Concurrent Testing of Processes *

M. Hennessy
University of Sussex

July 1, 1992

Abstract

We develop a noninterleaving semantic theory of processes based on testing. We assume that all actions have a non-zero duration and the allowed tests take advantage of this assumption. The result is a semantic theory in which concurrency is differentiated from nondeterminism.
We show that the semantic preorder based on these tests is preserved by so-called "stable" action refinements and may be characterised as the largest such preorder contained in the standard testing preorder.

*This work has been supported by the ESPRIT/BRA CEDISYS project and the SERC

1 Introduction

In recent years there has been much research into semantic theories of processes which distinguish nondeterminism from concurrency. See for example [DD89, DNM90, vGV87], [BC89]. Most of these are based on some variation of bisimulation equivalence, [Mil89]. This is a well-established behavioural equivalence between processes based on their ability to perform actions. Roughly speaking two processes p, q are bisimulation equivalent if whenever either can perform an action and be transformed into the process r then the other can also perform the same action and be transformed into some r' which is bisimulation equivalent to r. This equivalence leads to a so-called "interleaving" theory of concurrency in that it reduces parallelism to nondeterminism. For example the process which can perform the actions a and b in parallel is deemed to be equivalent to the purely sequential but nondeterministic process which either can perform a followed by b or b followed by a. However "non-interleaving" theories of concurrency, i.e. theories which distinguish parallelism from concurrency, can be obtained by varying the basic ingredients of the definition of bisimulation. For example the basic actions may be replaced by partial orders of actions where the order is induced by some idea of causality as in [DNM90] or the structure of processes may be taken into account as in [BCHK91].

Another well-established "interleaving" theory of processes is based on testing, [DH84]. Here processes are said to be equivalent if they are guaranteed to pass exactly the same tests. Although the framework of testing equivalence is quite general, apart from [MP91] it has only been applied, at least as far as the author is aware, to generate "interleaving theories". The purpose of this paper is use this framework to develop a "non-interleaving theory" of processes and in particular to investigate the application of this theory to action refinement.

In the standard theory a test, which is usually itself a process, is applied to a process by running both together in parallel. A particular run is considered to be successful if the test reaches a certain designated successful state and the process guarantees the test if every run is successful. The test and the process under observation interact by communicating with each other or synchronising. In most process algebras synchronisation is modelled as the simultaneous occurrence of complementary actions although there is a variety of definitions of complementation. But regardless of the precise definition the actions which comprise the synchronisations are considered to be instantaneous and indivisible. This of course is an idealisation and abstraction from reality but it has proved to be most useful as it has lead to a range of elegant mathematical theories of processes. Here we relax this restriction. Now we will assume that the synchronisations take a non-zero but indefinite amount of time. This is still an abstraction from reality as we are not saying exactly what form the interaction takes; only that it takes time. For example we could have in mind the rendez-vous mechanism of ADA or the existence of some non-trivial communication medium connecting the tester and the process. Under these assumptions we can see how the process performing a and b in parallel can be differentiated from its sequential counterpart which performs the actions in either order. Consider the test which requests a synchronisation via the action a and then will succeed only if it can successfully initiate a second synchronisation via the action b before the first synchronisation has terminated. The first process will always pass this test whereas the second will always fail.

Let us now discuss how this intuitive idea of non-instantaneous actions can be for-

We now give a more detailed account of the contents of each section of the paper. In the next section we define the language used in the paper and give the st-operational semantics. The language is a very

straightforward as $p\rho$ can diverge although p and $\rho(a)$, for every a

The existing work which appears closest to our results is reported in [Vog91a] and [JM92]. In [Vog91a] the author deals with *safe Petri Nets* and *failure equivalence*. A restricted form of refinement theorem is proved for a generalisation of *failure equivalence* based on *interval semi-words* and there is a characterisation with respect to the standard failures equivalence. As the author points out in a separate paper, [Vog91b], *interval semi-words* are more or less equivalent to st-sequences; it therefore follows (from the results of section three) that this equivalence is closely related to $\approx_c$. However the notion of action refinement used is more restrictive than what we allow. In particular when a refinement ρ is applied to a process there can be no interaction between occurrences of $\rho(a)$ and $\rho(b)$ in the refined process. In [JM92] this work is extended to a more general class of Petri Nets but the restrictions on the type of action refinements remain.

2 The Language

In this section we describe the language, taken from [AH92], its st-operational semantics and the testing preorder.

The language is parameterised on a set of actions Act which is ranged over by $a, b, \ldots$. We assume that there is a possibly partial complementation function defined over Act; we write the complement of a, if it exists, as $\overline{a}$ and we asssume that $\overline{\overline{a}}$ is a. We also assume a special action symbol, τ, different from all symbols in Act and a set of recursion variables X ranged over by x. Then the syntax of

those for sub-actions. Let L be an infinite set of labels, ranged over by l, $LSAct$ denote the set of labelled sub-actions $\{ s(a_l), f(a_l) \mid a \in Act, l \in L \}$ and $LAct$ denote the union of all the external actions, $LSAct \cup Act$. For any set S we use S_τ to denote $S \cup \{\tau\}$. So for example $LSAct_\tau$, Act_τ, denote the sets $LSAct \cup \{\tau\}$, $Act \cup \{\tau\}$ respectively. We let μ range over the set of all possible actions $LAct_\tau$, α over the set of external actions $LAct$, a over the set of complete actions Act and finally e over $LSAct$, the set of (labelled) sub-actions. The execution of the sub-actions will often lead to states of processes where actions have started and not yet terminated and therefore we have to enrich the language in order to define such states. We call the more general terms *configurations* and they are defined by

$$c \quad ::= \quad p \mid a_l \mid c \mid c \mid c; p \mid c \backslash a$$

where we assume that in $c \backslash a$ c contains no occurrences of any $a_l, \overline{a}_l$ and more importantly that every occurrence of a labelled action a_l is unique. An occurrence of a_l is meant to denote that there is an a action active and since we use the labels to distinguish different occurrences it is important that there is no duplication of labels. So for example the configuration $a_l; p \mid b_l; q \mid a_k; r$ describes a process which has three subprocesses, two of which are performing an a action and one a b action. We let $\mathcal{C}$ denote the set of configurations, ranged over by c, and for any $c \in \mathcal{C}$ $L(c)$ denotes $\{ a_k \mid a_k$ occurs in $c \}$.

