INFORMATION AND COMPUTATION 115, 179–247 (1994)

Adding Action Refinement to a Finite Process Algebra*

L. ACETO† AND M. HENNESSY

Computer Science, School of Cognitive and Computing Sciences, University of Sussex, Falmer, Brighton BN1 9QH, England

In this paper we present a Process Algebra for the specification of concurrent, communicating processes which incorporates operators for the refinement of actions by processes, in addition to the usual operators for communication, nondeterminism, internal actions, and restrictions, and study a suitable notion of semantic equivalence for it. We argue that action refinements should not, in some formal sense, interfere with the internal evolution of processes and their application to processes should consider the restriction operator as a "binder." We show that, under the above assumptions, the weak version of the refine equivalence introduced by Aceto and Hennessy ((1993) *Inform. and Comput.* **103**, 204–269) is preserved by action refinements and, moreover, is the largest such equivalence relation contained in weak bisimulation equivalence. We also discuss an example showing that, contrary to what happens in Aceto and Hennessy ((1993) *Inform. and Comput.* **103**, 204–269), refine equivalence and timed equivalence are different notions of equivalence over the language considered in this paper.

1. INTRODUCTION

Action refinement occurs naturally in the development of specifications for processes or systems. At one level of abstraction the specification might look like

$$SPEC \Leftarrow \ldots; input; output; \ldots$$

where *input* and *output*, at this level of abstraction, may be considered as uninterpreted or unanalyzed actions. At some stage during the refinement of the specification, it may be appropriate to describe in more detail how these actions are supposed to occur. These descriptions could be in terms of processes, say, P and Q, so that the new specification may look like

$$NSPEC \Leftarrow \ldots; P; Q; \ldots .$$

* The research reported in this paper was supported by SERC and the Esprit BRA action CEDISYS.

† Current address: Aalborg University Centre, 9220 Aalborg Ø, Denmark.

0890-5401/94 $6.00

In this case, the more detailed specification *NSPEC* is obtained from the more abstract one *SPEC* by *action refinement*, refining the action *input* to the process P and the action *output* to the process Q.

Process algebras have been developed as prototype specification languages for concurrent systems, see [35, 32, 10], but, as pointed out in, e.g., [14, 27], they do not support this mechanism of action refinement. The object of this paper is to develop a reasonable process algebra which incorporates such a refinement operator and to suggest a suitable notion of semantic equivalence for specifications written in this process algebra. This should be viewed as a contribution to the theoretical foundations of process algebras which may eventually allow their use as the basis of formal specification methodologies which support action refinement.

We take as our basic language a cross of finite CCS [35] and ACP [10, 8]. This language is based on a set of actions *Act* and contains, as usual, the binary choice combinator $+$, the restriction operator $\backslash\alpha$, where α is an action, and the binary parallel combinator $|$; $p|q$ means that the processes p and q are running in parallel and they may synchronize using complementary actions. In order to support action refinement, the usual action-prefixing operator from CCS is replaced by sequential composition ";." The introduction of this operator has further implications. As pointed out in [4], in the presece of sequential composition and restriction it is no longer sufficient to have only one notion of terminated process as in CCS. So we also have in the language a constant for the successfully terminated process, *nil*, and one for the completely deadlocked process, δ. The result is a very rich and expressive language which only lacks a facility for recursive definitions for it to be considered a standard process algebra.

The first question we ask is: what action refinements, i.e., substitutions of actions by processes, should be allowed in this rich setting? There are two somewhat opposing constraints. The first has to do with the issue of what action refinements are useful in practice, in the sense that any action refinement which might be used in practice should be allowed by our definition. The second is that the allowed refinements should be restricted so that a reasonable semantic theory can be developed. One constraint that we impose is that actions can *not* be refined into terminated processes. This is unlikely to constrain practical applications and, as we shall see, the presence of such refinements would make the development of an adequate, abstract semantic equivalence which is preserved by action refinements very difficult. The other constraints we impose have to do with complementation of actions and thus with the synchronization potential of processes. Recall that in CCS the set of actions has the structure $Act = \Lambda \cup \bar{\Lambda} \cup \{\tau\}$, where Λ is a basic set of actions, $\bar{\Lambda}$ is the set of their complements, and τ is a distinguished action meant to denote internal and unobservable actions. In view of the nature of τ it is reasonable to say that it cannot be

refined. Once more this is very natural from the point of view of applications, although from a theoretical standpoint we might have allowed the refinement of τ by processes that can only perform internal actions. Finally, we require that if action a is refined to the process p then its complement $\bar{a}$ is refined to some "complement of p," i.e., some process which is capable of communicating indefinitely with p until both are successfully terminated, in the same way that the primitive unrefined process a can communicate to completion with $\bar{a}$. One way of obtaining a complement of p is to replace each action with its complement in p. At the moment it is very difficult to say if this constraint will be restrictive in practice as there is very little experience of refining CCS specifications. However, it will be very convenient in developing our semantic theory.

The second question we address is how action refinements are to be applied. For the language considered in [3] the answer is straightforward because of the absence of restriction: an action refinement is applied to a process by syntactically substituting for each action symbol its corresponding refinement. However, this is no longer adequate in this enriched setting. For example, consider the process p,

$$((\lambda; p' + \alpha; q) \mid \bar{\alpha}; r)\backslash\alpha,$$

where λ does not occur in p', q, and r. If we now refine λ to the process $\alpha; w$ then, intuitively, we do not wish the result of the application of this refinement to p to be the process

$$(((\alpha; w); p' + \alpha; q) \mid \bar{\alpha}; r)\backslash\alpha.$$

In p the action α is a local action which is known only internally to the process. It is a "bound action" and semantically p should be equivalent to

$$((\lambda; p' + \beta; q) \mid \bar{\beta}; r)\backslash\beta,$$

at least assuming that α and β do not appear in p', q, and r. In other words, restricted actions should not be allowed to "capture" actions in the refining process. In defining the application of an action refinement we shall appeal to the standard theory of α-conversion and substitution; see, e.g., [38], where the restriction operator is viewed as a binder. So, for example, the effect of refining λ by $\alpha; w$ in p will be $(((\alpha; w); p' + \beta; q) \mid \bar{\beta}; r)\backslash\beta$, up to α-conversion.

The final problem we address is the development of an adequate notion of semantic equivalence over the language considered in this paper. One property we require of such an equivalence is that it abstract from the internal evolution of processes; i.e., that it interpret τ-actions as being internal or unobservable. A well-established and useful equivalence with this

property is *weak bisimulation equivalence*, $\approx$ [35]. This is defined in terms of an operational semantics for the language which defines next-state relations for each action a; intuitively, $p \stackrel{a}{\Longrightarrow} q$ implies that p may perform the action a, possibly interspersed with internal actions, and be transformed into q. Then the defining characteristic of the equivalence relation $\approx$ is that if $p \approx q$ then every move from p, $p \stackrel{a}{\Longrightarrow} p'$, can be matched by a corresponding move from q, $q \stackrel{a}{\Longrightarrow} q'$, such that the potential for further computation is retained, i.e., $p' \approx q'$. In a setting where there are no internal, invisible moves, one may analogously define an equivalence relation called *strong bisimulation equivalence*, $\sim$, using the next-state relations $\stackrel{a}{\longrightarrow}$ in place of $\stackrel{a}{\Longrightarrow}$; intuitively, $p \stackrel{a}{\longrightarrow} q$ means that p may evolve to q by performing the action a. The transition relations $\stackrel{a}{\longrightarrow}$ are sometimes called the *strong transition relations* and $\stackrel{a}{\Longrightarrow}$ the *weak transition relations*.

However, as is well-known (see, e.g., [14, 21]), $\approx$ is not adequate in the presence of action refinement. For if a semantic equivalence is to be of any use, it should be such that if two equivalent processes are refined the resulting processes should also be equivalent. This property does not hold for $\approx$. In fact, $a \mid b \approx a; b + b; a$, but if a is refined to $a_1; a_2$ the resulting processes are *not* equivalent with respect to $\approx$. (The details may be found in [3].) On the other hand, bisimulation equivalence has many appealing properties and its form is such that there are simple, but powerful proof techniques associated with it. Our main aim in this paper will thus be to define a reasonable "bisimulation-like" equivalence for the language we consider which is a congruence with respect to all the combinators in the algebra, including the action refinement combinator.

In [3], we gave a characterization of the largest congruence contained in strong bisimulation equivalence for a simple language without communication and restriction. The basic idea of such a characterization is straightforward. One immediate consequence of refining an action is that it is no longer atomic. Indeed, a minimal refinement is to refine every action a to a process, $begin(a); finish(a)$, capable of performing two atomic subactions, the *beginning* of action a and the *end* of a. A "bisimulation-like" equivalence based on these subactions is called "(strong) timed equivalence" in [3] and is denoted by $\sim_t$. In [3] we showed that $\sim_t$ gives a behavioural characterization of the largest congruence over the simple language contained in $\sim$. In other words, for two simple processes to be strong bisimulation equivalent under all action refinements it is sufficient for them to be so under the minimal refinement described above.

The corresponding theorem is no longer true for the richer language with communication and restriction. An example was provided by van Glabbeek and Vaandrager [23] in a slightly different setting and its formulation for our language is discussed in Section 6. In some sense, the fact that $\sim_t$ is the largest equivalence contained in $\sim$ which is preserved by action

refinements over the simple language considered in [3] is accidental. The proof given in [3] proceeds by showing that the above property is true of a more subtle, but more natural version of $\sim_t$, called *strong refine equivalence*, $\sim_r$. Subsequently, $\sim_t$ and $\sim_r$ are shown to coincide for the simple language. It is the latter step which breaks down for the richer language. However, we shall show in this paper that the appropriate version of refine equivalence which takes silent moves into account, denoted by $\approx_r$, is preserved by action refinement over the richer language and moreover is the largest such equivalence relation contained in weak bisimulation equivalence.

For $p \approx_r q$ to be true it is necessary that $p \approx_t q$, i.e., p and q must be weak bisimulation equivalent using the (weak) transition relations based on the subactions *begin*(a), *finish*(a), but, in addition, in the bisimulation the beginnings and endings must be properly matched. More specifically, if we match the action $p \stackrel{begin(a)}{\Longrightarrow} p'$ from p with the action $q \stackrel{begin(a)}{\Longrightarrow} q'$ then, when subsequently establishing that $p' \approx_r q'$, we can only match the *finish*(a) associated with $p \stackrel{begin(a)}{\Longrightarrow} p'$ with the *finish*(a) associated with $q \stackrel{begin(a)}{\Longrightarrow} q'$ and not with the end of some other a action which might have subsequently started. Intuitively, this equivalence is more natural than $\approx_r$, as it conforms more readily to our intuition about comparing the ability of processes to perform actions which happen to be nonatomic. However, the formulation of $\approx_r$ is more complicated than that of $\approx_t$, as the history of actions which have started and not yet finished must be taken into account. Technically, $\approx_r$ is defined in terms of a family of bisimulations $\approx_h$, where h records associations between unfinished actions.

The first major result of this paper is that the appropriate version of refine equivalence which takes internal moves into account, $\approx_r$, is indeed preserved by action refinement over the richer language. It is then a simple matter to characterize the largest congruence over the richer language contained in $\approx_r$, $=_r$:

$$p =_r q \text{ iff } p + a \approx_r q + a \text{ for some action } a \text{ not occurring in } p \text{ and } q.$$

Readers familiar with the theory of weak bisimulation equivalence will recognize the need for the new action a; it is necessary because, in general, weak bisimulation equivalence is not preserved by $+$.

In [3], we showed that, for a simple language $\mathbf{P}_\rho$, $\sim_r$ gave a behavioural characterization of the largest congruence over $\mathbf{P}_\rho$ contained in strong bisimulation equivalence. Namely, for each $p, q \in \mathbf{P}_\rho$, we proved that

$$p \sim_r q \text{ iff for all } \mathbf{P}_\rho\text{-contexts } C[\cdot],\ C[p] \sim C[q].$$

In other words, $\sim_r$ is the largest equivalence relation which on the one hand is preserved by all the operators in the language and on the other is

contained in strong bisimulation equivalence. A similar result holds for the weak version of $\sim_r$, $\approx_r$. Of course, for the usual reasons $\approx_r$ is not preserved by the $+$ operator and therefore the result has to be expressed in terms of $=_r$. The second major result then states that, for each p, q in the richer language,

$$p =_r q \text{ iff for every context } C[\cdot],\ C[p] \approx C[q].$$

This indicates that $=_r$ is a natural candidate for a semantic theory of process algebras which support action refinements.

We now give a brief outline of the contents of this paper. In Section 2 we introduce the language which will be studied in this paper and present several semantic equivalences for it based on variations on the notion of bisimulation. In setting up our semantic framework, we shall rely on work presented in [3, 4]. Section 2.2 is entirely devoted to the discussion of a suitable notion of action refinement for the language we consider. There we also introduce our technique for the application of action refinements to processes. The essential idea is to consider the restriction operator as a binding operator and to adapt the notion of substitution presented in [38] to our setting. A natural "weak" version of the refine equivalence introduced in [3], $\approx_r$, is then presented and analyzed in detail in Section 3. In the following section we show that $\approx_r$ and its closure with respect to $+$-contexts, $=_r$, are both preserved by action refinements. In Section 5 we prove the characterization of $=_r$ as the largest congruence contained in $\approx$. Section 6 is devoted to a discussion of an example from [23] showing that $\approx_t$ and $\approx_r$ are different equivalences over the language considered in this paper. We end with a section of concluding remarks, where we briefly compare our results with related ones in the literature.

2. The Language

2.1. *The Basic Language*

Let Λ denote a countable set of basic uninterpreted symbols ranged over by $\alpha, \beta, \gamma, \alpha', \ldots$. The set of *actions* over Λ, $Act(\Lambda)$, is defined to be $\Lambda \cup \bar{\Lambda} \cup \{\tau\}$, where $\bar{\Lambda} =_{\text{def}} \{\bar{\alpha} \mid \alpha \in \Lambda\}$ and τ is a distinguished symbol not in $\Lambda \cup \bar{\Lambda}$. For each $\alpha \in \Lambda$, $\bar{\alpha}$ will be called the *complement* of α. The complementation $\bar{\cdot}$ is extended to the whole of $Act(\Lambda)$ by $\bar{\bar{\alpha}} = \alpha$ and $\bar{\tau} = \tau$. Intuitively, Λ may be thought of as a set of channel names to be associated with communicating processes, in which case $\alpha \in \Lambda \subseteq Act(\Lambda)$ may be viewed as the action of receiving a synchronization signal from the channel α, $\bar{\alpha}$ as the action of sending a synchronization signal to α and τ as an internal or invisible action. We shall use μ to range over $Act(\Lambda)$ and a, b over $\Lambda \cup \bar{\Lambda}$, the set of *observable actions* which we sometimes denote by $V(\Lambda)$.

Given Λ, the set of *processes* over $Act(\Lambda)$, $\mathbf{P}_\Lambda$, is given by the following BNF definition:

$$p ::= nil \mid \delta \mid \mu(\mu \in Act(\Lambda)) \mid p; p \mid p + p \mid p \mid p \mid p\backslash\alpha \ (\alpha \in \Lambda).$$

We shall use $p, q, p', \ldots$ to range over $\mathbf{P}_\Lambda$. The language for processes given above is a mixture of CCS [34, 35] and ACP [9, 10]. The operators *nil*, $+$, $\mid$, and $\backslash\alpha$ are taken from CCS, but the CCS action-prefixing operator is replaced by sequential composition ";." As explained in [4], in the presence of a general sequential composition operator and of restriction it is necessary to have in the language a symbol for both successful termination, for which we use *nil*, and unsuccessful termination, which we represent by δ. We shall usually abbreviate $\mathbf{P}_\Lambda$ to $\mathbf{P}$. We also use Σ to refer to the set of operators used in the definition of $\mathbf{P}$ and a $\mathbf{P}$-context will be a term in the language which contains one "hole" into which a subterm may be slotted, $C[\cdot]$.

The operational semantics for $\mathbf{P}$ is given in terms of a collection of next-state relations $\xrightarrow{\mu} \subseteq \mathbf{P} \times \mathbf{P}$, one for each action μ, and a successful termination predicate $\surd$. These are given in Fig. 1 and are taken directly from [4]. The definition of the transition relations uses the predicate *admits* defined by

$$\alpha \textit{ admits } \mu \text{ iff } \mu \neq \alpha, \bar{\alpha}.$$

There are numerous variations which one could apply to these definitions (see, e.g., [5, 28, 7]), but in this paper we shall follow the approach in [4]. With these definitions we have a particular instance of a labelled transition system. A *labelled transition system with termination* is a quadruple $\langle P, \mathbf{A}, \rightarrow, \surd \rangle$, where

1. P is a set of processes,
2. $\mathbf{A}$ is a set of actions of the form $V \cup \{\tau\}$, where τ is a distinguished action symbol,
3. $\rightarrow \subseteq P \times \mathbf{A} \times P$ is a next-state relation, and
4. $\surd \subseteq P$ is a successful termination predicate.

This is a slight extension of the usual notion of labelled transition system which suits our language. In such an LTS a *strong bisimulation* is a symmetric relation $\mathscr{R} \subseteq P \times P$ which satisfies, for each $\langle p, q \rangle \in \mathscr{R}$ and $\mu \in \mathbf{A}$,

(i) if $p \xrightarrow{\mu} p'$ then there exists q' such that $q \xrightarrow{\mu} q'$ and $\langle p', q' \rangle \in \mathscr{R}$,

(ii) if $p\surd$ then $q\surd$.

$(T1)$	$nil\surd$		
$(T2)$	$p\surd$ and $q\surd$	imply	$p \odot q\surd$ $(\odot \in \{;, +, \mid\})$
$(T3)$	$p\surd$	implies	$p \setminus \alpha\surd$

(ACT)	$\mu \xrightarrow{\mu} nil$		
(SUM)	$p \xrightarrow{\mu} p'$	implies	$p + q \xrightarrow{\mu} p'$
			$q + p \xrightarrow{\mu} p'$
$(SC1)$	$p \xrightarrow{\mu} p'$	implies	$p; q \xrightarrow{\mu} p'; q$
$(SC2)$	$p\surd$ and $q \xrightarrow{\mu} q'$	implies	$p; q \xrightarrow{\mu} q'$
(PAR)	$p \xrightarrow{\mu} p'$	implies	$p \mid q \xrightarrow{\mu} p' \mid q$
		implies	$q \mid p \xrightarrow{\mu} q \mid p'$
(SYN)	$p \xrightarrow{a} p'$ and $q \xrightarrow{\bar{a}} q'$	imply	$p \mid q \xrightarrow{\tau} p' \mid q'$
(RES)	$p \xrightarrow{\mu} p'$ and α *admits* μ	imply	$p \setminus \alpha \xrightarrow{\mu} p' \setminus \alpha$

FIG. 1. Termination predicate and transition relations for **P**.

The second clause of the definition of strong bisimulation over an LTS with termination explicitly requires the matching of the termination potential of two processes, as expressed by the predicate $\surd$. This is needed to capture semantically the difference between successfully terminated and deadlocked processes.

A *weak bisimulation* is defined in essentially the same way by replacing $\xrightarrow{\mu}$ and $\surd$ by their "weak" counterparts, $\overset{\mu}{\Longrightarrow}$ and $\surd\!\!\!\surd$, respectively. Formally,

$$p \overset{\mu}{\Longrightarrow} q \text{ iff } p \xrightarrow{\tau}{}^* p_1 \xrightarrow{\mu} p_2 \xrightarrow{\tau}{}^* q,$$

for some p_1, p_2, and $p\surd\!\!\!\surd$ iff for all p', $p \xrightarrow{\tau}{}^* p' \not\xrightarrow{\tau}$ implies $p'\surd$. The exact definition of weak bisimulation also uses the relation $\overset{\varepsilon}{\Longrightarrow}$, the reflexive and transitive closure of $\xrightarrow{\tau}$, $\xrightarrow{\tau}{}^*$, and the notation $\hat{\mu}$, where $\hat{a}$ is simply a and $\hat{\tau} = \varepsilon$. Then a symmetric relation $\mathscr{R} \subseteq P \times P$ is a weak bisimulation if, for each $\langle p, q \rangle \in \mathscr{R}$ and $\mu \in \mathbf{A}$,

(i) if $p \xrightarrow{\mu} p'$ then there exists q' such that $q \overset{\hat{\mu}}{\Longrightarrow} q'$ and $\langle p', q' \rangle \in \mathscr{R}$,

(ii) if $p\surd\!\!\!\surd$ then $q\surd\!\!\!\surd$.

We use $\sim$, called *strong bisimulation equivalence*, to denote the largest strong bisimulation and $\approx$, called *weak bisimulation equivalence*, to denote the largest weak bisimulation. We shall be primarily interested in these equivalence relations as applied to the LTS $\langle \mathbf{P}, Act(\Lambda), \rightarrow, \surd \rangle$. Bisimulations have been studied in depth and we assume that the reader is familiar

with them. The basic reference is [35], where both strong and weak bisimulations are explained at length. However, they are not applied to our version of LTS with a successful termination predicate $\surd$, although much of the standard theory carries over. Bisimulation theory for languages with various forms of successful termination and deadlock have been studied for many years (see, e.g., [5, 6, 42]), but, as we have already stated, we are adopting the approach in [4], where a bisimulation preorder based on $\approx$ is investigated and characterized equationally for a minor variation of **P**. In the following, when referring to bisimulations or bisimulation equivalence we mean weak bisimulations or $\approx$, respectively. The following proposition can be proven following standard lines.

PROPOSITION 2.1 (Congruence Properties of $\sim$ and $\approx$). *The relation $\sim$ is a Σ-congruence over* **P** *and $\approx$ is preserved by all the operators in Σ, apart from $+$.*

Another variation on the theme of bisimulation, which has been investigated in [29, 3] for simple languages, is obtained by splitting each visible action a into two subactions $S(a)$, the start of a, and $F(a)$, the end or finish of a. To define an operational semantics based on these actions, we need to enlarge the set of terms in order to describe states of processes in which, for example, actions a_1 and a_2 have started but have not yet finished.

The set of *states* $\mathscr{S}$, or more formally $\mathscr{S}_\Lambda$, is the set of terms generated by the following BNF definition, where as usual p ranges over processes,

$$s ::= nil \mid \delta \mid \mu\ (\mu \in Act(\Lambda)) \mid F(a)\ (a \in V(\Lambda)) \mid s; p \mid p + p \mid s \mid s \mid s\backslash\alpha\ (\alpha \in \Lambda)$$

which satisfy the following constraint:

(CR) $s\backslash\alpha \in \mathscr{S}$ implies that $F(\alpha)$ and $F(\bar{\alpha})$ do not occur in s.

The naturality of this constraint will become clear after the definition of the operational semantics for $\mathscr{S}$. Note that **P** is a subset of $\mathscr{S}$; we shall use $s, s', \ldots$ to range over $\mathscr{S}$. Let $Act_s(\Lambda)$, the set of *subactions*, be given by

$$Act_s(\Lambda) =_{\text{def}} \{S(a), F(a) \mid a \in V(\Lambda)\} \cup \{\tau\}.$$

The operational semantics for $\mathscr{S}$ is given in terms of a collection of next-state relations $\xrightarrow{e}_t \subseteq \mathscr{S} \times \mathscr{S}$, where $e \in Act_s(\Lambda)$. These are defined as follows:

- The next-state relation $\xrightarrow{\tau}_t$ over $\mathscr{S}$ is obtained by simply adapting the axiom and relevant rules presented in Fig. 1 to $\mathscr{S}$. (Note that this also involves defining next-state relations $\xrightarrow{a}_t$ for $a \in V(\Lambda)$.)

(1)	$nil\surd$		
(2)	$s\surd$	implies	$s \setminus \alpha\surd$
(3)	$p\surd$ and $q\surd$	imply	$p + q\surd$
(4)	$s\surd$ and $p\surd$	imply	$s;p\surd$
(5)	$s_1\surd$ and $s_2\surd$	imply	$s_1 \mid s_2\surd$

(S1)	$a \xrightarrow{S(a)}_t F(a)$		
	$F(a) \xrightarrow{F(a)}_t nil$		
(S2)	$p \xrightarrow{e}_t p'$	implies	$p + q \xrightarrow{e}_t p'$
			$q + p \xrightarrow{e}_t p'$
(S3)	$s \xrightarrow{e}_t s'$	implies	$s;p \xrightarrow{e}_t s';p$
(S4)	$s\surd$ and $p \xrightarrow{e}_t s'$	imply	$s;p \xrightarrow{e}_t s'$
(S5)	$s_1 \xrightarrow{e}_t s_1'$	implies	$s_1 \mid s_2 \xrightarrow{e}_t s_1' \mid s_2$
			$s_2 \mid s_1 \xrightarrow{e}_t s_2 \mid s_1'$
(S6)	$s \xrightarrow{e}_t s'$ and α *admits* e	imply	$s \setminus \alpha \xrightarrow{e}_t s' \setminus \alpha$

FIG. 2. Termination predicate and observable next-state relations for $\mathscr{S}$.

• For each e in $Act_s(\Lambda)$ other than τ, the next-state relations $\xrightarrow{e}_t$ over $\mathscr{S}$ are given in Fig. 2, where the predicate *admits* is extended so that

$$\alpha \textit{ admits } e \text{ iff } e \neq S(\alpha), F(\alpha), S(\bar{\alpha}), F(\bar{\alpha}).$$

The definition of $\xrightarrow{e}_t$ is very similar to that of $\xrightarrow{a}$, except that we have the new clauses

$$a \xrightarrow{S(a)}_t F(a) \qquad \text{and} \qquad F(a) \xrightarrow{F(a)}_t nil.$$

Intuitively, the process a can start the action a and be transformed into the state $F(a)$. This is a state in which action a is active and may terminate at any time, which corresponds to performing action $F(a)$. Note that the subactions cannot synchronize and therefore it is inaccurate to view **P** with this operational semantics as processes in $\mathbf{P}_{\Lambda_s}$, where Λ_s is some collection of basic subactions.

The termination predicate $\surd$, defined in Fig. 1 on **P**, is extended in the obvious way to the set of states $\mathscr{S}$ (see Fig. 2).

PROPOSITION 2.2. *Let $s \in \mathscr{S}$ and $e \in Act_s(\Lambda)$. Then $s \xrightarrow{e}_t s'$ implies $s' \in \mathscr{S}$.*

We have already remarked that, for each process p, $p \in \mathscr{S}$. Thus, by the above proposition, each state s reachable from a process p is in $\mathscr{S}$ and thus satisfies condition (CR). This justifies our use of condition (CR). We are, in fact, interested in states only as means of defining the operational semantics for processes using subactions and terms built using the grammar for states which do not satisfy (CR) are *not* reachable from processes using the transition relations $\xrightarrow{e}_t$.

We now have a new labelled transition system with termination $\langle \mathscr{S}, Act_s(\Lambda), \rightarrow_t, \surd \rangle$. In this structure we let $\sim_t$ and $\approx_t$ denote the resulting strong and weak bisimulation equivalence, respectively. The subscript t refers to "time" as the equivalences are obtained by assuming that actions take nonzero time. They have been studied for sublanguages of **P** in [29, 3] and similar equivalences have been called "split-equivalences" in [22, 20, 19, 26].

PROPOSITION 2.3. *The relation $\sim_t$ is a congruence with respect to all the combinators in* **P** *and $\approx_t$ is preserved by all the combinators in Σ, apart from $+$.*

It is interesting to note that the definitions of the split-equivalences $\sim_t$ and $\approx_t$ given above do not require the matching of actions in $V(\Lambda)$. The addition of such a requirement to their definition would give rise to *different* equivalences. This will be demonstrated in Section 6 by means of an example.

2.2. *Action Refinements*

An action refinement may be considered to be simply a mapping from $Act(\Lambda)$ to **P** and the effect of applying an action refinement to a process is a new, more detailed, process obtained by substituting for each action the corresponding refining process. In this section we formalize these ideas for the language **P**.

