On Testing the Observable Actions of Processes

William Ferreira

Abstract. We present and investigate two testing preorders for a value-passing version of CCS, [Mil89] which distinguish processes by their observable actions. We develop an operational theory for the preorders, and compare and contrast them to *must* testing [NH84, Hen88, Ing94, HI93]. In doing so we prove an expressivity result that relates one of them to *must* testing under a mild assumption. Finally we show that both preorders are fully abstract with respect to variations of the value-passing acceptance tree model, AT^v, introduced in [HI93].

Keywords:

Concurrency, operational semantics, testing, non-determinism, process algebra.

1 Introduction

The process calculus CCS [Mil80, Mil89] is a well established abstraction for specifying concurrent, communicating systems, using a small set of well-defined operators. The semantics of CCS terms can be desribed using a transition system, captured from the operational rules [Plo81b] which define the actions a CCS term can perform. In addition, the theory of bisimulation [Par81, Mil89] which is defined in terms of the transition system of a term, is a well-established theory of equivalence for CCS processes. Essentially, two processes are deemed bisimilar if each can match the other's initial actions, in such a way that the states resulting from the performance of the actions are also bisimilar. Bisimulation and its myriad of variations has been used to develop rich theories of equivalence for a number of process calculi.

The theory of testing introduced in [NH84, Hen88] defines a behavioural theory of equivalence for a variation of the process calculus CCS, called τ-less CCS. In this theory CCS terms are distinguished by their ability to react to independent *observers* (also called *tests*), whose behaviour is also defined using a transition system. Very basically, an observer interacts with a process, resulting in a *computation*. A computation is deemed *successful* if during the computation, the process can provoke the observer into a passing through a pre-defined *success state*. If $Comp(O, P)$ denotes the set of all computations that may result from the interaction of an observer O and process P, two natural equivalences between processes can be derived, based on the quantification of success over the set of computations $Comp(O, P)$. The first, called *may testing*, states that two processes P and Q are equivalent if for all observers O, whenever there exists a successful computation of O and P then there exists a successful computation of O and Q, and vice-versa. This equivalence is sometimes called *trace equivalence*, and takes no account of the possible non-determinism a process may exhibit. By demanding success for all computations, one arrives at *must testing*, which is sensitive to non-determinism, and finer than *may testing*. The *may* and *must* equivalences are defined as the kernels of preorders $\sqsubseteq_{\mathcal{MY}}$ and $\sqsubseteq_{\mathcal{MT}}$, which are given by:

- $P \sqsubseteq_{\mathcal{MY}} Q$ if for all observers O, if there exists a successful computation in $Comp(O, P)$ then there exists a successful computation in $Comp(O, Q)$,
- $P \sqsubseteq_{\mathcal{MT}} Q$ if for all observers O, whenever all computations in $Comp(O, P)$ are successful then so are all computations in $Comp(O, Q)$.

As an example of the testing power of $\sqsubseteq_{\mathcal{MY}}$ and $\sqsubseteq_{\mathcal{MT}}$, consider the two τ-less CCS processes P and Q defined by:

$$P = a.\mathbf{0} + b.\mathbf{0} \text{ and } Q = a.\mathbf{0} \oplus b.\mathbf{0}$$

where $\oplus$ is the *internal choice* operator: it can resolve to either of its operands without interaction with the environment. We have $P \sqsubseteq_{\mathcal{MY}} Q$, but $P \not\sqsubseteq_{\mathcal{MT}} Q$ because for the observer $O = \overline{a}.\omega$ we have that $Comp(O, P)$ is successful in all computations; there is only one:

$$\overline{a}.\omega \parallel P \xrightarrow{\tau} \omega \parallel \mathbf{0}$$

where $||$ is the parallel operator, τ is the action resulting from an internal computation or communication, and ω is an observer success state. However we have that:

$$\overline{a}.\omega \mid\mid Q \xrightarrow{\tau} \overline{a}.\omega \mid\mid b.\mathbf{0}$$

which is stuck, because $\overline{a}.\omega$ is not a success state: it cannot perform the success action ω; P and Q are related by $\sqsubseteq_{\mathcal{MY}}$ because the computation:

$$\overline{a}.\omega \mid\mid Q \xrightarrow{\tau} a.\omega \mid\mid \overline{a}.\mathbf{0} \xrightarrow{\tau} \omega \mid\mid \mathbf{0}$$

ensures there is at least one successful computation in $Comp(O, Q)$.

In [Mor68] a behavioural theory is developed for the λ-calculus. In this case a basic property of λ-expressions is chosen - the ability of an expression to converge to *head normal form* - and this is used to construct a precongruence on expressions by closing up under all contexts. For example, if $P \Downarrow$ denotes that P converges to head normal form, then defining $P \sqsubseteq Q$ if for all program contexts C, $C[P] \Downarrow$ implies $C[Q] \Downarrow$. If we compare this form of testing to *must* testing we can see that the context C plays the rôle of the observer, and that success is convergence. By definition, $\sqsubseteq$ is a congruence i.e. whenever two λ terms are related by $\sqsubseteq$ we can place them in any context C and they will still be related. The limited context in which processes in *must* testing are placed i.e. in parallel with an observer process, means that one needs to prove a separate result showing that $\sqsubseteq_{\mathcal{MT}}$ (and indeed $\sqsubseteq_{\mathcal{MY}}$) is a congruence.

The testing power of an observer, and the context in which it is placed with a process, will both affect the derived testing preorder: the more limited the context the less discriminating the preorder. One can argue that a potential observer should be any context of the language. In [San92] Sangiorgi investigates a behavioural equivalence for CCS called *barbed congruence* which is defined by closing up an obervability predicate, called a *barb*, under all contexts of the language. He then shows how barbed congruence is equivalent to bisimulation [Mil89, Par81], which is defined independently of contexts. We are interested not in bisimulation, but testing, and in this report we contruct a behavioural theory for value-passing CCS based on the form of contextual testing developed in [Mor68] for the λ-calculus, by defining a basic notion of observability for processes, which can then be closed up under all contexts to obtain a pre-order.

A direct adapation of the definition of $\sqsubseteq$ to CCS is not very interesting. The nearest equivalent in CCS to a λ-calculus term in head normal form is a process with no further internal computation; but such a process may still be able to offer communication actions to its environment. For example if $P \Downarrow$ denotes that the CCS process P has no further internal computation, then we might define $P \sqsubseteq Q$ if for all contexts C, $C[P] \Downarrow$ implies $C[Q] \Downarrow$. Returning to the two processes P and Q defined above, we see that this preorder is quite coarse: there is no context C such that $C[P] \Downarrow$ and $C[Q] \not\Downarrow$; what is needed is an observability predicate which is sensitive to the states a process may reach, and they actions it may perform there.

In this report we define two behavioural preorders, called *guarantee* and *strong guarantee* testing, in terms of more primitive predicates on processes. The first predicate, called *can guarantee*, says that P *can guarantee* a if in all states that P can reach through internal computation, an a action can be performed. The *strongly guarantee* preorder is defined in terms of a stricter predicate called *can strongly guarantee*, which demands not only that a process can guarantee an action, but that it is convergent on that action, i.e. whenever the action is performed, the resulting state of the process is convergent. More formally, letting $P \stackrel{\varepsilon}{\Longrightarrow} P'$ denote that P can evolve to P' through a sequence of internal transitions, and $P \stackrel{a}{\Longrightarrow}$ denote that P can perform an a action, possibly interspersed with zero or more internal transitions, we have:

- P *can guarantee* a if P converges, and $P \stackrel{\varepsilon}{\Longrightarrow} P'$ implies $P' \stackrel{a}{\Longrightarrow}$, and
- P *can strongly guarantee* a if P converges, and $P \stackrel{\varepsilon}{\Longrightarrow} P'$ implies $P' \stackrel{a}{\Longrightarrow}$ and P' converges on a.

By closing up these predicates under all contexts of the language, we derive the *guarantee* and *strong guarantee* testing preorders, which we denote by $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ respectively.

The remainder of this report is devoted to investigating $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ when defined for *VPL*, the value-passing variant of τ-less CCS introduced in [HI93]. In section 3 we present formally the *guarantee* and *strongly guarantee* testing preorders for *VPL*, and derive equivalent alternative characterisations for them, defined independently of contexts. In section 4 we review *must* testing for *VPL*, and then compare *must* testing to *guarantee* and *strongly guarantee* testing. We prove an expressivity result relating *must* and *guarantee* testing under an assumption about the operational semantics of the conditional expression $\mathsf{if} \cdot \mathsf{then} \cdot \mathsf{else} \cdot$. In section 5 we construct two denotational models for the language, based on variations of *value-passing acceptance trees* [HI93], and in section 6 we prove that these models are fully abstract for their respective preorders.

2 Operational Semantics

In this section we present the syntax and operational semantics of *VPL*, the value-passing version of τ-less CCS introduced in [HI93]. Let:

- $v, v_1, v_2, \ldots \in \mathit{Val}$ be a set of values,
- $x, x_1, x_2, \ldots \in \mathit{Var}$ a set of expression variables,
- $op \in Op$ a set of functions or operator symbols,
- $X, Y, Z \in \mathit{VRec}$ a set of process variables, and
- $n, n_1, n_2, \ldots \in \mathit{Chan}$ a predefined set of channel names.

The abstract syntax of our language is given by the following grammar:

$$\begin{aligned}
e, e_1, \ldots \in \mathit{Exp} &:= \mathbf{0} \mid \alpha.e \mid \mathsf{if}\, l\, \mathsf{then}\, e\, \mathsf{else}\, e \mid e \,\Box\, e \mid e \backslash n \mid e[R] \mid \mu X.e \mid X \mid \Omega \\
\Box \in \mathit{BinOp} &:= \oplus \mid + \mid \| \\
\alpha, \alpha_1, \alpha_2, \ldots \in \mathit{Pre} &:= n?x \mid n!l \\
l, l_1, l_2, \ldots \in \mathit{SExp} &:= \mathsf{true} \mid \mathsf{false} \mid op(\vec{l_i}) \mid v \mid x
\end{aligned}$$

The set *Val* could be any flat domain of values such as the integers, in which *Op* would consist of the familiar operations of addition, subtraction etc.; we also assume that *Op* includes the Boolean operators. We ignore types, and assume that for any expression $\mathsf{if}\, l\, \mathsf{then}\, e_1\, \mathsf{else}\, e_2$ that l is a Boolean-valued expression, and that the use of the operator symbols *op* is type-respecting. We use the standard definition of free and bound variables for expressions, and use $\mathit{free}(e)$ to denote the set of free expression variables in e. We use $e\{\vec{v_i}/\vec{x_i}\}$ for the simultaneous substitution of values $\vec{v_i}$ for free expression variables $\vec{x_i}$ in e, while $e[\vec{e_i}/\vec{X_i}]$ denotes the simultaneous substitution of terms $\vec{e_i}$ for free process variables $\vec{X_i}$ in e. We use *VPL* to denote the set of closed expressions in *Exp*, which we refer to as *processes*. The constructs of *VPL* have the following informal meaning:

- $\mathsf{if}\, l\, \mathsf{then}\, e_1\, \mathsf{else}\, e_2$ – a process that behaves like e_1 if l evaluates to true, and like e_2 otherwise,
- $\alpha.e$ – a process that performs the communication action specified by α and then behaves like e,
- $e_1 \oplus e_2$ – a process that can evolve to either e_1 or e_2 without interaction with the environment,
- $e_1 + e_2$ – a process that behaves like e_1 or e_2 depending on the behaviour of the environment,
- $e_1 \| e_2$ – a process that allows the interleaving of the behaviours of e_1 and e_2, or communication between them,
- $e \backslash n$ – a process that behaves like e except that it cannot offer communications actions on channel n to the environment,
- $\mathbf{0}$ – the inactive process,
- $\mu X.e$ – the recursive process,
- $e[R]$ – a process that behaves like e except that the channel names of actions performed by e are renamed according to the renaming function R and,
- Ω – the undefined or divergent process

We now present the operational semantics for processes, and to make things simpler we ignore the evaluation of Boolean expressions. That is we assume that for each closed Boolean simple expression l there is a corresponding truth value $[\![l]\!]$ and more generally for any Boolean simple expression l

$$(\mathrm{Bot})\ \frac{}{\Omega \xrightarrow{\tau} \Omega} \qquad (\mathrm{Fix})\ \mu X.e \xrightarrow{\tau} e[\mu X.e/X] \quad (\mathrm{Com})\ \frac{e_1 \xrightarrow{a} e_1',\ e_2 \xrightarrow{\overline{a}} e_2'}{e_1 \parallel e_2 \xrightarrow{\tau} e_1' \parallel e_2'}$$

$$(\mathrm{IntL})\ e_1 \oplus e_2 \xrightarrow{\tau} e_1 \qquad (\mathrm{IntR})\ e_1 \oplus e_2 \xrightarrow{\tau} e_2 \qquad (\mathrm{Hide}\tau)\ \frac{e \xrightarrow{\tau} e'}{e\backslash n \xrightarrow{\tau} e'\backslash n}$$

$$(\mathrm{ExCL}\tau)\ \frac{e_1 \xrightarrow{\tau} e_1'}{e_1 + e_2 \xrightarrow{\tau} e_1' + e_2} \quad (\mathrm{ExCR}\tau)\ \frac{e_2 \xrightarrow{\tau} e_2'}{e_1 + e_2 \xrightarrow{\tau} e_1 + e_2'} \quad (\mathrm{Ren}\tau)\ \frac{e \xrightarrow{\tau} e'}{e[R] \xrightarrow{\tau} e'[R]}$$

FIGURE 1. Operational rules for reduction

$$(\mathrm{In})\ n?x.e \xrightarrow{n?v} e\{v/x\}\ \ \forall v \in \mathit{Val} \quad (\mathrm{Out})\ n!v.e \xrightarrow{n!v} e$$

$$(\mathrm{IfT})\ \frac{e_1 \xrightarrow{\mu} e_1',\ [\![l]\!] = \mathsf{true}}{\mathsf{if}\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \xrightarrow{\mu} e_1'} \qquad (\mathrm{ParL})\ \frac{e_1 \xrightarrow{\mu} e_1'}{e_1 \parallel e_2 \xrightarrow{\mu} e_1' \parallel e_2} \qquad (\mathrm{ExCL})\ \frac{e_1 \xrightarrow{a} e_1'}{e_1 + e_2 \xrightarrow{a} e_1'}$$

$$(\mathrm{IfF})\ \frac{e_2 \xrightarrow{\mu} e_2',\ [\![l]\!] = \mathsf{false}}{\mathsf{if}\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2 \xrightarrow{\mu} e_2'} \qquad (\mathrm{ParR})\ \frac{e_2 \xrightarrow{\mu} e_2'}{e_1 \parallel e_2 \xrightarrow{\mu} e_1 \parallel e_2'} \qquad (\mathrm{ExCR})\ \frac{e_2 \xrightarrow{a} e_2'}{e_1 + e_2 \xrightarrow{a} e_2'}$$

$$(\mathrm{RenAct})\ \frac{e \xrightarrow{a} e'}{e[R] \xrightarrow{R(a)} e'[R]} \qquad (\mathrm{Hide})\ \frac{e \xrightarrow{a} e'}{e\backslash n \xrightarrow{a} e'\backslash n}\ \ \mathit{chan}(a) \neq n$$

FIGURE 2. Operational rules for contexts.

and mapping ρ from expression variables to values, there is a Boolean value $[\![l]\!]\rho$. We also assume for each operator symbol $op \in Op$, that we have an associated function $[\![op]\!]$ over the set of values Val of the appropriate type and arity. The operational semantics of *VPL* is defined in terms of the three transition relations:

$$\begin{aligned} e &\xrightarrow{\tau} e' - \text{a single step evaluation from } e \text{ to } e' \\ e &\xrightarrow{n?v} e' - \text{the receipt of a value } v \text{ along channel } n \text{ by expression } e \\ e &\xrightarrow{n!v} e' - \text{the output of a value } v \text{ along channel } n \text{ by expression } e \end{aligned}$$

which are defined to be the least relations satisfying the rules in Figures 1 and 2, where:

$$\begin{aligned} a, a_1, a_2 \ldots \in \mathit{Act} &\stackrel{\text{def}}{=} \{n?v \mid n \in \mathit{Chan},\ v \in \mathit{Val}\} \cup \{n!v \mid n \in \mathit{Chan},\ v \in \mathit{Val}\} \\ \mu, \mu_1, \mu_2, \ldots \in \mathit{Act}_\tau &\stackrel{\text{def}}{=} \mathit{Act} \cup \{\tau\} \end{aligned}$$

and for any action $a \in \mathit{Act}$ we define its *complement* $\overline{a}$ by:

$$\begin{aligned} \overline{n?v} &\stackrel{\text{def}}{=} n!v \\ \overline{n!v} &\stackrel{\text{def}}{=} n?v \\ \overline{\tau} &\stackrel{\text{def}}{=} \tau \end{aligned}$$

and we use $\mathit{chan}(a)$ to denote the channel of a, e.g. $\mathit{chan}(n!v) = n$.

3 Guarantee and Strong Guarantee Testing

In this section we two predicates on processes, and use them to construct two preorders. The first predicate, which we call *can guarantee*, demands that each state of a process reachable by internal transitions can eventually perform a given prefix. Let *Pref* be the set of action prefixes defined by:

$$\pi, \pi', \ldots \in \mathit{Pref} \stackrel{\text{def}}{=} \{n?, n! \mid n \in \mathit{Chan}\}$$

with *InPref* the restriction of *Pref* to input prefixes, and likewise *OutPref* its restriction to output prefixes, and let $e \stackrel{\pi}{\Longrightarrow}$ denote that for some $v, e \stackrel{\pi v}{\Longrightarrow} e'$, we have:

DEFINITION 3.1. [Can Guarantee] e *can guarantee* π, written $e \downarrow^{\mathcal{G}} \pi$, if:

$$e \Downarrow \text{ and } e \stackrel{\varepsilon}{\Longrightarrow} e' \text{ implies } e' \stackrel{\pi}{\Longrightarrow}$$

□

For example:

$$n!v.\mathbf{0} \downarrow^{\mathcal{G}} n!$$

but:

$$n!v.\mathbf{0} \oplus \mathbf{0} \not\downarrow^{\mathcal{G}} n!$$

since:

$$n!v.\mathbf{0} \oplus \mathbf{0} \stackrel{\varepsilon}{\Longrightarrow} \mathbf{0}$$

The second predicate, called *can strongly guarantee*, is similar to *can guarantee* but demands in addition that whenever a given prefix can be performed, it can do so without leading to a divergent state. The definition of when a prefix leads to a divergent state differs depending on whether it is an input prefix or an output prefix. Let $\Downarrow\!\!\Downarrow$ be defined on *Pref* by:

$$e \Downarrow\!\!\Downarrow n? \text{ if } \exists v, \forall e' : e \stackrel{n?v}{\Longrightarrow} e', e' \Downarrow$$
$$e \Downarrow\!\!\Downarrow n! \text{ if } e \stackrel{n!v}{\Longrightarrow} e' \text{ implies } e' \Downarrow$$

If $e \Downarrow\!\!\Downarrow n?$ then this amounts to a guarantee to the environment that there is at least one value v which can sent to e along n, such that any continuation of e after inputting v along n will converge. For example $e_1 \Downarrow\!\!\Downarrow n?$ where:

$$e_1 \stackrel{\text{def}}{=} n?x.\mathsf{if}\ x = 1\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \Omega$$

but $e_2 \not\Downarrow\!\!\Downarrow n?$ where:

$$e_2 \stackrel{\text{def}}{=} (n?x.\mathsf{if}\ even(x)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \Omega) \oplus (n?x.\mathsf{if}\ odd(x)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \Omega)$$

since for all integers i there exists an e' such that $e \stackrel{n?i}{\Longrightarrow} e'$ and $e' \Uparrow$.

