BRICS
Basic Research in Computer Science

BRICS RS-95-22 A. Ingólfsdóttir: Value–Passing Processes, Late Approach, Part II

A Semantic Theory for Value–Passing Processes Late Approach

Part II: A Behavioural Semantics and Full Abstractness

Anna Ingólfsdóttir

BRICS Report Series RS-95-22
ISSN 0909-0878 April 1995

See back inner page for a list of recent publications in the BRICS Report Series. 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@daimi.aau.dk

BRICS publications are in general accessible through WWW and anonymous FTP:

`http://www.brics.aau.dk/BRICS/`
`ftp ftp.brics.aau.dk (cd pub/BRICS)`

Semantic Theory for Value–Passing Processes Late Approach

Part II: A Behavioural Semantics and Full Abstractness

Anna Ingólfsdóttir

BRICS*
Department of Mathematics and Computer Science
Aalborg University, Denmark

1 Introduction

This is the second of two companion papers on a semantic theory for communicating processes with values based on the late approach. In the first one, [Ing95], we explained the general idea of the late semantic approach. Furthermore we introduced a general syntax for value-passing process algebra based on the late approach and a general class of denotational models for these languages in the Scott-Strachey style. Then we defined a concrete language, CCS_L, which is an extension of the standard CCS with values according to the late approach. We also provided a denotational model for it, which is an instantiation of the general class. This model is a direct extension of the model given by Abramsky [Abr91] to model the pure calculus $SCCS$. Furthermore we gave an axiomatic semantics by means of a proof system based on inequations and proved its soundness and completeness with respect to the denotational semantics.

In this paper 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 developed in the companion paper [Ing95]. 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 given by the bottom element of the domain. 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].

One of the results in the pure case presented in [Abr91] is that the denotational model given in that reference is fully abstract with respect to the "finitely observable" part of the bisimulation preorder but not with respect to the bisimulation preorder which turns out to be too fine. Intuitively this is due to the

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

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 can not be said about the bisimulation preorder. In fact we need experiments of an infinite depth 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 can not 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 fails to be fully abstract with respect to the strong bisimulation preorder. In the same paper 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.

In both [Hen81] and [Abr91] the full abstractness of the respective denotational semantics with respect to $\precsim^F$ is 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.

As mentioned above the main aim of this paper is to give a bisimulation based behavioural semantics for our language CCS_L from [Ing95]. To reflect the late approach the operational semantics will be 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]. For this purpose we first introduce the notion of strong applicative prebisimulation and the corresponding strong applicative bisimulation preorder. Following the standard practice this preorder is obtained as the largest fixed point of a suitably defined monotonic functional. We show by an example that this 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 in the standard way by iterative application of the functional that induces the bisimulation preorder. This gives as a result a preorder which still is too fine to match the preorder induced by the denotational model. This will be shown by an example. Intuitively the reason for this is that we still need infinite experiments to decide the operational preorder, now because of an infinite breadth due to the possibility of an infinite number of values that have to be checked.

Then we give a suitable definition of the notion of the "finitary part" of the bisimulation preorder to meet the preorder induced by the denotational model. We recall that in [Ing95] we defined the so-called *compact* terms as the syntactically finite terms which only use a finite number of values in a non-trivial way. We also showed 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 finitary part, $\precapprox^F$, of the bisimulation preorder $\precapprox$ by

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

where c ranges over the set of syntactically compact terms. We also define yet another preorder, $\precapprox^f_\omega$, a coarser version of $\precapprox_\omega$ 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 $\precapprox^f_1$ 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 $\precapprox^f_n$ we test the constraints for the first n values only. (Here we would like to point out that this idea originally appears in [HP80].) It turns out that $\precapprox^f_\omega$ is the finitary part of $\precapprox$ in our new sense and that the model is fully abstract with respect to $\precapprox^f_\omega$. We will prove both these results in this paper using techniques which are similar to those used by Hennessy in the above mentioned reference [Hen81].

The structure of the paper is as follows: In Section 2 we give a short survey of the result from the companion paper [Ing95] needed in this study. The definition of the operational semantics and the notion of applicative bisimulation are the subject of Section 3. Section 4 is devoted to the analysis of the preorder and the definition of the value-finitary preorder $\precapprox^f_\omega$. In Section 5 we give a definition of the notion of finitary part of a relation and a finitary relation over processes. In the same section we prove that the preorder $\precapprox^f_\omega$ is finitary and that it coincides with the finitary part of the preorder $\precapprox$. Finally we prove the soundness and the completeness of the proof system with respect to the resulting preorder. The full abstractness of the denotational semantics for CCS_L, given in [Ing95], then follows from the soundness and the completeness of the proof system with respect to the denotational semantics. In Section 6 we give some concluding remarks.

2 Preliminaries

In this section we will give a brief review of the definitions, notation and proved results we need in this study from the companion paper [Ing95].

2.1 Syntax

First we 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. We do this by introducing 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. For motivation for these definitions we refer to the companion paper [Ing95].

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 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. $BExp$ 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 $[\![e]\!]\sigma$. Further we preassume an infinite set of process names, PN, ranged over by P, Q, etc. The set of (Σ, C)-*terms* is now given as the triple

$$T_{(\Sigma,C)} = (Proc_{(\Sigma,C)}, Fun_{(\Sigma,C)}, Pair_{(\Sigma,C)})$$

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

$$\begin{array}{ll} Proc_{(\Sigma,C)} : & p ::= op(\underline{p}), op \in \Sigma \mid c?.f \mid c!.\pi \mid \tau.p \mid be \rightarrow p, p' \\ Fun_{(\Sigma,C)} : & f ::= [x]p \\ Pairs_{(\Sigma,C)} : & \pi ::= (e, p) \end{array}$$

where we use the notation $\underline{p}$ to denote a vector of terms in $Proc_{(\Sigma,C)}$ of the appropriate length. If the process names in PN are added as primitives to the syntax for $T_{(\Sigma,C)}$, we write $T_{(\Sigma,C)}(PN)$ for the resulting triple of (Σ, C)-terms, and $T^{rec}_{(\Sigma,C)}(PN)$ if the recursive binding $rec\_.\_$ is also allowed.

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.

Prefixing by $[x]$ binds the data variable x and the recursion construct is a binding construct for process names. 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, $rec\ P.\_$. We assume a notion of substitution for both data variables and process names in terms defined in the usual way. For $f = [x]p$ and $v \in Val$ we use the convention $f(v) = ([x]p)(v) = p[v/x]$.

The language $CCS_L(PN) = (CCS^{proc}_L(PN), CCS^{fun}_L(PN), CCS^{pair}_L(PN))$ is obtained by instantiating the signature Σ by the standard operator of CCS. So we let Σ consist of the nullary operators NIL and Ω, the families of unary operators $\_ \setminus c, c \in C$ and $\_[R]$ where R is a finite permutation of the channel

$$
\begin{array}{lll}
CCS_L^{proc}(PN): & p ::= & NIL \,\big|\, \Omega \,\big|\, p[R] \,\big|\, p \setminus c \,\big|\, p+p \,\big|\, p|p \,\big|\, c?.f \,\big|\, c!.\pi \,\big|\, \tau.p \\
 & & \big|\, be \rightarrow p, p \,\big|\, P \,\big|\, recP.p \\
CCS_L^{fun}(PN): & f ::= & [x]p \\
CCS_L^{pair}(PN): & \pi ::= & (e,p)
\end{array}
$$

Figure 1: The Syntax for CCS_L

names and the binary operators $+$ and $|$. The syntax for $CCS_L(PN)$ is given in Figure 1. We let $CCS_L = (CCS_L^{proc}, CCS_L^{fun}, CCS_L^{pair})$ denote the closed terms in $CCS_L(PN)$. These will be referred to as processes, functions and pairs ranged over by cp, cf and $c\pi$.

2.2 Semantics

In the companion paper [Ing95] we gave two kinds of semantics to CCS_L: denotational semantics and an axiomatic semantics in terms of inequationally based proof system. We also showed the equivalence between them. The proof of the full abstractness of the behavioural semantics with respect to the behavioural semantics presented in this second paper does not rely on the details of the definition of the denotational model, but instead we use the properties of the proof system. Therefore we just assume the existence of the denotational model $\overline{ACT}$ and the related evaluation mapping but give a rather detailed description of the proof system. In particular we know from [Ing95] that the compact elements of the model may be denoted in the syntax by the so-called *syntactically compact terms*, $CoTerms(PN) = (CoProc(PN), CoFun(PN), CoPair(PN))$, which are defined below. Intuitively the syntactically compact terms are the recursion-free terms which only use a finite number of values in a nontrivial way. (Note that the number of channels used by the term is automatically finite.) We start by introducing some notation.

