately from 2. and the construction of $\overline{ACT}$.

5. We start by proving that for any $k \in \overline{K}$ there is a $\mathbf{c}t \in \mathbf{c}oCCS_L$ such that $\overline{K}[\![\mathbf{c}t]\!] = k$. First we prove the result for the set K which by definition equals K_{proc}. This may be proved by induction on the definition of K. Then we may easily extend the proof to K_{fun} and K_{pair}.

   Next assume that c is a compact element of $\overline{ACT}$. Then, by the construction of the model, $c = [k]_\approx$ for some $k \in \overline{K}$. From what we proved above we get that $\overline{K}[\![\mathbf{c}t]\!] = k$ for some $\mathbf{c}t \in \mathbf{c}oCCS_L$. From 2. we get

$$\overline{ACT}[\![\mathbf{c}t]\!] = [\overline{K}[\![\mathbf{c}t]\!]]_\approx = [k]_\approx = c.$$

   This completes the proof.

□

4.4 A Domain Equation for Applicative Communication Trees

In this section we will show an alternative but equivalent definition of the model $\overline{ACT}$; we will show how it may be obtained as a solution to a recursive domain equation. The section is mostly of historical interest and may safely be skipped by the reader.

The equation we put forward is basically the one of [MM79] where the modifications of Abramsky's, reported in [Abr91] and described in the introduction, are adopted. Thus we define an algebraic cpo as a solution to a domain equation using the Plotkin Power Domain with the empty set adjoined as an isolated element. Here the main difference is that we use a different representation for the Plotkin Power Domain to the one used in [Abr91]. The representation we use is the one due to Smyth, [Smy78] and will be described below. In the definition of the domain we use the following operations on cpos:

Cartesian product $\times$ ([Plo81, §2 and §6]): Let $\langle A, \sqsubseteq_A\rangle$ and $\langle A', \sqsubseteq_{A'}\rangle$ be two *po*s. We define the partial order $\sqsubseteq_{A\times A'}$ on $A \times A'$ by:

$$(a, a') \sqsubseteq_{A\times A'} (b, b') \text{ if } a \sqsubseteq_A b \text{ and } a' \sqsubseteq_{A'} b'.$$

This construction extends to any number of *po*s. It preserves completeness and algebraicity. Countable products preserve ω-algebraicity. If A and A' are algebraic *cpo*s, the set of compact elements can be obtained from the compact elements of A and A' by $Comp(A \times A') = Comp(A) \times Comp(A')$.

Separated Sum $\sum_{i\in I}$ ([Abr87, §3], [Plo81, §3 and §6],): Let I be a countable index set and $\{A_i\}_{i\in I}$ be a family of I–indexed *po*s. The separated sum $\langle \sum_{i\in I} A_i, \sqsubseteq_{\sum_{i\in I} A_i}\rangle$ is defined as follows:

$$\textstyle\sum_{i\in I} A_i = \{\bot\} \cup (\bigcup\{\{i\} \times A_i | i \in I\}),$$

$$x \sqsubseteq_{\sum_{i\in I} A_i} y \text{ if } x = \bot \text{ or if for some } i,\ x = \langle i, a\rangle\ ,\ y = \langle i, a'\rangle \text{ and } a \sqsubseteq_{A_i} a',$$

where we write $\langle i, a\rangle$ for the elements of the disjoint union and $\bot$ for the bottom element of the separated sum. The construction preserves completeness, algebraicity and ω-algebraicity. If each A_i is an algebraic *cpo*, the set of compact elements of $\langle \sum_{i\in I} A_i, \sqsubseteq_{\sum_{i\in I} A_i}\rangle$ is given by

$$Comp(\sum_{i\in I} A_i) = \{\bot\} \cup (\bigcup\{\{i\} \times Comp(A_i) | i \in I\}.$$

Function Space from a fixed set S, F_S ([Plo81, §3]): Let S be a fixed countable set. For a *po* $\langle A, \sqsubseteq_A\rangle$ we define $F_S(A) = S \longrightarrow A$, the set of all functions from S to A, with the pointwise ordering $\sqsubseteq_{F_S(A)}$ as follows:

$$f \sqsubseteq_{F_S(A)} g \text{ if } \forall s \in S. f(s) \sqsubseteq_A g(s).$$

This construction preserves completeness, algebraicity and ω-algebraicity. The compact elements of $F_S(A)$ can be obtained from those of A by $Comp(F_S(A)) = F_S^{fin}(Comp(A))$ where $F_S^{fin}(B) = \{f \in S \longrightarrow B | \{s \in S | f(s) \neq \bot\} \text{ is finite}\}$. Note that the constructions $\sum_{i\in I}$ and $F_S(A)$ may just as well be defined for non-countable sets I and S but then they do not preserve ω-algebraicity in general.

The Plotkin Power Domain ([Win85]): We give a construction of the *Plotkin Power Domain* [Plo76] due to Smyth [Smy78] and described in [Win85]. Let $\langle A, \sqsubseteq_A\rangle$ be an ω-algebraic *cpo* and $M[A]$ the family of finite, non-empty sets of compact elements of A. The *Egli-Milner order* on $M[A]$ is defined by:

$$\text{For } X, Y \in M[A],\ X \sqsubseteq_{EM} Y \text{ iff } \begin{array}{l} \forall x \in X \exists y \in Y.\, x \sqsubseteq_A y \text{ and} \\[1em] \forall y \in Y \exists x \in X.\, x \sqsubseteq_A y. \end{array}$$

The Plotkin Power Domain of $\langle A, \sqsubseteq_A\rangle$, $\langle P[A], \sqsubseteq_{P[A]}\rangle$ is the ideal completion of the preorder $\langle M[A], \sqsubseteq_{EM}\rangle$. As explained before, we know that $\langle P[A], \sqsubseteq_{P[A]}\rangle$ is an ω-algebraic *cpo* and $Comp(P[A]) = M[A]/ =_{EM}$ (up to isomorphism).

In the definition to follow we shall use the empty set to interpret the process *nil* and the least element of the domain to interpret the process Ω. As *nil* is an isolated element with respect to bisimulation preorder, i.e. only related to itself and Ω, we adopt Abramsky's modification of the Plotkin Power Domain, i.e. we add the empty set to the domain in such a way that it is only related to itself and the least element of the domain in the obvious way under the extended Egli-Milner order. This may be described as follows:

Given an ω-algebraic *cpo* we write $P^0[D]$ for the Plotkin Power Domain over D with the empty set adjoined as an isolated element in the preorder. More precisely the elements of $P^0[D]$ are given by $P[D] \cup \{\emptyset\}$ with the order:

$$X \sqsubseteq_{P^0[D]} Y \quad \text{if} \quad \begin{array}{rl} & X, Y \in P[D] \text{ and } X \sqsubseteq_{P[D]} Y \\ \text{or} & Y = \{\emptyset\} \text{ and } (X = \{\emptyset\} \text{ or } X = \bot). \end{array} \tag{1}$$

All the constructions on pos described above may be turned into covariant continuous functors in the category $\mathbf{CPO}^E$, the category of cpos with embeddings, in a straightforward way. For the details we refer to [Plo81]. Now the standard theory in [Plo81] ensures that the following definition is meaningful.

Definition 4.17 Let C (the set of channels) and Val (the set of values) be countable sets and let $Act = \{c?|c \in C\} \cup \{c!|c \in C\} \cup \{\tau\}$ (the set of actions). We define the applicative *cpo* of applicative communication trees, $\langle \overline{\mathcal{A}}, \sqsubseteq_{\overline{\mathcal{A}}} \rangle$, as follows: $\langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$ is the initial solution in $\mathbf{CPO}^E$ of the recursive domain equation:

$$D = P^0[\sum_{e \in Act} D_e]$$

where

- $D_{c?} = F_{Val}(D) = Val \to D$ (as defined on page 22),
- $D_{c!} = Val \times D$ and
- $D_\tau = D$.

Then we define $\langle \overline{\mathcal{A}}, \sqsubseteq_{\overline{\mathcal{A}}} \rangle$ as the unique fully applicative ω-algebraic *cpo* induced by $\langle \mathcal{A}, \sqsubseteq_{\mathcal{A}} \rangle$. For the sake of simplicity we refer to this domain as $\mathcal{A}$ or $\langle \mathcal{A}, \sqsubseteq \rangle$. □

From the general theory in [Plo81] we get a representation of the compact elements by unfolding the recursive definition of $\mathcal{A}$. Thus we define $COMP = \bigcup_{n=0}^{\infty} COMP^n$ where $COMP^0 = \{\bot\}$ and

$$COMP^{n+1} = M[\sum_{e \in Act} (COMP^n)_e] \cup \{\emptyset\}$$

where $(COMP^n)_{c?} = F_{Val}(COMP^n)$, $(COMP^n)_{c!} = Val \times COMP^n$ and $(COMP^n)_\tau = COMP^n$. We recall that for an algebraic *cpo* A, $M[A]$ is defined as the family of non-empty sets of compact elements of A. The empty set is added to the family $COMP^n$ as we are using the power domain operator P^0 rather than P. Defining $COMP$ in this way and ordering it by $\sqsubseteq^0_{EM}$, the Egli-Milner preorder over $COMP$ extended like in (1) above, gives a representation of the compact elements of the ω-algebraic cpo $\mathcal{A}$. This means that the kernel of the preorder is isomorphic to the partial order of compact elements of the domain $\mathcal{A}$, $\langle Comp(\mathcal{A}), \sqsubseteq_{Comp(\mathcal{A})} \rangle$. (For the sake of simplicity we assume that the kernel is *equal* to $Comp(\mathcal{A})$.) Now we prove that $\langle K, \prec \rangle$ defined above is equal to $\langle COMP, \sqsubseteq^0_{EM} \rangle$.

Proposition 4.18 $\langle K, \prec \rangle = \langle COMP, \sqsubseteq^0_{EM} \rangle$.

Proof First we prove that $K = COMP$. That $K \subseteq COMP$ can be proved by showing that $COMP$ is closed under $1. - 6.$ in the definition of K and then use the fact that K is the least set with this property. To prove the opposite inclusion, it is sufficient to show that, for every n, $COMP^n \subseteq K$.

Then we prove that the preorder $\prec$ coincides with the extended Egli-Milner preorder on K. To prove $\prec \subseteq \sqsubseteq^0_{EM}$, as before it is sufficient to prove that $\sqsubseteq^0_{EM}$ satisfies the definition of $\prec$. The details are straightforward and are left to the reader. To prove $\prec \supseteq \sqsubseteq^0_{EM}$ we first define the depth of the elements of K as follows:

1. $d(\emptyset) = d(\{\bot\}) = 0$,
2. $d(\{\langle \mu, k \rangle\}) = 1 + d(k)$,
3. $d(\{a_1, \ldots, a_n\}) = max\{d(\{a_i\})|i \leq n\}$,

4. $d(f) = max\{d(f(v))|v \in Val\}$ (recall that $\{f(v)|v \in Val\}$ is a finite set as f yields $\perp$ on all but finitely many values in Val),

5. $d(e, k) = d(k)$.

Then we prove by induction on $d(k)$ that $k \sqsubseteq^0_{EM} k' \Rightarrow k \prec k'$. The details of the proofs are straightforward and are left to the reader. $\Box$

Proposition 4.18 has the following direct corollary.

Corollary 4.19 *The algebraic cpos ACT and $\mathcal{A}$ are isomorphic.*

5 Algebraic Laws and Proof Systems

In this section we will introduce proof systems supported by the model $\overline{ACT}$. We proceed by introducing first a system E to reason about finite processes that reflects the structure of K. Then we extend this system to systems which take care of recursive processes in two different ways and reason about soundness and completeness of both these extensions with respect to the model.