We first put some natural conditions on action refinements, or more prosaically substitutions. We shall use $\rho, \rho', \ldots$ to range over them. Since τ is an internal, unobservable action, it makes no sense to be able to refine it. So, in effect, action refinements are functions from the set of visible actions $V(\Lambda)$ to **P**. One may also argue that if actions are allowed to be refined by successfully terminated processes then all occurrences of the supposedly internal and invisible action τ will have a significant effect on the behaviour of processes. For example, let p and q denote $(a;\tau)+b$ and $a+b$, respectively. One would expect p to be equivalent to q with respect to most "reasonable" notions of equivalence which abstract from internal transitions. However, if an equivalence were to be preserved by refinements of actions by successfully terminated processes then $p \neq q$. For let ρ denote

a refinement such that $\rho(a) = nil$ and $\rho(b) = b$. Then $p\rho$ and $q\rho$ should be $(nil; \tau) + b$ and $nil + b$, respectively, which are not bisimulation equivalent, nor indeed they would be equivalent with respect to most reasonable notions of equivalence.

As a further example, let p and q be $a; \tau; b$ and $a; b$, respectively. These two processes are again considered to be equivalent with respect to most reasonable semantic equivalences. However, under the same refinement we obtain $nil; \tau; b$ and $nil; b$, respectively. Once more, these two processes are not equivalent, at least if we use equivalences which are preserved by all language contexts. For example, $c + (nil; \tau; b)$ and $c + (nil; b)$ would be distinguished by most reasonable semantic equivalences.

A further constraint we impose on refinements is that they should not, in some sense, interfere with the internal evolution of processes. Let us explain this point with an example. Let p and q denote $\alpha | \bar{\alpha} + \tau$ and $\alpha | \bar{\alpha}$, respectively. Then, once more, we would consider these two processes to be semantically equivalent. However, if we apply a refinement such as $\rho(\alpha) = \beta$ and $\rho(\bar{\alpha}) = \beta'$, the resulting processes $\beta | \beta' + \tau$ and $\beta | \beta'$ will not be equivalent. The problem is that ρ has interfered with the communication potential between the complementary actions α and $\bar{\alpha}$. We shall forbid such refinements by demanding that any action refinement ρ satisfy the requirement

(ComPres) For each $a \in V(\Lambda)$ there exists r such that $\rho(a) | \rho(\bar{a}) \stackrel{\tau}{\Rightarrow} r$ and $r\surd$.

This means that actions and their complements must be refined to processes which can communicate indefinitely until they have successfully terminated. This restriction is satisfied by the most common form of refinement in the literature, namely relabelling of actions (see [35] for details). More generally, (ComPres) may be enforced by demanding that, for each action a,

- $\rho(a) \stackrel{\mu}{\longrightarrow} r$ for some process r and action $\mu \in Act(\Lambda)$, and
- $\rho(\bar{a}) = \overline{\rho(a)}$, where, for each process p, $\bar{p}$ is the process obtained from p by substituting each action by its complement.

In this case, an action refinement ρ would be uniquely determined by how it behaved on the set of channel names Λ.

DEFINITION 2.1 (Action Refinements). An *action refinement* is a mapping $\rho: V(\Lambda) \to \mathbf{P}$ with the following properties:

(i) for all a, *not* $\rho(a)\surd$ and

(ii) ρ satisfies (ComPres).

EXAMPLE 2.1. Let $\rho: V(\Lambda) \to \mathbf{P}$ be the substitution which maps each a to τ, i.e., $\rho(a) = \tau$ for all a. Then ρ is an action refinement. In fact, for all a, $\rho(a)$ is not successfully terminated and

$$\rho(a) \mid \rho(\bar{a}) \equiv \tau \mid \tau \stackrel{\tau}{\Longrightarrow} nil \mid nil\surd.$$

Similarly, it is easy to check that CCS-relabellings [34, 35] are action refinements.

The following property derived from axiom (ComPres) will be most useful in our technical analysis of the operational properties of processes of the form $p\rho$. First of all, for each $\sigma = a_1 \cdots a_n \in V(\Lambda)^*$, let $\stackrel{\sigma}{\Longrightarrow}$ denote the transition relation given by

$$p \stackrel{\sigma}{\Longrightarrow} q \text{ iff there exist } p_0, \ldots, p_n \text{ such that } p_0 = p,\ p_n = q \text{ and,}$$

$$\text{for each } i < n,\ p_i \stackrel{a_i}{\Longrightarrow} p_{i+1}.$$

We shall assume that the complementation of actions is homomorphically extended to strings in $V(\Lambda)^* \cup \{\tau\}$.

FACT 2.1. *Let ρ be an action refinement. Then, for each $a \in V(\Lambda)$, there exist $\sigma \in V(\Lambda)^+ \cup \{\tau\}$, $r_1, r_2 \in \mathbf{P}$ such that $\rho(a) \stackrel{\sigma}{\Longrightarrow} r_1$, $\rho(\bar{a}) \stackrel{\sigma}{\Longrightarrow} r_2$, $r_1\surd$ and $r_2\surd$.*

We now turn our attention to the effect of applying an action refinement ρ to a process p. The resulting process we denote by $p\rho$ and, intuitively, it should be the process which results from substituting each occurrence of a by the process $\rho(a)$ in p, for each a. However, because of the presence of restriction, care must be taken in order to preserve the intended purpose of this combinator, namely the scoping of channel names. In this setting, restriction is a "binding operator" and, in order to take this into account, it is appropriate to define $p\rho$ by adapting to our setting a theory of substitution in the presence of binders. In what follows, we shall use the theory of substitution developed in [38].

For each process p let $FC(p)$, the set of *free channel names* in p, be defined by

$$FC(nil) = FC(\delta) = FC(\tau) = \varnothing$$

$$FC(\alpha) = FC(\bar{\alpha}) = \{\alpha\}$$

$$FC(p; q) = FC(p + q) = FC(p \mid q) = FC(p) \cup FC(q)$$

$$FC(p\backslash\alpha) = FC(p) - \{\alpha\}.$$

As in [38], we define the function *new* by

$$new\ \alpha p\rho =_{\text{def}} \{\beta \mid \text{for each } \beta' \in FC(p) - \{\alpha\},\ \beta \notin FC(\rho(\beta')) \cup FC(\rho(\bar{\beta}'))\}.$$

That is, $new\ \alpha p\rho$ returns the set of innocuous channel names, none of which will capture any of the free channel names which appear when ρ is being applied to p.[1] The definition of substitution given below uses a choice function which takes such a set and returns some element. For example, if Λ were well-ordered we could choose the least element in the set. We also use the standard notation for the modification of substitutions: $\rho[a \to p]$ is the substitution which is identical to ρ except that it maps a to p. As a variation, $\rho[\alpha \mapsto \beta]$ will denote the substitution identical to ρ except that α is mapped to β and $\bar{\alpha}$ to $\bar{\beta}$. We shall use ι to denote the identity substitution.

DEFINITION 2.2 (Application of Action Refinements). For each $p \in \mathbf{P}$ and action refinement ρ, $p\rho$ is the process defined by

(i) $a\rho = \rho(a)$, $\tau\rho = \tau$, $nil\rho = nil$, $\delta\rho = \delta$

(ii) $(p \odot q)\rho = p\rho \odot q\rho$ $(\odot \in \{;, \,, +, |\})$

(iii) $(p\backslash\alpha)\rho = (p\rho[\alpha \mapsto \beta])\backslash\beta$ where $\beta = choice(new\ \alpha p\rho)$.

This is exactly the definition given in [38] except that there the binding operator is the λ-abstraction of the λ-calculus and the only operator is application. However, all of the results in [38] apply equally well here and their proofs are more or less identical. For example, suppose that for two action-refinements ρ_1 and ρ_2 we define the new refinement $\rho_2 \circ \rho_1$ by $\rho_2 \circ \rho_1(a) = \rho_1(a)\,\rho_2$. (We shall show that $\rho_2 \circ \rho_1$ so defined is indeed a refinement at the end of this section.) Then, by Theorem 3.2 of [38] (p. 321), we have

LEMMA 2.1 (Substitution Lemma). $(p\rho_1)\,\rho_2 = p(\rho_2 \circ \rho_1)$.

An example of the application of an action refinement to a process is now in order.

EXAMPLE 2.2. Let us assume, for the purpose of this example, that $\Lambda = \{\alpha_i \mid i \in \mathbf{N}\}$ and that, for all $i, j \in \mathbf{N}$, $\alpha_i < \alpha_j$ iff $i < j$. We assume, moreover, that for each subset X of Λ, $choice(X)$ is the least element of X with respect to $<$. Consider the process $p = ((\alpha_1; \alpha_2 + \alpha_3) \mid \bar{\alpha}_1)\backslash\alpha_1$ and the action refinement $\rho = \iota[\alpha_3 \mapsto \alpha_1]$. Then we have that $new\ \alpha_1((\alpha_1; \alpha_2 + \alpha_3) \mid \bar{\alpha}_1)\,\rho = \{\alpha_i \mid i \geq 3\}$ and $choice(\{\alpha_i \mid i \geq 3\}) = \alpha_3$. Thus

$$p\rho = (((\alpha_1; \alpha_2 + \alpha_3) \mid \bar{\alpha}_1)\,\rho[\alpha_1 \mapsto \alpha_3])\backslash\alpha_3 = ((\alpha_3; \alpha_2 + \alpha_1) \mid \bar{\alpha}_3)\backslash\alpha_3.$$

[1] Note that the definition of *new* takes into account the fact that in CCS the restriction operator α binds both α and its complement $\bar{\alpha}$.

In general, as made clear by the above example, the application of an action refinement to a process changes the restricted channel names and, in such a setting, it is more appropriate to replace syntactic identity with the so-called "α-conversion" (or α-congruence), $=_\alpha$. This is defined to be the last Σ-congruence over $\mathbf{P}$ which satisfies

(α) $\beta \notin FC(p)$ and $p\iota[\alpha \mapsto \beta] =_\alpha q$ imply $p\backslash\alpha =_\alpha q\backslash\beta$.

From Corollary 3.10 of [38, pp. 322–323] we obtain a useful syntactic characterization of $=_\alpha$.

PROPOSITION 2.4 (Syntactic Characterization of $=_\alpha$). *If $p =_\alpha q$ then one of the following conditions holds*:

(1) *p and q have the form $op(p_1, ..., p_k)$ and $op(q_1, ..., q_k)$, respectively, for some $op \in \Sigma$ of arity k and $p_i =_\alpha q_i$, for all i, or*

(2) *p and q have the form $p'\backslash\alpha$ and $q'\backslash\beta$, respectively, where*

$$\beta \notin FC(p') \text{ and } p'\iota[\alpha \mapsto \beta] =_\alpha q'.$$

Other important properties of substitution and α-congruence also follow more or less directly from [38]:

LEMMA 2.2 (Properties of Substitution and $=_\alpha$).

(a) *For each $p \in \mathbf{P}$, $p =_\alpha p\iota$.*

(b) *For each $p, q \in \mathbf{P}$, if $p =_\alpha q$ then, for every action refinement ρ, $p\rho = q\rho$, where $=$ denotes syntactic identity.*

(c) *For each $p \in \mathbf{P}$ and action refinements ρ, ρ', if $\rho(\alpha) = \rho'(\alpha)$ and $\rho(\bar{\alpha}) = \rho'(\bar{\alpha})$ for each $\alpha \in FC(p)$ then $p\rho = p\rho'$.*

(d) *For each $p \in \mathbf{P}$ and action refinement ρ, if $\rho(\alpha) = \alpha$, $\rho(\bar{\alpha}) = \bar{\alpha}$, and ρ is such that $\alpha \notin FC(\rho(\beta)) \cup FC(\rho(\bar{\beta}))$ for every $\beta \in FC(p)$, then $p\rho\backslash\alpha =_\alpha (p\backslash\alpha)\rho$.*

Proof. Statements (a) and (c) follow from parts (vi) and (v) of Lemma 3.1 of [38], respectively. Statement (b) follows from Theorem 3.5 of [38]. The proof of the final statement is a simple application of statement (c). For let $(p\backslash\alpha)\rho$ be $(p\rho[\alpha \mapsto \beta])\backslash\beta$. If β is α then $(p\rho[\alpha \mapsto \beta])\backslash\beta$ coincides with $p\rho\backslash\alpha$. Otherwise, $\beta \notin FC(p\rho)$ and, therefore, by the syntactic characterization of $=_\alpha$,

$$p\rho\backslash\alpha =_\alpha ((p\rho)\iota[\alpha \mapsto \beta])\backslash\beta.$$

However, by the substitution lemma, $(p\rho)\,\iota[\alpha \mapsto \beta] = p(\iota[\alpha \mapsto \beta] \circ \rho)$ and since, by the proviso of the statement, the substitutions $(\iota[\alpha \mapsto \beta] \circ \rho)$ and $\rho[\alpha \mapsto \beta]$ coincide on $FC(p)$, it follows, by statement (c), that

$$(p\rho)\,\iota[\alpha \mapsto \beta] = p\rho[\alpha \mapsto \beta]. \quad \blacksquare$$

The last statement of the above lemma will be applied in the "whence theorems" presented in Section 4. The following useful result states that the set of free channel names of a process never increases under derivations.

FACT 2.2. *Let* $p \in \mathbf{P}$. *Then*:

(1) $p\surd$ *implies* $FC(p) = \varnothing$;

(2) *for each* $\mu \in Act(\Lambda)$, $q \in \mathbf{P}$, $p \xrightarrow{\mu} q$ *implies that* $FC(q) \subseteq FC(p)$.

We have so far studied several useful syntactic properties of substitution and $=_\alpha$. However, many of the arguments we shall apply will not only be syntactic, but will also involve semantic reasoning. For this reason it will be useful to develop behavioural properties of $=_\alpha$. For our purposes, it will be sufficient to prove that $=_\alpha$ is contained in strong bisimulation, $\sim$. First of all, we establish a lemma about relabellings. A relabelling ϱ is a mapping from Λ to Λ. It is extended to a refinement by letting $\varrho(\bar{\alpha}) = \overline{\varrho(\alpha)}$. For conciseness of notation we shall assume that, for every relabelling ϱ, $\varrho(\tau) =_{\text{def}} \tau$.

LEMMA 2.3. *For each* $p \in \mathbf{P}$, *relabelling* ϱ *and* $\mu \in Act(\Lambda)$, $p \xrightarrow{\mu} p'$ *implies* $p\varrho \xrightarrow{\varrho(\mu)} r$ *for some* r *such that* $p'\varrho =_\alpha r$.

Proof. By induction on the proof of the derivation $p \xrightarrow{\mu} p'$. The only nontrivial case is when $p = q\backslash\alpha \xrightarrow{\mu} q'\backslash\alpha = p'$ because $q \xrightarrow{\mu} q'$ and α *admits* μ. In this case, $p\varrho$ has the form $(q\varrho[\alpha \mapsto \beta])\backslash\beta$ for some β such that $\beta \neq \varrho(\beta')$, $\varrho(\bar{\beta}')$, for each $\beta' \in FC(q) - \{\alpha\}$. This means that β *admits* $\varrho(\mu)$. We may now apply the inductive hypothesis to $q \xrightarrow{\mu} q'$ to obtain that $q\varrho[\alpha \mapsto \beta] \xrightarrow{\varrho(\mu)} r$ for some $r =_\alpha q'\varrho[\alpha \mapsto \beta]$. This is because $\varrho[\alpha \mapsto \beta](\mu) = \varrho(\mu)$, as α *admits* μ. By the operational semantics, we then have that

$$(q\backslash\alpha)\varrho = (q\varrho[\alpha \mapsto \beta])\backslash\beta \xrightarrow{\varrho(\mu)} r\backslash\beta.$$

We are thus left to show that $(q'\backslash\alpha)\varrho =_\alpha r\backslash\beta$. However, by Fact 2.2(2), we have that $\beta \in new\,\alpha q'\varrho$ and the claim follows easily from this observation. $\blacksquare$

The following lemma studies the relationships between the termination predicate $\surd$ and $=_\alpha$.

LEMMA 2.4. *For each* $p, q \in \mathbf{P}$, $p\surd$ *and* $p =_\alpha q$ *imply* $q\surd$.

Proof. By induction on the termination predicate $\surd$. The details are omitted. ∎

PROPOSITION 2.5. *For each* $p, q \in \mathbf{P}$, $p =_\alpha q$ *implies* $p \sim q$.

Proof. It is sufficient to show that the relation of α-congruence, $=_\alpha$, is a strong bisimulation. First of all, let us note that $=_\alpha$ is symmetric by definition. Assume now that $p =_\alpha q$. Then, by the previous lemma, $p\surd$ implies $q\surd$. We are thus left to show that

$$p \xrightarrow{\mu} p' \text{ implies } q \xrightarrow{\mu} q', \text{ for some } q' \text{ such that } p' =_\alpha q'.$$

We shall prove this statement by induction on the relation of α-congruence, $=_\alpha$. The proof proceeds by a case analysis on the structure of p and the syntactic characterization of $=_\alpha$ given in Proposition 2.4 will be most useful in the proof. We briefly examine two of the cases of the inductive proof.

- p has the form $p_1 \mid p_2$. By Proposition 2.4, q must have the form $q_1 \mid q_2$ with $p_i =_\alpha q_i$, $i = 1, 2$. We now examine why $p \xrightarrow{\mu} p'$. There are three possibilities; we analyze only one, when $\mu = \tau$, p' has the form $p_1' \mid p_2'$, $p_1 \xrightarrow{a} p_1'$, and $p_2 \xrightarrow{\bar{a}} p_2'$ for some a. By the inductive hypothesis, we then have that $q_1 \xrightarrow{a} q_1'$ and $q_2 \xrightarrow{\bar{a}} q_2'$ for some q_1' and q_2' such that $p_i' =_\alpha q_i'$, $i = 1, 2$. It follows that $q_1 \mid q_2 \xrightarrow{\tau} q_1' q_2'$ and $p_1' \mid p_2' =_\alpha q_1' \mid q_2'$.

- p has the form $p_1 \backslash \alpha$. By Proposition 2.4 there are two possible forms for q:

  (1) q has the form $q_1 \backslash \alpha$ with $p_1 =_\alpha q_1$, or

  (2) q has the form $q_1 \backslash \beta$ with $\beta \notin FC(p_1)$ and $p_1 \iota[\alpha \mapsto \beta] =_\alpha q_1$.

We only examine case (2). Assume that $p_1 \backslash \alpha \xrightarrow{\mu} p_1' \backslash \alpha$. Then $p_1 \xrightarrow{\mu} p_1'$ and α *admits* μ; i.e., $\mu \neq \alpha, \bar{\alpha}$. Now, $\iota[\alpha \mapsto \beta]$ is a relabelling and thus we may apply Lemma 2.3 to obtain that $p_1 \iota[\alpha \mapsto \beta] \xrightarrow{\mu} r$ for some r such that $r =_\alpha p_1' \iota[\alpha \mapsto \beta]$. By the inductive hypothesis, $q_1 \xrightarrow{\mu} q_1'$ for some $q_1' =_\alpha r =_\alpha p_1' \iota[\alpha \mapsto \beta]$. Since $\beta \notin FC(p_1)$ and α *admits* μ we have that $\mu \neq \beta, \bar{\beta}$. Thus β *admits* μ and, by the operational semantics, $q_1 \backslash \beta \xrightarrow{\mu} q_1' \backslash \beta$. Moreover, by $q_1' =_\alpha p_1' \iota[\alpha \mapsto \beta]$ and $\beta \notin FC(p_1) \supseteq FC(p_1')$, we have that $q_1' \backslash \beta =_\alpha p_1' \backslash \alpha$ by applying rule (α) of the definition of $=_\alpha$. ∎

We may now prove that the composition of refinements defined previously is indeed an action refinement. Let us recall, for the sake of clarity, that the composition of two action refinements, ρ_1 and ρ_2, denoted by $\rho_2 \circ \rho_1$, is the refinement given by $(\rho_2 \circ \rho_1)(a) = \rho_1(a)\,\rho_2$. In order to prove that $\rho_2 \circ \rho_1$ is indeed a refinement it is sufficient to show that

(1) $\rho_2 \circ \rho_1(a)$ is not terminated and

(2) $\rho_2 \circ \rho_1$ satisfies (ComPres).

The following result is an important consequence of the fact that, for each refinement ρ, $\rho(a)$ is not terminated.

FACT 2.3. *For each* $p \in \mathbf{P}$ *and action refinement* ρ, $p\rho\surd$ *iff* $p\surd$.

In view of the above result it is then easy to see that $\rho_2 \circ \rho_1$ satisfies condition (1) above. In order to prove that $\rho_2 \circ \rho_1$ satisfies (ComPres), we shall need the following lemma.

Notation 2.1. For each binary relation $\mathcal{R}$ over $\mathbf{P}$, we shall write $p \xrightarrow{\mu} \mathcal{R}q$ iff there exists p' such that $p \xrightarrow{\mu} p'$ and $p'\mathcal{R}q$. A similar notation will be used with respect to the weak transition relations $\stackrel{\sigma}{\Longrightarrow}$, $\sigma \in Act(\Lambda)^* \cup \{\tau\}$.

LEMMA 2.5. *Let* $p \in \mathbf{P}$ *and* ρ *be an action refinement. Then*:

(1) $p \xrightarrow{a} q$ *and* $\rho(a) \stackrel{\sigma}{\Longrightarrow} x\surd$, $\sigma \in V(\Lambda)^+ \cup \{\tau\}$, *imply* $p\rho \stackrel{\sigma}{\Longrightarrow} \sim q\rho$;

(2) $p \stackrel{\tau}{\Longrightarrow} q$ *implies* $p\rho \stackrel{\tau}{\Longrightarrow} \sim q\rho$.

Proof. Let $p \in \mathbf{P}$ and ρ be an action refinement. We prove the two statements separately.

(1) By induction on the proof of the derivation $p \xrightarrow{a} q$. We proceed by a case analysis on the last rule used in the proof and examine only two possibilities.

• $p = a \xrightarrow{a} nil = q$. Then $p\rho = \rho(a) \stackrel{\sigma}{\Longrightarrow} x \sim nil = q\rho$ because $x\surd$.

• $p = p_1\backslash\alpha \xrightarrow{a} p_1'\backslash\alpha = q$ because $p_1 \xrightarrow{a} p_1'$ and α *admits* a. By the definition of substitution, $(p_1\backslash\alpha)\rho = (p_1\rho[\alpha \mapsto \beta])\backslash\beta$ with $\beta = choice(new\ \alpha p_1\rho)$. This implies that β *admits* σ. By the inductive hypothesis, $p_1 \xrightarrow{a} p_1'$ and $\rho[\alpha \mapsto \beta](a) = \rho(a) \stackrel{\sigma}{\Longrightarrow} x\surd$ imply that

$$p_1\rho[\alpha \mapsto \beta] \stackrel{\sigma}{\Longrightarrow} \sim p_1'\rho[\alpha \mapsto \beta].$$

By the operational semantics and the substitutivity of $\sim$,

$$(p_1\backslash\alpha)\rho = (p_1\rho[\alpha \mapsto \beta])\backslash\beta \stackrel{\sigma}{\Longrightarrow} \sim (p_1'\rho[\alpha \mapsto \beta])\backslash\beta.$$

As $FC(p_1') \subseteq FC(p_1)$, we have that $new\ \alpha p_1\rho \subseteq new\ \alpha p_1'\rho$. We then have that $(p_1'\backslash\alpha)\rho =_\alpha (p_1'\rho[\alpha \mapsto \beta])\backslash\beta$. The result now follows because $=_\alpha \subseteq \sim$ and by the transitivity of $\sim$.

(2) By induction on the length of the derivation $p \stackrel{\tau}{\Longrightarrow} q$.

• Base case: $p \xrightarrow{\tau} q$. The proof proceeds by a subinduction on the length of the proof of the derivation $p \xrightarrow{\tau} q$. We only examine the most interesting case, which relies on axiom (ComPres).

Assume that $p = p_1 \mid p_2 \xrightarrow{\tau} q_1 \mid q_2 = q$ because $p_1 \xrightarrow{a} q_1$ and $p_2 \xrightarrow{\bar{a}} q_2$. As ρ satisfies (ComPres), we have that $\rho(a) \stackrel{\sigma}{\Longrightarrow} x_1\surd$ and $\rho(\bar{a}) \stackrel{\bar{\sigma}}{\Longrightarrow} x_2\surd$ for some $\sigma \in V(\Lambda)^+ \cup \{\tau\}$, x_1 and x_2. By statement (1) we then get that $p_1\rho \stackrel{\sigma}{\Longrightarrow} \sim q_1\rho$ and $p_2\rho \stackrel{\bar{\sigma}}{\Longrightarrow} \sim q_2\rho$. By the operational semantics and the substitutivity of $\sim$,

$$(p_1 \mid p_2)\rho = p_1\rho \mid p_2\rho \stackrel{\tau}{\Longrightarrow} \sim q_1\rho \mid q_2\rho = (q_1 \mid q_2)\rho.$$

- Inductive step, $p \xrightarrow{\tau} p' \stackrel{\tau}{\Longrightarrow} q$, for some p'. Immediate. ∎

Statement (2) of the above lemma may be seen as a formal version of the intuitive idea that action refinements should not interfere with the internal evolution of processes. The fragment of its proof that we have presented highlights the fundamental role played by the axiom (ComPres). We may now show that $\rho_2 \circ \rho_1$ is indeed an action refinement.

FACT 2.4. *Let ρ_2 and ρ_1 be action refinements. Then $\rho_2 \circ \rho_1$ is also an action refinement.*

Proof. We have already seen that, for each a, $(\rho_2 \circ \rho_1)(a)$ is not terminated. We are left to show that $\rho_2 \circ \rho_1$ satisfies (ComPres). Now, for any $a \in V(\Lambda)$,

$$(\rho_2 \circ \rho_1)(a) \mid (\rho_2 \circ \rho_1)(\bar{a}) = (\rho_1(a)\,\rho_2) \mid (\rho_1(\bar{a})\,\rho_2) = (\rho_1(a) \mid \rho_1(\bar{a}))\,\rho_2.$$

As ρ_1 satisfies (ComPres), we have that $\rho_1(a) \mid \rho_1(\bar{a}) \stackrel{\tau}{\Longrightarrow} r\surd$, for some r. By the above lemma, we then have that

$$(\rho_1(a) \mid \rho_1(\bar{a}))\,\rho_2 \stackrel{\tau}{\Longrightarrow} x \sim r\rho_2$$

for some x. Moreover, as $r\surd$, we have that $r\rho_2\surd$. This implies that $x\surd$. Hence $\rho_2 \circ \rho_1$ satisfies (ComPres). ∎

2.3. *Extending the Language*

Let the set of extended processes, $\mathbf{P}_\rho$, be the set of all those processes which are definable by adding action refinements to the basic language $\mathbf{P}$ as extra operators:

$$p ::= nil \mid \delta \mid \mu \mid p;\, p \mid p + p \mid p \mid p \mid p \backslash \alpha \mid p[\rho].$$

It is within a language such as this (of course extended at least with recursive definitions) that the development of process specifications could take place and we are interested in a semantic theory for it. However, we shall not give an operational semantics directly for it as, intuitively, the

behaviour of $p[\rho]$ should be identical to that of the process which results from applying ρ as a substitution to p. So, for any process p in $\mathbf{P}_\rho$, we can define the basic process in $\mathbf{P}$, $\mathbf{red}(p)$, which intuitively captures the behaviour of p by:

(i) $\mathbf{red}(p[\rho]) = \mathbf{red}(p)\rho$

(ii) $\mathbf{red}(op(p_1, ..., p_k)) = op(\mathbf{red}(p_1), ..., \mathbf{red}(pk))$ $(op \in \Sigma)$.