Notation 2.1 *Let $\underline{w}_n = (w_1, \ldots, w_n)$ and $\underline{p}_n = (p_1, \ldots, p_n)$ be vectors of values and processes respectively. We write $x : \underline{w}_n \longrightarrow \underline{p}_n$ for $x = w_1 \longrightarrow p_1, (x = w_2 \longrightarrow p_2, (\ldots x = w_n \longrightarrow p_n, \Omega)\ldots)$. (Intuitively $x : \underline{w_n} \longrightarrow \underline{p_n}$ stands for the function that maps w_i to p_i for $i = 1, \ldots, n$ and all the other values $w \in Val$ into Ω.) Further we let $\{\underline{w_n}\} = \{w_i | \underline{w_n} = (w_1, \ldots, w_n)\}$ and similarly for $\{\underline{p_n}\}$.*

Definition 2.2 [Syntactically Compact Terms] The set of syntactically compact terms is the triple

$$CoTerms(PN) = (CoProc(PN), CoFun(PN), CoPairs(PN))$$

where the sets $CoProc(PN)$, $CoFun(PN)$ and $CoPairs(PN)$ are the least sets satisfying:

1. $NIL, \Omega \in CoProc(PN)$ and $P \in CoProc(PN)$ for all $P \in PN$
2. $\overline{p} \in CoProc(PN)$ implies $op(\overline{p}) \in CoProc(PN)$, $op = |, +, \_\setminus, \_[R], \tau.\_$

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

Figure 2: Equations

3. $\pi \in CoPair(PN)$, $c \in C$ implies $c!\pi \in CoProc(PN)$

4. $f \in CoFun(PN)$ and $c \in CR$ implies $c?f \in CoProc(PN)$

5. $p \in CoProc(PN)$ and $e \in Exp$ implies $(e,p) \in CoPairs(PN)$

6. $\{\underline{p_n}\} \subseteq CoProc(PN)$, $\{\underline{w_n}\} \subseteq Val$ and $x \in Var$ implies $[x].x : \underline{w_n} \longrightarrow \underline{p_n} \in CoFun(PN)$.

We use the convention $CoTerms = CoTerms(\emptyset)$, $CoProc = CoProc(\emptyset)$, etc. and let them be ranged over by Cot, Cop, etc. We say that a term is *compact* if it belongs to $CoTerms(PN)$. $\square$

Note that $CoTerms = (CoProc, CoFun, CoPair) \subseteq CCS_L$ is closed under sub-terms. The proof system is based on the inequations in Figures 2–3. The inference rules, Figure 4, describe the structure of the model and its preorder and their interaction with the operators. 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

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 3: Interleaving Law

$(ref)\qquad p \sqsubseteq p$

$$(trans)\qquad \frac{p \sqsubseteq q,\ q \sqsubseteq r}{p \sqsubseteq r}$$

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

$$(pre)\qquad \frac{p \sqsubseteq q}{\mu.p \sqsubseteq \mu.q}$$

$$(rec)\qquad \frac{}{recP.p = p[recP.p/P]}$$

$$(inst)\qquad \frac{}{p\sigma \sqsubseteq q\sigma}\ \text{for every inequation } p \sqsubseteq q \text{ and closed instantiation } \sigma$$

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

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

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

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

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

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

Figure 4: The Proof System E_{rec}

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.

We refer to the whole system as E_{rec}, to the full system minus the ω-rule as $E_{rec}^{-\omega}$ and as E if both the ω-rule and the rule (rec) are omitted. The respective preorder are denoted by $\sqsubseteq_{E_{rec}}$, $\sqsubseteq_{E_{rec}^{-\omega}}$ and $\sqsubseteq_E$. The syntactically compact approximations used in the ω-rule of the proof system are defined as follows.

Definition 2.3 [Compact Approximations] The n-th compact approximation of a term is defined inductively by :

I. $i)\, p^{(0)} = \Omega$

$ii)$ 1. $P^{(n+1)} = P$

2. $(op(\underline{p}))^{(n+1)} = op(\underline{p}^{(n+1)})$

3. $(\mu.u)^{(n+1)} = \mu.u^{(n+1)}$

4. $(recP.p)^{(n+1)} = p^{(n+1)}[(recP.p)^{(n)}/P]$

5. $(be \longrightarrow p, q)^{(n+1)} = \begin{cases} p^{(n+1)} & \text{if } [\![be]\!] = T \\ q^{(n+1)} & \text{if } [\![be]\!] = F \end{cases}$

II. $([x]p)^{(n+1)} = [x](x \in V_{n+1} \longrightarrow p^{(n+1)}, \Omega)$

III. $((e,p))^{(n+1)} = \begin{cases} ([\![e]\!], p^{(n+1)}) & \text{if } [\![e]\!] \in V_{n+1} \\ ([\![e]\!], \Omega) & \text{otherwise} \end{cases}$

□

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

Theorem 2.4 *For all natural numbers n, $t \in CCS_L(PC)$ and $ct \in CCS_L$ and $\rho : PN \longrightarrow \overline{\mathrm{ACT}}$*

1. *$t^{(n)} \in CoTerms(PN)$, i.e. $t^{(n)}$ is a syntactically compact term.*

2. *$ct^{(n)} \sqsubseteq_{E_{rec}^{-\omega}} ct$.*

3. *$\overline{\mathrm{ACT}}[\![t]\!]\rho = \bigsqcup_n \overline{\mathrm{ACT}}[\![t^{(n)}]\!]\rho$.*

The soundness and completeness of the proof system E_{rec} with respect to the denotational semantics is stated in the following theorem, whose proof may be found in [Ing95].

Theorem 2.5 [Soundness and Completeness] *For all closed terms ct, cu in CCS_L we have*

$$ct \sqsubseteq_{E_{rec}} cu \textit{ if and only if } \overline{\mathrm{ACT}}[\![ct]\!] \sqsubseteq \overline{\mathrm{ACT}}[\![cu]\!],$$

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

In the theory to follow we need the following notion of Ω-normal forms and a corresponding normalization theorem.

Definition 2.6 **[Ω-normal form]** A compact term, $nt \in CoTerms$, is said to be in a Ω-normal form if the following hold:

1. If $nt = np \in CoProc$ then np has the form
$$nt = \sum_i \mu_i.nt_i\{+\Omega\}$$
where Ω is an optional summand and where nt_i is in an Ω-normal form. The empty sum is interpreted as NIL.

2. If $nt = (e, np) \in CoPairs$ then $e = v \in Val$ and np is in an Ω-normal form.

3. If $nt = [x]x : \underline{v_n} \longrightarrow \underline{np_n} \in Fun$ then np_i is in an Ω-normal form for $i \leq n$.

$\square$

Lemma 2.7 *For all $Cot \in CoTerms$ there is an Ω-normal form $n(Cot)$ such that $n(Cot) =_E Cot$.*

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

Corollary 2.8 *For all $Cot \in CoTerms$ and $cu \in CCS_L$*

$$Cot \sqsubseteq_{E_{rec}} cu \text{ implies } Cot \sqsubseteq_E cu^{(n)} \text{ for some } n$$

and therefore

$$Cot \sqsubseteq_{E_{rec}} cu \text{ iff } Cot \sqsubseteq_{E_{rec}^{-\omega}} cu.$$

