ISSN 0106-0791

# CONTEXT-DEPENDENT BISIMULATION BETWEEN PROCESSES

by

KIM GULDSTRAND LARSEN

Institute of Electronic Systems
Aalborg University Centre
Strandvejen 19, 4
DK-9000 Aalborg C
DENMARK

Doctor of Philosophy
University of Edinburgh
1986

# PREFACE

This preface provides an alternative and hopefully somewhat broarder introduction to the following Ph.D. thesis. The preface also presents us with an opportunity to touch on some new experiences, which have been gained since the submission of the thesis. It should be stressed that the preface was not part of the submitted and accepted thesis.

In recent years several equivalences between non-deterministic and concurrent processes have been proposed in order to capture different notions of the extensional behaviour of a process. The proposed equivalences are useful for relating process descriptions at different levels of abstraction. In terms of design and verification of concurrent systems each equivalence provides a notion of correctness of an implementation IMP (i.e. a concrete process description) with respect to a specification SPEC (i.e. an abstract process description): to prove the correctness of IMP with respect to SPEC simply consists in proving the equivalence SPEC $\equiv$ IMP, where $\equiv$ is the equivalence under consideration (see figure 1).

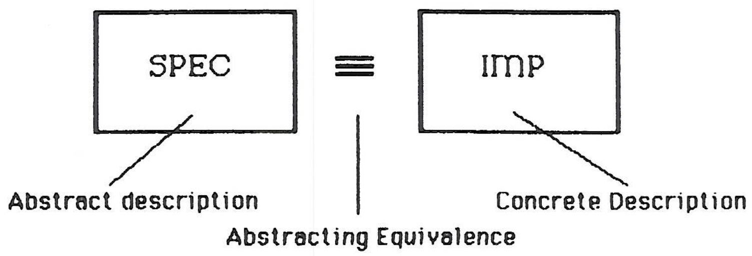



**Figure 1**
(Correctness of IMP wrt. SPEC)

Obviously, we do not in general expect to be able to derive IMP directly from SPEC. Rather, we expect to derive IMP through a series of small and succesive refinements of SPEC (so-called <u>stepwise refinement</u>). In order to guarantee the equivalence SPEC = IMP, and thus correctness of IMP with respect to SPEC, all the intermediate and increasingly more refined versions of SPEC introduced during the refinement process must be proved equivalent to their predecessor (see figure 2 below):

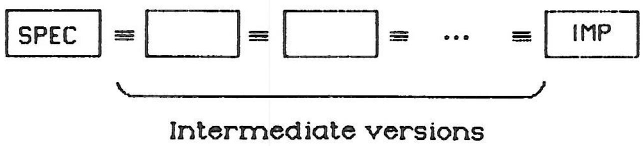



**Figure 2**
(Stepwise Refinement)

Each individual step of the refinement process consists of a small refinement of the current version (see figure 3).

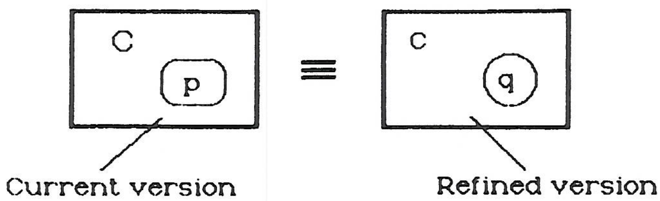



**Figure 3**
(A single refinement step)

I.e. some minor part $p$, which is considered too abstract for an implementation, is replaced by a more concrete "device" $q$. However, the bulk of the current version (the context $C$ in figure 3) is left unaffected by the refinement step. To complete the refinement step it remains to establish the equivalence between the new refined version $C[q]$ and it's predecessor $C[p]$. We may try to prove this equivalence directly by appealing to the definition of the equivalence. However, for any reasonable equivalence such a proof would have to deal not only with the parts $p$ and $q$, but also the common context $C$. Since we expect $C$ to be large relative to $p$ and $q$ — and increasingly so as the refinement process brings us closer to the final implementation — this <u>direct approach</u> clearly involves rather more work than we are willing to take on.

To provide an alternative approach, great care is normally taken to ensure that the equivalence proposed is in fact a <u>congruence</u> with respect to the various process constructing operations. If this is the case, it will suffice to prove $p \equiv q$ in order to establish the desired equivalence $C[p] \equiv C[q]$ (see figure 4). Thus, our proof will deal exclusively with the parts $p$ and $q$ and involve no considerations of the common context $C$ whatsoever. Unfortunately, from our past experiences with correctness proofs in the framework of CCS, it seems that this <u>congruence approach</u> may also lead to unnecessarily long and complicated proofs.

