Theoretical Computer Science 34 (1984) 83-133
North-Holland

TESTING EQUIVALENCES FOR PROCESSES

R. DE NICOLA and M.C.B. HENNESSY

Department of Computer Science, University of Edinburgh, Edinburgh EH8 9YL, United Kingdom

Abstract. Given a set of processes and a set of tests on these processes we show how to define in a natural way three different equivalences on processes. These equivalences are applied to a particular language CCS. We give associated complete proof systems and fully abstract models. These models have a simple representation in terms of trees.

Introduction

In recent years various programming languages with concurrent features have been proposed [2, 3, 16, 19, 22, 29]. This coincides with the increasing complexity of hardware that can be manufactured at reasonable cost. Indeed, if advantage is to be taken of the advances in hardware design and fabrication, where multiprocessor machines are now commonplace, then much work needs to be done on the theory of parallelism to model and analyse such hardware and the related software.

One outstanding and pressing problem is a suitable semantic theory. If one writes a program in PASCAL or FORTRAN, then, apart from considerations of efficiency, one is only interested in the input-output behaviour of the program, which can be considered as a function from the input domain to the output domain. So a semantic theory, suitable for PASCAL, is simply a theory of functions: those functions computed by PASCAL programs. The same remark holds true in general for any language for sequential programming, even though with more complicated languages the nature of the input and output domains of the functions may be rather difficult to discover.

If the language has concurrent features, then it is well known that one cannot represent its behaviour as a function. At least if one does represent it as a function, then much information is lost. However, if we are to build a semantic theory, then a counterpart to functions is needed: if we model programs written in such languages what are the objects in the model?

Various suggestions have been made in the literature [23, 21, 17, 25]. For example, in [23], *communication trees* are put forward but unfortunately they need to be factored by certain equivalences. Moreover, the behaviour which they describe seems too detailed in certain respects [5, 21]. In this paper we put forward another model, called *representation trees*, which is very similar to the models discussed in [18, 25, 21]. However, we show that they can be motivated in a very simple and appealing manner.

The behaviour of programs, or processes, can be investigated by a series of tests. For example with sequential programs we can associate a test with a pair consisting of a predicate on the input domain and a predicate on the output domain. It is very easy to see how the input-output function of a program can be characterised by a set of such tests. For more general programming languages more general kinds of tests are needed. Indeed, the nature of the programming language should suggest the type of test suitable for investigating the behaviour of programs. For example, if the language contains real time constructs, the tests should be able to take time into consideration.

In general one can think of a set of processes and a set of relevant tests. Then two processes are equivalent (with respect to this set of tests) if they pass exactly the same set of tests. The first section of this paper is an attempt at formalising this natural notion of equivalence. It turns out that a satisfactory formalisation must take the possibility of divergence into consideration. In view of this, the natural equivalence can be broken down into two preorders on processes. The first is formulated in terms of the ability to respond positively to a test, the second in terms of the inability not to respond positively to a test. In the latter case the process p will be considered 'less than' the process q if whenever p must respond positively to a particular test, q must also respond positively. Both these preorders have their counterparts in sequential programs, the first being partial correctness, the second total correctness. The natural equivalence between processes is obtained by taking the equivalence associated with the conjunction of these two preorders (which is a third preorder).

The remainder of the paper is devoted to applying these notions to a particular language CCS [23]. This is a primitive language for describing communicating processes but has the advantage of a simple and well-defined operational semantics. We take as the set of tests those tests which can be described in CCS, and examine the substitutive relation generated by the three preorders. (Two processes are substitutively related if they are related in every context.) In Sections 3 and 4 we give three sound and complete proof systems for these relations. These systems consist essentially of a set of axioms for manipulating processes and a form of induction. The completeness theorem leads naturally to fully-abstract denotational models for the language (Section 5), i.e., models in which processes are distinguished if and only if they are distinguished by the associated set of tests. These models are constructed in a very abstract way from the syntax of the language. Moreover, in Section 5 we show that they can be represented as collections of certain kinds of trees. Roughly speaking the tree associated with a process will contain the following information:

(i) the possible sequences of actions that a process can perform,

(ii) to each such sequence a finite set of subsets of actions is associated. This set, called the *acceptance set*, represents the possible futures of a process after a particular sequence of actions.

After performing a sequence of actions a process can do various internal moves to end up in a state represented by an element of the acceptance sets. Such an element, a subset of actions, represents the actions which the process can perform in that state. We have in fact three different models, the differences arising from how we handle divergence and consequently how we order those trees. Although these models are specifically for the language CCS, it is hoped that they can easily be adapted to handle other languages such as CSP [16], ADA [19], DP [3].

In the final section we relate our work with other active research in this area. In particular, the equivalences generated by the three preorders $\sqsubseteq_i$ are related to observational equivalence [23], failures equivalence [18] and weak equivalence [21]. A close relationship with the last one is established and is used to introduce a simpler class of tests which characterise our models. Our tree representations are most closely related to those in [18, 25] and in future work we hope to illuminate the similarities and the differences.

1. General setting

This section is devoted to setting up a rather general framework within which we may discuss testing of processes and the tabulation of the possible outcomes.

1.1

We assume a predefined set of states, *States*, and we let s range over *States*. A *computation* is any nonempty sequence of states. Let *Comp* denote the set of computations, ranged over by c. Note that a computation may be finite or infinite.

Let $\mathcal{O}$, $\mathcal{P}$ (ranged over by o, p respectively) be sets of predefined *observers* and *processes*. Observers may be thought of as agents who perform tests. The effect of observers performing tests on processes may be formalised by saying that for every o and p there is a nonempty set of computations $\mathrm{Comp}(o, p)$. If $c \in \mathrm{Comp}(o, p)$, then the result of o testing p may be the computation c. To indicate that a process passes a test we choose some subset of *States*, denoted *Success*, to be *successful* states. Then a computation is *successful* if it contains a successful state. On the other hand, a computation will be called *unsuccessful* if it contains no successful state. To develop a useful theory we need one further ingredient. The semantic theory of sequential computations, developed in [26, 28], was greatly facilitated by hypothesising the existence of 'partial objects'. For example, the symbol Ω is often used to denote a partial program whose behaviour is totally undefined. It will also be convenient for us to consider such partial objects. To this end we assume the existence of a unary post-fixed predicate on states, $\uparrow$. Informally, $s\uparrow$ means that s is a partial-state, whose properties are underdefined. We can now define *divergence*,

a unary postfixed predicate on computations, which we denote by $\Uparrow$:

$c\Uparrow$ if (i) c is unsuccessful, or

(ii) c contains a state s, such that $s\uparrow$ and is not preceded by a successful state.

By convention a state precedes itself. We also use $\Downarrow$ to denote the negation of $\Uparrow$. The usual notion on input-output can be viewed as a simple instance of the general setting, as can be seen from the following example.

Example. Let $\mathscr{P}$ denote a set of nondeterministic programs for computing over the natural numbers, with the property that they either compute forever or they halt with some natural number as output. For each pair of natural numbers $\langle n, m\rangle$ we have an observer $O(\langle n, m\rangle)$. This observer, when applied to a program p, will attempt to discover if p will give output m on input n. Thus $\mathrm{Comp}(O\langle n, m\rangle, p)$ will consist of all computations of the form

(i) a computation generated by p on input n, followed by

(ii) if this computation halts examine the output. If it is m, then go to a successful state s_S. If it is not m, go to a deadlocked state s_D.

For this simple example we did not need the predicate $\uparrow$ (or, more precisely, $s\uparrow$ for no state s) and there was only one successful and one deadlocked state. In more complicated cases such as CCS, the main example of the paper, the full generality of the notation will be required.

1.2

We may now tabulate the effect of an observer o testing a process p by noting the types of computations in $\mathrm{Comp}(o, p)$. For every $o \in \mathscr{O}$, $p \in \mathscr{P}$ let $R(o, p) \subseteq \{T, \bot\}$ (the *result* set) be defined as follows:

(i) $T \in R(o, p)$ if $\exists c \in \mathrm{Comp}(o, p)$ such that c is *successful.*

(ii) $\bot \in R(o, p)$ if $\exists c \in \mathrm{Comp}(o, p)$ such that $c\Uparrow$.

Note that we do not differentiate between an experiment which deadlocks, i.e., the computation is finite without reaching a successful state and an experiment which diverges, i.e., the computation goes on forever without ever reaching a successful state: they both contribute $\bot$ to the result set. The existence of partially-defined states introduces an additional auxiliary notion of divergence, i.e., when a computation reaches a partially-defined state before reaching a successful state. This also introduces $\bot$ into the result set. Thus, in effect we can distinguish between processes which cannot fail a test (the result set is $\{T\}$) and processes which may pass a test (the result set is $\{\bot, T\}$). This will be elaborated upon shortly.

A natural equivalence between processes immediately suggests itself:

$$p \sim^{c} q \text{ if, for every } o \in \mathscr{O},\ R(o, p) = R(o, q).$$

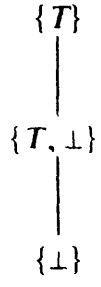
However, it will be more fruitful to consider instead preorders, i.e., relations which

are transitive and reflexive. A preorder $\sqsubseteq$ generates an equivalence $\simeq$ in a natural way, $\simeq = (\sqsubseteq \cap \sqsupseteq)$. In general, preorders (or partial orders) are easier to deal with mathematically and we can easily recover the equivalence $\sim^{\mathcal{C}}$ by studying a preorder which generates it. This gives us a certain amount of freedom since in general there may be more than one preorder which generates any given equivalence. Finally, preorders are more primitive than equivalences and therefore we may use them to concentrate on more primitive notions which combine to form the equivalence $\sim^{\mathcal{C}}$.

The set $\{T, \perp\}$ may be viewed as the simple two point lattice $\mathbb{O}$:

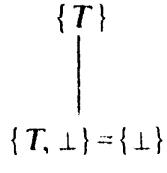
So every result set can be viewed as a subset of this lattice. The theory of powerdomains [24, 27], provides us with general methods of ordering subsets of (complete) partial orders. In [12] it was argued that three different powerdomain constructions arise naturally and that they correspond to three natural views of nondeterministic computations. Here we use these three constructions to give three different orderings on result sets. Since the partial order $\mathbb{O}$ is so trivial, we can avoid descriptions of the powerdomain constructions completely and give the resulting orderings on the subsets of $\mathbb{O}$.

(I)



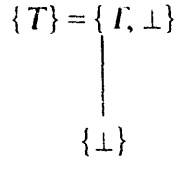
This ordering corresponds to the Egli-Milner powerdomain of $\mathbb{O}$, and we will denote it by $\sqsubseteq_1$.

(II)



This ordering corresponds to the Smyth powerdomain of $\mathbb{O}$. The sets $\{T, \perp\}$ and $\{\perp\}$ are identified and they are less than $\{T\}$. This corresponds to the view that possible divergence is catastrophic. We denote this order by $\sqsubseteq_2$.

(III)



This corresponds to the dual of the Smyth construction and was called the Hoare powerdomain in [12]. The sets $\{T\}$, $\{T, \perp\}$ are identified and both are greater than $\{\perp\}$. This ordering corresponds to the view that divergence is unimportant and is therefore ignored. We denote it by $\sqsubseteq_3$.

These three different orderings on result sets generate three different orderings on processes.

Definition 1.2.1. For given sets of observers and processes $\mathcal{O}$, $\mathcal{P}$ respectively, let $\sqsubseteq_i^{\mathcal{O}} \subseteq \mathcal{P} \times \mathcal{P}$, $i = 1, 2, 3$, be defined by

$$p \sqsubseteq_i^{\mathcal{O}} q \text{ if, } \forall o \in \mathcal{O},\ R(o, p) \sqsubseteq_i R(o, q).$$

We denote the related equivalences by $\simeq_i^{\mathcal{O}}$. The following results are trivial to establish.

Proposition 1.2.2. (a) *$p \sim^{\mathcal{O}} q$ if and only if $p \simeq_1^{\mathcal{O}} q$.*
(b) *$p \sqsubseteq_1^{\mathcal{O}} q$ if and only if $p \sqsubseteq_2^{\mathcal{O}} q$ and $p \sqsubseteq_3^{\mathcal{O}} q$.*

Thus we have reformulated the natural equivalence $\sim^{\mathcal{O}}$ as the equivalence generated by a preorder $\sqsubseteq_1$. This preorder is further broken down into two more primitive preorders $\sqsubseteq_2$, $\sqsubseteq_3$. The relevance of these primitive preorders can be motivated by the following definition and proposition.

Definition 1.2.3. (a) *p* **may satisfy** *o* if $T \in R(o, p)$.
(b) *p* **must satisfy** *o* if $\{T\} = R(o, p)$.

Thus *p may satisfy o* if there is a resulting successful computation whereas *p must satisfy o* if every resulting computation is successful.

Proposition 1.2.4. (a) *$p \sqsubseteq_3^{\mathcal{O}} q$ if, $\forall o \in \mathcal{O}$, p* **may satisfy** *o implies q* **may satisfy** *o.*
(b) *$p \sqsubseteq_2^{\mathcal{O}} q$ if, $\forall o \in \mathcal{O}$, p* **must satisfy** *o implies q* **must satisfy** *o.*

In the remainder of this paper we apply this general theory to the language CCS [23]. To do so we need to specify:

$\mathcal{P}$ - a set of processes (CCS terms),
$\mathcal{O}$ - a set of observers,
States - a set of states, together with a subset of successful states and the under-defined-predicate $\uparrow$ on states,
Comp - a method of assigning to every observer and process a nonempty set of computations (sequences of states).

The three resulting preorders have many interesting mathematical properties. We will give three complete proof systems for these orders and three fully-abstract denotational models.

2. CCS

2.1

In this section we review the definition of CCS and its operational semantics. We use 'pure' CCS [23] and our version will be closest to that presented in [13].

Let X be a set of variables, ranged over by x. Let Σ_k be a set of operators of arity k. We use Σ to denote $\bigcup\{\Sigma_k \mid k \geq 0\}$. The set of recursive terms over Σ, REC_Σ, ranged over by t, u, is defined by the following BNF-like notation:

$$t ::= x \mid \mathrm{op}(t_1, \ldots, t_k), \mathrm{op} \in \Sigma_k \mid \mathrm{rec}\, x.t$$

The operation rec x._ binds occurrences of x in the subterm t of rec $x.t$. This gives rise to the usual notions of free and bound variables in a term. Let $\mathrm{FV}(t)$ be the set of free variables in t. If $\mathrm{FV}(t) = \emptyset$, we say that t is *closed*. Let CREC_Σ denote the set of closed terms and we use p, q as meta-variables to range over this set. A term is *finite* if it is closed and contains no occurrence of rec x._. Let FREC_Σ denote the set of finite terms, and we use d, e as meta-variables. Let $t[u/x]$ denote the term which results from substituting u for every free occurrence of x in t. More generally, let SUB be the set of *substitutions*, i.e., mappings from variables to terms. We use ρ as a meta-variable over SUB. Let $t\rho$ denote the result of substituting $\rho(x)$ for every free occurrence of x in t, for every x in X. A substitution is *closed* if, for every x in X, $\rho(x)$ is closed.

Pure CCS may be defined by choosing a particular set of operators, Σ. Let Δ denote a set of unary operators, ranged over by α, β. Let $\bar{\Delta} = \{\bar{\alpha} \mid \alpha \in \Delta\}$. The operator $\bar{\alpha}$ is said to be the *complement* of α. It will also be convenient to let $\bar{\bar{\alpha}}$ denote α. Let $\Lambda = \Delta \cup \bar{\Delta} \cup \{\tau\}$, where τ is a distinguished unary operator not occurring in $\Delta \cup \bar{\Delta}$. Λ is often referred to as the set of basic *actions*, and we used μ to range over it. We use λ to range over $\Delta \cup \bar{\Delta}$.

Let PER denote the set of partial functions over Λ, such that $S \in \mathrm{PER}$ implies

(i) $S(\tau) = \tau$,

(ii) $S(\lambda)$ defined implies $S(\bar{\lambda})$ is defined and $S(\bar{\lambda}) = \overline{S(\lambda)}$,

(iii) $S(\lambda) = S(\lambda')$ implies $\lambda = \lambda'$.

We are now ready to define the operator set for CCS. Let

$$\Sigma_0 = \{\mathrm{NIL}, \Omega\},$$

$$\Sigma_1 = \{\mu \mid \mu \in \Lambda\} \cup \{[S] \mid S \in \mathrm{PER}\},$$

$$\Sigma_2 = \{+, |\},$$

$$\Sigma_n = \emptyset, \quad n \geq 3.$$

In accordance with [23], μ will be used in prefix form, $[S]$ in postfix form and $+, |$ in infix form. In the future when we refer to terms, closed terms etc., we mean terms generated by this set of operators.

The operational semantics is given in terms of labelled rewrite rules over closed terms. For each $\mu \in \Lambda$ we define a relation $\xrightarrow{\mu}$ over closed terms with the intuition that

(i) $p \xrightarrow{\lambda} q$ if p may evolve to q by reacting to a λ-stimulus from the environment,

(ii) $p \xrightarrow{\tau} q$ if p may evolve to q by performing an internal action, which is independent of the environment.

In pure CCS the only possible actions are synchronisations and therefore a τ move will correspond to a synchronisation of two subprocesses.

Definition 2.1.1. Let $\xrightarrow{\mu}$ be the least relation over closed terms which satisfies

(i) $\mu p \xrightarrow{\mu} p$;

(ii) $p_1 \xrightarrow{\mu} q$ implies $p_1 + p_2 \xrightarrow{\mu} q$,
$p_2 + p_1 \xrightarrow{\mu} q$,
$p_1 | p_2 \xrightarrow{\mu} q | p_2$,
$p_2 | p_1 \xrightarrow{\mu} p_2 | q$;

(iii) $p \xrightarrow{\mu} q$, $S(\mu)$ defined, implies $p[S] \xrightarrow{S(\mu)} q[S]$;

(iv) $p_1 \xrightarrow{\lambda} q_1$, $p_2 \xrightarrow{\bar{\lambda}} q_2$ implies $p_1 | p_2 \xrightarrow{\tau} q_1 | q_2$;

(v) $t[\text{rec } x.t/x] \xrightarrow{\mu} q$ implies $\text{rec } x.t \xrightarrow{\mu} q$.

We also need the following unary predicate on closed terms.

Definition 2.1.2. Let $\downarrow$ be the least predicate on closed terms which satisfies

(i) $\text{NIL}\downarrow$, $\alpha p\downarrow$,

(ii) $p\downarrow, q\downarrow$ implies $(p+q)\downarrow$, $(p|q)\downarrow$, $p[S]\downarrow$,

(iii) $t[\text{rec } x.t/x]\downarrow$ implies $\text{rec } x.t\downarrow$.

Let $p\uparrow$ if not $p\downarrow$. So, for example, $\Omega\uparrow$ and $\text{rec } x.(ap+x)\uparrow$. Informally $p\uparrow$ means that there is an unguarded recursion or an unguarded occurrence of Ω.

2.2

In this section we show how to view CCS as a particular example of the general setting explained in Section 1. The set of processes will just be closed CCS-terms, i.e., CREC_{Σ}, and the principal point to settle is how to describe observers. It seems reasonable to use the same language to describe both the processes and the observers. An observer may test a process by communicating with it and CCS was designed to describe communication. We do, however, need some additional machinery for indicating the success of a test. Let ω be a distinguished action symbol, not in Λ. We use ω as a special action which 'reports success'. Now let $\mathcal{O}$ be $\text{CREC}_{\Sigma_{\{\omega\}}}$, i.e., an observer is any term obtained from the augmented operator set.

Example. The term $o \equiv \bar{\alpha}\bar{\beta}\omega\mathrm{NIL}$ is an observer for testing whether a process can perform an α action followed by a β action. For example, the process $p \equiv \alpha(\beta\mathrm{NIL}+\gamma\mathrm{NIL})$ passes this test because when o and p are put in communication, success can be eventually reported:

$$o|p \xrightarrow{\tau} \bar{\beta}\omega\mathrm{NIL}|(\beta\mathrm{NIL}+\gamma\mathrm{NIL}) \xrightarrow{\tau} \omega\mathrm{NIL}|\mathrm{NIL} \xrightarrow{\omega}$$

CCS is 'applicative' in nature and therefore the natural set of states is the set of all closed terms. So let ***States*** $= \mathrm{CREC}_{\Sigma\cup\{\omega\}}$ (which includes CREC_{Σ}). Let the set of successful states, ***Success***, be $\{p \mid \exists p'.p \xrightarrow{\omega} p'\}$ and we have already defined the predicate $\uparrow$. So we view the existence of exposed unguarded recursions in a term as saying that the term is underdefined.