Proof First we note that for $Cou, Cot \in CoTerms$

$$Cou \sqsubseteq_{E_{rec}} Cot \text{ iff } Cou \sqsubseteq_E Cot. \tag{1}$$

Now we proceed as follows

$$
\begin{array}{ll}
 & Cot \sqsubseteq_{E_{rec}} cu \\[1em]
\text{implies} & \overline{ACT}[\![Cot]\!] \sqsubseteq_{\overline{ACT}} \overline{ACT}[\![cu]\!] \\
 & \text{as } E_{rec} \text{ is sound wrt. } \overline{ACT} \\[1em]
\text{implies} & \exists n.\ \overline{ACT}[\![Cot]\!] \sqsubseteq_{\overline{ACT}} \overline{ACT}[\![cu^{(n)}]\!] \sqsubseteq_{\overline{ACT}} \overline{ACT}[\![cu]\!] \\
 & \text{by Thm. 2.4 as } \overline{ACT}[\![Cot]\!] \text{ is compact} \\[1em]
\text{implies} & \exists n.\ Cot \sqsubseteq_{E_{rec}} cu^{(n)} \sqsubseteq_{E_{rec}} cu \\
 & \text{as } E_{rec} \text{ is complete wrt. } \overline{ACT}
\end{array}
$$

$$\begin{array}{ll} \text{implies} & \exists n.\ Cot \sqsubseteq_E cu^{(n)} \sqsubseteq_{E_{rec}^{-\omega}} cu \\ & \quad \text{by (1) and Thm. 2.4.} \end{array}$$

□

In the proof for the full abstractness of the model with respect to our representant for the behavioural preorder we will use some standard techniques which are used to prove similar full abstractness results in the literature [Hen88, AH92, HI93]. This gives us some guidelines about properties our behavioural interpretation of the preorder induced by the model should satisfy. This is formulated in the following lemma:

Theorem 2.9 *Assume that $\preceq \subseteq CCS_L \times CCS_L$ satisfies the following conditions:*

1. *Finitariness: For all $ct, cu \in CCS_L$*

$$ct \preceq cu \text{ iff } \forall Cot.\ Cot \preceq ct \Rightarrow \ Cot \preceq cu.$$

2. *Partial soundness: The proof system $E_{rec}^{-\omega}$ is sound with respect to $\preceq$.*
3. *Partial completeness: For all $Cot \in CoTerms$ and $ct \in CCS_L$*

$$Cot \preceq ct \text{ implies } Cot \sqsubseteq_{E_{rec}} ct.$$

Then for all $ct, cu \in CCS_L$

$$ct \preceq cu \text{ if and only if } ct \sqsubseteq_{E_{rec}} cu \text{ if and only if } \overline{\mathrm{ACT}}[\![ct]\!] \sqsubseteq_{\overline{\mathrm{ACT}}} \overline{\mathrm{ACT}}[\![cu]\!].$$

Proof First we have:

$$\begin{array}{lll} & ct \preceq cu & \\ \text{iff} & \forall Cot.\ Cot \preceq ct \Rightarrow Cot \preceq cu & \text{by 1.} \\ \text{iff} & \forall Cot.\ Cot \sqsubseteq_{E_{rec}} ct \Rightarrow Cot \sqsubseteq_{E_{rec}} cu & \text{by 2., 3. and Cor. 2.8.} \end{array}$$

Now we proceed as follows: Assume $ct \preceq cu$ and therefore that

$$\forall Cot.\ Cot \sqsubseteq_{E_{rec}} ct \Rightarrow Cot \sqsubseteq_{E_{rec}} cu. \tag{2}$$

As $ct^{(n)} \sqsubseteq_{E_{rec}} ct$ and $ct^{(n)} \in CoTerms$ then (2) implies that $ct^{(n)} \sqsubseteq_{E_{rec}} cu$. As this holds for all n, the ω-rule implies that $ct \sqsubseteq_{E_{rec}} cu$.

Next assume that $ct \sqsubseteq_{E_{rec}} cu$. To prove that $ct \preceq cu$ it is sufficient to prove that (2) holds. So assume that $Cot \sqsubseteq_{E_{rec}^{-\omega}} ct$. Then, by transitivity of $\sqsubseteq_{E_{rec}}$, $Cot \sqsubseteq_{E_{rec}} cu$ and therefore we get that $ct \preceq cu$. □

3 Operational Semantics

The aim of this section is to define an operational semantics and a suitable notion of preorder to describe the behaviour of our language. 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 of the value and the resulting process. For further motivations of this approach we refer to the companion paper [Ing95].

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
- $Act = Act_{Con} \uplus Act_{Pairs} \uplus Act_{Fun}$ is a set of actions.
- $\longrightarrow$ is a transition relation

$$\begin{aligned}\longrightarrow \subseteq\ & (Con \times Act_{Con} \times Con) \cup \\ & (Con \times Act_{Pairs} \times (Val \times Con)) \cup \\ & (Con \times Act_{Fun} \times (Val \longrightarrow Con)).\end{aligned}$$

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

We refer to $States = Con \cup (Val \times Con) \times (Val \longrightarrow Con)$ as the set of possible 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 $\mathcal{P}(Con \times Con)$. In order to obtain this we have to extend our notion of relation over configurations. 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 type 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 this way is a monotonic endofunction over the complete lattice $(\mathcal{P}(Con \times Con), \subseteq)$. 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 CCS_L^{proc}, as generated by the syntax in Figure 2.1, $\longrightarrow$ to be the least transition relation closed under the rules of Figures 5-6 and the convergence predicate $\downarrow$ to be the least relation on CCS_L^{proc} satisfying the rules in Figure 7. As usual the divergence predicate $\uparrow$ is defined as the complement of $\downarrow$. The basic rule for input has the form

$$c?.[x]p \xrightarrow{c?} [x]p,$$

(input) $c?.cf \xrightarrow{c?} cf$

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

(tau) $\tau.cp \xrightarrow{\tau} cp$

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

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

$$\frac{cp \xrightarrow{\tau} cp'}{cp[R] \xrightarrow{\tau} cp'[R]}$$

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

$$\frac{cp \xrightarrow{c!} (v, cp')}{cp\backslash c' \xrightarrow{c!} (v, cp'\backslash c')} \quad , c \neq c'$$

$$\frac{cp \xrightarrow{\tau} cp'}{cp\backslash c' \xrightarrow{\tau} cp'\backslash c'}$$

Figure 5: Operational semantics for CCS_L: Part 1

$$\text{(choice)} \quad \frac{cp \xrightarrow{\mu} ct}{cp + cq \xrightarrow{\mu} ct}$$

$$\text{(par)} \quad \frac{cp \xrightarrow{c?} [x]p'}{cp \mid cq \xrightarrow{c?} [x](p' \mid cq)}$$

$$\frac{cp \xrightarrow{c!} (v, cp')}{cp \mid cq \xrightarrow{c!} (v, cp' \mid cq)}$$

$$\frac{cp \xrightarrow{\tau} cp'}{cp \mid cq \xrightarrow{\tau} cp' \mid cq}$$

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

$$\text{(cond)} \quad \frac{cp \xrightarrow{\mu} ct}{(be \longrightarrow cp, cq) \xrightarrow{\mu} ct} \quad , [\![be]\!] = T$$

$$\frac{cq \xrightarrow{\mu} ct}{(be \longrightarrow cp, cq) \xrightarrow{\mu} ct} \quad , [\![be]\!] = F$$

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

The
choice, par and com rules have symmetric counterparts.

Figure 6: Operational semantics for CCS_L: Part 2

$$\frac{}{NIL \downarrow} \quad \frac{cp \downarrow, cp' \downarrow}{cp + cp' \downarrow} \quad \frac{cp \downarrow, cp' \downarrow}{cp \mid cp' \downarrow} \quad \frac{cp \downarrow}{cp \backslash c \downarrow} \quad \frac{cp \downarrow}{cp[R] \downarrow} \quad \frac{p[\Omega/P] \downarrow}{rec\ P.p \downarrow}$$

$$\frac{[\![be]\!] = T, cp_1 \downarrow}{be \longrightarrow cp_1, cp_2 \downarrow} \quad \frac{[\![be]\!] = F, cp_2 \downarrow}{be \longrightarrow cp_1, cp_2 \downarrow}$$

Figure 7: The convergence predicate

the one for output is

$$c!.(v,q) \xrightarrow{c!} (v,q)$$

and that for communication express the fact that synchronization takes the form of functions application,

$$\frac{p \xrightarrow{c?} [x]p', q \xrightarrow{c!} (v,q')}{p|q \xrightarrow{\tau} p'[v/x]|q'}.$$

The rules in Figure 5-6 are consistent in the following sense:

Lemma 3.5 *For all $cp \in CCS_L^{proc}$*

1. $cp \xrightarrow{\tau} ct$ *implies* $ct \in CCS_L^{proc}$,
2. $cp \xrightarrow{c?} ct$ *implies* $ct \in CCS_L^{fun}$,
3. $cp \xrightarrow{c!} ct$ *implies* $ct \in CCS_L^{pair}$.