Definition 2.1 Let $\surd$ be the least relation over configurations which satisfies

1. $nil\surd$
2. $p\surd$, $q\surd$ implies $p + q\surd$
3. $p\surd$, $c\surd$ implies $p; c\surd$
4. $c\surd$, $c'\surd$ implies $c \mid c'\surd$
5. $c\surd$ implies $c \backslash a\surd$
6. $t[rec\, x.\, t/t]\surd$ implies $rec\, x.\, t\surd$

□

Since BP is contained in $\mathcal{C}$ this also gives a definition of $\surd$ for the set of processes. Because of the way in which recursion is handled we also need, in the definition of testing, a

$(O1)$	$\mu \xrightarrow{\mu} nil$	
	$a \xrightarrow{s(a_l)} a_l$	for every label l
	$a_l \xrightarrow{f(a_l)} nil$	
$(O2)$	$p \xrightarrow{\mu} c$	implies $p + q \xrightarrow{\mu} c$
$(O\ )$	$c_1 \xrightarrow{\mu} c_1'$	implies $c_1 \mid c_2 \xrightarrow{\mu} c_1' \mid c_2$ provided $c_1' \mid c_2$ is in $\mathcal{C}$
$(O4)$	$c \xrightarrow{\mu} c'$	implies $c\backslash a \xrightarrow{\mu} c'\backslash a$ provided a admits μ
$(O5)$	$c \xrightarrow{\mu} c'$	$c;p \xrightarrow{\mu} c';p$
	$c\surd,\ p \xrightarrow{\mu} c'$	implies $c;p \xrightarrow{\mu} c'$
$(O6)$	$c_1 \xrightarrow{a} c_1',\ c_2 \xrightarrow{\overline{a}} c_2'$	implies $c_1 \mid c_2 \xrightarrow{\tau} c_1' \mid c_2'$
$(O8)$	$t[rec\,x.\ t/x] \xrightarrow{\mu} q$	implies $rec\,x.\ t \xrightarrow{\mu} q$
$(O9)$	$\Omega \xrightarrow{\tau} \Omega$	

Figure 1: Operational semantics

5. $c \downarrow$ implies $c\backslash a \downarrow$
6. $t[rec\,x.\ t/t] \downarrow$ implies $rec\,x.\ t \downarrow$

□

We often use $\uparrow$ for the converse to $\downarrow$. So for example $\Omega \uparrow$ and $rec\,x.\ a + x \uparrow$. The next state relations $\xrightarrow{\mu}$, for each $\mu \in LAct$

The last three rules are concerned with the derivation of internal moves and $(O6)$ is the most important. It says that an internal move may occur because of a communication between two subprocesses. The remaining rules are straightforward; Ω can only perform internal moves and the moves of a recursive definition are determined by its body.

One can check that if $c \stackrel{\mu}{\longrightarrow} c'$ and $c \in \mathcal{C}$ then c' is also in $\mathcal{C}$. One can also show that the actual identity of the labels generated in the derivations are relatively unimportant. Specifically if $c \stackrel{s(a_l)}{\longrightarrow} c'$ then for almost all labels k $c \stackrel{s(a_k)}{\longrightarrow} c'[k/l]$; one can use any k which does not occur already in c, which is a finite set of labels. As stated previously it is unnecessary to define the operational semantics of complete actions as they can be derived. This can be seen from the second part of the following lemma.

Lemma 2.3 *for every configuration c*

1. $c \stackrel{s(a_l)}{\longrightarrow} c' \stackrel{f(a_l)}{\longrightarrow} d$ *implies* $c \stackrel{a}{\longrightarrow} d$
2. $c \stackrel{a}{\longrightarrow} d$ *implies there exists a configuration c' such that for some l* $c \stackrel{s(a_l)}{\longrightarrow} c' \stackrel{f(a_l)}{\longrightarrow} d$
3. $c \stackrel{s(a_i)}{\longrightarrow} \stackrel{s(\overline{a_j})}{\longrightarrow} \stackrel{f(a_i)}{\longrightarrow} \stackrel{f(\overline{a_j})}{\longrightarrow} d$ *implies* $c \stackrel{\tau}{\longrightarrow} d$.

□

Of slightly more interest is the fact that many pairs of moves can be permuted.

Definition 2.4 Two elements, μ, μ' of $LSAct_\tau$ *weakly commutes* if for every pair of configurations c, c' $\exists d.c \stackrel{\mu}{\longrightarrow} d, d \stackrel{\mu'}{\longrightarrow} c'$ implies $\exists d.c \stackrel{\mu'}{\longrightarrow} d, d \stackrel{\mu}{\longrightarrow} c'$. □

The set of weakly commuting moves can be characterised as follows:

Proposition 2.5 *A pair of moves $< \mu, \mu' >$ is weakly commuting if and only if they are of one of the forms the forms*

1. $< s(a_i), \mu >$ *where μ is different than* $f(a_i)$
2. $< \mu, f(a_i) >$ *where μ is different than* $s(a_i)$.