A *computation* is any sequence of terms $\{p_n \mid n \geq 0\}$ (finite or infinite) such that (i) if p_n is the final element in the sequence, then $p_n \xrightarrow{\tau} p'$ for no p', and (ii) otherwise $p_n \xrightarrow{\tau} p_{n+1}$.

Finally, for $o \in \mathcal{O}$, $p \in \mathcal{P}$, let $\mathrm{Comp}(o, p)$ be the set of computations whose initial element is the term $(o|p)$.

These definitions immediately give three different preorders on $\mathcal{P}$, the set of closed CCS-terms. To emphasise their import we translate Definition 1.2.3 into this setting:

(a) p **may satisfy** o if $(o|p) \xrightarrow{\tau}{}^* q$ for some q such that $q \xrightarrow{\omega}$,

(b) p **must satisfy** o if whenever $o|p = o_0|p_0 \xrightarrow{\tau} o_1|p_1 \xrightarrow{\tau} \cdots$ is a computation from $o|p$, then (i) $\exists n \geq 0$ such that $o_n \xrightarrow{\omega}$, and (ii) $o_k \mid p_k\uparrow$ implies $o_{k'} \xrightarrow{\omega}$ for some $k' \leq k$.

Notation. We have used $q \xrightarrow{\mu}$ as a shorthand for $(\exists q'.q \xrightarrow{\mu} q')$ and $q \not\xrightarrow{\mu}$ is used as a shorthand for its negation. We will also use $\rightarrow$ to denote $\xrightarrow{\tau}$. Let $\mathrm{Dead} = \{p \mid p\downarrow, p \not\xrightarrow{\tau}\}$ and $\mathrm{Fail} = \{p \mid p \in \mathrm{Dead}, p \not\xrightarrow{\omega}\}$. From now on we will drop the occurrences of $\mathcal{O}$ when this leads to no confusion. Thus $\sqsubseteq_i^{\mathcal{O}}$ will be rendered as $\sqsubseteq_i$. We will also use the usual notation from [23] for CCS terms and their operational semantics. So the precedence of the operators is given by

$$[S] > \mu > | > +$$

The occurrences of NIL will usually be omitted from a term. So $\alpha\mathrm{NIL}+\beta\mathrm{NIL}$ will be rendered as $\alpha+\beta$.

The relation $\overset{\lambda}{\Rightarrow}$ is defined by $p \overset{\lambda}{\Rightarrow} q$ if there exist p_1, q_1 such that $p \xrightarrow{\tau}{}^* p_1 \xrightarrow{\lambda} q_1 \xrightarrow{\tau}{}^* q$ and $\overset{\tau}{\Rightarrow}$ will sometimes be used for $\xrightarrow{\tau}{}^*$, for the sake of uniformity. Then $S(p) = \{\lambda \mid p \overset{\lambda}{\Rightarrow} p'\}$ and $\mathrm{Der}_{\mu}(p) = \{p' \mid p \overset{\mu}{\Rightarrow} p'\}$. Note that $p \in \mathrm{Der}_{\tau}(p)$. If s is a sequence of actions, i.e., $s \in \lambda^*$, then $\overset{s}{\Rightarrow}$ is defined in a natural way from $\overset{\mu}{\Rightarrow}$, with $\overset{\varepsilon}{\Rightarrow}$ coinciding with $\overset{\tau}{\Rightarrow}$. The predicates $\Uparrow$, $\Downarrow$ apply only to computations but in future we will also apply them to terms. So $p\Uparrow$ will mean that there is a computation whose initial element is p and which is either infinite or contains a term q such that $q\uparrow$. We will also often revert to graphical representations of terms [23] which use only the operators μ, $+$, NIL, Ω.

The preorders $\sqsubseteq_i$ defined on closed terms may be extended in the usual way to arbitrary terms by

$t \sqsubseteq_i u$ if, for every closed substitution ρ, $t\rho \sqsubseteq_i u\rho$.

Finally, let $\sqsubseteq_i^c$ be the relations obtained by closing under contexts:

$t \sqsubseteq_i^c u$ if, for every context $C[\ ]$, $C[t] \sqsubseteq_i C[u]$.

These three relations $\sqsubseteq_i^c$ are the topics of the remainder of this paper and we close this section with some remarks on them.

Many observers are useless from the point of view of distinguishing processes. For example, if o contains no occurrence of ω, then for every p, $R(o, p) = \{\perp\}$. So these observers may be ignored. One can ask, in general, what is the smallest set of observers which generates the preorders $\sqsubseteq_i$. This question is discussed in Section 6.4. By and large, the preorders $\sqsubseteq_i$ are well behaved. For example, they are preserved by all the CCS operators except $+$.

As an example we prove the result for the composition operator $|$.

Proposition 2.2.1. *$p \sqsubseteq_i q$ implies $p|r \sqsubseteq_i q|r$.*

Proof. The result follows from the following remarks:

$i = 3$: For any $o \in \mathcal{O}$, $(p|r)$ **may satisfy** o if and only if p **may satisfy** $(r|o)$.

$i = 2$: For any $o \in \mathcal{O}$, $(p|r)$ **must satisfy** o if and only if p **must satisfy** $(t|o)$.

$i = 1$: Follows from the two previous cases. □

In general, $\sqsubseteq_i$, $i = 1, 2$, are not preserved by the operator $+$. For example, $\alpha \sqsubseteq_2 \tau\alpha$ but $\lambda + \alpha \not\sqsubseteq_2 \lambda + \tau\alpha$: if o denotes $\bar{\lambda}\omega$, then $\lambda + \alpha$ **must satisfy** o whereas $\lambda + \tau\alpha | 0 \rightarrow \alpha | \bar{\lambda}\omega \in$ Dead. However, $\sqsubseteq_3$ and $\sqsubseteq_3^c$ coincide but for uniformity we will treat the three cases together.

Definition 2.2.2. Let $p \sqsubseteq_i^+ q$ if, for every closed term r, $r + p \sqsubseteq_i r + q$.

We extend $\sqsubseteq_i^+$ to arbitrary terms as usual and let $\simeq_i^+$ denote the related equivalences.

Theorem 2.2.3. *$t \sqsubseteq_i^c u$ if and only if $t \sqsubseteq_i^+ u$.*

The only difficulty with this theorem is to prove that $\sqsubseteq_i^+$ is preserved by contexts involving the recursion operator. This requires some technical concepts which we have not yet developed. So we will postpone the proof until Section 4.1.

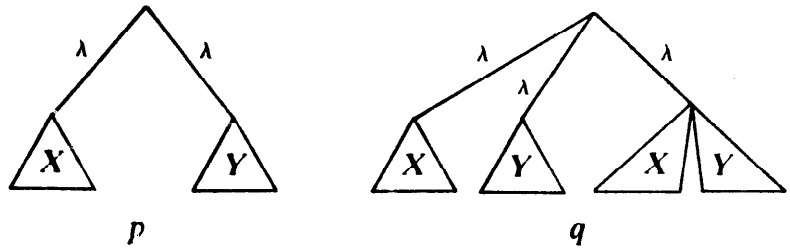
2.3

In this section we give some examples and counter-examples. These will mainly concern only the equivalence $\simeq_1$, which in the sequel we will abbreviate by $\simeq$.

Example 2.3.1. For any X, Y,

$$\lambda X + \lambda Y \simeq^+ \lambda X + \lambda Y + \lambda(X + Y).$$

Using the representation of terms by trees of [23] these may be described as



The reader may like to convince him- (or her-)self that for any observer o, p may satisfy o if and only if q may satisfy o and p must satisfy o if and only if q must satisfy o. In the next section we will give a set of axioms for transforming terms which preserve $\simeq^+$ and we will show how to transform p into q.

Example 2.3.2. For any X, Y, Z,

$$\lambda X + \lambda(X + Y + Z) \simeq^+ \lambda X + \lambda(X + Y) + \lambda(X + Y + Z).$$

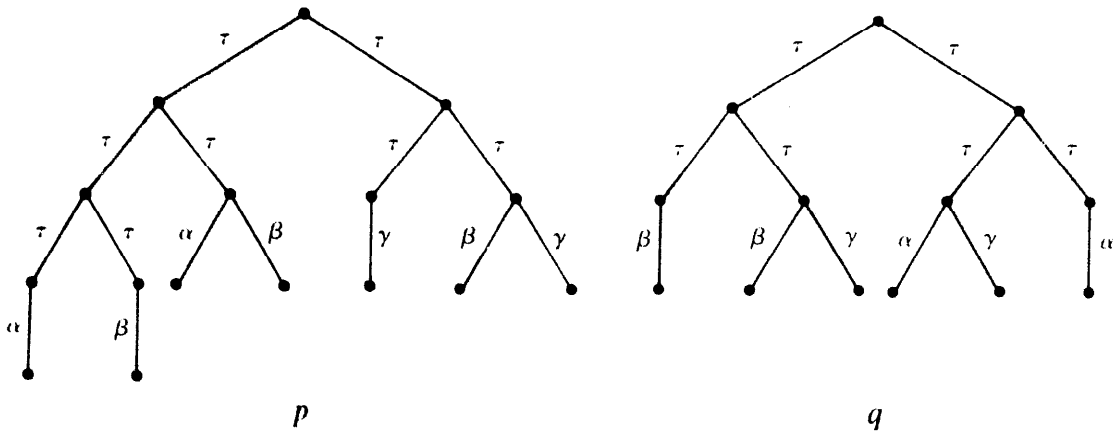
However, we can distinguish very similar pairs of trees.

Example 2.3.3. (a) $p \equiv \lambda\alpha + \lambda(\alpha + \beta + \gamma) \not\simeq \lambda\alpha + \lambda(\alpha + \beta + \gamma) + \lambda\beta \equiv q$. This follows since p **must satisfy** $\bar{\lambda}\bar{\alpha}\omega$ whereas $q \mid \bar{\lambda}\bar{\alpha}\omega \rightarrow \beta \mid \bar{\alpha}\omega \in$ Dead.

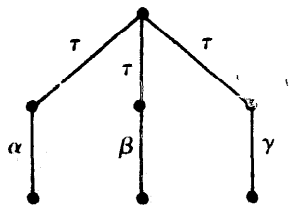
(b) $p \equiv \lambda\alpha + \lambda(\beta + \gamma) \not\simeq \lambda\alpha + \lambda\beta + \lambda(\beta + \gamma) \equiv q$. since p **must satisfy** $\bar{\lambda}(\bar{\alpha}\omega + \bar{\gamma}\omega)$ whereas $q \mid \bar{\lambda}(\alpha\omega + \bar{\gamma}\omega) \rightarrow \beta \mid (\bar{\alpha}\omega + \bar{\gamma}\omega) \in$ Dead.

We now consider some examples with internal moves.

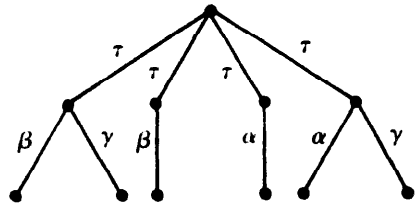
Example 2.3.4. Consider the two trees



It turns out that $p \not\simeq q$ although it is quite difficult to see. Now consider the two trees



p'

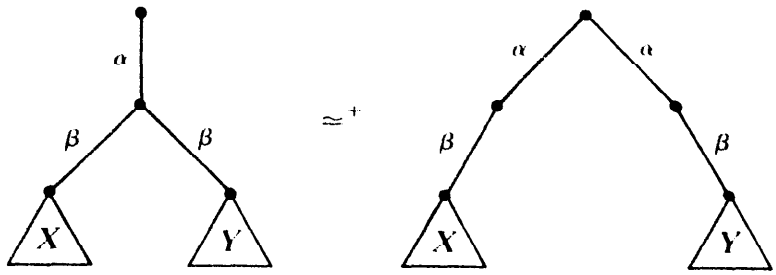


q'

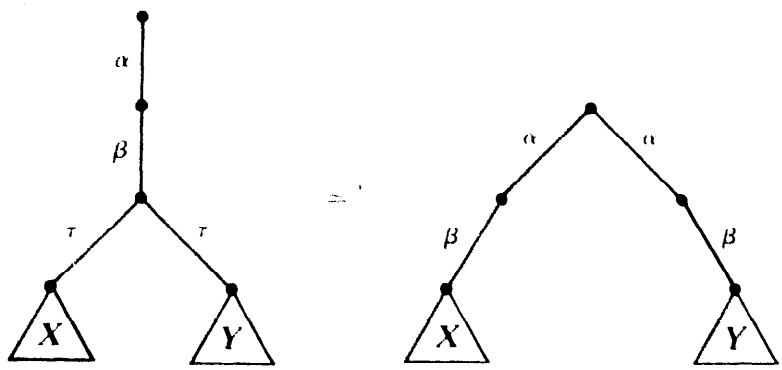
These trees have a much simpler 'τ-structure' and it is relatively easy to see that they are not equivalent. For example, no matter what internal move q' makes it can always perform either a β or an α. However, p' can make an internal move to become γNIL which can perform neither. So q' **must satisfy** $(\bar{\alpha}\omega + \bar{\beta}\omega)$ whereas p' **must satisfy** $(\bar{\alpha}\omega + \bar{\beta}\omega)$.

The axioms given in the next section enable us to transform any term into a term with a 'τ-structure' similar to that of p', q'. We will see that p may be transformed into p' and q transformed into q'.

Example 2.3.5. (a) $\alpha(\beta X + \beta Y) \simeq^+ \alpha\beta X + \alpha\beta Y$. In terms of trees:

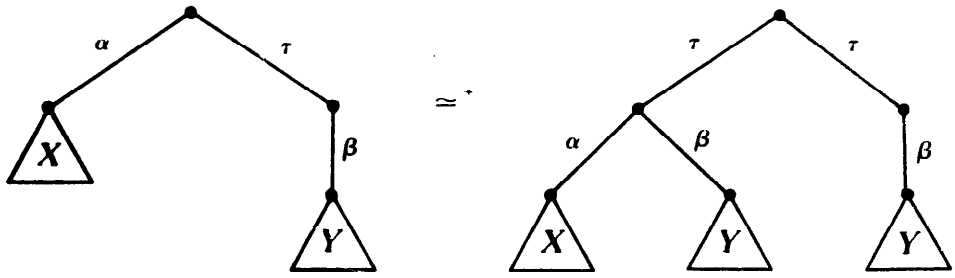


(b) $\alpha\beta(\tau X + \tau Y) \simeq^+ \alpha\beta X + \alpha\beta Y$. In terms of trees:



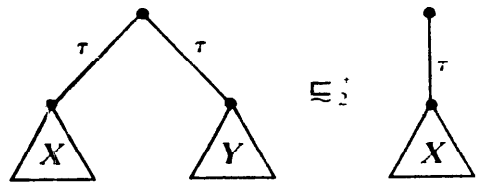
These two examples show that $\simeq^+$ tends to abstract away from 'when choices are made'.

Example 2.3.6. $\alpha X + \tau\beta Y \simeq^+ \tau(\alpha X + \beta Y) + \tau\beta Y$.



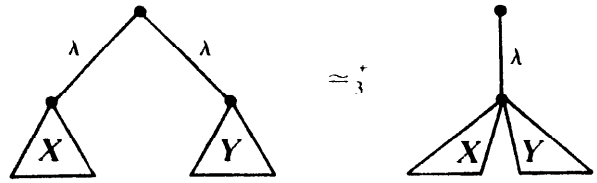
This will in fact be one of our more useful axioms. With it we may transform terms so that they represent processes in which all choices are either purely external or purely internal.

Example 2.3.7. (a) $\tau X + \tau Y \sqsubseteq_2^+ \tau X$. In graphical terms this may be rendered as



The presence of τ on the left-hand side is important. For example, $\alpha + \beta \not\sqsubseteq_2^+ \tau\alpha$. This follows since $\alpha + \beta$ **must satisfy** $\bar{\beta}\omega$ whereas $\tau\alpha \,|\, \bar{\beta}\omega \rightarrow \alpha \,|\, \bar{\beta}\omega \in$ Dead.

(b) $\lambda X + \lambda Y \simeq_3^+ \lambda(X + Y)$. In graphical terms we have



Thus the relation $\simeq_3^+$ ignores all the tree structure of terms. We will also see that $\tau X \simeq_3^+ X$, so that $\simeq_3^+$ is a very weak relation.

3. Proof systems

3.1

In this section we examine axiom systems for the three relations $\sqsubseteq_i^+$ defined in the previous section for CCS. The basic axioms are given in Table 1 (see p. 98). Most of them are given in terms of "=", and they are designed to be used in conjunction with the following rules:

$X = Y$ implies $X \sqsubseteq Y$, $Y \sqsubseteq X$

$X \sqsubseteq Y$, $Y \sqsubseteq X$ implies $X = Y$.

The axioms (A1)-(A4), (S1)-(S3), (C1), (Ω1), (Ω2) are essentially taken from [23, 13]. The summation notation used in (C1) is justified by the axioms (A1)-(A4). As in [23], $\sum_{i\in\emptyset} t_i$ denotes NIL. The notation $t\{+\Omega\}$ is meant to denote that the term Ω is optional as a summand. The axioms of particular interest are (N1)-(N4), which replace the τ-laws of [23]. Indeed, these new axioms imply these τ-laws.

Let A_1 denote the set of axioms in Table 1 other than (E1), (F1). Let A_2 be the set A_1 together with (E1) and A_3 be the set A_1 together with (F1). We write $t \sqsubseteq_i u$ ($t =_i u$) if $t \sqsubseteq u$ ($t = u$) can be derived from the axioms A_i. These axioms are rather low-level but we can derive many more complicated derived axioms. A list of some important ones are given in Table 2 (see p. 99). The remainder of this section is devoted to deriving these axioms and re-examining the examples of the previous section.

(D1)—$\tau X + \Omega \sqsubseteq X + \Omega$ from (N4).

Conversely

$X + \Omega \sqsubseteq X + \tau X + \Omega$ from (Ω2)

$\sqsubseteq \tau(X + X) + \Omega$ from (N2)

$= \tau X + \Omega$ from (A1).

(D2)—We use induction on the size of I:

(i) $|I| = 1$.

Then

$\mu X = \mu X + \mu X$ from (A1)

$= \mu(\tau X + \tau X)$ from (N1)

$= \mu\tau X$ from (A1).

(ii) $I = J \cup \{o\}$.

Then

$$\mu\left(\sum_{i\in I} \tau X_i\right) = \mu\left(\sum_{i\in J} \tau X_i + \tau X_0\right)$$

$$= \mu\left(\tau \sum_{i\in J} \tau X_i + \tau X_0\right)$$

using induction with $\mu = \tau$

$$= \mu\left(\sum_{i\in J} \tau X_i\right) + \mu X_0 \quad \text{from (N1)}$$

$$= \sum_{i\in J} \mu X_i + \mu X_0$$

using induction.

(D3)—$\mu(X + \Omega) + \Omega \sqsubseteq \mu(X + X) + \Omega$ from (Ω2)

$= \mu X + \Omega$ from (A1).