The proof system E is equationally based where the equations reflect naturally the properties of the operators in the model. As an example the equations

$$\begin{array}{rcl} X + (Y + Z) & = & (X + Y) + Z \\ X + Y & = & Y + X \\ X + X & = & X \end{array}$$

reflect the fact that the elements of K, and thus of $\overline{ACT}$, are defined as sets and $+$ as set union. The inference rules describe the structure and the preorder of the model and their interaction with the operators. Because of the two level structure of our syntax, we have the equations

$$(res\,in) \qquad (a?.[x]X) \setminus c = \begin{cases} a?.[x](X \setminus c) & \text{if } c \neq a \\ nil & \text{otherwise,} \end{cases}$$

$$(res\,out) \qquad (a!.(e, X)) \setminus c = \begin{cases} a!.(e, X \setminus c) & \text{if } c \neq a \\ nil & \text{otherwise,} \end{cases}$$

$$(ren\,in) \qquad (a?.[x]X)[R] = R(a)?.[x](X[R]),$$

$$(ren\,out) \qquad (a!.(e, X))[R] = R(a)!.(e, X[R])$$

and the rules

$$(fun) \qquad \frac{p[v/x] \sqsubseteq q[v/x] \text{ for every } v \in V}{[x]p \sqsubseteq [x]q}$$

$$(pair) \qquad \frac{[\![e_1]\!] = [\![e_2]\!], p \sqsubseteq q}{(e_1, p) \sqsubseteq (e_2, q)}$$

$$
\begin{array}{lrcl}
(+1) & X + (Y + Z) & = & (X + Y) + Z \\
(+2) & X + Y & = & Y + X \\
(+3) & X + X & = & X \\
(+4) & X + nil & = & X \\
(res+) & (X + Y) \setminus c & = & X \setminus c + Y \setminus c \\
(res\tau) & (\tau.X) \setminus c & = & \tau.(X \setminus c) \\
(res\,in) & (a?.[x]X) \setminus c & = & \begin{cases} a?.[x](X \setminus c) & \text{if } c \neq a \\ nil & \text{otherwise} \end{cases} \\
(res\,out) & (a!.(e, X)) \setminus c & = & \begin{cases} a!.(e, X \setminus c) & \text{if } c \neq a \\ nil & \text{otherwise} \end{cases} \\
(res\,cond) & (be \longrightarrow X, Y) \setminus c & = & be \longrightarrow X \setminus c, Y \setminus c \\
(res\,nil) & nil \setminus c & = & nil \\
(res\,div) & \Omega \setminus c & = & \Omega \\
(ren+) & (X + Y)[R] & = & X[R] + Y[R] \\
(ren\tau) & (\tau.X)[R] & = & \tau.(X[R]) \\
(ren\,in) & (a?.[x]X)[R] & = & R(a)?.[x](X[R]) \\
(ren\,out) & (a!.(e, X))[R] & = & R(a)!.(e, X[R]) \\
(ren\,cond) & (be \longrightarrow X, Y)[R]c & = & be[R] \longrightarrow X[R], Y[R] \\
(ren\,nil) & nil[R] & = & nil \\
(ren\,div) & \Omega[R] & = & \Omega \\
(nil\,par) & nil \mid X & = & X \mid nil \; = \; X
\end{array}
$$

Figure 3: Equations

that allow us to prove inequalities over function terms and pairs. The first extension of E is obtained by adding to E two new rules to take care of recursion. The new rules introduced are

$$(rec) \quad recP.p = p[recP.p/P]$$

and

$$(\omega - rule) \quad \frac{p^{(n)} \sqsubseteq q \text{ for all } n}{p \sqsubseteq q}$$

where $\mathbf{t}^{(n)}$, the syntactically compact approximations of a term $\mathbf{t}$, are defined in Definition 5.1. We write $\mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}$ if $\mathbf{t} \sqsubseteq \mathbf{u}$ can be proven using the rules given above and $\mathbf{t} \sqsubseteq_{E^{-\omega}_{rec}} \mathbf{u}$ if this inequality can be proven in the same system without applying the ω-rule.

Note here that the approximations that occur in the ω-rule are syntactically compact as the number of values in the approximations is finite just as the depth of the approximation is finite. Consequently we refer to this system as the value-finitary one. This correspondence with the compact elements enables us to take advantage of the algebraicity of the model when proving the completeness of the proof system. In the interleaving law the summation notation is justified by equations (+1)-(+4) and an empty sum is understood as nil. $\{+\Omega\}$ indicates that Ω is an optional summand of a term and Ω is a summand of the right hand side if it is a summand of X or Y on the left hand side. To simplify the notation we assume that i, j etc. in the sums $\sum_i$, $\sum_j$, etc. range over finite index sets I, J, etc.

Let $X = \sum_i \tau.X_i + \sum_j a'_j?.[x]X'_j + \sum_k a''_k!.(v_k, X''_k)\{+\Omega\}$ and $Y = \sum_l \tau.Y_l + \sum_m b'_m?.[y]Y'_m + \sum_n b''_n!.(v_n, Y''_n)\{+\Omega\}$. Then

$$X \mid Y = INTL(X,Y) + COMM(X,Y)\{+\Omega\}$$

where

$$INTL(X,Y) = INTL_\tau(X,Y) + INTL_{in}(X,Y) + INTL_{out}(X,Y)$$

where

$$\begin{array}{l} INTL_\tau(X,Y) = \sum_i \tau.(X_i|Y) + \sum_l \tau.(X|Y_l) \\ INTL_{in}(X,Y) = \sum_j a'_j?.[x](X'_j|Y) + \sum_m b'_m?.[y](X|Y'_m) \\ INTL_{out}(X,Y) = \sum_k a''_k!.(v_k, X''_k|Y) + \sum_n b''_n!.(v'_n, X|Y''_n) \end{array}$$

and

$$COMM(X,Y) = \sum_{j,n:a'_j=b''_n} \tau.X'_j[v_n/x]|Y''_n + \sum_{k,m:a''_k=b'_m} \tau.X''_k|Y'_m[v_k/y]$$

Figure 4: Interleaving Law

(ref) $\quad p \sqsubseteq p$

$(trans)$ $\quad \dfrac{p \sqsubseteq q,\ q \sqsubseteq r}{p \sqsubseteq r}$

$(least)$ $\quad \Omega \sqsubseteq p$

(sub) $\quad \dfrac{p_i \sqsubseteq q_i}{op(\underline{p}) \sqsubseteq op(\underline{q})}\ op \in \sum$

(pre) $\quad \dfrac{p \sqsubseteq q}{\mu.p \sqsubseteq \mu.q}$

$(cond1)$ $\quad \dfrac{[\![be]\!] = T}{be \longrightarrow p, q = p}$

$(cond2)$ $\quad \dfrac{[\![be]\!] = F}{be \longrightarrow p, q = q}$

(rec) $\quad \dfrac{}{recP.p = p[recP.p/P]}$

$(inst)$ $\quad \dfrac{}{p\sigma \sqsubseteq q\sigma}$ for every inequation $p \sqsubseteq q$ and closed instantiation σ

$(\omega - rule)$ $\quad \dfrac{p^{(n)} \sqsubseteq q \text{ for all } n}{p \sqsubseteq q}$

$(pair)$ $\quad \dfrac{[\![e]\!] = [\![e']\!], p \sqsubseteq q}{(e,p) \sqsubseteq (e',q)}$

(fun) $\quad \dfrac{p[v/x] \sqsubseteq q[v/x] \text{ for every } v \in V}{[x]p \sqsubseteq [x]q}$

$(\alpha - red)$ $\quad \dfrac{}{[x]p = [y]p[y/x]}$ if y not free in p

Figure 5: The Proof System E_{rec}

Now we define the syntactically compact (or just compact for short) approximations used in the ω-rule of the proof system.

Definition 5.1 [Compact Approximations] The n-th compact approximation of a term is defined inductively by :

1. (a) $p^{(0)} = \Omega$
   (b) i. $P^{(n+1)} = P$,
      ii. $(op(\underline{p}))^{(n+1)} = op(\underline{p}^{(n+1)})$,
      iii. $(\mu.u)^{(n+1)} = \mu.u^{(n+1)}$,
      iv. $(recP.p)^{(n+1)} = p^{(n+1)}[(recP.p)^{(n)}/P]$,
      v. $(be \longrightarrow p, q)^{(n+1)} = be \longrightarrow p^{(n+1)}, q^{(n+1)}$,
2. $([x]p)^{(n+1)} = [x](x \in V_{n+1} \longrightarrow p^{(n+1)}, \Omega)$,
3. $((e,p))^{(n+1)} = \begin{cases} ([\![e]\!], p^{(n+1)}) & \text{if } [\![e]\!] \in V_{n+1}, \\ ([\![e]\!], \Omega) & \text{otherwise.} \end{cases}$

$\square$

We remind the reader that $V_n = \{v_1, \ldots, v_n\}$ is the set of the n first values. The compact approximations have the following fundamental properties:

Theorem 5.2 *For all natural numbers n and all $\mathbf{t} \in \mathbf{C}CS_L$*

1. *$\mathbf{t}^{(n)} \in \mathbf{co}CCS_L$, i.e. $\mathbf{t}^{(n)}$ is a compact term.*
2. *$\mathbf{t}^{(n)} \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$.*
3. *$\overline{ACT}[\![\mathbf{t}]\!]\rho = \bigsqcup_n \overline{ACT}[\![\mathbf{t}^{(n)}]\!]\rho$ for all ρ.*

Proof

1. and 2. may be proved by an induction on n combined with an inner structural induction.

3. In what remains of the proof we write $[\![\_]\!]$ instead of $\overline{ACT}[\![\_]\!]$. We have to prove that $\bigsqcup[\![\mathbf{t}^{(n)}]\!]\rho \sqsubseteq [\![\mathbf{t}]\!]\rho$ and $[\![\mathbf{t}]\!]\rho \sqsubseteq \bigsqcup[\![\mathbf{t}^{(n)}]\!]\rho$. To prove the first inequality it is sufficient to prove that $[\![\mathbf{t}^{(n)}]\!]\rho \sqsubseteq [\![\mathbf{t}]\!]\rho$ for all n. This may be proved in the same way as a similar property for the pure calculus given in Lemma 4.2.10 in [Hen88a]. The proof for the opposite inequality, $[\![\mathbf{t}]\!]\rho \sqsubseteq \bigsqcup[\![\mathbf{t}^{(n)}]\!]\rho$, again follows the same pattern as the proof for a similar property given in Theorem 4.2.11 in [Hen88a]. The main difference is when $\mathbf{t} = [x]p \in \mathbf{C}CS_L^{fun}$. For this case we may proceed as follows.

   It is easy to see that

   $$[\![[x]p^{(0)}]\!]\rho \sqsubseteq \cdots [\![[x]p^{(n)}]\!]\rho \sqsubseteq \cdots \sqsubseteq [\![[x]p]\!]\rho.$$

   i.e. that $[\![[x]p]\!]\rho$ is an upper bound of the chain given above. We have to show that it is the least upper bound of the chain. So assume

   $$[\![[x]p^{(0)}]\!]\rho \sqsubseteq \cdots [\![[x]p^{(n)}]\!]\rho \sqsubseteq \cdots \sqsubseteq \mathbf{f}.$$

We have to show that $[\![[x]p]\!]\rho \sqsubseteq \mathbf{f}$. So assume $v \in Val$. Then $v \in V_N$ for some N. Therefore, for all $n \geq N$,

$$([\![([x]p)^{(n)}]\!]\rho)(v) = ([\![[x]x \in V_n \longrightarrow p^{(n)}, \Omega]\!]\rho)(v) =$$

$$[\![p^{(n)}[v/x]]\!]\rho \sqsubseteq \mathbf{f}(v).$$

By the structural induction, $[\![p[v/x]]\!]\rho$ is the least upper bound for the chain

$$[\![p^{(0)}[v/x]]\!]\rho \sqsubseteq [\![p^{(1)}[v/x]]\!]\rho \sqsubseteq \cdots \sqsubseteq [\![p^{(n)}[v/x]]\!]\rho \sqsubseteq \cdots.$$

This implies that $([\![[x]p]\!]\rho)(v) \sqsubseteq \mathbf{f}(v)$. As $v \in Val$ was arbitrary this implies that $[\![[x]p]\!]\rho \sqsubseteq \mathbf{f}$ as wanted.

□

Next we prove the soundness and completeness of the proof system E_{rec} with respect to the model. To prove the completeness we introduce a notion of Ω-*normal forms* for compact terms and a corresponding normalization theorem.

Definition 5.3 [**Ω-normal form**] A compact term $\mathbf{c}t \in \mathbf{c}oCCS_L$ is said to be in Ω-normal form if the following hold:

1. If $\mathbf{c}t = \mathbf{c}p \in \mathbf{c}oCCS_L^{proc}$ then $\mathbf{c}p$ has the form

   $$\sum_i a_i.\mathbf{c}t_i\{+\Omega\}$$

   where Ω is an optional summand and where $\mathbf{c}t_i$ is in Ω-normal form. The empty sum is interpreted as *nil*.

2. If $\mathbf{c}t = (e, \mathbf{c}p) \in \mathbf{c}oCCS_L^{pair}$ then $e = v \in Val$ and $\mathbf{c}p$ is in Ω-normal form.

3. If $\mathbf{c}t = [x]x : (v_1, \ldots, v_n) \longrightarrow (\mathbf{c}p_1, \ldots \mathbf{c}p_n) \in Fun$ then $\mathbf{c}p_i$ is in Ω-normal form for $i \leq n$.