In this way, any semantic equivalence Eq defined over $\mathbf{P}$ is automatically extended to $\mathbf{P}_\rho$ by

$$\langle p, q \rangle \in Eq \text{ iff } \langle \mathbf{red}(p), \mathbf{red}(q) \rangle \in Eq.$$

In particular, this gives a definition of $\sim$, $\sim_t$, $\approx$, and $\approx_t$ over $\mathbf{P}_\rho$. We are interested in developing a reasonable semantic equivalence for $\mathbf{P}_\rho$, in particular one which abstracts from internal actions. A minimal requirement is that it should be preserved by all $\mathbf{P}_\rho$-contexts. This immediately rules out $\approx$, as we know from [3] that it is not preserved by action refinements. For example, $a|b \approx a;b+b;a$, but if we use a refinement ρ such that $\rho(a) = a_1; a_2$ then $(a|b)\rho \not\approx (a;b+b;a)\rho$. In [3] it was shown that, for a simple subset of $\mathbf{P}_\rho$ without communication, internal actions, and restriction, $\sim_t$ is a reasonable equivalence, namely, one which may be characterized by

$$p \sim_t q \text{ iff for every simple context } C[\cdot],\ C[p] \sim C[q].$$

In the extended language, one would hope to have the analogous result:

$$p \approx_t q \text{ iff for every context } C[\cdot],\ C[p] \approx C[q]. \tag{1}$$

This is not true for the trivial reason that $\approx_t$, in common with most forms of bisimulation equivalence, is not preserved by "+ contexts": for instance, $a \approx_t \tau; a$, but $b + a \not\approx_t b + \tau; a$. However, even the restricted statement

$$p \approx_t q \text{ only if for every action refinement } \rho,\ p\rho \approx_t q\rho \tag{2}$$

is *not* true. In fact, $\approx_t$ is not preserved by action refinement over the language $\mathbf{P}$. The counterexample is due to van Glabbeek and Vaandrager [23], who originally developed it to show that, in general, the equivalence obtained by splitting an action in two is different from that obtained by splitting it in three. This example is discussed in Section 6.

In the next section we define a modified version of $\approx_t$, called *weak refine equivalence* and denoted by $\approx_r$, for which (2) and an appropriate version of (1) are true. This equivalence is essentially the weak version of $\sim_r$ defined in [3].

3. Refine Equivalence

The original motivation for developing the timed equivalences $\sim_t$ and $\approx_t$ was to develop a bisimulation theory for processes where actions are no longer atomic. The intention was to "match up" actions from equivalent processes in the usual way required for bisimulation-like equivalences, but, in addition, to allow them to take some time. However, by simply requiring that the subactions $S(a)$ and $F(a)$ be matched up properly, there is no guarantee that the original complete (but nonatomic) actions are also properly matched. In fact, it may be that the finish of a particular action is matched to the finish of a different action with the same name which started either before or after it. The equivalence $\sim_r$ of [3] was designed to ensure that this cannot occur and that therefore the complete actions are indeed properly matched. We shall now develop a weak version of $\sim_r$, which will be denoted by $\approx_r$.

The definition requires us to distinguish all occurrences of actions in a process and therefore technically we use labelled actions obtained from the actions in $Act(\Lambda)$. Let $LAct(\Lambda)$ denote the set of actions $\{a_i \mid a \in V(\Lambda), i \in \mathbf{N}\} \cup \{\tau\}$ ranged over by λ. We use $\mathbf{LP}$ and $\mathbf{L}\mathcal{S}$ to denote the sets of *labelled processes* and *labelled states*, respectively obtained by using $LAct(\Lambda)$ in place of $Act(\Lambda)$ in the definition of $\mathbf{P}$ and $\mathcal{S}$, subject to the following restrictions:

1. each index i occurs *at most once* in each labelled process and labelled state, and

2. each labelled state of the form $c\backslash\alpha$ in $\mathbf{L}\mathcal{S}$ satisfies the constraint that, for each i, $F(\alpha_i)$ and $F(\bar{\alpha}_i)$ do not occur in c (this constraint is the natural labelled version of condition (CR)).

Note that, in this case, restriction is still with respect to a channel name from Λ and *not* a labelled channel name. We use $\pi, \pi', \ldots$ to range over labelled processes and $c, d, \ldots$ to range over labelled states. In what follows, we shall often refer to the processes in $\mathbf{P}$ as "unlabelled processes."

The next-state relations $\overset{S(a_i)}{\longmapsto}$ and $\overset{F(a_i)}{\longmapsto}$ and the termination predicate $\surd$ are inherited directly from the rules in Fig. 2, provided the *admits* predicate is extended so that:

(i) α *admits* τ

(ii) α *admits* a_i, $S(a_i)$, $F(a_i)$ iff $a \neq \alpha, \bar{\alpha}$.

Moreover, the transitions $\overset{a_i}{\longmapsto}$ and $\overset{\tau}{\longmapsto}$ are obtained by simply adapting the axiom and rules presented in Fig. 1 to $\mathbf{L}\mathcal{S}$, where rule (SYN) in Fig. 1 is replaced with

(LSYN) $c \overset{a_i}{\longmapsto} c'$ and $d \overset{\bar{a}_j}{\longmapsto} d'$ imply $c \mid d \overset{\tau}{\longmapsto} c' \mid d'$.

The net effect of these modifications is quite natural; restriction applies to all occurrences of actions using the channel name in question and communication may occur between any occurrences of complementary actions. We say that a labelled state c is *stable* iff $c \stackrel{\tau}{\not\longmapsto}$. The weak termination predicate $\downarrow\!\!\!\!/$ over $\mathbf{L}\mathscr{S}$ is then defined as in Section 2.1.

Notation 3.1. The set of *labelled subactions*, $LAct_s(\Lambda)$, is given by

$$LAct_s(\Lambda) = \{S(a_i), F(a_i) \mid a_i \in LAct(\Lambda)\} \cup \{\tau\}.$$

The weak transition relations $\stackrel{\sigma}{\Longmapsto}$, for $\sigma \in LAct(\Lambda)^* \cup LAct_s(\Lambda)^*$, are defined in the standard way.

To ensure that complete actions, consisting of the occurrence of a start subaction $S(a_i)$ and of the corresponding finish subaction $F(a_i)$, are properly matched we need to retain information about which occurrences of actions have already been started and to whom they have been matched. Such information is recorded in what was called a *history* in [3]. A history h is a $V(\Lambda)$-indexed family of partial bijections over $\mathbf{N}$. We use $\mathscr{H}$ to denote the set of all histories (h, ϕ, φ will be used to range over $\mathscr{H}$). Labelled states will be equivalent or inequivalent with respect to a given history. For c and d to be equivalent with respect to h we require that:

(1) h must be a history compatible with c and d. In other words, all actions which have started in c or d and have not yet finished must be recorded in the history h. Formally, let $\mathbf{Places}(a, c) = \{i \mid F(a_i)$ occurs in $c\}$. We then say that h is compatible with c and d iff for all a, $\mathbf{Places}(a, c) = dom(h_a)$ and $\mathbf{Places}(a, d) = range(h_a)$.

(2) Start moves have to be matched. For example, every start move of c, $c \stackrel{S(a_i)}{\longmapsto} c'$, must be matched by a corresponding move $d \stackrel{S(a_j)}{\Longmapsto} d'$ of d such that c' and d' are equivalent with respect to the augmented history $h \cup \{(a, i, j)\}$. This is the way histories are built up dynamically.

(3) Finish moves must be matched in a way which is consistent with the history h. For example, every finish move from c, $c \stackrel{F(a_i)}{\longmapsto} c'$, must be matched by the proper finish move from d, i.e., $d \stackrel{F(a_j)}{\Longmapsto} d'$ for some d' and j such that $(i, j) \in h_a$ and c' and d' are equivalent with respect to the diminished history $h - \{(a, i, j)\}$.

(4) Silent moves must be matched as usual.

(5) Successful termination must be matched as usual.

This is the motivation underlying the following definition. First of all, let us say that a family of binary relations $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ over $\mathbf{L}\mathscr{S}$ is *symmetric* iff $\langle c, d\rangle \in \mathscr{R}_h$ implies $\langle d, c\rangle \in \mathscr{R}_{h^{op}}$, where $h^{op} = \{(a, j, i) \mid (a, i, j) \in h\}$.

DEFINITION 3.1 (Weak Refine Bisimulation). A symmetric family of binary relations $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ over $\mathbf{L}\mathscr{S}$ is called a *weak refine bisimulation* whenever it satisfies, for each $\langle c, d\rangle \in \mathscr{R}_h$ and $a_i \in LAct(\Lambda)$,

(1) h is compatible with $\langle c, d\rangle$,

(2) $c \xmapsto{S(a_i)} c'$ implies $d \xRightarrow{S(a_j)} d'$ for some j and d' such that $\langle c', d'\rangle \in \mathscr{R}_{h \cup \{(a,i,j)\}}$,

(3) $c \xmapsto{F(a_i)} c'$ implies $d \xRightarrow{F(a_j)} d'$ for some j and d' such that $h_a(i) = j$ and $\langle c', d'\rangle \in \mathscr{R}_{h - \{(a,i,j)\}}$,

(4) $c \xmapsto{\tau} c'$ implies $d \xRightarrow{\varepsilon} d'$ for some d' such that $\langle c', d'\rangle \in \mathscr{R}_h$,

(5) $c\!\not\!\!\downarrow$ implies $d\!\not\!\!\downarrow$.

For the standard reasons it turns out that there is a largest refine bisimulation, which we denote by $\approx_{\mathscr{H}} = \{\approx_h \mid h \in \mathscr{H}\}$, and in fact the relations $\approx_h$ can be defined by

$$\approx_h = \bigcup \{\mathscr{R}_h \mid \{\mathscr{R}_h \mid h \in \mathscr{H}\} \text{ is a refine bisimulation}\}.$$

Of course, we are only interested in $\approx_{\mathscr{H}}$ in so far as it induces an equivalence relation on unlabelled states and, in particular, unlabelled processes. For $c \in \mathbf{L}\mathscr{S}$ let $\mathbf{un}(c)$ be the unlabelled state obtained by replacing each occurrence of a_i and $F(a_i)$ by a and $F(a)$, respectively. We may then define

$$s \approx_{\mathrm{r}} s' \text{ iff for some } c, d, h, \mathbf{un}(c) = s, \mathbf{un}(d) = s' \text{ and } c \approx_h d.$$

If $\mathbf{un}(c)$ is a process, i.e., is in $\mathbf{P}$, then c must be in $\mathbf{LP}$ and any history compatible with labelled processes must be empty. Thus this definition specializes to:

$p \approx_{\mathrm{r}} q$ iff there exist labelled processes π and π' such that

(a) $\mathbf{un}(\pi) = p$, $\mathbf{un}(\pi') = q$ and

(b) $\pi \approx_{\varnothing} \pi'$.

The reader familiar with [20, 22, 26] will have noted the formal similarities between $\approx_{\mathrm{r}}$ and ST-bisimulation equivalence. Indeed, both these equivalences use information on the matchings between actions in an essential way. However, $\approx_{\mathrm{r}}$ only maintains the associations between the actions which are currently being executed, whilst ST-bisimulation uses the whole history of associations between actions. Refine equivalence is the central semantic equivalence of the paper, the remainder of which is devoted to showing that it has many natural properties and to convincing the reader that it is an appropriate equivalence for process algebras which support refinement.

DEFINITION 3.2. For each $c \in \mathbf{L}\mathscr{S}$, the history associated with c, $id(c)$, is given by

$$id(c) =_{\text{def}} \{(a, i, i) \mid a \in V(\Lambda) \text{ and } i \in \mathbf{Places}(a, c)\}.$$

FACT 3.1. (1) *For each $a \in V(\Lambda)$, the following statements hold:*

(a) $\mathbf{Places}(a, \pi) = \varnothing$, *for each* $\pi \in \mathbf{LP}$;

(b) $\mathbf{Places}(a, c; \pi) = \mathbf{Places}(a, c)$, *for each* $c; \pi \in \mathbf{L}\mathscr{S}$;

(c) $\mathbf{Places}(a, c_1 \mid c_2) = \mathbf{Places}(a, c_1) \cup \mathbf{Places}(a, c_2)$, *for each* $c_1 \mid c_2 \in \mathbf{L}\mathscr{S}$; *and*

(d) $\mathbf{Places}(a, c\backslash\alpha) = \mathbf{Places}(a, c)$, *for each* $c\backslash\alpha \in \mathbf{L}\mathscr{S}$.

(2) *The following properties of $id(c)$, $c \in \mathbf{L}\mathscr{S}$, hold:*

(a) $id(\pi) = \varnothing$,

(b) $id(c; \pi) = id(c)$,

(c) $id(c \mid d) = id(c) \cup id(d)$ *and*

(d) $id(c\backslash\alpha) = id(c)$.

Proof. The verifications of all the properties are straightforward. The only interesting point to note is that statements (1d) and (2d) depend on the natural condition (CR) that we have imposed on states and labelled states of the form $c\backslash\alpha$. This restriction ensures that, whenever $c \xrightarrow{F(a_i)}$ and $c\backslash\alpha \in \mathbf{L}\mathscr{S}$, one has that $a \neq \alpha, \bar{\alpha}$ and thus $c\backslash\alpha \xrightarrow{F(a_i)}$. ∎

The following properties of $\approx_{h'}$ will be useful in proving that $\approx_r$ is an equivalence relation over $\mathscr{S}$.

LEMMA 3.1. *For each $c, d, c' \in \mathbf{L}\mathscr{S}$ the following statements hold:*

(i) $c \approx_{id(c)} c$,

(ii) $c \approx_h d$ *iff* $d \approx_{h^{op}} c$,

(iii) $c \approx_h c'$ *and* $c' \approx_{h'} d$ *imply* $c \approx_{h' \circ h} d$.

PROPOSITION 3.1. $\approx_r$ *is an equivalence relation over* $\mathscr{S}$.

Proof. Reflexivity of $\approx_r$ follows by Lemma 3.1(i) and the fact that, for each $s \in \mathscr{S}$, there exists $c \in \mathbf{L}\mathscr{S}$ such that $s = \mathbf{un}(c)$. Symmetry and transitivity of $\approx_r$ are simple consequences of parts (ii) and (iii) of Lemma 3.1, respectively. ∎

In the definition of weak refine bisimulation we did not require the explicit matching of moves labelled by complete actions a_i. A natural question to ask is whether the addition of a clause requiring the matching of

such moves in the definition of $\approx_{\mathscr{H}}$ changes the resulting notion of equivalence. We shall now prove that, as argued informally above, $\approx_{\mathscr{H}}$ does ensure that complete actions are matched. The proof of this fact makes use of two useful "commuting transitions" results which will also find application in the proof of the refinement theorem. First of all, we define a version of $\approx_{\mathscr{H}}$ which explicitly requires the matching of transitions labelled by complete actions. It will then be shown that this new relation coincides with $\approx_{\mathscr{H}}$.

DEFINITION 3.3 (Augmented Weak Refine Bisimulation). A symmetric family of binary relations $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ over $\mathbf{L}\mathscr{S}$ is called an *augmented weak refine bisimulation* whenever it satisfies, for each $h \in \mathscr{H}$, $\langle c, d\rangle \in \mathscr{R}_h$, and $a_i \in LAct(\Lambda)$, clauses (1)–(5) of the definition of $\approx_{\mathscr{H}}$ and

(6) $c \overset{a_i}{\longmapsto} c'$ implies $d \overset{a_j}{\Longmapsto} d'$ for some j and d' such that $\langle c', d'\rangle \in \mathscr{R}_h$.

$\approx^a_{\mathscr{H}} = \{\approx^a_h \mid h \in \mathscr{H}\}$ will denote the largest such relation.

We shall now prove the first of the "commuting transitions" results. This result formalizes the intuition that if $c \overset{S(a_i)}{\longmapsto} c' \overset{\lambda}{\longmapsto} d$, for some c' and labelled action λ, then the move $c' \overset{\lambda}{\longmapsto} d$ does *not* depend on the occurrence of action a_i. Hence it could have occurred before the start of action a_i without influencing the resulting target state d. Formally:

PROPOSITION 3.2 (Commuting Start-Moves and Complete Moves). *Let* $c \in \mathbf{L}\mathscr{S}$ *and* $\lambda \in LAct(\Lambda)$. *Assume that* $c \overset{S(a_i)}{\longmapsto} c' \overset{\lambda}{\longmapsto} d$. *Then there exists* $\bar{c} \in \mathbf{L}\mathscr{S}$ *such that* $c \overset{\lambda}{\longmapsto} \bar{c} \overset{S(a_i)}{\longmapsto} d$.

Proof. By induction on the structure of c. ∎

The following result is an immediate corollary of the above proposition. In its statement we shall use the notation $l(c \overset{\sigma}{\Longmapsto} d)$ to denote the *length* of the derivation $c \overset{\sigma}{\Longmapsto} d$.

COROLLARY 3.1. *Let* $c \in \mathbf{L}\mathscr{S}$ *and* $\sigma \in LAct(\Lambda)^*$. *Assume that* $c \overset{S(a_i)}{\longmapsto} c' \overset{\sigma}{\Longmapsto} d$. *Then there exists* $\bar{c} \in \mathbf{L}\mathscr{S}$ *such that* $c \overset{\sigma}{\Longmapsto} \bar{c} \overset{S(a_i)}{\longmapsto} d$ *and* $l(c' \overset{\sigma}{\Longmapsto} d) = l(c \overset{\sigma}{\Longmapsto} \bar{c})$.

The following proposition and associated corollary state dual results to Proposition 3.2 and Corollary 3.1, respectively, for end-transitions. Intuitively, if $c \overset{\lambda}{\longmapsto} c' \overset{F(a_i)}{\longmapsto} d$ then the move $c' \overset{F(a_i)}{\longmapsto} d$ does not depend on the occurrence of action λ. Hence the finish action $F(a_i)$ might have occurred before the action λ without influencing the resulting target state.

PROPOSITION 3.3 (Commuting End-Moves and Complete Moves). *Let* $c \in \mathbf{L}\mathscr{S}$ *and* $\lambda \in LAct(\Lambda)$. *Assume that* $c \overset{\lambda}{\longmapsto} c' \overset{F(a_i)}{\longmapsto} d$. *Then there exists* $\bar{c} \in \mathbf{L}\mathscr{S}$ *such that* $c \overset{F(a_i)}{\longmapsto} \bar{c} \overset{\lambda}{\longmapsto} d$.

Proof. By structural induction on c. ∎

COROLLARY 3.2. *Let* $c \in \mathbf{L}\mathscr{S}$ *and* $\sigma \in LAct(\Lambda)^*$. *Assume that* $c \overset{\sigma}{\Longmapsto} c' \overset{F(a_i)}{\longmapsto} d$. *Then there exists* $\bar{c}$ *such that* $c \overset{F(a_i)}{\longmapsto} \bar{c} \overset{\sigma}{\Longmapsto} d$ *and* $l(c \overset{\sigma}{\Longmapsto} c') = l(\bar{c} \overset{\sigma}{\Longmapsto} d)$.

The following lemma relates the transition relation associated with a complete action a_i with those associated with its subactions $S(a_i)$ and $F(a_i)$.

LEMMA 3.2. *Let* $c, d \in \mathbf{L}\mathscr{S}$. *Then* $c \overset{a_i}{\longmapsto} d$ *iff there exists* c' *such that* $c \overset{S(a_i)}{\longmapsto} c' \overset{F(a_i)}{\longmapsto} d$.

Proof. The "only if" implication is proven by induction on the proof of the transition $c \overset{a_i}{\longmapsto} d$. The "if" implication can be shown by structural induction on c. ∎

We can now prove that $\approx_{\mathscr{H}}$ and $\approx^a_{\mathscr{H}}$ coincide over $\mathbf{L}\mathscr{S}$.

PROPOSITION 3.4. *For each* $c, d \in \mathbf{L}\mathscr{S}$ *and* $h \in \mathscr{H}$, $c \approx_h d$ *iff* $c \approx^a_h d$.

Proof. The "if" implication follows immediately by the definitions of the two relations. For the "only if" implication it is sufficient to show that $\approx_{\mathscr{H}}$ is an augmented weak refine bisimulation. The only interesting thing to check is that, for $c \approx_h d$,

$$c \overset{a_i}{\longmapsto} c' \text{ implies } d \overset{a_j}{\Longmapsto} d' \text{ for some } d' \text{ and } j \text{ such that } c' \approx_h d'.$$

Assume then that $c \approx_h d$ and $c \overset{a_i}{\longmapsto} c'$. By the above lemma, there exists $\bar{c}$ such that $c \overset{S(a_i)}{\longmapsto} \bar{c} \overset{F(a_i)}{\longmapsto} c'$. As $c \approx_h d$, there exist j and $\bar{d}$ such that

$$d \overset{S(a_j)}{\Longmapsto} \bar{d} \text{ and } \bar{c} \approx_{h \cup \{(a,i,j)\}} \bar{d}.$$

Again, as $\bar{c} \overset{F(a_i)}{\longmapsto} c'$, there exists d' such that

$$\bar{d} \overset{F(a_j)}{\Longmapsto} d' \text{ and } c' \approx_h d'.$$

Thus we have that $d \overset{S(a_j)}{\Longmapsto} \bar{d} \overset{F(a_j)}{\Longmapsto} d'$ for some d' such that $c' \approx_h d'$. We are now left to prove that $d \overset{a_j}{\Longmapsto} d'$. By Corollaries 3.1 and 3.2, there exist d_1 and d_2 such that

$$d \overset{\varepsilon}{\Longmapsto} d_1 \overset{S(a_j)}{\longmapsto} \bar{d} \overset{F(a_j)}{\longmapsto} d_2 \overset{\varepsilon}{\Longmapsto} d'.$$

By the above lemma, $d_1 \stackrel{S(a_j)}{\longmapsto} \bar{d} \stackrel{F(a_j)}{\longmapsto} d_2$ implies $d_1 \stackrel{a_j}{\longmapsto} d_2$. Thus we have that $d \stackrel{a_j}{\Longrightarrow} d'$.

We have thus shown that $\approx_{\mathscr{H}}$ is an augmented weak refine bisimulation. ▮

This finishes our examination of the definition of $\approx_r$. Let us now turn our attention to some of its properties. We shall prove, first of all, that $\approx_r$ is preserved by all the combinators in Σ apart from the nondeterministic choice operator $+$. This statement will follow from some useful properties of $\approx_{\mathscr{H}}$ studied in the following lemma.

LEMMA 3.3. (1) *Let* $c, d, \pi_1, \pi_2 \in \mathbf{L}\mathscr{S}$ *be such that* $c \approx_h d$, $\pi_1 \approx_{\varnothing} \pi_2$ *and* $c; \pi_1, d; \pi_2 \in \mathbf{L}\mathscr{S}$. *Then* $c; \pi_1 \approx_h d; \pi_2$.

(2) *Let* $c_i, d_i \in \mathbf{L}\mathscr{S}$, $i = 1, 2$, *be such that* $c_1 \approx_h c_2$, $d_1 \approx_{h'} d_2$ *and* $c_1 | d_1$, $c_2 | d_2 \in \mathbf{L}\mathscr{S}$. *Then* $c_1 | d_1 \approx_{h \cup h'} c_2 | d_2$.

(3) *Let* $c, d \in \mathbf{L}\mathscr{S}$ *and* $\alpha \in \Lambda$ *be such that* $c \approx_h d$ *and* $c\backslash\alpha$, $d\backslash\alpha \in \mathbf{L}\mathscr{S}$. *Then* $c\backslash\alpha \approx_h d\backslash\alpha$.

Proof. We only give the proof of statement (3). Assume that $c \approx_h d$ and $c\backslash\alpha$, $d\backslash\alpha \in \mathbf{L}\mathscr{S}$. In order to prove that $c\backslash\alpha \approx_h d\backslash\alpha$, it is sufficient to show that the family of relations $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ given by

$$\mathscr{R}_h = \{\langle c\backslash\alpha, d\backslash\alpha\rangle \mid c \approx_h d \text{ and } c\backslash\alpha, d\backslash\alpha \in \mathbf{L}\mathscr{S}\}$$

is a weak refine bisimulation. We check that the defining clauses of $\approx_{\mathscr{H}}$ are met by $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$. Assume that $\langle c\backslash\alpha, d\backslash\alpha\rangle \in \mathscr{R}_h$. Then:

- h is compatible with $\langle c\backslash\alpha, d\backslash\alpha\rangle \in \mathscr{R}_h$. In fact, for each $a \in V(\Lambda)$, $\mathbf{Places}(a, c\backslash\alpha) = \mathbf{Places}(a, c) = dom(h_a)$ and $\mathbf{Places}(a, d\backslash\alpha) = \mathbf{Places}(a, d) = range(h_a)$.
- Assume that $c\backslash\alpha \stackrel{S(a_j)}{\longmapsto} c'\backslash\alpha$. Then $c \stackrel{S(a_j)}{\longmapsto} c'$ and $a \neq \alpha, \bar{\alpha}$. As $c \approx_h d$, there exist d' and j such that $d \stackrel{S(a_j)}{\Longrightarrow} d'$ and $c' \approx_{h'} d'$, where $h' = h \cup \{(a, i, j)\}$. By the operational semantics, $d\backslash\alpha \stackrel{S(a_j)}{\Longrightarrow} d'\backslash\alpha$. As $c\backslash\alpha$, $d\backslash\alpha \in \mathbf{L}\mathscr{S}$, we then have that $c'\backslash\alpha$, $d'\backslash\alpha \in \mathbf{L}\mathscr{S}$. Thus $\langle c'\backslash\alpha, d'\backslash\alpha\rangle \in \mathscr{R}_{h'}$.
- Clauses (3) and (4) are checked in similar fashion.
- Assume that $c\backslash\alpha\!\!\downarrow$. Then it is easy to see that

$$\begin{aligned} c\backslash\alpha\!\!\downarrow &\Leftrightarrow c\!\!\downarrow \\ &\Rightarrow d\!\!\downarrow \qquad \text{as } c \approx_h d \\ &\Leftrightarrow d\backslash\alpha\!\!\downarrow. \end{aligned}$$

Hence $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ is a weak refine bisimulation. ▮

PROPOSITION 3.5. *$\approx_r$ is preserved by the operators in $\Sigma - \{+\}$.*

Proof. The claim is an immediate consequence of the above lemma. For example, assume that $s_1 \approx_r s_2$ and $s_1\backslash\alpha$, $s_2\backslash\alpha \in \mathscr{S}$. By the definition of $\approx_r$, there exist $c_i \in \mathbf{L}\mathscr{S}$, $i = 1, 2$, and $h \in \mathscr{H}$ such that $c_1 \approx_h c_2$ and $\mathbf{un}(c_i) = s_i$, $i = 1, 2$. It is easy to see that, as $s_i\backslash\alpha \in \mathscr{S}$, $c_i\backslash\alpha \in \mathbf{L}\mathscr{S}$ $(i = 1, 2)$. Then, by the above lemma, $c_1\backslash\alpha \approx_h c_2\backslash\alpha$. Thus $s_1\backslash\alpha \approx_r s_2\backslash\alpha$. ∎

As it is the case with many forms of weak bisimulation relations, $\approx_r$ is not preserved by $+$. In fact, it is easy to see that $a \approx_r \tau; a$, but $a + b \not\approx_r \tau; a + b$. However, we are mainly interested in the operator of action refinement and this brings us to the first major theorem of this paper.

THEOREM 3.1 (The Refinement Theorem). *Let $p, q \in \mathbf{P}$. Assume that $p \approx_r q$. Then, for any action refinement ρ, $p\rho \approx_r q\rho$.*

The proof of this theorem is quite complex and is relegated to the following section. It involves decomposing moves from labelled states of the form $c\rho$ into moves from c and the components of ρ and the converse, combining moves from c and the components of ρ to form moves of $c\rho$.

With this theorem we have that $\approx_r$, modified in the usual way to compensate for $+$, is preserved by all extended (i.e., $\mathbf{P}_\rho$) contexts. Let

$p =_r q$ iff for some action a not appearing in p and q, $p + a \approx_r q + a$.

THEOREM 3.2. *Let $p, q \in \mathbf{P}$. Then:*

(a) *$p =_r q$ iff for every $\mathbf{P}_\rho$-context $C[\cdot]$, $C[p] =_r C[q]$.*

(b) *$p =_r q$ iff for every $\mathbf{P}_\rho$-context $C[\cdot]$, $C[p] \approx_r C[q]$.*

Proof. We prove the two statements separately.

(a) The "if" direction is immediate by taking the empty context. For the converse, one can easily show that $=_r$, unlike $\approx_r$, is preserved by $+$. For the other operators in Σ, the proofs that they are preserved by $\approx_r$ may be trivially adapted to $=_r$ since these relations only differ for initial τ moves. So it remains to prove that $p =_r q$ implies $p\rho =_r q\rho$, for any action refinement ρ. Let $a \in V(\Lambda)$ be an action not appearing in $p\rho$ and $q\rho$ and, for convenience, assume also that a does not occur in p and q. We prove that $p\rho + a \approx_r q\rho + a$. Let ρ' be defined to coincide with ρ on all the free channels of p and q, but map a to itself. Since $p =_r q$, we have that $p + a \approx_r q + a$. By the refinement theorem it follows that $(p + a)\,\rho' \approx_r (q + a)\,\rho'$, i.e., $p\rho' + a \approx_r q\rho' + a$. The result now follows because $p\rho = p\rho'$ and $q\rho = q\rho'$ by Lemma 2.2(c).