Proof

An easy induction on the length of the derivation of $cp \xrightarrow{\mu} ct$. □

The bisimulation preorder $\lessapprox$ defined on this $ALTS$ satisfies:

Theorem 3.6

1. $\lessapprox$ *is a pre-congruence with respect to the operators in* Σ.
2. (a) *For all* cp_1, cp_2 $cp_1 \lessapprox cp_2$ *implies* $\tau.cp_1 \lessapprox \tau.cp_2$.
   (b) *For all* $c \in Chan$ *and* $c\pi_1, c\pi_2$, $c\pi_1 \lessapprox c\pi_2$ *implies* $c!.c\pi_1 \lessapprox c!.c\pi_2$.
   (c) *For all* $c \in Chan$ *and* cf_1, cf_2, $cf_1 \lessapprox cf_2$ *implies* $c?.cf_1 \lessapprox c?.cf_2$.
3. (a) *For all* $p_1, p_2 \in CCS_L^{proc}(\emptyset)$ *with* $FVV(p_i) \subseteq \{x\}, i = 1,2$, *whenever* $p_1[v/x] \lessapprox p_2[v/x]$ *for every* $v \in Val$ *then* $[x]p_1 \lessapprox [x]p_2$.
   (b) *For all* cp_1, cp_2 *and* v, $cp_1 \lessapprox cp_2$ *implies* $(v, cp_1) \lessapprox (v, cp_2)$.

Proof

1. We have to prove that for any operator $op \in \Sigma$ following holds:

$$\underline{cp} \lessapprox \underline{cq} \quad \Rightarrow \quad op(\underline{cp}) \lessapprox op(\underline{cq}).$$

We will only prove the statement for the case $op = \_|\_$, leaving the remaining cases to the interested reader to check.

So assume $cp_1 \lessapprox cp_2$ and $cq_1 \lessapprox cq_2$. This means that there are sa-prebisimulations, $\mathcal{R}_p$ and $\mathcal{R}_q$ such that $(cp_1, cp_2) \in \mathcal{R}_p$ and $(cq_1, cq_2) \in \mathcal{R}_q$. We define

$$\mathcal{R}_p|\mathcal{R}_q = \{(cp_1'|cq_1', cp_2'|cq_2') \bigm| (cp_1', cp_2') \in \mathcal{R}_p, (cq_1', cq_2') \in \mathcal{R}_q\}.$$

As $(cp_1|cq_1, cp_2|cq_2) \in \mathcal{R}_p|\mathcal{R}_q$ it is sufficient to show that $\mathcal{R}_p|\mathcal{R}_q$ is a sa-prebisimulation. We proceed as follows:

(a) Assume that $(cp'_1|cq'_1, cp'_2|cq'_2) \in \mathcal{R}_p|\mathcal{R}_q$ and that $cp'_1|cq'_1 \xrightarrow{\mu} cr_1$. We have the following cases:

   i. $\mu = \tau$, $cp'_1 \xrightarrow{\tau} cp''_1$ and $cr_1 = cp''_1|cq'_1$. Then there is a cp''_2 where $cp'_2 \xrightarrow{\tau} cp''_2$ and $(cp''_1, cp''_2) \in \mathcal{R}_p$. This implies that $cp'_2|cq'_2 \xrightarrow{\tau} cp''_2|cq'_2$ where $(cp''_1|cq'_1, cp''_2|cq'_2) \in \mathcal{R}_p|\mathcal{R}_q$.

   ii. $\mu = c?$, $cp'_1 \xrightarrow{c?} [x]p''_1$ and $cr_1 = [x](p''_1|cq'_1)$. Now there is a $[y]p''_2$ where $cp'_2 \xrightarrow{c?} [y]p''_2$ 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 cq'_1 and cq'_2 do not contain free value-variables this implies that for all $v \in Val$

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

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

$$cp'_2|cq'_2 \xrightarrow{c?} [y](p''_2|cq'_2).$$

   iii. $\mu = c!$, $cp'_1 \xrightarrow{c!} (v, cp''_1)$ and $cr_1 = (v, cp''_1|cq'_1)$. Now $cp'_2 \xrightarrow{c!} (v, cp''_2)$ for some cp''_2 where $(cp''_1, cp''_2) \in \mathcal{R}_p$. Thus

$$(cp''_1|cq'_1, cp''_2|cq'_2) \in \mathcal{R}_p|\mathcal{R}_q$$

   and therefore

$$((v, cp''_1|cq'_1), (v, cp''_2|cq'_2)) \in \mathcal{R}_p|\mathcal{R}_q^{pair}.$$

   Furthermore $cp'_2|cq'_2 \xrightarrow{c!} (v, cp''_2|cq'_2)$.

   iv. The symmetrical cases $\mu = \tau, c?, c!$ where the transition comes from cq'_1 instead of cp'_1 may be treated in the same way.

   v. $\mu = \tau$, $cp'_1 \xrightarrow{c?} [x]p''_1$, $cq'_1 \xrightarrow{c!} (v, cq''_1)$ and $cr_1 = p''_1[v/x]|cq''_1$. Then $cp'_2 \xrightarrow{c?} [y]p''_2$ 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 $cq'_2 \xrightarrow{c!} (v', cq''_2)$ where $((v, cq''_1), (v', cq''_2) \in \mathcal{R}_q^{pair}$, i.e. where $v = v'$ and $(cq''_1, cq''_2) \in \mathcal{R}_q$. This implies that $(p''_1[v/x]|cq''_1, p''_2[v/y]|cq''_2) \in \mathcal{R}_p|\mathcal{R}_q$. Furthermore $cp'_2|cq'_2 \xrightarrow{\tau} p''_2[v/y]|cq''_2$.

   vi. The symmetrical case where $cp'_1 \xrightarrow{c!} (v, cp''_1)$ and $cq'_1 \xrightarrow{c?} [x]q''_1$ may be treated similarly.

(b) Next assume that $cp'_1|cq'_1 \downarrow$. This implies that $cp'_1 \downarrow$ and $cq'_1 \downarrow$ and therefore that $cp'_2 \downarrow$ and $cq_2 \downarrow$. This in turn implies that $cp'_2|cq'_2 \downarrow$. Now assume that $cp'_1|cq'_1 \downarrow$, $cp'_2|cq'_2 \downarrow$ and $cp'_2|cq'_2 \xrightarrow{\mu} cr_2$. In the same way as in (a) we may show that $cp'_1|cq'_1 \xrightarrow{\mu} cr_1$ for some cr_1 such that $(cr_1, cr_2) \in \mathcal{R}_p|\mathcal{R}_q$.

2. Here we will only prove the last case, i.e. that $cf_1 \lessapprox cf_2$ implies $c?.cf_1 \lessapprox c?.cf_2$. The remaining cases may be proved similarly. So assume that $cf_1 = [x]p$ and $cf_2 = [y]q$ and that $[x]p \lessapprox [y]q$. This implies that

$([x]p,[y]q) \in \mathcal{R}^{fun}$ for some sa-prebisimulation $\mathcal{R}$. We define $c?.\mathcal{R} = \mathcal{R} \cup \{(c?.f, c?.g) \bigm| (f,g) \in \mathcal{R}^{fun}\}$. 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 $Proc$ to $Val \longrightarrow Proc$ and $Val \times Proc$.

$\square$

4 Analysis of the Preorders

The subject of this section is to give an operational characterization of the denotational semantics given in the companion paper [Ing95]. First we show by an example, Example 4.1, that 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 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 4.1 *As we only need an example from the pure calculus we use the notation* $\overline{a} = c?x$, $a.p = c!.(v_1, p)$ *and* $\_ \setminus a = \_ \setminus c$. *Let*

$$a^\omega = recY.a.Y \text{ and } p = [recX.(a^\omega + X)|\overline{a}] \setminus a.$$

Then the first unfolding of p *is*

$$p_1 = [(a^\omega + (recX.(a^\omega + X)|\overline{a}))|\overline{a}] \setminus a,$$

and the $n+1$*-th one*

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

The reader may convince himself that the behaviour of 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,* $\overline{ACT}[\![p + recP.\tau.P]\!] = \overline{ACT}[\![p]\!]$. *On the other hand the left hand side has the transition* $p + recP.\tau.P \xrightarrow{\tau} recP.\tau.P$ *where* $recP.\tau.P$ *can perform an infinite sequence of* τ*-moves. This move can therefore never be matched by the right hand side* p. *This implies that* $p + recP.\tau.P \not\sqsubsetapprox p$.