□

Lemma 5.4 *For all* $\mathbf{c}t \in \mathbf{c}oCCS_L$ *there is an* Ω*-normal form* $n(\mathbf{c}t)$ *such that* $n(\mathbf{c}t) =_E \mathbf{c}t$.

Proof First we define the depth, $\delta(\mathbf{c}t)$ of a compact term $\mathbf{c}t$ by

1. $\delta(nil) = \delta(\Omega) = 0$,
2. $\delta(\mathbf{c}p \setminus c) = \delta(\mathbf{c}p[R]) = \delta(\mathbf{c}p)$,
3. $\delta(\mathbf{c}p_1 + \mathbf{c}p_2) = max\{\delta(\mathbf{c}p_i) | i \leq n\}$,
4. $\delta(\mathbf{c}p_1 | \mathbf{c}p_2) = 1 + \delta(\mathbf{c}p_1) + \delta(\mathbf{c}p_2)$,
5. $\delta(pre.\mathbf{c}t) = 1 + \delta(\mathbf{c}t)$,

6. $\delta((e, \mathbf{c}p)) = \delta(\mathbf{c}p)$,
7. $\delta([x].x : (v_1, \cdots, v_n) \longrightarrow (\mathbf{c}p_1, \cdots, \mathbf{c}p_n) = max\{\delta(\mathbf{c}p_i) | i \leq n\}$.

To prove the result we prove the following stronger result:

> For all $\mathbf{c}t \in \mathbf{c}oCCS_L$ there is a Ω-normal form $n(\mathbf{c}t)$ such that $n(\mathbf{c}t) =_E \mathbf{c}t$ and $\delta(n(\mathbf{c}t)) \leq \delta(\mathbf{c}t)$.

To prove this we proceed by induction on $\delta(\mathbf{c}t)$. The details are left to the reader. □

Note that the notion of Ω-normal forms and the normalization lemma extend to syntactically finite terms in a natural way. This will be useful later in this study.

Now we will prove the soundness and completeness of the proof system E_{rec} with respect to the denotational semantics. We start with the following lemma.

Lemma 5.5 *For all $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$, $\mathbf{t} \sqsubseteq_E \mathbf{u}$ implies $\exists m \forall n \geq m. \mathbf{t}^{(n)} \sqsubseteq_E \mathbf{u}^{(n)}$.*

Proof This may be proved by induction on the depth of the proof for $\mathbf{t} \sqsubseteq_E \mathbf{u}$. The only non-trivial case is the base case when the interleaving law is used. We leave it to the reader to check the details of the proof. □

Theorem 5.6 (Soundness and completeness of the value-finitary proof system) *For all $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$ we have*

$$\mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u} \text{ if and only if } ACT[\![\mathbf{t}]\!] \sqsubseteq ACT[\![\mathbf{u}]\!]$$

i.e. the proof system E_{rec} is sound and complete with respect to the denotational semantics.

Proof

Soundness: The soundness of the ω-rule is the content of Theorem 5.2 whereas the soundness of the (rec)-rule follows from the definition of the semantics of $recP.p$ as a least fixed point. What remains to prove is the soundness of E. This, in turn, can be proven by reducing the proof to a proof of the soundness for syntactically compact terms with respect to $\overline{K}$.

The soundness of E over $\mathbf{c}oCCS_L$ with respect to $\overline{K}$ follows easily from the definition of K and the fact that the elements of $\mathbf{c}oCCS_L$ denote exactly the elements of $\overline{K}$. Now we may proceed as follows:

Assume $\mathbf{t} \sqsubseteq_E \mathbf{u}$. Then, by Lemma 5.5, for some m, $\mathbf{t}^{(n)} \sqsubseteq_E \mathbf{u}^{(n)}$ for all $n \geq m$. As $\mathbf{t}^{(n)}, \mathbf{u}^{(n)} \in \mathbf{c}oCCS_L$, the soundness of E with respect to $\overline{K}$ for this set implies

$$\overline{K}[\![\mathbf{t}^{(n)}]\!] \prec_{\overline{K}} \overline{K}[\![\mathbf{u}^{(n)}]\!] \text{ for all } n \geq m$$

or equivalently

$$\overline{ACT}[\![\mathbf{t}^{(n)}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}^{(n)}]\!] \text{ for all } n \geq m.$$

Theorem 5.2.3 implies

$$ACT[\![\mathbf{t}]\!] \sqsubseteq ACT[\![\mathbf{u}]\!].$$

Completeness: Again we reduce the proof to proving that E is complete for $\mathbf{co}CCS_L$ with respect to $\overline{K}$. We first note that Theorem 5.2 and the ω-algebraicity of the model imply

$$\begin{array}{ll} \overline{ACT}[\![\mathbf{t}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}]\!] & \Rightarrow \\ \forall n.\overline{ACT}[\![\mathbf{t}^{(n)}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}]\!] & \Rightarrow \\ \forall n \exists m.\overline{ACT}[\![\mathbf{t}^{(n)}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}^{(m)}]\!] & \Rightarrow \\ \forall n \exists m.\overline{K}[\![\mathbf{t}^{(n)}]\!] \prec \overline{K}[\![\mathbf{u}^{(m)}]\!]. & \end{array} \quad (2)$$

If E is complete for $\mathbf{co}CCS_L$ with respect to $\overline{K}$ then

$$\overline{K}[\![\mathbf{t}^{(n)}]\!] \prec \overline{K}[\![\mathbf{u}^{(m)}]\!] \Rightarrow \mathbf{t}^{(n)} \sqsubseteq_E \mathbf{u}^{(m)}. \quad (3)$$

Now $\mathbf{u}^{(m)} \sqsubseteq_{E_{rec}} \mathbf{u}$ by Lemma 5.2.2 so (2), (3) and the ω-rule give

$$\overline{ACT}[\![\mathbf{t}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}]\!] \Rightarrow \forall n.\mathbf{t}^{(n)} \sqsubseteq_{E_{rec}} \mathbf{u} \Rightarrow \mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}.$$

Thus what remains to prove is the completeness of E over $\mathbf{co}CCS_L$ with respect to $\overline{K}$. By Lemma 5.4 and the soundness of E it is even enough to prove the completeness for Ω-normal forms with respect to $\overline{K}$ because:

> Assume $\overline{K}[\![\mathbf{c}t_1]\!] \prec \overline{K}[\![\mathbf{c}t_2]\!]$. By Lemma 5.4 $\mathbf{c}t_i =_E \mathbf{n}_i, i = 1, 2$ where $\mathbf{n}_i, i = 1, 2$ are Ω-normal forms. By the soundness of E with respect to $\overline{K}$, $\overline{K}[\![\mathbf{n}_i]\!] = \overline{K}[\![\mathbf{c}t_i]\!], i = 1, 2$ and therefore $\overline{K}[\![\mathbf{n}_1]\!] \prec \overline{K}[\![\mathbf{n}_2]\!]$. If E is complete for Ω-normal forms with respect to $\overline{K}$ we may conclude that $\mathbf{n}_1 \sqsubseteq_E \mathbf{n}_2$. That $\mathbf{c}t_1 \sqsubseteq_E \mathbf{c}t_2$ now follows from the transitivity of the proof system.

To prove the completeness for the Ω-normal forms we proceed as follows:

> Assume $\mathbf{n}_1, \mathbf{n}_2$ are Ω-normal forms. We have to prove that
>
> $$\overline{K}[\![\mathbf{n}_1]\!] \prec \overline{K}[\![\mathbf{n}_2]\!] \Rightarrow \mathbf{n}_1 \sqsubseteq_E \mathbf{n}_2.$$
>
> We proceed by structural induction on $\mathbf{n}_1$.
>
> $\mathbf{n}_1 = nil + \Omega$: Obvious.
>
> $\mathbf{n}_1 = nil$: $\emptyset = \overline{K}[\![nil]\!] \prec \overline{K}[\![\mathbf{n}_2]\!]$ implies $\overline{K}[\![\mathbf{n}_2]\!] = \emptyset$ and therefore that $\mathbf{n}_2 = nil$.
>
> $\mathbf{n}_1 = \sum_{i \leq n} \mu_i.\mathbf{t}_i\{+\Omega\}$, $n \geq 1$: Then
>
> $$\overline{K}[\![\mathbf{n}_1]\!] = \{\langle \mu_i, \overline{K}[\![\mathbf{t}_i]\!]\rangle | i \leq n\}[\cup\{\perp\}]$$
>
> where $\perp \in \overline{K}[\![\mathbf{n}_1]\!]$ if and only if Ω is a summand of $\mathbf{n}_1$. As $\overline{K}[\![\mathbf{n}_1]\!] \prec \overline{K}[\![\mathbf{n}_2]\!]$ then $\mathbf{n}_2 \neq nil$ and $\mathbf{n}_2 \neq \Omega$, i.e. $\mathbf{n}_2$ has the form
>
> $$\mathbf{n}_2 = \sum_{j \leq m} \gamma_j.\mathbf{u}_j\{+\Omega\}$$
>
> and
>
> $$\overline{K}[\![\mathbf{n}_2]\!] = \{\langle \gamma_j, \overline{K}[\![\mathbf{u}_j]\!]\rangle | j \leq m\}[\cup\{\perp\}]$$

where $m \geq 1$. Assume that $\langle \mu_i, \overline{K}[\![\mathbf{t}_i]\!]\rangle \in \overline{K}[\![\mathbf{n}_1]\!]$. Then $\mu_i = \gamma_{j_i}$ and $\overline{K}[\![\mathbf{t}_i]\!] \prec \overline{K}[\![\mathbf{u}_{j_i}]\!]$ for some $j_i \leq m$. By induction $\mathbf{t}_i \sqsubseteq_E \mathbf{u}_{j_i}$. As this holds for any i we get that

$$\sum_{i \leq n} \mu_i.\mathbf{t}_i \sqsubseteq_E \sum_{i \leq n} \gamma_{j_i}.\mathbf{u}_{j_i} \tag{4}$$

First assume that Ω is a summand in $\mathbf{n}_1$. As obviously

$$\Omega \sqsubseteq_E \sum_j \gamma_j.\mathbf{u}_j\{+\Omega\}$$

we get, by (4), substitutivity and absorption of the proof system, that

$$\mathbf{n}_1 = \sum_i \mu_i.\mathbf{t}_i + \Omega \sqsubseteq_E \sum_i \gamma_{j_i}.\mathbf{u}_{j_i} + \sum_j \gamma_j.\mathbf{u}_j\{+\Omega\} =_E \mathbf{n}_2$$

which proves the statement in this case.
Next assume that Ω is not a summand in $\mathbf{n}_1$. This implies that $\perp \notin \overline{K}[\![\mathbf{n}_1]\!]$ which in turn implies that $\perp \notin \overline{K}[\![\mathbf{n}_2]\!]$. We may therefore conclude that Ω is not a summand of $\mathbf{n}_2$ either. In a similar way as before we get

$$\sum_{j \leq m} \mu_{i_j}.\mathbf{t}_{i_j} \sqsubseteq_E \sum_{j \leq m} \gamma_j.\mathbf{u}_j \tag{5}$$

where $\{i_j | j \leq m\} \subseteq \{1, \ldots n\}$. Now from (4), (5), the absorption and the substitutivity of the proof system we get

$$\mathbf{n}_1 = \textstyle\sum_{i \leq n} \mu_i.\mathbf{t}_i =_E \sum_{i \leq n} \mu_i.\mathbf{t}_i + \sum_{j \leq m} \mu_{i_j}.\mathbf{t}_{i_i}$$

$$\sqsubseteq_E \textstyle\sum_{i \leq n} \gamma_{j_i}.\mathbf{u}_{j_i} + \sum_{j \leq m} \gamma_j.\mathbf{u}_j = \sum_{j \leq m} \gamma_j.\mathbf{u}_j = \mathbf{n}_2$$

which completes the proof for this case.

$\mathbf{n}_1 \in \mathbf{c}oCCS_L^{fun}, \mathbf{c}oCCS_L^{pair}$: Follows easily from the induction.

□

As a consequence of the soundness and completeness theorem above together with Theorem 5.2.3 we get the following useful corollary.

Corollary 5.7 *For all* $\mathbf{c}t \in \mathbf{c}oCCS_L$ *and* $\mathbf{u} \in \mathbf{C}CS_L$

$$\mathbf{c}t \sqsubseteq_{E_{rec}} \mathbf{u} \text{ implies } \mathbf{c}t \sqsubseteq_E \mathbf{u}^{(n)} \text{ for some } n$$

and therefore

$$\mathbf{c}t \sqsubseteq_{E_{rec}} \mathbf{u} \text{ iff } \mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{u}.$$

The proof system we have introduced so far is nonstandard in that the ω-rule is based on compact approximations instead of the more common syntactically finite ones. Let $\mathbf{t}^n$ denote the *syntactically finite approximation* derived from Definition 5.1 with 2. replaced by $([x]p)^{n+1} = [x]p^{n+1}$ and 3. by $((e, \mathbf{p}))^{n+1} = ([\![e]\!], \mathbf{p}^{n+1})$ and let $\leq_{E_{rec}}$ denote the corresponding preorder. We refer to this system as the value-infinitary one. It is easy to see that following holds.