(b) The "only if" direction follows from (a) because $=_r$ is contained in $\approx_r$. The converse is trivial since $=_r$ is defined using the context $[\cdot]+a$. ∎

In Section 5 we shall see that this theorem can be strengthened considerably, giving a characterization of $=_r$ in terms of $\approx$.

4. The Refinement Theorem

This section will be entirely devoted to a detailed proof of Theorem 3.1 (the refinement theorem). Let us recall, for the sake of completeness, that the refinement theorem states that, for all $p, q \in \mathbf{P}$ and action refinement ρ,

$$p \approx_r q \text{ implies } p\rho \approx_r q\rho. \tag{3}$$

Naturally enough, given the fundamental role played by labelled states in the definition of $\approx_r$, the proof of statement (3) will rely upon a detailed analysis of some operational properties of labelled states and we shall present labelled counterparts to several properties of processes and states studied in the previous sections. The proof of (3) will be given in several stages. First of all, a suitable notion of *labelled action refinement* is introduced as a labelled counterpart of the action refinements given in Definition 2.1. The application of labelled action refinements to labelled processes and states will be defined following the approach outlined in Section 2.2 for processes. We shall then prove a series of composition/decomposition results for moves of labelled terms of the form $c\rho$, where $c \in \mathbf{L}\mathscr{S}$ and ρ is a labelled action refinement. These results will allow us to decompose moves of $c\rho$ into moves of c and of the components of ρ and, conversely, to compose moves of c and of the components of ρ to obtain moves of $c\rho$. Finally, the technical material developed in the stages outlined above will be used to prove a labelled version of (3) from which the unlabelled one follows immediately.

We shall now introduce the labelled counterpart of the action refinements presented in Section 2.2. The definition of these labelled action refinements is somewhat technical; the technicalities, however, are needed to deal with the added complexity introduced by the labelling on terms. In the following definition, we shall borrow some notation from the unlabelled calculus; we use $\rho(a_i) =_\varnothing \rho(a_j)$ to signify that $\rho(a_i) + b_k \approx_\varnothing \rho(a_j) + b_k$ for some action b and index k not occurring in $\rho(a_i)$ and $\rho(a_j)$. This ensures that any τ-move performed by $\rho(a_i)$ must be matched by a corresponding τ-move from $\rho(a_j)$ rather than the empty move.

DEFINITION 4.1 (Labelled Action Refinements). A *labelled action refinement* ρ is a map $\rho: LAct(\Lambda) \cup \{F(a_i) \mid a_i \in LAct(\Lambda)\} \to \mathbf{L}\mathcal{S}$ which satisfies:

(a) for all $a_i \in LAct(\Lambda)$, $\rho(a_i) \in \mathbf{LP}$ and $\rho(a_i) \not\in \surd$ (i.e., labelled actions are mapped to nonterminated labelled processes),

(b) ρ satisfies (ComPres), and

(c) for all $a \in V(\Lambda)$, $i, j \in \mathbf{N}$, $\rho(a_i) =_{\varnothing} \rho(a_j)$ (i.e., instances of the same action are mapped to congruent labelled processes).

Note that, unlike labelled actions, labelled action refinements may map states of the form $F(a_i)$ onto terminated processes. The application of labelled action refinements to labelled states can now be defined following the approach outlined in the previous section. For each labelled state c, let $FC(c)$, the set of *free channel names* in c, be defined in exactly the same way as for processes, except that the labelling of actions is ignored. In particular, we have that $FC(F(\alpha_i)) = FC(F(\bar{\alpha}_i)) = \{\alpha\}$. In order to simplify the definition of the function *new*, we now introduce some useful notation.

Notation 4.1. For each $c \in \mathbf{L}\mathcal{S}$, the set of *symbols* in c, $Sym(c)$, is given by

$$Sym(c) =_{\text{def}} \{a_i \in LAct(\Lambda) \mid a_i \text{ occurs in } c\} \cup \{F(a_i) \mid F(a_i) \text{ occurs in } c\}.$$

For each $e \in LAct(\Lambda) \cup \{F(a_i) \mid a_i \in LAct(\Lambda)\}$, $v(e) \in \Lambda$ will denote the underlying channel name of e.

Following [38] and our development in the previous section, we define the function *new* by

$$new\ \alpha c \rho =_{\text{def}} \{\beta \mid \text{for all } \beta' \in FC(c) - \{\alpha\},\ \beta \not\in FC(\rho(e)) \text{ for all } e \in Sym(c) \text{ such that } v(e) = \beta'\}.$$

Note that the set $new\ \alpha c \rho$ is always infinite because Λ is infinite and, for each c, $Sym(c)$ and $FC(c)$ are both finite sets. The result of applying a labelled action refinement ρ to a labelled state c can now be defined, following Definition 2.2. In the following definition we shall use the notational conventions about substitutions from the previous section, with the understanding that, for each labelled action refinement ρ and channel name α, $\rho[\alpha \mapsto \beta]$ will denote the substitution identical to ρ except that α_i is mapped to β_i and $\bar{\alpha}_i$ to $\bar{\beta}_i$, for all i.

DEFINITION 4.2 (Application of Labelled Action Refinements). For each $c \in \mathbf{L}\mathcal{S}$ and labelled action refinement ρ, $c\rho$ is the labelled state defined by:

(i) $a_i\rho = \rho(a_i)$, $\tau\rho = \tau$, $nil\rho = nil$, $\delta\rho = \delta$, $F(a_i)\rho = \rho(F(a_i))$;

(ii) $(c;\pi)\rho = c\rho;\pi\rho$, $(\pi_1 + \pi_2)\rho = \pi_1\rho + \pi_2\rho$, $(c_1 \mid c_2)\rho = c_1\rho \mid c_2\rho$;

(iii) $(c\backslash\alpha)\rho = (c\rho[\alpha \mapsto \beta])\backslash\beta$, where $\beta = choice(new\ \alpha c\rho)$.

All the definitions and results regarding α-congruence and action refinements apply equally well to labelled action refinements. In particular, in what follows we shall often make use of the Substitution Lemma and Lemma 2.2 applied to labelled action refinements. Moreover, following the lines of the proof of Proposition 2.5, it is possible to prove that $=_\alpha$ is a strong bisimulation over the LTS with termination $\langle \mathbf{L}\mathscr{S}, \mapsto, LAct_s(\Lambda), \surd \rangle$. By an abuse of notation, the largest strong bisimulation over that LTS will also be denoted by $\sim$. For the sake of clarity, we state the following proposition.

PROPOSITION 4.1. *$=_\alpha$ is a strong bisimulation over $\langle \mathbf{L}\mathscr{S}, \mapsto, LAct_s(\Lambda), \surd \rangle$.*

As pointed out above, the proof of the refinement theorem will make an essential use of several composition and decomposition results which relate the moves of a labelled term of the form $c\rho$ to those of c and of the components of ρ. These results always apply to c and ρ which are compatible with each other.

DEFINITION 4.3. Let $c \in \mathbf{L}\mathscr{S}$ and ρ be a labelled action refinement. Then ρ is *compatible* with c iff $c\rho \in \mathbf{L}\mathscr{S}$.

We shall now present a series of results which analyze the origin of the moves of labelled states of the form $c\rho$ in terms of moves of c and of the components of ρ. The following lemma will be useful in the decomposition results presented below.

LEMMA 4.1 (From Start to Complete Moves). *Let $c \in \mathbf{L}\mathscr{S}$. Then:*

(1) *$c \overset{S(a_i)}{\longmapsto} d$ implies $c \overset{a_i}{\longmapsto} c'$ for some c' such that $c' =_\alpha d\iota[F(a_i) \to nil]$;*

(2) *$c \overset{S(a_i)}{\longmapsto} c_1 \overset{S(\bar{a}_j)}{\longmapsto} d$ implies $c \overset{\tau}{\longmapsto} c'$ for some c' such that $c' =_\alpha d\iota[F(a_i) \to nil, F(\bar{a}_j) \to nil]$.*

Proof. We limit ourselves to giving a proof of the second statement.

(2) By induction on the structure of c. We examine only one possibility, namely the one corresponding to the restriction operator.

• $c = \bar{c}\backslash\alpha$. Assume that $\bar{c}\backslash\alpha \overset{S(a_i)}{\longmapsto} c_1 \overset{S(\bar{a}_j)}{\longmapsto} d$. Then, by the operational semantics, this is because $\bar{c} \overset{S(a_i)}{\longmapsto} c' \overset{S(\bar{a}_j)}{\longmapsto} d'$, $d = d'\backslash\alpha$, and $a \neq \alpha, \bar{\alpha}$.

By the inductive hypothesis, $\bar{c} \stackrel{\tau}{\longmapsto} c_2$ for some c_2 such that $c_2 =_\alpha d'\iota[F(a_i) \to nil,\ F(\bar{a}_j) \to nil]$. By the operational semantics, $\bar{c}\backslash\alpha \stackrel{\tau}{\longmapsto} c_2\backslash\alpha$. We are thus left to prove that

$$c_2\backslash\alpha =_\alpha (d'\backslash\alpha)\,\iota[F(a_i) \to nil,\ F(\bar{a}_j) \to nil],$$

which follows from Lemma 2.2(d). ∎

The following theorem gives a decomposition of the moves of labelled processes of the form $\pi\rho$ in terms of those of π and of the components of ρ. In what follows, with abuse of notation, we shall use e to range over the set of *labelled subactions* $\{S(a_i), F(a_i) \mid a_i \in LAct(\Lambda)\} \cup \{\tau\}$.

THEOREM 4.1 ("Whence Theorem" for Labelled Processes). *Let π be a labelled process and ρ be a labelled action refinement compatible with π. Then $\pi\rho \stackrel{e}{\longmapsto} \bar{c}$ implies that*

(1) *there exist $a_i \in LAct(\Lambda)$ and $c, x \in \mathbf{L}\mathscr{S}$ such that $\pi \stackrel{S(a_i)}{\longmapsto} c$, $\rho(a_i) \stackrel{e}{\longmapsto} x$ and $\bar{c} =_\alpha c\rho[F(a_i) \to x]$, or*

(2) *$e = \tau$ and there exist $a_i, b_j \in LAct(\Lambda)$, $\alpha \in V(\Lambda)$ and $c, d, x, y \in \mathbf{L}\mathscr{S}$ such that $\pi \stackrel{S(a_i)}{\longmapsto} c \stackrel{S(b_j)}{\longmapsto} d$, $\rho(a_i) \stackrel{\alpha_h}{\longmapsto} x$, $\rho(b_j) \stackrel{\bar{\alpha}_k}{\longmapsto} y$ and $\bar{c} =_\alpha d\rho[F(a_i) \to x, F(b_j) \to y]$, or*

(3) *$e = \tau$ and $\pi \stackrel{\tau}{\longmapsto} \pi'$, for some π' such that $\bar{c} =_\alpha \pi'\rho$.*

Proof. By induction on the structure of π. We shall only present the details of two selected cases.

• $\pi = \pi_1 \mid \pi_2$. Assume that $(\pi_1 \mid \pi_2)\rho = \pi_1\rho \mid \pi_2\rho \stackrel{e}{\longmapsto} \bar{c}$. By the operational semantics, there are three possibilities to examine:

(A) $\pi_1\rho \mid \pi_2\rho \stackrel{e}{\longmapsto} \bar{c}$ because $\pi_1\rho \stackrel{e}{\longmapsto} \bar{c}_1$ and $\bar{c} = \bar{c}_1 \mid \pi_2\rho$, or

(B) $\pi_1\rho \mid \pi_2\rho \stackrel{e}{\longmapsto} \bar{c}$ because $\pi_2\rho \stackrel{e}{\longmapsto} \bar{c}_2$ and $\bar{c} = \pi_1\rho \mid \bar{c}_2$, or

(C) $\pi_1\rho \mid \pi_2\rho \stackrel{\tau}{\longmapsto} \bar{c}$ because $\pi_1\rho \stackrel{\alpha_h}{\longmapsto} \bar{c}_1$, $\pi_2\rho \stackrel{\bar{\alpha}_k}{\longmapsto} \bar{c}_2$ and $\bar{c} = \bar{c}_1 \mid \bar{c}_2$.

We examine each possibility in turn.

(A) In this case, we may apply the inductive hypothesis to $\pi_1\rho \stackrel{e}{\longmapsto} \bar{c}_1$ to obtain that

(A.1) there exist $a_i \in LAct(\Lambda)$ and $c, x \in \mathbf{L}\mathscr{S}$ such that $\pi_1 \stackrel{S(a_i)}{\longmapsto} c$, $\rho(a_i) \stackrel{e}{\longmapsto} x$ and $\bar{c}_1 =_\alpha c\rho[F(a_i) \to x]$, or

(A.2) $e = \tau$ and there exist $a_i, b_j \in LAct(\Lambda)$, $\alpha \in V(\Lambda)$ and $c, d, x, y \in \mathbf{L}\mathscr{S}$ such that $\pi_1 \stackrel{S(a_i)}{\longmapsto} c \stackrel{S(b_j)}{\longmapsto} d$, $\rho(a_i) \stackrel{\alpha_h}{\longmapsto} x$, $\rho(b_j) \stackrel{\bar{\alpha}_k}{\longmapsto} y$ and $\bar{c}_1 =_\alpha d\rho[F(a_i) \to x, F(b_j) \to y]$, or

(A.3) $e = \tau$ and $\pi_1 \stackrel{\tau}{\longmapsto} \pi_1'$, for some π_1' such that $\bar{c}_1 =_\alpha \pi_1'\rho$.

If (A.1) holds then, by the operational semantics, $\pi_1 | \pi_2 \xmapsto{S(a_i)} c | \pi_2$. Moreover,

$$\begin{aligned}(c | \pi_2)\, \rho[F(a_i) \to x] &= (c\rho[F(a_i) \to x]) | (\pi_2 \rho[F(a_i) \to x]) \\ &=_\alpha \bar{c}_1 | (\pi_2 \rho[F(a_i) \to x]).\end{aligned}$$

Now, as $F(a_i)$ does not occur in π_2, it is easy to see that $\pi_2 \rho[F(a_i) \to x] = \pi_2 \rho$. Thus we have that $(c | \pi_2)\, \rho[F(a_i) \to x] =_\alpha \bar{c}_1 | \pi_2 \rho = \bar{c}$. Clause (1) of the statement of the theorem is then met.

If (A.2) holds then, by the operational semantics, $\pi_1 | \pi_2 \xmapsto{S(a_i)} c | \pi_2 \xmapsto{S(b_j)} d | \pi_2$. Moreover,

$$\begin{aligned}&(d | \pi_2)\, \rho[F(a_i) \to x, F(b_j) \to y] \\ &\quad = (d\rho[F(a_i) \to x, F(b_j) \to y]) | (\pi_2 \rho[F(a_i) \to x, F(b_j) \to y]) \\ &\quad =_\alpha \bar{c}_1 | (\pi_2 \rho[F(a_i) \to x, F(b_j) \to y]).\end{aligned}$$

Now, as $F(a_i)$ and $F(b_j)$ do not occur in π_2, it follows that $\pi_2 \rho[F(a_i) \to x, F(b_j) \to y] = \pi_2 \rho$. Hence $(d | \pi_2)\, \rho[F(a_i) \to x, F(b_j) \to y] =_\alpha \bar{c}_1 | \pi_2 \rho = \bar{c}$. Clause (2) of the statement of the theorem is then met.

If (A.3) holds then, by the operational semantics, $\pi_1 | \pi_2 \xmapsto{\tau} \pi_1' | \pi_2$. Moreover,

$$(\pi_1' | \pi_2)\rho = (\pi_1' \rho | \pi_2 \rho) =_\alpha \bar{c}_1 | \pi_2 \rho.$$

Clause (3) of the statement of the theorem is then met. This completes the verification of case (A).

(B) Symmetrical to the one above.

(C) By the inductive hypothesis, $\pi_1 \rho \xmapsto{\alpha_h} \bar{c}_1$ implies that $\pi_1 \xmapsto{S(a_i)} c_1$, $\rho(a_i) \xmapsto{\alpha_h} x$ and $\bar{c}_1 =_\alpha c_1 \rho[F(a_i) \to x]$, for some a_i, c_1, and x. Again by the inductive hypothesis, $\pi_2 \rho \xmapsto{\bar{\alpha}_k} \bar{c}_2$ implies that $\pi_2 \xmapsto{S(b_i)} c_2$, $\rho(b_j) \xmapsto{\bar{\alpha}_k} y$ and $\bar{c}_2 =_\alpha c_2 \rho[F(b_j) \to y]$, for some b_j, c_2, and y. By the operational semantics, $\pi_1 | \pi_2 \xmapsto{S(a_i)} c_1 | \pi_2 \xmapsto{S(b_j)} c_1 | c_2$. Moreover,

$$\begin{aligned}&(c_1 | c_2)\, \rho[F(a_i) \to x, F(b_j) \to y] \\ &\quad = (c_1 \rho[F(a_i) \to x, F(b_j) \to y]) | (c_2 \rho[F(a_i) \to x, F(b_j) \to y]) \\ &\quad = (c_1 \rho[F(a_i) \to x]) | (c_2 \rho[F(b_j) \to y]),\end{aligned}$$

because $F(a_i)$ does not occur in c_2 and $F(b_j)$ does not occur in c_1. By the inductive hypothesis, we then have that

$$(c_1 | c_2)\, \rho[F(a_i) \to x, F(b_j) \to y] =_\alpha \bar{c}_1 | \bar{c}_2.$$

Clause (3) of the statement of the theorem is then met.

• $\pi = \pi_1 \backslash \alpha$. Assume that $(\pi_1 \backslash \alpha)\,\rho \stackrel{e}{\longmapsto} \bar{c}$. First of all, note that

$$(\pi_1 \backslash \alpha)\,\rho = (\pi_1 \rho[\alpha \mapsto \beta]) \backslash \beta,$$

where $\beta = choice(new\ \alpha \pi_1 \rho)$. By the operational semantics, we then have that $\pi_1 \rho[\alpha \mapsto \beta] \stackrel{e}{\longmapsto} \bar{c}_1$, $e \neq \beta, \bar{\beta}$ and $\bar{c} = \bar{c}_1 \backslash \beta$, for some $\bar{c}_1$. By the inductive hypothesis, $\pi_1 \rho[\alpha \mapsto \beta] \stackrel{e}{\longmapsto} \bar{c}_1$ implies that

(A) there exist $a_i \in LAct(\Lambda)$ and $c, x \in \mathbf{L}\mathscr{S}$ such that $\pi_1 \stackrel{S(a_i)}{\longmapsto} c$, $\rho[\alpha \mapsto \beta](a_i) \stackrel{e}{\longmapsto} x$, and $\bar{c}_1 =_\alpha c\rho[\alpha \mapsto \beta][F(a_i) \to x]$, or

(B) $e = \tau$ and there exist $a_i, b_j \in LAct(\Lambda)$, $\gamma \in V(\Lambda)$ and $c, d, x, y \in \mathbf{L}\mathscr{S}$ such that $\pi_1 \stackrel{S(a_i)}{\longmapsto} c \stackrel{S(b_j)}{\longmapsto} d$, $\rho[\alpha \mapsto \beta](a_i) \stackrel{\gamma_k}{\longmapsto} x$, $\rho[\alpha \mapsto \beta](b_j) \stackrel{\bar{\gamma}_k}{\longmapsto} y$, and $\bar{c}_1 =_\alpha d\rho[\alpha \mapsto \beta][F(a_i) \to x, F(b_j) \to y]$, or

(C) $e = \tau$ and $\pi_1 \stackrel{\tau}{\longmapsto} \pi_1'$, for some π_1' such that $\bar{c}_1 =_\alpha \pi_1' \rho[\alpha \mapsto \beta]$.

We proceed by examining the three possibilities separately.

(A) As $e \neq \beta, \bar{\beta}$ and $\rho[\alpha \mapsto \beta](\alpha_i) = \beta_i$, for all i, it must be the case that $a \neq \alpha, \bar{\alpha}$. Thus α *admits* $S(a_i)$ and, by the operational semantics, we have that $\pi_1 \backslash \alpha \stackrel{S(a_i)}{\longmapsto} c \backslash \alpha$. Moreover, as $a \neq \alpha, \bar{\alpha}$,

$$\rho(a_i) = \rho[\alpha \mapsto \beta](a_i) \stackrel{e}{\longmapsto} x.$$

We shall now show that

$$\bar{c}_1 \backslash \beta =_\alpha (c \backslash \alpha)\,\rho[F(a_i) \to x], \tag{4}$$

where $\bar{c}_1 =_\alpha c\rho[\alpha \mapsto \beta][F(a_i) \to x]$. In order to prove (4), note that

$$(c \backslash \alpha)\,\rho[F(a_i) \to x] = (c\rho[F(a_i) \to x][\alpha \mapsto \beta']) \backslash \beta',$$

where $\beta' = choice(new\ \alpha c \rho[F(a_i) \to x])$. We proceed by distinguishing two cases, depending on whether $\beta = \beta'$ or not.

If $\beta = \beta'$ then

$$c\rho[F(a_i) \to x][\alpha \mapsto \beta] = c\rho[\alpha \mapsto \beta][F(a_i) \to x] \quad \text{as } a \neq \alpha, \bar{\alpha}$$
$$=_\alpha \bar{c}_1.$$

The claim then follows by the substitutivity of $=_\alpha$.

Assume now that $\beta \neq \beta'$. Then it is easy to see that $\beta \in new\ \alpha c \rho[F(a_i) \to x]$. This implies that

$$\beta \notin FC(c\rho[F(a_i) \to x][\alpha \mapsto \beta']).$$

By the substitution lemma, we then have that

$$\begin{aligned}(c\rho[F(a_i) \to x][\alpha \mapsto \beta'])\,\iota[\beta' \mapsto \beta] &= c(\iota[\beta' \mapsto \beta] \circ \rho[F(a_i) \to x][\alpha \mapsto \beta']) \\ &= c\rho[F(a_i) \to x][\alpha \mapsto \beta] \\ &\stackrel{*}{=} c\rho[\alpha \mapsto \beta][F(a_i) \to x] \\ &=_\alpha \bar{c}_1.\end{aligned}$$

Equality $\stackrel{*}{=}$ is justified by the fact that $a \neq \alpha, \bar{\alpha}$. The claim then follows by rule (α) of the definition of $=_\alpha$. By $\pi_1\backslash\alpha \xmapsto{S(a_i)} c\backslash\alpha$, $\rho(a_i) \xmapsto{e} x$ and $\bar{c}_1\backslash\beta =_\alpha (c\backslash\alpha)\,\rho[F(a_i) \to x]$, we now have that clause (1) of the statement of the theorem is met.

(B) First of all, note that, because $\beta \in new\ \alpha\pi_1\rho$, $\rho[\alpha \mapsto \beta](a_i) \xmapsto{\gamma_k} x$ and $\rho[\alpha \mapsto \beta](b_j) \xmapsto{\bar{\gamma}_k} y$ imply that

(a) $a, b \neq \alpha, \bar{\alpha}$ or

(b) by symetry, we may assume that, w.l.o.g., $a = \alpha$ and $b = \bar{\alpha}$.

We examine the two possibilities separately.

(a) Assume that $a, b \neq \alpha, \bar{\alpha}$. Then, by the operational semantics, $\pi_1\backslash\alpha \xmapsto{S(a_i)} c\backslash\alpha \xmapsto{S(b_j)} d\backslash\alpha$. Moreover, we have that $\rho(a_i) = \rho[\alpha \mapsto \beta](a_i) \xmapsto{\gamma_k} x$ and $\rho(b_j) = \rho[\alpha \mapsto \beta](b_j) \xmapsto{\bar{\gamma}_k} y$. A simple argument will show that

$$(d\backslash\alpha)\,\rho[F(a_i) \to x, F(b_j) \to y] = \bar{c}_1\backslash\beta,$$

where $\bar{c}_1 =_\alpha d\rho[\alpha \mapsto \beta][F(a_i) \to x, F(b_j) \to y]$ and we therefore have that clause (2) of the statement of the theorem is met.

(b) Assume that $a = \alpha_i$ and $b_j = \bar{\alpha}_j$. Note that we then have that $\pi_1 \xmapsto{S(\alpha_i)} c \xmapsto{S(\bar{\alpha}_j)} d$, $\rho[\alpha \mapsto \beta](\alpha_i) = \beta_i \xmapsto{\beta_i} nil = x$ and $\rho[\alpha \mapsto \beta](\bar{\alpha}_j) = \bar{\beta}_j \xmapsto{\bar{\beta}_j} nil = y$. Moreover,

$$\bar{c}_1 =_\alpha d\rho[\alpha \mapsto \beta][F(\alpha_i) \to nil, F(\bar{\alpha}_j) \to nil].$$

We shall prove that clause (3) of the statement of the theorem is then met. First of all, note that, by Lemma 4.1, $\pi_1 \xmapsto{S(\alpha_i)} c \xmapsto{S(\bar{\alpha}_j)} d$ implies that

$$\pi_1 \xmapsto{\tau} \pi_1' =_\alpha d\iota[F(\alpha_i) \to nil, F(\bar{\alpha}_j) \to nil]$$

for some π_1'. By the operational semantics, $\pi_1 \xmapsto{\tau} \pi_1'$ implies that

$$\pi_1\backslash\alpha \xmapsto{\tau} \pi_1'\backslash\alpha =_\alpha (d\iota[F(\alpha_i) \to nil, F(\bar{\alpha}_j) \to nil])\backslash\alpha.$$

In order to show that clause (3) of the statement of the theorem is met, it is sufficient to show that

$$\bar{c}_1\backslash\beta =_\alpha (\pi_1'\backslash\alpha)\rho,$$

which again follows by a simple analysis of the result of applying ρ to $(\pi_1'\backslash\alpha)$ and by applying Lemma 2.2(b).

The verification of case (B) is thus complete.

(C) In this case, by the operational semantics, we have that $\pi_1\backslash\alpha \xmapsto{\tau} \pi_1'\backslash\alpha$. In order to prove that clause (3) of the statement of the theorem is met, it is sufficient to show that

$$\bar{c}_1\backslash\beta =_\alpha (\pi_1'\backslash\alpha)\rho.$$

This verification follows the same pattern as the similar ones given above and is thus omitted. ∎

We shall now work towards a "whence theorem" for configurations of the form $c\rho$. Such a result will allow us to decompose moves of a configuration of the form $c\rho$ into moves of c and of the components of ρ. In order to deal with action refinements mapping states of the form $F(a_i)$, or *places*, to terminated processes, we shall adapt the definition of the map η, defined for the simpler language in [3], to the richer setting we are now working in. For each labelled state c, $\eta(c)$ will be a nonempty collection of pairs of the form (X, d), where $X \subseteq \{F(a_i) \mid a_i \in LAct(\Lambda)\}$. Intuitively, if $(X, d) \in \eta(c)$ then, for every action refinement ρ mapping each element of X to a terminated process, $c\rho$ "behaves like" $d\rho$. The presence of condition (CR) on states of the form $c\backslash\alpha$ will allow us to give a neat compositional definition of the map η.

DEFINITION 4.4. The map η is defined by structural induction on c as follows:

(i) $\eta(\pi) = \{(\varnothing, \pi)\}$

(ii) $\eta(F(a_i)) = \{(\{F(a_i)\}, nil)\}$

(iii) $\eta(c; \pi) = \{(X, c'; \pi) \mid (X, c') \in \eta(c)\}$

(iv) $\eta(c \mid d) = \{(X, c' \mid d) \mid (X, c') \in \eta(c)\} \cup \{(Y, c \mid d') \mid (Y, d') \in \eta(d)\}$
$\cup \{(X \cup Y, c' \mid d') \mid (X, c') \in \eta(c) \text{ and } (Y, d') \in \eta(d)\}$

(v) $\eta(c\backslash\alpha) = \{(X, c'\backslash\alpha) \mid (X, c') \in \eta(c)\}$.

We shall now study some basic properties of η which will find application in what follows.