Obviously Example 4.1 rules out the possibility that the behavioural preorder $\sqsubsetapprox$ characterizes the preorder of the model. Our second suggestion for a behavioural characterization of the model is the weaker version of $\sqsubsetapprox$, the *strong applicative* ω*-bisimulation preorder*, derived from the function $\mathcal{F}$ by iterated application.

Definition 4.2 [Strong Applicative ω-Prebisimulation]
The kth sa-prebisimulation $\sqsubseteq_k$ is defined inductively by:

1. $\sqsubseteq_0 = Con \times Con$,
2. $\sqsubseteq_{n+1} = \mathcal{F}(\sqsubseteq_n)$.

The sa-ω-prebisimulation $\sqsubseteq_\omega$ is defined as $\sqsubseteq_\omega = \bigcap_k \sqsubseteq_k$ and $\sim_\omega = \sqsubseteq_\omega \cap \sqsubseteq_\omega^{-1}$. $\square$

For all k we have that $\sqsubseteq \subseteq \sqsubseteq_{k+1} \subseteq \sqsubseteq_k$ which implies that $\sqsubseteq \subseteq \sqsubseteq_\omega$.

Again this preorder is too fine to match the preorder from the model as the following example shows:

Example 4.3 *Let $AT = \langle Con, Act, \longrightarrow, \downarrow \rangle$ be an applicative transition system and the process p be given by the derivation graph described by the infinite sum $\sum_n c?.[x]x \leq n \longrightarrow NIL, \Omega$[1]. Now let*

$$q = p + c?.[x]NIL.$$

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

Intuitively the reason for why $\sqsubseteq_\omega$ is too fine for processes with values is that the values give rise to a new kind of infinity. We recall from [Ing95] 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 4.4 [Value-Finitary Strong Applicative ω-Prebisimulation]

Let $AT = \langle Con, Val, Act, \longrightarrow, \downarrow \rangle$ be an applicative 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 $\sqsubseteq_n^f$ is defined by:

1. $\sqsubseteq_0^f = Con \times Con$,
2. $\sqsubseteq_{n+1}^f = (\mathcal{F}(\sqsubseteq_n^f))|_{V_{n+1}}$.

[1] Whether this process can be expressed in the syntax of CCS_L or a similar language is an open question.

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

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

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 $\underset{\approx}{\sqsubset}{}^f_n \subseteq \underset{\approx}{\sqsubset}{}^f_{n+1}$ for all n. We also note that the only difference between this definition and Definition 4.2 is the restriction on the values in the definition of $\underset{\approx}{\sqsubset}{}^f_{n+1}$. Obviously $\underset{\approx}{\sqsubset}_n \subseteq \underset{\approx}{\sqsubset}{}^f_n$ for all n which implies $\underset{\approx}{\sqsubset} \subseteq \underset{\approx}{\sqsubset}_\omega \subseteq \underset{\approx}{\sqsubset}{}^f_\omega$. It is easy to prove that $\underset{\approx}{\sqsubset}{}^f_\omega$ actually is a preorder and has all the properties stated in Theorem 3.6. The proof for this is straightforward and is left to the reader. Now let us have a further look at our previous example, Example 4.3.

Example 4.5 *Let p and q be defined as in Example 4.3. Obviously $p \underset{\approx}{\sqsubset} q$ and therefore $p \underset{\approx}{\sqsubset}{}^f_\omega q$. We have also shown that $q \not\underset{\approx}{\sqsubset}_\omega p$ and thereby $q \not\underset{\approx}{\sqsubset} p$. On the other hand one may show that $q \underset{\approx}{\sqsubset}{}^f_\omega p$ by showing that $q \underset{\approx}{\sqsubset}{}^f_n p$ for all n by induction.*

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

Lemma 4.6 $\underset{\approx}{\sqsubset} \subseteq \underset{\approx}{\sqsubset}_\omega \subseteq \underset{\approx}{\sqsubset}{}^f_\omega$ *but* $\underset{\approx}{\sqsubset}{}^f_\omega \not\subseteq \underset{\approx}{\sqsubset}_\omega \not\subseteq \underset{\approx}{\sqsubset}$.

5 The Full Abstractness

In this last section we will prove the full abstractness of the model with respect to the behavioural preorder $\underset{\approx}{\sqsubset}{}^f_\omega$. As we explained in Section 2 the proof may be reduced to proving the following three properties: the finitariness of the preorder, the soundness of the proof system $E^{-\omega}_{rec}$ and the partial completeness for the proof system E_{rec} with respect to $\underset{\approx}{\sqsubset}{}^f_\omega$.

5.1 The Finitary Part of the Preorders

In this section we will define a suitable notion of a "finitary part" of a relation over processes and that of a "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 recursion-free terms. We will then show that the preorder $\underset{\approx}{\sqsubset}{}^f_\omega$ is the finitary part of the preorders $\underset{\approx}{\sqsubset}$ and $\underset{\approx}{\sqsubset}{}^f_\omega$ and therefore that $\underset{\approx}{\sqsubset}{}^f_\omega$ is finitary in our sense. We start by defining the finitary part of a relation over CCS_L.

Definition 5.1 For any relation $\mathcal{R}$ over CCS_L we define the finitary part of $\mathcal{R}$, $\mathcal{R}^F$, by

$$ct\mathcal{R}^F cu \text{ iff for all } Cot \in CoTerms, Cot\mathcal{R}ct \text{ implies } Cot\mathcal{R}cu.$$

$\mathcal{R}$ is finitary if $\mathcal{R} = \mathcal{R}^F$. $\Box$

Following [Hen81] next we will prove that on $CoTerms \times CCS_L$ the preorders, $\sqsubseteq$ and $\sqsubseteq^f_\omega$, coincide. Consequently they both have the same finitary part. To show this we need a measure on $CoTerms$ that both measures the structural depth of the term and the number of values it uses. We give the following definitions.

Definition 5.2 For syntactically finite terms, 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 $cd \stackrel{\mu}{\longrightarrow} ct$ then $sd(ct) \leq sd(cd)-1$. 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 5.3 The support of the term $Cot \in CoTerms$, $Supp(Cot)$, is defined by structural recursion as:

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

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

Now we prove the following.

Proposition 5.4 *For all $Cot \in CoTerms$ and $ct \in CCS_L$,*

$$Cot \sqsubseteq^f_\omega ct \text{ if and only if } Cot \sqsubseteq ct.$$

and therefore

$$(\sqsubseteq^f_\omega)^F = \sqsubseteq^F .$$

Proof As the "if" part is already known it is sufficient to prove the "only if" part. We do this by proving the following stronger result.

For all $Cot \in CoTerms$ and $ct \in CCS_L$

$$Cot \sqsubseteq^f_m ct \Rightarrow Cot \sqsubseteq ct$$

for all m where $m \geq m(Cot) = sd(Cot) + vd(Cot)$.

The proof of this statement proceeds by induction on $m(Cot)$.

$m(Cot) = 0$: We have two cases: $Cot = \Omega$, which is trivial, and $Cot = NIL$ which we will have a further look at. Now, as $NIL \downarrow$, the definition of $\sqsubseteq^f_m$ implies that $ct \downarrow$. Furthermore as $NIL \stackrel{\mu}{\not\longrightarrow}$ for all μ this is also true for ct. This proves that $Cot = NIL \sqsubseteq ct$.

$m(Cot) = k+1$: Assume we have proved the result for all Cot' with $m(Cot') \leq k$ and we will prove that it is true for Cot where $m(Cot) = k+1$. We assume that $Cot \sqsubseteq^f_m ct$, where $m \geq m(Cot) = k+1$. As $\mathcal{F}(\sqsubseteq) = \sqsubseteq$ it is sufficient to show that $Cot\mathcal{F}(\sqsubseteq)ct$. We proceed by case analysis on the structure of Cot.