Lemma 5.8 *For all* $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$,

1. *if* $\mathbf{t} \leq_{E_{rec}} \mathbf{u}$ *then* $\mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}$,
2. *if* $\mathbf{d}$ *is syntactically finite then* $\mathbf{d} \leq_{E_{rec}} \mathbf{t}$ *iff* $\mathbf{d} \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$.

The first part of Lemma 5.8 tells us that the new system is sound with respect to the denotational model. To reason about the completeness of this system turns out to be more complicated. We will postpone the discussion to Section 6 where we show that the completeness of the system follows from results regarding the bisimulation based semantics.

6 Comparison with the Operational Semantics

The subject of this section is to compare the denotational semantics and the operational one given in the previous sections. First we show by an example, Example 6.1, that for the language CCS_L the bisimulation preorder, defined in Section 3, is too fine to coincide with the partial order in the model in the sense that the model is not fully abstract with respect to this behavioural preorder. This observation supports our intuition about that bisimulation is in general too fine to be completely characterized by any semantics induced by an algebraic *cpo* as explained in the introduction to this paper.

Example 6.1 *As we only need an example from the pure calculus, we use the notation* $\overline{a}.t = c?.[x]t$, $a.t = c!.(v_1, t)$ *and* $\_ \setminus a = \_ \setminus c$. *Let* $a^\omega = recY.a.Y$ *and* $\mathbf{p} = [recX.(a^\omega + X|\overline{a})] \setminus a$. *Then the first unfolding of* $\mathbf{p}$ *is* $\mathbf{p}_1 = [(a^\omega + (recX.(a^\omega + X)|\overline{a}))|\overline{a}] \setminus a$, *and the* $n+1$*-th one*

$$\mathbf{p}_{n+1} = [\overbrace{(a^\omega + ((a^\omega + ((a^\omega + \ldots + ((a^\omega}^{n} + recX.(a^\omega + X)|\overline{a}))|\underbrace{\overline{a} \ldots |\overline{a}))|\overline{a}}_{n}] \setminus a.$$

The reader may convince himself that the behaviour of $\mathbf{p}$ *can be given by the derivation tree described by the infinite sum* $\Omega + \sum_{i \in \omega} \tau^i.nil$, *i.e. a tree which has an infinite number of branches which all have a finite depth. Then, because of the algebraicity of the model,* $\overline{ACT}[\![\mathbf{p} + recP.\tau.P]\!] = \overline{ACT}[\![\mathbf{p}]\!]$. *On the other hand, the left hand side has the transition* $\mathbf{p} + recP.\tau.P \xrightarrow{\tau} recP.\tau.P$ *where* $recP.\tau.P$ *can perform an infinite sequence of* τ*-moves. This move can therefore never be matched, up to bisimulation, by the right hand side* $\mathbf{p}$. *This implies that* $\mathbf{p} + recP.\tau.P \not\precsim \mathbf{p}$.

Obviously Example 6.1 rules out the possibility that the behavioural preorder $\precsim$ characterizes the preorder of the model over CCS_L. Our second suggestion for a behavioural characterization of the model is the weaker version of $\precsim$, the *strong applicative* ω*-bisimulation preorder*, derived from the function $\mathcal{F}$ by iterated application. This is a straight forward extension of the ω-bisimulation for the pure calculus known from the literature [Mil83].

Definition 6.2 [Strong Applicative ω-Prebisimulation]
The kth sa-prebisimulation $\precsim_k$ is defined inductively by:

1. $\precsim_0 = Con \times Con$,
2. $\precsim_{n+1} = \mathcal{F}(\precsim_n)$.

The sa-ω-prebisimulation $\precsim_\omega$ is defined as $\precsim_\omega = \bigcap_k \precsim_k$ and $\sim_\omega = \precsim_\omega \cap \precsim_\omega^{-1}$. □

For all k we have that $\lesssim \subseteq \lesssim_{k+1} \subseteq \lesssim_k$ which implies that $\lesssim \subseteq \lesssim_\omega$.

Again this preorder is too fine to match the preorder from the model in general. This is explained by the following example.

Example 6.3 *If we extend our language with parameterized recursion we can define* $\mathbf{p} = P(1)$*, where* P *is be given by*

$$P(n) = c?x.[x](x \leq n \longrightarrow nil, \Omega) + P(n+1),$$

and $\mathbf{q} = \mathbf{p} + c?.[x]nil$*. In any denotational semantics based on an algebraic cpo* D*, it is clear that* $D[\![\mathbf{p}]\!] = D[\![\mathbf{q}]\!]$*. On the other hand* $\mathbf{q}$ *has the derivation* $\mathbf{q} \xrightarrow{c?} [x].nil$ *which can never be matched by* $\mathbf{p}$ *up to* $\lesssim_\omega$ *and consequently* $\mathbf{q} \not\lesssim_\omega \mathbf{p}$*.*

Whether $\mathbf{p}$ *and* $\mathbf{q}$ *can be expressed in our language* CCS_L *is not obvious at this point but later we will prove that they cannot.*

Intuitively the reason for why $\lesssim_\omega$ is too fine to match a preorder induced by an algebraic cpo for processes with values in general is that the values give rise to a new kind of infinity. We recall that in the model the preorder is decided completely by the compact elements. We also recall that the compact elements both have finite "depth" and "width", i.e. map all but finite number of values to $\perp$. These considerations motivate the following definition of *value-finitary strong applicative* ω*-prebisimulation*. This definition is a slight modification of the one given in [HP80].

Definition 6.4 [Value-Finitary Strong Applicative ω-Prebisimulation]
Let $AT = \langle Con, Val, Act, \longrightarrow, \downarrow \rangle$ be an applicative labelled transition system, $V \subseteq Val$ and $\mathcal{R} \subseteq Con \times Con$. Then we define the V-restricted extension of $\mathcal{R}$, $\mathcal{R}|_V$ by

1. $c_1 \mathcal{R}|_V c_2$ iff $c_1 \mathcal{R} c_2$
2. $(v_1, c_1)\mathcal{R}|_V(v_2, c_2)$ iff ($v_1 \in V$ or $v_2 \in V$) implies ($v_1 = v_2$ and $c_1 \mathcal{R} c_2$).
3. $f_1 \mathcal{R}|_V f_2$ iff $f_1(v)\mathcal{R} f_2(v)$ for all $v \in V$.

The nth value-finitary sa-bisimulation preorder $\lesssim^f_n$ is defined by:

1. $\lesssim^f_0 = Con \times Con$,
2. $\lesssim^f_{n+1} = (\mathcal{F}(\lesssim^f_n))|_{V_{n+1}}$.

The value-finitary sa-ω-bisimulation preorder $\lesssim^f_\omega$ is defined by $\lesssim^f_\omega = \bigcap_k \lesssim^f_k$ with the derived equivalence $\sim^f_\omega = \lesssim^f_\omega \cap (\lesssim^f_\omega)^{-1}$. □

From this definition we get that $(v_1, c_2) \lesssim^f_n (v_2, c_2)$ if and only if $v_1, v_2 \notin V_n$ or $v_1 = v_2 \in V_n$ and $c_1 \lesssim^f_n c_2$.

We note that $\mathcal{R}|_V$ is decreasing in V, i.e. $V \subseteq W$ implies $\mathcal{R}|_V \supseteq \mathcal{R}|_W$. This implies that $\lesssim^f_{n+1} \subseteq \lesssim^f_n$ for all n. We also note that the only difference between this definition and Definition 6.2 is the restriction on the values in the definition of $\lesssim^f_{n+1}$. Obviously $\lesssim_n \subseteq \lesssim^f_n$ for all n which implies $\lesssim \subseteq \lesssim_\omega \subseteq \lesssim^f_\omega$. It is easy to prove that $\lesssim^f_\omega$ actually is a preorder and has all the properties stated in Theorem 3.5. The proof for this is straightforward and is left to the reader. Now let us have a further look at our previous example, Example 6.3.

Example 6.5 *Let* $\mathbf{p}$ *and* $\mathbf{q}$ *be defined as in Example 6.3. Obviously* $\mathbf{p} \lesssim \mathbf{q}$ *and therefore* $\mathbf{p} \lesssim^f_\omega \mathbf{q}$*. We have also shown that* $\mathbf{q} \not\lesssim_\omega \mathbf{p}$ *and thereby* $\mathbf{q} \not\lesssim \mathbf{p}$*. On the other hand one may show that* $\mathbf{q} \lesssim^f_\omega \mathbf{p}$ *by showing that* $\mathbf{q} \lesssim^f_n \mathbf{p}$ *for all* n *by induction.*

We summarize these results of this section in the following lemma:

Lemma 6.6

1. *On any* ALTS, $\lesssim \subseteq \lesssim_\omega \subseteq \lesssim^f_\omega$.
2. *There is an* ALTS *on which* $\lesssim^f_\omega \not\subseteq \lesssim_\omega \not\subseteq \lesssim$.

7 The Full Abstractness

We have already proven that he preorder $\sqsubseteq_{E_{rec}}$ and the preorder induced by the denotational model, which we from now on refer to as$\sqsubseteq_{\overline{ACT}}$, coincide over $\mathbf{C}CS_L$. In this section we will prove that $\lesssim^f_\omega$ also coincides with these preorders over this language. Furthermore we prove that the preorders $\lesssim_\omega$ and $\leq_{E_{rec}}$ also coincide. Finally we show that over the language $\mathbf{C}CS_L$, all five preordered mentioned above, coincide. Here we want to remind the reader of, that considered over a general $ALTS$, $\lesssim_\omega$ is strictly included in $\lesssim^f_\omega$. Also it should be clear from Example 6.3 that in general $\leq_{E_{rec}}$ cannot be expected to coincide with a preorder derived from an algebraic model (although at this point it is not clear how the syntactically finite or compact approximations should look like for languages with parameterized recursion).

To prove that $\sqsubseteq_{E_{rec}} = \lesssim^f_\omega$ over $\mathbf{C}CS_L$ it is sufficient to prove the soundness and the completeness of the rules that define $\sqsubseteq_{E_{rec}}$ with respect to $\lesssim^f_\omega$ over this language. Similarly, as clearly $\leq_{E_{rec}} \subseteq \sqsubseteq_{E_{rec}}$, to prove that $\leq_{E_{rec}} = \lesssim_\omega$, it is sufficient to prove that the rules that define $\leq_{E_{rec}}$ are sound and complete with respect to $\lesssim_\omega$ over this language.

We start by proving the first of these three properties. In the proof we will use some standard techniques which are used to prove similar completeness results in the literature [Hen88a, AH92, HI93]. Therefore we start by defining a suitable notion of a "finitary part", i. e. the *value-finitary part*, of a relation over processes and that of a *value-finitary relation.* The definition is based on the same idea as the one given in [Hen81]. The only difference is that we use syntactically compact terms in our definition whereas Hennessy uses syntactically finite or recursion-free terms. We will then show that the preorder $\lesssim^f_\omega$ is the value-finitary part of the preorders $\lesssim$ and $\lesssim^f_\omega$ and therefore that $\lesssim^f_\omega$ is value-finitary in our sense. We start by defining the value-finitary part of a relation over CCS_L.

Definition 7.1 For any relation $\mathcal{R}$ over CCS_L we define the *value-finitary part* of $\mathcal{R}$, $\mathcal{R}^F$, by

$$\mathbf{t}\mathcal{R}^F\mathbf{u} \text{ iff for all } \mathbf{c}t \in \mathbf{c}oCCS_L, \mathbf{c}t\mathcal{R}\mathbf{t} \text{ implies } \mathbf{c}t\mathcal{R}\mathbf{u}.$$

$\mathcal{R}$ is *value-finitary* if $\mathcal{R} = \mathcal{R}^F$. □

The following lemma shows how we may structure the proof of full abstractness.

Lemma 7.2 *Assume that the preorder* $\preceq \subseteq CCS_L \times CCS_L$ *satisfies the following conditions:*

1. *Value-finitariness: For all* $\mathbf{t}, \mathbf{u} \in CCS_L$ $\mathbf{t} \preceq \mathbf{u}$ *iff* $\forall \mathbf{c}t.\ \mathbf{c}t \preceq \mathbf{t} \Rightarrow\ \mathbf{c}t \preceq \mathbf{u}$.
2. *Partial soundness: The proof system* $E^{-\omega}_{rec}$ *is sound with respect to* $\preceq$.