LEMMA 4.2. *Let* $c \in \mathbf{L}\mathcal{S}$. *Then*

(1) $(X, d) \in \eta(c)$ *implies* $X \subseteq \{F(a_i) \mid a_i \in LAct(\Lambda)\}$;

(2) *there exists* d *such that* $(\varnothing, d) \in \eta(c)$ *iff* c *is a labelled process*;

(3) $(X, d) \in \eta(c)$ *and* $\varnothing \neq Y \subseteq X$ *imply* $(Y, d') \in \eta(c)$ *for some* d'.

Condition (CR) plays an important role in the proof of the following lemma, which states a useful interaction between the map η and the operational semantics for labelled states.

LEMMA 4.3. *Let* $c \in \mathbf{L}\mathcal{S}$ *and assume that* $(\{F(a_i)\}, d) \in \eta(c)$. *Then* $c \xrightarrow{F(a_i)} d$.

Proof. By structural induction on c. We only give the case in which condition (CR) is used.

• $c = c_1 \backslash \alpha$. Assume that $(\{F(a_i)\}, d) \in \eta(c_1 \backslash \alpha)$. Then $d = d_1 \backslash \alpha$ and $(\{F(a_i)\}, d_1) \in \eta(c_1)$. By the inductive hypothesis, $c_1 \xrightarrow{F(a_i)} d_1$. By condition (CR), $c_1 \backslash \alpha \in \mathbf{L}\mathcal{S}$ implies that $a \neq \alpha, \bar{\alpha}$. Hence α *admits* $F(a_i)$ and, by the operational semantics, $c_1 \backslash \alpha \xrightarrow{F(a_i)} d_1 \backslash \alpha = d$. ∎

We shall now investigate the relationships between the map η and the function *new*. More precisely, we shall prove that whenever $(X, d) \in \eta(c)$ and ρ is a labelled action refinement such that $\rho(X)\surd$ then, for each channel name α, $new\ \alpha c \rho = new\ \alpha d \rho$.

LEMMA 4.4. *Let* $c \in \mathbf{L}\mathcal{S}$ *and* ρ *be a labelled action refinement. Then* $(X, d) \in \eta(c)$ *and* $\rho(X)\surd$ *imply that* $new\ \alpha c \rho = new\ \alpha d \rho$, *for all* $\alpha \in \Lambda$.

Proof. By induction on the structure of c. We shall examine only three of the possible forms of c and leave the remaining ones to the reader.

• $c = F(a_i)$. By the definition of η, (X, d) must be $(\{F(a_i)\}, nil)$. Moreover we have that $\rho(F(a_i))\surd$. By the definition of *new*, for all $\alpha \in \Lambda$,

$$\begin{aligned} new\ \alpha F(a_i) \rho &= \Lambda \\ &= new\ \alpha\ nil\ \rho, \end{aligned}$$

because $\rho(F(a_i))\surd$ implies that $FC(\rho(F(a_i))) = \varnothing$.

• $c = c_1 ; \pi$. Assume that $(X, d) \in \eta(c_1 ; \pi)$ and $\rho(X)\surd$. By the definition of η, it must be the case that $(X, c_1') \in \eta(c_1)$ and $d = c_1' ; \pi$, for some c_1'. Now, by the definition of *new*, it is easy to see that

$$\begin{aligned} new\ \alpha c_1 ; \pi \rho &= new\ \alpha c_1 \rho \cap new\ \alpha \pi \rho \\ &= new\ \alpha c_1' \rho \cap new\ \alpha \pi \rho \qquad \text{by induction} \\ &= new\ \alpha c_1' ; \pi \rho. \end{aligned}$$

• $c = c_1 \backslash \beta$. Assume that $(X, d) \in \eta(c_1 \backslash \beta)$ and $\rho(X)\surd$. Then it must be the case that, for some c_1', $(X, c_1') \in \eta(c_1)$ and $d = c_1' \backslash \beta$. Let $\alpha \in \Lambda$. We proceed by distinguishing two possibilities, depending on whether $\alpha = \beta$ or $\alpha \neq \beta$.

Assume, first of all, that $\alpha = \beta$. Then it is easy to see that

$$\begin{aligned} new\ \beta(c_1 \backslash \beta)\rho &= new\ \beta c_1 \rho \\ &= new\ \beta c_1' \rho \qquad \text{by induction} \\ &= new\ \beta(c_1' \backslash \beta)\rho. \end{aligned}$$

If $\alpha \neq \beta$ then, by the definition of *new* it is easy to see that

$$\begin{aligned} new\ \alpha(c_1 \backslash \beta)\rho &= (new\ \alpha c_1 \rho) - \bigcup \{FC(\rho(e)) \mid e \in Sym(c),\ v(e) = \beta\} \\ &= (new\ \alpha c_1' \rho) - \bigcup \{FC(\rho(e)) \mid e \in Sym(c_1),\ v(e) = \beta\} \\ &\qquad \text{by induction} \\ &= new\ \alpha(c_1' \backslash \beta)\rho. \quad \blacksquare \end{aligned}$$

The above proposition will be very useful in proving that, for all $c, d \in \mathbf{L}\mathscr{S}$, whenever $(X, d) \in \eta(c)$ and $\rho(X)\surd$, $c\rho$ "behaves like" $d\rho$. In fact, we shall now show that, under the above assumptions, there is a close syntactic relation between $c\rho$ and $d\rho$; namely, $c\rho$ and $d\rho$ are syntactically the "same process" up to renaming of successfully terminated processes.

DEFINITION 4.5. $=_{\surd}$ denotes the least congruence over $\mathbf{L}\mathscr{S}$ which satisfies the following axioms:

$$\begin{aligned} &(TER1) && nil; nil = nil \\ &(TER2) && nil \mid nil = nil \\ &(TER3) && nil + nil = nil \\ &(TER4) && nil \backslash \alpha = nil. \end{aligned}$$

The following lemma is easily established by induction on the relation $\surd$.

LEMMA 4.5. *For each $c \in \mathbf{L}\mathscr{S}$, $c\surd$ implies $c =_{\surd} nil$.*

Using $=_{\surd}$, we are now able to formalize the syntactic relationship between $c\rho$ and $d\rho$ whenever $(X, d) \in \eta(c)$ and $\rho(X)\surd$.

PROPOSITION 4.2. *Let $c \in \mathbf{L}\mathscr{S}$ and ρ be a labelled action refinement compatible with c. Assume that $(X, d) \in \eta(c)$ and $\rho(X)\surd$. Then the following statements hold:*

(1) *$id(c\rho) = id(d\rho)$ and*

(2) *$c\rho =_{\surd} d\rho$.*

Proof. Both statements can be shown by structural induction on c. We only give the proof of the case $c = c_1\backslash\alpha$ for statement (2).

• $c = c_1\backslash\alpha$. Assume that $(X, d) \in \eta(c_1\backslash\alpha)$ and $\rho(X)\surd$. Then, by the definition of η, it must be the case that $d = c_1'\backslash\alpha$ and $(X, c_1') \in \eta(c_1)$, for some c_1'. Now, letting $\beta = choice(new\ \alpha c_1\rho)$,

$$\begin{aligned}(c_1\backslash\alpha)\rho &= (c_1\rho[\alpha \mapsto \beta])\backslash\beta \\ &=_{\surd} (c_1'\rho[\alpha \mapsto \beta])\backslash\beta \qquad \text{by the inductive hypothesis,}\end{aligned}$$

which is applicable because, as for each $F(a_i) \in X$ we have that $a \neq \alpha, \bar{\alpha}$, $\rho[\alpha \mapsto \beta](X) = \rho(X)\surd$. By Lemma 4.4, $new\ \alpha(c_1\backslash\alpha)\rho = new\ \alpha(c_1'\backslash\alpha)\rho$. It is now easy to see that $(c_1'\rho[\alpha \mapsto \beta])\backslash\beta = (c_1'\backslash\alpha)\rho$. ∎

As we have seen, for each $(X, d) \in \eta(c)$ and labelled action refinement ρ such that $\rho(X)\surd$, the relationship between $d\rho$ and $c\rho$ can be expressed syntactically in terms of $=_{\surd}$. However, in what follows, we shall often need to use $=_{\surd}$ in conjunction with the relation of α-congruence and a behavioural characterization of $=_{\surd}$ would be of considerable help. We already know that $=_{\alpha}$ is a strong bisimulation over the LTS $\langle \mathbf{L}\mathcal{S}, \mapsto, LAct_s(\Lambda), \surd \rangle$. As it may be expected, $=_{\surd}$ will also turn out to be a strong bisimulation and this will allow us to exploit standard properties of $\sim$ in reasoning about the syntactic relations $=_{\alpha}$ and $=_{\surd}$. The reader may easily establish that:

PROPOSITION 4.3. *$=_{\surd}$ is a strong bisimulation over the LTS $\langle \mathbf{L}\mathcal{S}, \mapsto, LAct_s(\Lambda), \surd \rangle$.*

We can now prove the promised "whence theorem" for moves of configurations of the form $c\rho$. Intuitively, if $c\rho \stackrel{e}{\longmapsto} \bar{c}$ then one of the following situations occurs:

• c is capable of starting action a_i and the component of ρ corresponding to a_i, $\rho(a_i)$, is capable of performing e. From that moment onwards, $F(a_i)$ acts as a place-holder for what remains to be executed of the process $\rho(a_i)$; or

• the move is due to one of the components of ρ of the form $\rho(F(a_i))$, or

• there is a pair (X, d) associated to c by η with the property that ρ maps all the places in X to successfully terminated processes and $d\rho$ is capable of performing e, or

• $e = \tau$ and the move is due to an internal move of c, or

• $e = \tau$ and the move is generated by a synchronization between different components of ρ. (Clauses (3), (6) and (7) in the statement of the theorem below cater for all the possible interactions of this kind.)

Formally:

THEOREM 4.2 ("Whence Theorem" for Configurations). *Let* $c \in \mathbf{L}\mathscr{S}$, $e \in LAct_s(\Lambda)$, *and* ρ *be a labelled action refinement compatible with* c. *Then* $c\rho \overset{e}{\longmapsto} \bar{c}$ *implies that*

(1) *there exist* $c', x \in \mathbf{L}\mathscr{S}$ *and* $a_i \in LAct(\Lambda)$ *such that* $c \overset{S(a_i)}{\longmapsto} c'$, $\rho(a_i) \overset{e}{\longmapsto} x$ *and* $\bar{c} =_\alpha c'\rho[F(a_i) \to x]$, *or*

(2) $e = \tau$ *and there exist* $a_i, b_j \in LAct(\Lambda)$, $\alpha \in \Lambda$ *and* $c', d, x, y \in \mathbf{L}\mathscr{S}$ *such that* $a_i \neq b_j$, $c \overset{S(a_i)}{\longmapsto} c' \overset{S(b_j)}{\longmapsto} d$, $\rho(a_i) \overset{\alpha_h}{\longmapsto} x$, $\rho(b_j) \overset{\bar{\alpha}_k}{\longmapsto} y$ *and* $\bar{c} =_\alpha d\rho[F(a_i) \to x, F(b_j) \to y]$, *or*

(3) *there exist* $a_i \in LAct(\Lambda)$ *and* $x \in \mathbf{L}\mathscr{S}$ *such that* $F(a_i)$ *occurs in* c, $\rho(F(a_i)) \overset{e}{\longmapsto} x$ *and* $\bar{c} =_\alpha c\rho[F(a_i) \to x]$, *or*

(4) $e = \tau$ *and* $c \overset{\tau}{\longmapsto} c'$ *for some* c' *such that* $\bar{c} =_\alpha c'\rho$, *or*

(5) $e = \tau$ *and there exist* $a_i, b_j \in LAct(\Lambda)$, $a' \in V(\Lambda)$ *and* $x, y \in \mathbf{L}\mathscr{S}$ *such that* $c \overset{S(a_i)}{\longmapsto} c'$, $F(b_j)$ *occurs in* c, $\rho(a_i) \overset{a'_h}{\longmapsto} x$, $\rho(F(b_j)) \overset{\bar{a}'_k}{\longmapsto} y$ *and* $\bar{c} =_\alpha c'\rho[F(a_i) \to x, F(b_j) \to y]$, *or*

(6) $e = \tau$ *and there exist* $a_i, b_j \in LAct(\Lambda)$, $x, y \in \mathbf{L}\mathscr{S}$ *such that* $a_i \neq b_j$, $F(a_i)$ *and* $F(b_j)$ *occur in* c, $\rho(F(a_i)) \overset{\alpha_h}{\longmapsto} x$, $\rho(F(b_j)) \overset{\bar{\alpha}_k}{\longmapsto} y$ *and* $\bar{c} =_\alpha c\rho[F(a_i) \to x, F(b_j) \to y]$, *or*

(7) *there exists* $(X, c') \in \eta(c)$ *such that* $X \neq \varnothing$, $\rho(X)\surd$ *and* $c'\rho \overset{e}{\longmapsto} \bar{c}$.

Proof. By structural induction on c. The proof is very similar to the one of Theorem 4.1. Its only new feature is the use of clause (2) in dealing with moves of configurations of the form $c\rho; \pi\rho$ in which $c\rho\surd$, but not $c\surd$. The details of the proof are omitted. ∎

The "whence theorems" proven above give us a complete picture of the possible origin of moves of labelled processes and states of the form $\pi\rho$ and $c\rho$, respectively. We also need a "converse theorem." More precisely, we shall need results that allow us to derive moves of $c\rho$ from those of a labelled state c and of the components of a labelled action refinement ρ. This information will be provided by the following "whither theorem." An example of such a result would be:

$$c \overset{\tau}{\Longmapsto} d \text{ implies } c\rho \overset{\tau}{\Longmapsto} d\rho.$$

We shall show that this is indeed true if we work up to strong bisimulation equivalence. In the proof of the "whither theorem" extensive use is made of the fact that both $=_\alpha$ and $=_\surd$ are strong bisimulations over the LTS $\langle \mathbf{L}\mathscr{S}, \mapsto, LAct_s(\Lambda), \surd \rangle$.

The statement of the "whither theorem" is in terms of the weak moves, $\overset{e}{\Longmapsto}$, rather than the strong moves, $\overset{e}{\longmapsto}$. This is because, in proving the

refinement theorem, we shall be required to reconstruct a move of $c\rho$ from weak moves of c and ρ, whereas it is sufficient to decompose strong moves of $c\rho$ into strong moves of c and ρ. This is a natural consequence of the definition of a bisimulation where strong moves are matched by weak ones. The "whither theorem" provides a converse for each possibility which arises in the "whence theorem" except the last one, which deals with $F(a_i)$'s instantiated with terminated processes. In the refinement theorem this case will be handled by induction.

THEOREM 4.3 (Whither Theorem). *Let $c \in \mathbf{L}\mathcal{S}$ and let ρ be a labelled action refinement compatible with it. Then*:

(1) $c \stackrel{S(a_i)}{\Longrightarrow} c'$ *and* $\rho(a_i) \stackrel{e}{\Longrightarrow} x$ *imply* $c\rho \stackrel{e}{\Longrightarrow} \sim c'\rho[F(a_i) \to x]$.

(2) *Assume that* $a_i \neq b_j$, $c \stackrel{S(a_i)}{\Longrightarrow} c' \stackrel{S(b_j)}{\Longrightarrow} d$, $\rho(a_i) \stackrel{x_k}{\Longrightarrow} x$, *and* $\rho(b_j) \stackrel{x_k}{\Longrightarrow} y$. *Then* $c\rho \stackrel{\tau}{\Longrightarrow} \sim d\rho[F(a_i) \to x, F(b_j) \to y]$.

(3) *Assume that* $F(a_i)$ *occurs in* c *and* $\rho(F(a_i)) \stackrel{e}{\Longrightarrow} x$. *Then* $c\rho \stackrel{e}{\Longrightarrow} \sim c\rho[F(a_i) \to x]$.

(4) $c \stackrel{\tau}{\Longrightarrow} d$ *implies* $c\rho \stackrel{\tau}{\Longrightarrow} \sim d\rho$.

(5) *Assume that* $c \stackrel{S(a_i)}{\Longrightarrow} c'$, $F(b_j)$ *occurs in* c, $\rho(a_i) \stackrel{a'_k}{\Longrightarrow} x$ *and* $\rho(F(b_j)) \stackrel{\bar{a}'_k}{\Longrightarrow} y$. *Then* $c\rho \stackrel{\tau}{\Longrightarrow} \sim c'\rho[F(a_i) \to x, F(b_j) \to y]$.

(6) *Assume that* $a_i \neq b_j$, $F(a_i)$ *and* $F(b_j)$ *occur in* c, $\rho(a_i) \stackrel{x_k}{\Longrightarrow} x$ *and* $\rho(F(b_j)) \stackrel{\bar{x}_k}{\Longrightarrow} y$. *Then* $c\rho \stackrel{\tau}{\Longrightarrow} \sim c\rho[F(a_i) \to x, F(b_j) \to y]$.

Proof. The proof of each of these results is relatively straightforward, but tedious. For example, the proof of (4) is very similar to that of Lemma 2.5. (We should point out that (4) is required in the proof of most of the other results.) Structural induction is used in proving (3) and (6) while the proofs of the remaining statements use induction on the length of the derivations involved. As an example, we briefly examine the proof of statement (5). The interested reader is referred to [1] for more details.

(5) By induction on the length of the derivation $c \stackrel{S(a_i)}{\Longrightarrow} c'$.

• Base case, $c \stackrel{S(a_i)}{\longmapsto} c'$. The proof proceeds by a subinduction on the length of the proof of the derivation $c \stackrel{S(a_i)}{\longmapsto} c'$. We examine only two of the possible cases, leaving the remaining ones to the reader.

— $c = c_1 | c_2 \stackrel{S(a_i)}{\longmapsto} c'_1 | c_2 = c'$ because $c_1 \stackrel{S(a_i)}{\longmapsto} c'_1$. We distinguish two possibilities depending on whether $F(b_j)$ occurs in c_1 or c_2.

If $F(b_j)$ occurs in c_1 then we may apply the subinductive hypothesis to obtain $c_1\rho \stackrel{\tau}{\Longrightarrow} \sim c'_1\rho[F(a_i) \to x, F(b_j) \to y]$. By the operational semantics and the substitutivity of $\sim$,

$$(c_1 | c_2)\rho = c_1\rho | c_2\rho \stackrel{\tau}{\Longrightarrow} \sim (c'_1\rho[F(a_i) \to x, F(b_j) \to y]) | c_2\rho.$$

Moreover, as $F(a_i)$ and $F(b_j)$ do not occur in c_2,

$$(c_1'\rho[F(a_i)\to x, F(b_j)\to y])\,|\,c_2\rho = (c_1'\,|\,c_2)\,\rho[F(a_i)\to x, F(b_j)\to y].$$

If $F(b_j)$ occurs in c_2 then, by statement (1), $c_1 \overset{S(a_i)}{\longmapsto} c'26_1$ and $\rho(a_i) \overset{a_h'}{\Longmapsto} x$ imply $c_1\rho \overset{a_h'}{\Longmapsto} \sim c_1'\rho[F(a_i)\to x]$. By statement (3), $F(b_j)$ occurs in c_2 and $\rho(F(b_j)) \overset{\bar a_k'}{\Longmapsto} y$ imply $c_2 \overset{a_k'}{\Longmapsto} c_2\rho[F(b_j)\to y]$. By the operational semantics and the substitutivity of $\sim$,

$$(c_1\,|\,c_2)\rho = c_1\rho\,|\,c_2\rho \overset{\tau}{\Longmapsto} \sim(c_1'\rho[F(a_i)\to x])\,|\,(c_2\rho[F(b_j)\to y]).$$

Moreover, as $F(a_i)$ and $F(b_j)$ do not occur in c_2 and c_1', respectively, we have that

$$(c_1'\rho[F(a_i)\to x])\,|\,(c_2\rho[F(b_j)\to y]) = (c_1'\,|\,c_2)\,\rho[F(a_i)\to x, F(b_j)\to y].$$

— $c = c_1\backslash\beta \overset{S(a_i)}{\longmapsto} c_1'\backslash\beta = c'$ because $c_1 \overset{S(a_i)}{\longmapsto} c_1'$ and $a \neq \beta, \bar\beta$. First of all, note that

$$(c_1\backslash\beta)\rho = (c_1\rho[\beta\mapsto\beta'])\backslash\beta',$$

where $\beta' = choice(new\ \beta c_1\rho)$. As $F(b_j)$ occurs in $c_1\backslash\beta$, by axiom (CR) we have that also $b \neq \beta, \bar\beta$. Moreover $F(b_j)$ occurs in c_1. By these observations we have that $\rho[\beta\mapsto\beta'](a_i) = \rho(a_i) \overset{a_h'}{\Longmapsto} x$ and $\rho[\beta\mapsto\beta'](F(b_j)) = \rho(F(b_j)) \overset{\bar a_k'}{\Longmapsto} y$. We may then apply the subinductive hypothesis to $c_1 \overset{S(a_i)}{\longmapsto} c_1'$ to obtain

$$c_1\rho[\beta\mapsto\beta'] \overset{\tau}{\Longmapsto} \sim c_1'\rho[\beta\mapsto\beta'][F(a_i)\to x, F(b_j)\to y].$$

By the operational semantics and the substitutivity of $\sim$, we then have that

$$(c_1\backslash\beta)\rho = (c_1\rho[\beta\mapsto\beta'])\backslash\beta' \overset{\tau}{\Longmapsto} \sim(c_1'\rho[\beta\mapsto\beta'][F(a_i)\to x, F(b_j)\to y])\backslash\beta'.$$

A simple case analysis of whether or not $\beta' = choice(new\ \beta c_1'\rho[F(a_i)\to x, F(b_j)\to y])$ will establish that

$$(c_1'\backslash\beta)\,\rho[F(a_i)\to x, F(b_j)\to y] =_\alpha (c_1'\hat\rho)\backslash\beta', \tag{5}$$

where $\hat\rho = \rho[\beta\mapsto\beta'][F(a_i)\to x, F(b_j)\to y]$. The claim then follows for the fact that $=_\alpha$ is a strong bisimulation and the transitivity of $\sim$.

• Inductive step, $c \overset{S(a_i)}{\Longmapsto} c'$ and $l(c \overset{S(a_i)}{\Longmapsto} c') > 1$. By Corollary 3.2, we may assume, w.l.o.g. that $c \overset{\varepsilon}{\Longmapsto} c'' \overset{S(a_i)}{\longmapsto} c'$ for some c''. By repeatedly applying statement (4), we have that $c \overset{\varepsilon}{\Longmapsto} c''$ implies $c\rho \overset{\varepsilon}{\Longmapsto} \sim c''\rho$ while the base case, applied to $c'' \overset{S(a_i)}{\longmapsto} c'$, gives $c''\rho \overset{\tau}{\Longmapsto} \sim c'\rho[F(a_i)\to x, F(b_j)\to y]$.

By combining these two derivations we then have that $c\rho \stackrel{\tau}{\Longrightarrow} \sim c'\rho[F(a_i) \to x, F(b_j) \to y]$. ∎

As a useful corollary of the whither theorem we have the following result, which will find application in the proof of Theorem 4.4.

COROLLARY 4.1. *Let $c, d \in \mathbf{L}\mathcal{S}$ and ρ be a labelled action refinement compatible with c. Then $c \stackrel{F(a_i)}{\Longrightarrow} d$ and $\rho(F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x\surd$ imply $c\rho \stackrel{\varepsilon}{\Longrightarrow} \sim d\rho$.*

Proof. Assume that $c \stackrel{F(a_i)}{\Longrightarrow} d$ and $\rho(F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x\surd$. By Corollary 3.2, we may assume, without loss of generality, that, for some $\bar{d}$, $c \stackrel{F(a_i)}{\longmapsto} \bar{d} \stackrel{\varepsilon}{\Longrightarrow} d$. It is now sufficient to prove that

$$c \stackrel{F(a_i)}{\longmapsto} \bar{d} \text{ and } \rho(F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x\surd \text{ imply } c\rho \stackrel{\varepsilon}{\Longrightarrow} \sim \bar{d}\rho. \tag{6}$$

In fact, by using (6), we have that $c\rho \stackrel{\varepsilon}{\Longrightarrow} \sim \bar{d}\rho$. By repeated application of statement (4) of the whither theorem, $\bar{d} \stackrel{\varepsilon}{\Longrightarrow} d$ implies $\bar{d}\rho \stackrel{\varepsilon}{\Longrightarrow} \sim d\rho$. It is now easy to see that $c\rho \stackrel{\varepsilon}{\Longrightarrow} \sim \bar{d}\rho$ and $\bar{d}\rho \stackrel{\varepsilon}{\Longrightarrow} \sim d\rho$ imply that

$$c\rho \stackrel{\varepsilon}{\Longrightarrow} \sim d\rho.$$

We are thus left to prove (6). The proof of this statement is by induction on the length of the proof of the derivation $c \stackrel{F(a_i)}{\longmapsto} \bar{d}$. We examine only two of the cases which arise in such a proof and leave the remaining ones to the reader.

- $c = F(a_i) \stackrel{F(a_i)}{\longmapsto} nil = \bar{d}$. Then $c\rho = \rho(F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x \sim nil$, because $x\surd$ by the assumptions of the statement.
- $c = c_1\backslash\alpha \stackrel{F(a_i)}{\longmapsto} d'\backslash\alpha = \bar{d}$ because $c_1 \stackrel{F(a_i)}{\longmapsto} d'$ and $a \neq \alpha, \bar{\alpha}$. First of all, recall that

$$(c_1\backslash\alpha)\rho = (c_1\rho[\alpha \mapsto \beta])\backslash\beta,$$

where $\beta = choice(new\ \alpha c_1\rho)$. As $a \neq \alpha, \bar{\alpha}$, we have that $\rho[\alpha \mapsto \beta](F(a_i)) = \rho(F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x\surd$. We may then apply the inductive hypothesis to $c_1 \stackrel{F(a_i)}{\longmapsto} d'$ and $\rho[\alpha \mapsto \beta](F(a_i)) \stackrel{\varepsilon}{\Longrightarrow} x\surd$ to obtain that, for some y,

$$c_1\rho[\alpha \mapsto \beta] \stackrel{\varepsilon}{\Longrightarrow} y \sim d'\rho[\alpha \mapsto \beta].$$

By the operational semantics and the substitutivity of $\sim$,

$$(c_1\backslash\alpha)\rho = (c_1\rho[\alpha \mapsto \beta])\backslash\beta \stackrel{\varepsilon}{\Longrightarrow} y\backslash\beta \sim (d'\rho[\alpha \mapsto \beta])\backslash\beta.$$

As $\beta \in new\ \alpha c_1\rho$, it follows that $\beta \in new\ \alpha d'\rho$. It is now easy to see that $(d'\backslash\alpha)\rho =_\alpha (d'\rho[\alpha \mapsto \beta])\backslash\beta$. The claim now follows from the fact that $=_\alpha$ is a strong bisimulation and from the transitivity of $\sim$. ∎

The whence and the whither theorems will enable us to prove a labelled version of the refinement theorem which states that $\pi \approx_{\varnothing} \pi'$ implies $\pi\rho \approx_{\varnothing} \pi'\rho$ for any labelled action refinement compatible with both π and π'. However, we shall have to prove a much more general statement because, as $\pi\rho$ and $\pi'\rho$ evolve, the labelled processes π and π' evolve to labelled states and the two copies of ρ may evolve into different labelled action refinements. However, these two copies must remain compatible with each other, at least up to refine equivalence. The precise form of this compatibility is captured in the following technical definition, which may be viewed as a pointwise extension of $\approx_{\mathscr{H}}$ "modulo a history ϕ." Intuitively, for two labelled action refinements ρ and ρ' to be "equivalent modulo ϕ," we shall require that, for each action a, they map occurrences of a to congruent processes and whenever $F(a_i)$ and $F(a_j)$ are related to each other by ϕ, $\rho(F(a_i))$ and $\rho'(F(a_j))$ are equivalent with respect to some history $\varphi(a, i, j)$. For use in the proof of the refinement theorem, we shall collect the histories $\varphi(a, i, j)$ for $(a, i, j) \in \phi$, in a *ϕ-vector* σ. Formally:

DEFINITION 4.6. (1) For each $h \in \mathscr{H}$, an *h-vector* σ is a map $\sigma: h \to \mathscr{H}$. For each $(a, i, j) \in h$, $\sigma^{(a,i,j)}$ will denote the history associated with it by σ.