Conversely,

$\mu X + \Omega \sqsubseteq \mu X + \mu(X + \Omega) + \Omega$ from (Ω2)

$= \mu(\tau X + \tau(X + \Omega)) + \Omega$ from (N1)

$$= \mu(\tau X + X + \Omega) + \Omega \qquad \text{from (D1)}$$

$$\sqsubseteq \mu(\tau(X + X) + \Omega) + \Omega \qquad \text{from (N2)}$$

$$= \mu(\tau X + \Omega) + \Omega \qquad \text{from (A1)}$$

$$= \mu(X + \Omega) + \Omega \qquad \text{from (D1).}$$

(D4)—$X + \tau Y = X + \tau Y + \tau Y$ from (A1)

$$\sqsubseteq \tau(X + Y) + \tau Y \qquad \text{from (N2).}$$

Conversely,

$$\tau(X + Y) + \tau Y \sqsubseteq X + Y + \tau Y \qquad \text{from (N4)}$$

$$\sqsubseteq X + \tau Y \qquad \text{from (N2), (A1).}$$

(D5)—This is Example 2.3.1 of the previous section:

$$\mu X + \mu Y = \mu(\tau X + \tau Y) \qquad \text{from (N1)}$$

$$= \mu(\tau X + \tau(X + \tau Y)) \qquad \text{from (D4)}$$

$$= \mu(\tau X + \tau(\tau(X + Y) + \tau Y)) \qquad \text{from (D4)}$$

$$= \mu(\tau X + \tau(X + Y) + \tau Y) \qquad \text{from (N1)}$$

$$= \mu X + \mu(X + Y) + \mu Y \qquad \text{from (D2).}$$

(D6)—Example 2.3.2:

(i) $\mu = \tau$:

$$\tau X + \tau(X + Y + Z)$$

$$= \tau(X + Y + Z + \tau X) + \tau(X + Y + Z) \qquad \text{from (D4)}$$

$$= \tau(X + Y + Z + \tau X + \tau(X + Y)) + \tau(X + Y + Z) \qquad \text{from (D4), (A1)}$$

$$= \tau(\tau(X + Y + Z) + \tau X + \tau(X + Y)) + \tau(X + Y + Z) \qquad \text{from (D4), (A1)}$$

$$= \tau(X + Y + Z) + \tau X + \tau(X + Y) \qquad \text{from (D2), (A1).}$$

(ii) $\mu = \lambda$:

$$\lambda X + \lambda(X + Y + Z) = \lambda(\tau X + \tau(X + Y + Z)) \qquad \text{from (N1)}$$

$$= \lambda(\tau X + \tau(X + Y) + \tau(X + Y + Z)) \qquad \text{from (i)}$$

$$= \lambda X + \lambda(X + Y) + \lambda(X + Y + Z) \qquad \text{from (N2).}$$

Note that in (D5) to prove equality between terms which contain no occurrence of τ, we first introduce τ's, then use some τ-laws, and then eliminate τ. The same method is used in (D6), where it is seen that (D2) can be used to translate τ properties into λ properties.

Table 1. Basic axioms.

$$X+X=X \tag{A1}$$

$$X+Y=Y+X \tag{A2}$$

$$X+(Y+Z)=(X+Y)+Z \tag{A3}$$

$$X+\mathrm{NIL}=X \tag{A4}$$

$$\mu X+\mu Y=\mu(\tau X+\tau Y) \tag{N1}$$

$$X+\tau Y\sqsubseteq\tau(X+Y) \tag{N2}$$

$$\mu X+\tau(\mu Y+Z)=\tau(\mu X+\mu Y+Z) \tag{N3}$$

$$\tau X\sqsubseteq X \tag{N4}$$

$$\mathrm{NIL}[S]=\mathrm{NIL} \tag{S1}$$

$$(X+Y)[S]=X[S]+Y[S] \tag{S2}$$

$$\mu X[S]=\begin{cases} S(\mu)X[S] & \text{if } S(\mu) \text{ defined} \\ \mathrm{NIL} & \text{otherwise} \end{cases} \tag{S3}$$

Let t denote $\sum_{i\in I}\mu_i t_i\{+\Omega\}$, u denote $\sum_{j\in J}\gamma_j u_j\{+\Omega\}$.

$$t|u=\sum_{i\in I}\mu_i(t_i|u)+\sum_{j\in J}\gamma_j(t|u_j)+\sum_{\mu_i=\gamma_j}\tau(t_i|u_j) +\{\Omega|\Omega \text{ is a summand of } t \text{ or } u\} \tag{C1}$$

$$\Omega[S]=\Omega \tag{Ω1}$$

$$\Omega\sqsubseteq X \tag{Ω2}$$

$$t[\mathrm{rec}\ x.t/x]\sqsubseteq\mathrm{rec}\ x.t \tag{REC1}$$

$$\tau X+\tau Y\sqsubseteq X \tag{E1}$$

$$X\sqsubseteq\tau X+\tau Y \tag{F1}$$

(D7)—An instance of (D2).

(D8)—$X+\tau(X+Y)=\tau(X+X+Y)+\tau(X+Y)$ from (D4)

$=\tau(X+Y)$ from (A1).

(D9)—$\mu(X+\tau Y)+\mu Y=\mu(\tau(X+\tau Y)+\tau Y)$ from (N1)

$=\mu(\tau(\tau(X+Y)+\tau Y)+\tau Y)$ from (D4)

$=\mu(\tau(X+Y)+\tau Y+\tau Y)$ from (N1)

Table 2. Derivable axioms.

$$\tau X + \Omega = X + \Omega \tag{D1}$$

$$\mu\left(\sum_{i\in I} \tau X_i\right) = \sum_{i\in I} \mu X_i \,,\ I \neq \emptyset \tag{D2}$$

$$\mu X + \Omega = \mu(X + \Omega) + \Omega \tag{D3}$$

$$X + \tau Y = \tau(X + Y) + \tau Y \tag{D4}$$

$$\mu X + \mu Y = \mu X + \mu Y + \mu(X + Y) \tag{D5}$$

$$\mu X + \mu(X + Y + Z) = \mu X + \mu(X + Y) + \mu(X + Y + Z) \tag{D6}$$

$$\mu\tau X = \mu X \tag{D7}$$

$$X + \tau(X + Y) = \tau(X + Y) \tag{D8}$$

$$\mu(X + \tau Y) + \mu Y = \mu(X + \tau Y) \tag{D9}$$

$$\Omega + \mu\left(\sum_{i\in I} X_i + \Omega\right) = \Omega + \sum_{i\in I} \mu X_i \,,\ I \neq \emptyset \tag{D10}$$

$$X + \Omega = \Omega \tag{E2}$$

$$\tau X + \tau Y \sqsubseteq \tau X \tag{E3}$$

$$X + \Omega = X \tag{F2}$$

$$= \mu(\tau(X + Y) + \tau Y) \qquad \text{from (A1)}$$

$$= \mu(X + \tau Y) \qquad \text{from (D4).}$$

These last three derived rules are the τ-laws from [15, 23].

(D10)—We use induction on the size of I. If $|I| = 1$, then (D10) coincides with (D3). Otherwise we may write $\sum_{i\in I} X_i$ as $X_0 + Y$ where $Y = \sum_{i\in J} X_i$. Then

$$\Omega + \sum_{i\in J} \mu X_i + \mu X_0 = \Omega + \mu(Y + \Omega) + \mu X_0 \qquad \text{by induction}$$

$$= \Omega + \mu(\tau(Y + \Omega) + \tau X_0) \qquad \text{from (N1)}$$

$$= \Omega + \mu(Y + X_0 + \Omega) \qquad \text{from (D1).}$$

(E2)—This is derived from the set of axioms A_2.

$\Omega \sqsubseteq X + \Omega$ is an instance of $(\Omega 2)$.

Conversely,

$$X + \Omega \sqsubseteq X + \tau X + \tau\Omega \qquad \text{from } (\Omega 2)$$

$$= \tau X + \tau\Omega \qquad \text{from (D8), (A1)}$$

$$\sqsubseteq \Omega \qquad \text{from (E1).}$$

(E3)—This is again derived from A_2.

$$\tau X + \tau Y = \tau\tau X + \tau Y \qquad \text{from (D7)}$$
$$\sqsubseteq \tau X \qquad \text{from (E1).}$$

(F2)—A derived rule from A_3.

$X + \Omega \sqsubseteq X$ follows from (Ω2), (A1).

Conversely,

$$X \sqsubseteq \tau X + \tau\Omega \qquad \text{from (F1)}$$
$$\sqsubseteq X + \Omega \qquad \text{from (N4).}$$

Most of the examples of the previous section have already been covered by these derivations. We examine two exceptions.

Example 3.1.1. Derivable in A_1:

$$\alpha\beta X + \alpha\beta Y = \alpha(\tau\beta X + \tau\beta Y) \qquad \text{from (N1)}$$
$$= \alpha(\beta X + \tau\beta X + \beta Y + \tau\beta Y) \qquad \text{from (D8)}$$
$$= \alpha(\tau(\beta X + \beta Y) + \tau(\beta X + \beta Y)) \qquad \text{from (N3) twice}$$
$$= \alpha\tau(\beta X + \beta Y) \qquad \text{from (A1)}$$
$$= \alpha(\beta X + \beta Y) \qquad \text{from (D7)}$$
$$= \alpha\beta(\tau X + \tau Y) \qquad \text{from (N1).}$$

Finally, we examine the set of axioms A_3 in detail:

$$X \sqsubseteq \tau X + \tau X \qquad \text{from (F1)}$$
$$= \tau X \qquad \text{from (A1).}$$

Together with (N4) this shows that $X = \tau X$ is a derived axiom. Using this in (N1) we obtain $\mu X + \mu Y = \mu(X + Y)$.

Indeed the axioms (N1)-(N4), (F1) may be replaced by

$$X = \tau X$$
$$\mu X + \mu Y = \mu(X + Y)$$
$$X \sqsubseteq X + Y.$$

However, our presentation has the advantage that it shows the duality between the two systems A_2 and A_3 and how they are obtained from (A1).

3.2

In this section we examine complete proof systems.

In [23] it was pointed out that Turing machines can be simulated in pure CCS. Moreover, this simulation may be carried out in such a way that a Turing machine TM will diverge on a blank input tape if and only if its translation $[\![TM]\!]$ is such that $[\![TM]\!] \simeq_i^+ \Omega$, $i = 1, 2$ or 3. Consequently, there cannot be any recursively enumerable complete axiomatisation of any of the relations $\sqsubseteq_i^+$, $i = 1, 2, 3$. We will give complete systems which are not recursively enumerable. These are of considerable interest in themselves. For example, they show that the axioms A_i are complete for finite terms and if we add a sufficiently powerful form of induction, we get completeness for arbitrary (closed) terms. It is the required form of induction which introduces the nonrecursive enumerability.

An arbitrary term t may be considered as a finite notation for an infinite tree. This tree is obtained by 'unwinding' the recursive terms via

$$\text{rec } x.u \to u[\text{rec } x.u/x]$$

We are interested in the set of finite trees which approximate this unwinding of t. These may be defined in the following way: Let $<$ be the least relation between terms which satisfies ($\Omega 2$), (REC1) and

(1) $t_1 < u_i$, $0 \leq i \leq k$ implies $\text{op}(t_1, \ldots, t_k) < \text{op}(u_1, \ldots, u_k)$ for every $\text{op} \in \Sigma_k$,

(2) $t < t$,

(3) $t < u$ $u < r$ implies $t < r$.

The relation $<$ will be referred to as the least pre-congruence (w.r.t. Σ) generated by the axioms ($\Omega 2$), (REC1). Let $\text{FIN}(t) = \{d \mid d \in \text{FREC}_\Sigma \mid d < t\}$.

These sets of finite approximations have been studied at length in [11, 10, 4]. For example, $\text{FIN}(t)$ is directed with respect to the relation $<$ [11]. The unwindings of terms may be defined in the following way:

(i) $t^0 = \Omega$,

(ii) $(\text{rec } x.t)^{n+1} = t[(\text{rec } x.t)^n/x]$,

(iii) $\text{op}(t_1, \ldots, t_k)^{n+1} = \text{op}(t_1^{n+1}, \ldots, t_k^{n+1})$.

Lemma 3.2.1. *If $d \in \text{FIN}(t)$, then there exists an $n \geq 0$ such that $d < t^n$.*

Proof. For the proof, see [11]. □

We are now ready to give the proof system. This is given in terms of a set of rules of the form

$$\frac{S}{S'}$$

where S, S' are sets of statements. Such a rule is to be interpreted as: if every statement in S, the set of premises, can be derived, then any statement in S', the set of conclusions, may be derived.

(R1) (*Equality*)

$$\frac{t = u}{t \sqsubseteq u,\ u \sqsubseteq t} \qquad \frac{t \sqsubseteq u,\ u \sqsubseteq t}{t = u}$$

(R2) (*Partial Order*)

$$\frac{t \sqsubseteq u,\ u \sqsubseteq r}{t \sqsubseteq r} \qquad \frac{}{t \sqsubseteq t}$$

(R3) (*Substitutivity*)

(i) $\dfrac{t \sqsubseteq u}{tp \sqsubseteq up}$ (ii) $\dfrac{t \sqsubseteq u}{\mathrm{rec}\ x.t \sqsubseteq \mathrm{rec}\ x.u}$

(iii) $\dfrac{t_i \sqsubseteq u_i,\ 1 \leqslant i \leqslant k}{\mathrm{op}(t_1, \ldots, t_k) \sqsubseteq \mathrm{op}(u_1, \ldots, u_k)}$ for every $\mathrm{op} \in \Sigma_k$

(R4) (*General Induction*)

$$\frac{d \sqsubseteq u, \forall d \in \mathrm{FIN}(t)}{t \sqsubseteq u}$$

We write $A \vdash t \sqsubseteq u$ if $t \sqsubseteq u$ can be derived from the set of axioms A using the rules (R1)-(R4). Note that (R4) is an infinitary rule since it has an infinite number of premises. Recursively enumerable proof systems can be obtained by replacing it by a finitary form of induction, such as Fixpoint Induction or Scott Induction [28]. Indeed, these may be derived from (R4). As a simple example we derive Fixpoint Induction:

(FP) $$\frac{t[u/x] \sqsubseteq u}{\mathrm{rec}\ x.t \sqsubseteq u}$$

Lemma 3.2.2. *If A contains the axioms* (Ω2), (REC1), *then* (FP) *is a derived rule in the system with rules* (R1)-(R4) *and axioms A.*

Proof. Let r denote rec $x.t$. We first show that $A \vdash r^n \sqsubseteq u$ for every $n \geqslant 0$.

(i) $n = 0$: $A \vdash r^0 \sqsubseteq u$ follows from (Ω2).

(ii) $n = k + 1$: We assume $A \vdash r^k \sqsubseteq u$.

Then

$$\begin{aligned} A \vdash r^{k+1} &= t[r^k/x] \\ &\sqsubseteq t[u/x] && \text{by repeated application of (R3) and induction on } k \\ &\sqsubseteq u && \text{from the premises.} \end{aligned}$$

Also note that if $t < u$, then $A \vdash t \sqsubseteq u$ since A contains the axioms which generate $<$.

Now let $d \in \mathrm{FIN}(r)$. Then, from Lemma 3.2.1, $A \vdash d \sqsubseteq r^n$ for some n and therefore $A \vdash d \sqsubseteq u$. Since this is true for every $d \in \mathrm{FIN}(r)$, we may apply (R4) to obtain $A \vdash r \sqsubseteq u$. □

Lemma 3.2.3. *If A contains the axioms* (Ω2), (REC1), *then the rule* (R3)(ii) *is derivable.*

Proof. We assume that $A \vdash t \sqsubseteq u$. Let ρ be the substitution defined by

$$\rho(x) = \text{rec } x.u$$

$$\rho(y) = y, \quad y \neq x.$$

Then applying the first part of (R3) we get $A \vdash t\rho \sqsubseteq u\rho$, i.e.,

$$A \vdash t[\text{rec } x.u/x] \sqsubseteq u[\text{rec } x.u/x]$$

$$\sqsubseteq \text{rec } x.u, \quad \text{using (REC1).}$$

Now applying (FP) we get $A \vdash \text{rec } x.t \sqsubseteq \text{rec } x.u$. □

We have, however, decided to include this rule as part of (R3) for the sake of clarity. We now state the main results of this paper.

Theorem 3.2.4. *For $i = 1, 2, 3$, $A_i \vdash t \sqsubseteq u$ $(t = u)$ implies $t \sqsubseteq_i^c u$ $(t \simeq_i^c u)$.*

Theorem 3.2.5. *For $i = 1, 2, 3$, and closed terms p, q, $p \sqsubseteq_i^c q$ implies $A_i \vdash p \sqsubseteq q$.*

The next section is entirely devoted to the proofs of these results.

4. Proof of completeness theorems

4.1

In this section we derive the soundness results, i.e., Theorem 3.2.4. The main difficulty lies in justifying (R4) and to do so we need some lemmas relating the behaviour of terms to the behaviour of their finite approximants. Before tackling this problem we concentrate on proving the axioms and the remainder of the rules sound. We write $t \sqsubseteq_i u$ if $t \sqsubseteq u$ can be derived from the set of axioms A_i using rules (R1), (R2) and (R3)(i), (iii).

Lemma 4.1.1. (a) *$p \sqsubseteq_3^+ q$ implies $S(p) \subseteq S(q)$.*

(b) *$p\Downarrow$, $p \sqsubseteq_2^+ q$ implies* (i) *$S(q) \subseteq S(p)$,*

(ii) *$q \xrightarrow{\tau}$ implies $p \xrightarrow{\tau}$.*

Proof. (a) Obvious.

(b) (i) Suppose $\lambda \in S(q)$, $\lambda \notin S(p)$. Then p **must satisfy** $(\bar{\lambda} + \tau\omega)$ whereas $q \mid (\bar{\lambda} + \tau\omega) \rightarrow q' \mid \text{NIL}$ for some q', which can never lead to a successful state.

(ii) Suppose $q \xrightarrow{\tau}$, $p \not\xrightarrow{\tau}$. Then, for any λ not appearing in p or q, $\lambda + p$ **must satisfy** $\bar{\lambda}\omega$ whereas $\lambda + q \mid \bar{\lambda}\omega \rightarrow q' \mid \bar{\lambda}\omega$, which again can never lead to a successful state. □

Lemma 4.1.2. *If $\text{op} \in \Sigma_k$, $t_j \sqsubseteq_i^+ u_j$, $1 \leqslant j \leqslant k$, then $\text{op}(t_1, \ldots, t_k) \sqsubseteq_i^+ \text{op}(u_1, \ldots, u_k)$.*

Proof. It is sufficient to prove it for closed terms. We examine each of the operators in turn, the cases NIL, Ω being trivial.

(a) $p \sqsubseteq_i^+ q \Rightarrow p|r \sqsubseteq_i^+ q|r$, $i = 1, 2, 3$.

($i = 3$) It is sufficient to show $r'+(p|r)$ **may satisfy** o implies $r'+(q|r)$ **may satisfy** o.

From the hypothesis $(r'+(p|r))|o \rightarrow^* p_1$, $p_1 \in$ ***Success***. If this computation does not involve the subterm $p|r$, then we have immediately $r'+(q|r)|o \rightarrow^* p_1'$ for some $p_1' \in$ ***Success***. Otherwise, $(p|r)|o \rightarrow^* p_1$. Therefore, $p|(r|o) \rightarrow^* p_1'$ for some $p_1' \in$ ***Success***. Since $p \sqsubseteq_3 q$, $q|(r|o) \rightarrow^* q_1' \in$ ***Success*** since $(r|o)$ is an observer. Therefore, $(q|r)|o \rightarrow^* q_1$ for some $q_1 \in$ ***Success***.

($i = 2$) It is sufficient to show $r'+(p|r)$ **must satisfy** o implies $r'+q|r$ **must satisfy** o.

We distinguish two cases.

Case 1. $p|r$ **must satisfy** o.

This implies p **must satisfy** $r|o$ and, since $p \sqsubseteq_2 q$, q **must satisfy** $r|o$. This implies $q|r$ **must satisfy** o. It follows that $r'+q|r$ **must satisfy** o since every computation from $(r'+q|r)|o$ which starts with an action or communication from r' is also a computation from $(r'+p|r)|o$.

Case 2. $\neg p|r$ **must satisfy** o.

If $p|r \xrightarrow{\tau}$, then $r+p|r$ **must satisfy** o implies $p|r$ **must satisfy** o. So we may assume $p|r \xrightarrow{\tau}$. Therefore, $\lambda + p$ **must satisfy** $\bar{\lambda}\omega + r$ where λ does not appear in p, r. Since $p \sqsubseteq_2^+ q$, $\lambda + q$ **must satisfy** $\bar{\lambda}\omega + r$, i.e., $q|r \xrightarrow{\tau}$. Now by a simple case analysis on r' we can establish that $r'+q|r$ **must satisfy** o.

($i = 1$) Follows from the previous two cases.

(b) $p \sqsubseteq_i^+ q$ implies $\mu p \sqsubseteq_i^+ \mu q$, $i = 1, 2, 3$.

We show the case $i = 2$ only. The case $i = 3$ is similar and the case $i = 1$ follows as usual.

It is sufficient to prove μp **must satisfy** o implies μq **must satisfy** o.