If $e \Downarrow\!\!\Downarrow n!$ then we can interact with e along n and be sure that whatever value v we receive, the continuation of e after outputting v along n will converge. For example we have that $e_3 \Downarrow\!\!\Downarrow n!$ where:

$$e_3 \stackrel{\text{def}}{=} n!v_1.\mathbf{0} \oplus n!v_2.\mathbf{0}$$

but $e_4 \not\Downarrow\!\!\Downarrow n!$ where:

$$e_4 \stackrel{\text{def}}{=} n!v_1.\mathbf{0} \oplus n!v_2.\Omega$$

We can now present the *strongly guarantee* predicate.

DEFINITION 3.2. [Can Strongly Guarantee] e *can strongly guarantee* π, written $e \downarrow^{\mathcal{SG}} \pi$, if:

$$e \Downarrow \text{ and } e \stackrel{\varepsilon}{\Longrightarrow} e' \text{ implies } e' \stackrel{\pi}{\Longrightarrow} \text{ and } e' \Downarrow\!\!\Downarrow \pi$$

□

Let $\mathbb{C}$ denote a context, i.e. a term e with one free process variable X, which we write as $[\,]$, and $\mathbb{C}[e']$ be the replacement of X in e by the closed term e'. We have:

DEFINITION 3.3. [Guarantee Testing] For $e_1, e_2 \in VPL$ we define $e_1 \sqsubseteq_{\mathcal{G}} e_2$ if for all contexts $\mathbb{C}$ and prefixes π:

$$\mathbb{C}[e_1] \downarrow^{\mathcal{G}} \pi \text{ implies } \mathbb{C}[e_2] \downarrow^{\mathcal{G}} \pi$$

□

and:

DEFINITION 3.4. [Strong Guarantee Testing] For $e_1, e_2 \in VPL$ we define $e_1 \sqsubseteq_{\mathcal{SG}} e_2$ if for all contexts $\mathbb{C}$ and prefixes π:

$$\mathbb{C}[e_1] \downarrow^{\mathcal{SG}} \pi \text{ implies } \mathbb{C}[e_2] \downarrow^{\mathcal{SG}} \pi$$

□

For both preorders, we define their kernels $\eqsim_{\mathcal{G}}$ and $\eqsim_{\mathcal{SG}}$ as $\sqsubseteq_{\mathcal{G}} \cap \sqsubseteq_{\mathcal{G}}^{-1}$ and $\sqsubseteq_{\mathcal{SG}} \cap \sqsubseteq_{\mathcal{SG}}^{-1}$ respectively.

The universal quantification over contexts in the definitions of the *guarantee* and *strong guarantee* preorders makes them unsuitable as a tractable theory of process behaviour. We now derive alternative characterisations of these preorders, which are defined independently of contexts. We begin with a brief review of *must* testing for *VPL* [Ing94], and its alternative characterisation.
We define the set of observers $O, O', \ldots \in \mathcal{O}$ to be the set of all closed terms formed from *Exp* by extending *Pre* with the action ω:

$$Pre := \ldots \mid \omega$$

Let $\| \subseteq (\mathcal{O} \times VPL)$ be the least relation satisfying the rules:

$$\begin{aligned} O \xrightarrow{\tau} O' &\text{ implies } O \| e \xrightarrow{\tau} O' \| e \\ e \xrightarrow{\tau} e' &\text{ implies } O \| e \xrightarrow{\tau} O \| e' \\ O \xrightarrow{a} O' \text{ and } e \xrightarrow{\overline{a}} e' &\text{ implies } O \| e \xrightarrow{\tau} O' \| e' \end{aligned}$$

A *computation* of $O \| e$ is any maximal finite or infinite sequence of τ transitions of the form:

$$O \| e = O_0 \| e_0 \xrightarrow{\tau} O_1 \| e_1 \xrightarrow{\tau} \cdots$$

and we use $Comp(O,e)$ to denote the set of all such computations for O and e. If c is a computation of $Comp(O,e)$ then we use c_i to denote the ith component of c, which is of the form $O_i \| e_i$. We say that e *must* O if for all computations $c \in Comp(O,e)$ we have $O_i \xrightarrow{\omega}$ for some component c_i of c.

Definition 3.5. [Must Testing] For $e_1, e_2 \in VPL$ define $e_1 \sqsubseteq_{\mathcal{MT}} e_2$ if for all $O \in \mathcal{O}$:

$$e_1 \textit{ must } O \text{ implies } e_2 \textit{ must } O$$

□

To present the alternative characterisation of *must* testing we require some auxiliary relations on the transition system induced by the operational semantics. Firstly we extend the transition relation $\longrightarrow$ to a relation $\Longrightarrow$ on sequences of actions, in the following way:

$$\begin{aligned} e &\overset{\varepsilon}{\Longrightarrow} e' \text{ if } e \xrightarrow{\tau}{}^* e' \\ e &\overset{a}{\Longrightarrow} e' \text{ if } e \xrightarrow{\tau}{}^* \circ \xrightarrow{a} \circ \xrightarrow{\tau}{}^* e' \\ e &\overset{a.s}{\Longrightarrow} e' \text{ if } e \overset{a}{\Longrightarrow} \circ \overset{s}{\Longrightarrow} e' \end{aligned}$$

We also extend the function $chan(\cdot)$ to finite sequences of actions in an obvious way. We say that a closed term e *diverges*, written $e \Uparrow$ or $e \not\Downarrow$, if there exists an infinite sequence of transitions of the form:

$$e = e_0 \xrightarrow{\tau} e_1 \xrightarrow{\tau} e_2 \xrightarrow{\tau} \cdots$$

We say e *converges*, written $e \Downarrow$, if e does not diverge, and we extend convergence to sequences of actions in Act^* by:

$$\begin{aligned} e \Downarrow \varepsilon &\text{ if } e \Downarrow \\ e \Downarrow n?v.s &\text{ if } e \Downarrow \text{ and } e \overset{n?v}{\Longrightarrow} e' \text{ implies } e' \Downarrow s \\ e \Downarrow n!v.s &\text{ if } e \Downarrow \text{ and } e \overset{n!v}{\Longrightarrow} e' \text{ implies } e' \Downarrow s \end{aligned}$$

We have:

$$\begin{aligned} \mathcal{S}(e) &\overset{\text{def}}{=} \{\pi \in Pref \mid \exists v : e \overset{\pi v}{\Longrightarrow}\} \; - \text{ the } \textit{successors} \text{ of } e \textit{ after } s \\ \mathcal{A}(e,s) &\overset{\text{def}}{=} \{\mathcal{S}(e') \mid e \overset{s}{\Longrightarrow} e' \overset{\tau}{\not\longrightarrow}\} \; - \text{ the } \textit{acceptances} \text{ of } e \textit{ after } s \end{aligned}$$

The acceptances of a process can be ordered in the following way:

$$\mathcal{A} \ll \mathcal{B} \text{ if } X \in \mathcal{A} \text{ implies } Y \subseteq X \text{ for some } Y \in \mathcal{B}$$

We can now define the alternative characterisation of $\sqsubseteq_{\mathcal{MT}}$:

Definition 3.6. For $e_1, e_2 \in VPL$ and $s \in Act^*$ define $e_1 \ll_{\mathcal{MT}} e_2$ if $e_1 \Downarrow s$ implies:

1. $e_2 \Downarrow s$ and,

2. $\mathcal{A}(e_2, s) \ll \mathcal{A}(e_1, s)$.

□

THEOREM 3.7. *For $e_1, e_2 \in VPL$:*

$$e_1 \sqsubseteq_{\mathcal{MT}} e_2 \text{ if and only if } e_1 \ll_{\mathcal{MT}} e_2$$

PROOF. See [Ing94], THEOREM 3.2.6 p. 53. □

We now present alternative characterisations of $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$. To do this we need another relation over closed terms and sequences of actions called *guaranteed convergence*, denoted by $\cdot \Downarrow^{\mathcal{G}} \cdot$, and defined by:

$$\begin{array}{rl} e \Downarrow^{\mathcal{G}} \varepsilon & \text{if } e \Downarrow \\ e \Downarrow^{\mathcal{G}} n?v.s & \text{if } e \Downarrow \text{ and } e \stackrel{n?v}{\Longrightarrow} e' \text{ implies } e' \Downarrow^{\mathcal{G}} s \\ e \Downarrow^{\mathcal{G}} n!v.s & \text{if } e \Downarrow \text{ and } e \stackrel{n!v}{\Longrightarrow} e' \text{ implies } e' \Downarrow^{\mathcal{G}} s \text{ and } e \downdownarrows n! \end{array}$$

Note that the defining clause for convergence on an output action requires that the expression converges for *all* values output on the given channel. For example:

$$n!v_1.\mathbf{0} + n!v_2.\mathbf{0} \Downarrow^{\mathcal{G}} n!v_1 \quad \text{but} \quad n!v_1.\mathbf{0} + n!v_2.\Omega \not\Downarrow^{\mathcal{G}} n!v_1$$

DEFINITION 3.8. For all $e_1, e_2 \in VPL$ and $s \in Act^*$ define $e_1 \ll_{\mathcal{G}} e_2$ if $e_1 \Downarrow^{\mathcal{G}} s$ implies:

1. $e_2 \Downarrow^{\mathcal{G}} s$ and,
2. $\mathcal{A}(e_2, s) \ll \mathcal{A}(e_1, s)$.

□

To define the alternative characterisation of $\sqsubseteq_{\mathcal{SG}}$ we need to isolate certain actions that an expression may perform and that always lead to a divergent state:

$$\begin{aligned} \mathcal{D}(e, s) &\stackrel{\text{def}}{=} \{n? \mid \forall v, \exists e' : e \stackrel{s.n?v}{\Longrightarrow} e', e' \Uparrow\} \cup \{n! \mid \exists v, e' : e \stackrel{s.n!v}{\Longrightarrow} e', e' \Uparrow\} \\ \mathcal{A}_{\mathcal{S}}(e, s) &\stackrel{\text{def}}{=} \{A \setminus \mathcal{D}(e, s) \mid A \in \mathcal{A}(e, s)\} \end{aligned}$$

We call $\mathcal{D}(e, s)$ the *divergences* of e *after* s, and $\mathcal{A}_{\mathcal{S}}(e, s)$ the *strong acceptances* of e *after* s. We sometimes write $\mathcal{A}_{\mathcal{S}}(e, s)$ as $\mathcal{A}(e, s) \setminus \mathcal{D}(e, s)$, or more generally for any $\mathcal{A}$ a finite set of finite sets:

$$\mathcal{A} \setminus X \stackrel{\text{def}}{=} \{A \setminus X \mid A \in \mathcal{A}\}$$

Using these constructs we can now define the alternative characterisation of $\sqsubseteq_{\mathcal{SG}}$:

DEFINITION 3.9. For $e_1, e_2 \in VPL$, $e_1 \ll_{\mathcal{SG}} e_2$ if $e_1 \Downarrow^{\mathcal{G}} s$ implies:

1. $e_2 \Downarrow^{\mathcal{G}} s$ and,
2. $\mathcal{A}_{\mathcal{S}}(e_2, s) \ll \mathcal{A}_{\mathcal{S}}(e_1, s)$.

□

The motivation for the definition of $\mathcal{A}_{\mathcal{S}}(e, s)$ and thus of $\ll_{\mathcal{SG}}$, is that the divergences of a process represent actions about which nothing can be strongly guaranteed; therefore they can be removed from the acceptances of the process. For example consider the following two processes:

$$e_1 \stackrel{\text{def}}{=} (n?x.\Omega + m!v.\mathbf{0}) \oplus m'!v'.\mathbf{0} \text{ and } e_2 \stackrel{\text{def}}{=} m!v.\mathbf{0} \oplus m'!v'.\mathbf{0}$$

We have:

$$\mathcal{A}(e_1, \varepsilon) = \{\{n?, m!\}, \{m'!\}\}$$

$$\neq \{\{m!\}, \{m'!\}\}$$
$$= \mathcal{A}(e_2, \varepsilon)$$

but:

$$\mathcal{A}_{\mathcal{S}}(e_1, \varepsilon) = \{\{m!\}, \{m'!\}\}$$
$$= \mathcal{A}(e_2, \varepsilon)$$

The only difference between e_1 and e_2 is that e_1 can perform an input at n; but any input will leave it in a divergent state which removes the possibility of the testing context from strongly guaranteeing anything. There is a close connection between the divergences $\cdot \Downarrow^{\mathcal{G}} \cdot$, $\cdot \downarrow^{\mathcal{SG}} \cdot$ and the divergences of a process, which is embodied in the following lemma:

LEMMA 3.10.

1. *For $s \in Act^*$, if $e_1 \Downarrow^{\mathcal{G}} s$ and $e_2 \Downarrow^{\mathcal{G}} s$ then $\mathcal{D}(e_2, s) \subseteq \mathcal{D}(e_1, s)$ and,*
2. *If $\pi \in \mathcal{D}(e, \varepsilon)$ then $e \not\downarrow^{\mathcal{SG}} \pi$.*

PROOF. We prove each part separately:

1. Suppose the hypotheses of the lemma are true, then $e_2 \Downarrow^{\mathcal{G}} s$. If $\pi \in \mathcal{D}(e_2, s)$ we want to show that $\pi \in \mathcal{D}(e_1, s)$ and there are two cases to the proof, depending on the form of π:
   - $\pi = n!$ for some n – since $n! \in \mathcal{D}(e_2, s)$, then for some v, e_2' we have that $e_2 \stackrel{s.n!v}{\Longrightarrow} e_2'$ and $e_2' \Uparrow$, i.e. $e_2 \Uparrow s.n!v$. Now $e_1 \Downarrow^{\mathcal{G}} s$ and by the definition of $\ll_{\mathcal{G}}$ it must be that for some v', e_1' that $e_1 \stackrel{s.n!v'}{\Longrightarrow} e_1'$ and $e_1' \Uparrow$, i.e. $\pi \in \mathcal{D}(e_1, s)$.
   - $\pi = n?$ for some n – since $\pi \in \mathcal{D}(e_2, s)$ then for all v there exists some e_2^v such that $e_2 \stackrel{s.n?v}{\Longrightarrow} e_2^v$ and $e_2^v \Uparrow$, i.e. for all v, $e_2 \Uparrow s.n?v$ and this in turn implies $e_1 \Uparrow s.n?v$ for all v. Since $e_1 \Downarrow^{\mathcal{G}} s$ then for all v there exists e_1^v such that $e_1 \stackrel{s.n?v}{\Longrightarrow} e_1^v$ and $e_1^v \Uparrow$, i.e. $\pi \in \mathcal{D}(e_1, s)$.
2. Suppose $\pi \in \mathcal{D}(e, \varepsilon)$, there are two cases to the proof, depending on the form of π:
   - $\pi = n!$ for some n – then for some v and e' we have $e \stackrel{n!v}{\Longrightarrow} e'$ and $e' \Uparrow$; therefore $e \stackrel{\varepsilon}{\Longrightarrow} e$ and $e \not\Downarrow n!$ i.e. $e \not\downarrow^{\mathcal{SG}} n!$,
   - $\pi = n?$ for some n – then for all v there exists some e_v such that $e \stackrel{\varepsilon}{\Longrightarrow} e_v$ and $e_v \Uparrow$; therefore $e \stackrel{\varepsilon}{\Longrightarrow} e$ and $e \not\Downarrow n?$ i.e. $e \not\downarrow^{\mathcal{SG}} n?$

□

We now show that $\sqsubseteq_{\mathcal{G}}$ coincides with $\ll_{\mathcal{G}}$, and that $\sqsubseteq_{\mathcal{SG}}$ coincides with $\ll_{\mathcal{SG}}$, and we begin with an outline of the proof strategy. First we show that $\ll_{\mathcal{G}}$ and $\ll_{\mathcal{SG}}$ preserve all finite contexts of the language, i.e. contexts which do not use the recursion operator $\mu X.\cdot$.

PROPOSITION 3.11. *For all $e_1, e_2 \in VPL$ and finite closed contexts $\mathbb{C}$ we have:*

1. *$e_1 \ll_{\mathcal{G}} e_2$ implies $\mathbb{C}[e_1] \ll_{\mathcal{G}} \mathbb{C}[e_2]$ and,*
2. *$e_2 \ll_{\mathcal{SG}} e_2$ implies $\mathbb{C}[e_1] \ll_{\mathcal{SG}} \mathbb{C}[e_2]$.*

PROOF. It is sufficient to show for each context $\mathbb{C} = [\,] \oplus e, [\,] + e, [\,] \parallel e, [\,][R], [\,]\backslash n$ and $\alpha.[\,]$ that:

1. $e_1 \ll_{\mathcal{G}} e_2$ implies $\mathbb{C}[e_1] \ll_{\mathcal{G}} \mathbb{C}[e_2]$ and,
2. $e_2 \ll_{\mathcal{SG}} e_2$ implies $\mathbb{C}[e_1] \ll_{\mathcal{SG}} \mathbb{C}[e_2]$.

The proof is by a detailed examination of the transitions of a process in each context above, and we omit it. □

To prove the general case of PROPOSITION 3.11 i.e. for arbitrary contexts, is far from straightforward. Firstly, we would need to lift the definitions of $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ terms with free process variables.