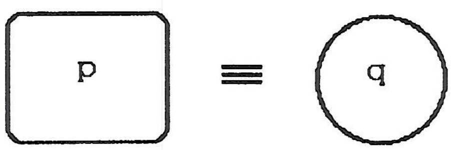



**Figure 4**
(Congruence Approach)

Firstly, it might be that $p$ and $q$ are not at all equivalent even though the desired equivalence $C[p] \equiv C[q]$ holds. This could happen if the subparts distinguishing $p$ and $q$ are not accessible in the context of $C$. In this case, the congruence approach will obviously fail and we must therefore face the direct approach. Secondly, even if $p$ and $q$ are equivalent, the equivalence proof would have to deal with the full behaviour of $p$ and $q$ including behaviour which is inaccessible, hence irrelevant, in the context $C$. Thus, we are proving more than necessary. In fact, we are proving $p$ and $q$ interchangeable in <u>any</u> context not just in the context $C$.

To summarize, in the direct approach all of the context $C$ must be dealt with, whereas $C$ will be totally disregarded in the congruence approach. At present no third alternative is available and a choice has to be made between these two extreme and completely opposite approaches. For the reasons given above we find this situation very unsatisfactory. In fact, we should like there to exist, not only a third alternative, but a whole continuous spectrum of new alternatives, where it suffices to prove an equivalence between $p$ and $q$ using some appropriate amount of information about the context $C$ in which $p$ and $q$ are going to be placed (see figure 5). In this setting the direct approach corresponds to proving $p \equiv q$ using every possible

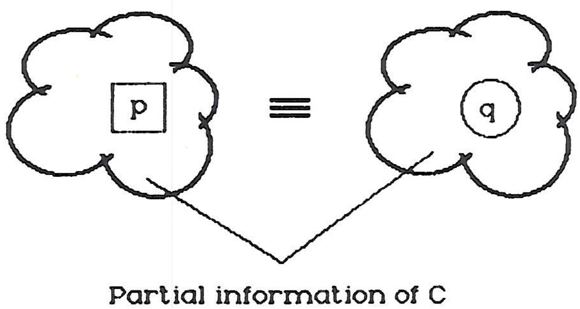
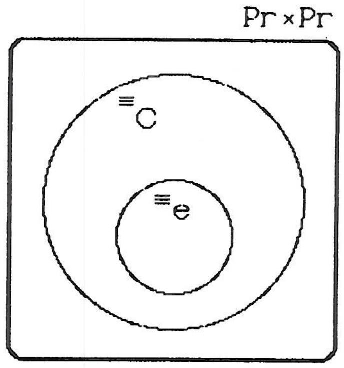



**Figure 5**

information of $C$, whereas the congruence approach corresponds to proving $p \equiv q$ using no information of $C$ at all.

To achieve this spectrum of alternatives, we assume the existence of a set $I$ containing all the possible information relevant for any context. Also, for each element $e$ of $I$, a <u>parameterized version</u>, $\equiv_e$, of $\equiv$ must be provided in order to determine what it means for processes to be equivalent relative to an element of $I$. An element $e$ can then be seen as partial information about a context $C$ just in case $C[p] \equiv C[q]$ whenever $p \equiv_e q$ (see figure 6).




Pr: set of processes
$\equiv_e$: equivalence relative to e
$\equiv_C = \{(p,q) \mid C[p] \equiv C[q]\}$

**Figure 6**
(e partial information about C)

However, not all elements of $I$ are equally informative. Intuitively, if both $e$ and $f$ are partial information of the context $C$ and $\equiv_f \subseteq \equiv_e$, then $e$ captures more accuratly than $f$ the notion of interchangeability in $C$. Hence, we may consider $e$ more informative than $f$ (see figure 7). In particular, we should like $I$ to contain for each context $C$ an element $w(C)$ being the most informative partial information of $C$. In this case, an element $e$ of $I$ is partial information

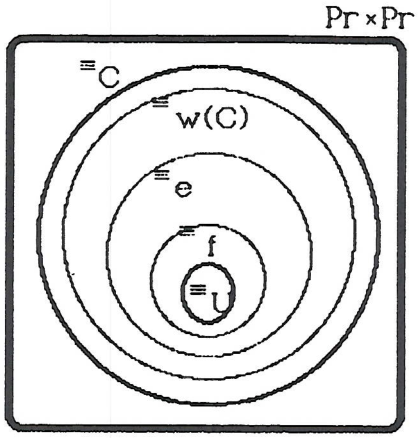



**Figure 7**

of a context C just in case $\equiv_e \subseteq \equiv_{w(C)}$. I may also contain a least informative element: in particular if U is an element of I such that $\equiv_U = \equiv$, then U contains no information of any context whatsoever. Thus, we may expect $\equiv_U \subseteq \equiv_e$ for all e in I.