$Cot \in CoProc$:

1. Assume $Cot \xrightarrow{\mu} cu$. By definition of $\sqsubseteq^f_m$, $ct \xrightarrow{\mu} cu'$ for some cu' such that $cu \sqsubseteq^f_{m-1} cu'$. Also, by definition of $CoTerms$, $cu \in CoTerms$. Now $vd(cu) \leq vd(Cot)$ and $sd(cu) \leq sd(Cot) - 1$. Thus $m - 1 \geq k \geq m(cu)$ and by the induction $cu \sqsubseteq cu'$.
2. Now assume $Cot \downarrow$, by definition of the preorder $\sqsubseteq^f_\omega$ also $ct \downarrow$. Furthermore assume that $Cot \downarrow$, $ct \downarrow$ and that $ct \xrightarrow{\mu} cu'$. Then $Cot \xrightarrow{\mu} cu$ for some cu such that $cu \sqsubseteq^f_{m-1} cu'$. In a similar way as before the induction implies $cu \sqsubseteq cu'$, which completes the proof in this case.

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

$Cot \in CoPair$: Now $Cot = Co\pi = (v', Cot')$ and $ct = c\pi = (v'', ct'')$. By the definition of the preorder and the assumption on m, $v' = v'' \in V_m$

and $Cot' \lessapprox^f_m ct$. As before $m > k \geq m(Cot')$ and the result follows from the induction.

□

We will now show that the preorder $\lessapprox^f_\omega$ is finitary and therefore that it is the finitary part of $\lessapprox$. 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, t, $Sort(t)$, as the set of channel names it uses.

Definition 5.5 The sort of $ct \in CCS_L$ is given by

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

□

Note that, because of our restriction to finite renamings, $Sort(ct)$ is finite for all ct [Abr91, AH92].

Definition 5.6 [Compact Projections] We define the n-th projection of ct on $CoTerms$ inductively as follows:

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

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. □

The syntactically compact projections have the following properties:

Lemma 5.7 *For all ct and all n,*

1. $ct^{[n]} \in CoTerms$,

2. $ct^{[n]} \sim^f_n ct$.

Proof

1. A simple induction on n, using a case analysis on the structure of ct for the inductive step.

2. First we prove $ct \precsim_n^f ct^{[n]}$ by induction on n.

   $n = 0$: Trivial.

   $n = k+1$: Let us assume that $ct \precsim_k^f ct^{[k]}$. We have to prove that $ct \precsim_{k+1}^f ct^{[k+1]}$. We proceed by a case analysis on the structure of ct.

   $ct = cp \in CCS_L^{proc}$:

   (a) Assume $cp \xrightarrow{\mu} cu$. By the definition of $cp^{[i]}$, $cp^{[k+1]} \xrightarrow{\mu} cu^{[k]}$. From the induction we get that $cu \precsim_k^f cu^{[k]}$ and the first clause of the definition for $\precsim_{k+1}^f$ is met.

   (b) First we note that $cp \downarrow$ if and only if $cp^{[i]} \downarrow$ for all i. Thus $cp \downarrow$ implies $cp^{[k+1]} \downarrow$. Furthermore assume that $cp \downarrow$, $cp^{[k+1]} \downarrow$ and that $cp^{[k+1]} \xrightarrow{\mu} cu'$. Then $cp \xrightarrow{\mu} cu$ such that $cu^{[k]} = cu'$. Again, by the induction, $cu \precsim_k^f cu'$.

   $ct \in CCS_L^{pair}s, CCS_L^{fun}$: Follows easily from the previous case.

   It remains to prove that $ct^{[n]} \precsim_n^f ct$. The proof of this fact is similar to the previous one and is left to the reader.

□

The following results investigate the relationship between a term, ct, and its syntactically compact projections in more detail.

Lemma 5.8

1. $ct^{[0]} \precsim_\omega^f ct^{[1]} \precsim_\omega^f \cdots \precsim_\omega^f ct^{[n]} \precsim_\omega^f \cdots \precsim_\omega^f ct$

2. *If* $ct^{[0]} \precsim_\omega^f ct^{[1]} \precsim_\omega^f \cdots \precsim_\omega^f ct^{[n]} \precsim_\omega^f \cdots \precsim_\omega^f cu$ *then* $ct \precsim_\omega^f cu$, *i.e.* ct *is a minimal upper bound* [2] *of the chain with respect to* $\precsim_\omega^f$.

3. *The term* ct *is a minimal upper bound for the set* $App(ct) = \{Cot \in CoTerms | Cot \precsim_\omega^f ct\}$ *with respect to* $\precsim_\omega^f$.

Proof

1. We first prove that for all n

$$ct^{[n]} \precsim_\omega^f ct^{[n+1]}.$$

   In order to do that we prove a slightly stronger result:

$$\forall m \geq n.\, ct^{[n]} \precsim_m^f ct^{[n+1]}.$$

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

We prove this by induction on n. The base case, $n = 0$, is immediate as $ct^{[0]} \sqsubseteq^f_m ct^{[1]}$ for all m is trivial. So assume

$$ct^{[k]} \sqsubseteq^f_m ct^{[k+1]} \text{ for } m \geq k$$

and we will prove that

$$ct^{[k+1]} \sqsubseteq^f_{m+1} ct^{[k+2]} \text{ for } m \geq k.$$

We proceed by a case analysis on the form of ct.

$ct = cp \in CCS^{proc}_L$: Assume $cp^{[k+1]} \xrightarrow{\mu} ct$, then $cp \xrightarrow{\mu} cu$ for some cu such that $cu^{[k]} = ct$. Also $cp^{[k+2]} \xrightarrow{\mu} cu^{[k+1]}$ and by the induction , as $m \geq k$, $cu^{[k]} \sqsubseteq^f_m cu^{[k+1]}$. Thus the first condition of the definition of the preorder $\sqsubseteq^f_{m+1}$ is satisfied. We now note that $cp \downarrow$ if and only if $cp^{[i]} \downarrow$ for all i and the second condition of the definition can be met in a similar way to the first one.

$ct = [x]p \in CCS^{fun}_L$: By definition

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

We have to prove that

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

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

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

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

$ct = (v, cp) \in CCS^{pair}_L$: Similar.

Next we prove $ct^{[n]} \sqsubseteq^f_\omega ct$ for all n. We know from Lemma 5.7 that $ct^{[k]} \sqsubseteq^f_k ct$ for all k. Furthermore for any $m \geq n$

$$ct^{[n]} \sqsubseteq^f_\omega ct^{[m]} \sqsubseteq^f_m ct.$$

Thus $ct^{[n]} \sqsubseteq^f_m ct$ for all $m \geq n$ which proves the statement.

2. To prove that ct is a minimal upper bound of the chain assume

$$ct^{[0]} \sqsubseteq^f_\omega ct^{[1]} \sqsubseteq^f_\omega \cdots ct^{[n]} \sqsubseteq^f_\omega \cdots \sqsubseteq^f_\omega cu.$$

As $ct \sim_n ct^{[n]}$ this implies $ct \sqsubseteq^f_n cu$ for all n and therefore $ct \sqsubseteq^f_\omega cu$.

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

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

Theorem 5.9

1. $\precsim^f_\omega = (\precsim^f_\omega)^F$

2. *The preorder* $\precsim^f_\omega$ *is the finitary part of* $\precsim$, *i.e.* $\precsim^F = \precsim^f_\omega$.

Proof

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

$$\forall Cot \in CoTerms.\ Cot \precsim^f_\omega ct \text{ implies } Cot \precsim^f_\omega cu.$$

This is equivalent to saying that $App(ct) \subseteq App(cu)$ and the result follows from Lemma 5.8.

2. By Proposition 5.4, $\precsim^F = (\precsim^f_\omega)^F$ and the result follows from part 1. of this theorem.

□

5.2 The Partial Completeness and The Full Abstractness

This last 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 $\precsim^f_\omega$. 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 is the content of the following Lemma.

Lemma 5.10 (Partial Soundness) *The proof system* $E^{-\omega}_{rec}$ *is sound with respect to the behavioural preorders* $\precsim$ *and* $\precsim^f_\omega$.

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

$$ct \sqsubseteq_{E^{-\omega}_{rec}} cu \text{ implies } ct \precsim cu$$

by induction on the depth of the proof tree for $ct \sqsubseteq_{E^{-\omega}_{rec}} cu$. The soundness of $E^{-\omega}_{rec}$ with respect to $\precsim^f_\omega$ follows from this as $\precsim \subseteq \precsim^f_\omega$. 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 $\precsim$ as shown by Example 4.1. Furthermore proving the soundness of the ω-rule for $\precsim^f_\omega$ directly is notationally quite complicated.