3. Partial completeness: For all $\mathbf{c}t \in \mathbf{c}oCCS_L$ *and* $\mathbf{t} \in CCS_L$ $\mathbf{c}t \preceq \mathbf{t}$ *implies* $\mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$.

Then for all $\mathbf{t}, \mathbf{u} \in CCS_L$

$$\mathbf{t} \preceq \mathbf{u} \textit{ if and only if } \mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}.$$

Proof First we have:

$$\begin{array}{lll} & \mathbf{t} \preceq \mathbf{u} & \\ \text{iff} & \forall \mathbf{c}t.\ \mathbf{c}t \preceq \mathbf{t} \Rightarrow \mathbf{c}t \preceq \mathbf{u} & \text{by 1.} \\ \text{iff} & \forall \mathbf{c}t.\ \mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t} \Rightarrow \mathbf{c}t \sqsubseteq_{E_{rec}} \mathbf{u} & \text{by 2. and 3.} \end{array}$$

Now we proceed as follows: Assume $\mathbf{t} \preceq \mathbf{u}$ and therefore that

$$\forall \mathbf{c}t.\ \mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t} \Rightarrow \mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{u}. \tag{6}$$

As $\mathbf{t}^{(n)} \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$ and $\mathbf{t}^{(n)} \in \mathbf{c}oCCS_L$ then (6) implies that $\mathbf{t}^{(n)} \sqsubseteq_{E_{rec}} \mathbf{u}$. As this holds for all n, the ω-rule implies that $\mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}$.

Next assume that $\mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}$. To prove that $\mathbf{t} \preceq \mathbf{u}$ it is sufficient to prove that (6) holds. So assume that $\mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$. Then, by transitivity of $\sqsubseteq_{E_{rec}^{-\omega}}$, $\mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{u}$. □

7.1 Value-finitariness

Following [Hen81] next we will prove that on $\mathbf{c}oCCS_L \times \mathbf{C}CS_L$ the preorders $\sqsubsetapprox$ and $\sqsubsetapprox_\omega^f$ coincide. Consequently they both have the same value-finitary part. To show this we need a measure on $\mathbf{c}oCCS_L$ that both measures the structural depth of the term and the number of values it uses. We give the following definitions.

Definition 7.3 For a syntactically finite term d we define the structural depth $sd(d)$ by:

1. $sd(nil) = sd(\Omega) = 0$,
2. $sd(\mu.d) = 1 + sd(d)$,
3. $sd(op(d_1, \ldots, d_n) = 1 + \sum_{i=1}^{n} sd(d_i)$, $op \in \Sigma$,
4. $sd([x]d) = sd(e, d) = 1 + sd(d)$,
5. $sd(be \longrightarrow d_1, d_2) = 1 + sd(d_1) + sd(d_2)$.

□

From this definition we can easily derive that if $\mathbf{d} \xrightarrow{\mu} \mathbf{t}$ then $sd(\mathbf{t}) \leq sd(\mathbf{d}) - 1$. Also for all $v \in Val$, $sd(d[v/x]) = sd(d)$ and therefore $sd(([x]d)(v)) = 1 + sd(d)$.

The *support* of a compact term is the set of values the term uses in a non-trivial way. Formally this is defined as follows:

Definition 7.4 The support of the term $\mathbf{c}t \in \mathbf{c}oCCS_L$, $Supp(\mathbf{c}t)$, is defined by structural recursion as:

1. $Supp(nil) = Supp(\Omega) = \emptyset$,
2. $Supp(op(\mathbf{c}p_1, \ldots, \mathbf{c}p_n)) = \bigcup_{i=1}^n Supp(\mathbf{c}p_i)$,
3. $Supp(pre.\mathbf{c}t) = Supp(\mathbf{c}t)$,
4. $Supp((e, \mathbf{c}p)) = Supp(\mathbf{c}p) \cup \{[\![e]\!]\}$,
5. $Supp([x]\,(x : \underline{w} \longrightarrow \underline{\mathbf{c}p})) = \{\underline{w}\} \cup \bigcup_{i=1}^n Supp(\mathbf{c}p_i)$.

Note that $Supp(\mathbf{c}t)$ is a finite set. We define the value-depth of $\mathbf{c}t$, $vd(\mathbf{c}t)$ by $vd(\mathbf{c}t) = min\{n | Supp(\mathbf{c}t) \subseteq V_n\}$. □

Now we prove the following.

Proposition 7.5 *For all* $\mathbf{c}t \in \mathbf{c}oCCS_L$ *and* $\mathbf{t} \in \mathbf{C}CS_L$,

$$\mathbf{c}t \sqsubseteq_\omega^f \mathbf{t} \textit{ if and only if } \mathbf{c}t \sqsubseteq \mathbf{t}$$

and therefore $(\sqsubseteq_\omega^f)^F = \sqsubseteq^F$ *on* $\mathbf{C}CS_L$.

Proof As the "if" part is already known it is sufficient to prove the "only if" part. To this end we prove the following stronger result.

For all $\mathbf{c}t \in \mathbf{c}oCCS_L$ and $\mathbf{t} \in \mathbf{C}CS_L$

$$\mathbf{c}t \sqsubseteq_m^f \mathbf{t} \Rightarrow \mathbf{c}t \sqsubseteq \mathbf{t}$$

for all m where $m \geq M(\mathbf{c}t) = sd(\mathbf{c}t) + vd(\mathbf{c}t)$.

The proof of this statement proceeds by induction on $M(\mathbf{c}t)$.

$M(\mathbf{c}t) = 0$: We have two cases: $\mathbf{c}t = \Omega$, which is trivial, and $\mathbf{c}t = nil$ which we will have a further look at. As $nil \downarrow$, the definition of $\sqsubseteq_m^f$ implies that $\mathbf{t} \downarrow$. Furthermore as $nil \not\xrightarrow{\mu}$ for all μ this is also true for $\mathbf{t}$. This proves that $\mathbf{c}t = nil \sqsubseteq \mathbf{t}$.

$M(\mathbf{c}t) = k + 1$: Assume we have proved the result for all $\mathbf{c}t'$ with $M(\mathbf{c}t') \leq k$ and that $\mathbf{c}t \sqsubseteq_m^f \mathbf{t}$, where $m \geq M(\mathbf{c}t) = k + 1$. We want to prove that $\mathbf{c}t \sqsubseteq \mathbf{t}$. As $\mathcal{F}(\sqsubseteq) = \sqsubseteq$, it is sufficient to show that $\mathbf{c}t\mathcal{F}(\sqsubseteq)\mathbf{t}$. We proceed by case analysis on the structure of $\mathbf{c}t$.

$\mathbf{c}t \in \mathbf{C}CS_L^{proc}$:

1. Assume $\mathbf{c}t \xrightarrow{\mu} \mathbf{u}$. By definition of $\sqsubseteq_m^f$, $\mathbf{t} \xrightarrow{\mu} \mathbf{u}'$ for some $\mathbf{u}'$ such that $\mathbf{u} \sqsubseteq_{m-1}^f \mathbf{u}'$. Also, by definition of $\mathbf{c}oCCS_L$, $\mathbf{u} \in \mathbf{c}oCCS_L$. Now $vd(\mathbf{u}) \leq vd(\mathbf{c}t)$ and $sd(\mathbf{u}) < sd(\mathbf{c}t)$. Thus $m - 1 \geq k \geq M(\mathbf{u})$ and by the induction $\mathbf{u} \sqsubseteq \mathbf{u}'$.
2. Now assume $\mathbf{c}t \downarrow$, by definition of the preorder $\sqsubseteq_\omega^f$ also $\mathbf{t} \downarrow$. Furthermore assume that $\mathbf{c}t \downarrow$, $\mathbf{t} \downarrow$ and that $\mathbf{t} \xrightarrow{\mu} \mathbf{u}'$. Then $\mathbf{c}t \xrightarrow{\mu} \mathbf{u}$ for some $\mathbf{u}$ such that $\mathbf{u} \sqsubseteq_{m-1}^f \mathbf{u}'$. In a similar way as before, the induction implies $\mathbf{u} \sqsubseteq \mathbf{u}'$, which completes the proof in this case.

$\mathbf{c}t \in \mathbf{c}oCCS_L^{fun}$: Then $\mathbf{t}$ and $\mathbf{c}t$ have the form $\mathbf{t} = [x]p'$ where $x \in Var$, $p' \in CCS_L^{proc}$ and $\mathbf{c}t = [y]p$ for some $y \in Var$ where $p = y : \underline{w} \longrightarrow \underline{\mathbf{c}p'}$, for some $\underline{w}$ and $\underline{\mathbf{c}p'}$. Our assumption is that $[y]p \sqsubseteq^f_m [x]p'$, i.e. that $p[v/y] \sqsubseteq^f_m p'[v/x]$ for all $v \in V_m$. We have to prove that $[y]p \sqsubseteq [x]p'$, i.e. that $p[v/y] \sqsubseteq p'[v/x]$ for all $v \in Val$. This is obviously true for $v \notin \{\underline{w}_n\}$ as in that case $p[v/y] \sim \Omega$. So assume that $v \in \{\underline{w}\}$. As $m \geq vd(\mathbf{c}t)$, $\{\underline{w}\} \subseteq V_m$. Furthermore we know from the assumption that for all $w_i, i \leq n$, $p[w_i/y] \sqsubseteq^f_m p'[w_i/x]$ and $p[w_i/y] \sim \mathbf{c}p'_i$. This implies that $\mathbf{c}p'_i \sqsubseteq^f_m p'[w_i/x]$. Now we have that $M(\mathbf{c}p'_i) < M(\mathbf{c}t) = k + 1$, i. e. $M(\mathbf{c}p'_i) \leq k$. As $m \geq k + 1 > k \geq M(\mathbf{c}p'_i)$ the induction applies and we may conclude that $\mathbf{c}p'_i \sqsubseteq p'[w_i/x]$. Again, as $\mathbf{c}p'_i \sim p[w_i/y]$, this implies that $p[w_i/y] \sqsubseteq p'[w_i/y]$ as we wanted to prove.

$\mathbf{c}t \in \mathbf{c}oCCS_L^{pair}$: Now $\mathbf{c}t = (v', \mathbf{c}p')$ and $\mathbf{t} = (v'', \mathbf{p}'')$. By the definition of the preorder and the assumption on m, $v' = v'' \in V_m$ and $\mathbf{c}p' \sqsubseteq^f_m \mathbf{p}$. As before $m > k \geq M(\mathbf{c}p')$ and the result follows from the induction.

□

We will now show that the preorder $\sqsubseteq^f_\omega$ is finitary and therefore that it is the finitary part of $\sqsubseteq$. Again following closely [Hen81], we introduce the so called compact projections and show some of their properties. The remainder of this section is devoted to this. We adopt Abramsky's definition of the *sort* of a term $\mathbf{t}$, $Sort(\mathbf{t})$, as the set of channel names it uses.

Definition 7.6 The sort of $\mathbf{t} \in \mathbf{C}CS_L$, $Sort(\mathbf{t}) \subseteq Chan$, is given by

1. $Sort(\mathbf{p}) = \{c \in Chan | \mathbf{p} \xrightarrow{a}, chan(a) = c\} \cup \bigcup\{Sort(\mathbf{u}) | \exists \mu. \mathbf{p} \xrightarrow{\mu} \mathbf{u}\}$,
2. $Sort([x]t) = \bigcup\{Sort(t[v/x]) | v \in Val\}$,
3. $Sort(e, \mathbf{q}) = Sort(\mathbf{q})$.

□

Note that, because of our restriction to finite renamings, $Sort(\mathbf{t})$ is finite for all $\mathbf{t}$ [Abr91, AH92].

Definition 7.7 (Compact Projections) We define the n-th projection of $\mathbf{t}$ on $\mathbf{c}oCCS_L$ inductively as follows:

1. (a) $\mathbf{p}^{[0]} = \Omega$,
   (b) $\mathbf{p}^{[n+1]} = \sum\{\mu.\mathbf{t}^{[n]} | \mathbf{p} \xrightarrow{\mu} \mathbf{t}\} + \{\Omega | \mathbf{p} \uparrow\}$,
2. (a) $([x]p)^{[0]} = [x]\Omega$,
   (b) $([x]p)^{[n+1]} = [x]x : (v_1, \cdots, v_{n+1}) \to ((p[v_1/x])^{[n+1]}, \cdots, (p[v_{n+1}/x])^{[n+1]})$,
3. $(v, \mathbf{p})^{[n+1]} = \begin{cases} (v, \mathbf{p}^{[n+1]}) & \text{if } v \in V_{n+1}, \\ (v, \Omega) & \text{otherwise.} \end{cases}$

Note that the sum in 1.(b) only makes sense as we are summing over a finite set (up to commutativity, absorption and α-congruence). That this is the case may be proved by induction on n. $\Box$

The syntactically compact projections have the following properties:

Lemma 7.8 *For all* $\mathbf{t} \in \mathbf{C}CS_L$ *and all* n,

1. $\mathbf{t}^{[n]} \in \mathbf{c}oCCS_L$,
2. $\mathbf{t}^{[n]} \sim_n^f \mathbf{t}$.