(2) Let ρ and ρ' be two labelled action refinements. Then, for each $h \in \mathscr{H}$ and h-vector σ, $\rho =_{(h,\sigma)} \rho'$ iff for each $a \in V(\Lambda)$,

(a) $\rho(a_i) =_{\varnothing} \rho'(a_j)$, for all i, j,

(b) for each $i \in dom(h_a)$, $\rho(F(a_i)) \approx_{\sigma^{(a,i,j)}} \rho'(F(a_j))$, where $j = h_a(i)$.

Note that, in this definition, we require not that $\rho(a_i) \approx_{\varnothing} \rho(a_j)$, but the stronger property that $\rho(a_i) =_{\varnothing} \rho(a_j)$. We recall that this ensures that any τ-move performed by $\rho(a_i)$ must be matched by a corresponding τ-move from $\rho(a_j)$ rather than the empty move. This stronger property is required as we wish to prove that $c \approx_{\varphi} d$ and $\rho =_{(\varphi,\sigma)} \rho'$ imply $c\rho \approx_{\phi} d\rho'$, for an appropriate history ϕ. This would *not* be true if we simply required $\rho(a_i)$ and $\rho(a_j)$ to be equivalent. For example, let $\rho(a_1) = a_1$, $\rho(b_2) = b_2$, $\rho'(a_1) = \tau; a_1$ and $\rho'(b_2) = b_2$. Then $\rho(a_1) \approx_{\varnothing} \rho'(a_1)$, but not $(a_1 + b_2)\rho \approx_{\varnothing} (a_1 + b_2)\rho'$.

Part of the task of establishing a refine bisimulation is to compare the termination capabilities of the related processes. This is the subject of the following theorem which relates the termination capabilities of $c\rho$ and $d\rho'$, where $c \approx_{\phi} d$ and $\rho =_{(\phi,\sigma)} \rho'$ for some $\phi \in \mathscr{H}$ and ϕ-vector σ. Unfortunately, the proof of the theorem is rather involved because, in general, there is no simple relationship between the termination potential of a labelled state c and that of $c\rho$. For instance, it is easy to see that *not* $a_i + b_j \surd$, but $(a_i + b_j)\rho \surd$, for any ρ such that $\rho(a_i) = \tau$ and $\rho(b_j) = b_j$. On the other hand, $a_i + \tau \surd$, but *not* $(a_i + \tau)\rho \surd$ for any labelled action refinement ρ such that $\rho(a_i) = \tau; \delta$.

THEOREM 4.4. *Let* $c, d \in \mathbf{L}\mathcal{S}$, $\phi \in \mathcal{H}$ *and* ρ, ρ' *be labelled action refinements such that* $\rho =_{(\phi,\sigma)} \rho'$ *for some* ϕ*-vector* σ*. Assume that* $c \approx_\phi d$*. Then* $c\rho\not\!\!\!\downarrow$ *implies* $d\rho'\not\!\!\!\downarrow$.

Proof. By induction on the relation $\prec_r$ which is defined as follows:

For all $c_i, d_i \in \mathbf{L}\mathcal{S}$, $\phi_i \in \mathcal{H}$ and labelled action refinements ρ_i, ρ_i', $i = 1, 2$, we write $\langle c_1\rho_1, c_2\rho_2\rangle \prec_r \langle d_1\rho_1', d_2\rho_2'\rangle$ iff $c_1 \approx_{\phi_1} c_2$, $d_1 \approx_{\phi_2} d_2$ and there exist a ϕ_1-vector σ_1 and a ϕ_2-vector σ_2 such that $\rho_1 =_{(\phi_1,\sigma_1)} \rho_2$ and $\rho_1' =_{(\phi_2,\sigma_2)} \rho_2'$ and the following conditions hold,

(1) $|c_1| + |c_2| < |d_1| + |d_2|$ or

(2) $|c_1| + |c_2| = |d_1| + |d_2|$ and $|c_1\rho_1| + |c_2\rho 2| < |d_1\rho_1'| + |d_2\rho_2'|$,

where, for each $c \in \mathbf{L}\mathcal{S}$, $|c|$ denotes the depth of its derivation tree.

We assume, as inductive hypothesis, that the claim holds for all $\langle c_1\rho_1, c_2\rho_2\rangle \prec_r \langle c\rho, d\rho'\rangle$ and prove that it holds for $\langle c\rho, d\rho'\rangle$, where $c \approx_\phi d$ and $\rho =_{(\phi,\sigma)} \rho'$, for some ϕ-vector σ. Assume that $c\rho\not\!\!\!\downarrow$. We shall prove that $d\rho\not\!\!\!\downarrow$ by distinguishing two possibilities:

(A) $c\rho \overset{\tau}{\not\longmapsto}$ and $d\rho' \overset{\tau}{\not\longmapsto}$, i.e., both $c\rho$ and $d\rho'$ are *stable*, or

(B) either $c\rho$ or $d\rho'$ is not stable.

We examine the two possibilities in turn.

(A) Assume that $c\rho$ and $d\rho'$ are stable. As $c\rho\not\!\!\!\downarrow$ by the hypothesis of the theorem, we then have that $c\rho\surd$. We shall prove that $d\rho'\surd$. First of all, it is easy to check that $c\rho\surd$ implies that $\rho(F(a_i))\surd$ for all $F(a_i)$ occurring in c and that, for each $(X, c') \in \eta(c)$, $\rho(X)\surd$ and $c'\rho\surd$. Moreover, by the whither theorem, we have that $d\rho' \overset{\tau}{\not\longmapsto}$ implies

(1) $d \overset{\tau}{\not\longmapsto}$ and

(2) $\rho'(F(a_i)) \overset{\tau}{\not\longmapsto}$, for all $F(a_i)$ occurring in d'.

Let $(X, c') \in \eta(c)$. (Note that such a pair always exists because $\eta(c)$ is nonempty.) We distinguish two possibilities depending on whether X is empty or not.

If $X = \varnothing$ then, by Lemma 4.2(1), c is a labelled process. It is easy to see that, for labelled processes, $c\rho\surd$ implies that $c\surd$. As $c \approx_\phi d$ by the hypothesis of the theorem, we have that $d\not\!\!\!\downarrow$. Hence, by (A.1), $d\surd$ and this implies that $d\rho'\surd$.

Assume now that $X \neq \varnothing$. Then there exist $F(a_i) \in X$. By Lemma 4.2(3), there exists $\bar{c}$ such that $(\{F(a_i)\}, \bar{c}) \in \eta(c)$. By Lemma 4.3, $c \overset{F(a_i)}{\longmapsto} \bar{c}$. As $\rho(F(a_i))\surd$, we may apply Corollary 4.1 to obtain that

$$c\rho \overset{\varepsilon}{\Longmapsto} x \sim \bar{c}\rho, \text{ for some } x.$$

As $c\rho$ is stable, we have that $c\rho \sim \bar{c}\rho$ and $\bar{c}\rho\surd$. As $c \approx_\phi d$, we have that $c \xmapsto{F(a_i)} \bar{c}$ and d stable imply that $d \xmapsto{F(a_j)} d'$ for some j and d' such that $\phi_a(i) = j$ and $\bar{c} \approx_{\phi'} d'$, where $\phi' = \phi - \{(a, i, j)\}$. As $\rho =_{(\phi,\sigma)} \rho'$ by the hypotheses of the theorem and $\rho(F(a_i))\surd$, we have that $\rho'(F(a_j))\not\surd$. Moreover, by (A.2), it follows that $\rho'(F(a_j))\surd$. We may then apply Corollary 4.1 to obtain that, as $d\rho'$ is stable, $d\rho' \sim d'\rho'$. Note now that $\rho =_{(\phi,\sigma)} \rho'$ implies that $\rho =_{(\phi',\sigma')} \rho'$, where σ' is the restriction of σ to a ϕ'-vector. Moreover, as $|\bar{c}| + |d'| < |c| + |d|$, $\langle \bar{c}, d' \rangle \prec_r \langle c, d \rangle$. We may then apply the inductive hypothesis to $\bar{c} \approx_{\phi'} d'$, $\rho =_{(\phi',\sigma')} \rho'$ and $\bar{c}\rho\surd$ to obtain that $d'\rho'\not\surd$. As $d\rho' \sim d'\rho'$ and $d\rho$ is stable, we then derive that $d\rho'\surd$. This completes the proof of case (A).

(B) Without loss of generality, we restrict ourselves to examining the case in which $d\rho'$ is not stable. (The case in which $d\rho'$ is stable and $c\rho$ is not can be dealt with in a similar fashion.) In order to prove that $d\rho'\not\surd$, it is sufficient to show that $d\rho' \xmapsto{\tau} \bar{y} \overset{\bar{c}}{\Longmapsto} y$ and y stable imply $y\surd$. By Theorem 4.2 and the fact that $=_\alpha$ is a strong bisimulation, $d\rho' \xmapsto{\tau} \bar{y}$ implies

(1) there exist $a_i \in LAct(\Lambda)$ and $d', x \in \mathbf{L}\mathscr{S}$ such that $d \xmapsto{S(a_i)} d'$, $\rho'(a_i) \xmapsto{\tau} x$ and $\bar{y} \sim d'\rho'[F(a_i) \to x]$, or

(2) there exists $(X, d') \in \eta(d)$ such that $X \neq \emptyset$, $\rho'(X)\surd$ and $d'\rho' \xmapsto{\tau} \bar{y}$, or

(3) there exist $a_i, b_j \in LAct(\Lambda)$, $\alpha \in \Lambda$ and $\bar{d}, d', x_1, x_2 \in \mathbf{L}\mathscr{S}$ such that $d \xmapsto{S(a_i)} \bar{d} \xmapsto{S(b_j)} d'$, $\rho'(a_i) \xmapsto{\alpha_h} x_1$, $\rho'(b_j) \xmapsto{\bar{\alpha}_k} x_2$ and $\bar{y} \sim d'\rho'[F(a_i) \to x_1, F(b_j) \to x_2]$, or

(4) there exist $a_i \in LAct(\Lambda)$ and $x \in \mathbf{L}\mathscr{S}$ such that $F(a_i)$ occurs in d, $\rho'(F(a_i)) \xmapsto{\tau} x$ and $\bar{y} \sim d\rho'[F(a_i) \to x]$, or

(5) there exists $d' \in \mathbf{L}\mathscr{S}$ such that $d \xmapsto{\tau} d'$ and $\bar{y} \sim d'\rho'$, or

(6) there exist $a_i, b_j \in \mathbf{L}\mathscr{S}$ and $d', x_1, x_2 \in \mathbf{L}\mathscr{S}$ such that $d \xmapsto{S(a_i)} d'$, $F(b_j)$ occurs in d, $\rho'(a_i) \xmapsto{a'_h} x_1$, $\rho'(F(b_j)) \xmapsto{\bar{a}'_k} x_2$ and $\bar{y} \sim d'\rho'[F(a_i) \to x_1, F(b_j) \to x_2]$, or

(7) there exist $a_i, b_j \in LAct(\Lambda)$ and $x_1, x_2 \in \mathbf{L}\mathscr{S}$ such that $F(a_i), F(b_j)$ occur in d, $\rho'(F(a_i)) \xmapsto{\alpha_h} x_1$ and $\rho'(F(b_j)) \xmapsto{\bar{\alpha}_k} x_2$ and $\bar{y} \sim d\rho'[F(a_i) \to x_1, F(b_j) \to x_2]$.

We proceed by showing that $y\surd$ in each of the following possibilities. We shall only examine three representative cases, leaving the remaining ones to the reader.

(1) Assume that $d\rho' \xmapsto{\tau} \bar{y}$ because $d \xmapsto{S(a_i)} d'$, $\rho'(a_i) \xmapsto{\tau} x$ and $\bar{y} \sim d'\rho'[F(a_i) \to x]$. As $c \approx_\phi d$, we then have that $c \overset{S(a_{i_1})}{\Longmapsto} c'$ for some i_1 and c' such that $c' \approx_{\phi'} d'$, where $\phi' = \phi \cup \{(a, i_1, i)\}$. As $\rho =_{(\phi,\sigma)} \rho'$, we have

that $\rho(a_{i_1}) =_{\varnothing} \rho'(a_i)$. Hence there exists x' such that $\rho(a_{i_1}) \overset{\tau}{\Longrightarrow} x'$ and $x' \approx_{\varnothing} x$. By statement (1) of the whither theorem, $c \overset{S(a_{i_1})}{\Longrightarrow} c'$ and $\rho(a_{i_1}) \overset{\tau}{\Longrightarrow} x'$ imply $c\rho \overset{\tau}{\Longrightarrow} \bar{x} \sim c'\rho[F(a_{i_1}) \to x']$, for some $\bar{x}$. As $c\rho\not\!\!\downarrow$ we then have that $c'\rho[F(a_{i_1}) \to x']\not\!\!\downarrow$.

It is easy to see that, as $\rho =_{(\phi,\sigma)} \rho'$, we have that $\rho[F(a_{i_1}) \to x'] =_{(\phi',\sigma')} \rho'[F(a_i) \to x]$, where σ' is the ϕ'-vector given by

$$\sigma'^{(b,h,k)} =_{\text{def}} \begin{cases} \varnothing & \text{if } (b,h,k) = (a, i_1, i) \\ \sigma^{(b,h,k)} & \text{otherwise.} \end{cases}$$

As $|c'| + |d'| < |c| + |d|$, we may now apply the inductive hypothesis to $c' \approx_{\phi'} d'$, $\rho[F(a_{i_1}) \to x'] =_{(\phi',\sigma')} \rho'[F(a_i) \to x]$ and $c'\rho[F(a_{i_1}) \to x']\not\!\!\downarrow$ to obtain that $d'\rho'[F(a_i) \to x]\not\!\!\downarrow$. As $\bar{y} \sim d'\rho'[F(a_i) \to x]$, we then have that $\bar{y}\not\!\!\downarrow$. This implies that $y\surd$.

(2) Assume that $d\rho' \overset{\tau}{\longmapsto} \bar{y}$ because there exists $(X, d') \in \eta(c)$ such that $X \neq \varnothing$, $\rho'(X)\surd$ and $d'\rho' \overset{\tau}{\longmapsto} \bar{y}$. As $X \neq \varnothing$, there exists $F(a_j) \in X$. By Lemma 4.2(3), $(\{F(a_j)\}, \bar{d}) \in \eta(d)$ for some $\bar{d}$ and, as $\rho'(X)\surd$, $\rho'(F(a_j))\surd$. By Proposition 4.2, we have that

$$d\rho' =_{\surd} d'\rho' =_{\surd} \bar{d}\rho'.$$

Moreover, by Lemma 4.3, $d \overset{F(a_j)}{\longmapsto} \bar{d}$. As $c \approx_{\phi} d$, there exist i and c' such that $c \overset{F(a_i)}{\Longrightarrow} c'$, $\phi_a(i) = j$, and $c' \approx_{\phi'} \bar{d}$, where $\phi' = \phi - \{(a, i, j)\}$. As $\rho =_{(\phi,\sigma)} \rho'$ and $\rho'(F(a_j))\surd$, we have that

- $\rho(F(a_i)) \approx_{\varnothing} \rho'(F(a_j))$ and $\rho(F(a_j))\not\!\!\downarrow$, and
- $\rho =_{(\phi',\sigma')} \rho'$, where σ' is the restriction of σ to a ϕ'-vector.

As $\rho(F(a_i))\not\!\!\downarrow$, there exists x such that $\rho(F(a_i)) \overset{\varepsilon}{\Longrightarrow} x\surd$. By Corollary 4.1, $c \overset{F(a_i)}{\Longrightarrow} c'$ and $\rho(F(a_i)) \overset{\varepsilon}{\Longrightarrow} x\surd$ imply $c\rho \overset{\varepsilon}{\Longrightarrow} \sim c'\rho$. As $c\rho\not\!\!\downarrow$, we have that $c'\rho\not\!\!\downarrow$. We may now apply the inductive hypothesis to $c' \approx_{\phi'} \bar{d}$, $\rho =_{(\phi',\sigma')} \rho'$ and $c'\rho\not\!\!\downarrow$ to obtain that $\bar{d}\rho'\not\!\!\downarrow$. As $d\rho' =_{\surd} \bar{d}\rho'$ and $=_{\surd}$ is a strong bisimulation we then have that $d\rho'\not\!\!\downarrow$. Hence $y\surd$.

(4) Assume that $d\rho' \overset{\tau}{\longmapsto} \bar{y}$ because $F(a_i)$ occurs in d, $\rho'(F(a_i)) \overset{\tau}{\longmapsto} x$ and $\bar{y} \sim d\rho'[F(a_i) \to x]$. Then, as $c \approx_{\phi} d$, there exists j such that $\phi_a(j) = i$ and $F(a_j)$ occurs in c. Moreover, as $\rho =_{(\phi,\sigma)} \rho'$, we have that $\rho(F(a_j)) \approx_{\sigma^{(a,j,i)}} \rho'(F(a_i))$. Thus there exists x' such that $\rho(F(a_j)) \overset{\varepsilon}{\Longrightarrow} x'$ and $x' \approx_{\sigma^{(a,j,i)}} x$. By statement (3) of the whither theorem, we have that $c\rho \overset{\varepsilon}{\Longrightarrow} \sim c\rho[F(a_j) \to x']$. As $c\rho\not\!\!\downarrow$, $c\rho[F(a_j) \to x']\not\!\!\downarrow$. Moreover, as $\rho =_{(\phi,\sigma)} \rho'$, we have that $\rho[F(a_j) \to x'] =_{(\phi,\sigma)} \rho'[F(a_i) \to x]$. Note now that, as $|d\phi'[F(a_i) \to x]| < |d\rho'|$,

$$|c\rho[F(a_j) \to x']| + |d\rho'[F(a_i) \to x]| < |c\rho| + |d\rho'|.$$

We may then apply the inductive hypothesis to obtain that $d\rho'[F(a_i) \to x]\not\surd$. As $\bar{y} \sim d\rho'[F(a_i) \to x]$, we then have that $\bar{y}\not\surd$. This implies that $y\surd$.

The proof of (B) can be completed by checking the remaining cases in a similar way. ∎

We are now ready to prove the labelled version of the refinement theorem.

DEFINITION 4.7. For each $\phi \in \mathscr{H}$, let $\mathscr{R}_\phi^{\text{sub}}$ be defined as follows:

$$\mathscr{R}_\phi^{\text{sub}} =_{\text{def}} \left\{ \langle c\rho, d\rho' \rangle \;\middle|\; \begin{array}{l} \text{(i) there exists } \varphi \in \mathscr{H} \text{ such that } c \approx_\varphi d, \\ \text{(ii) } c\rho, d\rho' \in \mathbf{L}\mathscr{S}, \text{ and} \\ \text{(iii) there exists a } \varphi\text{-vector } \sigma \text{ such that } \rho =_{(\varphi,\sigma)} \rho' \text{ and} \\ \qquad \phi = \bigcup_{(a,i,j) \in \varphi} \sigma^{(a,i,j)} \end{array} \right\}.$$

THEOREM 4.5 (Labelled Refinement Theorem). *For each* $\phi \in \mathscr{H}$, $\langle c\rho, d\rho' \rangle \in \mathscr{R}_\phi^{\text{sub}}$ *implies* $c\rho \approx_\phi d\rho'$.

Proof. It will be convenient to prove a slightly more complicated statement. For each $\phi \in \mathscr{H}$, let $\mathscr{S}_\phi$ be given by

$$\mathscr{S}_\phi = \{ \langle c\rho, d\rho' \rangle \mid c\rho \sim x \text{ and } y \sim d\rho' \text{ for some } \langle x, y \rangle \in \mathscr{R}_\phi^{\text{sub}} \}.$$

We show that $\{\mathscr{S}_\phi \mid \phi \in \mathscr{H}\}$ is a weak refine bisimulation; i.e., we work "up to strong bisimulation." The claim will then follow because, for each ϕ, $\mathscr{R}_\phi^{\text{sub}} \subseteq \mathscr{S}_\phi$ by the definition of $\mathscr{R}_\phi^{\text{sub}}$ and the reflexivity of $\sim$.

It is straightforward, but tedious, to check that $\{\mathscr{S}_\phi \mid \phi \in \mathscr{H}\}$ is a symmetric family of relations and, moreover, that the compatibility requirements are met. So it remains to show that moves of $c\rho$ may be matched by corresponding moves of $d\rho'$. This we prove by induction on the combined size of c and d. We shall limit ourselves to proving the claim for $\langle c\rho, d\rho' \rangle \in \mathscr{R}_\phi^{\text{sub}}$. The more general claim will then follow immediately from the definitions of $\sim$ and $\mathscr{R}_\phi^{\text{sub}}$. The general nature of this argument is to use the composition/decomposition results. For example, given $\langle c\rho, d\rho' \rangle \in \mathscr{R}_\phi^{\text{sub}}$ and a move of $c\rho$, we shall use Theorem 4.2 to decompose this move into a move from c and move(s) of the components of ρ. The equivalence of c and d with respect to some history and the pointwise equivalence between ρ and ρ' will then be used to derive "matching moves" from d and the components of ρ'. These moves will then be composed by means of the statements of the "whither theorem" to obtain a move of $d\rho'$,

which will be shown to match the original move of $c\rho$ with respect to $\{\mathscr{S}_\phi \mid \phi \in \mathscr{H}\}$.

Assume now that $\langle c\rho, d\rho' \rangle \in \mathscr{R}_\phi^{\text{sub}}$. Then, by the definition of $\mathscr{R}_\phi^{\text{sub}}$, we have that

(A) $c \approx_\varphi d$, for some φ,

(B) $c\rho, d\rho' \in \mathbf{L}\mathscr{S}$, and

(C) there exists a φ-vector σ such that $\rho =_{(\varphi,\sigma)} \rho'$ and

$$\phi = \bigcup_{(a,i,j)\in\varphi} \sigma^{(a,i,j)}.$$

We shall now show that the defining clauses of $\approx_{\mathscr{H}}$ are met by $\langle c\rho, d\rho' \rangle \in \mathscr{R}_\phi^{\text{sub}}$. We shall limit ourselves to giving the proof for two of the clauses.

• Assume that $c\rho \xmapsto{S(a_i)} \bar{c}$. We shall prove that $d\rho' \xmapsto{S(a_i)} \bar{d}$, for some $\bar{d}$ and j such that $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi \cup \{(a,i,j)\}}$. By Theorem 4.2, $c\rho \xmapsto{S(a_i)} \bar{c}$ implies that

(1.a) there exist b_n, c' and x such that $c \xmapsto{S(b_n)} c'$, $\rho(b_n) \xmapsto{S(a_i)} x$ and $\bar{c} \sim c'\rho[F(b_n) \to x]$, or

(1.b) there exists $(X, c') \in \eta(c)$ such that $X \neq \varnothing$, $\rho(X)\surd$ and $c'\rho \xmapsto{S(a_i)} \bar{c}$, or

(1.c) there exist b_n and x such that $F(b_n)$ occurs in c, $\rho(F(b_n)) \xmapsto{S(a_i)} x$ and $\bar{c} \sim c\rho[F(b_n) \to x]$.

(None of the other possibilities in Theorem 4.2 applies to start-moves.)

We proceed by examining each possibility in turn.

(1.a) Assume that there exist b_n, c' and x such that $c \xmapsto{S(b_n)} c'$, $\rho(b_n) \xmapsto{S(a_i)} x$ and $\bar{c} \sim c'\rho[F(b_n) \to x]$. Then, as $c \approx_\varphi d$ by (A), we have that $d \xRightarrow{S(b_m)} d'$ for some m and d' such that $c' \approx_{\varphi'} d'$, where $\varphi' = \varphi \cup \{(b, n, m)\}$. As $\rho =_{(\varphi,\sigma)} \rho'$ by (C), we have that $\rho(b_n) =_\varnothing \rho'(b_m)$. Thus there exist j and y such that $\rho'(b_m) \xmapsto{S(a_j)} y$ and $x \approx_{\{(a,i,j)\}} y$. By statement (1) of Theorem 4.3, $d \xRightarrow{S(b_m)} d'$ and $\rho'(b_m) \xRightarrow{S(a_j)} y$ imply that

$$d\rho' \xmapsto{S(a_j)} \bar{d} \sim d'\rho'[F(b_m) \to y],$$

for some $\bar{d}$. It remains to be checked that $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi \cup \{(a,i,j)\}}$. It is sufficient to prove that

$$\langle c'\rho[F(b_n) \to x], d'\rho'[F(b_m) \to y] \rangle \in \mathscr{R}^{\text{sub}}_{\phi \cup \{(a,i,j)\}}.$$

We have already seen that $c' \approx_{\varphi'} d'$, where $\varphi' = \varphi \cup \{(b, n, m)\}$. Consider now the φ'-vector σ_1 given by

$$\sigma_1^{(b',l,l')} = \begin{cases} \{(a, i, j)\} & \text{if } (b', l, l') = (b, n, m) \\ \sigma^{(b',l,l')} & \text{otherwise.} \end{cases}$$

As $\rho =_{(\varphi,\sigma)} \rho'$ and $x \approx_{\{(a,i,j)\}} y$, it is easy to see that

$$\rho[F(b_n) \to x] =_{(\varphi',\sigma_1)} \rho'[F(b_m) \to y].$$

Moreover, we have that

$$\bigcup_{(b',l,l') \in \varphi'} \sigma_1^{(b',l,l')} = \left(\bigcup_{(b',l,l') \in \varphi \quad \{(b,n,m)\}} \sigma^{(b',l,l')} \right) \cup \sigma_1^{(b,n,m)}$$
$$= \phi \cup \{(a, i, j)\} \qquad \text{by (C) and } \sigma_1^{(b,n,m)} = \{(a, i, j)\}.$$

Hence we have shown that $\langle c'\rho[F(b_n) \to x],\ d'\rho'[F(b_m) \to y] \rangle \in \mathscr{R}^{\text{sub}}_{\phi \cup \{(a,i,j)\}}$. This completes the verification of subcase (1.a).

(1.b) Assume that there exists $(X, c') \in \eta(c)$ such that $X \neq \varnothing$, $\rho(X)\surd$, and $c'\rho \xmapsto{S(a_i)} \bar{c}$. Then, by Proposition 4.2, $c\rho =_{\surd} c'\rho$. As $=_{\surd}$ is a strong bisimulation, we have that $c\rho \sim c'\rho$. By the definition of $\mathscr{S}_\phi$, we have that $\langle c'\rho, d\rho' \rangle \in \mathscr{S}_\phi$. As the combined size of c' and d is less than that of c and d, we may apply the inductive hypothesis to obtain that $d\rho' \overset{S(a_j)}{\Longmapsto} \bar{d}$ for some $\bar{d}$ such that $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi \cup \{(a,i,j)\}}$.

(1.c) The proof of this subcase is similar to that of (2.b) below and is thus omitted.

• Assume that $c\rho \xmapsto{F(a_i)} \bar{c}$. We shall prove that $d\rho' \overset{F(a_j)}{\Longmapsto} \bar{d}$, for some $\bar{d}$ and j such that $\phi_a(i) = j$ and $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi \quad \{(a,i,j)\}}$. By Theorem 4.2, $c\rho \xmapsto{F(a_i)} \bar{c}$ implies that

(2.a) there exists $(X, c') \in \eta(c)$ such that $X \neq \varnothing$, $\rho(X)\surd$ and $c'\rho \xmapsto{F(a_i)} \bar{c}$, or

(2.b) there exist a'_n and x such that $F(a'_n)$ occurs in c, $\rho(F(a'_n)) \xmapsto{F(a_i)} x$ and $\bar{c} \sim c\rho[F(a'_n) \to x]$.

(None of the other possibilities in Theorem 4.2 applies to end-moves).

We proceed by examining the two possibilities separately.

(2.a) Assume that there exists $(X, c') \in \eta(c)$ such that $X \neq \varnothing$, $\rho(X)\surd$ and $c'\rho \xmapsto{F(a_i)} \bar{c}$. Then, by Proposition 4.2, $c\rho =_{\surd} c'\rho$. As $=_{\surd}$ is a strong bisimulation, we have that $c\rho \sim c'\rho$. By the definition of $\mathscr{S}_\phi$, we have that $\langle c'\rho, d\rho' \rangle \in \mathscr{S}_\phi$. As the combined size of c' and d is less than that of c and d, we may apply the inductive hypothesis to obtain that $d\rho' \overset{F(a_j)}{\Longmapsto} \bar{d}$ for some $\bar{d}$ such that $\phi_a(i) = j$ and $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi - \{(a,i,j)\}}$.