Proof: For pairs not of this form it is easy to think of counterexamples. If the pair $< \mu, \mu' >$ is of this form and $c \stackrel{\mu}{\longrightarrow} d \stackrel{\mu'}{\longrightarrow} c'$ then one can show by induction on the derivation of $c \stackrel{\mu}{\longrightarrow} d$ that there exists a d' such that $c \stackrel{\mu'}{\longrightarrow} d' \stackrel{\mu}{\longrightarrow} c'$ The detailed case analysis depends crucially on the allowed structure of configurations. □

A stronger notion of commuting may be obtained by replacing the implication in the above definition with "if and only if". Let us say that such a pair is *strongly commuting"*. There are far fewer strongly commuting pairs; the only ones are of the form $< s(a_l), s(b_k) >$ and $< f(a_l), f(b_k) >$.

The reason for developing this st-operational semantics is to formalise the concurrent testing discussed in the introduction. However in order to be able to describe the appropriate tests we need a language which is strictly more expressive than BP_{Act}. This is because in this form of testing we need to be able to test the ability of processes to initiate new synchronisations before other previously initiated synchronisations have

terminated. Such tests are not possible in the basic language BP_{Act}. So we include in our set of tests processes which can perform as fully-fledged actions the subactions of the language BP_{Act}. Specifically we use as the set of actions

$$\Delta = \{\, s(a_l), \overline{s(a_l)}, f(a_l), \overline{f(a_l)} \mid a \in Act \,\} \;\cup\; Act.$$

There is another problem which can not be resolved by mimicing the framework of testing developed in [Hen88]. Here we have two forms of termination, in *nil* and δ and it is difficult to see how they can be differentiated by purely computation means. Accordingly we introduce into the testing language the ability to recognise proper termination; this takes the form of a special action *term* which the test can execute only when the process under observation is properly terminated. So we use as the set of tests, *Tests*, closed terms in the language BP_Φ where Φ is the set of actions

$$\Delta \cup \{term, \omega\}.$$

Here the action ω will be used to report the successful completion of an experiment. Of course there are many tests in this language which will not be used but, at least here, it is not of great importance to characterise the collection of meaningful tests.

We now define formally how tests and processes, or more generally configurations, interact. An *experimental state* takes the form $e \parallel c$ where e is a test and c a configuration. An experiment proceeds by moving from state to state and this is defined using a transition relation of the form $e \parallel c \;\mapsto\; e' \parallel c'$.

Definition 2.6 Let $\mapsto$ be the least relation between experimental states which satisfies

Example 2.8 The processes $a \mid b$ and $a; b + b; a$ are also incomparable: $a \mid b$ guarantees the test $s($

1. $c \stackrel{\varepsilon}{\Longrightarrow} c$
2. $c \stackrel{s}{\Longrightarrow} d, \quad d \stackrel{\mu}{\longrightarrow} d'$ implies $c \stackrel{s.\mu}{\Longrightarrow} d'$ for every μ in $LAct_\tau$

. $c \stackrel{s}{\Longrightarrow} d, \quad d \stackrel{\tau}{\longrightarrow} d'$ implies $c \stackrel{s}{\Longrightarrow} d'$.

Note the subtlety in this definition; $c \stackrel{\tau^n}{\Longrightarrow} d$ means that c can move to d by performing at least n internal moves. For any c let $S(c) = \{\, a \in Act \mid c \stackrel{a}{\Longrightarrow} \,\}$; $S(c)$ only contains complete actions which are by definition unlabelled.
We say c is *stable* if $c \downarrow$ and $c \stackrel{\tau}{\longrightarrow} c'$ for no c' and it is *live* if $c\surd$ is not true. i.e. if it has not properly terminated. We also say it *converges*, written $c \Downarrow$, if intuitively it can not diverge, i.e. every sequence of the form

$$c = c_0 \stackrel{\tau}{\longrightarrow} c_1 \stackrel{\tau}{\longrightarrow} c_2 \ldots$$

is finite and for every configuration c_k in such a sequence $c_k \downarrow$. The set of *acceptance sets of p after s*, for $s \in LAct^*$, is defined by

$$\mathcal{A}(p,s) = \{\, S(c) \mid p \stackrel{s}{\Longrightarrow} c,\ c \ \textit{stable} \text{ and } \ \textit{live} \,\}.$$

In this definition the requirement that c be live is crucial. Acceptance sets are compared as in [Hen88], using a slight variation on subset inclusi(as)5 A.18604TfΩset=

from $e \parallel p$ to e_{nk}

Proof: (Outline) The proof relies on the fact that if $s \in LAct^*$ $p \stackrel{s}{\Longrightarrow} p_1$ then the starts and finishes in s are in the proper order and $p \stackrel{s}{\Longrightarrow} p_1$ if and only if $p \stackrel{s'}{\Longrightarrow} p_1'$ where $s' \in LAct^*$ is obtained from s by some systematic renaming to the duplicate labels in the resulting sequence to ensure uniqueness and p_1'

It follows more or less immediately that

Proposition 3.13 *For all processes p, q $p \ll_{st} q$ if and only if $p \ll''_{st} q$.* □

One could go even further and show how these equivalence classes can be interpreted as

Definition 4.1 For each action refinement ρ and term t let $sub(\rho, t)$ be the term defined by

1. $sub(\rho, a) = \rho(a)$
2. $sub(\rho, (t\backslash a)) = sub(\rho[a \mapsto a'], t)\backslash a'$ where a' is different than all action names in $FA(\rho(a))$ for each a occurring free in t.
3. $sub(\rho, op(\ldots, t, \ldots)) = \quad op(\ldots, sub(\rho, t), \ldots)$
4. $sub(\rho, x) = x$
5. $sub(\rho, rec\, x.\ t) = rec\, x.\ sub(\rho, t)$.