Secondly, given two terms e_1 and e_2 with free process variable X we would need to show that whenever $e_1 \sqsubseteq_{\mathcal{G}} e_2$ (resp. $e_1 \sqsubseteq_{\mathcal{SG}} e_2$) then $\mu X.e_1 \sqsubseteq_{\mathcal{G}} \mu X.e_2$ (resp. $\mu X.e_1 \sqsubseteq_{\mathcal{SG}} \mu X.e_2$). However the operational rule (Fix) in Figure 1 ensures that whenever a term of the form $\mu X.e$ performs a sequence of actions $\mu X.e \stackrel{s}{\Longrightarrow} e'$ then e' may contain $\mu X.e$ as a sub-term; this breaks the structural induction on syntax used in the proof of the proposition. In section 5 we will use the connection of the preorders with the denotational models we construct to lift this proposition to all contexts.
We then show how $\ll_{\mathcal{G}}$ and $\ll_{\mathcal{SG}}$ characterise the underlying definitions of $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ respectively:

Proposition 3.12. *For $e_1, e_2 \in VPL$ we have:*

1. *$e_1 \ll_{\mathcal{G}} e_2$ and $e_1 \downarrow^{\mathcal{G}} \pi$ implies $e_2 \downarrow^{\mathcal{G}} \pi$ and,*
2. *$e_1 \ll_{\mathcal{SG}} e_2$ and $e_1 \downarrow^{\mathcal{SG}} \pi$ implies $e_2 \downarrow^{\mathcal{SG}} \pi$*

Proof. We prove each part separately:

1. If $e_1 \ll_{\mathcal{G}} e_2$ and $e_1 \downarrow^{\mathcal{G}} \pi$ then $e_2 \Downarrow$ by definition of $\ll_{\mathcal{G}}$. We must show that whenever $e_2 \stackrel{\varepsilon}{\Longrightarrow} e_2' \not\longrightarrow$ then $e_2' \stackrel{\pi}{\longrightarrow}$. We know that $\mathcal{A}(e_2, \varepsilon) \ll \mathcal{A}(e_1, \varepsilon)$ so there exists some $X \in \mathcal{A}(e_1, \varepsilon)$ such that $X \subseteq \mathcal{S}(e_2')$, but $X = \mathcal{S}(e_1')$ where $e_1 \stackrel{\varepsilon}{\Longrightarrow} e_1' \not\longrightarrow$ in which case $\pi \in X$ therefore $e_2' \stackrel{\pi}{\longrightarrow}$ as well,
2. First note that $\pi \notin \mathcal{D}(e_2, \varepsilon)$ otherwise by lemma 3.10 and the definition of $\ll_{\mathcal{SG}}$ we would have:
$$\begin{aligned}\pi \in \mathcal{D}(e_2, \varepsilon) &\text{ implies } \pi \in \mathcal{D}(e_1, \varepsilon)\\ &\text{ implies } e_1 \not\downarrow^{\mathcal{SG}} \pi\end{aligned}$$
which is a contradiction; from this it is straightforward to show that $e_2 \stackrel{\varepsilon}{\Longrightarrow} e_2'$ implies $e_2' \Downarrow \pi$. It remains only to show that $e_2 \stackrel{\varepsilon}{\Longrightarrow} e_2'$ implies $e_2' \stackrel{\pi}{\Longrightarrow}$. Suppose $e_2 \stackrel{\varepsilon}{\Longrightarrow} e_2'$, since $e_2 \Downarrow$ we can extend e_2' to a stable state e_2'', and $\mathcal{S}(e_2'') \subseteq \mathcal{S}(e_2')$. Furthermore by the definition of $\ll_{\mathcal{SG}}$ we have that $X \in \mathcal{A}_{\mathcal{S}}(e_2, \varepsilon)$ where $X = \mathcal{S}(e_2'') \setminus \mathcal{D}(e_2, \varepsilon)$. Since $\pi \notin \mathcal{D}(e_2, \varepsilon) \subseteq \mathcal{D}(e_1, \varepsilon)$ we have that $Y \subseteq X$ for some $Y = \mathcal{S}(e_1') \setminus \mathcal{D}(e_1, \varepsilon)$ with $e_1 \stackrel{\varepsilon}{\Longrightarrow} e_1' \not\longrightarrow$. Furthermore since $e_1 \downarrow^{\mathcal{SG}} \pi$ we know that $\pi \in Y$ which implies $\pi \in \mathcal{S}(e_2')$ implies $e_2' \stackrel{\pi}{\Longrightarrow}$, as required.

□

Let $\sqsubseteq^f_{\mathcal{G}}$ and $\sqsubseteq^f_{\mathcal{SG}}$ denote the closure of $\cdot \downarrow^{\mathcal{G}} \cdot$ and $\cdot \downarrow^{\mathcal{SG}} \cdot$ respectively, under all *finite* contexts. We have as a corollary of propositions 3.11 and 3.12:

Corollary 3.13 (of propositions 3.11 and 3.12). *For $e_1, e_2 \in VPL$ we have:*

1. *$e_1 \ll_{\mathcal{G}} e_2$ implies $e_1 \sqsubseteq^f_{\mathcal{G}} e_2$ and,*
2. *$e_1 \ll_{\mathcal{SG}} e_2$ implies $e_1 \sqsubseteq^f_{\mathcal{SG}} e_2$*

□

It remains to show that $\sqsubseteq_{\mathcal{G}} \subseteq \ll_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}} \subseteq \ll_{\mathcal{SG}}$. To do this we show that the defining properties $\ll_{\mathcal{G}}$ and $\ll_{\mathcal{SG}}$ can be characterised in $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ respectively by particular classes of contexts. The first class of contexts we require enables us to test when a process can *guaranteed converge* on a sequence of actions in Act^*. For each $s \in Act^*$ let $\mathbb{N}_s$ be defined by:

$$\mathbb{N}_s \stackrel{\text{def}}{=} [\,] \parallel con(s)$$

where:

$$\begin{aligned}con(\varepsilon) &\stackrel{\text{def}}{=} \mathbf{0}\\ con(n_1?v.s) &\stackrel{\text{def}}{=} n_1!v.con(s)\\ con(n_1!v.s) &\stackrel{\text{def}}{=} n?x.\mathsf{if}\ x = v\ \mathsf{then}\ con(s)\ \mathsf{else}\ \mathbf{0}\end{aligned}$$

We have the following property for the context $\mathbb{N}_s$:

Proposition 3.14. *For $e \in VPL$ and $s \in Act^*$ we have:*

$$\mathbb{N}_s[e] \Downarrow \text{ if and only if } e \Downarrow^{\mathcal{G}} s$$

PROOF. For the *if* case we prove the contra-positive, so suppose that $e \not\Downarrow^{\mathcal{G}} s$. If $s = \varepsilon$ then we have $e \Uparrow$ in which case $\mathbb{N}_s[e] \Uparrow$ as well; otherwise for some s_1, s_2 with $s = s_1 s_2$ we have either:

- $e \xRightarrow{s_1} e', e' \Uparrow$ – in this case we can show that $\mathbb{N}_s[e] \xRightarrow{\varepsilon} e_1$ with either:
$$e_1 = e' \parallel \mathsf{if\ true\ then}\ con(s_2)\ \mathsf{else}\ \mathbf{0}$$
or,
$$e_1 = e' \parallel con(s')$$
and therefore $\mathbb{N}_s[e] \Uparrow$ or,
- $s_1 = s_1'.n!v$ and $e \xRightarrow{s_1'.n!v'} e'$ with $e' \Uparrow$ – in this case we can show that:
$$\mathbb{N}_s[e] \xRightarrow{\varepsilon} e' \parallel \mathsf{if\ false\ then}\ con(s'')\ \mathsf{else}\ \mathbf{0}$$
and therefore $\mathbb{N}_s[e] \Uparrow$.

The *only if* case is proved by induction on s. If $s = \varepsilon$ then $\mathbb{N}_s[e] = e \parallel \mathbf{0}$ and therefore $\mathbb{N}_s[e] \Downarrow$ implies $e \Downarrow$ trivially. If $s = n?v.s'$ and $e \xRightarrow{n?v} e'$ we have:

$$\begin{aligned} \mathbb{N}_s[e] &\xRightarrow{\varepsilon} e' \parallel con(s') \\ &= \mathbb{N}_{s'}[e'] \end{aligned}$$

and $\mathbb{N}_s[e] \Downarrow$ implies $\mathbb{N}_{s'}[e'] \Downarrow$ implies $e' \Downarrow s'$ by induction. If $s = n!v.s'$ and $e \xRightarrow{n!v'} e'$ we have either:

$$\begin{aligned} \mathbb{N}_s[e] &\xRightarrow{\varepsilon} e' \parallel \mathsf{if\ false\ then}\ con(s')\ \mathsf{else}\ \mathbf{0} \\ &= e' \parallel \mathbf{0} \end{aligned}$$

and $\mathbb{N}_s[e] \Downarrow$ implies $e' \parallel \mathbf{0} \Downarrow$ implies $e' \Downarrow$, or $e \xRightarrow{n!v} e'$ and:

$$\begin{aligned} \mathbb{N}_s[e] &\xRightarrow{\varepsilon} e' \parallel \mathsf{if\ true\ then}\ con(s')\ \mathsf{else}\ \mathbf{0} \\ &= e' \parallel con(s') \\ &= \mathbb{N}_{s'}[e'] \end{aligned}$$

and $\mathbb{N}_s[e] \Downarrow$ implies $\mathbb{N}_{s'}[e']$ implies $e' \Downarrow s'$ by induction. □

The next class of contexts characterises the sequences of actions a process can perform. Let $\mathbb{R}^n_{s,a}$ be defined by:

$$\mathbb{R}^n_{s,a} \stackrel{\text{def}}{=} [\,] \parallel rej(s, a, n)$$

where:

$$\begin{aligned} rej(\varepsilon, n_1?v, n) &\stackrel{\text{def}}{=} n_1!v.\mathbf{0} + n!w \\ rej(\varepsilon, n_1!v, n) &\stackrel{\text{def}}{=} n_1?x.\mathsf{if}\ x = v\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ n!w + n!w \\ rej(n_1?v.s, a, n) &\stackrel{\text{def}}{=} n_1!v.rej(s, a, n) + n!w \\ rej(n_1!v.s, a, n) &\stackrel{\text{def}}{=} n_1?x.\mathsf{if}\ x = v\ \mathsf{then}\ rej(s, a, n)\ \mathsf{else}\ n!w + n!w \end{aligned}$$

Let $\mathcal{L}(e)$ denote the *language* of e, defined by:

$$\mathcal{L}(e) \stackrel{\text{def}}{=} \{s \mid e \xRightarrow{s} e'\}$$

We have the following property for the context $\mathbb{R}^n_{s,a}$:

PROPOSITION 3.15. *For $e \in VPL, a \in Act, s \in Act^*$ and $n \in Chan$ a fresh channel, we have:*

$$e \Downarrow^{\mathcal{G}} sa \textit{ implies } sa \notin \mathcal{L}(e) \textit{ if and only if } \mathbb{R}^n_{s,a}[e] \downarrow^{\mathcal{G}} n!$$

PROOF. Suppose $e \Downarrow^{\mathcal{G}} sa$, for the *if* case we prove the contra-positive by showing that whenever $sa \in \mathcal{L}(e)$, i.e. $e \xRightarrow{sa} e'$ that:

$$\mathbb{R}^n_{s,a}[e] \xRightarrow{\varepsilon} e_1$$

where either $e_1 = \mathbf{0}$ or $e_1 = \mathsf{if\,true\,then\,0\,else}\, n!$, in which case $\mathbb{R}^n_{s,a}[e] \not\downarrow^{\mathcal{G}} n!$.

For the *only if* case we show by induction on s that whenever $\mathbb{R}^n_{s,a}[e] \stackrel{\varepsilon}{\Longrightarrow} e' \not\longrightarrow$ then $e' \stackrel{n!}{\Longrightarrow}$ which is possible, because if $e \Downarrow^{\mathcal{G}} s$ then all computations from $\mathbb{R}^n_{s,a}[e]$ are finite. □

By the structure of $\mathbb{R}^n_{s,a}$ we have as a corollary the following:

Corollary 3.16 (of proposition 3.15). *For $e \in VPL, a \in Act, s \in Act^*$ and $n \in Chan$ a fresh channel, we have:*

$$e \Downarrow^{\mathcal{G}} sa \text{ implies } sa \notin \mathcal{L}(e) \text{ if and only if } \mathbb{R}^n_{s,a}[e] \downarrow^{\mathcal{SG}} n!$$

□

We now present two classes of context of a particular form; the first characterises the *acceptances* of a process while the second characterises the *strong acceptances*. We begin with the characterisation of the acceptances of a process. If $s \in Act^*$ and $f : Chan \longrightarrow Chan$ we define $f(s)$ by:

$$f(\varepsilon) \stackrel{\text{def}}{=} \varepsilon$$
$$f(n?v.s) \stackrel{\text{def}}{=} f(n)?v.f(s) \text{ and,}$$
$$f(n!v.s) \stackrel{\text{def}}{=} f(n)!v.f(s)$$

Let $\mathbb{A}^n_s$, $\mathbb{T}^{n,f}_{s,A}$ and $\mathbb{G}^{n,f}_{s,A}$ be defined by:

$$\mathbb{A}^n_s \stackrel{\text{def}}{=} [\,] \parallel acc(s,n)$$
$$\mathbb{T}^{n,f}_{s,A} \stackrel{\text{def}}{=} ([\,] \parallel trans(s,f))[R^A_n]$$
$$\mathbb{G}^{n,f}_{s,A} \stackrel{\text{def}}{=} \mathbb{A}^n_{f(s)}[\mathbb{T}^{n,f}_{s,A}]$$

where:

$$acc(\varepsilon, n) \stackrel{\text{def}}{=} \mathbf{0}$$
$$acc(n_1?v.s, n) \stackrel{\text{def}}{=} n_1!v.acc(s,n) + n!$$
$$acc(n_1!v.s, n) \stackrel{\text{def}}{=} n_1?x.\mathsf{if}\ x = v\ \mathsf{then}\ acc(s,n)\ \mathsf{else}\ n! + n!$$

and:

$$trans(\varepsilon, f) \stackrel{\text{def}}{=} \mathbf{0}$$
$$trans(n?v.s, f) \stackrel{\text{def}}{=} f(n)?x.n!x.trans(s,f)$$
$$trans(n!v.s, f) \stackrel{\text{def}}{=} n?x.f(n)!x.trans(s,f)$$

and R^A_n is the renaming function defined by:

$$R^A_n(a) \stackrel{\text{def}}{=} \begin{cases} n & \text{if } a \in A \\ a & \text{otherwise} \end{cases}$$

We have the following property of the context $\mathbb{G}^{n,f}_{s,A}$:

Proposition 3.17. *Let $e \in VPL, s \in Act^*$ and A a finite subset of $Pref$, and choose N, n and f such that:*

- *$N \cup \{n\}$ is a finite subset of $Chan$ with $N \cap \{n\} = \emptyset$,*
- *N and n are completely fresh,*
- *$|N| = |chan(s)|$ and,*
- *$f : chan(s) \to N$ is a bijection*

then $e \Downarrow^{\mathcal{G}} s$ implies:

$$\mathbb{G}^{n,f}_{s,A}[e] \downarrow^{\mathcal{G}} n! \text{ if and only if for all } B \in \mathcal{A}(e,s), A \cap B \neq \emptyset$$

PROOF. Assume the hypotheses of the proposition are true; firstly we show that:

$$e \stackrel{\varepsilon}{\Longrightarrow} e' \text{ if and only if } \mathbb{T}^{n,f}_{s,A}[e] \stackrel{f(s)}{\Longrightarrow} \mathbb{T}^{n,f}_{\varepsilon,A}[e']$$

by induction on s. For the *only if* part of the proposition we prove the contra-positive, so suppose there exists $B \in \mathcal{A}(e,s)$ such that $A \cap B = \emptyset$. Therefore $e \stackrel{s}{\Longrightarrow} e' \not\xrightarrow{\tau}$ and $B = \mathcal{S}(e')$ for some e'. By examination of the transitions from $\mathbb{T}^{n,f}_{s,A}[e]$ we can show that:

$$\mathbb{T}^{n,f}_{s,A}[e] \stackrel{f(s)}{\Longrightarrow} (e' \parallel \mathbf{0})[R^A_n]$$

and therefore:

$$\mathbb{G}^{n,f}_{s,A}[e] \stackrel{\varepsilon}{\Longrightarrow} (e' \parallel \mathbf{0})[R^A_n] \parallel \mathbf{0}$$

Since $e' \not\xrightarrow{a}$ for any $a \in A$ we have that $\mathbb{G}^{n,f}_{s,A}[e] \not\downarrow^{\mathcal{G}} n!$. For the *only if* case the proof is by induction on s. $\square$

The class of contexts needed to characterise *strong acceptances* is similar to that for the acceptances, except we need to record some additional information in the context about the set of prefixes A. Let $In(A)$ denote the elements of A which are input prefixes, i.e. of the form $n?$ for some n, and f_A a finite partial function from $In(A)$ to Val. We define the context $\mathbb{S}^{n,f}_{s,A}$ by:

$$\mathbb{S}^{n,f}_{s,A} \stackrel{\text{def}}{=} [\,] \parallel strong(s, A, f, n)$$

where:

$$\begin{aligned}
strong(\varepsilon, A, f, n) &\stackrel{\text{def}}{=} strong(A, f, n)\\
strong(n?v.s, A, f, n) &\stackrel{\text{def}}{=} n!v.strong(s, A, f, n) + n!\\
strong(n!v.s, A, f, n) &\stackrel{\text{def}}{=} n?x.\mathsf{if}\ x = v\ \mathsf{then}\ strong(s, A, f, n)\ \mathsf{else}\ n! + n!
\end{aligned}$$

and:

$$\begin{aligned}
strong(A, f, n) &\stackrel{\text{def}}{=} \sum\{strong(\pi, f) \mid \pi \in A\}\\
strong(n?, f) &\stackrel{\text{def}}{=} n!f(n?).n!\\
strong(n!, f) &\stackrel{\text{def}}{=} n?x.n!
\end{aligned}$$

The set of prefixes A in the context $\mathbb{S}^{n,f}_{s,A}$ represent prefixes drawn from the strong acceptances of a process e after some sequence of actions s has been performed. In this case, by the definition of strong acceptances, we know that whenever $n? \in A$ then for some $v \in Val$ we have that $e \stackrel{sn?v}{\Longrightarrow} e'$ implies $e' \Downarrow$. The function f in the context $\mathbb{S}^{n,f}_{s,A}$ records for any $n? \in A$ a value satisfying this property. Note that this is not required for output prefixes $n! \in A$, since we know they converge after any value output.