(2.b) Assume that there exist a'_n and x such that $F(a'_n)$ occurs in c, $\rho(F(a'_n)) \xmapsto{F(a_i)} x$, and $\bar{c} \sim c\rho[F(a'_n) \to x]$. As $c \approx_\varphi d$ by (A) and $F(a'_n)$ occurs in c, we have that $F(a'_m)$ occurs in d, with $m = \phi_a(n)$. As $\rho =_{(\varphi,\sigma)} \rho'$ by (C), we have that $\rho(F(a'_n)) \approx_{\sigma^{(a',n,m)}} \rho'(F(a'_m))$. Thus $\rho(F(a'_n)) \xmapsto{F(a_i)} x$ implies that $\rho'(F(a'_m)) \xRightarrow{F(a_j)} y$, for some y such that $\sigma_a^{(a',n,m)}(i) = j$ and

$$x \approx_{\sigma^{(a',n,m)} - \{(a,i,j)\}} y.$$

By statement (3) of Theorem 4.3, $F(a'_m)$ occurs in d and $\rho'(F(a'_m)) \xRightarrow{F(a_j)} y$ imply that $d\rho' \xRightarrow{F(a_j)} \bar{d} \sim d\rho'[F(a'_m) \to y]$, for some $\bar{d}$. We are left to prove that $\langle \bar{c}, \bar{d} \rangle \in \mathscr{S}_{\phi - \{(a,i,j)\}}$. By the definition of $\mathscr{S}_{\phi - \{(a,i,j)\}}$, it is sufficient to show that

$$\langle c\rho[F(a'_n) \to x], d\rho'[F(a'_m) \to y] \rangle \in \mathscr{R}^{\text{sub}}_{\phi - \{(a,i,j)\}}.$$

We know that $c \approx_\varphi d$. Consider now the φ-vector σ_1 given by

$$\sigma_1^{(b',l,l')} = \begin{cases} \sigma^{(a',n,m)} - \{(a,i,j)\} & \text{if } (b',l,l') = (a',n,m) \\ \sigma^{(b',l,l')} & \text{otherwise.} \end{cases}$$

Then, as $\rho =_{(\varphi,\sigma)} \rho'$ and $x \approx_{\sigma^{(a',n,m)} - \{(a,i,j)\}} y$, it is easy to see that

$$\rho[F(a'_n) \to x] =_{(\varphi,\sigma_1)} \rho'[F(a'_m) \to y].$$

Moreover, we have that

$$\begin{aligned} \bigcup_{(b',l,l') \in \varphi} \sigma_1^{(b',l,l')} &= \left(\bigcup_{(b',l,l') \in \varphi - \{(a',n,m)\}} \sigma^{(b',l,l')} \right) \cup \sigma_1^{(a',n,m)} \\ &= \left(\bigcup_{(b',l,l') \in \varphi - \{(a',n,m)\}} \sigma^{(b',l,l')} \right) \cup (\sigma^{(a',n,m)} - \{(a,i,j)\}) \\ &\overset{*}{=} \bigcup_{(b',l,l') \in \varphi} \sigma^{(b',l,l')} - \{(a,i,j)\} \\ &= \phi - \{(a,i,j)\}. \end{aligned}$$

Step $\overset{*}{=}$ in the above verification is justified by the fact that $(a,i,j) \notin \sigma^{(b',l,l')}$ for $(b',l,l') \neq (a',n,m)$. This completes the verification of (2.b). ▮

The refinement theorem for $\approx_r$ is now an obvious consequence of the previous result.

THEOREM 4.6 (The Refinement Theorem). *Let $p, q \in \mathbf{P}$ and ρ be an action refinement. Then $p \approx_r q$ implies $p\rho \approx_r q\rho$.*

Proof. For the sake of simplicity, we shall restrict ourselves to giving the details of the proof for an action refinement ρ which acts like the identity on all actions but $a \in V(\Lambda)$. Let ρ be such that

$$\rho(a) = r$$

$$\rho(\bar{a}) = r'$$

$$\rho(b) = b \qquad \text{for all} \quad b \neq a, \bar{a}.$$

The general case is only notationally more cumbersome. Assume that $p, q \in \mathbf{P}$ and $p \approx_r q$. Then there exist labelled processes π_1, π_2 such that $\mathbf{un}(\pi_1) = p$, $\mathbf{un}(\pi_2) = q$ and $\pi_1 \approx_\varnothing \pi_2$. We shall now construct a labelled action refinement ρ' compatible with both π_1 and π_2 (see Definition 4.3) such that, for each i, $\mathbf{un}(\rho'(a_i)) = \rho(a)$ and $\mathbf{un}(\rho'(\bar{a}_i)) = \rho(\bar{a})$.

Let $\{i_1, ..., i_k\}$ be the set of indices of occurrences of a in π_1 and π_2. Similarly, let $\{j_1, ..., j_h\}$ be the set of indices of occurrences of $\bar{a}$ in π_1 and π_2. Correspondingly, let $\{\pi_{i_1}, ..., \pi_{i_k}, \pi\}$ and $\{\bar{\pi}_{j_1}, ..., \bar{\pi}_{j_h}, \bar{\pi}\}$ be sets of processes with pairwise disjoint labellings such that

(a) for all $1 \leqslant n \leqslant k$, $\mathbf{un}(\pi_{i_n}) = \mathbf{un}(\pi) = r$, and

(b) for all $1 \leqslant n \leqslant h$, $\mathbf{un}(\bar{\pi}_{j_n}) = \mathbf{un}(\bar{\pi}) = \bar{r}$.

The labelled action refinement ρ': $LAct(\Lambda) \cup \{F(a_i) \mid a_i \in LAct(\Lambda)\} \to \mathbf{L}\mathscr{S}$ is now given by:

- $\rho'(a_n) = \pi_{i_m}$ if $n = i_m$, π otherwise;
- $\rho'(\bar{a}_n) = \bar{\pi}_{j_m}$ if $n = j_m$, $\bar{\pi}$ otherwise;
- ρ' is the identity everywhere else.

By the construction of ρ' and the fact that ρ is an action refinement, it is easy to see that ρ' is a labelled action refinement and that $\rho' =_{(\varnothing,\varnothing)} \rho'$. By construction, ρ' is compatible with both π_1 and π_2. Moreover, $\langle \pi_1\rho', \pi_2\rho' \rangle \in \mathscr{R}^{\text{sub}}_\varnothing$ and, therefore, by Theorem 4.5, it follows that $\pi_1\rho' \approx_\varnothing \pi_2\rho'$. By definition this means that $\mathbf{un}(\pi_1\rho') \approx_r \mathbf{un}(\pi_2\rho')$; i.e., $p\rho \approx_r q\rho$. ∎

In fact, the labelled refinement theorem enables us to prove a more general theorem for the unlabelled calculus. For action refinements ρ and ρ', let $\rho =_r \rho'$ if for all $a \in V(\Lambda)$, $\rho(a) =_r \rho'(a)$. Then it is straightforward to establish that

$$p =_r q \text{ and } \rho =_r \rho' \text{ imply that } p\rho =_r q\rho'.$$

5. Weak Bisimulation Equivalence Determines Weak Refine Equivalence

In this section we show that the congruence determined by weak refine equivalence, $=_r$, is the largest congruence with respect to all the operators in our language which is contained in weak bisimulation equivalence; i.e., that for every $p, q \in \mathbf{P}_\rho$,

$$p =_r q \text{ if and only if for all } \mathbf{P}_\rho\text{-contexts } C[\cdot],\ C[p] \approx C[q].$$

This result makes weak refine equality, $=_r$, a reasonable candidate for a semantic equivalence for process algebras which support action refinement. For it is preserved by action refinement and at least all of the other operators of CCS. In addition it is contained in weak bisimulation equivalence and therefore inherits many of the reasonable properties of this equivalence which has been much used in the literature. If these are the only constraints which we impose on a desired equivalence then the above result shows that $=_r$ is the largest relation which satisfies them; i.e., it satisfies the constraints and apart from that it makes the most identifications.

In one direction the above-stated characterization result for $=_r$ is straightforward; it follows from Theorem 3.2 because $\approx_r$ is contained in $\approx$. To prove the converse we design a specific action refinement σ which depends slightly on p and q such that

$$p\sigma \approx q\sigma \text{ implies } p \approx_r q. \tag{7}$$

As might be expected from the role played by the labelling of actions and subactions in the definition of $\approx_r$, claim (7) will follow from an analogous result for the labelled calculus. This involves extending action refinements to refinements which act on labelled states, i.e., mappings from labelled actions to processes.

We choose as the range of the specific action refinement an instantiation of $\mathbf{P}_\Lambda$ obtained by choosing a particular Λ. Recall that Λ is the basic set of action symbols used in the definition of the set of processes $\mathbf{P}_\Lambda$. We choose a derived set of action symbols defined by

$$\Delta = \{S(a_i), F(a_i) \mid a \in \Lambda \cup \bar{\Lambda},\ i \in N\} \cup \Lambda.$$

We use $\mathbf{P}_\Delta$, which inherits an operational semantics from Fig. 1, as the range of our refinement. Note that here the complement of the actions $S(a_i)$ and $F(a_i)$ is *not* $S(\bar{a}_i)$ and $F(\bar{a}_i)$, respectively, but $\overline{S(a_i)}$ and $\overline{F(a_i)}$.

Definition 5.1. A *standard refinement* σ is a mapping from $LAct(\Lambda) \cup \{F(a_i) \mid a_i \in LAct(\Lambda)\}$ to $\mathbf{P}_\Delta$ which satisfies, for each $a \in V(\Lambda)$,

1. for every i, j, $\sigma(a_i) = \sigma(a_j) = a + \sum \{S(a_k); F(a_k) \mid 1 \leqslant k \leqslant K\}$ for a sufficiently large K,

2. for every i there exists a $1 \leqslant j \leqslant K$ such that $\sigma(F(a_i)) = F(a_j)$.

We say that a standard refinement σ is suitable for a labelled state c if K is greater than any index occurring in c.

The idea behind this refinement is that the operational semantics of a labelled state c on which refine equivalence is based can be simulated by the ordinary operational semantics of the unlabelled process $c\sigma$ on which bisimulation equivalence is based.[2] Note for example, that communication is not allowed between any pair $\sigma(F(a_i))$, $\sigma(F(b_j))$ and is only allowed between $\sigma(a_i)$, $\sigma(b_j)$ if a and b are complementary, in which case it corresponds to one actually performing the action a and the other its complement. This simulation works sufficiently well provided we choose our labellings carefully. For labelled states c, d we say that they are *separated* if the sets of indices occurring in c and d are disjoint. Our aim is to prove the following theorem:

THEOREM 5.1. *Let π, π' be separated labelled processes and let σ be a standard refinement suitable for π and π'. Then $\pi\sigma \approx \pi'\sigma$ implies that $\pi \approx_{\varnothing} \pi'$.*

For the moment let us assume that we can prove this theorem. By convention we may also view standard refinements as ordinary action refinements,

$$\sigma: Act(\Lambda) \to P_{\Delta},$$

by letting $\sigma(a)$ be $\sigma(a_i)$ for any i. Note that this satisfies the requirements of an action refinement. Moreover, it is easy to see that for any labelled process π, $\pi\sigma = \mathbf{un}(\pi)\sigma$. From this we immediately obtain:

COROLLARY 5.1. *For all processes $p, q \in \mathbf{P}$, $p\sigma \approx q\sigma$ implies $p \approx_r q$.*

From this corollary we also obtain the main result of this section:

COROLLARY 5.2. *For all processes $p, q \in \mathbf{P}$, $p =_r q$ if and only if, for every $\mathbf{P}_\rho$-context $C[\cdot]$, $C[p] \approx C[q]$.*

[2] A similar idea underlies the refinements used by Vogler in [40, 39] to prove "largest-congruence" theorems for failures equivalence and weak bisimulation equivalence over safe Petri nets and prime Event Structures, respectively. However, working at a syntactic, operational level, we also have to take into account synchronization among actions. The problem does not arise if action refinement is defined on semantic models like Petri nets or Event Structures.

Proof. As stated above it is sufficient to prove the if direction, i.e., that $p + a \approx_r q + a$, where a does not appear in p and q, on the assumption that $C[p] \approx C[q]$ for every P_ρ-context $C[\cdot]$. To prove this we apply the above corollary to the context $([\cdot] + a)\sigma$ where σ is a suitable standard refinement. ∎

So it remains to prove Theorem 5.1 and this involves defining a weak refine equivalence. Let us say that the standard refinement σ is *compatible* with the history h if

$$(a, i, j) \in h \text{ implies } \sigma(F(a_i)) = \sigma(F(a_j)) = \text{either } F(a_i) \text{ or } F(a_j).$$

Then, for every history h, let $\mathscr{R}_h$ be the set of pairs of labelled states c, d which satisfy

1. (c, d) is compatible with h,
2. (c, d) are separated,
3. there is a standard refinement σ, suitable for c and d, such that
   (a) σ is compatible with h and
   (b) $c\sigma \approx d\sigma$.

We show that $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ is a weak refine bisimulation, at least "up to strong bisimulation." Theorem 5.1 then follows since every σ is compatible with the empty history. Before starting our analysis of the proof of this theorem, we give a lemma which states a fundamental property of standard refinements which are compatible with histories relating separated labelled states.

LEMMA 5.1. *Let $h \in \mathscr{H}$ and σ be a standard refinement compatible with h. Assume that $\{i \mid (a, i, j) \in h, \text{ for some } a, j\} \cap \{i \mid (a, j, i) \in h, \text{ for some } a, j\} = \varnothing$. Then, for every $a \in V(\Lambda)$, σ is injective on both $dom(h_a)$ and $range(h_a)$.*

The proof that $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$ is a weak refine bisimulation depends on particular properties of weak bisimulation equivalence over $\mathscr{P}_\Delta$ which we now explain. All the arguments revolve around the application of the Whence and Whither theorems of the previous section. Strictly speaking, these apply to the operational semantics of Section 3, using the next state relation $\mapsto$ rather than the relation $\rightarrow$ of Section 2. However, they can be trivially adapted to the latter and here it is this version of these theorem which we apply.

The next two lemmas are direct applications of the Whither and Whence theorems to standard refinements. The first states that moves from c can be accurately reflected in moves from $c\sigma$, whereas the second implies the converse.

LEMMA 5.2. *For every configuration c and standard refinement σ suitable for c:*

1. $c \overset{S(a_i)}{\longmapsto} c'$ *implies* $c\sigma \xrightarrow{S(a_i)} \sim c'\sigma[F(a_i) \to F(a_i)]$;
2. $c \overset{F(a_i)}{\longmapsto} c'$ *implies* $c\sigma \xrightarrow{F(a_j)} \sim c'\sigma$, *where* $\sigma(F(a_i)) = F(a_j)$;
3. $c \overset{\tau}{\longmapsto} c'$ *implies* $c\sigma \xrightarrow{\tau} \sim c'\sigma$.

Proof. Each result follows by an application of the Whither theorem.

1. Suppose $c \overset{S(a_i)}{\longmapsto} c'$. Then, as σ is suitable for c, K is greater than any index occurring in c and this implies that $\sigma(a_i) \xrightarrow{S(a_i)} F(a_i)$. Therefore, by statement (1) of the Whither theorem, $c\sigma \xrightarrow{S(a_i)} \sim c'\sigma[F(a_i) \to F(a_i)]$.

2. Suppose $c \overset{F(a_i)}{\longmapsto} c'$. By statement (3) of the Whither theorem, $c\sigma \xrightarrow{F(a_j)} \sim c'\sigma[F(a_i) \to nil]$. But the result now follows since $c'\sigma[F(a_i) \to nil] \sim c'\sigma$.

3. Immediate by applying statement (4) of the Whither theorem. ∎

LEMMA 5.3. *For every labelled state c and standard refinement σ:*

1. $c\sigma \overset{S(a_i)}{\Longrightarrow} x$ *implies* $c \overset{S(a_j)}{\Longrightarrow} c'$ *for some j and c' such that* $x \sim c'\sigma[F(a_j) \to F(a_i)]$;
2. $c\sigma \overset{S(a_i)}{\Longrightarrow} x$ *implies* $c \overset{F(a_j)}{\Longrightarrow} c'$ *for some j and c' such that $F(a_j)$ occurs in c,* $\sigma(F(a_j)) = F(a_i)$ *and* $x \sim c'\sigma$;
3. $c\sigma \overset{\tau}{\Longrightarrow} x$ *implies* $c \overset{\tau}{\Longrightarrow} c'\sigma$ *such that* $x \sim c'\sigma$.

Proof. These statements follow by applying the Whence theorem and we prove them in turn. We start by proving statement (3), which is used in the proof of the others.

3. It is sufficient to prove this for the case when $c\sigma \xrightarrow{\tau} x$, as the more general case, $c\sigma \overset{\tau}{\Longrightarrow} x$, will follow by induction on the number of moves in this derivation. So suppose that $c\sigma \xrightarrow{\tau} x$. Because of the form of σ, only two of the possibilities in the Whence theorem apply to $c\sigma \xrightarrow{\tau} x$, namely (2) and (4). Since (4) is exactly what we want, we concentrate on (2). Here there must be some a such that $c \overset{S(a_i)}{\longmapsto} c' \overset{S(\bar{a}_j)}{\longmapsto} c''$, $\sigma(a_i) \xrightarrow{a} nil$, $\sigma(\bar{a}_j) \xrightarrow{\bar{a}} nil$, and $x \sim c''\sigma[F(a_i) \to nil, F(\bar{a}_j) \to nil]$. By Lemma 4.1, $c \overset{\tau}{\longmapsto} \sim c''\iota[F(a_i) \to nil, F(\bar{a}_j) \to nil]$. The result now follows since, by the substitution lemma, $(c''\iota[F(a_i) \to nil, F(\bar{a}_j) \to nil])\sigma = c''\sigma[F(a_i) \to nil, F(\bar{a}_j) \to nil]$.

2. Again it is sufficient to consider the simple case when $c\sigma \xrightarrow{F(a_i)} x$ because the more general case will follow from it and Case (3). The only case of the Whence theorem which can apply here is case (3), from which we obtain some $F(a_j)$ occurring in c such that $\sigma(F(a_j)) = F(a_i)$ and $x \sim c\sigma[F(a_j) \to nil]$. Let c' be such that $c \overset{F(a_j)}{\longmapsto} c'$. By induction on this

derivation it is easy to prove that $c' =_\alpha c\iota[F(a_j) \to nil]$. The result now follows from Lemma 2.2(b) and the fact that, by the substitution lemma, $c\sigma[F(a_j) \to nil] = (c\iota[F(a_j) \to nil])\sigma$.

1. Similar to the statement above. ▮

We now state a useful lemma relating the termination potential of a labelled state c with that of the process $c\sigma$, where σ is a standard refinement. Because of the particular structure of our standard refinements, such a correspondence takes a very simple and natural form. (Compare with Theorem 4.4.)

LEMMA 5.4. *For every labelled state c and standard refinement σ suitable for c, $c\not\surd$ if and only if $c\sigma\not\surd$.*

Proof. First note that c is stable if and only if $c\sigma$ is stable. For an application of Lemma 5.2 implies that if c can perform a τ move then so can $c\sigma$ and if $c\sigma$ is not stable then an application of Lema 5.3 gives a τ move from c. Also if $c\surd$ then, for any refinement ρ, $c\rho\surd$. But the converse is also true for standard refinements. More generally, if the range of ρ contains only nonterminated processes, then $c\rho\surd$ implies $c\surd$. This can be shown by induction on the proof that $c\rho\surd$. In other words we have that $c\surd$ if and only if $c\sigma\surd$. Let us now prove the result.

(*Only if*) Suppose $c\not\surd$ and $c\sigma \stackrel{\varepsilon}{\Longrightarrow} x$ for some stable x. We must show that $x\surd$. By repeatedly applying Lemma 5.3, it follows that $c \stackrel{\varepsilon}{\Longrightarrow} c'$ for some c' such that $c'\sigma \sim x$. Since x is stable, $c\sigma$ is also stable and therefore so is c'. Since $c\not\surd$ we have that $c'\surd$. Therefore $c'\sigma\surd$ and also $x\surd$.

(*If*) Let us assume that $c\sigma\not\surd$ and $c \stackrel{\varepsilon}{\Longrightarrow} c'$ for some stable c'. By Lemma 5.2, we have that $c\sigma \stackrel{\varepsilon}{\Longrightarrow} \sim c'\sigma$ and therefore $c'\sigma$ is stable. This implies that $c'\sigma\surd$, from which it follows that $c\surd$. ▮

Combining all of these lemmas we may now prove the required result:

PROPOSITION 5.1. *For all $h \in \mathscr{H}$, $\langle c, d\rangle \in \mathscr{R}_h$ implies $c \approx_h d$.*

Proof. We shall prove that the family of relations $\{\mathscr{R}'_h \mid h \in \mathscr{H}\}$ given by

$$\mathscr{R}'_h = \{\langle c, d\rangle \mid \text{there exist } \langle x, y\rangle \in \mathscr{R}_h \text{ such that } c \sim x \text{ and } y \sim d\}$$

is a weak refine bisimulation. The claim will then follow by the reflexivity of $\sim$. It is easy to see that $\{\mathscr{R}'_h \mid h \in \mathscr{H}\}$ is symmetric because, by definition, so is $\{\mathscr{R}_h \mid h \in \mathscr{H}\}$. The compatibility requirements are also easily checked by calculation. Also, the termination requirements follow from Lemma 5.4. So it remains to check that moves are properly matched and we shall limit ourselves to considering the case $\langle c, d\rangle \in \mathscr{R}_h$. (The more general claim will then follow immediately by the properties of $\sim$.)

Let $\langle c, d\rangle \in \mathscr{R}_h$ and suppose that $c \overset{F(a_i)}{\longmapsto} c'$. Then by Lemma 5.2, $c\sigma \overset{F(a_j)}{\longrightarrow} \sim c'\sigma$, where $\sigma(F(a_i)) = F(a_j)$. Since $c\sigma \approx d\sigma$ it follows that $d\sigma \overset{F(a_j)}{\Longrightarrow} x$ for some x such that $x \approx c'\sigma$. Now applying Lemma 5.3 we may assume that $x \sim d'\sigma$, where $d \overset{F(a_k)}{\Longrightarrow} d'$ for some k such that $\sigma(F(a_k)) = F(a_j)$. It follows from the construction of $\mathscr{R}_h$ that $(a, i, k) \in h$. For let i' denote the label such that $(a, i, i') \in h$. Then $\sigma(F(a_i)) = \sigma(F(a_{i'})) = \sigma(F(a_k))$. But the fact that c and d are separated and that σ is compatible with h ensures that σ is injective on both the domain and range of h_a, by Lemma 5.1; so $i' = k$. Now let $h' = h - \{(a, i, k)\}$. Then it is easy to check that $\langle c', d'\rangle \in \mathscr{R}_{h'}$.

The proof that the other forms of moves from c, $c \overset{S(a_i)}{\longmapsto} c'$ and $c \overset{S(a_i)}{\longmapsto} c'$ are matched by moves from d is equally straightforward. ∎

6. An Example Distinguishing $\approx_t$ from $\approx_r$

In this section, we shall present an example showing that $\approx_t$ and $\approx_r$ are different equivalences over the language **P**. The example is based on a similar one devised by van Glabbeek to show that, in general, splitting an action into three gives rise to a different equivalence from $\approx_t$. What van Glabbeek's example, called the *owl-example* by its author, essentially shows is that $\sim_t$ and $\approx_t$ are *not* preserved by action refinements over a language which is rich enough to describe the processes used in it. It turns out that a slightly modified version of the processes used in the owl-example can indeed be described in the language **P** considered in this paper. As a corollary of van Glabbeek's result we then have that

Fact 6.1. *$\approx_t$ is not preserved by action refinement over* **P**.

In the remainder of this section we shall use van Glabbeek's owl-example [23] to prove that $\approx_t$ and $\approx_r$ are different equivalences over **P**. The version of the owl-example which is presented below is due to Vaandrager, who translated the original example into CCS.[3] Let P be the process given by

$$P = (a.(\bar{\beta} \mid (\bar{\alpha}' + c.(\alpha.e + \gamma.P_1))) \mid b.(\bar{\alpha} \mid (\bar{\gamma} + c.(\beta.d + \alpha'.P_2))))\backslash\alpha\backslash\alpha'\backslash\beta\backslash\gamma,$$

where $P_1 = d \mid c.\tau.e$ and $P_2 = e \mid c.\tau.d$. Let Q be identical to P, but with the roles of d and e interchanged. We shall show that $P \approx_t Q$, but $P \not\approx_r Q$. The arguments given below will hopefully be made more comprehensible by working with a semantic representation of the processes P and Q. The labelled transition systems denoted by the processes P and Q are given in

[3] We thank Frits Vaandrager for making this example available to us.

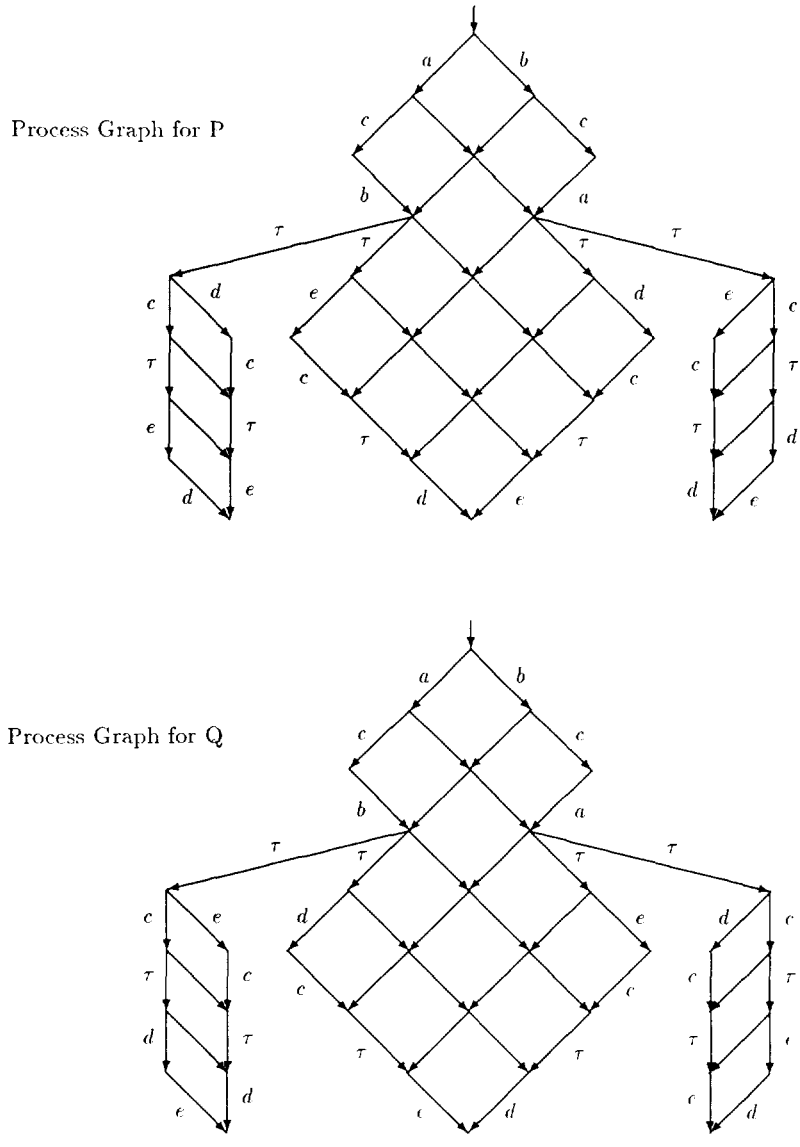

FIG. 3. The process graphs for P and Q.