Proof Follows by a simple induction on n, combined with an inner induction on the structural depth of $\mathbf{t}$ for the inductive step. $\Box$

The following results investigate the relationship between a term $\mathbf{t}$ and its syntactically compact projections in more detail.

Lemma 7.9 *For all* $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$

1. $\mathbf{t}^{[0]} \sqsubseteq_\omega^f \mathbf{t}^{[1]} \sqsubseteq_\omega^f \cdots \sqsubseteq_\omega^f \mathbf{t}^{[n]} \sqsubseteq_\omega^f \cdots \sqsubseteq_\omega^f \mathbf{t}$.
2. *If* $\mathbf{t}^{[0]} \sqsubseteq_\omega^f \mathbf{t}^{[1]} \sqsubseteq_\omega^f \cdots \sqsubseteq_\omega^f \mathbf{t}^{[n]} \sqsubseteq_\omega^f \cdots \sqsubseteq_\omega^f \mathbf{u}$ *then* $\mathbf{t} \sqsubseteq_\omega^f \mathbf{u}$, *i.e.* $\mathbf{t}$ *is a minimal upper bound* [2] *of the chain with respect to* $\sqsubseteq_\omega^f$.
3. *The term* $\mathbf{t}$ *is a minimal upper bound for the set* $App(\mathbf{t}) = \{\mathbf{c}t \in \mathbf{c}oCCS_L | \mathbf{c}t \sqsubseteq_\omega^f \mathbf{t}\}$ *with respect to* $\sqsubseteq_\omega^f$.

Proof

1. We first prove that for all n
$$\mathbf{t}^{[n]} \sqsubseteq_\omega^f \mathbf{t}^{[n+1]}.$$
In order to do that we prove a slightly stronger result:
$$\forall m \geq n.\, \mathbf{t}^{[n]} \sqsubseteq_m^f \mathbf{t}^{[n+1]}.$$
We prove this by induction on n. The base case $n = 0$ is immediate as $\mathbf{t}^{[0]} \sqsubseteq_m^f \mathbf{t}^{[1]}$ for all m is trivial. So assume
$$\mathbf{t}^{[k]} \sqsubseteq_m^f \mathbf{t}^{[k+1]} \text{ for } m \geq k$$
and we will prove that
$$\mathbf{t}^{[k+1]} \sqsubseteq_{m+1}^f \mathbf{t}^{[k+2]} \text{ for } m \geq k.$$
To this end assume $m \geq k$. We proceed by induction on the structural depth of $\mathbf{t}$. We have the following cases.

[2]Note that a minimal upper bound of a preorder is unique up to the induced equivalence.

$\mathbf{t} = \mathbf{p} \in \mathbf{C}CS_L^{proc}$: Assume $\mathbf{p}^{[k+1]} \xrightarrow{\mu} \mathbf{t}$, then $\mathbf{p} \xrightarrow{\mu} \mathbf{u}$ for some $\mathbf{u}$ such that $\mathbf{u}^{[k]} = \mathbf{t}$. Also $\mathbf{p}^{[k+2]} \xrightarrow{\mu} \mathbf{u}^{[k+1]}$ and by the induction , as $m \geq k$, $\mathbf{u}^{[k]} \sqsubseteq_m^f \mathbf{u}^{[k+1]}$. Thus the first condition of the definition of the preorder $\sqsubseteq_{m+1}^f$ is satisfied. We now note that $\mathbf{p} \downarrow$ if and only if $\mathbf{p}^{[i]} \downarrow$ for all i and the second condition of the definition can be met in a similar way to the first one.

$\mathbf{t} = [x]p \in \mathbf{C}CS_L^{fun}$: By definition

$$([x]p)^{[i+1]} = [x]x : (v_1, \cdots, v_{i+1}) \to ((p[v_1/x])^{[i+1]}, \cdots, (p[v_{i+1}/x])^{[i+1]}).$$

We have to prove that

$$(([x]p)^{[k+1]})(v) \sqsubseteq_{m+1}^f ([x]p^{[k+2]})(v)$$

for all $v \in Val$. First we note that for all $v \in V_{k+1}$

$$([x]p)^{[k+1]}(v) \sim (p[v/x])^{[k+1]}$$

and

$$([x]p)^{[k+2]}(v) \sim (p[v/x])^{[k+2]}$$

and the result follows from the inner induction, the transitivity and the fact that $\sqsubseteq \subseteq \sqsubseteq_{m+1}^f$. Otherwise if $v \notin V_{k+1}$ then $([x]p)^{[k+1]}(v) \sim \Omega$ and the result follows.

$\mathbf{t} = (v, \mathbf{p}) \in \mathbf{C}CS_L^{pair}$: Similar.

Next we prove $\mathbf{t}^{[n]} \sqsubseteq_\omega^f \mathbf{t}$ for all n. We know from Lemma 7.8 that $\mathbf{t}^{[k]} \sqsubseteq_k^f \mathbf{t}$ for all k. Furthermore for any $m \geq n$

$$\mathbf{t}^{[n]} \sqsubseteq_\omega^f \mathbf{t}^{[m]} \sqsubseteq_m^f \mathbf{t}.$$

Thus $\mathbf{t}^{[n]} \sqsubseteq_m^f \mathbf{t}$ for all $m \geq n$ which proves the statement.

2. To prove that $\mathbf{t}$ is a minimal upper bound of the chain assume

$$\mathbf{t}^{[0]} \sqsubseteq_\omega^f \mathbf{t}^{[1]} \sqsubseteq_\omega^f \cdots \mathbf{t}^{[n]} \sqsubseteq_\omega^f \cdots \sqsubseteq_\omega^f \mathbf{u}.$$

As $\mathbf{t} \sim_n \mathbf{t}^{[n]}$ this implies $\mathbf{t} \sqsubseteq_n^f \mathbf{u}$ for all n and therefore $\mathbf{t} \sqsubseteq_\omega^f \mathbf{u}$.

3. Follows from statement 1., as $\{\mathbf{t}^{[n]} | n = 1, \cdots\} \subseteq App(\mathbf{t})$.

$\square$

The following theorem is a direct consequence of the lemma above.

Theorem 7.10 *Over $\mathbf{C}CS_L$ following holds.*

1. $\sqsubseteq_\omega^f = (\sqsubseteq_\omega^f)^F$.

2. *The preorder $\sqsubseteq_\omega^f$ is the value-finitary part of $\sqsubseteq$, i.e. $\sqsubseteq^F = \sqsubseteq_\omega^f$.*

Proof

1. That $\lesssim^f_\omega \subseteq (\lesssim^f_\omega)^F$ is obvious so we only have to prove the other inclusion. Thus assume

$$\forall \mathbf{c}t \in \mathbf{co}CCS_L.\ \mathbf{c}t \lesssim^f_\omega \mathbf{t} \text{ implies } \mathbf{c}t \lesssim^f_\omega \mathbf{u}.$$

   This is equivalent to saying that $App(\mathbf{t}) \subseteq App(\mathbf{u})$ and the result follows from Lemma 7.9.

2. By Proposition 7.5, $\lesssim^F = (\lesssim^f_\omega)^F$ and the result follows from part 1. of this theorem.

□

7.2 Partial Soundness and Completeness

This subsection is devoted to the proof of the soundness of the proof system $E^{-\omega}_{rec}$ and the partial completeness of E_{rec} with respect to $\lesssim^f_\omega$.

We start by proving the soundness of the proof system $E^{-\omega}_{rec}$, i.e the proof system that consists of the system E_{rec} where the ω-rule is omitted. This follows from the following Lemma.

Lemma 7.11 (Partial Soundness) *The proof system $E^{-\omega}_{rec}$ is sound with respect to $\lesssim$ over $\mathbf{C}CS_L$.*

Proof The soundness of $E^{-\omega}_{rec}$ with respect to $\lesssim$ can be shown by proving

$$\mathbf{t} \sqsubseteq_{E^{-\omega}_{rec}} \mathbf{u} \text{ implies } \mathbf{t} \lesssim \mathbf{u}$$

by induction on the depth of the proof tree for $\mathbf{t} \sqsubseteq_{E^{-\omega}_{rec}} \mathbf{u}$. The details of the proofs are omitted. □

Here we want to point out that the ω-rule is not sound with respect to the preorder $\lesssim$ as shown by Example 6.1. Furthermore at this point we have not proved the soundness of the ω-rules for $\lesssim_\omega$ or $\lesssim^f_\omega$.
Next we prove the mentioned partial completeness result, i.e. that for all $\mathbf{c}t$ and $\mathbf{t}$,

$$\mathbf{c}t \lesssim \mathbf{t} \Rightarrow \mathbf{c}t \sqsubseteq_{E_{rec}} \mathbf{t}.$$

This proof follows very much the same pattern as the proof for a similar partial completeness result in [AH92]. First we introduce the notion of *head normal forms* and prove a corresponding normalization theorem.

Definition 7.12 A term in CCS^{proc}_L is said to be in a *head normal form* if it has the form $\sum_i \mu_i t_i$.

□

Lemma 7.13 *If $\mathbf{p} \in \mathbf{C}CS^{proc}_L$ and $\mathbf{p}\downarrow$ then there is a head normal form, $hnf(\mathbf{p})$, such that $\mathbf{p} =_{E^{-\omega}_{rec}} hnf(\mathbf{p})$.*

Proof We prove the Lemma by induction on the length of the derivation of $\mathbf{p}\downarrow$. We proceed by a case analysis on the structure of $\mathbf{p}$. □

Here it is important to notice that we only use the partial proof system $E_{rec}^{-\omega}$ in the normalization procedure as the soundness of the ω-rule with respect to the preorder $\precsim_{\omega}^{f}$ has not been proven yet.

Notation 7.14 *Let $p, q \in CCS_L^{proc}$ and $t, u \in \mathbf{C}CS_L$. To simplify the notation we will in what follows use the following convention (where abs stands for abstraction and app for application):*

1. *$abs(t|u)$ for*
   - (a) $abs(p|q) = p|q$,
   - (b) $abs([x]t|p) = [x](t|p)$,
   - (c) $abs(p|[x]t) = [x](p|t)$,
   - (d) $abs((v,p)|q) = (v, p|q) = abs(p|(v,q)$.
2. *$app(t|u)$ for*
   - (a) $app([x]t|(v,p)) = t[v/x]|p$,
   - (b) $app((v,p)|[x]t) = p|t[v/x]$.

Using this notation we get that if $\mathbf{p} \xrightarrow{\mu} \mathbf{p}'$ then $\mathbf{p}|\mathbf{q} \xrightarrow{\mu} abs(\mathbf{p}'|\mathbf{q})$ and $\mathbf{q}|\mathbf{p} \xrightarrow{\mu} abs(\mathbf{q}|\mathbf{p}')$. Furthermore if $\mathbf{p} \xrightarrow{c!} \mathbf{o}$ and $\mathbf{q} \xrightarrow{c?} \mathbf{f}$ then $\mathbf{p}|\mathbf{q} \xrightarrow{\tau} app(\mathbf{o}|\mathbf{f})$ and $\mathbf{q}|\mathbf{p} \xrightarrow{\tau} app(\mathbf{f}|\mathbf{o})$. We use this notation to formulate the following lemma:

Lemma 7.15 *For all $\mathbf{p} \in \mathbf{C}CS_L^{proc}$, $\mathbf{t} \in \mathbf{C}CS_L$ and μ we have that $\mathbf{p} \xrightarrow{\mu} \mathbf{t}$ implies $\mathbf{p} =_{E_{rec}} \mathbf{p} + \mu.\mathbf{t}$.*

Proof We prove the statement by proving that for all closed terms $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{t}$ following holds:

1. $(\mathbf{p} + \mu.\mathbf{t})|\mathbf{q} =_{E_{rec}} (\mathbf{p} + \mu.\mathbf{t})|\mathbf{q} + \mu.abs(\mathbf{t}|\mathbf{q})$
2. $\mathbf{q}|(\mathbf{p} + \mu.\mathbf{t}) =_{E_{rec}} \mathbf{q}|(\mathbf{p} + \mu.\mathbf{t}) + \mu.abs(\mathbf{q}|\mathbf{t})$
3. $(\mathbf{p} + a.\mathbf{t})|(\mathbf{q} + \overline{a}.\mathbf{u}) =_{E_{rec}} (\mathbf{p} + a.\mathbf{t})|(\mathbf{q} + \overline{a}.\mathbf{u}) + \tau.app(\mathbf{t}|\mathbf{u})$.