PROPOSITION 3.18. *If $e \in VPL$, $s \in Act^*$, A is a finite subset of $Pref$ and $f : In(A) \to Val$ such that:*

- *$n! \in A$ implies $n! \notin \mathcal{D}(e, s)$ and,*
- *$n? \in A$ implies for all e' such that $e \stackrel{sn?f(n?)}{\Longrightarrow} e' : e' \Downarrow$*

then $e \Downarrow^{\mathcal{G}} s$ implies:

$$\mathbb{S}^{n,f}_{s,A}[e] \downarrow^{\mathcal{SG}} n! \textit{ if and only if for all } B \in \mathcal{A}_{\mathcal{S}}(e,s), A \cap B \neq \emptyset$$

PROOF. Similar to the proof of PROPOSITION 3.17. $\square$

The next two theorems show that we are a short step away from showing that $\ll_{\mathcal{G}}$ is an alternative characterisation of $\sqsubseteq_{\mathcal{G}}$, and $\ll_{\mathcal{SG}}$ is an alternative characterisation of $\sqsubseteq_{\mathcal{SG}}$.

THEOREM 3.19. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ implies } e_1 \ll_{\mathcal{G}} e_2$$

PROOF. We prove the contra-positive; suppose that $e_1 \sqsubseteq_{\mathcal{G}} e_2$ and $e_1 \not\ll_{\mathcal{G}} e_2$. If $e_1 \Downarrow^{\mathcal{G}} s$ and $e_2 \not\Downarrow^{\mathcal{G}} s$ then by PROPOSITION 3.14 we have that $\mathbb{N}_s[e_1] \parallel n! \downarrow^{\mathcal{G}} n!$ and $\mathbb{N}_s[e_2] \parallel n! \not\downarrow^{\mathcal{G}} n!$ for some fresh n which contradicts the hypothesis of the theorem, so we assume that $e_1 \Downarrow^{\mathcal{G}} s$ and $e_2 \Downarrow^{\mathcal{G}} s$. Suppose that $s \in \mathcal{L}(e_2)$; if $s = \varepsilon$ then $s \in \mathcal{L}(e_1)$ trivially, so assume $s = s'a$ for some $a \in Act$. By PROPOSITION 3.15 for some fresh n we have that $\mathbb{R}^n_{s,a}[e_1] \downarrow^{\mathcal{G}} n!$ and $\mathbb{R}^n_{s,a}[e_2] \not\downarrow^{\mathcal{G}} n!$ which again contradicts the hypothesis of the theorem, so we assume that $s \in \mathcal{L}(e_1)$. Finally suppose that $\mathcal{A}(e_2, s) \not\ll \mathcal{A}(e_1, s)$. Then for some $Y \in \mathcal{A}(e_2, s)$ we have that $X \not\subseteq Y$ for all $X \in \mathcal{A}(e_1, s)$, i.e. for each $X \in \mathcal{A}(e_1, s)$ there is some $\pi_X \in X$ such that $\pi_X \notin Y$. Let A be the set of all π_X for each $X \in \mathcal{A}(e_2, s)$ satisfying this property, and pick N, n and f such that they fulfill the hypotheses of PROPOSITION 3.17. Then we have $\mathbb{G}^{n,f}_{s,A}[e_1] \downarrow^{\mathcal{G}} n!$ but $\mathbb{G}^{n,f}_{s,A}[e_2] \not\downarrow^{\mathcal{G}} n!$ which again contradicts the hypothesis of the theorem, and so we assert that $e_1 \ll_{\mathcal{G}} e_2$. □

THEOREM 3.20. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \sqsubseteq_{\mathcal{SG}} e_2 \text{ implies } e_1 \ll_{\mathcal{SG}} e_2$$

PROOF. The proof differs from that of THEOREM 3.19 only in the treatment of the strong acceptances. Suppose than that $e_1 \Downarrow^{\mathcal{G}} s, e_2 \Downarrow^{\mathcal{G}} s$ and $s \in \mathcal{L}(e_1) \cap \mathcal{L}(e_2)$. If $\mathcal{A}_{\mathcal{S}}(e_2, s) \not\ll \mathcal{A}_{\mathcal{S}}(e_1, s)$ then for some $Y \in \mathcal{A}_{\mathcal{S}}(e_2, s)$ we have that $X \not\subseteq Y$ for all $X \in \mathcal{A}_{\mathcal{S}}(e_1, s)$, i.e. for all $A \in \mathcal{A}_{\mathcal{S}}(e_1, s)$ there exists $\pi_X \in X$ such that $\pi_X \notin Y$. Furthermore since each X is of the form $\mathcal{A}(e_1, s) \setminus \mathcal{D}(e_1, s)$ we know that $\pi_X \notin \mathcal{D}(e_1, s)$ and that whenever $\pi_X = n?$ for some n, then for some v we have that $e \stackrel{sn?v}{\Longrightarrow} e'$ implies $e' \Downarrow$. Let A be the set of all π_X satisfying this property and construct a function $f : In(A) \rightharpoonup_{fin} Val$ according to the hypothesis of PROPOSITION 3.18. Then $\mathbb{S}^{n,f}_{s,A}[e_1] \downarrow^{\mathcal{SG}} n!$ and $\mathbb{S}^{n,f}_{s,A}[e_2] \not\downarrow^{\mathcal{SG}} n!$ which contradicts the hypothesis of the theorem; therefore we assert that $e_1 \ll_{\mathcal{SG}} e_2$. □

From COROLLARY 3.13 we know that the converses of THEOREMS 3.19 and 3.20 hold for the finite preorders $\sqsubseteq^f_{\mathcal{G}}$ and $\sqsubseteq^f_{\mathcal{SG}}$ respectively. We will use the full abstraction result in section 6 to lift the results of this corollary to the full preorders $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$.

4 Comparing Guarantee, Strong Guarantee and Must Testing

In this section we compare $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ to each other and to $\sqsubseteq_{\mathcal{MT}}$, and prove some expressivity results about *VPL* with respect to these preorders. Our first result shows that $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ are both coarser than $\sqsubseteq_{\mathcal{MT}}$.

PROPOSITION 4.1. *We have:*

$$\sqsubseteq_{\mathcal{MT}} \subseteq \sqsubseteq_{\mathcal{G}} \subseteq \sqsubseteq_{\mathcal{SG}}$$

PROOF. It is sufficient to prove that:

1. $e_1 \sqsubseteq_{\mathcal{MT}} e_2$ and $e_1 \downarrow^{\mathcal{G}} \pi$ implies $e_2 \downarrow^{\mathcal{G}} \pi$, and
2. $e_1 \sqsubseteq_{\mathcal{G}} e_2$ and $e_1 \downarrow^{\mathcal{SG}} \pi$ implies $e_2 \downarrow^{\mathcal{SG}} \pi$

since for all contexts $\mathbb{C}$ we have:

$$e_1 \sqsubseteq_{\mathcal{MT}} e_2 \text{ implies } \mathbb{C}[e_1] \sqsubseteq_{\mathcal{MT}} \mathbb{C}[e_2]$$

and:

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ implies } \mathbb{C}[e_1] \sqsubseteq_{\mathcal{G}} \mathbb{C}[e_2]$$

Therefore if $e_1 \sqsubseteq_{\mathcal{MT}} e_2$ and $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} \pi$ then:

$$\begin{aligned} e_1 \sqsubseteq_{\mathcal{MT}} e_2 & \text{ implies } \mathbb{C}[e_1] \sqsubseteq_{\mathcal{MT}} \mathbb{C}[e_2] \\ & \text{ implies } \mathbb{C}[e_2] \downarrow^{\mathcal{G}} \pi \text{ by (1) above} \end{aligned}$$

as required. A similar argument can be used to prove $\sqsubseteq_{\mathcal{G}} \subseteq \sqsubseteq_{\mathcal{SG}}$.

Suppose then that $e_1 \sqsubseteq_{\mathcal{MT}} e_2$ and $e_1 \downarrow^{\mathcal{G}} \pi$, then $e_1 \ll_{\mathcal{MT}} e_2$ and $e_1 \Downarrow$ implies $e_2 \Downarrow$. Furthermore $\mathcal{A}(e_2, s) \ll \mathcal{A}(e_1, s)$ by THEOREM 3.7 so if $e_2 \stackrel{\varepsilon}{\Longrightarrow} e_2' \not\longrightarrow$ then for some $X \in \mathcal{A}(e_1, s)$ we have $X \subseteq \mathcal{S}(e_2')$ and $\pi \in X$ because $e_1 \downarrow^{\mathcal{G}} \pi$; therefore $e_2' \stackrel{\pi}{\Longrightarrow}$ and so $e_1 \sqsubseteq_{\mathcal{G}} e_2$.

Suppose $e_1 \sqsubseteq_{\mathcal{G}} e_2$ and $e_1 \downarrow^{\mathcal{SG}} \pi$. By the definition of $\cdot \downarrow^{\mathcal{SG}} \cdot$ and $\cdot \downarrow^{\mathcal{G}} \cdot$ we have that:

$$\begin{aligned} e_1 \downarrow^{\mathcal{SG}} \pi & \text{ implies } e_1 \downarrow^{\mathcal{G}} \pi \\ & \text{ implies } e_2 \downarrow^{\mathcal{G}} \pi \end{aligned}$$

so to prove $e_2 \downarrow^{\mathcal{SG}} \pi$ it is sufficient to show that $e_2 \Downarrow \pi$. If this is not the case then, either $\pi = n!$ and for some v we have $e_2 \stackrel{n!v}{\Longrightarrow} e_2'$ and $e_2' \Uparrow$ in which case $e_2 \not\Downarrow^{\mathcal{G}} n!v$, which implies $e_1 \not\Downarrow^{\mathcal{G}} n!v$ which implies $e_1 \not\downarrow^{\mathcal{SG}} n!$ which is a contradiction, or $\pi = n?$ and for all v there exists some e_2^v such that $e_2 \stackrel{n?v}{\Longrightarrow} e_2^v$ and $e_2^v \Uparrow$; again we can show that $e_1 \not\downarrow^{\mathcal{SG}} n?$ which is a contradiction, so we may assume $e_2 \Downarrow \pi$ as required. □

The following examples show that these inclusions are strict:

$$\begin{aligned} n!v_1.\Omega + n!v_2.\mathbf{0} & \not\sqsubseteq_{\mathcal{MT}} n!v_2.\Omega \\ n!v_1.\Omega + n!v_2.\mathbf{0} & \sqsubseteq_{\mathcal{G}} n!v_2.\Omega \\ n?x.\Omega & \not\sqsubseteq_{\mathcal{G}} \mathbf{0} \\ n?x.\Omega & \sqsubseteq_{\mathcal{SG}} \mathbf{0} \end{aligned}$$

We now show that under a slight modification of the operational rules for $\mathsf{if} \cdot \mathsf{then} \cdot \mathsf{else} \cdot$, $\sqsubseteq_{\mathcal{MT}}$ coincides with $\sqsubseteq_{\mathcal{G}}$. We replace the conditional expression of *VPL* with $\mathsf{if}^+ \cdot \mathsf{then} \cdot \mathsf{else} \cdot \mathsf{fi}$, which has the following behaviour:

$$\frac{[\![l]\!] = \mathsf{true}}{\mathsf{if}^+\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2\ \mathsf{fi} \stackrel{\tau}{\longrightarrow} e_1} \qquad \frac{[\![l]\!] = \mathsf{false}}{\mathsf{if}^+\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2\ \mathsf{fi} \stackrel{\tau}{\longrightarrow} e_2}$$

We refer to this version of the language and operational semantics as VPL^+. Furthermore we denote by $\sqsubseteq^+_{\mathcal{MT}}$ and $\sqsubseteq^+_{\mathcal{G}}$ the obvious definition of the preorders for this new language, and the extension of the observers of *VPL* by $\mathcal{O}^+$. Replacing if by if^+ in *VPL* weakens the testing power of the language with respect to $\sqsubseteq_{\mathcal{MT}}$. For example we have:

$$n!v_1.\Omega + n!v_2.\mathbf{0} \sqsubseteq^+_{\mathcal{MT}} n!v_2.\Omega$$

To attempt to distinguish between these processes, one requires an observer which tests for convergence after an output of v_1 on channel n; such an observer would need to be insensitive to the fact that the left-hand term diverges when outputting v_2 on the same channel. For $\sqsubseteq_{\mathcal{MT}}$ and *VPL*, an appropriate observer might take the form:

$$O \stackrel{\text{def}}{=} n?x.\mathsf{if}\ x = v_1\ \mathsf{then}\ (\omega.\mathbf{0} \oplus \omega.\mathbf{0})\ \mathsf{else}\ \omega.\mathbf{0}$$

and the operational rules for if ensure that success is assured even if the value v_2 is received, because we can infer:

$$\frac{\omega.\mathbf{0} \stackrel{\omega}{\longrightarrow} \mathbf{0}, [\![v_2 = v_1]\!] = \mathsf{false}}{\mathsf{if}\ v_2 = v_1\ \mathsf{then}\ \omega.\mathbf{0} \oplus \omega.\mathbf{0}\ \mathsf{else}\ \omega.\mathbf{0} \stackrel{\omega}{\longrightarrow} \mathbf{0}}$$

i.e. $\mathsf{if}\ v_2 = v_1\ \mathsf{then}\ \omega.\mathbf{0} \oplus \omega.\mathbf{0}\ \mathsf{else}\ \omega.\mathbf{0}$ is a success state, even in the presence of Ω. However this is not the case for if^+ since we can only infer:

$$\frac{[\![v_2 = v_1]\!] = \mathsf{false}}{\mathsf{if}^+\ v_2 = v_1\ \mathsf{then}\ \omega.\mathbf{0} \oplus \omega.\mathbf{0}\ \mathsf{else}\ \omega.\mathbf{0}\ \mathsf{fi} \stackrel{\tau}{\longrightarrow} \omega.\mathbf{0}}$$

We have the following lemma:

LEMMA 4.2. *Let $O \in \mathcal{O}^+$ be an open term with free variables $\vec{x}$, and ρ a substitution with $\vec{x_i} \subseteq dom(\rho)$, then:*

$$O\rho \xrightarrow{\omega} \text{ implies } O\rho' \xrightarrow{\omega}$$

for all substitutions with $\vec{x_i} \subseteq dom(\rho')$.

PROOF. The proof is by induction on the structure of O. □

The import of this lemma is that there are many more observers in $\mathcal{O}^+$ which are capable of performing the success action ω. This is precisely what makes $\sqsubseteq^+_{\mathcal{MT}}$ no more discriminating than $\sqsubseteq^+_{\mathcal{G}}$ for VPL^+.

THEOREM 4.3. *For $e_1, e_2 \in VPL^+$ we have:*

$$e_1 \sqsubseteq^+_{\mathcal{MT}} e_2 \text{ if and only if } e_1 \sqsubseteq^+_{\mathcal{G}} e_2$$

PROOF. The proof of the *only if* case uses the fact that for observers:

$$O_? \stackrel{\text{def}}{=} n!v.\omega \oplus n!v.\omega \text{ and,}$$
$$O_! \stackrel{\text{def}}{=} n?x.\omega \oplus n?x.\omega$$

$O_? \parallel e$ is successful in the *must* sense precisely when $e \downarrow^{\mathcal{G}} n?$ and, $O_! \parallel e$ is successful in the *must* sense precisely when $e \downarrow^{\mathcal{G}} n!$. Therefore for any context $\mathbb{C}$ we have:

$$\begin{aligned} \mathbb{C}[e_1] \downarrow^{\mathcal{G}} n! \;\; & \text{implies } \; O_! \parallel \mathbb{C}[e_1] \\ & \text{implies } \; O_! \parallel \mathbb{C}[e_2] \\ & \text{implies } \; \mathbb{C}[e_2] \downarrow^{\mathcal{G}} n! \end{aligned}$$

as required, and similarly for $n?$ using $O_?$.

For the *if* case suppose that $e \sqsubseteq^+_{\mathcal{G}} f$, e *must* O and c is a computation of the form:

$$O \parallel f = O_0 \parallel f_0 \xrightarrow{\tau} O_1 \parallel f_1 \xrightarrow{\tau} \cdots$$

we must show that $O_i \xrightarrow{\omega}$. By deconstructing c we have $O \stackrel{\bar{s}}{\Longrightarrow}$ and $f \stackrel{s}{\Longrightarrow}$, and there are two cases:

- $e \Downarrow^{\mathcal{G}} s$ – in this case we can show that $e \stackrel{s}{\Longrightarrow}$ follows from $e \sqsubseteq^+_{\mathcal{G}} f$ and so we may construct a computation c' of the form:

  $$O \parallel e = O_0 \parallel e_0 \xrightarrow{\tau} O_1 \parallel e_1 \xrightarrow{\tau} \cdots$$

  and since e *must* O we have that $O_i \xrightarrow{\omega}$ as required.

- $e \not\Downarrow^{\mathcal{G}} s$ – there are two sub-cases:
  - there exists some prefix s' of s such that $e \stackrel{s'}{\Longrightarrow} e'$ and $e' \Uparrow$ – as in the previous case we may construct a computation of the form:

    $$O \parallel e = O_0 \parallel e_0 \xrightarrow{\tau} O_1 \parallel e_1 \xrightarrow{\tau} \cdots O_k \parallel e' \xrightarrow{\tau} \cdots$$

    and therefore $O_i \xrightarrow{\omega}$ for $o \leq i \leq k$, as required.
  - there exists some prefix $s''.n!v$ of s and $e \stackrel{s''.n!v'}{\Longrightarrow} e'$ with $e' \Uparrow$ – therefore $e \stackrel{s''}{\Longrightarrow} e'' \xrightarrow{n!v'} e'$ for some e'' and we can construct a computation of the form:

    $$O \parallel e = O_0 \parallel e_0 \xrightarrow{\tau} O_1 \parallel e_1 \xrightarrow{\tau} \cdots e'' \parallel O_k \xrightarrow{\tau} e' \parallel O_{k+1}$$

    Suppose $O_i \stackrel{\omega}{\not\longrightarrow}$ for all $i \leq k$, (if this is not the case then for some $i \leq k$ we have $O_i \xrightarrow{\omega}$ which is enough to make c a successful computation), then since $e' \Uparrow$ we must have that $O_{k+1} \xrightarrow{\omega}$ because e *must* O. Consider O_{k+1} which must be of the form $O'[v'/x]$ where $O_k \xrightarrow{n?v'} O'[v'/x]$. By LEMMA 4.2 we have that $O'[v/x] \xrightarrow{\omega}$ and therefore c is successful.