□

For each process p and action refinement ρ $sub(\rho, p)$ is also a process and we take the behaviour of $p\rho$ to be that of $sub(\rho, p)$. Via this reduction we may view $p\rho$ as a process in BP and in what follows we will take this for granted.

We wish to investigate under circumstances the new semantic preorder is preserved by action refinement. Specifically let $\rho \sqsubseteq^c_{\approx_c} \sigma$ if $\rho(a) \sqsubseteq^c_{\approx_c} \sigma(a)$ for every a. Then we wish to know under what circumstances

$$p \sqsubseteq^c_{\approx_c} q \text{ and } \rho \sqsubseteq^c_{\approx_c} \sigma \text{ imply } p\rho \sqsubseteq^c_{\approx_c} q\sigma,$$

or speaking more strictly $p \sqsubseteq^c_{\approx_c} q$ and $\rho \sqsubseteq^c_{\approx_c} \sigma$ imply $sub(\rho, p) \sqsubseteq^c_{\approx_c} sub(\sigma, q)$.

In [AH91a] a similar problem was posed for bisimulation equivalence and it was shown to be true for *standard* refinements.

Definition 4.2 An action refinement ρ is *standard* if it satisfies

1. for each a $\rho(a) \mid \rho(\overline{a}) \stackrel{\varepsilon}{\Longrightarrow} r$ for some r such that $r\surd$.
2. for each a not $\rho(a)\surd$.

□

The first condition says that after the refinement the resulting process should be able to mimic the complete synchronisation between a and $\overline{a}$ and the second says that an action can not be refined to a process which is properly terminated. These conditions are also necessary if we wish $\sqsubseteq_{\approx_c}$ to be preserved by refinements.

Example 4.3 Let p be $(a + d) \mid \overline{d}; b$ and q be $p + \tau; b$. Then $p \approx^c_c q$ but if $\rho(d) = c$, $\rho(\overline{d}) = e$ then $p\rho \not\approx_c q\rho$; the former guarantees the test $a; \omega$. Note that this refinement does not satisfy the first requirement of *standard*. □

Example 4.4 Let p, q be $((c + a; \overline{d}) \mid d; b)\backslash d$, $c + a; b$ respectively. Then $p \approx^c_c q$ but if $\rho(a) = nil$, a refinement which does not satisfy the second condition, then $p\rho \not\approx_c q\rho$. □

However more restrictions are neccessary as can be seen from the next example.

Example 4.5 Let p be $a \mid (b;c) + a + b$, q be $a + b$ and ρ a standard refinement such that $\rho(a) = d$ and $\rho(b) = \overline{d}$. Then $p \sqsubseteq_{\approx_c} q$ but p guarantees the test $c\omega$ whereas q fails it. □

Example 4.6 An standard action refinemant is called *stable* if

1. $\rho(a) \not\xrightarrow{\tau}$for every action a
2. if $\rho(a) \mid \rho(b) \xrightarrow{\tau}$ implies $a = \overline{b}$

□

The example just discussed violates the second condition but we conjecture that the first is unnecessary.

Theorem 4.7 *(The Refinement Theorem) For every pair of stable refinements* ρ, σ, $p \sqsubseteq^c_{\approx_c} q$ *and* $\rho \sqsubseteq^c_{\approx_c} \sigma$ *imply* $p\rho \sqsubseteq^c_{\approx_c} q\sigma$.

Before beginning the proof of this theorem we should point out that it depends on the syntax of the language used. In particular it does not hold for the language *EPL*, [Hen88], with respect to which much of the previous work on testing has been developed. The following example has been pointed out by L. Jategoankar, [Jat92]. The main difference between *EPL* and *CCS* is that τ is replaced by a binary "internal choice operator" $\oplus$ with the operational semantics determined by the rules

$$p \oplus q \xrightarrow{\iota} p \text{ and } p \oplus q \xrightarrow{\iota} q.$$

The relation $\xrightarrow{\iota}$ is much the same as $\xrightarrow{\tau}$ except that it does not decide the choice in $p + q$. Specifically it satisfies

$$p \xrightarrow{\iota} p' \text{ implies } p + q \xrightarrow{\iota} p' + q$$

and the obvious symmetric couterpart.
Using this different operational semantics on can check that $a \mid b + a;b + b;a \sqsubseteq_{\approx_c} a \mid b$ but if we apply the action refinement ρ which is the identity except that it maps b to $\overline{a}$ then they can be distingushed by the test $a;\omega$.

The remainder of this section is devoted to proving this theorem. Most of intermediate results only depend on refinements being *standard*; the extra conditions are needed is one place, the proof of Theorem 4.22. So we only assume that all refinements are standard and we will explicitly mention stability when it is required.