(i) $\mu = \tau$. In this case p **must satisfy** o and, since $p \sqsubseteq_2 q$, q **must satisfy** o. Therefore, μq **must satisfy** o.

(ii) $\mu = \lambda$. In this case, p **must satisfy** o' for o' such that $o \overset{\lambda}{\Rightarrow} o'$. Since $p \sqsubseteq_2 q$, q **must satisfy** o'. By analysing whether or not $o \overset{\tau}{\Rightarrow} o' \xrightarrow{\lambda}$, it is sufficient to show that λq **must satisfy** o.

(c) $p \sqsubseteq_i^+ q$ implies $p[S] \sqsubseteq_i^+ q[S]$, $i = 1, 2, 3$. We show the case $i = 3$ only.

It is sufficient to show $p[S]$ **may satisfy** o implies $q[S]$ **may satisfy** o. The partial permutation is defined on A only but we may extend it by $S(\omega) = \omega$. Define S' by $S'(\mu) = \mu'$ if $S(\mu') = \mu$. Then S' is well defined and $p[S]$ **may satisfy** o if and only if p **may satisfy** $o[S']$. The result now follows by applying the fact that $p \sqsubseteq_1 q$.

(d) $p \sqsubseteq_i^+ q$ implies $p+r \sqsubseteq_i^+ q+r$, $i = 1, 2, 3$.

($i = 3$) It is necessary to show $r'-p-r$ **may satisfy** o implies $r'+q+r$ **may satisfy** o.

This is left to the reader.

($i=2$) We show $r'+p+r$ **must satisfy** o implies $r'+q+r$ **must satisfy** o. If $p|o \xrightarrow{\tau}$, then, from Lemma 4.1.1(b), $q|o \xrightarrow{\tau}$ and it immediately follows that $r'+q+r$ **must satisfy** o. Otherwise we have $r'+p$ **must satisfy** o. Since $p \sqsubseteq_i^+ q$, $r'+q$ **must satisfy** o and therefore $r'+q+r$ **must satisfy** o.

($i=1$) Follows from the previous two cases. □

Note that the proof of this lemma was facilitated by the generality of the observers.

Lemma 4.1.3. *If* $\mathrm{Der}_\mu(p)=\mathrm{Der}_\mu(q)$ *for every* $\mu \in A$, *and* $p\downarrow$ *if and only if* $q\downarrow$, *then* $p \simeq_i^+ q$, $i=1, 2, 3$.

The proof is left to the reader.

Lemma 4.1.4. *If* $t \sqsubseteq u$ ($t=u$) *is an instance of an axiom from* A_i, $i=1, 2$ *or* 3, *then* $t \sqsubseteq_i^+ u$ ($t \simeq_i^+ u$).

Proof. It is sufficient to consider closed terms. If it is an instance of (A1)-(A4), (S1)-(S3), (C1), (Ω1) or (REC1), then we can apply the previous lemma. Consider an instance of the axiom (E2), $\tau p+\tau q \sqsubseteq p$, and suppose $\tau p+\tau q$ **must satisfy** o. It follows that τp **must satisfy** o and therefore p **must satisfy** o. The soundness of (F1) is similar. So the only remaining axioms are (N1)-(N4).

(N1) Let p, q denote $r+\mu p_1+\mu p_2$, $r+\mu(\tau p_1+\tau p_2)$ respectively. Then it is easy to see that $\exists c \in \mathrm{Comp}(o, p)$ s.t. $c\Uparrow$ if and only if $\exists c' \in \mathrm{Comp}(o, q)$ s.t. $c'\Uparrow$ and $o|p \overset{\omega}{\Rightarrow}$ if and only if $o|q \overset{\omega}{\Rightarrow}$. This is sufficient to establish that $p \simeq_1 q$.

(N2) Let p, q denote $r+p_1+\tau p_2$, $r+\tau(p_1+p_2)$. Let $c \in \mathrm{Comp}(o, q)$. Then either $c' \in \mathrm{Comp}(o, p)$ where c' differs from c in at most the first two terms or else c is of the form $\langle o|q, o|p_1+p_2\rangle$. Then $\langle o|p, o|p_2\rangle \in \mathrm{Comp}(o, p)$ and these remarks are sufficient to establish that $p \sqsubseteq_1 q$.

(N3) Let p, q denote $r'+\mu p_1+\tau(\mu p_2+r)$, $r+\tau(\mu p_1+\mu p_2+r)$ respectively. It is trivial to show that $o|p \overset{\omega}{\Rightarrow}$ if and only if $o|q \overset{\omega}{\Rightarrow}$ and $\mathrm{Comp}(o, p)$ contains an infinite computation if and only if $\mathrm{Comp}(o, p)$ does. Now suppose $o|p \overset{\tau}{\Rightarrow} s$ where $s\uparrow$ or $s \in$ Fail. Then either $o|q \overset{\tau}{\Rightarrow} s$ or else s is of the form $o'|(\mu p_2+r)$. In the latter case, $o|q \overset{\tau}{\Rightarrow} o'|(\mu p_1+\mu p_2+r)$, i.e., $\perp \in R(o, p)$. A similar analysis will show that $o|q \overset{\tau}{\Rightarrow} s$, $s\uparrow$ or $s \in$ Fail implies $\perp \in R(o, p)$. It follows that $p \simeq_1 q$.

(N4) Similar to (N2). □

Proposition 4.1.5. (a) $t \sqsubseteq_i u$ *implies* $t \sqsubseteq_i^+ u$.

(b) $t =_i u$ *implies* $t \simeq_i u$.

Proof. The proof follows from the previous lemma and Lemma 4.1.2. The soundness of rules (R1), (R2) is immediate. □

We now concentrate on justifying (R4).

Lemma 4.1.6. *$p < q$ implies $p \sqsubseteq_i^+ q$.*

Proof. From the previous proposition, $\sqsubseteq_i^+$ satisfies ($\Omega 2$) and (REC1) and the three implications in the definition of $<$. Since $<$ is the least such relation, it follows that $p < q$ implies $p \sqsubseteq_i^+ q$. □

Lemma 4.1.7

(a) *$p \xrightarrow{\mu} q$, $d < q$ implies $\exists e < p$ such that $e \xrightarrow{\mu} d$.*

(b) *$p \xrightarrow{\mu} q$, $p < p'$ implies $p' \xrightarrow{\mu} q'$ for some q' such that $q < q'$.*

Proof. In each case the proof is by induction on the proof of $p \xrightarrow{\mu} q$. □

Lemma 4.1.8. *$d \sqsubseteq_i p$ implies that, for some $e \in \mathrm{FIN}(p)$, $d \sqsubseteq_i e$.*

Proof. By induction on the number of times (REC1) is used in the proof of $d \sqsubseteq_i p$. □

Proposition 4.1.9. (a) *p* **may satisfy** *o implies d* **may satisfy** *o for some $d \in \mathrm{FIN}(p)$.*

(b) *p* **must satisfy** *o implies d* **must satisfy** *o for some $d \in \mathrm{FIN}(p)$.*

Proof. (a) From the hypothesis, $p|o \rightarrow^* r \xrightarrow{\omega} r'$. Since $\Omega < r'$, we may apply Lemma 4.1.7(a) to find some $e < r$ such that $e \xrightarrow{\omega}$. By repeated application of this lemma we have some $d' < p|o$ such that $d' \rightarrow^* e$. Now d' must be of the form $d_1|d_2$, where $d_1 < p$, $d_2 < o$. By applying part (b) of this lemma sufficiently often we obtain $d_1|o \rightarrow^* r'' \xrightarrow{\omega}$, i.e., d_1 **may satisfy** o.

(b) The proof is easy using the notion of head normal form. Since this has yet to be done, we relegate the proof to Appendix A. □

Proposition 4.1.10. *If $d \sqsubseteq_i q$ for every $d \in \mathrm{FIN}(p)$, then $p \sqsubseteq_i q$.*

Proof. Apply the previous proposition. □

Corollary 4.1.11. *For $i = 1, 2, 3$, $A_i \vdash t \sqsubseteq u$ ($t = u$) implies $t \sqsubseteq_i^+ u$ ($t =_i^+ u$).*

Proof. It is a simple matter to show that the previous proposition justifies (R4) for $\sqsubseteq_i^+$ for closed terms and therefore open terms. The remaining rules were treated in Proposition 4.1.5 apart from (R3)(ii) which can be derived from (R4) as was pointed out in Lemma 3.2.3. □

Corollary 4.1.12 (Theorem 2.2.3). *$t \sqsubseteq_i^+ u$ if and only if $t \sqsubseteq_i^s u$.*

Proof. If $t \sqsubseteq_i^s u$, then obviously $t \sqsubseteq_i^+ u$. Conversely, suppose $t \sqsubseteq_i^+ u$. Then for any context $C[\ ]$ we can apply (R3) to prove that $C[t] \sqsubseteq_i^+ C[u]$. This can be proven by structural induction on $C[\ ]$. It follows that $C[t] \sqsubseteq_i C[u]$. Therefore, $t \sqsubseteq_i^s u$. □

Corollary 4.1.13 (Theorem 3.2.4). *For $i = 1, 2, 3$, $A_i \vdash t \sqsubseteq u$ $(t = u)$ implies $t \sqsubseteq_i^c u$.*

4.2

The proof of completeness depends very heavily on the existence of normal forms. We consider four kinds and this section is devoted entirely to them.

If $\mathscr{L}$ is a set of sets let $A(\mathscr{L}) = \{x \mid x \in X$, for some $s \in \mathscr{L}\}$. $\mathscr{L}$ *implies* a set S if there exists a set $S' \in \mathscr{L}$ such that $S' \subseteq S \subseteq A(\mathscr{L})$. $\mathscr{L}$ is said to be *saturated* if $S \in \mathscr{L}$ whenever $\mathscr{L}$ implies S. For example, $\{\{a\}, \{a, b\}\}$ is saturated whereas $\{\{a\}, \{b\}\}$ is not. We only use saturated $\mathscr{L}$ which are finite and contain finite sets. For such $\mathscr{L}$ being saturated is equivalent to the following two conditions:

(i) $X, Y \in \mathscr{L}$ implies $X \cup Y \in \mathscr{L}$,

(ii) $X, Y \in \mathscr{L}$, $X \subseteq Z \subseteq Y$ implies $Z \in \mathscr{L}$.

Definition 4.2.1 (*normal form* (*nf*)). (i) If $\mathscr{L}$ is a nonempty finite saturated set of finite subsets of $\Delta \cup \bar{\Delta}$, then $\sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda p_\lambda$ is in *τnf* provided each term p_λ is

(a) in *τnf*, or

(b) in *λnf* and $p_\lambda \Uparrow$.

(ii) $\sum_{\lambda \in L} \lambda p_\lambda \{+ \Omega\}$ is in *λnf* if each term p_λ satisfies (a) or (b) above, and

(c) whenever Ω is a summand it is also a summand of each p_λ.

(iii) p is in *normal form* (*nf*) if it is in τnf or λnf.

Note that from the definition if p is a nf and $p \Uparrow$, then p is a λnf. Also every normal form is a finite term and if p is a normal form and $p \rightarrow^* \xrightarrow{\lambda} p'$, then p' is also a normal form.

Examples. (i) $\lambda(\alpha + \beta) + \Omega$ is not in nf because (c) is not satisfied and the subterm $(\alpha + \beta)$ is not in τnf.

(ii) $\tau\lambda_1 \tau\alpha + \tau\lambda_2 \tau\beta$ is not in nf because $\{\{\lambda_1\}\{\lambda_2\}\}$ is not saturated. There is no subterm corresponding to the set of labels $K = \{\lambda_1, \lambda_2\}$. The normal form corresponding to this term will be $\tau\lambda_1 \tau\alpha + \tau\lambda_2 \tau\beta + \tau(\lambda_1 \tau\alpha + \lambda_2 \tau\beta)$.

(iii) $\tau\alpha\tau\beta + \tau(\alpha\tau\gamma + \lambda\tau\beta)$ is not in normal form. If p is a normal form and for any λ, $p \rightarrow^* \xrightarrow{\lambda} p_1$, $p \rightarrow^* \xrightarrow{\lambda} p_2$, then p_1 and p_2 must be identical. This example violates this requirement. Because of this property we can use a suggestive notation: for a normal form p we may let p_λ denote the unique subterm (if it exists) such that $p \rightarrow^* \xrightarrow{\lambda} p_\lambda$.

(iv) NIL is a λnf. In the definition we merely let $L = \emptyset$. Similarly, τNIL is a τnf and Ω is a λnf. Note that here we have again used the brackets { } to denote that Ω is an optional summand in a λnf.

Definition 4.2.2 (*strong normal form* (*snf*)). (i) Ω is a *snf*.

(ii) if p is a nf other than Ω, then p is a *snf* whenever

(a) each p_λ is a snf,

(b) p does not contain Ω as a summand.

If we look upon terms as trees, then a snf is a normal form which only contains Ω as a leaf. We will use the notation τsnf, λsnf in the obvious way.

Definition 4.2.3 (*weak normal form* (*wnf*)). (i) Ω is a *wnf*.
(ii) if p_λ is a wnf for each $\lambda \in L$, then $\sum_{\lambda \in L} \lambda p_\lambda + \Omega$ is a *wnf*.

Later we will see that a wnf corresponds to a prefix-closed set of strings from $\Delta \cup \bar{\Delta}$.

Definition 4.2.4 (*head normal form* (*hnf*)). (i) $\sum_{\lambda \in L} \lambda p_\lambda$ is a λ*hnf*.
(ii) $\sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda p_\lambda$ is a τ*hnf* if the set $\mathscr{L}$ is saturated and nonempty.

Note that if p is a hnf, then $p\Downarrow$. So, for example, Ω is not a hnf. Our first task is to show that every finite closed term can be transformed to a normal form using the axioms from A_1.

Lemma 4.2.5. *If p in nf, then there exists a λnf n such that $p + \Omega =_1 n$ and $n\uparrow$.*

Proof. (a) Suppose $p =_1 \sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda p_\lambda$.
Then

$$p + \Omega =_1 \sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda(p_\lambda + \Omega) + \Omega \quad \text{using (D3).}$$

By induction there exists a λnf n_λ such that $p_\lambda + \Omega =_1 n_\lambda$. Therefore,

$$p + \Omega =_1 \sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda n_\lambda + \Omega$$

$$=_1 \sum_{\lambda \in L} \lambda n_\lambda + \Omega \quad \text{using (D1).}$$

(b) The proof for λnf is similar. □

Lemma 4.2.6. *If p, q are in nf, then there exists a normal form n such that $\tau p + \tau q =_1 n$. Moreover, n is a τnf or else $n\Uparrow$.*

Proof. We use induction on the size of p and q. There are four cases depending on what kind of nf p and q are.

Case (i) p is $\sum \tau p_i$, q is $\sum \tau q_j$. Then $\tau p + \tau q =_1 \sum \tau p_i + \sum \tau q_j$ using (D2).

Let r denote the right-hand side. Now r may not be in normal form for various reasons. For example, it may be that $r \to^* \xrightarrow{\lambda} r_1$, $r \to^* \xrightarrow{\lambda} r_2$ such that r_1, r_2 are not syntactically identical. Let $N(r)$ be the number of such pairs. We show, by induction on $N(r)$ that r can be transformed into an r' such that $N(r') = \emptyset$.

If r_1, r_2 is such a pair, then the axioms (A1)-(A4) may be used to rewrite r as follows:

$$r =_1 \tau(\lambda r_1 + r_1') + \tau(\lambda r_2 + r_2') + r'$$

$$=_1 \lambda r_1 + \tau(\lambda r_1 + r_1') + \lambda r_2 + \tau(\lambda r_2 + r_2') + r' \quad \text{using (D8)}$$

$$=_1 \tau(\lambda r_1 + \lambda r_2 + r_1') + \tau(\lambda r_1 + \lambda r_2 + r_2') + r' \quad \text{using (N3) twice}$$

$$=_1 \tau(\lambda(\tau r_1 + \tau r_2) + r_1') + \tau(\lambda(\tau r_1 + \tau r_2) + r_2') + r' \quad \text{using (N1) twice.}$$

Now, by induction, $\tau r_1 + \tau r_2$ has a normal form n of the required form. Therefore $r =_1 \tau(\lambda n + r_1') + \tau(\lambda n + r_2') + r'$.

If s denotes the right-hand side, then $N(s) < N(r)$. So by induction we may now assume that r is such that $N(r) = 0$. Therefore, r may now be written as $\sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L_i} \lambda r_\lambda$. Moreover, each r_λ is in τnf or is in λnf and diverges. So if this term is not in nf, it must be that $\mathscr{L}$ is not saturated. Let K be such that $L_j \subseteq K \subseteq A(\mathscr{L})$. We show by induction on the size of K that $r =_1 r + \tau \sum_{\lambda \in K} \lambda n_\lambda$.

If $K = L_j$, then the claim is immediate. Otherwise, K may be written as $K_1 \cup \{\lambda_0\}$, λ_0 belonging to some $L_i \in \mathscr{L}$. We may assume $r =_1 r + \tau \sum_{\lambda \in K_1} \lambda n_\lambda$ and let r_1 denote $\sum_{\lambda \in K_1} \lambda n_\lambda$. Then

$$r =_1 r + \tau r_1 + \tau(r' + \lambda_0 n_{\lambda_0}) \quad \text{using (A1)-(A4)}$$

$$=_1 r + \tau r_1 + \tau(r' + \lambda_0 n_{\lambda_0}) + \tau(r_1 + r' + \lambda_0 n_{\lambda_0}) \quad \text{using (D5)}$$

$$=_1 r + \tau r_1 + \tau(r' + \lambda_0 n_{\lambda_0}) + \tau(r_1 + r' + \lambda_0 n_{\lambda_0}) + \tau(r_1 + \lambda_0 n_{\lambda_0}) \quad \text{using (D6)}$$

$$=_1 r + \tau(r_1 + \lambda_0 n_{\lambda_0}) \quad \text{as required.}$$

Therefore, by induction we may assume that

$$r =_1 r + \tau \sum_{\lambda \in K} \lambda n_\lambda \quad \text{for every such } K.$$

Now by systematically applying this result, r may be transformed into a term of the form $\sum_{L \in \mathscr{L}} \tau \sum_{\lambda \in L} \lambda n_\lambda$ where $\mathscr{L}$ is saturated, and this term is in τnf.

Case (ii) p is $\sum \lambda_i p_i$, q is $\sum \tau q_j$. Then $\tau p + \tau q =_1 \tau p + q$ using (D2) and we proceed as in part (i).

Case (iii) p is $\sum \lambda_i p_i$, q is $\sum \lambda_j q_j$. Then $\tau p + \tau q$ is an instance of (i).

Case (iv) p is $\sum_{\lambda \in L} \lambda p + \Omega$. Then $\tau p + \tau q = p + q$ by (D1), (D3).

From Lemma 4.2.5 there exists a λnf n such that $q + \Omega =_1 n$. Let n be $\Omega + \sum_{\lambda \in K} \lambda n_\lambda$. Then

$$p + n =_1 \Omega + \sum_{\lambda \in L/K} \lambda p_\lambda + \sum_{\lambda \in K/L} \lambda n_\lambda + \sum_{\lambda \in L \cap K} \lambda(\tau p_\lambda + \tau q_\lambda) \quad \text{using (N1).}$$

By induction there exists a normal form r such that $r_\lambda =_1 \tau p_\lambda + \tau q_\lambda$. Therefore,

$$p + n =_1 \Omega + \sum_{\lambda \in L/K} \lambda(p_\lambda + \Omega) + \sum_{\lambda \in K/L} \lambda(n_\lambda + \Omega)$$

$$+ \sum_{\lambda \in L \cap K} \lambda(r_\lambda + \Omega) \quad \text{using (D3).}$$

We now apply Lemma 4.2.5 to obtain the required normal form. □

Lemma 4.2.7. *If p, q are in nf, then there exists a nf n such that $p+q =_1 n$.*

Proof. We use induction on the size of p, q. There are four cases depending on the form of p, q.

Case (a) p is $\sum_{\lambda\in L}\lambda p_\lambda + \Omega$. Then $p+q =_1 \tau p+\tau q$, using (D1), and we may apply Lemma 4.2.6.

Case (b) p and q are in τnf. Then $p+q =_1 \tau p+\tau q$, using (D2) and we may again apply Lemma 4.2.6.