The proof is basically identical to a proof for similar properties in [AH92] and is omitted. □

Theorem 7.16 (Partial Completeness) *For all $\mathbf{c}t \in \mathbf{c}oCCS_L$ and $\mathbf{t}\mathbf{C}CS_L$, $\mathbf{c}t \precsim \mathbf{t}$ implies $\mathbf{c}t \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{t}$.*

Proof It is sufficient to prove the result for $\mathbf{c}t$ in Ω-normal form as the general result follows from the normalization result in Lemma 5.4, and the soundness of $\sqsubseteq_{E_{rec}^{-\omega}}$ with respect to $\precsim$. We proceed by a case analysis on the form of $\mathbf{c}t$ but only give the details of the case where $\mathbf{c}t = \mathbf{c}p \in \mathbf{c}oCCS_L$. In this case $\mathbf{t} = \mathbf{p} \in \mathbf{C}CS_L^{proc}$.

To this end assume $\mathbf{n}p \sqsubseteq\!\!\!\!\!\sim \mathbf{p}$ where $\mathbf{n}p$ is an Ω-normal form and we will prove that $\mathbf{n}p \sqsubseteq_{E_{rec}} \mathbf{p}$. The proof proceeds by induction on $sd(\mathbf{n}p)$, the structural depth of $\mathbf{n}p$ defined in Definition 7.3. So assume the theorem is true for all $\mathbf{n}p'$ with $sd(\mathbf{n}p') \leq k$ and that $sd(\mathbf{n}p) = k + 1$. We proceed by a case analysis on the form of $\mathbf{n}p$.

$\mathbf{n}p = nil + \Omega$: Then $\mathbf{n}p =_{E_{rec}^{-\omega}} \Omega \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{p}$.

$\mathbf{n}p = nil$: $nil \sqsubseteq\!\!\!\!\!\sim \mathbf{p}$ implies $\mathbf{p} \downarrow$. Thus $\mathbf{p}$ has a head normal form $h(\mathbf{p})$ with $h(\mathbf{p}) =_{E_{rec}^{-\omega}} \mathbf{p}$. As $nil \stackrel{\mu}{\not\longrightarrow}$, $h(\mathbf{p}) \stackrel{\mu}{\not\longrightarrow}$ for all μ which implies that $h(\mathbf{p}) = nil$. Therefore $\mathbf{n}p =_{E_{rec}^{-\omega}} h(\mathbf{p}) =_{E_{rec}^{-\omega}} \mathbf{p}$.

$\mathbf{n}p = \sum_i \mu_i.\mathbf{p}_i\{+\Omega\}$: We prove this case in three steps.

1. $\mathbf{p}+\mathbf{n}p \sqsubseteq_{E_{rec-\omega}} \mathbf{p}$: Assume $\mathbf{n}p \xrightarrow{\mu} \mathbf{p}'$ then $\mu = \mu_i$ and $\mathbf{p}' = \mathbf{p}_i$ for some i. As $\mathbf{n}p \sqsubseteq\!\!\!\!\!\sim \mathbf{p}$ this implies that $\mathbf{p} \xrightarrow{\mu_i} \mathbf{q}_i$ for some $\mathbf{q}_i$ where $\mathbf{p}_i \sqsubseteq\!\!\!\!\!\sim \mathbf{q}_i$. By applying the induction hypothesis we obtain that $\mathbf{p}_i \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{q}_i$ and, by substitutivity, that $\mu_i.\mathbf{p}_i \sqsubseteq_{E_{rec}^{-\omega}} \mu_i.\mathbf{q}_i$. Thus by substitutivity and Lemma 7.15
$$\mathbf{p} + \mu_i.\mathbf{p}_i \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{p} + \mu_i.\mathbf{q}_i =_{E_{rec-\omega}} \mathbf{p}.$$
Repeated use of this result, substitutivity and transitivity implies $\mathbf{p} + \mathbf{n}p \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{p}$.
2. $\mathbf{n}p \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{p} + \mathbf{n}p$: If Ω is a summand of $\mathbf{n}p$ then $\mathbf{n}p \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{n}p + \Omega \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{n}p + \mathbf{p}$. So assume that $\mathbf{n}p \downarrow$. As $\mathbf{n}p \sqsubseteq\!\!\!\!\!\sim \mathbf{p}$ this implies $\mathbf{p} \downarrow$ and therefore that $\mathbf{p}$ has a head normal form $\mathbf{p} =_{E_{rec}^{-\omega}} h(\mathbf{p}) = \sum_j \gamma_j.\mathbf{q}_j$. As the proof system $E_{rec}^{-\omega}$ is sound with respect to $\sqsubseteq\!\!\!\!\!\sim$, $\mathbf{n}p \sqsubseteq\!\!\!\!\!\sim h(\mathbf{p})$. Thus $\mathbf{p} \xrightarrow{\gamma_j} \mathbf{q}_j$ implies that $\gamma_j = \mu_{i_j}$ for some i_j and that $\mathbf{n}p \xrightarrow{\mu_{i_j}} \mathbf{p}_{i_j}$ for some $\mathbf{p}_{i_j}$ such that $\mathbf{p}_{i_j} \sqsubseteq\!\!\!\!\!\sim \mathbf{q}_j$. Now by proceeding in a similar way as in the previous case we get that
$$\mathbf{n}p = \mathbf{n}p + \sum_j \mu_{i_j}.\mathbf{p}_{i_j} \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{n}p + \sum_j \gamma_j.\mathbf{q}_j \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{n}p + \mathbf{p}.$$
3. Finally *1.* and *2.* imply $\mathbf{n}p \sqsubseteq_{E_{rec}^{-\omega}} \mathbf{p}$.

The remaining cases can be proven in a similar way. □

Theorem 7.17 (Soundness and completeness of the value-finite) *For all* $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$,
$$\mathbf{t} \sqsubseteq\!\!\!\!\!\sim^{f}_{\omega} \mathbf{u} \quad \textit{if and only if} \quad \mathbf{t} \sqsubseteq_{E_{rec}} \mathbf{u}.$$

Proof This follows from Theorem 7.2 as the conditions of the theorem are ensured by Theorem 7.10, Lemma 7.11 and Theorem 7.16. □

Theorem 7.18 (Soundness and completeness for value-infinite proof system) *For all* $\mathbf{t}, \mathbf{u} \in \mathbf{C}CS_L$,
$$\mathbf{t} \sqsubseteq\!\!\!\!\!\sim_{\omega} \mathbf{u} \quad \textit{if and only if} \quad \mathbf{t} \leq_{E_{rec}} \mathbf{u}.$$

Proof If we replace the "compact elements" $\mathbf{t}$ in Theorem 7.2 by "syntactically finite elements" $\mathbf{d}$ and $\sqsubseteq_{E_{rec}}$ by $\leq_{E_{rec}}$, its statement still holds (and exactly the same proof applies). Thus it is sufficient to prove that properties 1., 2. and 3. of that theorem hold for $\underset{\sim}{\sqsubset}_{\omega}$ with these changes.

The proof of property 1., i.e. that

$$(\forall \mathbf{d}.\mathbf{d} \underset{\sim}{\sqsubset}_{\omega} \mathbf{p} \Rightarrow \mathbf{d} \underset{\sim}{\sqsubset}_{\omega} \mathbf{q}) \text{ implies } \mathbf{p} \underset{\sim}{\sqsubset}_{\omega} \mathbf{q},$$

is a simplification of the proof of Theorem 7.10, the main difference being that we replace $\underset{\sim}{\sqsubset}^{f}_{\omega}$ by $\underset{\sim}{\sqsubset}_{\omega}$, the "compact terms" by the "syntactically finite terms" and the "compact projections" in Definition 7.7 by a similar notion of "syntactically finite projections" $t^{\{n\}}$ $t^{[n]}$ obtained from the same definition with the second clause replaced by $([x]p)^{\{n+1\}} = [x]p^{\{n+1\}}$ and the third one by $(v, \mathbf{p})^{\{n+1\}} = (v, \mathbf{p}^{\{n+1\}})$. Property 2. is still the content of Lemma 7.11 and 3. can be obtained in exactly the same way as the proof of Theorem 7.16 with the notion of Ω-normal forms extended to syntactically finite terms instead of only applying to compact terms. The details of the proof are left to the reader to check. □

Finally we compare the value-finitary and value-infinitary semantics.

Theorem 7.19 $\overline{ACT}[\![\mathbf{t}]\!] \sqsubseteq \overline{ACT}[\![\mathbf{u}]\!]$ *iff* $\mathbf{t} \underset{\sim}{\sqsubset}_{\omega} \mathbf{u}$.

Proof Clearly we only have to prove the "only if" implication". We will give an outline of the proof but refer to [Ing94, Chap.2-3] where the details of a similar proof are given. Let $(D, \prec)$ be the preordered set defined in the same way as $(K, \prec)$ in Definition 4.10 but with clause 2. replaced by

$$2.' \;\; c \in C, f \in Val \longrightarrow D \text{ implies } \langle c?, f\rangle \in D.$$

Furthermore we define D_n as the subset of D consisting of elements of depth at most n, ordered as before. We also define the operators exactly in the same way as on K. In the following we write $[\![\_]\!]$ instead of $\overline{ACT}[\![\_]\!]$ and $\sqsubseteq$ instead of $\sqsubseteq_{\overline{ACT}}$. Then the following holds (up to isomorphism in some cases):

1. The unique (Σ, C)-domain induced by $(\overline{D}, \prec, \Sigma_{\overline{D}}, C_{\overline{D}})$ is isomorphic to $\overline{ACT}$.
2. $D_0 \lhd D_1 \lhd \ldots$ is a directed ω-chain with ACT as the co-limit.
3. For each i there is projection mapping $\_^{i} : D \longrightarrow D^{i}$ that has the following properties:
   (a) If $d \in ACT$ then $d^i \in D_i$ and $d^i \sqsubseteq_{ACT} d$.
   (b) If $e \in D_i$ and $e \sqsubseteq d \in ACT$ then $e \sqsubseteq d^i$.
   (c) $d = \bigsqcup_i d^i$.
   (d) If $\mathbf{d}$ is a syntactically finite closed process term then $[\![\mathbf{d}]\!] \in D_i$ for some i.
   (e) For all i and $\mathbf{p}$, $[\![\mathbf{p}^{\{i\}}]\!] = [\![\mathbf{p}]\!]^i$. (This property can be proven by induction on i; it is language dependent and is actually the key to the proof of the full abstractness of the model with respect to $\underset{\sim}{\sqsubset}_{\omega}$ for the language CCS_L. Again we remind the reader of that in general these two semantics do not coincide as explained earlier.)

Now we proceed as follows:

$$[\![\mathbf{p}]\!] \sqsubseteq [\![\mathbf{p}]\!] \text{ implies } \forall i.[\![\mathbf{p}]\!]^i \sqsubseteq [\![\mathbf{q}]\!]^i \text{ implies } \forall i.[\![\mathbf{p}^{\{i\}}]\!] \sqsubseteq [\![\mathbf{q}^{\{i\}}]\!].$$

By induction on i we may prove that

$$\forall i.\mathbf{p}^{\{i\}} \sqsubset\!\!\!\sim_\omega \mathbf{q}^{\{i\}} \sqsubset\!\!\!\sim_\omega \mathbf{q}$$

which in turn implies that $\mathbf{p} \sqsubset\!\!\!\sim_\omega \mathbf{q}$. $\square$

8 Conclusion

In this last section we will give a summary of the main result of this paper and suggest some directions for further work.

In this paper we have defined a syntax and three kinds of semantics that support the late approach for value-passing calculi. First we defined a general syntax that supports this approach by extending the standard notion of a signature to the so-called applicative signature, a pair (Σ, C) consisting of a signature Σ and a set of channels C. Then we defined the language Late-CCS (CCS_L) which is a modification of the standard CCS with values due to the late semantic approach. This language is basically the π-calculus where the values allowed for are restricted to be of the simple type only. The language is obtained as an instantiation of the general class of languages we defined where the signature Σ is taken to be the set of the standard operators of CCS.