Next we prove the mentioned partial completeness result, i.e. that for all Cot and ct,

$$Cot \sqsubseteq_{\approx} ct \Rightarrow Cot \sqsubseteq_{E_{rec}} ct$$

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 5.11 A process term is said to be in a *head normal form* if it has the form $\sum_i \mu_i t_i$.

□

Lemma 5.12 *If $cp \downarrow$ then there is a head normal form, $hnf(cp)$, such that $cp =_{E_{rec}^{-\omega}} hnf(cp)$.*

Proof We prove the Lemma by induction on the length of the derivation of $cp \downarrow$ which we refer to as n. We proceed by a case analysis on the structure of cp.

$n = 1$: We have the two cases: $cp = NIL$ and $cp = \mu.t$ which both are trivial.

$n = m + 1$: We proceed by structural induction on cp.

$cp = NIL, \mu.t$: Already proven.

$cp = \Omega$: Vacuous.

$cp = cp_1 + cp_2, cp_1 \setminus c, cp_1[R], be \longrightarrow cp_1, cp_2$: Follows from the induction and a simple use of the proof system.

$cp = cp_1 | cp_2$: By induction cp_1 and cp_2 have head normal forms h_1 and h_2. If either h_1 or h_2 is NIL, the result follows from Equation (NILpar) in Figure 2. Otherwise assume

$$h_1 = \sum_i \mu_i.t_i \text{ and } h_2 = \sum_i \gamma_i.u_i.$$

By substitutivity and the interleaving law in Figure 3

$$cp =_{E_{rec}^{-\omega}} cp_1 | cp_2 =_{E_{rec}^{-\omega}} h_1 | h_2 =_{E_{rec}^{-\omega}}$$

$$INTL(h_1, h_2) + COMM(h_1, h_2)$$

where each of the summands is in a Head normal form.

$cp = recP.q$: Then $cp \downarrow$ because $q[recP.q/P] \downarrow$. Thus by the induction $q[recP.q/P]$ has a head normal form and the result follows from (rec) in Figure 4.

□

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

Notation 5.13 *Let $p, q \in Proc$ and $t, u \in Terms$. 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 $cp \xrightarrow{\mu} cp'$ then $cp|cq \xrightarrow{\mu} abs(cp'|cq)$ and $cq|cp \xrightarrow{\mu} abs(cq|cp')$. Furthermore if $cp \xrightarrow{c!} \pi$ and $cq \xrightarrow{c?} f$ then $cp|cq \xrightarrow{\tau} app(\pi|f)$ and $cq|cp \xrightarrow{\tau} app(f|\pi)$. We use this notation to formulate the following lemma:

Lemma 5.14 *For all closed terms cp, cq and ct*

1. $(cp + \mu.ct)|cq =_{E_{rec}} (cp + \mu.ct)|cq + \mu.abs(ct|cq)$
2. $cq|(cp + \mu.ct) =_{E_{rec}} cq|(cp + \mu.ct) + \mu.abs(cq|ct)$
3. $(cp + a.ct)|(cq + \overline{a}.cu) =_{E_{rec}} (cp + a.ct)|(cq + \overline{a}.cu) + \tau.app(ct|cu)$.

Proof We only prove the first statement as the second one follows by the commutativity of $|$ and substitutivity and the third is similar and is left to the reader. First assume that cp and cq are syntactically compact. By Lemma 2.7 we may assume that they are in Ω-normal forms; $cp = \sum_i \mu_i.t_i + \{\Omega|cp\uparrow\}$ and $cq = \sum_j \gamma_j.u_j + \{\Omega|cq\uparrow\}$. The result now follows as an easy consequence of the interleaving law in Figure 3. Next assume that cp, cq and ct are any terms. It is easy to see that

$$(abs(ct|cu))^{(n)} = abs(ct^{(n)}|cu^{(n)})$$

for all n. Therefore we have that:

$$[(cp + \mu.ct)|cq]^{(n)} =$$

$$(cp^{(n)} + \mu.ct^{(n)})|cq^{(n)} =_{E_{rec}}$$

$$(cp^{(n)} + \mu.ct^{(n)})|cq^{(n)} + \mu.abs(ct^{(n)}|cq^{(n)}) =_{E_{rec}}$$

$$[(cp + \mu.ct)|cq + \mu.abs(ct|cq)]^{(n)}$$

for all n and the result follows from the ω-rule. $\square$

Now we can prove the following useful property:

Proposition 5.15 *For all cp, ct and μ we have that $cp \xrightarrow{\mu} ct$ implies $cp =_{E_{rec}} cp + \mu.ct$.*

Proof The proof is by induction on the length of the derivation of $cp \xrightarrow{\mu} ct$. We proceed by a case analysis on the structure of cp. We only examine two cases, leaving the remaining ones to the reader.

$cp = recP.u$: Now $cp \xrightarrow{\mu} ct$ because $u[recP.u/P] \xrightarrow{\mu} ct$. By induction $u[recP.u/P] =_{E_{rec}} u[recP.u/P] + \mu.ct$ and by (rec) $cp =_{E_{rec}} cp + \mu.ct$.

$cp = cp_1|cp_2$: We have three cases:

1. $cp_1 \xrightarrow{\mu} ct_1$ and $ct = abs(ct_1|cp_2)$: By induction

$$cp_1 =_{E_{rec}} cp_1 + \mu.ct_1.$$

By the substitutivity and the first statement of Lemma 5.14, we have that

$$\begin{aligned} cp_1|cp_2 &=_{E_{rec}} (cp_1 + \mu.ct_1)|cp_2 \\ &=_{E_{rec}} (cp_1 + \mu.ct_1)|cp_2 + \mu.abs(ct_1|cp_2) \\ &=_{E_{rec}} cp_1|cp_2 + \mu.abs(ct_1|cp_2) \end{aligned}$$

$$= cp + \mu.ct.$$

2. $cp_2 \xrightarrow{\mu} ct_2$ and $ct = abs(cp_1|ct_2)$: This can be proved in the same way as the previous case by using statement 2 of Lemma 5.14 instead of Lemma 5.14(1).

3. $cp_1 \xrightarrow{a} ct_1$, $cp_2 \xrightarrow{\overline{a}} ct_2$, $\mu = \tau$ and $ct = app(ct_1|ct_2)$: By induction

$$cp_1 =_{E_{rec}} cp_1 + a.ct_1$$

and

$$cp_2 =_{E_{rec}} cp_2 + \overline{a}.ct_2.$$

By statement 3 of Lemma 5.14 and substitutivity, we then have that

$$\begin{aligned} cp_1|cp_2 &=_{E_{rec}} (cp_1 + a.ct_1)|(cp_2 + \overline{a}.ct_2) \\ &=_{E_{rec}} (cp_1 + a.ct_1)|(cp_2 + \overline{a}.ct_2) + \tau.app(ct_1|ct_2) \\ &=_{E_{rec}} cp_1|cp_2 + \tau.app(ct_1|ct_2) \end{aligned}$$

$$= cp + \mu.ct.$$

□

In the following we will state and prove the promised partial completeness result for the proof system E_{rec}.

Theorem 5.16 *For all compact terms Cot and all closed terms ct*

$$Cot \sqsubseteq^{f}_{\omega} ct \qquad \text{if and only if}$$

$$Cot \sqsubseteq_{E_{rec}} ct$$

Proof By Proposition 5.4 it is sufficient to prove the statement with $\sqsubseteq^{f}_{\omega}$ replaced by $\sqsubseteq$ which we do as follows:.

$Cot \sqsubseteq ct$ implies $Cot \sqsubseteq_{E_{rec}} ct$: It is sufficient to prove the result for Cot in Ω-normal form. The general result follows from the normalization result, Lemma 2.7, and the soundness of E with respect to $\sqsubseteq$. We proceed by a case analysis on the form of Cot but only give the details of the case where $Cot = Cotp \in CoProc$. In this case $ct = cp \in CCS^{proc}_{L}$.

So assume $np \sqsubseteq cp$ where np is an Ω-normal form and we will prove that $np \sqsubseteq_{E_{rec}} cp$. The proof proceeds by induction on $sd(np)$, the structural depth of np defined in Definition 5.2. So assume the theorem is true for all np' with $sd(np') \leq k$ and that $sd(np) = k+1$. We proceed by a case analysis on the form of np.