Case (c) p is $\sum_{\lambda\in L}\lambda p_\lambda$, q is $\sum_{i\in I}\tau q_i$. Then $p+q =_1 \tau(p+q_1)+\tau q$ using (D4).

By induction, $p+q_1$ may be transformed into a normal form n and, by Lemma 4.2.6, $\tau n+\tau q$ may be transformed into a normal form.

Case (d) The only remaining case is when p is $\sum_{\lambda\in L}\lambda p_\lambda$, q is $\sum_{\lambda\in K}\lambda q_\lambda$. Then

$$p+q =_1 \sum_{\lambda\in L\setminus K}\lambda p_\lambda + \sum_{\lambda\in K\setminus L}\lambda q_\lambda + \sum_{\lambda\in K\cap L}\lambda(\tau p_\lambda+\tau q_\lambda) \quad \text{using (N1).}$$

From the previous lemma each $\tau p_\lambda+\tau q_\lambda$ can be transformed into a nf of the required form. □

Corollary 4.2.8. *If p_i are in nf, $1\leq i\leq k$, then there exists a nf n such that $\sum_{1\leq i\leq k}\tau p_i =_1 n$, and n is either in τnf or else $n\Uparrow$ and is in λnf.*

Proof. The proof follows by induction on k.

Basis. $k=1$. If p_1 is in λnf and $p_1\Downarrow$, then τp_1 is already in τnf. Otherwise, $\tau p_1 =_1 p_1$, using (D1) or (D2).

Induction Step. $k=m+1$.

$$\sum_{1\leq i\leq k}\tau p_i =_1 \sum_{1\leq i\leq m}\tau p_i+\tau p_k$$

$$=_1 n_1+\tau p_k \quad \text{by induction.}$$

If n_1 is in τnf, we can apply Lemma 4.2.6. Otherwise, $n_1+\tau p_k =_1 n_1+p_k$, using (D1) and we may apply Lemma 4.2.7. □

Corollary 4.2.9. *If p_i are in nf, $1\leq i\leq k$, then there exists a nf n such that $\sum_{1\leq i\leq k}p_i =_1 n$.*

Proof. The proof follows by induction on k. □

Proposition 4.2.10. *For every finite closed term d there exists a normal form* nf(d) *such that $d =_1$ nf(d).*

Proof. By applying rules (S1)-(S3), (C1) and (Ω1) we may eliminate all occurrences of $|$ and $[S]$. So without loss of generality we may assume that d does not contain these operators and may be written as $\sum_{i\in I}\lambda_i d_i+\sum_{j\in J}\tau e_j$.

Now by induction we may assume all d_i, e_j are in nf. Using (D7) if necessary each $\lambda_i d_i$ may be considered to be in nf. So by Corollary 4.2.9 there exists a normal form n_1 such that $n_1 =_1 \sum_{i \in I} \lambda_i d_i$. Similarly by Corollary 4.2.8 there exists a normal form n_2 such that $n_2 =_1 \sum_{j \in J} \tau e_j$. Therefore apply Corollary 4.2.9 again and we get a normal form n such that

$$d =_1 n_1 + n_2 =_1 n. \quad \square$$

This proposition enables us to derive similar results on strong and weak normal forms.

Proposition 4.2.11. *For every finite closed term d there exists a strong normal form* $\mathrm{snf}(d)$ *such that* $d =_2 \mathrm{snf}(d)$.

Proof. By the previous proposition we may assume that d is in normal form.

(a) d is $\sum \lambda d_\lambda \{+\Omega\}$. If Ω is a summand, then $d =_2 \Omega$ from (E2). Otherwise, by induction we may assume that each d_λ is in snf. Then, using (D7), $d =_2 \sum \{\lambda d_\lambda \mid d_\lambda$ in τsnf or $d_\lambda \Uparrow\} + \sum \{\lambda \tau d_\lambda \mid d_\lambda$ is in λnf and $d_\lambda \Downarrow\}$.

(b) If d is in τnf, the proof is similar. $\square$

Proposition 4.2.12. *For every finite closed term d there exists a weak normal form w such that* $d + \Omega =_3 w$.

Proof. We may assume d is in nf.

(a) d is $\sum \lambda d_\lambda \{+\Omega\}$. Then

$$d + \Omega =_1 \sum \lambda d_\lambda + \Omega$$

$$=_1 \sum \lambda (d_\lambda + \Omega) + \Omega \quad \text{using (D3).}$$

By induction each $(d_\lambda + \Omega)$ has a weak normal form and the result thus follows.

(b) d is $\sum_{L \in \mathcal{L}} \tau \sum_{\lambda \in L} \lambda d_\lambda$. Then

$$d + \Omega = \sum_{\lambda \in A(\mathcal{L})} d_\lambda + \Omega \quad \text{using (D1).}$$

We may now apply part (a). $\square$

Corollary 4.2.13. *For every finite closed term d there exists a weak normal form* $\mathrm{wnf}(d)$ *such that* $d =_3 \mathrm{wnf}(d)$.

Proof. Apply the previous proposition and (F2). $\square$

Weak normal forms may also be considered as prefix-closed sets of strings from $\Delta \cup \bar{\Delta}$. If $s \in (\Delta \cup \bar{\Delta})^*$, then we also use s to denote its representation as a term: the representation of ε the empty string is Ω and that of λs is λt where t is the term representing s.

Lemma 4.2.14. *If d is a weak normal form, then there exists a prefix-closed set of strings s_i, $1 \leq i \leq n$, such that $d =_1 \Omega + \sum s_i$.*

Proof. We use induction on d.

(i) d is Ω. Immediate.

(ii) d is $\sum_{\lambda \in L} \lambda d_\lambda + \Omega$. By induction $d_\lambda =_1 \Omega + \sum_{i \in I_\lambda} s_i$, where $\{s_i \mid i \in I_\lambda\}$ is prefix-closed. Therefore,

$$d =_1 \sum_{\lambda \in L} \lambda\left(\sum_{i \in I_\lambda} s_i + \Omega\right) + \Omega$$

$$=_1 \sum_{\lambda \in L} \lambda\left(\sum_{i \in I_\lambda} s_i + \Omega\right) + \sum_{\lambda \in L} \lambda\Omega + \Omega$$

since $\lambda(X + \Omega) =_1 \lambda(X + \Omega) + \lambda\Omega$

$$=_1 \sum_{\lambda \in L} \sum_{i \in I_\lambda} \lambda s_i + \sum_{\lambda \in L} \lambda\Omega + \Omega \quad \text{from (D10).} \qquad \square$$

It remains to consider head normal forms which unfortunately must be treated in much the same way as normal forms. Rather than giving the entire proofs of the lemmas we merely state the required results.

Lemma 4.2.15. *If p_i are in head normal form, $1 \leq i \leq k$, then there exist head normal forms h_1, h_2 such that*

$$\sum_{1 \leq i \leq k} \tau p_i =_1 h_1, \qquad \sum_{1 \leq i \leq k} p_i =_1 h_2.$$

Proof. Similar to the proofs of Lemmas 4.2.6–4.2.7 and Corollaries 4.2.8–4.2.9. $\square$

In the next proposition we use $p = q$ to denote that "$p = q$" can be derived from the axioms A_1, using all the rules (R1)–(R4). Unfortunately, $=_1$ is not sufficient since we need to be able to rewrite $\mathrm{rec}\, x.t$ as $t[\mathrm{rec}\, x.t/x]$. We have $t[\mathrm{rec}\, x.t/x] \sqsubseteq_1 \mathrm{rec}\, x.t$ but not the converse. However, $\mathrm{rec}\, x.t = t[\mathrm{rec}\, x.t/x]$.

Proposition 4.2.16. *If $p\Downarrow$, then there exists a head normal form* $\mathrm{hnf}(p)$ *such that* $p = \mathrm{hnf}(p)$.

Proof. We use induction on the size of $\mathrm{Der}_\tau(p)$. So we may assume the result for every p' such that $p \xrightarrow{} p'$. If $p\Downarrow$, *then* $p\downarrow$ and we now use induction to prove this.

(i) $\lambda p'$. By definition this is in hnf.

(ii) $\tau p'$. Then $p'\downarrow$. By induction p' has a hnf, $\mathrm{hnf}(p')$. If $\mathrm{hnf}(p')$ is in λhnf, then $\tau\mathrm{hnf}(p')$ is in τhnf. Otherwise

$$p = \tau\mathrm{hnf}(p')$$

$$= \mathrm{hnf}(p') \quad \text{using (D2).}$$

(iii) p_1+p_2. Then $p_1\downarrow$, $p_2\downarrow$ and so by induction both hnf(p_1) and hnf(p_2) exist. Therefore

$$p_1+p_2 = \text{hnf}(p_1)+\text{hnf}(p_2)$$
$$= h \quad \text{for some hnf } h, \text{ using the previous lemma.}$$

(iv) $p'[S]$. Then $p'\downarrow$ and so by induction hnf(p') exists. We may now use (S1)-(S3) to transform hnf(p')[S] into hnf.

(v) rec $x.t$. Then $t[\text{rec } x.t/x]\downarrow$ and by induction there exists a hnf h such that $h = t[\text{rec } x.t/x]$. The result now follows since rec $x.t = t[\text{rec } x.t/x]$.

(vi) $p_1|p_2$. Then $p_1\downarrow$, $p_2\downarrow$ and by induction hnf(p_1), hnf(p_2) exist. There are three cases depending on their form.

(a) hnf(p_1) is $\sum \lambda p_\lambda$, hnf(p_2) $=\sum \gamma q_\lambda$. Then applying (C1) we get

$$p_1|p_2 = \sum \lambda(p_\lambda|\text{hnf}(p_2)) + \sum \gamma(\text{hnf}(p_1)|q_\gamma) + \sum_{\lambda=\bar{\gamma}} \tau(p_\lambda|q_\gamma).$$

Since $p_1|p_2 \rightarrow p_\lambda|q_\gamma$ whenever $\lambda = \bar{\gamma}$ we may assume by induction that each of these terms have hnfs. The result now follows from Lemma 4.2.14.

(b) hnf(p_1) is in λhnf, hnf(p_2) is in τhnf.

(c) Both hnf(p_1) and hnf(p_2) are in τhnf.

These cases are similar to (a). □

4.3

In this section we apply the normal form results to prove completeness of the proof systems. We first need some lemmas.

Lemma 4.3.1. (a) *If $p \sqsubseteq_i q$ and $p \xrightarrow{\tau}$, then $p \sqsubseteq_i^+ q$.*
(b) *If $p \sqsubseteq_i q$ and $p\Uparrow$, then $p \sqsubseteq_i^+ q$.*

Proof. Now $p \sqsubseteq_3 q$ implies $p \sqsubseteq_3^+ q$ so the only nontrivial case is $i=2$.

(a) $r+p$ **must satisfy** $o \Rightarrow p$ **must satisfy** o since $p \xrightarrow{\tau}$
$\Rightarrow q$ **must satisfy** o.

It follows by a case analysis on whether $r \xrightarrow{\tau}$ or not that $r+q$ **must satisfy** o.

(b) Trivial. □

Lemma 4.3.2. *If d is in nf and p in hnf, then $d \sqsubseteq_i p$ implies $d_\lambda \sqsubseteq_i p_\lambda$ whenever both d_λ and p_λ exist.*

Proof. (a) $i=3$.

d_λ **may satisfy** o
$\Rightarrow d$ **may satisfy** λo,
$\Rightarrow p$ **may satisfy** λo
$\Rightarrow p_\lambda$ **may satisfy** o.

(b) $i=2$.

d_λ **must satisfy** o

$\Rightarrow$ d **must satisfy** λo

$\Rightarrow$ p **must satisfy** λo

$\Rightarrow$ p_λ **must satisfy** o.

(c) $i=1$. Follows from (a) and (b). □

Lemma 4.3.3. (a) *$d\Uparrow$ implies $d =_1 d+\Omega$.*

(b) *$p \xrightarrow{\mu} p'$ implies $p =_1 p+\mu p'$.*

(c) *$p \overset{s}{\Rightarrow} p'$ implies $p+\Omega =_1 p+s+\Omega$.*

Proof. (a) We may assume d is in nf. If $d\Uparrow$, then d must have a λnf and the result is immediate.

(b) By induction on the proof that $p \xrightarrow{\mu} p'$. The proof may be found in [23], using the axioms (D7), (D8), (D9).

(c) By induction on the length of the derivation $p \overset{s}{\Rightarrow} p'$.

(i) $s=\varepsilon$. Immediate

(ii) $s=\lambda s'$. There are two cases:

Case 1. $p \xrightarrow{\lambda} p'$ for some p' such that $p' \overset{s'}{\Rightarrow}$. Then

$$\begin{aligned} p+\Omega &=_1 p+\lambda p'+\Omega \quad \text{using (b)} \\ &=_1 p+\lambda(p'+\Omega)+\Omega \quad \text{using (D3)} \\ &=_1 p+\lambda(p'+\Omega+s')+\Omega \quad \text{by induction} \\ &=_1 p+\lambda p'+\lambda s'+\Omega \quad \text{using (D10)} \\ &=_1 p+\lambda s'+\Omega. \end{aligned}$$

Case 2. $p \xrightarrow{\tau} p'$ for some p' such that $p' \overset{s}{\Rightarrow}$. By induction, $p'+\Omega =_1 p'+\Omega+s$. Therefore

$$\begin{aligned} p+\Omega &=_1 p+\tau p'+\Omega \quad \text{using (b)} \\ &=_1 p+\tau(p'+\Omega)+\Omega \quad \text{using (D3)} \\ &=_1 p+\tau(p'+\Omega+s)+\Omega \quad \text{by induction} \\ &=_1 p+\tau(p'+\Omega+s)+s+\Omega \quad \text{using (D8)} \\ &=_1 p+s+\Omega. \end{aligned}$$ □

Definition 4.3.4. (a) If p, q are in hnf, let $p \prec_2 q$ if $\langle p, q\rangle$ satisfies any one of the conditions below:

(i) p is $\sum_{\lambda\in L}\lambda p_\lambda$, q is $\sum_{\lambda\in L}\lambda q_\lambda$.

(ii) p is $\sum_{L\in\mathscr{H}}\tau\sum_{\lambda\in L}\lambda p_\lambda$, q is $\sum_{K\in\mathscr{H}}\tau\sum_{\lambda\in K}\lambda q_\lambda$, where $\mathscr{H}\subseteq\mathscr{S}$.

(iii) p is as in (ii), q is $\sum_{\lambda\in K}\lambda q_\lambda$, where $K\in\mathscr{S}$.

(b) If p, q are in hnf, let $p \prec_1 q$ if, in addition to satisfying part (a), $S(p)\subseteq S(q)$.

Note that if $p <_1 q$ and $\langle p, q\rangle$ satisfy condition (a)(ii), then $A(\mathscr{K}) = A(\mathscr{L})$. If they satisfy condition (a)(iii), we have that $K = A(\mathscr{L})$.

Lemma 4.3.5. *If p, q are in hnf and $p \sqsubseteq_i^+ q$, then $p <_i q$ for $i = 1$ or 2.*

Proof. (a) $i = 2$. From Lemma 4.1.1, $S(p) \supseteq S(q)$.

(i) Suppose both p, q are in λhnf and $p \xrightarrow{\lambda}$. Then p **must satisfy** $\lambda\omega$. So q **must satisfy** $\lambda\omega$, i.e., $q \xrightarrow{\lambda}$. Therefore $S(p) = S(q)$ and $\langle p, q\rangle$ satisfies condition (i).

(ii) Suppose both p, q are in τhnf as in condition (ii). We show $\mathscr{K} \subseteq \mathscr{L}$. Because $\mathscr{L}$ is saturated, it is sufficient to show that $K \in \mathscr{K}$ implies $L \subseteq K$ for some $L \in \mathscr{L}$.

Suppose on the contrary that there exists a $K_0 \in \mathscr{K}$ such that $L \subseteq K_0$ for no $L \in \mathscr{L}$. Let o be the observer $\sum\{\bar{\lambda}\omega \mid \lambda \in S(p), \lambda \notin K_0\}$. Then p **must satisfy** o whereas $q \mid o \to^* r \in$ ***Fail*** which contradicts $p \sqsubseteq_2 q$. Since $\mathscr{L}$ is saturated, it follows that $\mathscr{K} \subseteq \mathscr{L}$.

(iii) Suppose p, q are as in condition (iii) and suppose $K \neq A(\mathscr{L}')$ for any $\mathscr{L}' \subseteq \mathscr{L}$. Then $(\forall L \in \mathscr{L})(\exists \lambda \in L)(\lambda \notin K)$.

Let o be the observer $\sum\{\bar{\lambda}\omega \mid \lambda \in S(p), \lambda \notin K\}$. Then p **must satisfy** o whereas $q \mid o \in$ ***Fail***. This is a contradiction.

Now from Lemma 4.1.1(b) if $p \sqsubseteq_2^+ q$ and p is in λnf, then q is also in λnf. It therefore follows that cases (i), (ii) and (iii) are exhaustive and $p <_2 q$.

(b) $i = 1$. Since $p \sqsubseteq_1 q$ implies $p \sqsubseteq_2 q$, $\langle p, q\rangle$ satisfies the conditions from part (a). Since $p \sqsubseteq_1 q$ implies $p \sqsubseteq_3 q$, it follows from Lemma 4.1.1 that $S(p) \subseteq S(q)$. □

Lemma 4.3.6. *If p, q are in hnf, $p <_i q$, and $p_\lambda \sqsubseteq_i q_\lambda$ for every $\lambda \in S(q)$, then $p \sqsubseteq_i q$, $i = 1, 2$.*

Proof. (a) $i = 1$. Then since $S(p) = S(q)$, $p \sqsubseteq_1 r$, where r denotes $p[q_\lambda / p_\lambda, \lambda \in S(p)]$. Now since $p <_1 q$, $\langle p, q\rangle$ must satisfy either (i), (ii) or (iii) of Definition 4.3.4(a).

(i) Then r is q.

(ii) Then

$$r \sqsubseteq_1 q + \sum_{L \in \mathscr{L}/\mathscr{K}} \tau \sum_{\lambda \in L} \lambda q_\lambda$$

$$\sqsubseteq_1 q + \sum_{\lambda \in K} \lambda q_\lambda, \quad K = A(\mathscr{L}/\mathscr{K}), \quad \text{using (N4)}.$$

Now each λ in K must appear in a summand of q and therefore we may apply (D8) to prove

$$q + \sum_{\lambda \in K} \lambda q_\lambda =_1 q.$$

(iii) Let $\mathscr{K} = \mathscr{L}'/\mathscr{L}$, where $\mathscr{L}' = \{K\}$. Then

$$r =_1 \sum_{L \in \mathscr{L}'} \tau \sum_{\lambda \in L} \lambda q_\lambda + \sum_{L \in \mathscr{K}} \tau \sum_{\lambda \in L} \lambda q_\lambda$$

$$\sqsubseteq_1 \sum_{L \in \mathscr{L}'} \tau \sum_{\lambda \in L} \lambda q_\lambda + \sum_{\lambda \in K'} \lambda q_\lambda, \quad K' = A(\mathscr{K}) \quad \text{using (N4)}$$

$$=_1 \sum_{L \subset \mathscr{L}'} \tau \sum_{\lambda \in L} \lambda q_\lambda \quad \text{using (D8) since } K' \subseteq A(\mathscr{L}')$$

$$\sqsubseteq_1 \sum_{\lambda \in A(\mathscr{L}')} \lambda q_\lambda = q \quad \text{using (N4).}$$

(b) $i=2$. Similar to part (a). When $\langle p, q\rangle$ satisfy case (ii) of Definition 4.3.4(b) we apply (E3). If they satisfy case (iii), we apply (E3) and then (N4). □

We are now ready to prove the main part of the completeness theorem.

Proposition 4.3.7. *$d \sqsubseteq_i^{\dagger} p$ implies $A_i \vdash d \sqsubseteq p$, $i = 1, 2, 3$.*

Proof. The proof is divided into three parts, according to the value of i. We use induction on the size of d.

(a) $i=3$. From Corollary 4.2.13 we may assume d is in wnf. We will then show that $d \sqsubseteq_1 p$. In fact, d can be rewritten as $\Omega + \sum s_j$ by repeated use of (D10) and so it will be sufficient to prove $s_j + \Omega + p =_1 p + \Omega$ for every j. However, it is relatively easy to see that $p \overset{s_j}{\Rightarrow}$ and therefore we need only to apply Lemma 4.3.3(c).