□

Another interesting property of $\sqsubseteq_{\mathcal{G}}$ is that its discriminatory power is dependent on the presence of the renaming operator; this is implicit in the proof of PROPOSITION 3.17. For example suppose that the operator $[R]$ is removed from the language, then we have no way of distinguishing between the two terms:

$$e_1 \stackrel{\text{def}}{=} n_1!v.((n_2!v.\Omega + n_3!v.\Omega) \oplus (n_4!v.\Omega + n_5!v.\Omega)) \text{ and: } e_2 \stackrel{\text{def}}{=} n_1!v.(n_3!v.\Omega \oplus n_5!v.\Omega)$$

First note that we cannot use any of the prefixes $n_2 \dots n_5$ to distinguish between e_1 and e_2 because there is no context $\mathbb{C}$ and $n_i!$ for $2 \le i \le 5$ such that $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} n_i!$. If we try to utilise some fresh prefix π, then we run into problems because any context that tries to communicate with sub-terms $(n_2!v.\Omega + n_3!v.\Omega)$ or $(n_4!v.\Omega + n_5!v.\Omega)$ of e_1 to guarantee π, will leave e_1 in a divergent state. The renaming operator allows the context to avoid making any communication, by renaming the actions of the process that we wish to communicate with to some fresh action. Note that e_1 and e_2 are distinguished in $\sqsubseteq_{\mathcal{G}}$ by the context:

$$\mathbb{C} \stackrel{\text{def}}{=} ([\,] \parallel n?x.\mathbf{0})[R]$$

where:

$$R(n_i) = \begin{cases} n' & \text{if } i = 2, 4 \\ n_i & \text{otherwise} \end{cases}$$

where n' is a fresh channel name, since $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} n'!$ and $\mathbb{C}[e_2] \not\downarrow^{\mathcal{G}} n'!$.
To recapture the testing power of $\sqsubseteq_{\mathcal{G}}$ without the renaming operator we need to strengthen the predicate $\cdot \downarrow^{\mathcal{G}} \cdot$ to sets of prefixes, i.e. we need to define $\cdot \downarrow^{\mathcal{G}} \cdot$ as:

$$e \downarrow^{\mathcal{G}} A \text{ if } e \Downarrow \text{ and } e \stackrel{\varepsilon}{\Longrightarrow} e' \text{ implies } e' \stackrel{\pi}{\Longrightarrow} \text{ for some } \pi \in A$$

Let $\sqsubseteq$ be the preorder derived from the above definition of $\cdot \downarrow^{\mathcal{G}} \cdot$ by closing up under all contexts. Then we have $e_1 \not\sqsubseteq e_2$ since $\mathbb{C}[e_1] \downarrow^{\mathcal{G}} \{n_1!, n_3!\}$ and $\mathbb{C}[e_2] \not\downarrow^{\mathcal{G}} \{n_1!, n_3!\}$ where:

$$\mathbb{C} \stackrel{\text{def}}{=} ([\,] \parallel n?x.\mathbf{0})$$

We have the following result:

PROPOSITION 4.4. *For $e_1, e_2 \in$ VPL we have:*

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \textit{ if and only if } e_1 \sqsubseteq e_2$$

PROOF. The *if* case is immediate from the definition of $\sqsubseteq$ and $\cdot \downarrow^{\mathcal{G}} \cdot$. The *only if* case follows from the fact that for fresh n:

$$\mathbb{F}_A[e] \downarrow^{\mathcal{G}} n! \text{ if and only if } e \downarrow^{\mathcal{G}} A$$

where:

$$\mathbb{F}_A \stackrel{\text{def}}{=} [\,][R^A_n]$$

and:

$$R^A_n(a) \stackrel{\text{def}}{=} \begin{cases} n & \text{if } a \in A \\ a & \text{otherwise} \end{cases}$$

□

From the above example for the sub-language of *VPL* without the renaming operator, we have that $\sqsubseteq \subset \sqsubseteq_{\mathcal{G}}$.

The strong guarantee preorder $\sqsubseteq_{\mathcal{SG}}$ is not affected by the removal of the renaming operator from the language. This is because whenever $e \downarrow^{\mathcal{SG}} \pi$ we know that e will converge *after* it performs π in any stable state.

5 Denotational Semantics

In this section we construct two denotational models **G** and **SG** in which we can interpret *VPL*. The models are constructed to reflect the testing power of the equivalences $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$ respectively, and are derived from the value passing acceptance tree model, AT^v, presented in [Ing94]. We begin by reviewing some concepts from domain theory; the reader is invited to consult [Gun92, Plo81a, Pie91] for further details. We then give an overview of AT^v and discuss the modifications necessary to arrive at **G** and **SG**.

An ω-algebraic pointed complete partial order (ωpcpo, or just cpo) is an ordered set $\langle D, \leq_D\rangle$ where:

- $\leq_D$ is reflexive, anti-symmetric and transitive relation on D,
- there is an element $\perp_D \in D$ such that $\perp_D \leq_D d$ for all $d \in D$,
- every directed subset X of D has a least upper bound in D, written $\bigsqcup X$,
- there is a subset of the elements of D called the *compact elements*, written $\mathsf{K}(D)$, satisfying for all $k \in \mathsf{K}(D)$ and directed set X:

$$k \leq_D \bigsqcup X \text{ implies } k \leq_D d \text{ for some } d \in X \text{ and,}$$

- for each $d \in D$:

$$\{k \leq_D d \mid k \in \mathsf{K}(D)\} \text{ is directed, and } d = \bigsqcup\{k \leq_D d \mid k \in \mathsf{K}(D)\}$$

We refer to a set D satisfying these properties except that it may not contain a least element as a *pre-cpo*. We can construct a cpo from any pre-cpo D by adding a least element $\perp_D$ to D in a straightforward way. We denote by $\mathsf{up}(D)$ the pre-cpo D augmented in this fashion. We will also write $\mathsf{up}(f) : \mathsf{up}(D) \longrightarrow \mathsf{up}(E)$ to denote the strict extension of the function $f : D \longrightarrow E$ defined by:

$$\mathsf{up}(f)(d) \stackrel{\text{def}}{=} \begin{cases} \perp_E & \text{if } d = \perp_D \\ f(d) & \text{otherwise} \end{cases}$$

and which we extend to functions of the form $f : D^n \longrightarrow E$ in a pointwise manner. If D and E are cpos, then we say the function $f : D \longrightarrow E$ is *continuous* if for any directed set $X \subseteq D$, $f(X)$ is directed and:

$$\bigsqcup f(X) = f(\bigsqcup X)$$

We will use the term *domain* to refer to ω-pcpos, and *pre-domain* to ω-pre-cpos.

The model AT^v is derived as the initial fixed point of a domain equation in the category $\omega\mathbf{CPO}^E$ of ω-algebraic complete partial orders with embeddings [Gun92, Plo81a, Pie91]. The domain equation is constructed from the bi-functor F where:

$$F(I, O) = \mathsf{up}(\langle \mathcal{A}, (f_{in} : \mathit{InPref} \rightharpoonup_{\mathit{fin}} I) \uplus (f_{out} : \mathit{OutPref} \rightharpoonup_{\mathit{fin}} O)\rangle)$$

and:

- $\mathcal{A}$ is a finite set of finite sets of Pref called an *acceptance set*,
- I is a domain describing sequels to input prefixes in $|\mathcal{A}|$,
- O is a domain describing sequels to output prefixes in $|\mathcal{A}|$,
- $\mathit{dom}(f_{in}) \cup \mathit{dom}(f_{out}) = |\mathcal{A}|$ and,
- $\uplus$ is the disjoint union of sets.

Processes in *VPL* can input arbitrary values along channels which are then substituted for free variables in open terms. Accordingly the domain I describing the sequels to input prefixes is given by $(\mathit{Val} \longrightarrow D)$ the set of all total functions from the set of values Val to domain D, ordered pointwise. In contrast processes can only output a finite number of distinct values on a channel in any given state. Therefore the domain O is given by $(\mathit{Val} \rightharpoonup_{\mathit{fin}} D)$, the set of all finite and partial functions from Val to domain D, ordered by:

$$f \preceq g \text{ if } \mathit{dom}(g) \subseteq \mathit{dom}(f) \text{ and for all } v \in \mathit{dom}(g), g(v) \leq_D f(v)$$

In fact O under this ordering is a pre-domain as it lacks a least element. The domain AT^v is defined to be the initial fixed point in $\omega\mathbf{CPO}^E$ of the domain equation:

$$G(D) = F(\mathit{Val} \longrightarrow D, \mathit{Val} \rightharpoonup_{\mathit{fin}} D)$$

An *interpretation* of *VPL* in a domain D is given by a *semantic function* $D[\![\ ]\!]$ with type:

$$D[\![\ ]\!] : Exp \longrightarrow [Env_V \longrightarrow [Env_D \longrightarrow D]]$$

where Env_V denotes the set of *Val* environments: mappings from the set of variables *Var* to the set of values *Val*, and Env_D is the set of D environments: mappings from the set of process variables *VRec* to the model D. The function $D[\![\ ]\!]$ is defined by structural induction on expressions as:

$$\begin{aligned}
D[\![x]\!]\rho\sigma &= \rho(x)\\
D[\![\mathbf{0}]\!]\rho\sigma &= \mathbf{0}_D\\
D[\![\Omega]\!]\rho\sigma &= \bot\\
D[\![e[R]]\!]\rho\sigma &= rename_D\ R\,[\![e]\!]\rho\sigma\\
D[\![op(\vec{l_i})]\!]\rho\sigma &= [\![op]\!](\rho(\vec{l_i}))\ \text{ for each } op \in Op\\
D[\![\Box(\vec{e_i})]\!]\rho\sigma &= \Box_D(D[\![\vec{e_i}]\!]\rho\sigma)\quad \text{for } \Box \in \{\oplus, +, \|\}\\
D[\![\mu X.e]\!]\rho\sigma &= \mathsf{fix}(\lambda d.D[\![e]\!]\rho\sigma[X \mapsto d])\\
D[\![\mathsf{if}\ l\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2]\!]\rho\sigma &= \begin{cases} D[\![e_1]\!]\rho & \text{if } [\![l]\!]\rho\sigma = \mathsf{true}\\ D[\![e_2]\!]\rho & \text{otherwise}\end{cases}\\
D[\![n?x.e]\!]\rho\sigma &= in_D\ n\ \lambda v.D[\![e]\!]\rho[x \mapsto v]\sigma\\
D[\![n!l.e]\!]\rho\sigma &= \begin{cases} out_D\ n\ \rho(l)\ D[\![e]\!]\rho\sigma & \text{if } l \in Var\\ out_D\ n\ l\ D[\![e]\!]\rho\sigma & \text{otherwise}\end{cases}
\end{aligned}$$

where each of the functions $\Box_D, rename_D$ are continuous on D, and the functions in_D and out_D have type:

$$\begin{aligned}
in_D &: Chan \longrightarrow ((Val \longrightarrow D) \longrightarrow D)\\
out_D &: Chan \longrightarrow (Val \longrightarrow (D \longrightarrow D))
\end{aligned}$$

where in_D is continuous in its second argument and out_D is continuous in its third argument, and fix is the least fixed point operator. In [Ing94] an interpretation for *VPL* is given in domain AT^v.

The goal of the next section is to show how models $\mathbf{G}$ and $\mathbf{SG}$ are *fully abstract* with respect to the preorders $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$, i.e. that for terms $e_1, e_2 \in VPL$ we have:

$$\begin{aligned}
e_1 \sqsubseteq_{\mathcal{G}} e_2 &\ \text{ if and only if }\ \mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \text{ and,}\\
e_1 \sqsubseteq_{\mathcal{SG}} e_2 &\ \text{ if and only if }\ \mathbf{SG}[\![e_1]\!] \leq_{\mathbf{SG}} \mathbf{SG}[\![e_2]\!]
\end{aligned}$$

To see that the model AT^v is not fully abstract for $\sqsubseteq_{\mathcal{G}}$ under the interpretation in [Ing94], consider the term:

$$e \stackrel{\text{def}}{=} n!v_1.\mathbf{0} \oplus n!v_2.\Omega$$

We have:

$$\begin{aligned}
AT^v[\![e]\!] &= (out_{AT^v}\ n\ v_1\ \mathbf{0}_{AT^v}) \oplus_{AT^v} (out_{AT^v}\ n\ v_2\ \bot_{AT^v})\\
&= \langle \{\{n!\}\}, \{n! \mapsto \{v_1 \mapsto \bot, v_2 \mapsto \mathbf{0}_{AT^v}\}\}\rangle
\end{aligned}$$

but:

$$\begin{aligned}
AT^v[\![n!v_2.\Omega]\!] &= out_{AT^v}\ n\ v_2\ \bot_{AT^v}\\
&= \langle \{\{n!\}\}, \{n! \mapsto \{v_2 \mapsto \mathbf{0}_{AT^v}\}\}\rangle
\end{aligned}$$

and we know from section 3 that $e \eqsim_{\mathcal{G}} n!v_2.\Omega$. Clearly the definition of $out\ n\ v_2\ d$ is a special case when $d = \bot$. We must choose an interpretation which ensures for all v_1, v_2 and d that:

$$\begin{aligned}
(out\ n\ v_1\ \bot) \oplus (out\ n\ v_2\ d) &= (out\ n\ v_1\ d) \oplus (out\ n\ v_2\ \bot)\\
&= out\ n\ v_1\ \bot\\
&= out\ n\ v_2\ \bot
\end{aligned}$$

We do this by defining a domain $(Val \otimes D)$ whose components consist of finite subsets of $(Val \times D)$ such that:

$$(v_1 \otimes \bot) \oplus (v_2 \otimes d) = (v_1 \otimes d) \oplus (v_2 \otimes \bot)$$

$$
\begin{aligned}
&= v_1 \otimes \bot \\
&= v_2 \otimes \bot \\
&= \bot
\end{aligned}
$$

and using $(Val \otimes D)$ as the domain for modelling the sequels to output prefixes. Suppose D is a domain and $\oplus_D$ is a continuous function on D satisfying for all elements $d_1, d_2 \in D$:

$$d_1 \oplus_D d_2 \leq d_1 \tag{1}$$

$$d_1 \oplus_D d_2 = d_2 \oplus_D d_1 \tag{2}$$

$$d \oplus_D d = d \tag{3}$$

then the pair $\langle D, \oplus_D \rangle$ is called a *continuous upper semi-lattice* [Gun92, Hen94]. We will use the function $\oplus_D$ as the interpretation of the internal choice operator $\oplus$ of *VPL*. We sometimes write $\langle D, \oplus_D \rangle$ for the domain D with a continuous function $\oplus_D$ satisfying (1) – (3) above.

Suppose $\langle D, \oplus_D \rangle$ and $\langle E, \oplus_E \rangle$ are domains:

- $f : Val \times D \longrightarrow Val \times E$ is *right-linear* if for elements $d_1, d_2 \in D$:

$$f(v, d_1) \oplus_E f(v, d_2) = f(v, d_1 \oplus_D d_2)$$

- $f : D \longrightarrow E$ is *linear* if for $d_1, d_2 \in D$:

$$g(d_1 \oplus_D d_2) = g(d_1) \oplus_E g(d_2) \text{ and,}$$

- $f : Val \times D \longrightarrow E$ is *right-strict* if:

$$f(v, \bot_D) = \bot_E$$

For domain $\langle D, \oplus_D \rangle$ let $(Val \otimes D)$ be the set characterised by the following universal property:

1. there is a right-linear, right-strict function $\imath : Val \times D \longrightarrow Val \otimes D$ and,
2. if $\langle E, \oplus_E \rangle$ is a domain and $f : Val \times D \longrightarrow E$ a right-linear, right-strict function then there exists a unique strict linear function $g^f : Val \otimes D \longrightarrow E$ such that the following diagram commutes:

$$
\begin{tikzcd}
Val \times D \arrow[d, "\imath"'] \arrow[dr, "f"] & \\
Val \otimes D \arrow[r, "g^f"'] & E
\end{tikzcd}
$$

The universal property above gives an axiomatic definition of the domain $(Val \otimes D)$. We now give a concrete presentation of such a domain. To do this we need only consider its compact elements. These are defined to be the least set satisfying the rules:

$$\frac{}{\bot_K \in K}$$

$$\frac{V_f \subseteq_{fin} Val, K_f \subseteq_{fin} (\mathsf{K}(D) \setminus \{\bot_D\})}{\{(v, k_{i_1} \oplus_D \ldots \oplus_D k_{i_j}) \mid v \in V_f, \{k_{i_1}, \ldots, k_{i_j}\} \subseteq K_f\} \in K}$$

where $\subseteq_{fin}$ denotes a finite subset. We define a preorder $\sqsubseteq^{\sharp}$ on K by:

$$
\begin{aligned}
\bot_K \sqsubseteq^{\sharp} S &\quad \text{for all } S \in K \\
S_1 \sqsubseteq^{\sharp} S_2 &\quad \text{if } (v, k') \in S_2 \text{ implies } k \leq_D k' \text{ for some } (v, k) \in S_1
\end{aligned}
$$

Let $Val \otimes D$ be defined to be the completion by ideals [Gun92] of $\langle K, \sqsubseteq^{\sharp} \rangle$, which has compact elements of the form $\downarrow(S)$ for each $S \in K$ where:

$$\downarrow(S) \stackrel{\text{def}}{=} \{S' \mid S' \sqsubseteq^{\sharp} S\}$$

The semi-lattice function $\oplus_D$ can be extended to $Val \otimes D$ by the strict extension of $\oplus_{Val \otimes D}$ where:

$$X \oplus_{Val \otimes D} Y \stackrel{\text{def}}{=} G(X \cup Y)$$

where:

$$G \stackrel{\text{def}}{=} \mathsf{fix}(\lambda F.\lambda Z.(Z \cup F(\{(v, k_1 \oplus_D k_2) \mid \{(v,k_2),(v,k_2)\} \in Z\})))$$

We will write $\oplus_\otimes$ to refer to $\oplus_{Val \otimes D}$.