Then we defined a Plotkin style operational semantics [Plo81], and a suitable extension of the standard strong prebisimulation [Hen81, Wal90] to take value-passing based on the late approach into account. Thus we introduce the notion of applicative labelled transition system and the related notion of strong applicative bisimulation.

Next we have set up a general framework for describing the denotational semantics for value-passing calculi which support the late approach. For this purpose the standard notion of Σ-algebras and Σ-orders have been extended to (Σ, C)-algebras and (Σ, C)-orders.

A denotational model for Late-CCS is defined, an instantiation of the general class of models we defined. The carrier set of this model is an ω-algebraic cpo and is obtained as a ideal completion of the kernel of a suitably defined partial order. The operators are defined by first defining them as monotonic functions on the preorder, taking the induced monotonic functions on the kernel and taking their unique continuous extension to the whole cpo.

Of historical reasons the carrier of the model is also described as the solution to a recursive domain equation. It is a direct extension of a similar equation given by Abramsky in [Abr91] and a modification of the one given by Milne and Milner in [MM79]. As all the constructions we use in the definition of the equation are standard and well known to preserve completeness and ω-algebraicity the solution we obtain is an ω-algebraic *cpo*. By unfolding the recursive definition we get a representation of the compact elements, a preorder that turns out to coincide with our original preorder used to construct the model.

We have also presented two equationally based proof systems and compared them to the model. The first one is based on the ability of approximating infinite terms by so-called compact terms, syntactically finite terms which are interpreted as compact elements in the model, and is proven to be sound and complete with respect to the model. The algebraicity of the model and the way we define the operators makes it possible for us to reduce the proof of the soundness and completeness to a proof of the same property on a sub-language of the actual language, the so-called compact terms. This is an inductively defined language which denotes exactly the compact elements of the model.

The second proof system is based on the more standard syntactically finite approximation. The soundness of this proof system follows directly from the soundness of the stronger proof system based on the compact approximations. To prove the completeness of this system we have to prove that if $\mathbf{p}$ is smaller than $\mathbf{q}$ in the model then necessarily all the syntactic approximations of $\mathbf{p}$ have to be smaller than $\mathbf{q}$ in the model too. The proof of this property turned out to be nontrivial and was postponed until the investigation of the operational semantics.

Our next task was to compare the three different kinds of semantics we had put forward so far. First we show that for Late-CCS the original late bisimulation is too strong to meet the preorder of the model ACT. The algebraicity of the model implies that the preorder in the model is completely determined by the compact elements. Behaviourally this can be interpreted as meaning that the preorder may be obtained by some kind of finite observations. This is not the case for bisimulation as it is well known even for the pure calculus. Therefore we define a value-finitary version of the preorder $\sqsubseteq^f_\omega$ is defined by mimicking the ω-algebraicity of the model on the syntactical level and requiring that it is completely decided by the syntactically compact elements. This preorder is shown to coincide with the preorder in the model in the sense that the model is fully abstract with respect to it by proving that the value-finitary proof system is sound and complete with respect to it.

We also show that the ω-prebisimulation $\sqsubseteq_\omega$ is in general strictly finer than $\sqsubseteq^f_\omega$ but if restricted to $\mathbf{C}CS_L$ it is fully abstract with respect to the denotational model and thus coincides with $\sqsubseteq^f_\omega$. To prove this we have to appeal to more general results regarding the constructions of the type of model we offer. Thus we recall that the cpo ACT may be obtained as the co-limit of an ω-chain of domains $(D_i)_{i\in\omega}$ each of which consist of trees of finite depth but which are not necessarily value finite. Consequently each element d of ACT can be approximated by an ω-chain $(d^i)_{i\in\omega}$ of finitely deep trees which are not necessarily compact elements. The key to the completeness proof is the observation that for any $\mathbf{t} \in \mathbf{C}CS_L$ and any i, $\overline{ACT}[\![\mathbf{t}^{\{i\}}]\!] = \overline{ACT}[\![\mathbf{t}]\!]^i$ where $\mathbf{t}^{\{i\}}$ is the ith syntactically finite projection that appears in the proof of the finitariness of $\sqsubseteq_\omega$ and d^i is the the projection on D_i, the ith approximation of the cpo ACT. This is a language dependent property that cannot be expected to hold in such languages in general as Example 6.3 shows.

The example we used to distinguish $\sqsubseteq_\omega$ and $\sqsubseteq^f_\omega$ was expressed in an extension of CCS_L that allows for parameterized recursive definitions. Therefore we may conclude that this extension of CCS_L is strictly more expressive than CCS_L.

The main conclusion we can draw from the study performed in this paper is that operational semantics is more intuitive and in general more suitable for describing the semantics for communicating processes than denotational semantics, in particular if the communication involves exchange of data. However denotational models, based on an algebraic cpo, automatically ensure the finitariness of the semantics, a property that is reasonable to require from such a semantic description.

An obvious extension of this work would be to give a similar theory for the more useful version of CCS_L that allows parameterized recursion.

Acknowledgment

The author would like to thank the anonymous referees for many useful comments that led to a thorough revision of the paper.

References

[Abr87] S. Abramsky. Domain theory in logical form. *Annals of Pure and Applied Logic*, 51(01):1–78, 1991.

[Abr90] S. Abramsky. The lazy lambda calculus. In D. Turner, editor, *Research Topics in Functional Programming*, pages 65–117. Addison-Wesley, 1990.

[Abr91] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.

[AH92] L. Aceto and M. Hennessy. Termination, deadlock and divergence. *Journal of the ACM*, 39(1):147–187, January 1992.

[DZ82] J. W. de Bakker and J. I. Zucker. Processes and the denotational semantics of concurrency. *Information and Control*, 54(1/2):70-120, July/August 1982.

[FMS96] M. Fiore, E. Moggi and D. Sangiorgi. A Fully-Abstract Model for the π-calculus. In Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science, pages 43–54. IEEE Computer Society Press, 1996.

[GTWW77] J.A. Goguen, J.W. Thatcher, E.G. Wagner, and J.B. Wright. Initial algebra semantics and continuous algebras. *Journal of the ACM*, 24(1):68–95, 1977.

[Hen81] M. Hennessy. A term model for synchronous processes. *Information and Computation*, 51(1):58–75, 1981.

[Hen88a] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts, 1988.

[Hen94] M. Hennessy. A fully abstract denotational model for higher-order processes. *Information and Computation*, 112(1):55–95, July 1994.

[Hen00] M. Hennessy. A fully abstract denotational semantics for the pi-calculus. To appear in *Theoretical Computer Science.*

[HI93] M. Hennessy and A. Ingólfsdóttir. A theory of communicating processes with value-passing. *Information and Computation*, 107(2):202–236, 1993.

[HL93a] M. Hennessy and H. Lin. Proof systems for message-passing process algebras. In E. Best, editor, *Proceedings of CONCUR 93,* Hildesheim, volume 715 of *Lecture Notes in Computer Science*, pages 202–216. Springer-Verlag, 1993.

[HL93b] M. Hennessy and X. Liu. A modal logic for message passing processes. In C. Courcoubetis, editor, *Proceedings of the Fifth International Conference Computer Aided Verification,* Elounda, Greece, July 1993, volume 697 of *Lecture Notes in Computer Science*, pages 359–370. Springer-Verlag, 1993.

[Hoa78] C.A.R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.

[HP80] M. Hennessy and G.D. Plotkin. A term model for CCS. In P. Dembiński, editor, 9^{th} *Symposium on Mathematical Foundations of Computer Science*, volume 88 of *Lecture Notes in Computer Science*, pages 261–274. Springer-Verlag, 1980.

[Ing93] A. Ingólfsdóttir. Late and early semantics coincide for testing. *Theoretical Computer Science*, 146:341–349,1995.

[Ing94] A. Ingólfsdóttir. *Semantic Models for Communicating Process with Value-Passing.* PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, June 1994. Computer Science Report 8/94. Also available as Report R–94–2044, Department of Mathematics and Computer Science, Aalborg University.

[Mil80] R. Milner. *A Calculus of Communicating Systems*, volume 92 of *Lecture Notes in Computer Science*. Springer-Verlag, 1980.

[Mil83] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall International, Englewood Cliffs, 1989.

[Mil91] R. Milner. The polyadic π-calculus: a tutorial. Technical Report ECS–LFCS–91–180, LFCS, Department of Computer Science, University of Edinburgh, October 1991. In *Proceedings of the International Summer School on Logic and Algebra of Specification*, Marktoberdorf, August 1991.

[MM79] G. Milne and R. Milner. Concurrent processes and their syntax. *Journal of the ACM*, 26(2):302–321, 1979.

[MPW91] R. Milner, J. Parrow, and D. Walker. Modal logics for mobile processes. In J.C.M. Baeten and J.F. Groote, editors, *Proceedings CONCUR 91,* Amsterdam, volume 527 of *Lecture Notes in Computer Science*, pages 45–60. Springer-Verlag, 1991.

[MPW92] R. Milner, J. Parrow, and D. Walker. A calculus of mobile processes, Part I + II. *Information and Computation*, 100(1):1–77, 1992.

[Par81] D.M.R. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, 5^{th} *GI Conference*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Plo76] G.D. Plotkin. A powerdomain construction. *SIAM Journal on Computing*, 5:452–487, 1976.

[Plo81] G.D. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1981.

[Plo81] G.D. Plotkin. Lecture notes in domain theory, 1981. University of Edinburgh.

[San93] D. Sangiorgi. From π-calculus to higher-order π-calculus — and back. In *TAPSOFT '93*, volume 668 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Smy78] M. Smyth. Powerdomains. *Journal of Computer and System Sciences*, 16(1), 1978.

[Sta96] I. Stark. A Fully Abstract Domain Model for the π-Calculus. In Proceedings of the Eleventh Annual IEEE Symposium on Logic in Computer Science, pages 36–42. IEEE Computer Society Press, 1996.

[Tar55] A. Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5, 1955.

[Tho89] B. Thomsen. Plain CHOCS. Report DOC 89/4, Department of Computing, Imperial College, 1989.

[Win85] G. Winskel. On powerdomains and modality. *Theoretical Computer Science*, 36:127–137, 1985.

[Wal90] D.J. Walker. Bisimulation and divergence. *Information and Computation*, 85(2):202–241, 1990.

Recent BRICS Report Series Publications

RS-03-15 Anna Ingólfsdóttir. ***A Semantic Theory for Value–Passing Processes Based on the Late Approach*****. March 2003. 48 pp.**

RS-03-14 Mads Sig Ager, Dariusz Biernacki, Olivier Danvy, and Jan Midtgaard. ***From Interpreter to Compiler and Virtual Machine: A Functional Derivation*****. March 2003. 36 pp.**

RS-03-13 Mads Sig Ager, Dariusz Biernacki, Olivier Danvy, and Jan Midtgaard. ***A Functional Correspondence between Evaluators and Abstract Machines*****. March 2003. 28 pp.**

RS-03-12 Mircea-Dan Hernest and Ulrich Kohlenbach. ***A Complexity Analysis of Functional Interpretations*****. February 2003. 70 pp.**

RS-03-11 Mads Sig Ager, Olivier Danvy, and Henning Korsholm Rohde. ***Fast Partial Evaluation of Pattern Matching in Strings*****. February 2003. 14 pp. To appear in Leuschel, editor,** ***ACM SIGPLAN Workshop on Partial Evaluation and Semantics-Based Program Manipulation*****, PEPM '03 Proceedings, 2003.**

RS-03-10 Federico Crazzolara and Giuseppe Milicia. ***Wireless Authentication in χ-Spaces*****. February 2003. 20 pp.**

RS-03-9 Ivan B. Damgård and Gudmund Skovbjerg Frandsen. ***An Extended Quadratic Frobenius Primality Test with Average and Worst Case Error Estimates*****. February 2003. 53 pp.**

RS-03-8 Ivan B. Damgård and Gudmund Skovbjerg Frandsen. ***Efficient Algorithms for gcd and Cubic Residuosity in the Ring of Eisenstein Integers*****. February 2003. 11 pp.**

RS-03-7 Claus Brabrand, Michael I. Schwartzbach, and Mads Vanggaard. ***The METAFRONT System: Extensible Parsing and Transformation*****. February 2003. 24 pp.**

RS-03-6 Giuseppe Milicia and Vladimiro Sassone. ***Jeeg: Temporal Constraints for the Synchronization of Concurrent Objects*****. February 2003. 41 pp. Short version appears in Fox and Getov, editors,** ***Joint ACM-ISCOPE Conference on Java Grande*****, JGI '02 Proceedings, 2002, pages 212–221.**

RS-03-5 Aske Simon Christensen, Anders Møller, and Michael I. Schwartzbach. ***Precise Analysis of String Expressions*****. February 2003. 15 pp.**