We use the characterisation theorem of the previous section and therefore much of the proof is concerned with $\ll_{st}$ instead of $\sqsubseteq_{\approx_c}$. The proof depends on the ability to break down a sequence of moves from $p\rho$ into the contributions made by p and each $\rho(a)$ and conversely the ability to recombine appropriate sequences of moves from p and each $\rho(a$rerned2TdΩ(with)a.-999. 49fΩ21pnsTdΩ())T()]TJl990TdΩ[(i.7(recom)1000.26(bine)-120sed.)]T(in)-1

Definition 4.8 An *(extended) action refinement* is a mapping

$$\rho\colon Act \cup \{\, a_i \mid a \in Act \,\} \longmapsto C$$

such that $\rho(a) \in BP$ for all $a \in Act$. □

We say that a pair of such refinements are *compatible* if they agree on all the unlabelled actions, i. e. for all $a \in Act$, $\rho(a) = \rho'(a)$. A pair (c, ρ) is *compatible* if $sub(\rho, c)$, defined in the same way as above for terms, is a configuration. This simply means that the labels used in $\rho(a_i)$, $\rho(b_j)$ where a_i, b_j occur in c must be distinct. We will assume that whenever we write $c\rho$ in the pair is actually compatible. We will also say the pair is *proper* if it is compatible and $\rho(a_i)\surd$ if and only if $a_i \notin L(c)$. We will identify an action refinement ρ with its automatic extension, defined by letting $\rho(a_i) = nil$ for each a_i. For extended action refinements obtained in this way every pair (p, ρ) is proper and we tend to refer to "extended action refinements" simply as "action refinements".

Because substitution in general changes the names of restricted channels it is convenient to work modulo $=_\alpha$ or even a slightly more general relation which includes $=_\alpha$. Let $\equiv$ denote strong bisimulation defined over configurations using labelled sub-actions. Specifically it is the largest equivalence relation over C such if $c \equiv d$ then

1. $c\surd$ implies $c\surd$
2. for all $\mu \in LAct_\tau$ $c \xrightarrow{\mu} c'$ implies there exists a d' such that $d \xrightarrow{\mu} d'$ where $c' \equiv d'$.

This is quite a strong relation as even the

Proposition 4.12 *(The Whither theorem) Let c be a configuration and ρ a refinement. Then*

- *starting moves*

  1. $c \xrightarrow{s(a_i)} c'$, $\rho(a) \stackrel{\mu}{\Longrightarrow} x$ *implies* $c\rho \stackrel{\mu}{\Longrightarrow} c'\rho[a_i \mapsto x]$
  2. *if* $a_i \neq b_j$ *then* $c \xrightarrow{s(a_i)} \xrightarrow{s(b_j)} c'$, $\rho(a_i) \stackrel{a'}{\Longrightarrow} x$, $\rho(b_j) \stackrel{\overline{a'}}{\Longrightarrow} y$ *implies* $c\rho \stackrel{\tau}{\Longrightarrow} c'\rho[a_i \mapsto x, b_j \mapsto y]$
  3. *if* b_j *occurs in* c *then* $c \xrightarrow{s(a_i)} c'$, $\rho(a) \stackrel{a'}{\Longrightarrow} x$, $\rho(b_j) \stackrel{\overline{a'}}{\Longrightarrow} y$ *implies* $c\rho \stackrel{\tau}{\Longrightarrow} c'\rho[a_i \mapsto x, b_j \mapsto y]$

- *continuing moves*

  4. *if* a_j *occurs in* c *then* $\rho(a_j) \stackrel{\mu}{\Longrightarrow} x$ *implies* $c\rho \stackrel{\mu}{\Longrightarrow} c\rho[a_i \mapsto x]$
  5. *If* a_i, b_j *occur in* c *and* $a_i \neq b_j$ *then* $\rho(a_i) \stackrel{a'}{\Longrightarrow} x$, $\rho(b_i) \stackrel{\overline{a'}}{\Longrightarrow} y$ *imply* $c\rho \stackrel{\tau}{\Longrightarrow} c\rho[a_i \mapsto x, b_j \mapsto y]$

- *only c moving*

  6. $c \xrightarrow{\tau} c'$ *implies* $c\rho \stackrel{\tau}{\Longrightarrow} c'\rho$.

Proof: The first two statements are proved directly by induction on the length of the proof of $c \xrightarrow{s(a_i)} c'$; however the first is needed to prove the second. The fourth statement is proved by structural induction on c and note that in this case c can not be of the form $rec\, x.\ t$. Statements ( ) and (5) are also proved by structural induction on c and (1) and (4) are needed in the proof of ( ) while only (4) is used in that of (5). The final statement is proved by induction on the length of the proof of the derivation $c \xrightarrow{\tau} c'$. However in this last case an

Since $\ll_{st}$ is defined in terms of sequences the Whence and Whither theorems must be generalised to sequences. This is not particularly easy but it helps somewhat if we confine our interest to st-sequences. The idea is to define a predicate which relates tuples of sequences of contributions from the action refinement ρ and the underlying process p to the sequence produced by the refined process $p\rho$. However in the production of a sequence of moves from $p\rho$

□

One can prove various properties of $\mathcal{M}_T$ by induction on its definition. For example if $< u, \mathbf{r}, s >\in \mathcal{M}_T$ then a_i is in the domain of $\mathbf{r}$ if and only if $s(a_i)$ appears in u. Also if $< u, \mathbf{r}, s >\in \mathcal{M}_T$ then $(a, s') \in T$ if and only if there is some $f(a_i)$ in u such that $\mathbf{r}(a_i) = s'$. The main property of $\mathcal{M}_T$ is that it can record the manner in which derivations are made from processes of the form $p\rho$. The next two theorems state that $p\rho$ can perform a sequence s from $LAct^*$ essentially if and only if $< u, \mathbf{r}, s >\in \mathcal{M}_T$ for some $T \subseteq T(\rho)$ where u and $\mathbf{r}$ record the contributions of p and ρ to the computation of s. The many extra conditions in both theorems are required in order to carry out the proofs using induction.