PROPOSITION 5.1. *$\langle Val \otimes D, \oplus_\otimes \rangle$ satisfies the universal property given above.*

PROOF. Let $\imath$ be defined on elements (v,k) of $\mathsf{K}(Val \times D)$ by:

$$\imath(v,k) \stackrel{\text{def}}{=} \begin{cases} \bot_K & \text{if } k = \bot_D \\ \{(v,k)\} & \text{otherwise} \end{cases}$$

If $f : Val \times D \longrightarrow E$ is right-linear and right-strict, we define $g^f : Val \otimes D \longrightarrow E$ on elements of K by:

$$g^f(S) \stackrel{\text{def}}{=} \begin{cases} \bot_E & \text{if } S = \bot_K \\ f(v_1,k_1) \oplus_E \ldots \oplus_E f(v_m,k_m) & \text{otherwise} \end{cases}$$

We have:

$$\begin{aligned} g^f(\imath(v,k)) &= g^f\{(v,k)\} \\ &= f(v,k) \end{aligned}$$

as required. Furthermore suppose $h : Val \otimes D \rightarrow E$ is a strict linear function satisfying $\imath; h = f$, then for any $S = \{(v_1,k_1),\ldots,(v_m,k_m)\} \in K$ we have:

$$\begin{aligned} h(S) &= h(\{(v_1,k_1,\ldots,(v_m,k_m))\}) & \\ &= h(\imath(v_1,k_1) \otimes_\oplus \cdots \otimes_\oplus \imath(v_m,k_m)) & \text{definition of } \otimes_\oplus \\ &= h(\imath(v_1,k_1)) \otimes_E \cdots \otimes_E h(\imath(v_m,k_m)) & h \text{ is linear} \\ &= f(v_1,k_1) \oplus_E \cdots \oplus_E f(v_m,k_m) & \text{property of } h \\ &= g^f(S) & \end{aligned}$$

as required. □

PROPOSITION 5.2. *If $\langle D, \oplus_D \rangle$ is a domain then so is $\langle Val \otimes D, \otimes_\oplus \rangle$.* □

Before we present the construction of domain **G** we need to review the concept of an *acceptance set* [Hen88]. Let $\mathcal{A}, \mathcal{B}, \ldots \in \mathcal{A}(Pref)$ denote the set of all finite subsets of finite subsets of *Pref*. A set $\mathcal{A} \in \mathcal{A}(Pref)$ is an *acceptance set* if it satisfies the following rules:

$$\begin{aligned} A_1, A_2 \in \mathcal{A} &\;\text{ implies }\; A_1 \cup A_2 \in \mathcal{A} \\ A_1, A_2 \in \mathcal{A} \text{ and } A_1 \subseteq A \subseteq A_2 &\;\text{ implies }\; A \in \mathcal{A} \end{aligned}$$

The closure operator c on acceptance sets is defined to be the least function satisfying the following rules:

$$\frac{A \in \mathcal{A}}{A \in c(\mathcal{A})} \qquad\qquad \frac{A, B \in c(\mathcal{A})}{A \cup B \in c(\mathcal{A})}$$

$$\frac{A_1 \subseteq A \subseteq A_2, A_1, A_2 \in c(\mathcal{A})}{A \in c(\mathcal{A})}$$

The set $c(\mathcal{A})$ is the least acceptance set containing $\mathcal{A}$. We have the following lemma which connects acceptance sets to the acceptances of a process defined section 3.

LEMMA 5.3. *If $|\mathcal{A}| = |\mathcal{B}|$ then $\mathcal{A} \ll \mathcal{B}$ if and only if $c(\mathcal{A}) \subseteq c(\mathcal{B})$.*

PROOF. See [Hen88] p. 88. □

Let $F_{\mathbf{G}}$ be the function on domains defined by:

$$\begin{aligned} F_{\mathbf{G}}(I,O) \stackrel{\text{def}}{=} \mathsf{up}(\{\langle \mathcal{A}, f_{in} \uplus f_{out}\rangle \mid\ & f_{in} : \mathit{InPref} \rightharpoonup_{fin} I, \\ & f_{out} : \mathit{OutPref} \rightharpoonup_{fin} O, \\ & \mathcal{A} \text{ is an acceptance set}, \\ & dom(f_{in}) \cup dom(f_{out}) = |\mathcal{A}|\}) \end{aligned}$$

and let $\oplus_{F_{\mathbf{G}}}$ be the strict extension of the following function:

$$\langle \mathcal{A}, f\rangle \oplus_{F_{\mathbf{G}}} \langle \mathcal{B}, g\rangle \stackrel{\text{def}}{=} \langle c(\mathcal{A} \cup \mathcal{B}), (f_{in} \oplus_I g_{in}) \uplus (f_{out} \oplus_O g_{out})\rangle$$

where for $f, g \in (X \rightharpoonup_{fin} D)$ we use $f \oplus_D g$ to denote the function $(f \oplus_D g)$ defined by:

$$(f \oplus_D g)(x) = \begin{cases} f(x) \oplus_D g(x) & \text{if } x \in dom(f) \cap dom(g) \\ f(x) & \text{if } x \in dom(f) \setminus dom(g) \\ g(x) & \text{if } x \in dom(g) \setminus dom(f) \end{cases}$$

The ordering on elements of $F_{\mathbf{G}}(I,O)$ is given by:

$\langle \mathcal{A}, f_{in} \uplus f_{out}\rangle \leq \langle \mathcal{B}, g_{in} \uplus g_{out}\rangle$ if:

$\mathcal{B} \subseteq \mathcal{A}$,
$f_{in} \preceq g_{in}$ and,
$f_{out} \preceq g_{out}$

Proposition 5.4. *If $\langle D, \oplus_D\rangle$ and $\langle E, \oplus_E\rangle$ are domains, then so is $\langle F_{\mathbf{G}}(D,E), \oplus_{F_{\mathbf{G}}}\rangle$.* □

The domain constructors $(\mathit{Val} \longrightarrow \cdot), (\mathit{Val} \otimes \cdot)$ and $F_{\mathbf{G}}(\cdot,\cdot)$ can be lifted to continuous functors [Plo81a, Pie91] on $\omega\mathbf{CPO}^E$ in a straightforward way, in which case we can define the model $\mathbf{G}$ as the initial fixed point in $\omega\mathbf{CPO}^E$ of the domain equation:

$$G(D) = F_{\mathbf{G}}(\mathit{Val} \longrightarrow D, \mathit{Val} \otimes D)$$

It remains only to provide an interpretation of *VPL* in $\mathbf{G}$. The interpretations of the operators, $\cdot + \cdot, \cdot \backslash n$ and $\cdot \| \cdot$ are the same as the interpretation of their counterparts in [Ing94]; the interpretation of $\mathit{rename}_{\mathbf{G}}$ is given in the appendix. The functions $\mathit{in}_{\mathbf{G}}$ and $\mathit{out}_{\mathbf{G}}$ are given by:

$$\begin{aligned} \mathit{in}_{\mathbf{G}}\ n\ f &\stackrel{\text{def}}{=} \langle \{\{n?\}\}, \{n? \mapsto f\}\rangle \text{ and,} \\ \mathit{out}_{\mathbf{G}}\ n\ v\ d &\stackrel{\text{def}}{=} \langle \{\{n!\}\}, \{n! \mapsto v \otimes d\}\rangle \end{aligned}$$

We have:

$$\begin{aligned} \mathit{out}_{\mathbf{G}}\ n\ v_1\ \bot &= \mathit{out}_{\mathbf{G}}\ n\ v_2\ \bot \\ &= \langle \{\{n!\}\}, \{n! \mapsto \bot\}\rangle \end{aligned}$$

and:

$$\begin{aligned} (\mathit{out}_{\mathbf{G}}\ n\ v_1\ \bot) \oplus_{\mathbf{G}} (\mathit{out}_{\mathbf{G}}\ n\ v_2\ d) &= (\mathit{out}_{\mathbf{G}}\ n\ v_1\ d) \oplus_{\mathbf{G}} (\mathit{out}_{\mathbf{G}}\ n\ v_2\ \bot) \\ &= \langle \{\{n!\}\}, \{n! \mapsto ((v_1 \otimes d) \oplus_{\otimes} (v_2 \otimes \bot))\}\rangle \\ &= \langle \{\{n!\}\}, \{n! \mapsto \bot\}\rangle \end{aligned}$$

The requirements for the construction of domain $\mathbf{SG}$ are very similar to those for $\mathbf{G}$, however we need to capture the strong acceptances of a process in the model, and this requires an extra component. To do this we need to define a generalisation of acceptance sets. If $\mathcal{A} \in \mathcal{A}(\mathit{Pref})$ and $X \subseteq |\mathcal{A}|$, then $\mathcal{A}$ is an X-*acceptance set* if it satisfies the following rules:

$$\begin{aligned} A \in \mathcal{A} &\text{ implies } A \cup X \in \mathcal{A} \\ A \in \mathcal{A} &\text{ implies } A \setminus X \in \mathcal{A} \\ A_1, A_2 \in \mathcal{A} &\text{ implies } A_1 \cup A_2 \in \mathcal{A} \\ A_1 \subseteq A \subseteq A_2 \text{ and } A_1, A_2 \in \mathcal{A} &\text{ implies } A \in \mathcal{A} \end{aligned}$$

For a given $\mathcal{A}$ and $X \subseteq |\mathcal{A}|$ we define $c_X(\mathcal{A})$ as the least set satisfy the rules:

$$\frac{A \in \mathcal{A}}{A \cup X \in c_X(\mathcal{A})} \qquad \frac{A \in \mathcal{A}}{A \setminus X \in c_X(\mathcal{A})}$$

$$\frac{A_1, A_2 \in c_X(\mathcal{A})}{A_1 \cup A_2 \in c_X(\mathcal{A})} \qquad \frac{A_1 \subseteq A \subseteq A_2, A_1, A_2 \in c_X(\mathcal{A})}{A \in c_X(\mathcal{A})}$$

We leave the proof of the following result to the reader:

PROPOSITION 5.5. *$c_X(\mathcal{A})$ is the least X-acceptance set containing $\mathcal{A}$.* □

Informally elements of **SG** are triples $\langle \mathcal{A}, X, f \rangle$ where:

- X is a subset of $|\mathcal{A}|$,
- $\mathcal{A}$ is an X-acceptance set and,
- f is a function recording the sequels to input and output prefixes, with domain $dom(f) = |\mathcal{A}| \setminus X$.

The set $\mathcal{A}$ represents the strong acceptances of a process in the same way that the acceptance set of an element $\langle \mathcal{A}, f \rangle$ of **G** represents the acceptances of a process. The set X represents the divergences of a process; the connection between the strong acceptances, divergences and X-acceptances is given by the following lemma, which is a generalisation of LEMMA 5.3:

LEMMA 5.6. *If $X \subseteq Y$, $X \subseteq |\mathcal{A}|$, $Y \subseteq |\mathcal{B}|$ and $|\mathcal{A}| \subseteq |\mathcal{B}|$ then:*

$$\mathcal{A} \setminus X \ll \mathcal{B} \setminus Y \text{ if and only if } c_X(\mathcal{A}) \subseteq c_Y(\mathcal{B})$$

PROOF. For the *only if* case suppose that the hypotheses of the lemma hold and that $\mathcal{A} \setminus X \ll \mathcal{B} \setminus Y$; we show by induction on the proof of the statement $A \in c_X(\mathcal{A})$ that $A \in c_Y(\mathcal{B})$, and there are four cases:

- $A = A_1 \setminus X$ with $A_1 \in \mathcal{A}$. Then $A \in \mathcal{A} \setminus X$ in which case, for some $B \in \mathcal{B} \setminus Y$, $B \subseteq A$. Now $A \subseteq |\mathcal{A}|$ which implies $A \subseteq |\mathcal{B}|$, and $|\mathcal{B}| \in c_Y(\mathcal{B})$. Therefore we have that:

  $$B \subseteq A \subseteq |\mathcal{A}|$$

  which implies $A \in cYB$, as required.
- $A = A_1 \cup X$ with $A_1 \in \mathcal{A}$. In this case we have:

  $$\begin{aligned} A_1 \in \mathcal{A} &\Rightarrow A_1 \setminus X \in \mathcal{A} \setminus X \\ &\Rightarrow B \subseteq A_1 \setminus X \text{ for some } B \in \mathcal{B} \setminus Y \\ &\Rightarrow B \subseteq A_1 \setminus X \subseteq |\mathcal{A}| \subseteq |\mathcal{B}| \\ &\Rightarrow A_1 \setminus X \in c_Y(\mathcal{B}) \end{aligned}$$

  Furthermore $X \subseteq Y$ implies:

  $$B \subseteq A_1 \setminus X \subseteq A_1 \cup X \subseteq (A_1 \setminus X) \cup Y$$

  and therefore $A_1 \cup X \in c_Y(\mathcal{B})$, as required.

The remaining cases $A = A_1 \cup A_2$ or $A_1 \subseteq A \subseteq A_2$ with $A_1, A_2 \in c_X(\mathcal{A})$ follow by induction.

For the *if* case we first show that if $Y \subseteq |\mathcal{B}|$ then:

$$A \in c_Y(\mathcal{B}) \text{ implies } B \subseteq A \text{ for some } B \in \mathcal{B} \setminus Y \tag{4}$$

by induction on the proof of the statement $A \in c_Y(\mathcal{B})$. Suppose that the hypotheses of the lemma are true and $A \in \mathcal{A} \setminus X$. Since $\mathcal{A} \setminus X \subseteq c_X(\mathcal{A})$, if $A \in \mathcal{A} \setminus X$ we have $A \in c_Y(\mathcal{B})$, and by (4) $B \subseteq A$ for some $B \in \mathcal{B} \setminus Y$. □

Let $F_{\mathbf{SG}}$ be the function on domains defined by:

$$\begin{aligned} F_{\mathbf{SG}}(I,O) \stackrel{\text{def}}{=} \mathsf{up}(\,\{\langle \mathcal{A}, X, f_{in} \uplus f_{out}\rangle \mid\ & X \subseteq |\mathcal{A}|, \\ & f_{in} : \mathit{InPref} \rightharpoonup_{\mathit{fin}} I, \\ & f_{out} : \mathit{OutPref} \rightharpoonup_{\mathit{fin}} O, \\ & \mathcal{A} \text{ is an } X\text{-acceptance set}, \\ & \mathit{dom}(f_{in}) \cup \mathit{dom}(f_{out}) = |\mathcal{A}| \setminus X\}) \end{aligned}$$

Let $\Omega : (\mathit{Pref} \rightharpoonup_{\mathit{fin}} D \times \mathit{Pref} \rightharpoonup_{\mathit{fin}} D) \longrightarrow \mathit{Pref}$ be the function defined by:

$$\Omega(f,g) \stackrel{\text{def}}{=} \{n? \mid \forall v \in \mathit{Val}, f(n?)(v) = \bot \text{ or } g(n?)(v) = \bot\} \cup \{n! \mid f(n!) = \bot \text{ or } g(n!) = \bot\}$$

and let $\oplus_{F_{\mathbf{SG}}}$ be the strict extension of the following function:

$$\langle \mathcal{A}, X, f\rangle \oplus_{F_{\mathbf{SG}}} \langle \mathcal{B}, Y, g\rangle \stackrel{\text{def}}{=} \langle c_Z(\mathcal{A} \cup \mathcal{B}), Z, (f_{in} \oplus_I g_{in})\lceil C \uplus (f_{out} \oplus_O g_{out})\lceil C\rangle$$

where:

$$\begin{aligned} C &\stackrel{\text{def}}{=} (|\mathcal{A}| \cup |\mathcal{B}|) \setminus Z \text{ and,} \\ Z &\stackrel{\text{def}}{=} X \cup Y \cup \Omega(f,g) \end{aligned}$$

and $f\lceil X$ denote the function f with domain restricted to the elements of X. The ordering on elements of $F_{\mathbf{SG}}(D,E)$ is given by:

$\langle \mathcal{A}, X, f_{in} \uplus f_{out}\rangle \le \langle \mathcal{B}, Y, g_{in} \uplus g_{out}\rangle$ if:

$\mathcal{B} \subseteq \mathcal{A}$,
$f_{in} \preceq g_{in}$,
$f_{out} \preceq g_{out}$ and,
$Y \subseteq X$

Proposition 5.7. *If $\langle D, \oplus_D\rangle$ and $\langle E, \oplus_E\rangle$ are domains, then so is $\langle F_{\mathbf{SG}}(D,E), \oplus_{F_{\mathbf{SG}}}\rangle$.*

Proof.

□

We can lift $F_{\mathbf{SG}}(\cdot,\cdot)$ to a continuous functor on $\omega\mathbf{CPO}^E$, and we define $\mathbf{SG}$ to be the initial fixed point of the domain equation:

$$D = F_{\mathbf{SG}}((\mathit{Val} \longrightarrow D), (\mathit{Val} \otimes D))$$

All that remains is to provide interpretations of *VPL* in $\mathbf{SG}$. The interpretations of $\mathbf{0}$ and the functions *in* and *out* are defined below:

$$\mathbf{0}_{\mathbf{SG}} \stackrel{\text{def}}{=} \langle \{\emptyset\}, \emptyset, \emptyset\rangle$$

$$\mathit{in}_{\mathbf{SG}}\ n\ f \stackrel{\text{def}}{=} \begin{cases} \langle \{\{n?\}\}, \emptyset, \{n? \mapsto f\}\rangle & \text{if } f \neq \bot_{\mathit{Val} \longrightarrow \mathbf{SG}} \\ \langle \{\{n?\}, \emptyset\}, \{n?\}, \emptyset\rangle & \text{otherwise} \end{cases}$$

$$\mathit{out}_{\mathbf{SG}}\ n\ v\ d \stackrel{\text{def}}{=} \begin{cases} \langle \{\{n!\}\}, \emptyset, \{n! \mapsto v \otimes d\}\rangle & \text{if } d \neq \bot \\ \langle \{\{n!\}, \emptyset\}, \{n!\}, \emptyset\rangle & \text{otherwise} \end{cases}$$

The interpretations of $\|_{\mathbf{SG}}$, $\cdot\backslash$ and $\cdot[R]$ are the same as for $\mathbf{G}$; the interpretation for $\cdot + \cdot$ is given in the appendix.