Fig. 3. Intuitively, in this representation of processes, confluence in the process graph represents concurrency between actions, whilst absence of confluence stands for conflict. Note that only the labels of the transitions along the edges of the graph are explicitly given; the labels of the inner transitions

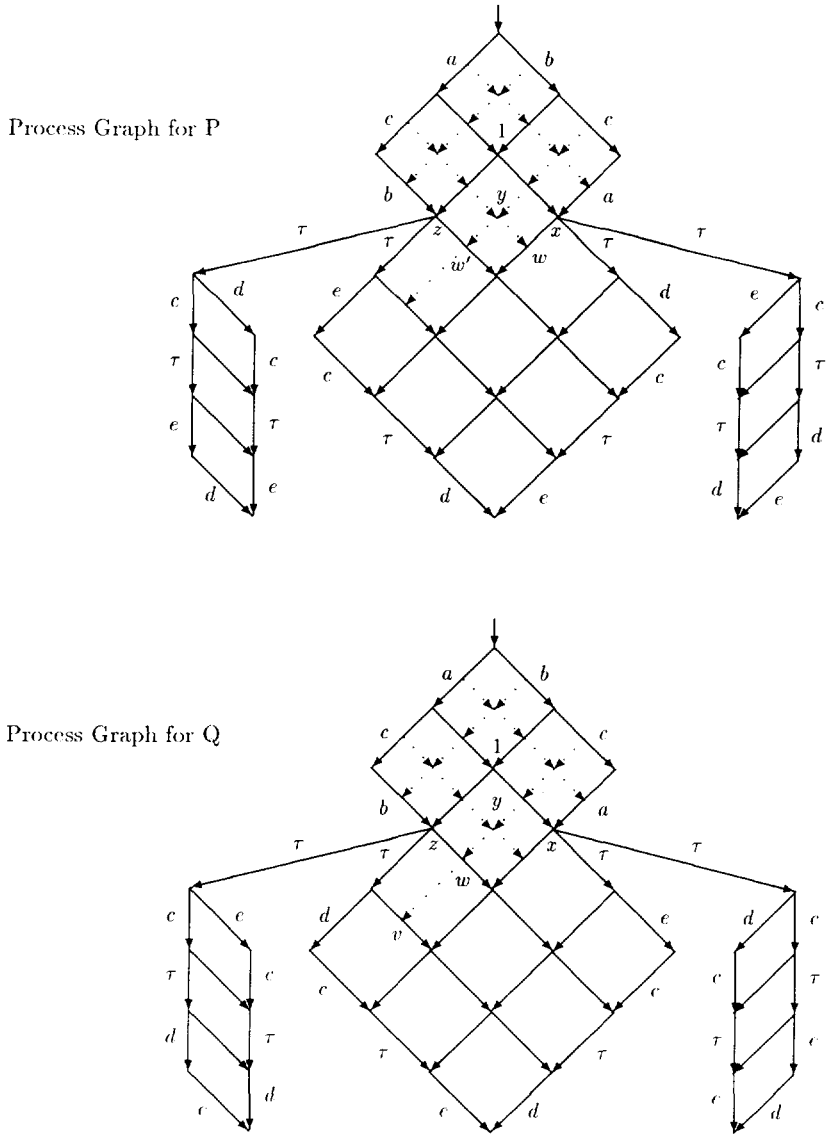

FIG. 4. The timed process graphs for P and Q.

can be obtained by assigning the same label to "parallel edges" in the graph.

We shall now prove that $\mathscr{G}(P) \approx_t \mathscr{G}(Q)$, where $\mathscr{G}(P)$ and $\mathscr{G}(Q)$ denote the process graphs for P and Q, respectively. In order to make our

argument clearer, we refer the reader to Fig. 4, where the interesting parts of the process graphs associated with P and Q with respect to the timed operational semantics are given. There the dotted arrows stand for beginnings and endings of the actions labelling the transitions which will be used in the arguments to follow. Before arguing informally that $\mathcal{G}(P) \approx_t \mathcal{G}(Q)$, it will be useful to introduce some notation.

Notation 6.1. Given a graph $\mathcal{G}$ and a node n of $\mathcal{G}$, $\mathcal{G}_n$ will denote the subgraph of $\mathcal{G}$ whose root is n. Isomorphism between graphs will be denoted by $\cong$.

We shall now informally sketch the construction of a timed bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$. The core of such a bisimulation is given by the relation

$$\mathcal{R} = \{\langle n, m\rangle \mid n \text{ is a node of } \mathcal{G}(P),\ m \text{ is a node of } \mathcal{G}(Q),\ \text{and } \mathcal{G}(P)_n \cong \mathcal{G}(Q)_m\}.$$

Note that, for instance, $\mathcal{R}$ relates the node labelled x in $\mathcal{G}(P)$ to the one labelled z in $\mathcal{G}(Q)$ and, conversely, the node labelled z in $\mathcal{G}(P)$ to the labelled x in $\mathcal{G}(Q)$. Moreover, $\langle y, y\rangle$ and $\langle w, w\rangle$ are in $\mathcal{R}$. It is easy to see that $\mathcal{R}$ is a timed bisimulation between $\mathcal{G}(P)_1$ and $\mathcal{G}(Q)_1$. We shall now extend $\mathcal{R}$ to a timed bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$. Note that *any* timed bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$ *must* relate the nodes labelled x and those labelled z in the two graphs, i.e., must contain the pairs $\langle x, x\rangle$ and $\langle z, z\rangle$. This is because

- the nodes labelled x in $\mathcal{G}(P)$ and $\mathcal{G}(Q)$ are the unique ones reachable from the roots of these graphs via the sequence of subactions $S(b)\,F(b)\,S(c)\,F(c)\,S(a)\,F(a)$ in which the start of an e-action and of a d-action are both possible, and
- the nodes labelled z in $\mathcal{G}(P)$ and $\mathcal{G}(Q)$ are the unique ones reachable from the roots of these graphs via the sequence of subactions $S(a)\,F(a)\,S(c)\,F(c)\,S(b)\,F(b)$ in which the start of an e-action and of a d-action are both possible.

An exhaustive analysis of all the possible transitions originating from the nodes labelled x and z in the two graphs shows that $\mathcal{G}(P)_x \approx_t \mathcal{G}(Q)_x$ and $\mathcal{G}(P)_z \approx_t \mathcal{G}(Q)_z$. We are thus left to extend the bisimulation relation to the nodes in $\mathcal{G}(P)$ and $\mathcal{G}(Q)$ lying above those labelled x, y, and z. Again, an exhausitve argument shows that "nodes in the same position" in the two graphs are indeed timed bisimilar. We have thus informally hinted at how to construct a timed bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$. Hence $\mathcal{G}(P) \approx_t \mathcal{G}(Q)$, as claimed.

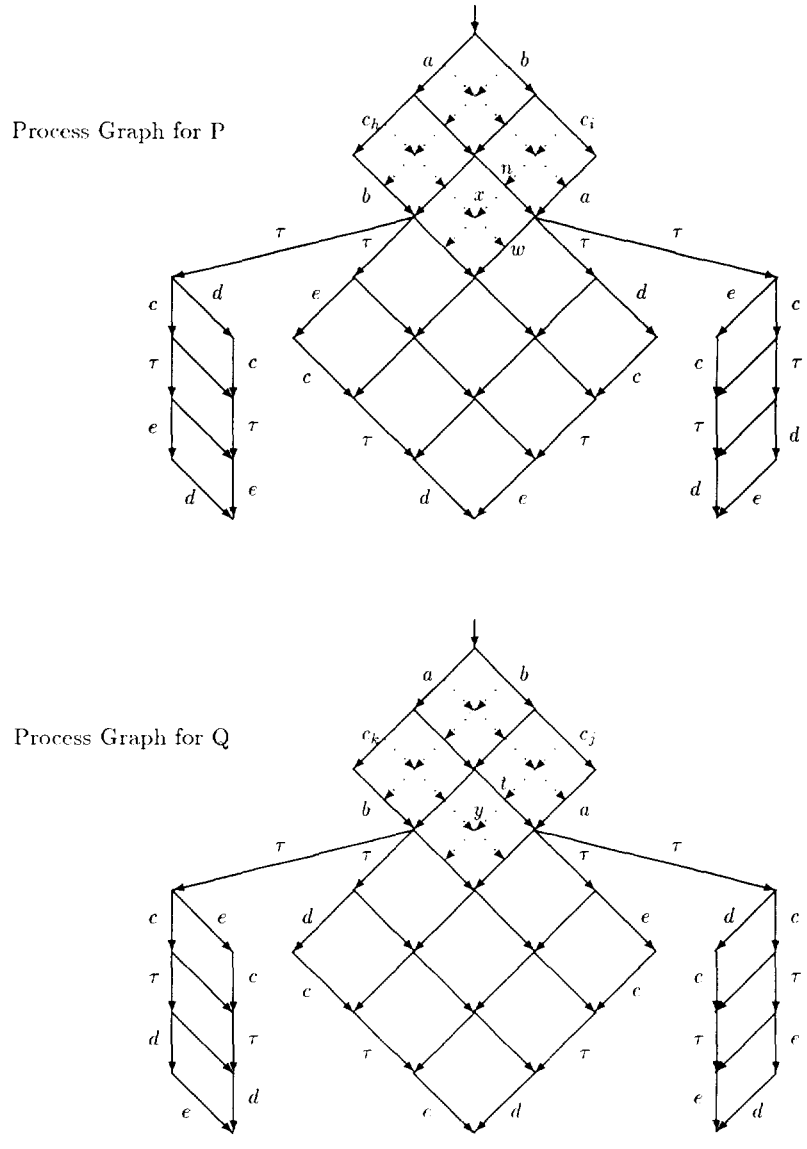

FIG. 5. (Parts of) the labelled process graphs for P and Q.

We now turn our attention to proving that $\mathscr{G}(P) \not\approx_r \mathscr{G}(Q)$. In order to clarify our argument, we refer the reader to Fig. 5, where the occurrences of action c relevant to proving the claim are labelled. Note, first of all, that *any* weak refine bisimulation between $\mathscr{G}(P)$ and $\mathscr{G}(Q)$ must relate the nodes

labelled n and t in Fig. 5 with respect to the history $\{(c, i, j)\}$. This is because n and t are the unique nodes in $\mathcal{G}(P)$ and $\mathcal{G}(Q)$, respectively, reachable from the roots by performing the sequences $S(b)\,F(b)\,S(c_i)\,S(a)\,F(a)$ and $S(b)\,F(b)\,S(c_j)\,S(a)\,F(a)$, respectively. However, we argue that n and t cannot be refine equivalent with respect to this history. For, when in the state corresponding to node n, the process denoted by $\mathcal{G}(P)$ can perform $S(c_h)$ and enter state x. This would have to be matched by t performing $S(c_k)$ to enter state y. However, x and y are *not* equivalent with respect to the history $\phi = \{(c, i, j), (c, h, k)\}$. In fact, we can get from state x to state w in $\mathcal{G}(P)$ by performing the end of action c_i, $F(c_i)$, and in state w it is possible to start a weak d-move. But none of the states reachable from y in $\mathcal{G}(Q)$ by performing the end of action c_j, the one associated with c_i by the history ϕ, can start a d-move. Hence there is no refine bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$; i.e., $\mathcal{G}(P) \not\approx_r \mathcal{G}(Q)$. It is easy to see that $\approx_r$ is contained in $\approx_t$. We then have that

FACT 6.2. *$\approx_r$ is strictly contained in $\approx_t$ over* **P**.

The same example can be used to prove that the definition of timed equivalence is not robust, in the sense that slight modifications in its definition give rise to different notions of equivalence. In Section 4.3, Proposition 3.4, we proved that adding a clause requiring the matching of complete actions to the definition of $\approx_r$ does not change the resulting notion of equivalence. A natural question to ask is whether the same is true of $\approx_t$. We shall now argue that the addition of a clause requiring the matching of complete actions in the definition of $\approx_t$ gives rise to a different notion of equivalence by showing that the resulting equivalence distinguishes the processes P and Q above. Again, it will be convenient to use the process graph representation of P and Q. Consider the state w' in Fig. 4 reachable from the root in $\mathcal{G}(P)$ via the sequence $\sigma = bS(c)\,ac$. Any timed bisimulation requiring in addition the matching of complete actions should relate w' to some state with equivalent potential reachable from the root of $\mathcal{G}(Q)$ via σ. There are only two states reachable from the root of $\mathcal{G}(Q)$ via a weak σ-transition. These states are labelled w and v in Fig. 4. We claim that w' can be equivalent to neither w nor v. In fact, w' can perform a weak e-move, while w and v cannot do so. This shows that there can be *no* timed bisimulation between $\mathcal{G}(P)$ and $\mathcal{G}(Q)$ which requires the matching of complete actions.

The situation described above does not present itself with respect to $\approx_t$. In fact, in the definition of $\approx_t$, complete actions are not allowed to occur atomically and the transition σ from the root of $\mathcal{G}(P)$ to the state w' considered above corresponds to performing the sequence $\sigma' F(c)$, where $\sigma' = S(b)\,F(b)\,S(c)\,S(a)\,F(a)\,S(c)$. The only state reachable from the root of

$\mathscr{G}(P)$ by performing σ' is state y in $\mathscr{G}(P)$. Similarly, the only state reachable from the root of $\mathscr{G}(Q)$ by performing σ' is state y in $\mathscr{G}(Q)$. We have already seen that $\mathscr{G}(P)_y \cong \mathscr{G}(Q)_y$. Hence the transition from y to w' in $\mathscr{G}(P)$ can indeed be matched, up to $\approx_t$, by a transition in $\mathscr{G}(Q)$.

7. Concluding Remarks

This paper has been devoted to the development of a process algebra for the specification of concurrent, communicating systems, which incorporates an operator for the refinement of actions by processes, and to the study of a suitable semantic equivalence between specifications written in this language. As we have seen, the syntactic and semantic treatment of action refinement for such a language is much more delicate than it was the case for the simple language considered in [3]. In particular, synchronization between complementary actions, the internal nature of the τ action, and the "scoping" of channel names induced by the restriction operator have to be taken into account both in the definition of a suitable notion of action refinement for the richer language considered in this paper and in the application of action refinements to processes.

At the semantic level, a suitable notion of equivalence for the language we have considered can still be obtained by assuming that the actions that processes perform during their evolution are not atomic. However, due to the expressive power of the language under consideration, the formalization of a suitable notion of bisimulation equivalence for it based on this intuition has turned out to be more involved than for the language considered in [3]. In particular, the natural weak version of the refine equivalence introduced in [3] turns out to play a more fundamental role than weak timed equivalence.

The treatment of action refinement for a language with communication, an internal action and restriction presented in this paper is, at least to the authors' knowledge, new. Most of the studies of action refinement in the literature deal with this operator at the *semantic* level and do not address *syntactic* issues like the treatment of binders. (See, e.g., [21, 24, 15, 20, 40, 39, 33].) Moreover, synchronization between concurrent activities does not require special treatment when dealt with at the semantic level only. As an example, consider the processes $p = a \mid \bar{a}$ and $q = p + \tau$. Semantically, i.e., when interpreted as objects such as event structures, these two processes are identical, as a τ-summand would be present in the semantic representation of p. Thus when action refinements are studied at this level there is no need to apply restrictions which ensure that they preserve intuitive semantic properties which are already represented in the semantic model, such as being able to perform an internal synchronization. On the other hand in

our treatment of action refinement, the action refinement operators are applied to syntactic descriptions of processes and thus have had to be restricted to those which, in some formal sense, preserve the intended semantics of processes. As an example, we have to restrict ourselves to considering action refinements which satisfy the axiom (ComPres) in order to ensure that action refinements do not interfere with the internal evolution of processes.

In this paper we have not addressed the issue of proof systems for refine equality over the language **P**. However, the second author has recently developed a complete proof system for refine equality over a language which extends a sublanguage of **P** with the left-merge and communication merge operations from ACP. The interested reader is invited to consult [31] for more details.

The refinement theorem presented in this paper is related to the ones for ST-bisimulation over Prime Event Structures [43] presented in [20, 39]. In fact, ST-bisimulation and refine equivalence, although defined in slightly different ways and on different domains, are both attempts to formalize the idea of a bisimulation-like relation between processes based on the matching of nonatomic actions. In [20], van Glabbeek only considers Prime Event Structures without internal τ-actions. Moreover, because of the chosen system model, he restricts the class of refinements to the finite, conflict-free ones, i.e., for each action a, $\rho(a)$, the process used to refine a, is a finite, conflict-free Event Structure. These restrictions on the allowed refinements are also present in [39]. In that reference, Vogler has shown that the ideas underlying ST-bisimulation can be used to turn, in a uniform way, several bisimulation-based equivalences over Prime Event Structures with internal actions into congruences with respect to action refinement. The relations so obtained are indeed the largest equivalences contained in the original ones which are preserved by action refinement.

The ideas underlying our refinement theorem and the ones presented in [20, 39] are closely related to those upon which the *failures semantics based on interval semiwords* proposed in [40] is based. There the author proposes a notion of failures semantics [13] for safe Petri Nets [37], which is the largest congruence with respect to action refinement contained in the standard, interleaving failures equivalence. This result of Vogler's has been recently generalized, mostly regarding the class of nets which are allowed as refinements of actions, in [33]. Related results are presented in [30], where a notion of concurrent testing of processes based upon the ST-idea is developed for a very expressive process algebra, essentially an extension of the one considered in this paper with recursive definitions of processes. The resulting ST-testing preorder is shown to be preserved by a rich class of syntactic action refinements and, moreover, is the largest such precongruence contained in the standard testing preorders [16]. A bridge

between the results presented in [39, 33] and those in [30] is provided by the paper [41]. In that reference, interval semiwords, which underlie the models in [39, 33], have been shown to be closely related to the so-called ST-traces [20], which are used in [30] to characterize the ST-testing preorder.

A syntactic theory of action refinement for a process algebra without communication and restriction/hiding operators has been presented in [36, 18]. There the authors provide a natural fully abstract model with respect to the largest congruence over their simple language contained in standard trace equivalence [32]. A similar result, but in the setting of failures equivalence, has been presented in [2].

In this paper we have been concerned with the study of operations which allow one to refine actions by processes. A dual approach is that of turning (executions of) processes into *atomic actions*. This has been studied at length in, e.g., [25, 17] in terms of *atomic action refinements*. Syntactically these are similar to our action refinements but the refined processes must be executed "atomically" and therefore the intention is very different; conceptually these kinds of action refinements are more easily construed as mechanisms for abstracting processes to atomic actions. Process algebras with mechanisms for "declaring" computations to be atomic actions have been presented in, e.g., [12, 27, 11], where several examples of the power and flexibility of such constructs can also be found. In particular, the approach followed in [12, 11] seems to be promising for the description and analysis of object-based systems by means of process algebraic techniques.

RECEIVED February 6, 1991; FINAL MANUSCRIPT RECEIVED May 7, 1992

REFERENCES

1. L. ACETO (1991), "Action-Refinement in Process Algebras," Ph.D. thesis; Report 3/91, Department of Computer Science, Univ. of Sussex. To appear in the "Distinguished Dissertations in Computer Science" series of Cambridge Univ. Press.
2. ACETO, L., AND ENGBERG, U. (1991), Failures semantics for a simple process language with refinement, *in* "FST and TCS 11, Foundations of Software Technology and Theoretical Computer Science, New Delhi, India" (S. Biswas and K. V. Nori, Eds.), pp. 89–108, Lecture Notes in Computer Science, Vol. 560, Springer-Verlag, Berlin/New York.
3. ACETO, L., AND HENNESSY, M. (1989), Towards action-refinement in process algebras, *in* "Proceedings 4th Annual Symposium on Logic in Computer Science, Asilomar, California," pp. 138–145, IEEE Comput. Soc. Press, 1989. Full version *Inform. and Comput.* (1993) **103**, 204–269.

4. ACETO, L., AND HENNESSY, M. (1992), Termination, deadlock and divergence, *J. Assoc. Comput. Mach.* **39**(1), 147–187.
5. BAETEN, J. C. M., AND VAN GLABBEEK, R. J. (1987), Merge and termination in process algebra, *in* "Proceedings, 7th Conference on Foundations of Software Technology and Theoretical Computer Science, Pune, India" (K. V. Nori, Ed.), pp. 153–172, Lecture Notes in Computer Science, Vol. 287, Springer-Verlag, Berlin/New York.
6. BAETEN, J. C. M., AND VAN GLABBEEK, R. J. (1989), Abstraction and empty process in process algebra, *Fundam. Informat.* **12**, 221–242.
7. BAETEN, J. C. M., AND VAANDRAGER, F. W. (1989), "An Algebra for Process Creation," Report CS-R8907, CWI, Amsterdam. Revised version to appear in *Acta Informat.*
8. BAETEN, J. C. M., AND WEIJLAND, W. P. (1990), "Process Algebra," Cambridge Tracts in Theoretical Computer Science, No. 18. Cambridge Univ. Press, London/New York.
9. BERGSTRA, J. A., AND KLOP, J. W. (1984), Process algebra for synchronous communication, *Inform. and Comput.* **60**, 109–137.
10. BERGSTRA, J. A., AND KLOP, J. W. (1985), Algebra of communicating processes with abstraction, *Theoret. Comput. Sci.* **37**(1), 77–121.
11. BOUDOL, G. (1989), Atomic actions, *Bull. Eur. Assoc. Theoret. Comput. Sci.* **38**, 136–144.
12. BOUDOL, G., AND CASTELLANI, I. (1988), Concurrency and atomicity, *Theoret. Comput. Sci.* **59**(1/2), 25–84.
13. BROOKES, S. D., HOARE, C. A. R., AND ROSCOE, A. W. (1984), A theory of communicating sequential processes, *J. Assoc. Comput. Mach.* **31**(3), 560–599.
14. CASTELLANO, L., DE MICHELIS, G., AND POMELLO, L. (1987), Concurrency vs interleaving: An instructive example, *Bull. Eur. Assoc. Theoret. Comput. Sci.* **31**, 12–15.
15. DARONDEAU, PH., AND DEGANO, P. (1989), "About Semantic Action Refinement," Technical Report 11/89, Dipartimento di Informatica, Università di Pisa. To appear in *Fundam. Informat.*
16. DE NICOLA, R., AND HENNESSY, M. (1984), Testing equivalences for processes, *Theoret. Comput. Sci.* **34**, 83–133.
17. DEGANO, P., AND GORRIERI, R. (1991), Atomic refinement in process description languages, *in* "16th Symposium on Mathematical Foundations of Computer Science" (A. Tarlecki, Ed.), pp. 121–130, Lecture Notes in Computer Science, Vol. 520, Springer-Verlag, Berlin/New York.
18. ENGBERG, U. (1990), "Partial Orders and Fully Abstract Models for Concurrency," Ph.D. Thesis, University of Århus. Published as DAIMI Report PB-307.
19. VAN GLABBEEK, R. J. (1990), "Comparative Concurrency Semantics and Refinement of Actions," Ph.D. Thesis, Free University, Amsterdam.
20. VAN GLABBEEK, R. J. (1990), The refinement theorem for ST-bisimulation semantics, "Proceedings, IFIP TC2 Working Conference on Programming Concepts and Methods, Sea of Galilee, Israel" (M. Broy and C. B. Jones, Eds.), North-Holland, Amsterdam.
21. VAN GLABBEEK, R. J., AND GOLTZ, U. (1989), Equivalence notions for concurrent systems and refinement of actions, *in* "Arbeitspapiere der GMD 366, Gesellschaft für Mathematik und Datenverarbeitung, Sankt Augustin." An extended abstract appeared *in* "Proceedings, 14th Symposium on Mathematical Foundations of Computer Science, Porąbka-Kozubnik, Poland, August/September 1989" (A. Kreczmar and G. Mirkowska, Eds.), pp. 237–248, Lecture Notes in Computer Science, Vol. 379, Springer-Verlag, Berlin/New York.
22. VAN GLABBEEK, R. J., AND VAANDRAGER, F. W. (1987), Petri net models for algebraic theories of concurrency, *in* "Proceedings PARLE Conference, Eindhoven, Vol. II (Parallel Languages)" (J. W. de Bakker, A. J. Nijman, and P. C. Treleaven, Eds.), pp. 224–242, Lecture Notes in Computer Science, Vol. 259, Springer-Verlag, Berlin/New York.

23. VAN GLABBEEK, R. J., AND VAANDRAGER, F. W. (1991), The difference between splitting in n and $n+1$, in preparation.
24. VAN GLABBEEK, R. J., AND WEIJLAND, W. P. (1989), "Refinement in Branching Time Semantics," Report CS-R8922, CWI, Amsterdam. Also appeared *in* "Proceedings AMAST Conference, Iowa, May 1989," pp. 197–201.
25. GORRIERI, R. (1990), "Refinement, Atomicity and Transactions for Process Description Languages," Dissertazione per il titolo di dottore di ricerca in informatica (terzo ciclo), Dipartimento di Informatica, Università di Pisa.
26. GORRIERI, R., AND LANEVE, C. (1991), The limit of split$_n$-bisimulations for CCS agents, *in* "16th Symposium on Mathematical Foundations of Computer Science" (A. Tarlecki, Ed.), Lecture Notes in Computer Science, Vol. 520, Springer-Verlag, Springer-Verlag, Berlin/New York.
27. GORRIERI, R., MARCHETTI, S., AND MONTANARI, U. (1988), A^2CCS: A simple extension of CCS for handling atomic actions, *in* "Proceedings CAAP 88, Nancy, France" (M. Dauchet and M. Nivat, Eds.), pp. 258–270, Lecture Notes in Computer Science, Vol. 299, Springer-Verlag, Berlin/New York.
28. GROOTE, J. F., AND VAANDRAGER, F. W. Structured operational semantics and bisimulation as a congruence (extended abstract), *in* "Proceedings 16th ICALP, Stresa" (G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, Eds.), pp. 423–438, Lecture Notes in Computer Science, Vol. 372, Springer-Verlag, Berlin/New York; Full version *Inform. and Comput.* (1992) **100**, 202–260.
29. HENNESSY, M. (1988), Axiomatising finite concurrent processes, *SIAM J. Comput.* **17**(5), 997–1017.
30. HENNESSY, M. (1991), "Concurrent Testing of Processes," Computer Science Report 11/91, Univ. of Sussex.
31. HENNESSY, M. (1991), "A Proof System for Weak ST-Bisimulation over a Finite Process Algebra," Computer Science Report 6/91, Univ. of Sussex.
32. HOARE, C. A. R. (1985), "Communicating Sequential Processes," Prentice–Hall, Englewood Cliffs, NJ.
33. JATEGOANKAR, L., AND MEYER, A. (1992), Testing equivalence for Petri nets with action refinement, unpublished manuscript.
34. MILNER, R. (1980), "A Calculus of Communicating Systems," Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, Berlin/New York.
35. MILNER, R. (1989), "Communication and Concurrency," Prentice–Hall, Englewood Cliffs, NJ.
36. NIELSEN, M., ENGBERG, U., AND LARSEN, K. S. (1989), Fully abstract models for a process language with refinement, *in* "REX School/Workshop on Linear Time, Branching Time and Partial Order in Logics and Models for Concurrency, Noordwijkerhout" (J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, Eds.), pp. 523–548, Lecture Notes in Computer Science, Vol. 354, Springer-Verlag, Berlin/New York.
37. REISIG, W. (1985), "Petri Nets—An Introduction," EATCS Monographs on Theoretical Computer Science, Vol. 4. Springer-Verlag, Berlin/New York.
38. STOUGHTON, A. (1988), Substitution revisited, *Theoret. Comput. Sci.* **59**, 317–325.
39. VOGLER, W. (1990), "Bisimulation and Action Refinement," SFB-Bericht 342/10/90, Technische Universität München.
40. VOGLER, W. (1990), Failures semantics based on interval semiwords is a congruence for refinement, *in* "Proceedings STACS 90, Rouen" (T. Lengauer and C. Choffrut, Eds.), pp. 285–297, Lecture Notes in Computer Science, Vol. 415, Springer-Verlag, Berlin/New York.
41. VOGLER, W. (1991), "Is Partial Order Semantics Necessary for Action Refinement?" SFB-Bericht 342/1/91, Technische Universität München.

42. VRANCKEN, J. L. M. (1986), "The Algebra of Communicating Processes with Empty Process," Report FVI 86-01, Dept. of Computer Science, University of Amsterdam.
43. WINSKEL, G. (1987), Event structures, *in* "Petri Nets: Applications and Relationships to Other Models of Concurrency, Advances in Petri Nets 1986, Part II: Proceedings of an Advanced Course, Bad Honnef" (W. Brauer, W. Reisig, and G. Rozenberg, Eds.), pp. 325–392, Lecture Notes in Computer Science, Vol. 255, Springer-Verlag, Berlin/New York.