Theorem 4.15 *(The Decomposition Theorem)*
If (p, ρ) is proper and $p\rho \stackrel{s}{\Longrightarrow} \overline{c}$ then there exists a configuration c, a refinement ρ' and a tuple $< u, \mathbf{r}, s >\in \mathcal{M}_T$ such that

1. $\overline{c} = c\rho'$
2. $p \stackrel{u}{\Longrightarrow} c$
3. $\rho(a) \stackrel{\mathbf{r}(a_i)}{\Longrightarrow} \rho'(a_i)$ *for every* $a_i \in L(c)$
4. $T \subseteq T(\rho)$
5. (c, ρ') *is proper and* ρ *and* ρ' *are compatible.*

Proof: The proof is by induction ontributionsi24TfΩ-47.0TdΩ(on)TjΩ17.299ndeTfΩ 1TjΩ-442.0814.4

of $\mathcal{M}_T$, to ensure that the pair (c, ρ') is proper.

As an example suppose that clause (1) of the Whence theorem applies, $d \xrightarrow{s(a_i)} d''$, $\rho_1(a) \xrightarrow{\mu} x$ and $\overline{c} \equiv d''\rho_1[a$

$\rho(a)\sqrt{\mathbf{r}}(a_i)$. So let ρ

Unfortunately this is not the only result we need about divergence. This is because $c\rho$ may diverge although c and $\rho(a)$, for every a, are well behaved processes which never diverge. As a simple example let c be $rec\,x.\ a;x \mid rec\,x.\ b;x$ with ρ the standard refinement defined by $\rho(a) = a + a'$, $\rho(b) = b + \overline{a'}$. Note that ρ is not stable. There are similar forms of divergence for stable refinements although the behaviour is not quite as complicated. For example let c be $a \mid b$ and ρ the stable refinement [illegible]

Proof:

Proof: We use the alternative characterisation of $\sqsubseteq^c_{\approx_c}$, expressed in terms of st-sequences and stability. In view of Proposition 4.1 it is sufficient to show $p\rho \ll_{st} q\sigma$. This will follow from $p \ll_{st} q$ and $\rho \sqsubseteq^c_{\approx_c} \sigma$.
Now for any st-sequence s suppose $p\rho \Downarrow s$. We have to prove three statements:

1. $q\sigma \Downarrow s$.
   We prove the contrapositive: assuming $q\sigma \Uparrow s$ we prove $p\sigma \Uparrow s$. If $q\sigma \Uparrow s$ then for some prefix s' of s $q\sigma \stackrel{s'}{\Longrightarrow} \overline{c}$ such that $\overline{c} \Uparrow$. To avoid unnecessary complexity we just assume that s' is s.
   To any derivation $q\sigma \stackrel{s}{\Longrightarrow} \overline{c}$ we can apply the Decomposition theorem to obtain $< u, \mathbf{r}, s > \in \mathcal{M}_T$ for some $T \subseteq T(\sigma)$ such that

   (a) $\overline{c} \equiv c'\sigma'$
   (b) $T \subseteq T(\sigma)$
   (c) $q \stackrel{u}{\Longrightarrow} c'$
   (d) for every $a_i \in L(c')$ $\sigma(a) \stackrel{\mathbf{r}(a_i)}{\Longrightarrow} \sigma'(a_i)$.

   In order to prove $p\rho \Uparrow s$ we will try to apply Proposition 4.17 and in most cases we will be successful. Note that becasue $\rho \sqsubseteq_{\approx_c} \sigma$ we can always assume that for such a decomposition $T \prec T(\rho)$. There are three possibilities:

   - There exist some $\overline{c}$ such that $q\sigma \stackrel{s}{\Longrightarrow} \overline{c}$ and the above decomposition gives a c' such that $c' \Uparrow$.
     In this case since $p \sqsubseteq_{\approx_c} q$ it follows that $p \Uparrow u$. From $\rho \sqsubseteq_{\approx_c} \sigma$ it then follows that all the requirements of Proposition 4.17 hold and we can conclude $p\rho \Uparrow$.
   - There exist some $\overline{c}$ such that $q\sigma \stackrel{s}{\Longrightarrow} \overline{c}$ and the above decomposition gives either some $a_i \in L(c')$ such that $\sigma'(a_i) \Uparrow$ or some $a \in S(c')$ such that $\sigma(a) \Uparrow$. Again one can show that all the requirements of Proposition 4.17 are satisfied.
   - Otherwise we can assume that in the decomposition above that $(d, \sigma') \Downarrow$.
     Here we use the predicate $\mathcal{N}_T$. By its decomposition theorem we obtain for each $n \geq 0$, $< u_1, \mathbf{r}_1, n > \in \mathcal{N}_{T_1}$ for some $T_1 \subseteq T(\sigma)$ such that

     $c' \stackrel{u_1}{\Longrightarrow} c_n$
     $\sigma'(a_i) \stackrel{\mathbf{r}_1(a_i)}{\Longrightarrow}$ for each $a_i \in L(c') \cap L(c_n) \cap domain(\mathbf{r}_1)$
     $\sigma'(a) \stackrel{\mathbf{r}_1(a_i)}{\Longrightarrow}$ for each $a_i \in L(c_n) - L(c')$.

     Let $u', T', \mathbf{r}'$ denote $u.u_1, T \cup T_1$ and $\mathbf{r} \cdot \mathbf{r}_1$ respectively.
     Then since $p \sqsubseteq_{\approx_c} q$ and $\rho \sqsubseteq_{\approx_c} \sigma$ it follows that conditions i) and ii) of Proposition 4.17 are true. If for some n condition iii) is also true then, since one can show (using the definitions of $\mathcal{M}_T$ and $\mathcal{N}_T$) that $< u', \mathbf{r}', s > \in \mathcal{M}_T$, we can apply this proposition to obtain $p\rho \Uparrow s$.
     So we may assume $\rho(a) \Downarrow \mathbf{r}'(a_i)$ for each $a_i \in L(c_n)$ and since $(a, s') \in T'$ implies $s' = \mathbf{r}'(a_j)$ for some j it follows that $T' \subseteq T(\rho)$. So we may apply the Composition theorem for $\mathcal{M}_T$ followed by that for $\mathcal{N}_T$ to obtain $p\rho \stackrel{s\tau^n}{\Longrightarrow}$. Since this is true for each n it follows that $p\rho \Uparrow s$.