The definition of $\oplus_{F_{\mathbf{SG}}}$ deserves some explanation. Recall that $\oplus_{\mathbf{SG}}$ provides the interpretation of the internal choice operator $\oplus$ of *VPL*; consider the two *VPL* processes:

$$e_1 \stackrel{\text{def}}{=} n?x.\mathsf{if}\ \mathit{even}(x)\ \mathsf{then}\ \mathbf{0}\ \mathsf{else}\ \Omega \text{ and: } e_2 \stackrel{\text{def}}{=} n?x.\mathsf{if}\ \mathit{even}(x)\ \mathsf{then}\ \Omega\ \mathsf{else}\ \mathbf{0}$$

where $\mathit{even}(x)$ is true if x is divisible by two and false otherwise. We have:

$$e_i \not\sqsubseteq_{\mathcal{SG}} \mathbf{0} \text{ for } i = 1,2$$

by taking the contexts:

$$\mathbb{C}_1 \stackrel{\text{def}}{=} [\,] \,||\, n!2.m!.\mathbf{0} \text{ and,}$$
$$\mathbb{C}_2 \stackrel{\text{def}}{=} [\,] \,||\, n!1.m!.\mathbf{0}$$

since $\mathbb{C}_1[e_1] \downarrow^{\mathcal{SG}} m!, \mathbb{C}_2[e_2] \downarrow^{\mathcal{SG}} m!$ and obviously $\mathbb{C}_i[\mathbf{0}] \not\downarrow^{\mathcal{SG}} m!$. When e_1 and e_2 are combined using $\oplus$ the prefix $n?$ becomes a divergence of the process $e_1 \oplus e_2$, although it is not a divergence of either e_1 or e_2. Let f_{even} and f_{odd} be the functions which converge for even and odd values respectively, and diverge otherwise. From the definition of $\oplus_{\mathbf{SG}}$ we have:

$$\begin{aligned} \mathbf{SG}[\![e_1 \oplus e_2]\!] &= \langle \{\{n?\}\}, \emptyset, \{n? \mapsto f_{even}\}\rangle \oplus_{\mathbf{SG}} \langle \{\{n?\}\}, \emptyset, \{n? \mapsto f_{odd}\}\rangle \\ &= \langle \{\{n?\}, \emptyset\}, \{n?\}, \emptyset\rangle \\ &\leq \langle \{\emptyset\}, \emptyset, \emptyset\rangle \\ &= \mathbf{SG}[\![\mathbf{0}]\!] \end{aligned}$$

6 Full Abstraction

This section of the report is devoted to showing that $\sqsubseteq_{\mathcal{G}}$ is fully abstract for $\mathbf{G}$ and $\sqsubseteq_{\mathcal{SG}}$ is fully abstract for $\mathbf{SG}$ i.e. that for terms $e_1, e_2 \in \mathit{VPL}$ we have:

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ if and only if } \mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \text{ and,}$$
$$e_1 \sqsubseteq_{\mathcal{SG}} e_2 \text{ if and only if } \mathbf{SG}[\![e_1]\!] \leq_{\mathbf{SG}} \mathbf{SG}[\![e_2]\!]$$

We begin by outlining the proof technique used in [Ing94] to show that AT^v is fully abstract for $\sqsubseteq_{\mathcal{MT}}$. Firstly a new ordering $\ll_{AT^v}$ is defined on elements of AT^v, using concepts similar to those which provide the alternative characterisations for *must* testing. Secondly it is shown that $\ll_{AT^v}$ is *internally fully abstract* with respect to AT^v, i.e. for all $d_1, d_2 \in AT^v$:

$$d_1 \ll_{AT^v} d_2 \text{ if and only if } d_1 \leq_{AT^v} d_2$$

The goal is then to show that for all $e_2, e_2 \in \mathit{VPL}$:

$$e_1 \ll_{\mathcal{MT}} e_2 \text{ if and only if } AT^v[\![e_1]\!] \ll_{AT^v} AT^v[\![e_2]\!]$$

This is achieved by:

1. defining an equational proof systems P_{AT^v} on expressions, with judgements of the form $\vdash e_1 =_{\mathcal{M}} e_2$, which is sound for AT^v with respect to $\sqsubseteq_{\mathcal{MT}}$ and $\leq_{AT^v}$, i.e:
$$\vdash e_1 =_{\mathcal{M}} e_2 \Rightarrow e_1 \sqsubseteq_{\mathcal{MT}} e_2 \text{ and } AT^v[\![e_1]\!] \leq_{AT^v} AT^v[\![e_2]\!],$$
2. defining a class of closed expressions of a particular form, called *head normal forms*, and showing that for each $e \in \mathit{VPL}$ there exists a term $\mathit{hnf}(e)$ in head normal form, such that $\vdash e =_{\mathcal{M}} \mathit{hnf}(e)$ and,
3. showing that for terms $\mathit{hnf}(e_1)$ and $\mathit{hnf}(e_2)$ in head normal form:
$$\mathit{hnf}(e_1) \ll_{\mathcal{MT}} \mathit{hnf}(e_2) \text{ if and only if } AT^v[\![\mathit{hnf}(e_1)]\!] \ll_{AT^v} AT^v[\![\mathit{hnf}(e_2)]\!].$$

Using these results it is straightforward to prove full abstraction. For all $e_1, e_2 \in \mathit{VPL}$ we have:

$$\begin{array}{llll} e_1 \sqsubseteq_{\mathcal{MT}} e_2 & \text{if and only if} & \mathit{hnf}(e_1) \sqsubseteq_{\mathcal{MT}} \mathit{hnf}(e_2) & \text{by 1 and 2 above} \\ & \text{if and only if} & \mathit{hnf}(e_1) \ll_{\mathcal{MT}} \mathit{hnf}(e_2) & \text{by the alternative characterisation} \\ & \text{if and only if} & AT^v[\![\mathit{hnf}(e_1)]\!] \ll_{AT^v} AT^v[\![\mathit{hnf}(e_2)]\!] & \text{by 3 above} \\ & \text{if and only if} & AT^v[\![e_1]\!] \ll_{AT^v} AT^v[\![e_2]\!] & \text{by 1 and 2 above} \\ & \text{if and only if} & AT^v[\![e_1]\!] \leq_{AT^v} AT^v[\![e_2]\!] & \text{by internal full abstraction} \end{array}$$

The head normal forms defined in [Ing94] reflect the structure of elements of AT^v directly in the syntax of *VPL*. Suppose $\mathcal{A} \in \mathcal{A}(\mathit{Pref})$ is non-empty and for each $a \in |\mathcal{A}|$ there is an expression e_a satisfying:

1. If $a = n?$ then e_a has the form $n?x.e'$
2. If $a = n!$ then e_a has the form $\sum\{n!v.f(v) \mid v \in \mathit{dom}(f)\}$ where $f \in (\mathit{Val} \rightharpoonup_{\mathit{fin}} \mathit{VPL})$.

Then the term:

$$\bigoplus\{e_A \mid A \in \mathcal{A}\}$$

is in *head normal form* if each e_A is the simple sum form:

$$\sum\{e_a \mid a \in A\}$$

where $\bigoplus$ denotes the application of the operator $\oplus$ to a non-empty, finite set of expressions, and $\sum$ the application of $+$ to a finite set of expressions, where by convention if the set is finite then the expression denotes $\mathbf{0}$. Let $\cdot \xrightarrow{a}_{AT^v} \cdot$ be the least infix partial function on elements of AT^v satisfying the following rules:

$$\frac{d = \langle \mathcal{A}, f\rangle, f(n!)(v) = d'}{d \xrightarrow{n!v}_{AT^v} d'} \qquad \frac{d = \langle \mathcal{A}, f\rangle, f(n?)(v) = d'}{d \xrightarrow{n?v}_{AT^v} d'}$$

The relationship between terms in head normal form and their denotations in AT^v is embodied in the following lemma, and forms the crux of the full abstraction proof for AT^v and $\sqsubseteq_{\mathcal{MT}}$ in [Ing94]:

LEMMA 6.1. *For $e \in VPL$ we have:*

$$hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{a} e' \textit{ if and only if } AT^v[\![hnf(e)]\!] \xrightarrow{a}_{AT^v} AT^v[\![e']\!]$$

□

The close relationship between $\sqsubseteq_{\mathcal{MT}}, \sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{SG}}$, and the models AT^v, $\mathbf{G}$ and $\mathbf{SG}$ will enable us to take advantage of the full abstraction proof for $\sqsubseteq_{\mathcal{MT}}$ and AT^v. By PROPOSITION 4.1 we have that:

$$\vdash e_1 =_{\mathcal{M}} e_2 \text{ implies } e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ and } e_1 \sqsubseteq_{\mathcal{SG}} e_2$$

and it is also straightforward to show that:

$$\vdash e_1 =_{\mathcal{M}} e_2 \text{ implies } \mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \text{ and } \mathbf{SG}[\![e_1]\!] \leq_{\mathbf{SG}} \mathbf{SG}[\![e_2]\!]$$

i.e. that the proof system defined for $\sqsubseteq_{\mathcal{MT}}$ in [Ing94] is sound for $\mathbf{G}$ and $\mathbf{SG}$. In particular this means that for each convergent term e there is a head normal form $hnf(e)$ such that $e \eqsim_{\mathcal{G}} hnf(e)$ and $e \eqsim_{\mathcal{SG}} hnf(e)$, and also $\mathbf{G}[\![e]\!] = \mathbf{G}[\![hnf(e)]\!]$ and $\mathbf{SG}[\![e]\!] = \mathbf{SG}[\![hnf(e)]\!]$.
Let $\cdot \xrightarrow{a}_{\mathbf{G}} \cdot$ be the least infix partial function on elements of AT^v satisfying the following rules:

$$\frac{d = \langle \mathcal{A}, f\rangle, f(n!) = \bot}{d \xrightarrow{n!v}_{\mathbf{G}} d'} \qquad \frac{d = \langle \mathcal{A}, f\rangle, f(n?)(v) = d'}{d \xrightarrow{n?v}_{\mathbf{G}} d'}$$

$$\frac{d = \langle \mathcal{A}, f\rangle, \{(v, d')\} \subseteq f(n!)}{d \xrightarrow{n!v}_{\mathbf{G}} d'}$$

We can use $\xrightarrow{a}_{\mathbf{G}}$ to define an alternative characterisation of $\leq_{\mathbf{G}}$ on $\mathbf{G}$. Let $\cdot \Downarrow \cdot$ be the least relation on elements of $\mathbf{G}$ and Act^* satisfying:

$$d \Downarrow \varepsilon \text{ if } d \neq \bot$$

$$d \Downarrow a.s \text{ if } d \Downarrow \text{ and } d \xrightarrow{a}_{\mathbf{G}} d' \text{ implies } d' \Downarrow s$$

Let $\mathcal{A}(d, s)$, the *acceptances* of d *after* s be defined by:

$$\mathcal{A}(d, \varepsilon) \stackrel{\text{def}}{=} \begin{cases} \mathcal{A} & \text{if } d = \langle \mathcal{A}, f\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{A}(d, a.s) \stackrel{\text{def}}{=} \begin{cases} \mathcal{A}(d', s) & \text{if } d = \langle \mathcal{A}, f\rangle \text{ and } d \xrightarrow{a}_{\mathbf{G}} d' \\ \emptyset & \text{otherwise} \end{cases}$$

We can now present the alternative characterisation of $\leq_{\mathbf{G}}$ on $\mathbf{G}$:

DEFINITION 6.2. For $d_1, d_2 \in \mathbf{G}$ and $s \in Act^*$ let $d_1 \ll_{\mathbf{G}} d_2$ if $d_1 \Downarrow s$ implies:

- $d_2 \Downarrow s$ and,
- $\mathcal{A}(d_2, s) \subseteq \mathcal{A}(d_1, s)$.

□

THEOREM 6.3. *For $d_1, d_2 \in \mathbf{G}$ we have:*

$$d_1 \ll_{\mathbf{G}} d_2 \text{ if and only if } d_1 \leq_{\mathbf{G}} d_2$$

PROOF. For the *only if* case we first prove a sub-result showing that whenever $d_1 \ll_{\mathbf{G}} d_2$ and $d_1 \Downarrow a$ then $d_2 \xrightarrow{a}_{\mathbf{G}} d_2'$ implies $d_1 \xrightarrow{a}_{\mathbf{G}} d_1'$ for some d_1' with $d_1' \ll_{\mathbf{G}} d_2'$. We then show using this result that whenever $d_1 \leq_{\mathbf{G}} d_2$ and $d_1 \Downarrow s$ then $d_2 \Downarrow s$ and $\mathcal{A}(d_2, s) \subseteq \mathcal{A}(d_1, s)$, by induction on s.

For the *if* case we show that $d_1 \ll_{\mathbf{G}} d_2$ implies $d_1^n \leq_{\mathbf{G}} d_2$ for each $n \geq 0$ where d^k denotes the *kth finite approximation* to $d \in \mathbf{G}$ and where:

$$d = \bigsqcup_k \{d^k\}$$

□

Unfortunately the property of head normal forms embodied in LEMMA 6.1 does not hold for $\mathbf{G}$. For example consider the head normal form $e \stackrel{\text{def}}{=} n!v_1.\Omega \oplus n!v_2.\mathbf{0}$. We have:

$$e \xrightarrow{n!v_2} \mathbf{0}$$

but:

$$\mathbf{G}[\![e]\!] \xrightarrow{n!v_2}_{\mathbf{G}} \bot \neq \mathbf{G}[\![\mathbf{0}]\!]$$

however we do have the following result:

LEMMA 6.4.

- $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n?v} e'$ *if and only if* $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n?v}_{\mathbf{G}} \mathbf{G}[\![e']\!]$,
- $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{G}} d \neq \bot$ *implies* $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ *and* $\mathbf{G}[\![e']\!] = d$,
- $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{G}} \bot$ *if and only if* $\exists v' : hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v'} e'$ *and* $e' \Uparrow$,
- $hnf(e) \Downarrow^{\mathcal{G}} n!v$ *and* $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ *implies* $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{G}} \mathbf{G}[\![e']\!]$.

PROOF. By examination of the structure of head normal forms, and the interpretation of the operators of *VPL* in $\mathbf{G}$. □

The full abstraction result for $\sqsubseteq_{\mathcal{G}}$ and $\mathbf{G}$ is a consequence of the following two lemmas:

LEMMA 6.5. *For $e \in VPL$ we have:*

$$e \Downarrow^{\mathcal{G}} s \text{ if and only if } \mathbf{G}[\![e]\!] \Downarrow s$$

PROOF. The proof is by induction on s.

- $s = \varepsilon$ – For the *only if* case, we have $e \Downarrow$ implies $e \eqsim_{\mathcal{G}} hnf(e)$ and by the structure of head normal forms we have $\mathbf{G}[\![e]\!] = \mathbf{G}[\![hnf(e)]\!] \neq \bot$. For the *if* case we define *finite approximations* e^k, $k \geq 0$ to e such that $e^k \ll_{\mathbf{G}} e$ and $\mathbf{G}[\![e]\!] = \bigsqcup\{\mathbf{G}[\![e^k]\!] \mid k \geq 0\}$. In this case if $\mathbf{G}[\![e]\!] \neq \bot$ then $\mathbf{G}[\![e^k]\!] \neq \bot$ for some k, and we can show that this implies $e^k \Downarrow$ in which case by the definition of $\ll_{\mathbf{G}}$ then $e \Downarrow$ as well.
- $s = n!v.s'$ – For the *only if* case suppose that $e \Downarrow^{\mathcal{G}} s$, then $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{G}[\![e]\!] = \mathbf{G}[\![hnf(e)]\!]$. If $\mathbf{G}(hnf(e)) \xrightarrow{n!v}_{\mathbf{G}} d = \bot$ then by LEMMA 6.4 we have that $hnf(e) \not\Downarrow^{\mathcal{G}} n!v$ which is a contradiction; therefore $d \neq \bot$ in which case by LEMMA 6.4, we have that $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ and $\mathbf{G}[\![e']\!] = d$. Furthermore $e' \Downarrow^{\mathcal{G}} s'$ and therefore by induction we have that $d \Downarrow s'$ as required.

  For the if case suppose that $\mathbf{G}[\![e]\!] \Downarrow s$, then by the base case we have that $e \Downarrow$ which in turn implies that $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{G}[\![e]\!] = \mathbf{G}[\![hnf(e)]\!]$. Suppose $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v'} e'$ and $e' \Uparrow$, then by LEMMA 6.4 we have that $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{G}} \bot$ which contradicts the fact that $\mathbf{G}[\![e]\!] \Downarrow s$, so we may assume that $hnf(e) \Downarrow^{\mathcal{G}} n!v$. If $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ then by LEMMA 6.4 we have that $\mathbf{G}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{G}} \mathbf{G}[\![e']\!]$ and $\mathbf{G}[\![e']\!] \Downarrow s'$ by definition. Therefore by induction we may assume that $e' \Downarrow^{\mathcal{G}} s$.

- $s = n?v.s'$ – this case is simpler than the case $s = n!v.s'$.