$np = NIL + \Omega$: Then $np =_{E_{rec}} \Omega \sqsubseteq_{E_{rec}} cp$.

$np = NIL$: $NIL \sqsubseteq cp$ implies $cp \downarrow$. Thus cp has a head normal form $h(cp)$ with $h(cp) =_{E^{-\omega}_{rec}} cp$. As $NIL \not\xrightarrow{\mu}$ then $h(cp) \not\xrightarrow{\mu}$ for all μ which implies that $h(cp) = NIL$. Therefore $np =_{E_{rec}} h(cp) =_{E_{rec}} cp$.

$np = \sum_i \mu_i.cp_i\{+\Omega\}$: We prove this case in three steps.

1. $cp + np \sqsubseteq_{E_{rec}} cp$: Assume $np \xrightarrow{\mu} cp'$ then $\mu = \mu_i$ and $cp' = cp_i$ for some i. As $np \sqsubseteq cp$ this implies that $cp \xrightarrow{\mu_i} cq_i$ where $cp_i \sqsubseteq cq_i$. By applying induction we have that $cp_i \sqsubseteq_{E_{rec}} cq_i$ and, by substitutivity, that $\mu_i.cp_i \sqsubseteq_{E_{rec}} \mu_i.cq_i$. Thus by substitutivity and Proposition 5.15

$$cp + \mu_i.cp_i \sqsubseteq_{E_{rec}} cp + \mu_i.cq_i =_{E_{rec}} cp.$$

   Repeated use of this result, substitutivity and transitivity implies $cp + np \sqsubseteq_{E_{rec}} cp$.
2. $np \sqsubseteq_{E_{rec}} cp + np$: If Ω is a summand of np then $np \sqsubseteq_{E_{rec}} np + \Omega \sqsubseteq_{E_{rec}} np + cp$. So assume that $np \downarrow$. As $np \sqsubseteq cp$ this implies $cp \downarrow$ and therefore that cp has a head normal form $cp =_{E^{-\omega}_{rec}} h(cp) = \sum_j \gamma_j.cq_j$. As the proof system $E^{-\omega}_{rec}$ is sound with respect to $\sqsubseteq$ then $np \sqsubseteq h(cp)$. Thus $cp \xrightarrow{\gamma_j} cq_j$ implies that $\gamma_j = \mu_{i_j}$ for some i_J and that $np \xrightarrow{\mu_{i_j}} cp_{i_j}$ for some cp_{i_j} such that $cp_{i_j} \sqsubseteq cq_j$. Now by proceeding in a similar way as in the previous case we get that

$$np = np + \sum_j \mu_{i_j}.cp_{i_j} \sqsubseteq_{E_{rec}} np + \sum_j \gamma_j.cq_j \sqsubseteq_{E_{rec}} np + cp.$$

3. Finally *1.* and *2.* imply $np \sqsubseteq_{E_{rec}} cp$.

$Cot \sqsubseteq_{E_{rec}} cp$ implies $Cot \precsim^{f}_{\omega} cp$: By Corollary 2.8 we get that $Cot \sqsubseteq_{E^{-\omega}_{rec}} cp$ and the result follows from the soundness of $E^{-\omega}_{rec}$ with respect to the preorder $\precsim^{f}_{\omega}$, stated in Lemma 5.10.

□

We are now ready to prove the main result of this section, namely the full abstractness of the denotational semantics with respect to the finitary behavioural semantics based on the preorder $\precsim^{f}_{\omega}$. This is the content of the following theorem.

Theorem 5.17 (Full Abstractness) *For all closed terms, ct and cu, in CCS_L*

$$ct \precsim^{f}_{\omega} cu \quad \textit{if and only if} \quad ct \sqsubseteq_{E_{rec}} cu \quad \textit{if and only if} \quad \overline{\mathrm{ACT}}[\![ct]\!] \sqsubseteq \overline{\mathrm{ACT}}[\![cu]\!].$$

Proof The first equivalence follows from Theorem 2.9 as the conditions of the theorem are ensure by Theorem 5.9, Lemma 5.10 and Theorem 5.16. The second one follows from the soundness and the completeness of the proof system with respect to the model stated in Theorem 2.5. □

6 Conclusion

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

6.1 Summary of Results

In the first paper of this sequel of two paper we defined a general syntax for value passing processes which reflects the late semantic approach. We also gave a general class of denotational models to describe the semantics of languages defined by the general syntactic class. Furthermore we defined a concrete language, CCS_L which is a direct extension of the standard CCS by adding values to the language following the late semantic approach. We then defined a concrete denotational model which is a instantiation of the general class of models. This model is a direct extension of the one given for the pure language $SCCS$ by Abramsky in [Abr91] and a slight modification of the model defined by Milne and Milner in [MM79]. We finish the paper by defining a proof system based on a set of inequation and proof its soundness and completeness with respect to the denotational model model.

In this second paper of the sequel the main focus is on giving 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. One of the main purposes with this second paper is to make the semantic description of the language CCS_L more complete by giving an operational characterization of the preorder derived from the denotational model defined in [Ing95]. Therefore we introduce a suitable notion of a finitary part of a relation and a finitary relation over CCS_L processes. Then we define a value-finite version of the strong applicative ω-bisimulation preorder and show that it is finitary in our sense and is exactly the finitary part of the strong applicative bisimulation preorder. Finally we show the soundness and completeness of the proof system with respect to the value finite strong ω-bisimulation preorder. The full abstractness of the denotational semantics with respect to the value finite strong ω-bisimulation preorder follows directly from this and the soundness and the completeness of the proof system with respect to the denotational semantics.

6.2 Future Work

The results in these papers may be extended in several directions. In the following we will give some examples.

1. Giving a similar semantic description of a CCS-like language with focus on the early semantic approach.
2. Extending the theory to the notion of weak bisimulation preorder and observational congruence.
3. Extending the theory to higher order calculi.
4. Applying symbolic methods to the applicative prebisimulation.

References

[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.

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

[Hen88] M. Hennessy. *Algebraic Theory of Processes.* MIT Press, Cambridge, Massachusetts, 1988.

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

[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.

[Ing95] A. Ingólfsdóttir. A semantic theory for value-passing processes late approach — Part I: A denotational model and its complete axiomatization. Report RS–95–3, brics, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg University Centre, 1995.

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

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

[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.

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

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

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

Recent Publications in the BRICS Report Series

RS-95-22 **Anna Ingólfsdóttir.** ***A Semantic Theory for Value–Passing Processes, Late Approach, Part II: A Behavioural Semantics and Full Abstractness.*** **April 1995. 33 pp.**

RS-95-21 **Jesper G. Henriksen, Ole J. L. Jensen, Michael E. Jørgensen, Nils Klarlund, Robert Paige, Theis Rauhe, and Anders B. Sandholm.** ***MONA: Monadic Second-Order Logic in Practice.*** **May 1995. 17 pp.**

RS-95-20 **Anders Kock.** ***The Constructive Lift Monad.*** **March 1995. 18 pp.**

RS-95-19 **François Laroussinie and Kim G. Larsen.** ***Compositional Model Checking of Real Time Systems.*** **March 1995. 20 pp.**

RS-95-18 **Allan Cheng.** ***Complexity Results for Model Checking.*** **February 1995. 18pp.**

RS-95-17 **Jari Koistinen, Nils Klarlund, and Michael I. Schwartzbach.** ***Design Architectures through Category Constraints.*** **February 1995. 19 pp.**

RS-95-16 **Dany Breslauer and Ramesh Hariharan.** ***Optimal Parallel Construction of Minimal Suffix and Factor Automata.*** **February 1995. 9 pp.**

RS-95-15 **Devdatt P. Dubhashi, Grammati E. Pantziou, Paul G. Spirakis, and Christos D. Zaroliagis.** ***The Fourth Moment in Luby's Distribution.*** **February 1995. 10 pp. To appear in *Theoretical Computer Science.***

RS-95-14 **Devdatt P. Dubhashi.** ***Inclusion–Exclusion(3) Implies Inclusion–Exclusion(n).*** **February 1995. 6 pp.**

RS-95-13 **Torben Braüner.** ***The Girard Translation Extended with Recursion.*** **1995. Full version of paper to appear in Proceedings of CSL '94, LNCS, 1995.**

RS-95-12 **Gerth Stølting Brodal.** ***Fast Meldable Priority Queues.*** **February 1995. 12 pp.**