2. If $q\surd s$ then $p\surd s$.
   We leave this to the reader.

. If $A \in \mathcal{A}(q\sigma, s)$ then $B \subseteq A$ for some $B \in \mathcal{A}(p\rho, s)$.
If $A \in \mathcal{A}(q\sigma, s)$ then $q\sigma \stackrel{s}{\Longrightarrow} \overline{c}$ for some live and stable $\overline{c}$ such that $A = S(\overline{c})$. Applying the decomposition theorem to this derivation we obtain $< u, \mathbf{r}, s > \in \mathcal{M}_T$ for some $T \subseteq T(\sigma)$ and $\overline{c} \equiv d\sigma'$, where (d, σ') is proper, such that $q \stackrel{u}{\Longrightarrow} d$ and $\sigma(a) \stackrel{\mathbf{r}(a_i)}{\Longrightarrow} \sigma'(a_i)$ for every $a_i \in L(d)$. In view of Proposition 4.17 and the fact that $p\rho \Downarrow s$ we may assume that $p \Downarrow u$ and $\rho(a) \Downarrow \mathbf{r}(a_i)$, $\rho(a) \Downarrow$ for each $a \in S(d)$ and $T \subseteq T(\rho)$.
Since $d\sigma'$ is stable it follows that d, $\sigma'(a_i)$, $\sigma'(a)$ are all stable, for each $a_i \in L(d)$ and $a \in S(d)$. These are also live because $d\sigma'$ is live and (d, σ') is proper.
So from $p \stackrel{c}{\underset{\approx}{\sqsubset}}_c q$ and $\rho \underset{\approx}{\sqsubset}_c \sigma$ we can obtain stable and live c and x_{a_i} such that $p \stackrel{u}{\Longrightarrow} c$, $\rho(a) \stackrel{\mathbf{r}(a_i)}{\Longrightarrow} x_{a_i}$ and $S(c) \subseteq S(d)$, $S(x_{a_i}) \subseteq S(\sigma'(a_i))$. Applying the Composition theorem for $\mathcal{M}_T$ we obtain $p\rho \stackrel{s}{\Longrightarrow} c\rho_1$ where ρ_1 denotes $\rho[a_i \mapsto x_{a_i}]$.

We show that $c\rho_1$ is stable and for this we need the fact that the refinements are stable. Suppose $c\rho_1 \stackrel{\tau}{\longrightarrow}$. We use the Whence theorem to derive a contradiction. This theorem gives seven different possible decompositions for this internal move and examining each in turn we see that none of them are possible. The first is not possible since ρ is stable. In the second decomposition the stability of ρ implies that $a = \overline{b}$ and so by Lemma 2. $c \stackrel{\tau}{\longrightarrow}$ which contradicts the fact that c is stable. In the third case since $S(c) \subseteq S(d)$ we can use the third clause of the Whither theorem to show that $d\sigma' \stackrel{\tau}{\longrightarrow}$ which contradicts the stability of $d\sigma'$. A similar arguement rules out the fifth case while the fourth is impossible because each x_{a_i} is stable. The stabilty of c rules out the sixth case and finally the last case is not possible since each x_{a_i} is live.

One can also check that $c\rho_1$ is live and by construction $S(c\rho_1) \subseteq A$. The result now follows since $p\rho \stackrel{s}{\Longrightarrow} c\rho_1$.

□

5 Characterising ST-Testing

The more standard notion of testing, based on complete, non-divisible actions as in [Hen88] may also be applied to our language; we denote the resulting preorder by $\underset{\approx}{\sqsubset}_s$. Because only complete actions are used this equivalence does not distinguish between nondeterminism and concurrency. It is also not preserved by action refinement.

In this section we examine the relationship between $\underset{\approx}{\sqsubset}_c$ and $\underset{\approx}{\sqsubset}_s$. The main result is that $\stackrel{c}{\underset{\approx}{\sqsubset}}_c$ may be characterised by $\stackrel{c}{\underset{\approx}{\sqsubset}}_s$ and action refinement. Specifically $\stackrel{c}{\underset{\approx}{\sqsubset}}_c$ satisfies the properties

1. it is contained in $\stackrel{c}{\underset{\approx}{\sqsubset}}_s$
2. it is preserved by stable action refinements.

We prove that $\stackrel{c}{\underset{\approx}{\sqsubset}}_c$ is the largest relation with these two properties.