(b) $i=2$. From Proposition 4.2.11 we may assume d is in snf. If $d\Uparrow$, then from Lemma 4.3.3(a) and (E2), $d =_2 \Omega$ and the result is immediate. So we may assume $d\Downarrow$. This in turn implies $p\Downarrow$. (Consider the observer $\tau\omega$.) So we may assume from Proposition 4.2.15 that p is in hnf. Then from Lemmas 4.3.2, 4.3.1 we may assume that $d_\lambda \sqsubseteq_2^{\dagger} p_\lambda$ for every $\lambda \in S(p)$. By induction, $d_\lambda \sqsubseteq_2 p_\lambda$. Applying the previous two lemmas we get $d \sqsubseteq_2 p$.

(c) $i=1$. If $d\Uparrow$, then $d =_1 d + \Omega$ and so by Proposition 4.2.12, d has a weak normal form. The result now follows from part (a). If $d\Downarrow$, the proof is similar to part (b). □

Theorem 4.3.8 (Theorem 3.2.5). *For $i = 1, 2, 3$ and closed terms p, q, $p \sqsubseteq_i^{\S} q$ implies $A_i \vdash p \sqsubseteq q$.*

Proof. We know that $p \sqsubseteq_i^{\dagger} q$. In order to apply (R4) to derive $p \sqsubseteq q$ it is sufficient to show that $A_i \vdash d \sqsubseteq q$ for every d in FIN(p). Now $d \prec p$ and therefore, by Lemma 4.1.6, $d \sqsubseteq_i^{\dagger} p$. So $d \sqsubseteq_i^{\dagger} q$. By the previous proposition,

$$A_i \vdash d \sqsubseteq q. \qquad \square$$

4.4

In this section we redo the completeness theorems for finite terms. This will give more insight into the nature of normal forms and we will use the additional information when proving the representation theorems of Section 5.

Let NF_1, NF_2, NF_3 denote the set of normal forms, strong normal forms and weak normal forms respectively. These sets may be ordered in a way similar to Definition 4.3.4.

Definition 4.4.1. For $d, d' \in \mathrm{NF}_\omega$, let $d \prec_i d'$ be defined by

(a) $i = 3$:

$$d \prec_3 d' \quad \text{if} \quad S(d) \subseteq S(d'),$$

(b) $i = 2$:

$$d \prec_2 d' \quad \text{if} \quad d \text{ is } \Omega \text{ or if } \langle d, d' \rangle \text{ satisfies Definition 4.3.4(a).}$$

(c) $i = 1$:

$$d \prec_1 d' \quad \text{if} \quad d \prec_2 d' \text{ and } d \prec_3 d'.$$

We have similar results for $\prec_i$ on normal forms of finite terms as on head normal forms of finite and infinite terms.

Proposition 4.4.2. *For $d, d' \in \mathrm{NF}_t$:*

(a) *$d \sqsubseteq_i^+ d'$ implies $d \prec_i d'$,*

(b) *if $d_\lambda \sqsubseteq_i d'_\lambda$ whenever both d_λ and d'_λ are defined and $d \prec_i d'$, then $d \sqsubseteq_i d'$.*

Proof. Similar to Lemmas 4.3.5 and 4.3.6. □

Definition 4.4.3. For $d, d' \in \mathrm{NF}_\omega$, let $d \prec_i d'$ if

(i) $d \prec_i d'$,

(ii) $d_\lambda \prec_i d'_\lambda$ whenever both d_λ and d'_λ are defined.

Theorem 4.4.4. *For $d, d' \in \mathrm{NF}_\omega$, $d \sqsubseteq_i^+ d'$ implies $d \prec_i d'$.*

Proof. By induction on the size of both d and d'. From Lemmas 4.3.2 and 4.3.1 we may assume that $d_\lambda \sqsubseteq_i^+ d'_\lambda$ whenever both d_λ and d'_λ exist. By induction, $d_\lambda \prec_i d'_\lambda$. The result now follows from Proposition 4.4.2(a). □

Note that we can also use Proposition 4.4.2(b) to show that $d \prec_i d'$ implies $d \sqsubseteq_i d'$. We also have the converse.

Corollary 4.4.5. *For $d, d' \in \mathrm{NF}_\omega$, $d \sqsubseteq_i d'$ implies $d \prec_i d'$.*

Proof. By the soundness theorem (Corollary 4.1.14), $d \sqsubseteq_i d'$ implies $d \sqsubseteq_i^+ d'$. Now apply the previous theorem. □

It is this corollary which justifies the use of the same normal form for the terms in NF_ω. This result will be used in the next section.

5. Denotational semantics

We have presented pure CCS as the set of recursive terms over a set of operators. This enables us to give a denotational semantics in a very straightforward way, using the techniques of [28, 11, 10, 9]. To save space we assume familiarity with notions such as Σ-partial order or Σ-po, Σ-cpo, finite element, algebraic cpo, ideal completion, etc. Details may be found in the above references.

5.1

Let D be a Σ-cpo. The set of D-environments, ENV_D, ranged over by e is the set of mappings from variables in X to D. As usual we let $e(d/x)$ be the environment identical with e except at x where its value is d. The terms in REC_Σ can be interpreted in D by defining

$$\mathscr{V}_D : \mathrm{REC}_\Sigma \to \mathrm{ENV}_D \to D$$

as follows:

(i) $\mathscr{V}_D[\![\mathrm{op}(t_1, \ldots, t_k)]\!]e = \mathrm{op}_D(\mathscr{V}_D[\![t_1]\!]e \ldots, \mathscr{V}_D[\![t_k]\!]e)$

(ii) $\mathscr{V}_D[\![\mathrm{rec}\ x.t]\!]e = Y\lambda d.\mathscr{V}_D[\![t]\!]e(d/x)$

where Y denotes the least fixpoint operator.

We now turn our attention to pure CCS, a particular example of the above.

Definition 5.1.1. (a) A Σ-cpo D is *sound with respect to* $\sqsubseteq_i^c$ if $\mathscr{V}_D[\![t]\!] < \mathscr{V}_D[\![u]\!]$ implies $t \sqsubseteq_i^c u$.

(b) A Σ-cpo D is *complete with respect to* $\sqsubseteq_i^c$ if $t \sqsubseteq_i^c u$ implies $\mathscr{V}_D[\![t]\!] < \mathscr{V}_D[\![u]\!]$.

(c) A Σ-cpo D is *fully abstract with respect to* $\sqsubseteq_i^c$ if it satisfies both (a) and (b).

Let I_i be the initial Σ-cpo in the category of Σ-cpos which satisfies the set of axioms A_i, $i = 1, 2, 3$. For the construction of I_i see [1, 4, 11]. I_i is the ideal completion of the Σ-po generated by the axioms A_i.

Theorem 5.1.2. *For $i = 1, 2, 3$, I_i is fully abstract with respect to $\sqsubseteq_i^c$.*

Proof. For convenience in this proof we write $t <_i u$ if $\mathscr{V}_i[\![t]\!] < \mathscr{V}_i[\![u]\!]$.

(a) *Soundness.* We first prove it for closed terms. So suppose $p <_i q$. Let $d < p$. Then $d <_i p$ and therefore $d <_i q$. Since I_i is algebraic there exists a finite term e such that $d <_i e$, $e < q$. By the construction of I_i it follows that $d \sqsubseteq_i e$. From the soundness theorem for the proof system $d \sqsubseteq_i^c e$ and so $d \sqsubseteq_i^c q$. Since this is true for every $d \in \mathrm{FIN}(p)$ it follows that $p \sqsubseteq_i^c q$.

Now suppose $t <_i u$. Then we must show $t\rho \sqsubseteq_i^c u\rho$ for every closed substitution ρ. However, $t <_i u$ implies $t\rho <_i u\rho$ and therefore we may apply the first part.

(b) *Completeness.* If $A_i \vdash t \sqsubset u$, then $t <_i u$. This follows because by construction $<_i$ satisfies the axioms, since I_i is algebraic it is preserved by (R4) and it is trivial to see that it is preserved by the other rules. Therefore, if $p \sqsubseteq_i^c q$ it follows that $p <_i q$ since we can apply the completeness theorem for the proof system.

More generally, suppose $t \sqsubseteq_i^c u$. We must show $te <_i ue$ for every environment e. An environment e is *finite* if each $e(x)$ is finite. Since $\mathcal{V}_D$ is continuous in its second argument, it is sufficient to show $te <_i ue$ for every finite e. However, by construction every finite element a of I_i is denoted by a finite term $t(a)$ in FREC_Σ, i.e., $\mathcal{V}_{I_i}[\![t(a)]\!] = a$. Then the result follows since $\mathcal{V}_{I_i}[\![t]\!]e = \mathcal{V}_{I_i}[\![t\rho_e]\!]$, where ρ_e is the closed substitution defined by $\rho_e(x) = t(e(x))$. □

This theorem is simply a restatement of the soundness and completeness theorems for the proof systems. It is, however, of considerable interest since the models I_i have simple representations as trees.

5.2

In this section we consider the representations of I_i as particular kinds of trees.

If L is a set of labels, let PCT_L denote the set of (finite or infinitely branching) trees whose branches are labelled by labels from L in such a way that for every $\lambda \in L$ every node has at most one outgoing branch labelled by λ.

Every node in such a tree, tr, can be uniquely identified by a string from L. We denote the node in tr identified by s as $\mathrm{tr}(s)$. We let $N(\mathrm{tr})$ denote the set of strings which identify every node in tr. $N(\mathrm{tr})$ is prefixed closed, i.e., $s \in N(\mathrm{tr})$ and $s = s_1 s_2$ implies $s_1 \in N(\mathrm{tr})$. There is in fact an isomorphism between PCT_L and the set of prefixed closed strings from L. We prefer, however, the more graphical notation of trees. We also let $S(\mathrm{tr}(s))$ denote the set of labels on the branches from the node $\mathrm{tr}(s)$, and $\mathrm{tr}(s)_\lambda$ denote the successor tree of $\mathrm{tr}(s)$ along the unique branch λ from $\mathrm{tr}(s)$, if it exists. Our model will consist of trees from PCT_L whose nodes are labelled in a special way.

Definition 5.2.1. Let RT denote the set of trees in $\mathrm{PCT}_{A \cup \bar{A}}$ such that

(a) every node is either open (represented by ○) or closed (represented by ●).

(b) every closed node $\mathrm{tr}(s)$ is labelled by a saturated set of subsets of $S(\mathrm{tr}(s))$, denoted by $\mathscr{A}(\mathrm{tr}(s))$, and the following conditions hold:

(i) if a node has an infinite number of successors, it is open,

(ii) if a node is open, every successor is open,

(iii) if $\mathscr{A}(\mathrm{tr}(s))$ is empty, then $\mathrm{tr}(s)$ is the root,

(iv) if $\mathscr{A}(\mathrm{tr}(s))$ is not empty, then $\lambda \in S(\mathrm{tr}(s))$ implies $\exists A \in \mathscr{A}(\mathrm{tr}(s))$ such that $\lambda \in A$.

The sets $\mathscr{A}(\mathrm{tr}(s))$ are called acceptance sets. By definition they are finite collections of finite subsets of actions. Note that we distinguish between the empty acceptance set $\emptyset$ and the acceptance set containing the empty subset, $\{\emptyset\}$. Thus the two trees

$$\bullet\emptyset \qquad \bullet\{\emptyset\}$$

are different. In fact one will represent NIL, the other τNIL.

Definition 5.2.2. Let SRT denote the set of trees tr in RT such that every open node in tr is a leaf.

Definition 5.2.3. Let WRT denote the set of trees in RT all of whose nodes are open.

Note that WRT is in fact isomorphic to $PCT_{\Delta\cup\bar{\Delta}}$ since both are isomorphic to the set of prefix closed sets of strings over $\Delta\cup\bar{\Delta}$. Examples of trees are given in Figs. 1 and 2. For convenience we omit $\mathscr{A}(\mathrm{tr}(s))$ if it is $\emptyset$.

It will be convenient in the remainder of this section to rename RT, SRT, WRT by the less suggestive RT_1, RT_2, RT_3 respectively. All of the examples are finite trees, i.e., have a finite number of nodes. These will play an important role and we let FRT_i denote the set of finite trees from RT_i.

Definition 5.2.4. (a) For $\mathrm{tr}, \mathrm{tr}' \in RT_2$ let $\mathrm{tr} <_3 \mathrm{tr}'$ if $N(\mathrm{tr}) \subseteq N(\mathrm{tr}')$.

(b) For $\mathrm{tr}, \mathrm{tr}' \in RT_1$ let $\mathrm{tr} <_2 \mathrm{tr}'$ if, for every $s \in N(\mathrm{tr}')$, $\mathrm{tr}(s)$ closed implies:

(i) $\mathrm{tr}'(s)$ is closed,

(ii) $\mathscr{A}(\mathrm{tr}(s)) \supseteq \mathscr{A}(\mathrm{tr}'(s))$,

(iii) $\mathscr{A}(\mathrm{tr}'(s)) = \emptyset$ implies $S(\mathrm{tr}'(s)) \in \mathscr{A}(\mathrm{tr}(s))$ or $\mathscr{A}(\mathrm{tr}(s)) = \emptyset$ and $S(\mathrm{tr}(s)) = S(\mathrm{tr}'(s))$.

(c) For $\mathrm{tr}, \mathrm{tr}' \in RT_1$ let $\mathrm{tr} <_1 \mathrm{tr}'$ if $\mathrm{tr} <_2 \mathrm{tr}'$ and $\mathrm{tr} <_3 \mathrm{tr}'$.

Referring to Figs. 1 and 2 we have $p_1 <_1 q_1$, $p_2 <_2 q_2$ and $p_3 <_1 q_3$. Note that if $\mathscr{A}(\mathrm{tr}(s)) \neq \emptyset$, then $\mathscr{A}(\mathrm{tr}(s)) \supseteq \mathscr{A}(\mathrm{tr}'(s))$ implies $S(\mathrm{tr}(s)) \supseteq S(\mathrm{tr}'(s))$. This follows from condition (iv) in the definition of RT. Therefore, if $\mathrm{tr} <_2 \mathrm{tr}'$, we have that $S(\mathrm{tr}'(s)) \subseteq S(\mathrm{tr}(s))$ whenever $\mathrm{tr}(s)$ is closed and $\mathrm{tr}'(s)$ exists. If $\mathrm{tr} <_1 \mathrm{tr}'$, then $\mathrm{tr} <_3 \mathrm{tr}'$ and so $S(\mathrm{tr}(s)) \subseteq S(\mathrm{tr}'(s))$ for every $s \in N(\mathrm{tr})$ and $S(\mathrm{tr}'(s))$ whenever $\mathrm{tr}(s)$ is closed.

Proposition 5.2.5. *$\langle RT_i, <_i\rangle$ is an algebraic cpo with finite elements FRT_i, $i = 1, 2, 3$.*

Proof. (a) $i = 2$. The least element is the trivial tree with an open leaf. It should be obvious that $<_2$ is a partial order. Let $\{t_i \mid i \in I\}$ be a directed set in RT_2. Define t as follows:

(i) $N(t) = \{s \mid s \in N(t_i)$ for almost all $i \in I\}$.

Note that $N(t)$ is prefix closed.

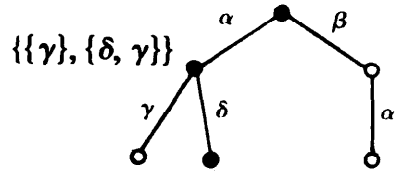
(ii) For $s \in N(t)$ let $t(s)$ be open if $t_i(s)$ is open for almost all $i \in I$.

(iii) Otherwise $t(s)$ is closed. In this case $\exists k \in I$ s.t. $t_k(s)$ is closed. Since $\{t_i \mid i \in I\}$ is directed, this means that $t_i(s)$ is closed for almost all $i \in I$.

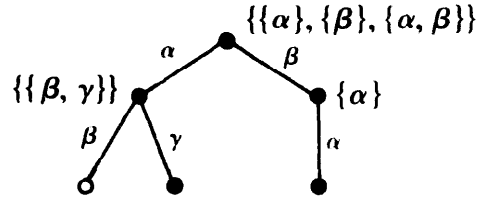
(iv) If $t(s)$ is closed let

$$\mathscr{A}(t(s)) = \bigcap\{\mathscr{A}(t_1(s)) \mid t_i(s) \text{ is closed}\}.$$

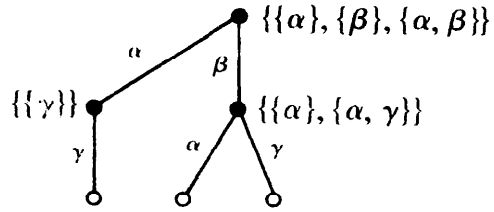
It is easy to check that $t \in RT_2$. Moreover, $\forall s \in N(t)$ there exist k_s such that $S(t(s)) = S(t_{k_s}(s))$ and $\mathscr{A}(t(s)) = \mathscr{A}(t_{k_s}(s))$. This is sufficient to show that t is the lub of $\{t_i \mid i \in I\}$. We leave it to the reader to check that FRT_2 generates RT_2.



(i) representation of $p_1 \equiv \alpha(\tau\gamma\Omega + \tau(\gamma\Omega + \delta)) + \beta(\alpha\Omega + \Omega)$.

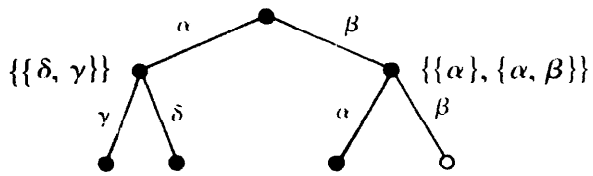


(ii) representation of $p_2 \equiv \tau\alpha\tau(\beta\Omega + \gamma) + \tau\beta\tau\alpha + \tau(\alpha\tau(\beta\Omega + \gamma) + \beta\tau\alpha)$.

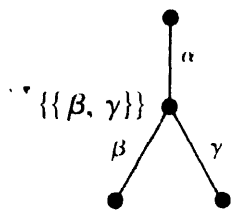


(iii) representation of $p_3 \equiv \tau\alpha r_1 + \tau\beta r_2 + \tau(\alpha r_1 + \beta r_2)$ where $r_1 \equiv \tau\gamma\Omega$ and $r_2 \equiv \tau\alpha\Omega + \tau(\alpha\Omega + \gamma\Omega)$.

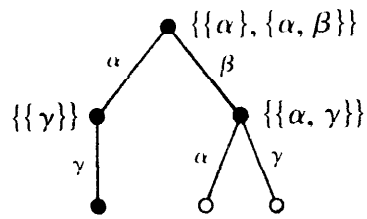
Fig. 1.



(i) representation of $q_1 \equiv \alpha(\tau(\gamma + \delta) + \beta(\tau\alpha + \tau(\alpha + \beta\Omega)))$.



(ii) representation of $q_2 \equiv \alpha\tau(\beta + \gamma)$.



(iii) representation of $q_3 \equiv \tau\alpha\tau\gamma + \tau(\alpha\tau\gamma + \beta(\tau(\alpha\Omega + \gamma\Omega)))$.

Fig. 2.

(b) $i = 1$. It is trivial to check that $<_3$ is a partial order and therefore $<_1$ is also a partial order, with the least element defined in (a). The least upper bound of a directed sequence is defined as in (a). Simple calculations will show that $\mathbf{FRT}_1$ are finite elements and generate $\mathbf{RT}_1$.

The case $i = 3$ is similar. □

The representation theorem depends on a strong correspondence between normal forms and finite trees in RT, which we now describe.

Let $\phi : \mathbf{NF} \to \mathbf{FRT}$ be defined by structural induction as follows:

(a) If n is $\sum_{\lambda \in L} \lambda n_\lambda + \Omega$, then $\phi(n)$ is the tree whose root is open and is joined to the subtree $\phi(n_\lambda)$ by a branch labelled λ, for every $\lambda \in L$.

(b) If n is $\sum_{\lambda \in L} \lambda n$, then $\phi(n)$ is as in (a) except that the root is closed and is labelled by $\emptyset$, the empty set at subsets.

(c) If n is $\sum_{L \in \mathscr{S}} \tau \sum_{\lambda \in L} \lambda n_\lambda$, then $\phi(n)$ is as in (b) except that the root is labelled by $\mathscr{S}$.