□

LEMMA 6.6. *For all $e \in VPL$ and $s \in Act^*$, $e \Downarrow^{\mathcal{G}} s$ implies:*

$$\mathcal{A}(\mathbf{G}[\![e]\!], s) = c(\mathcal{A}(e, s))$$

PROOF. If $e \Downarrow^{\mathcal{G}} s$ then $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{G}[\![e]\!] = \mathbf{G}[\![hnf(e)]\!]$, so it is sufficient to show that:

$$\mathcal{A}(\mathbf{G}[\![hnf(e)]\!], s) = c(\mathcal{A}(hnf(e), s))$$

which follows by examination of the structure of head normal forms, the closure operator c and the interpretation of the operators $\oplus, extch$ and $\alpha.$ in $\mathbf{G}$. □

THEOREM 6.7. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ if and only if } \mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!]$$

PROOF. First show that:

$$e_1 \ll_{\mathcal{G}} e_2 \text{ if and only if } \mathbf{G}[\![e_1]\!] \ll_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \tag{5}$$

using LEMMAS 6.5 and 6.6. By (5) and THEOREMS 6.3 and 3.19 we have that:

$$e_1 \sqsubseteq_{\mathcal{G}} e_2 \text{ implies } e_1 \ll_{\mathcal{G}} e_2 \text{ implies } \mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \tag{6}$$

By (5), COROLLARY 3.13 and THEOREM 6.3 we have that:

$$\mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \text{ implies } e_1 \sqsubseteq^f_{\mathcal{G}} e_2 \tag{7}$$

so suppose that $\mathbb{C}$ is some arbitrary context; to complete the proof of the theorem we must show that:

$$\mathbf{G}[\![e_1]\!] \leq_{\mathbf{G}} \mathbf{G}[\![e_2]\!] \text{ implies } \mathbb{C}[e_1] \sqsubseteq^f_{\mathcal{G}} \mathbb{C}[e_2]$$

therefore:

$$\begin{aligned} \mathbf{G}[\![\mathbb{C}[e_1]]\!] &= \mathbb{C}_{\mathbf{G}}[\mathbf{G}[\![e_1]\!]] \text{ by compositionality of } \mathbf{G} \\ &\leq_{\mathbf{G}} \mathbb{C}_{\mathbf{G}}[\mathbf{G}[\![e_2]\!]] \\ &= \mathbf{G}[\![\mathbb{C}[e_2]]\!] \end{aligned}$$

implies:

$$\mathbb{C}[e_1] \sqsubseteq^f_{\mathcal{G}} \mathbb{C}[e_2]$$

by (7), as required. □

The proof of full abstraction for $\sqsubseteq_{\mathcal{SG}}$ and $\mathbf{SG}$ is very similar to that for $\mathbf{G}$ and $\sqsubseteq_{\mathcal{G}}$. Firstly we define an alternative characterisation of $\leq_{\mathbf{SG}}$ on $\mathbf{G}$. Let $\cdot \xrightarrow{a}_{\mathbf{SG}} \cdot$ be the least infix partial function on elements of AT^v satisfying the following rules:

$$\frac{d = \langle \mathcal{A}, X, f\rangle, \pi \in X}{d \xrightarrow{\pi v}_{\mathbf{SG}} \bot} \qquad \frac{d = \langle \mathcal{A}, X, f\rangle, f(n?)(v) = d'}{d \xrightarrow{n?v}_{\mathbf{SG}} d'}$$

$$\frac{d = \langle \mathcal{A}, X, f\rangle, \{(v, d')\} \subseteq f(n!)}{d \xrightarrow{n!v}_{\mathbf{SG}} d'}$$

For each $d \in \mathbf{SG}$ and $s \in Act^*$ let $\mathcal{D}(d, s)$ be defined by:

$$\mathcal{D}(d, \varepsilon) \stackrel{\text{def}}{=} \begin{cases} X & \text{if } d = \langle \mathcal{A}, X, f\rangle \\ \emptyset & \text{otherwise} \end{cases}$$

$$\mathcal{D}(d, a.s') \stackrel{\text{def}}{=} \begin{cases} \mathcal{D}(d', s') & \text{if } d = \langle \mathcal{A}, X, f\rangle \text{ and} \\ \emptyset & \text{otherwise} \end{cases}$$

and for each $d \in \mathbf{SG}$ and $s \in Act^*$ let $\mathcal{A}_{\mathcal{S}}(d,s)$ denote the obvious extension of the acceptances of d after s to elements of $\mathbf{SG}$.

DEFINITION 6.8. For $d_1, d_2 \in \mathbf{SG}$ and $s \in Act^*$ let $d_1 \ll_{\mathbf{SG}} d_2$ if $d_1 \Downarrow s$ implies:

- $d_2 \Downarrow s$,
- $\mathcal{D}(d_2, s) \subseteq \mathcal{D}(d_1, s)$ and,
- $\mathcal{A}_{\mathcal{S}}(d_2, s) \subseteq \mathcal{A}_{\mathcal{S}}(d_1, s)$.

□

We have the following result which is the analogue for $\mathbf{SG}$ of THEOREM 6.3:

THEOREM 6.9. *For $d_1, d_2 \in \mathbf{SG}$ we have:*

$$d_1 \ll_{\mathbf{SG}} d_2 \textit{ if and only if } d_1 \leq_{\mathbf{SG}} d_2$$

PROOF. Similar to the proof of THEOREM 6.3. □

We now prove an analogous result to LEMMA 6.4 which links the behaviour of head normal forms in the operational semantics and $\mathbf{SG}$.

LEMMA 6.10.

- $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{SG}} d \neq \perp$ *implies* $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ *and* $\mathbf{SG}[\![e']\!] = d$,
- $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{SG}} \perp$ *if and only if* $\exists v' : hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v'} e'$ *and* $e' \Uparrow$,
- $hnf(e) \Downarrow^{\mathcal{G}} n!v$ *and* $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n!v} e'$ *implies* $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n!v}_{\mathbf{SG}} \mathbf{SG}[\![e']\!]$,
- $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n?v}_{\mathbf{SG}} \perp$ *if and only if* $\forall v, \exists e' : hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n?v} e'$ *and* $e' \Uparrow$,
- $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n?v}_{\mathbf{SG}} d \neq \perp$ *implies* $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n?v} e'$ *and* $\mathbf{SG}[\![e']\!] = d$ *and*,
- $hnf(e) \stackrel{\varepsilon}{\Longrightarrow}\xrightarrow{n?v} e'$ *implies* $\mathbf{SG}[\![hnf(e)]\!] \xrightarrow{n?v}_{\mathbf{SG}} \mathbf{SG}[\![e']\!]$.

PROOF. Similar to the proof of LEMMA 6.4. □

Full abstraction for $\sqsubseteq_{\mathcal{SG}}$ and $\mathbf{SG}$ is a consequence of the following three lemmas:

LEMMA 6.11. *For $e \in VPL$ we have:*

$$e \Downarrow^{\mathcal{G}} s \textit{ if and only if } \mathbf{SG}[\![e]\!] \Downarrow s$$

PROOF. The proof is virtually identical to that of LEMMA 6.5 and uses LEMMA 6.10. □

LEMMA 6.12. *If $e \in VPL$ and $e \Downarrow^{\mathcal{G}} s$ then:*

$$\mathcal{D}(e, s) = \mathcal{D}(\mathbf{SG}[\![e]\!], s)$$

PROOF. If $e \Downarrow^{\mathcal{G}} s$ then we have $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{SG}[\![e]\!] = \mathbf{SG}[\![hnf(e)]\!]$, so it is sufficient to show that:

$$\mathcal{D}(hnf(e), s) = \mathcal{D}(\mathbf{SG}[\![hnf(e)]\!], s)$$

The proof is by induction on s, and uses LEMMA 6.10. □

LEMMA 6.13. *If $e \in VPL$ and $e \Downarrow^{\mathcal{G}} s$ then:*

$$c_{\mathcal{D}(e,s)}(\mathcal{A}_{\mathcal{S}}(e, s)) = \mathcal{A}_{\mathcal{S}}(\mathbf{SG}[\![e]\!], s)$$

PROOF. Since $e \Downarrow^{\mathcal{G}} s$ we have $e \eqsim_{\mathcal{G}} hnf(e)$ and $\mathbf{SG}[\![e]\!] = \mathbf{SG}[\![hnf(e)]\!]$, so it is sufficient to show that:

$$c_{\mathcal{D}(hnf(e),s)}(\mathcal{A}_{\mathcal{S}}(hnf(e), s)) = \mathcal{A}_{\mathcal{S}}(\mathbf{SG}[\![hnf(e)]\!], s)$$

The proof is by induction on s, and follows from LEMMA 6.10, the structure of head normal forms and the interpretations of the operators $\oplus, +$ and $\alpha.$ in $\mathbf{SG}$. □

We can now present our final result:

THEOREM 6.14. *For $e_1, e_2 \in VPL$ we have:*

$$e_1 \sqsubseteq_{\mathcal{SG}} e_2 \textit{ if and only if } \mathbf{SG}[\![e_1]\!] \leq_{\mathbf{SG}} \mathbf{SG}[\![e_2]\!]$$

PROOF. The proof is similar to the proof of THEOREM 6.7, and differs only in the proof of:

$$e_1 \ll_{\mathcal{SG}} e_2 \textit{ if and only if } \mathbf{SG}[\![e_1]\!] \ll_{\mathbf{SG}} \mathbf{SG}[\![e_2]\!]$$

For the *only if* case suppose $\mathbf{SG}[\![e_1]\!] \Downarrow s$, we have:

$$\begin{array}{llll} \mathbf{SG}[\![e_1]\!] \Downarrow s & \text{implies} & e_1 \Downarrow^{\mathcal{G}} s & \text{by LEMMA 6.11} \\ & \text{implies} & e_2 \Downarrow^{\mathcal{G}} s & \text{hypothesis} \\ & \text{implies} & \mathbf{SG}[\![e_2]\!] \Downarrow s & \text{by LEMMA 6.11} \end{array}$$

furthermore:

$$\begin{array}{llll} e_1 \ll_{\mathcal{SG}} e_2 & \text{implies} & \mathcal{A}_{\mathcal{S}}(e_2, s) \ll \mathcal{A}_{\mathcal{S}}(e_1, s) & \text{by definition} \\ & \text{and} & \mathcal{D}(e_2, s) \subseteq \mathcal{D}(e_1, s) & \text{by LEMMA 3.10} \end{array}$$

and:

$$\mathcal{D}(e_2, s) \subseteq \mathcal{D}(e_1, s) \text{ implies } \mathcal{D}(\mathbf{SG}[\![e_2]\!], s) \subseteq \mathcal{D}(\mathbf{SG}[\![e_1]\!], s) \text{ by LEMMA 6.12}$$

Therefore:

$$\begin{array}{llll} e_1 \ll_{\mathcal{SG}} e_2 & \text{if and only if} & \mathcal{A}(e_2, s) \setminus \mathcal{D}(e_2, s) \ll \mathcal{A}(e_1, s) \setminus \mathcal{D}(e_1, s) & \text{by definition} \\ & \text{implies} & c_{\mathcal{D}(e_2,s)}(\mathcal{A}(e_2, s)) \subseteq c_{\mathcal{D}(e_1,s)}(\mathcal{A}(e_1, s)) & \text{by LEMMA 5.6} \\ & \text{implies} & c_{\mathcal{D}(e_2,s)}(\mathcal{A}_{\mathcal{S}}(e_2, s)) \subseteq c_{\mathcal{D}(e_1,s)}(\mathcal{A}_{\mathcal{S}}(e_1, s)) & \text{since } c_X(\mathcal{A}) = c_X(\mathcal{A} \setminus X) \\ & \text{implies} & \mathcal{A}_{\mathcal{S}}(\mathbf{SG}[\![e_2]\!], s) \subseteq \mathcal{A}_{\mathcal{S}}(\mathbf{SG}[\![e_1]\!], s) & \text{by LEMMA 6.13} \end{array}$$

as required. The *if* case of the theorem is the same except we apply the relevant results in the reverse order. □

7 Conclusion

Our goal was to investigate the behavioural preorders obtained, by closing up basic notions of observability for a (version of) a value-passing process algebra, under all contexts of the language. The two notions of observability we considered, *can guarantee* and *can strongly guarantee*, and their respective preorders, *guarantee* testing and *strong guarantee* testing, provide complementary accounts to *must* testing as a behavioural and denotational theory for the value-passing process algebra *VPL*. However this work is not the first to attempt such an investigation. In [Foc95] Focardi introduces two behavioural preorders defined contextually from basic obervable properties, for CCS [Mil89]. The first preorder, denoted $\subseteq_{\mathbf{nr}\downarrow}$, corresponds to our *guarantee* testing preorder, while the second preorder, $\subseteq_{\mathbf{smust}}$, corresponds to our *strong guarantee* preorder. Focardi compares these preorders to each other, and to *must* testing, and also shows how $\subseteq_{\mathbf{smust}}$ corresponds to a re-formulation of *must* testing, for a slightly modified notion of success for a computation. No algebraic theory or models are developed for these preorders.

In [Fer96] and [FH97] a language very similar to *VPL* called *PAVP* (for Process Algebra with Value Production) is studied, where elements of *PAVP* may be viewed either as processes in the sense of *VPL*, or expressions of a first-order language which may evaluate to a canonical set of values. The production of a value is signalled by a transition of the form $e \xrightarrow{\surd v} e'$ where e' is a possibly non-trivial *side-effect* associated with the evaluation. Two contextual preorders are defined for *PAVP* using the production of a value as the basic unit of observability. The first preorder is the analogous version of *must* testing for this language, while the second preorder, called *guarantee* testing, is definitionally equivalent to DEFINITION 3.3, except that instead of the predicate $e \downarrow^{\mathcal{G}} n$, a predicate $e \downarrow v$ used where:

$$e \downarrow v \text{ if } e \Downarrow \text{ and } e \stackrel{\varepsilon}{\Longrightarrow} e' \stackrel{\tau}{\not\longrightarrow} \text{ implies } e' \stackrel{\surd v}{\longrightarrow}$$

However the operational semantics of *PAVP* is such that whenever $e \Downarrow$ and $e \stackrel{\surd v}{\longrightarrow} e'$ then $e' \Downarrow$ as well, so in effect *guarantee* testing in *PAVP* has the same testing power as *strong guarantee* testing in *VPL*. The work in this present report can be viewed as an extension of the work on *PAVP* to the more general setting of *VPL*. In particular in [Fer96] the model defined for *guarantee* testing is derived as a *retract* of the model for *must* testing, whereas the models in this report have been constructed from first principles.

Independently, Boreale, De Nicola and Pugliese [BNP97] have investigated testing preorders induced by observability predicates on processes, for the pure version of τ-less CCS [NH84, Hen88]. Using combinations of the predicates $!\ell$, $\downarrow$ and $\downarrow \ell$ defined by:

- $P!\ell$ if $P \stackrel{\varepsilon}{\Longrightarrow} P'$ implies $P' \stackrel{\ell}{\Longrightarrow}$,
- $P\downarrow$ if P converges and,
- $P \downarrow \ell$ if $P \stackrel{\varepsilon}{\Longrightarrow} P'$ implies $P' \stackrel{\ell}{\Longrightarrow} P''$ and $P'' \downarrow$.

they construct five contextual preorders. The preorder obtained by closing up the conjunction of $\downarrow$ and $!\ell$ under all contexts, is the analogous version of $\sqsubseteq_{\mathcal{G}}$ for pure τ-less CCS. In addition, the authors show that their version of $\sqsubseteq_{\mathcal{G}}$ coincides with *must* testing, which proves a conjecture in [Fer96]; we have seen that $\sqsubseteq_{\mathcal{G}}$ and $\sqsubseteq_{\mathcal{MT}}$ only coincide in the value-passing case (THEOREM 4.3) under a mild assumption about the operational semantics of the conditional expression. The preorder in [BNP97] defined from the conjunction of $\downarrow \ell$ and $\downarrow$, and denoted *safe-must* testing, corresponds to our *strong guarantee* preorder $\sqsubseteq_{\mathcal{SG}}$, although we need to treat the convergence of an action differently in the value-passing case.

References

[BNP97] M. Boreale, R. De Nicola, and R. Pugliese. Basic observables for processes. In *Proceedings of ICALP*. Springer-Verlag, 1997. To appear.

[Fer96] W. Ferreira. *Semantic Theories for Concurrent ML*. D.Phil thesis, University of Sussex, Department of Cognitive and Computing Sciences, February 1996.

[FH97] W. Ferreira and M. Hennessy. Testing value production in concurrent ML. *Theoretical Computer Science*, 1997. To appear.

[Foc95] Filippo Focardi. *Equivalenza e Giustizia nelle Algebre di Processo*. Tesi di Laurea, Facolta' di Scienze Matematiche Fisiche e Naturali, Universita' degli Studi di Roma *La Sapienza*, 1995. (In Italian).

[Gun92] Carl A. Gunter. *Semantics of Programming Languages*. MIT Press, Cambridge Massachusetts, 1992.

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

[Hen94] M. Hennessy. Higher-order processes and their models. In Serge Abiteboul and Eli Shamir, editors, *Proceedings ICALP*, volume 820 of *Lecture Notes in Computer Science*, pages 286–303. Springer-Verlag, 1994.

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

[Ing94] A. Ingólfsdóttir. *Semantic Models for Communicating Processes with Value Passing*. D.Phil thesis, University of Sussex, Department of Cognitive and Computing Sciences, 1994.

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

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

[Mor68] J. H. Morris. *Lambda Calculus Models of Programming Languages*. Ph.D. thesis, M.I.T., 1968.

[NH84] R. De Nicola and M. Hennessy. Testing equivalences for processes. *Theoretical Computer Science*, 24(0):83–113, 1984.

[Par81] D. M. R. Park. *Concurrency and Automata on Infinite Sequences*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer-Verlag, 1981.

[Pie91] Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT Press, Cambridge, Massachusetts, 1991.

[Plo81a] Gordon D. Plotkin. Lecture notes in domain theory, 1981.

[Plo81b] Gordon D. Plotkin. A structural approach to operational semantics. Technical Report DAIMI-FN-19, Computer Science Dept, Aarhus University, Denmark, 1981.

[San92] Davide Sangiorgi. *Expresing Mobility in Process Algebras: First Order and Higher-Order Paradigms.* Ph.D. thesis, LFCS, Edinburgh University, 1992.

A Interpretation of the remaining operators of *VPL* in G and SG.

Let $rename_{\mathbf{G}}$ be defined by $rename_{\mathbf{G}} \stackrel{\text{def}}{=} \lambda R.\,\mathsf{fix}(\lambda F.\mathsf{up}(\Theta_R\, F))$ where:

$$\Theta_R\, F\,\langle \mathcal{A}, f\rangle \stackrel{\text{def}}{=} \bigoplus_{\mathbf{G}}\{\sum_{\mathbf{G}} T_A \mid A \in \mathcal{A}\}$$

and:

$$T_A \stackrel{\text{def}}{=} \{in_{\mathbf{G}}\; R(n)\; \lambda v.F(f(n?)(v)) \mid n? \in A\} \;\cup\; \{out_{\mathbf{G}}\; R(n)\; v\; F(d) \mid n! \in A, \{(v,d)\} \subseteq f(n!)\}$$

and where we have used $\bigoplus_{\mathbf{G}}$ and $\sum_{\mathbf{G}}$ to denote the application of $\oplus_{\mathbf{G}}$ and $+_{\mathbf{G}}$ to finite subsets of $\mathbf{G}$.
Let $+_{\mathbf{SG}}$ be the strict extension of the following function:

$$\langle \mathcal{A}, X, f\rangle +_{\mathbf{SG}} \langle \mathcal{B}, Y, g\rangle \stackrel{\text{def}}{=} \langle c_Z(\mathcal{A} \vee \mathcal{B}), Z, (f_{in} \oplus_{\mathbf{SG}} g_{in})\lceil C \uplus (f_{out} \oplus_{\mathbf{SG}} g_{out})\lceil C\rangle$$

where:

$$C \stackrel{\text{def}}{=} (|\mathcal{A}| \cup |\mathcal{B}|) \setminus Z,$$

$$Z \stackrel{\text{def}}{=} X \cup Y \cup \Omega(f,g) \text{ and,}$$

$$\mathcal{A} \vee \mathcal{B} \stackrel{\text{def}}{=} \{A \cup B \mid A \in \mathcal{A}, B \in \mathcal{B}\}$$