Let us first define the testing relation $\sqsubseteq_{\approx_s}$. Although it may be obtained by restricting the tests in section 2 it makes the development clearer if we reiterate the various definitions. For each $\nu \in Act_\tau$ let $\stackrel{\nu}{\longrightarrow}_s$ be the least relation over BP which satisfies all of the requirements $(O1)$ to $(O7)$ in Figure 1 with the single change that in $(O1)$ the second and ed.9805 Ω(of)TjΩ14.4So10TdΩ0if

Corollary 5.4 *For all processes* p, q $p \sqsubseteq_{\approx_c} q$ *implies* $p \sqsubseteq_{\approx_s} q$. □

Let $\sqsubseteq_{\approx_s}^{c}$ be defined in the obvious way from $\sqsubseteq_{\approx_s}$: $p \sqsubseteq_{\approx_s}^{c} q$ if $p \sqsubseteq_{\approx_s} q$ and p *stable* implies q *stable*. It follows trivially that $\sqsubseteq_{\approx_c}^{c} \subseteq \sqsubseteq_{\approx_s}^{c}$. We also know that $\sqsubseteq_{\approx_c}^{c}$ is preserved by action refinement and therefore the main result of this section will follow if we can construct a particular stable refinement σ with the property that for all processes p, q, $p\sigma \ll q\sigma$ implies $p \ll_{st} q$. We define σ as a mapping

$$\sigma: Act \longmapsto BP_{\Delta}$$

where Δ is the set of actions defined in section ; it contains all the elements of Act and all labelled begin and end actions together with their complements. Then σ is defined if

actually occur, clauses (2) and (6). In each case it is straightforward to find the required c' and σ''. □

As an immediate corollary we have

Corollary 5.9 *For every* st*-sequence* s, $p \Uparrow s$ *if and only if* $p\sigma \Uparrow s$.

Proof: First suppose $p \Uparrow s$. Without loss of generality we can assume $p \stackrel{s}{\Longrightarrow} c$ such that $c \Uparrow$. By Corollary 5.6 $p\sigma \stackrel{s}{\Longrightarrow}_s c\sigma_c$ and by the Whither theorem $c \Uparrow$ implies $c\sigma_c \Uparrow$.

Conversely suppose $p\sigma \Uparrow s$, say $p\sigma \stackrel{s}{\Longrightarrow}_s \overline{c}$ such that $\overline{c} \Uparrow$. Applying Lemma 5.7 $\overline{c} \equiv c\sigma'$ for some extension σ' of σ compatible with c and $p \stackrel{s'}{\Longrightarrow} c$ for some $s' \equiv s$. By the previous lemma it follows that $c \Uparrow$, i. e. $p \Uparrow s'$ and so $p \Uparrow s$. □

We also have the following properties of the refinement σ.

Lemma 5.10 *For every extension* σ' *of* σ *compatible with* c

1. $c\surd$ *if and only if* $c\sigma'\surd$
2. c *is stable if and only if* $c\sigma'$ *is stable*
3. *if* c *is stable* $S(c) = S(c\sigma')$.

Proof: Straighforward using Proposition 4.10 and the Whither and Whence theorems. □

Finally we can prove the crucial property of the refinement σ.

Theorem 5.11 $p\sigma \sqsubseteq\!\!\!\!\approx_s q\sigma$ *implies* $p \sqsubseteq\!\!\!\!\approx_c q$.

Proof: Suppose $q \Uparrow s$ where s is an st-sequence. By the previous corollary $q\sigma \Uparrow s$. This in turn implies $p\sigma \Uparrow s$ from which $p \Uparrow s$ follows, again by the corollary.

We leave the reader to check $q\surd s$ implies $p\surd s$ and we check the condition on acceptance sets. Suppose $A \in \mathcal{A}(q, s)$, i. e. $q \stackrel{s}{\Longrightarrow} c$ such that c is live and stable and $A = S(c)$. By Corollary 5.6 $q\sigma \stackrel{s}{\Longrightarrow}_s c\sigma_c$ and by the previous lemma $c\sigma_c$ is also live and stable and $A = S(c\sigma_c)$. So $p\sigma \stackrel{s}{\Longrightarrow}_s \overline{c}$ for some live and stable $\overline{c}$ such that $S(\overline{c}) \subseteq A$. Applying Lemma 5.7 $\overline{c} \equiv c\sigma'$ for some extension σ' compatible with c and $p \stackrel{s'}{\Longrightarrow} c$ for some s' such that $s \equiv s'$. Again by the previous lemma c is live and stable and $S(\overline{c}) = S(c\sigma') = S(c)$. So $S(\overline{c}) \in \mathcal{A}(p, s') = \mathcal{A}(p, s)$. □

As a corollary we have the main result of the section:

Corollary 5.12 *The relation* $\sqsubseteq\!\!\!\!\approx_c^c$ *is the largest preorder contained in* $\sqsubseteq\!\!\!\!\approx_s^c$

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

[MP91] D. Murphy and D. Pitt. Testing, betting and true concurrency. In *Proceedings of Concur 91*, number 527 in Lecture Notes in Computer Science, 1991.

[Sto88] A. Stoughton. *Fully Abstract Models of Programming Languages.* Research Notes in Theoretical Computer Science, Pitman/Wiley, 1988.

[TV87] D. Taubner and W. Vogler. The step failures semantics. In F.J. Brandenburg et. al., editor, *Proceedings of STACS 87*, number 247 in Lecture Notes in Computer Science, pages 48– 59. Springer–Verlag, 1987.

[vG90] R.J. van Glabbeek. The refinement theorem for ST-bisimulation. In *Prooceedings IFIP Working Group, Sea of Galilee*, Lecture Notes in Computer Science. Springer–Verlag, 1990.

[vGV87] R.J. van Glabbeek and F.W. Vaandrager. Petri net models for algebraic theories of concurrency. In J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, editors, *Prooceedings PARLE conference*, number 259 in Lecture Notes in Computer Science, pages 224–242. Springer–Verlag, 1987.

[Vog90] W. Vogler. Bisimulation and action refinement. Technical report, Technische Universität München, 1990.

[Vog91a] W. Vogler. Failure semantics based on interval semiwords is a congruence for refinement. *Distributed Computing*, 4:1 9–162, 1991.

[Vog91b] W. Vogler. Is partial order semantics necessary for action refinement ? Technical report, Technische Universität München, 1991.