Lemma 5.2.6. (a) $\forall n \in \mathbf{NF}_i, \phi(n) \in \mathbf{RT}_i$, $i = 1, 2, 3$.

(b) $n <_i n'$ *implies* $\phi(n) <_i \phi(n')$, $i = 1, 2, 3$.

Proof. (a) is proven by structural induction on n.

(b) If $n <_i n'$, then, from Definition 4.4.3, $n_\lambda <_i n_\lambda$ whenever both are defined. By induction we may assume that $\phi(n_\lambda) <_i \phi(n'_\lambda)$. The proof is now completed by a case analysis on why $n <_i n'$. □

In the opposite direction we can define a mapping $\psi : \mathbf{FRT} \to \mathbf{NF}$ by induction on the depth of the tree:

(a) If $\mathrm{tr}(\varepsilon)$ is open, then

$$\psi(\mathrm{tr}) = \sum \{\lambda \psi(\mathrm{tr}(\varepsilon)_\lambda) \mid \lambda \in S(\mathrm{tr}(\varepsilon))\} + \Omega.$$

(b) If $\mathrm{tr}(\varepsilon)$ is closed and $\mathscr{A}(\mathrm{tr}(\varepsilon)) = \emptyset$, then

$$\psi(\mathrm{tr}) = \sum \{\lambda \psi(\mathrm{tr}(\varepsilon)_\lambda) \mid \lambda \in S(\mathrm{tr}(\varepsilon))\}.$$

(c) If $\mathrm{tr}(\varepsilon)$ is closed and $\mathscr{A}(\mathrm{tr}(\varepsilon)) \neq \emptyset$, then

$$\psi(\mathrm{tr}) = \sum \{\tau \sum \{\lambda \psi(\mathrm{tr}(\varepsilon)_\lambda) \mid \lambda \in L\} \mid L \in \mathscr{A}(\mathrm{tr}(\varepsilon))\}.$$

Lemma 5.2.7. (a) $\mathrm{tr} \in \mathbf{RT}_i$ *implies* $\psi(\mathrm{tr}) \in \mathbf{NF}_i$, $i = 1, 2, 3$.

(b) $\mathrm{tr} <_i \mathrm{tr}'$ *implies* $\psi(\mathrm{tr}) <_i \psi(\mathrm{tr}')$.

(c) ϕ *and* ψ *are inverses.*

Proof. (a) and (c) are easily proven by structural induction.

(b) If $\mathrm{tr} <_i \mathrm{tr}'$, then a case analysis will show that $\psi(\mathrm{tr}) <_i \psi(\mathrm{tr}')$. By induction, $\psi(\mathrm{tr}(\varepsilon)_\lambda) <_i \psi(\mathrm{tr}'(\varepsilon)_\lambda)$ whenever both $\mathrm{tr}(\varepsilon)_\lambda$, $\mathrm{tr}'(\varepsilon)_\lambda$ are defined. It then follows that $\psi(\mathrm{tr}) <_i \psi(\mathrm{tr}')$. □

We now turn our attention to the construction of the models I_i. We assume the reader is familiar with the details of such constructions, which may be found in [10, 11, 4]. I_i may be described as $(\mathrm{FREC}_\Sigma/\sqsubseteq_i)^\infty$, where $\mathrm{FREC}_\Sigma/\sqsubseteq_i$ denotes the Σ-partial order over finite terms generated by $\sqsubseteq_i$, and $(\ )^\infty$ denotes the ideal completion. We now use the results of Section 4.4 to show that $\mathrm{FREC}_\Sigma/\sqsubseteq_i$ has a simple representation using normal forms.

For $\mathrm{op} \in \Sigma_k$ define $\mathrm{op}_i : \mathrm{NF}_i^k \to \mathrm{NF}_i$, $i = 1, 2, 3$ by

(a) $\mathrm{op}_1(n_1, \ldots, n_k) = \mathrm{nf}(\mathrm{op}(n_1, \ldots, n_k))$,

(b) $\mathrm{op}_2(n_1, \ldots, n_k) = \mathrm{snf}(\mathrm{op}(n_1, \ldots, n_k))$,

(c) $\mathrm{op}_3(n_1, \ldots, n_k) = \mathrm{wnf}(\mathrm{op}(n_1, \ldots, n_k))$.

It follows from Corollary 4.4.5 that op_i is monotonic, i.e., it preserves $<_i$.

Proposition 5.2.8. *With these operators,* $\langle \mathrm{NF}_i, <_i \rangle$ *is isomorphic to* $\mathrm{FREC}_\Sigma/\sqsubseteq_i$ *as* Σ*-pos.*

Proof. Let $\mathrm{id} : \mathrm{NF}_i \to \mathrm{FREC}$ be the identity and $\varepsilon_i : \mathrm{FREC}_\Sigma \to \mathrm{NF}_i$ be defined by $\varepsilon_1 = \mathrm{nf}$, $\varepsilon_2 = \mathrm{snf}$ and $\varepsilon_3 = \mathrm{wnf}$. Then obviously id preserves the order and the operators. From Corollary 4.4.5, if $d \sqsubseteq_i d'$, then $\varepsilon_i(d) <_i \varepsilon_i(d)$. The same result shows that ε_i preserves the operators. Therefore $\langle \mathrm{id}, \varepsilon_i \rangle$ is an isomorphism pair. □

In a similar fashion we can consider FRT_i as Σ-pos. For $\mathrm{op} \in \Sigma_k$ define $\mathrm{op}_i' : \mathrm{FRT}_i^k \to \mathrm{FRT}_i$ by

$$\mathrm{op}_i'(\mathrm{tr}_1, \ldots, \mathrm{tr}_k) = \phi(\mathrm{op}_i(\psi(\mathrm{tr}_1), \ldots, \psi(\mathrm{tr}_k))).$$

Proposition 5.2.9. *With these operators* $\langle \mathrm{FRT}_i, <_i \rangle$ *is a* Σ*-po and is isomorphic to* $\langle \mathrm{NF}_i, <_i \rangle$.

Proof. The proof follows from Lemmas 5.2.6 and 5.2.7. □

We are now able to state the final result of the paper.

Theorem 5.2.10. *For* $i = 1, 2, 3$, RT_i *is a* Σ*-cpo which is isomorphic to* I_i *and therefore fully-abstract w.r.t.* $\sqsubseteq_i^c$.

Proof. We have already shown that RT_i is an algebraic cpo with FRT_i as finite elements. To define the continuous functions $\mathrm{op}_i : \mathrm{RT}_i^k \to \mathrm{RT}_i$ it is sufficient to define it for the finite elements. This we have already done and so we may consider RT_i as a Σ-cpo. Moreover, an algebraic Σ-cpo is uniquely determined (up to isomorphism) by the Σ-po induced on its finite elements. From Propositions 5.2.9 and 5.2.8 we may conclude that I_i is isomorphic to RT_i. The result now follows from Theorem 5.1.2. □

The weakness of this representation theorem is that we have not given a natural definition of the operators of CCS which apply directly to the trees in RT_i. We have

defined them indirectly by defining them on normal forms and using the isomorphism between normal forms and the finite trees in RT_Ω. However, these definitions may be found in [7].

6. Alternative characterisations

In this section we relate the equivalences for processes generated by the three preorders introduced in this paper with equivalences presented in other work, notably observational equivalence [23], weak equivalence [21] and failures equivalence [18]. In particular we concentrate on Kennaway's weak equivalence which allows a deeper insight of our $\simeq_2$. For all comparisons we consider only strongly convergent agents, i.e., finite or infinite agents which do not contain Ω as a subprocess. Formally, a closed term p is *strongly convergent* if for every $s \in A^* p \stackrel{s}{\Rightarrow} p'$ implies $p'\Downarrow$. This is mainly because either the definitions of the other equivalences do not involve any notion of divergence [23, 21] or an approach very different from ours is taken [18]. These comparisons suggest alternative characterisations of our preorders which are discussed in Section 6.4.

6.1

The original observation equivalence for CCS [23, 15] is much smaller than $\simeq_1$. We can show that $\simeq_3$ coincides with $\approx_1$ and $\simeq_1$ lies between $\approx_1$ and $\approx_2$. (The equivalences $\approx_n$ are defined in [23, p. 99].) So the observation equivalence of CCS distinguishes many more terms than we do. The main reason for $\approx$ being finer (more discriminating) than $\simeq_1$ seems to be the recursive nature of its definition. In some sense in order to decide if two agents are observationally equivalent one needs to check that they can perform the same sequences of actions and that the subagents reached after each sequence still have equivalence behaviour. Some of the resulting distinctions are concerned only with the internal structure of processes and an interesting critique of weak equivalence is given in [5]. There the author gives another equivalence. However, it only applies to finite terms and there is no obvious extension to recursive terms. Even for finite terms his language is less expressive than ours. Nevertheless, the exact relationship between $\simeq_1$ and his equivalence is not known.

6.2

In [21], Kennaway introduces a new notion of equivalence ($=$ weak equivalence) for his calculus (NSCP). His equivalence though based on Milner's one takes into account considerations which are similar to those used in the definition of our $\simeq_2$. An interesting result is that, though $=$ has the same recursive structure as $\approx$ and recursiveness seems to give a deeper insight into the structure of agents, $\simeq_2$ and $=$ turn out to coincide for strongly convergent agents. To simplify the comparison we work entirely in CCS. We adapt Kennaway's definition to CCS agents and then

prove that this definition can be reduced to an equivalent non-recursive one. Finally, we prove that on strongly convergent CCS agent $=$ and $=_2$ coincide.

We start with some definitions based on those of Section 2.

Let A be the set of visible actions, $A = \Delta \cup \bar{\Delta}$. L will range over finite subsets of A. We let SCCREC_{Σ} denote the set of all strongly convergent closed CCS agents and let P, Q, R range over subsets of SCCREC_{Σ}.

Let

$$\mathrm{Init}(p) = \left\{ a \in A \mid p \overset{a}{\Rightarrow} \right\}, \qquad \mathrm{Traces}(p) = \left\{ s \in A^* \mid p \overset{s}{\Rightarrow} \right\}$$

and extend the relation $\Downarrow$ to $\Downarrow s$ for every s in A^* in a natural way:

(i) $p \Downarrow \varepsilon$ if $p \Downarrow$,

(ii) $p \Downarrow \alpha s$ if $p \Downarrow$ and $p \overset{\alpha}{\Rightarrow} p'$ implies $p' \Downarrow s$.

As might be expected, $\Uparrow s$ denotes the negation of $\Downarrow s$.

$$p \textbf{ after } \varepsilon = p$$

$$p \textbf{ after } \alpha = \left\{ p' \mid p \overset{\alpha}{\Rightarrow} p' \right\}$$

$$p \textbf{ after } \alpha s = (p \textbf{ after } \alpha) \textbf{ after } s$$

and

$$P \textbf{ after } \varepsilon = P$$

$$P \textbf{ after } \alpha = \bigcup \{ p \textbf{ after } \alpha \mid p \in P \}$$

$$P \textbf{ after } \alpha s = \bigcup \{ p \textbf{ after } \alpha \mid p \in P \} \textbf{ after } s.$$

These preliminary definitions allow us to state the following important concepts. p **must** L if and only if for all p' such that $p \overset{\tau}{\Rightarrow} p'$, $\exists \alpha \in L$ such that $p' \overset{\alpha}{\Rightarrow}$. p **must** L if and only if p **must** L for all $p \in P$.

We are now ready to adapt Kennaway's equivalence to CCS.

Definition 6.2.1. $P =_0 Q$ is always true.

$P =_{n+1} Q$ if and only if $\forall$ finite $L \subseteq A$, P **must** $L \Leftrightarrow Q$ **must** L, and

$$\forall \alpha \in A,\ P \textbf{ after } \alpha =_n Q \textbf{ after } \alpha.$$

$P = Q$ if and only if, $\forall n \geq 0$, $P =_n Q$.

We first give an alternative characterisation of $=$ which does not involve any recurrence.

Theorem 6.2.2. *$P = Q$ if and only if $\forall s \in A^*$, $\forall$ finite $L \subseteq A$,*

$$(P \textbf{ after } s) \textbf{ must } L \Leftrightarrow (Q \textbf{ after } s) \textbf{ must } L.$$

Proof. ($\Leftarrow$) We prove that $P \neq Q$ implies $\exists s \in A^*$, $\exists L \subseteq A$ such that (P **after** s) **must** L and (Q **after** s) **must** L.

If $P \neq Q$, then there exists an $n > 0$ such that $P \neq_n Q$. We prove the claim by induction on n.

Induction basis. $P \simeq_1 Q$ implies there exists some L such that P **must** L and Q **mμst** L. It follows trivially that (P **after** ε) **must** L and (Q **after** ε) **mμst** L.

Inductive step. We have $P \not\simeq_{n+1} Q$ if and only if

(i) $P \not\simeq_1 Q$, or

(ii) $\exists \alpha \in A$ such that P **after** $\alpha \not\simeq_n Q$ **after** α.

In case (i) the claim follows from the induction basis. In case (ii) we have by the inductive hypothesis that for some $\alpha \in A$, $s \in A^*$,

(P **after** α) **after** s **must** L and (Q **after** α) **after** s **mμst** L,

i.e., (P **after** αs) **must** L and (Q **after** αs) **mμst** L.

($\Rightarrow$) Suppose there exists some $s \in A^*$ and some finite $L \subseteq A$ such that (P **after** s) **must** L and (Q **after** s) **mμst** L. We prove by induction on s that $P \not\simeq Q$.

Induction basis: $s = \varepsilon$. Since P **after** $\varepsilon = P$, it follows that $P \not\simeq_1 Q$, i.e., $P \not\simeq Q$.

Inductive step: $s = \alpha s'$. Then (P **after** α) **after** s' **must** L whereas (Q **after** α) **after** s' **mμst** L. By induction, P **after** $\alpha \not\simeq Q$ **after** α and so $P \not\simeq Q$. □

This result allows us to derive an easy corollary.

Corollary 6.2.3. *If p and q are strongly convergent, then $p \simeq q$ implies* traces(p) = traces(q).

Proof. Suppose $\exists s$ such that $s \in$ traces(p) and $s \notin$ traces(Q). Let α be such that $p \xrightarrow{\alpha}$ (α exists since P is strongly convergent). Then (p **after** s) **mμst** $\{\alpha\}$ whereas vacuously (q **after** s) **must** $\{\alpha\}$, i.e., $p \not\simeq q$. □

Given the alternative characterisation of Kennaway's equivalence for CCS we can relate it to the equivalence generated by the preorder $\sqsubseteq_2$.

We first prove the following lemma.

Lemma 6.2.4. *If p and q are strongly convergent and $p \simeq_2 q$, then* traces(p) = traces(q).

Proof. Suppose there exist s such that $s \in$ traces(p) and $s \notin$ traces(q). If s denotes $\alpha_1 \alpha_2 \ldots \alpha_n$, let $o_1 = \tau\omega + \bar{\alpha}_1(\tau\omega + \cdots + \tau\omega + \bar{\alpha}_n)\ldots)$. Then since $q \Downarrow s$ for all $s \in A^*$, q **must satisfy** o_1 while p **mμst satisfy** o_1, which contradicts the fact that $p \simeq_2 q$. □

We can now prove the main theorem of this section which states the equivalence of $\simeq$ and $\simeq_2$. In fact, because of the previous lemma it is immediate that $\simeq_2$ and $\simeq_1$ coincide for strongly convergent terms. Consequently we will also have that, in this case, $\simeq$ coincides with $\simeq_1$.

Theorem 6.2.5. *If p and q are strongly convergent, then $p \simeq_2 q$ if and only if $p \simeq q$.*

Proof. ($\Rightarrow$) We prove $p \not\simeq q$ implies $p \not\simeq_2 q$. From Theorem 6.2.2 we have that $p \not\simeq q$

implies $\exists s, L$ such that (p **after** s) **must** L and (q **after** s) **must** L or vice versa. Without loss of generality we can assume that there exist $s \in A^*$ and a finite $L \subseteq A$ such that $p \stackrel{s}{\Rightarrow} p'$ implies there exists an $\alpha \in L$ such that $p' \stackrel{\alpha}{\Rightarrow}$ while either $q \stackrel{s}{\not\Rightarrow}$ or there exist q', $q \stackrel{s}{\Rightarrow} q'$, and $q' \stackrel{\alpha}{\not\Rightarrow}$ for no $\alpha \in L$. In case $q \stackrel{s}{\not\Rightarrow}$, the previous lemma implies $p \not\approx_2 q$. In case $q \stackrel{s}{\Rightarrow} q'$ and $q' \stackrel{\alpha}{\not\Rightarrow}$ for all $\alpha \in L$, if $s = \alpha_1\alpha_2 \ldots \alpha_n$ let o_2 denote $\tau\omega + \bar{\alpha}_1(\tau\omega + \bar{\alpha}_2(\ldots \bar{\alpha}_{n-1}(\tau\omega + \bar{\alpha}_n(\sum_{\alpha \in L} \bar{\alpha}\omega)\ldots)$. Then, since $p \Downarrow s$ for all $s \in A^*$, p **must satisfy** o_2 while q **must satisfy** o_2 and so $p \not\approx_2 q$.

($\Leftarrow$) We prove $p \not\approx_2 q$ implies $p \not\approx q$. We have that $p \not\approx q$ implies there exists an observer o such that p **must satisfy** o and q **must satisfy** o or vice versa. Suppose q **must satisfy** o. This would imply:

(a) there exists a finite derivation

$$q \mid o = q_0 \mid o_0 \rightarrow q_1 \mid o_1 \rightarrow \cdots \rightarrow q_n \mid o_n$$

such that $q_n \mid o_n \not\rightarrow$ and $o_i \stackrel{\omega}{\not\rightarrow}$ for every i, $0 \leq i \leq n$, or

(b) there exists an infinite derivation

$$q \mid o = q_{0_m} \mid o_0 \rightarrow q_1 \mid o_1 \rightarrow \cdots \rightarrow q_k \mid o_k \rightarrow.$$

Let s denote the sequence of actions performed by q in these derivations. In case (a) we have by Corollary 6.2.3 that $s \in \text{traces}(p)$. Since p **must satisfy** o, (p **after** s) **must** Init(o_n). It follows that $p \not\approx q$ since (q **after** s) **must** Init(o_n). In case (b), p **must satisfy** o implies that $s(k) \notin \text{traces}(p)$ for some finite prefix, $s(k)$, of s. Once more by Corollary 6.2.3 it follows that $p \not\approx q$. □

As an immediate consequence of Theorems 6.2.2 and 6.2.5 we have an alternative characterization of $\approx$.

Corollary 6.2.6. *For strongly convergent terms p, q, $p \approx_2 q$ if and only if*

$$\forall s, \forall L (p \textbf{ after } s) \textbf{ must } L \textit{ if and only if } (q \textbf{ after } s) \textbf{ must } L$$

This corollary and the observers used in the proofs of Theorem 6.2.5 and Lemma 6.2.4 suggest that we only need specific observers in order to distinguish between processes.

Let

$$\mathcal{O}_2 = \text{NIL} \mid \sum_{\omega \in I} \alpha\omega \mid \tau\omega + \beta\mathcal{O}_2$$

and

$$p \sqsubseteq_2 q \quad \text{if} \quad \forall o \in \mathcal{O}_2\ p \textbf{ must satisfy } o \text{ implies } q \textbf{ must satisfy } o.$$

We have the following theorem.

Theorem 6.2.7. *For strongly convergent terms p, q,*

$$p \approx_2 q \textit{ if and only if } p \simeq_2 q.$$

Proof. The result in one direction is trivial since $\mathcal{O}_2 \subseteq \mathcal{O}$.

To prove the converse we need to show that $p \simeq q$ implies there exists an $o \in \mathcal{O}_2$ such that p **must satisfy** o and q **must satisfy** o or vice versa. But the claim follows from the previous corollary and from part ($\Rightarrow$) of the proof of Theorem 6.2.5, since the observers used in the proof of Lemma 6.2.4 and Theorem 6.2.5 belong to $\mathcal{O}_2$ ($o_1, o_2 \in \mathcal{O}_2$). □

6.3

A mathematical model of processes, called the refusal set model was introduced in [18] and developed further in [21] as refusal-acceptance-machines. Basically, the refusal set model defines a process as a set of pairs $\langle s, X \rangle$ where s is a string of actions and X is a nonempty set of sets of actions, the refusal sets. Such a pair means that the process may perform the sequence of actions s and may then refuse any set of actions in X. This is quite similar to our SRT with our acceptance sets being the complement of the refusal sets. There are seemingly minor but important differences. For example, it is crucial for us to have the empty set as an acceptance set in order to handle τ and it is not clear what the corresponding refusal set is. An ordering is defined on this model which is somewhat similar to $<_2$ defined on SRT (at least the version of the ordering in [21]). There is, however, a crucial difference between $<_2$ and their order. There are two reasons why $p <_2 q$. The first is that intuitively p is more nondeterministic than q, the second that some undefined component of p (specified by Ω or an open leaf) has been improved upon in q. The second component seems to be absent from their ordering. Nevertheless, this ordering turns out to be a complete partial order and various operators are shown to be continuous. (A notable omission from the relevant theorem [21, Theorem 4.1] is their version of 'hiding', direct image.) These operators form the basis of a language for processes and the model is then a denotational model in the sense of [28]. A direct comparison with our work is somewhat hampered by the very different set of operators. Moreover, the handling of divergence with refusal sets seems to raise various problems. For a detailed discussion of this point, see [6]. However, we can modify the refusal set model to distinguish between τp and p. With some other adjustments it should be possible to get a strong relationship between SRT and a version of the refusal set model. As regards equivalences (two processes are 'failures equivalent' if they are denoted by the same refusal set) it is easy to prove that failures equivalence and $\simeq_2$ coincide for strongly convergent terms. We can prove in fact that $\langle s, X \rangle \in \text{failures}(p)$ if and only if p **after** s **must** X, and the result then follows from the new characterisation of $\simeq_2$ in Corollary 6.2.6.

6.4

In Section 6.2 we have given an alternative characterisation of $\simeq_2$ for strongly convergent terms which is independent from the notion of observers. This kind of characterisation can be given for all the preorders $\sqsubseteq_1$ over arbitrary terms. In this

section we give this alternative characterisation for $\sqsubseteq_i$ and use it to define directly $\sqsubseteq_i^c$, i.e., preorders which are preserved by all CCS operators and which are also independent of the notion of observers.

Definition 6.4.1

$p \sqsubseteq_3' q$ if $\text{traces}(p) \subseteq \text{traces}(q)$,

$p \sqsubseteq_2' q$ if $\forall s \in A^*, \forall$ finite $L \subseteq A$, $p \Downarrow s$ implies
(i) $q \Downarrow s$, and
(ii) (p **after** s) **must** L implies (q **after** s) **must** L,

$p \sqsubseteq_1' q$ if $p \sqsubseteq_3' q$ and $p \sqsubseteq_2' q$.

Before proving the main characterisation theorem we need two more lemmas.

Lemma 6.4.2. *If $p \sqsubseteq_2 q$, then, for all $s \in A^*$, $p \Downarrow s$ implies*
(i) $q \Downarrow s$,
(ii) $s \in \text{traces}(q)$ *implies* $s \in \text{traces}(p)$.

Proof. (i) Suppose there exists $s = \alpha_1 \ldots \alpha_n$ such that $p \Downarrow s$ and $q \Uparrow s$. Then if we choose $o_2^1 = \tau\omega + \bar{\alpha}_1(\tau\omega + \cdots + \bar{\alpha}_{n-1}(\tau\omega + \bar{\alpha}_n\tau\omega)\ldots)$ we have p **must satisfy** o_2^1 and q **must satisfy** o_2, i.e., $p \not\sqsubseteq_2 q$.

(ii) Suppose there exist $s = \alpha_1 \ldots \alpha_n$ such that $p \Downarrow s$, $s \in \text{traces}(q)$ and $s \notin \text{traces}(p)$. Then if we choose $o_2^s = \tau\omega + \bar{\alpha}_1(\tau\omega + \cdots + \bar{\alpha}_{n-1}(\tau\omega + \bar{\alpha}_n)\ldots)$ we have p **must satisfy** o_2^s and q **must satisfy** o_2^s, i.e., $p \not\sqsubseteq_2 q$. □

Lemma 6.4.3. *If* (q **after** s) **must** L *for some* $L \subseteq A$, *then* $s \in \text{traces}(q)$.

Proof. Suppose $s \notin \text{traces}(q)$, then q **after** $s = \emptyset$ and we have by definition $\emptyset$ **must** L for every finite $L \subseteq A$. □

Lemma 6.4.4. *If $p \Downarrow s$ and $p \sqsubseteq_2' q$, then $s \in \text{traces}(q)$ implies $s \in \text{traces}(p)$.*

Proof. Suppose there exists some s such that $s \in \text{traces}(q)$, $p \Downarrow s$ and $s \notin \text{traces}(p)$. By the previous lemma, (p **after** s) **must** L for every finite $L \subseteq A$. Since $q \Downarrow s$ we have that $\bigcup\{\text{Init}(q'), q' \in q \textbf{ after } s\}$ is finite. Consequently, we can find an α such that $q \xrightarrow{\alpha\alpha}$. Then ($q$ **after** s) **must** $\{\alpha\}$ while (p **after** s) **must** $\{\alpha\}$, which contradicts the fact that $p \sqsubseteq_2' q$. □

We are now ready to prove the main characterisation theorem.

Theorem 6.4.5

$p \sqsubseteq_i q$ if and only if $p \sqsubseteq_i' q$ for $i = 1, 2, 3$.

Proof. Because the way $\sqsubseteq_1$ and $\sqsubseteq_1'$ have been defined we need only to prove the theorem for $i = 2, 3$.

$i = 3$: For $s \in A^*$ let $o_3 = s\omega$. Then $s \in \text{traces}(p)$ if and only if p **may satisfy** o_3. The claim is an easy corollary of this fact.

$i = 2$: (a) We prove first $p \not\sqsubseteq_2' q$ implies $p \not\sqsubseteq_2 q$.

If $p \not\sqsubseteq_2' q$, then $\exists s, L$ such that $p \Downarrow s$ and ($q \Uparrow s$ or for some finite $L \subseteq A$ (p **after** s) **must** L and (q **after** s) **must** L). If $p \Downarrow s$ and $q \Uparrow s$ by Lemma 6.4.2 we have $p \not\sqsubseteq_2 q$. If $p \Downarrow s$ and $q \Downarrow s$ but for some finite L, (p **after** s) **must** L and (q **after** s) **must** L, then by Lemma 4.6.3 we have that $s \in \text{traces}(q)$. Therefore, if we define, as in part ($\Rightarrow$) of the proof of Theorem 6.2.5, $o_2^L = \tau\omega + \bar{\alpha}_1(\tau\omega + \cdots + \bar{\alpha}_{n-1}(\tau\omega + \bar{\alpha}_n \sum_{\alpha \in L} \bar{\alpha}\omega) \ldots)$ we have p **must satisfy** o_2^L and q **must satisfy** o_2^L.

(b) We sketch the proof of $p \not\sqsubseteq_2 q$ implies $p \not\sqsubseteq_2' q$.

It follows the same line as part ($\Leftarrow$) of the proof of Theorem 6.2.5. We need also to take into account the possibility q **must satisfy** o because $q \mid o = q_0 \mid o_0 \longrightarrow \cdots \longrightarrow q_n \mid o_n$ and $q_n \mid o_n \uparrow$ and $o_i \overset{\omega}{\not\Rightarrow}$ for all $o \leqslant n$, i.e., there exists some s such that $q \overset{s}{\Rightarrow} q_n$, $o \overset{s}{\Rightarrow} o_n$ and $q \Uparrow s$ or $o \Uparrow \bar{s}$. However, since p **must satisfy** o, we have $p \Downarrow s$ and either $p \overset{s}{\not\Rightarrow}$ or $o \Downarrow \bar{s}$. In the former case, Lemma 6.4.4 implies $p \not\sqsubseteq_2' q$; in the latter case we have $q \Uparrow s$ which also implies $p \not\sqsubseteq_2' q$ since $p \Downarrow s$. □

The proofs of Lemma 6.4.2 and Theorem 6.4.5 suggest we need only specific observers to distinguish between processes. Let $\mathcal{O}_2$, $\mathcal{O}_3$ be the set of observers generated by the following grammars:

$$t ::= \text{NIL} \mid \tau\omega \mid \sum_{\alpha \in I} \alpha\omega \mid \tau\omega + \beta t$$

and

$$t ::= \omega \mid \alpha t.$$

Finally, let $\mathcal{O}_1 = \mathcal{O}_2 \cup \mathcal{O}_3$.

Theorem 6.4.6

$p \sqsubseteq_i q$ if and only if $p \sqsubseteq_i^{\mathcal{O}_i} q$, $i = 1, 2, 3$.

Proof. We need to prove the claim only for $i = 2, 3$. The 'only if' part is immediate since $\mathcal{O}_i \subset \mathcal{O}$, where $\mathcal{O}$ is the set of observers used to generate $\sqsubseteq_i$. We only need to prove the 'if' part.

$i = 3$: The claim directly follows from the corresponding case of Theorem 6.4.5.

$i = 2$: Notice o_2^L, $o_2^{\emptyset}$ and o_2^I all belong to $\mathcal{O}_2$. We have $p \not\sqsubseteq_2 q$ implies $p \not\sqsubseteq_2' q$ which implies, as shown in the proofs of Theorem 6.4.5 and Lemma 6.4.2, that p **must satisfy** o and q **must satisfy** o for some $o \in \mathcal{O}_2$. □

We conclude by deriving a characterisation of $\sqsubseteq_1'$ in the same vein as that for $\sqsubseteq_1$ given in Theorem 6.4.5. We first need the following lemmas.

Lemma 6.4.7. *If* $p \sqsubseteq_2 q$, *then*

(i) $p \xrightarrow{\tau}$ *implies* $p \sqsubseteq_2^c q$

(ii) $p \xrightarrow{\tau}$ *implies* $\tau p \sqsubseteq_2^c q$

(iii) $p \not\xrightarrow{\tau}$, $q \not\xrightarrow{\tau}$ *implies* $p \sqsubseteq_2^c q$.

Proof. (i) Suppose r **must satisfy** o. Then, since $p \xrightarrow{\tau}$, p **must satisfy** o. Therefore q **must satisfy** o which in turn implies that $r+q$ **must satisfy** o. It follows that $p \sqsubseteq_2^c q$.

(ii) Suppose $r+\tau p$ **must satisfy** o. Then once more p **must satisfy** o which leads, as in case (i) to the result that $r+q$ **must satisfy** o.

(iii) Suppose $r+p$ **must satisfy** o. If $p \mid o \not\xrightarrow{\tau}$, then from the characterisation of $\sqsubseteq_2$ in Theorem 6.4.5 it follows that $q \mid o \not\xrightarrow{\tau}$. It follows that in this case $r+q$ **must satisfy** o. On the other hand, if $p \mid o \xrightarrow{\tau}$, then p **must satisfy** o. Since we are assuming that $p \sqsubseteq_2 q$ it follows that q **must satisfy** o and therefore $r+q$ **must satisfy** o. □

Lemma 6.4.8. *If* $p \sqsubseteq_2 q$ *and* $(p\Downarrow, q \xrightarrow{\tau})$ *implies* $p \xrightarrow{\tau}$, *then* $p \sqsubseteq_2^c q$.

Proof. If $p \xrightarrow{\tau}$, then the result follows from the previous lemma. If $p\Uparrow$, then the result is trivially true. Therefore, we may assume that $p\Downarrow$, $q \not\xrightarrow{\tau}$ and $p \not\xrightarrow{\tau}$. From part (iii) of the previous lemma it follows once more that $p \sqsubseteq_2^c q$. □

Theorem 6.4.9. (i) $p \sqsubseteq_3^c q$ *if and only if* $\mathrm{traces}(p) \subseteq \mathrm{traces}(q)$.

(ii) $p \sqsubseteq_2^c q$ *if and only if*

(a) $p\Downarrow$ *and* $q \xrightarrow{\tau}$ *implies* $p \xrightarrow{\tau}$,

(b) *for all* $s \in A^*$ *and finite* $L \subseteq A$, $p \Downarrow s$ *implies*

(i) $q \Downarrow s$,

(ii) $(p$ **after** $s)$ **must** L *implies* $(q$ **after** $s)$ **must** L.

Proof. (i) Follows from the fact that $\sqsubseteq_3$ preserves all of the operators of CCS.

(ii) In one direction it follows from Lemma 4.1.1 and Theorem 6.4.5. In the opposite direction it follows from the same theorem and the previous lemma. □

7. Conclusion

We first of all recapitulate on the results of the paper. We started with a rather general notion of equivalence between processes based on a simple tabulation of the possible effects of interactions between observers and processes. This equivalence was in turn decomposed into three different preorders in a natural way. The remainder of the paper is an investigation of these preorders in the language CCS. For each of these we gave a complete proof system based on a set of axioms and a rule of induction. These proof systems lead in a natural way to fully abstract denotational models. These are constructed as term models but in Section 5 we showed that they have very intuitive representations as particular kinds of trees.

Moreover, in the final section, we gave alternative characterisations of the various preorders and of the largest precongruences contained in them which are independent from the notion of observers.

Much remains to be done. For example, the representations of the fully-abstract models need to be more fully investigated. These new equivalences should also be investigated for the more general version of CCS which allows value-passing. More generally, we should be able to produce models for CSP [16], Distributed Processes [3] and such languages which are more intuitive than the model in [8], for example. The axioms systems presented in this paper may also lead to more convenient calculi for proving equivalence of processes and more generally the correctness of processes. It would be interesting to examine the various example proofs in [23] to see if our axioms lead to simpler proofs.

Acknowledgment

The authors would like to thank R. Milner and G. Plotkin for many useful discussions. The first author would like to acknowledge the financial assistance of the Italian Research Council (C.N.R.) and the second author wishes to express his gratitude to the University of Genoa for their hospitality. Credit for the excellent typing is due to E. Kerse.

Appendix A. Proof of Proposition 4.1.9(b)

Suppose p **must satisfy** o. Consider the computation tree from $o|p$ where the leaves are labelled by terms $o'|p'$ such that $o' \xrightarrow{\omega}$. Every term has a finite number of successors. Therefore this tree is finite. We use induction on the size of the tree. Furthermore, because of Lemma 4.1.8 it is sufficient to show that there exists a d such that $d \sqsubseteq_1 p$ and d **must satisfy** o. If $p\uparrow$, then $o \xrightarrow{\omega}$ and the required d is Ω. So we may assume $p\Downarrow$ and therefore in hnf. Furthermore, we may assume $o \xrightarrow{\omega}$. There are two cases according to the form of p:

Case (i) $p = \sum_{\lambda \in L} \lambda p_\lambda$. Let L' denote the set of $\lambda \in L$ such that $o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} o_n \xrightarrow{\lambda}$, where $o_k \xrightarrow{\omega}$ for $1 \leq k \leq n$. If $L' = \emptyset$, then the required d is $\sum_{\lambda \in L} \lambda\Omega$. So we may assume $L' \neq \emptyset$. For each $\lambda \in L'$ let $D(\lambda)$ denote the set of o' such that $o \xrightarrow{\tau} o_1 \xrightarrow{\tau} \cdots \xrightarrow{\tau} o_n \xrightarrow{\lambda} o'$ where $o_k \xrightarrow{\omega}$ for $1 \leq k \leq n$. Then, for every $o' \in D(\lambda)$, p_λ **must satisfy** o'. By induction on the size of the computation tree there exists a finite term $d(\lambda, o')$ such that $d(\lambda, o')$ **must satisfy** o' and $d(\lambda, o') \sqsubseteq_1 p_\lambda$. The required d is $\sum \{\lambda d(\lambda, o') \mid \lambda \in L', o' \in D(\lambda)\} \cup \sum_{\lambda \notin L} \lambda\Omega$.

Case (ii) $p = \sum_{I} \tau_{\lambda \in I} \lambda p_\lambda$. Let p_I denote $\sum_{\lambda \in I} \lambda p_\lambda$. Then for every $I \in L'$, p_I **must satisfy** o. By induction on the size of computation tree there exists a d_I such that d_I **must satisfy** o, and $d_I \sqsubseteq_1 p_I$. The required d is $\sum_{I} \tau d_I$. □

References

[1] S. Bloom, Varieties of ordered algebras, *J. Comput. Systems Sci.* **13** (1976) 200-212.

[2] P. Brinch-Hansen, The programming language concurrent Pascal, *IEEE TSI* **1** (1975) 99-205.

[3] P. Brinch-Hansen, Distributed processes: A concurrent programming concept, *Comm. ACM* **21** (11) (1978) 934-941.

[4] B. Courcelle and M. Nivat, Algebraic families of interpretations, *Proc. 17th FOCS Ann. Symp.*, Houston, 1976.

[5] Ph. Darondeau, An enlarged definition and complete axiomatization of observational congruence of finite processes, in: *Lecture Notes in Computer Science* **137** (Springer, Berlin, 1982) pp. 47-62.

[6] R. De Nicola, A complete set of axioms for a theory of communicating sequential processes, *Proc. FCT '83*, Lecture Notes in Computer Science **158** (Springer, Berlin, 1983) pp. 115-126.

[7] R. De Nicola, Ph.D. Thesis, Dept. of Computer Science, Univ. of Edinburgh, forthcoming.

[8] N. Francez, C.A.R. Hoare, D.J. Lehmann and W.-P. de Roever, Semantics of nondeterminism, concurrency and communication, *J. Comput. System Sci.* **19** (1979) 290-308.

[9] M. Gordon, *The Denotational Description of Programming Languages* (Springer, Berlin, 1979).

[10] J.A. Goguen, J.W. Thatcher and J.B. Wright, Initial algebra semantics and continuous algebras, *J. ACM* **24** (1) (1977) 68-95.

[11] I. Guessarian, Algebraic semantics, in: *Lecture Notes in Computer Science* **99** (Springer, Berlin, 1981).

[12] M. Hennessy, Powerdomains and nondeterministic recursive definitions, in: *Lecture Notes in Computer Science* **137** (Springer, Berlin, 1982) pp. 178-193.

[13] M. Hennessy and G. Plotkin, A term model for CCS, in: *Lecture Notes in Computer Science* **88** (Springer, Berlin, 1980) pp. 261-274.

[14] M. Hennessy, A term model for synchronous process, *Information and Control* **51** (1) (1981) 58-75.

[15] M. Hennessy and R. Milner, On observing nondeterminism and concurrency, in: *Lecture Notes in Computer Science* **85** (Springer, Berlin, 1980) pp. 299-309.

[16] C.A.R. Hoare, Communicating sequential processes, *Comm. ACM* **21** (8) (1978).

[17] C.A.R. Hoare, A model for communicating sequential processes, Tech. Monograph Prg-22, Computing Laboratory, University of Oxford, 1981.

[18] C.A.R. Hoare, S.D. Brookes and A.D. Roscoe, A theory of communicating sequential processes, Tech. Monograph Prg-16, Computing Laboratory, University of Oxford, 1981.

[19] J.D. Ichbiah et al., Reference Manual for the ADA Programming language, United States Department of Defence, July 1980.

[20] J.K. Kennaway and C.A.R. Hoare, A theory of nondeterminism, in: *Lecture Notes in Computer Science* **85** (Springer, Berlin, 1980) pp. 338-350.

[21] J.K. Kennaway, Formal semantics of nondeterminism and parallelism, Ph.D. Thesis, University of Oxford, 1981.

[22] B.W. Lampson and D.D. Redell, Experience with processes and monitors in Mesa, *Comm. ACM* **23** (2) (1980) 105-107.

[23] R. Milner, A calculus of communicating systems, in: *Lecture Notes in Computer Science* **92** (Springer, Berlin, 1980).

[24] G. Plotkin, A powerdomain construction, *SIAM J. Comput.* **5** (1976) 452-486.

[25] W.C. Rounds and S.D. Brookes, Possible futures, acceptances, refusals and communicating processes, *Proc. 22nd FOCS Ann. Symp.*, Nashville, TN, 1981.

[26] D.S. Scott, Data types as lattices, *SIAM J. Comput.* **5** (3) (1976).

[27] M.B. Smyth, Power domains, *J. Comput. System Sci.* **2** (1978) 23-26.

[28] J. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory* (MIT Press, 1977).

[29] N. Wirth, Programming in Module-2 (Springer, Berlin, 1983).