Let us at this point make a short digression in order to point out some strong similarities to code optimization and data flow analyses. Code optimization often consists of replacing pieces of code with other, more efficient pieces of code while at the same time maintaining the semantics of the overall program. Consider for example the following simple program $P_1$:

$$P_1 \equiv \quad x:=z \;;\; y:=5 \;;\; x:=y+x \;;\; z:=y \;;\; z:=3$$

Obviously several optimizations are possible. In particular, we can replace the variable y with the constant 5 in the third assignment and totally remove the fourth assignment without altering the overall semantics (i.e. assuming the semantics is a function from input-state to output-state). Thus, $P_1$ is equivalent to the following optimized program $P_2$:

$$P_2 \equiv \quad x:=z \;;\; y:=5 \;;\; x:=5+x \;;\; z:=3$$

The equivalence between $P_1$ and $P_2$ can of course be proved directly by comparing their denotations. However, the pieces of code being changed are in general magnitudes smaller than the overall program, so this seems a pretty vasteful strategy and even more so if a <u>sequence</u> of optimizations is to be performed. Instead we would like to argue for the equivalence by comparing only the pieces of code involved. In the above example these are:

old-code ≡ x:=y+x ; z:=y
new-code ≡ x:=5+x

But, what is the connection between the two pieces of code ? Clearly they are not equivalent, let alone congruent. However, let us consider the context C in which they are used:

C ≡ x:=z ; y:=5 ; [ ] ; z:=3

The properties of C which are important in this situation — properties which could be calculated by some <u>data flow analyses</u> — seems to be the following: first, whenever execution reaches the hole in C the variable y must have the value 5. Second, when execution leaves the hole of C the value of the variable z has no affect on the final state. Under these assumptions it is easy to prove that old-code and new-code are substitutive in C and hence that $P_1$ and $P_2$ are equivalent.

Thus far we have tried to motivate the general idea of parameterizing equivalences between processes or programs with information of surrounding contexts. The above digression demonstrates that this idea is already being used for sequential programs, even though it might not always be presented as such. In the thesis that follows we prusue this general idea in the case of the bisimulation equivalence /P81B,Mil83/, and develop a complete theory of parameterized bisimulation equivalence between concurrent processes. In particular we give answers to the following questions:

1. What is the set of context information $I$ ?

2. For an element $e$ of $I$ what is the parameterized equivalence $\equiv_e$ ?

3. When is an element $e$ partial information of a context $C$ or equivalently: what conditions COND will ensure soundness of the following inference rule ?

$$\text{(R1)} \qquad \frac{p \equiv_e q}{C[p] \equiv C[q]} \quad ; \ \mathrm{COND}(C,e)$$

   $C[p]$ and $C[q]$ may themselves be situated in some outer context $D$ of which some partial information $f$ is known. We are therefore also interested in conditions COND which will make the following more general inference rule sound:

$$\text{(R2)} \qquad \frac{p \equiv_e q}{C[p] \equiv_f C[q]} \quad ; \ \mathrm{COND}(C,e,f)$$

4. Given the context $C$ and the outer environment $f$, there might exist a least discriminating environment $w(C,f)$ such that $p \equiv_{w(C,f)} q$ implies $C[p] \equiv_f C[q]$. In this case we may take the condition of R2 simply to be the inclusion $\equiv_e \subseteq \equiv_{w(C,f)}$. This raises the following questions: first, when does $w(C,f)$ exist and second, how can we determine inclusions of the form $\equiv_e \subseteq \equiv_f$ stating that $f$ is more informative than $e$ ? By equating environments with

predicates the environment $w(C,f)$ becomes an analogy to Dijkstra's weakest precondition and contexts are treated as predicate transformers. We pursue these analogies further in the thesis.

5. Having developed a theory of parameterized bisimulation equivalence it is paramount that we investigate how the theory may facilitate design and verification of concurrent systems. This involves extensive case-studies of which the thesis provides the beginning. The study of cases has continued after the submission of the thesis. In particular, in collaboration with Robin Milner we have recently applied the techniques of the thesis to the Alternating Bit Protocol. The outcome of this experiment makes us strongly believe in the potential usefulness of our techniques. Case-studies planned for the future include the class of Sliding Window Protocols of which the Alternating Bit Protocol is a particular simple example.

6. Through the study of examples it has become evident that the analyses involved is often tedious and delicate even for moderate size examples. Thus, it is rather clear that computer assistence is essential; not just to ensure the correctness of the proof but even to make the analyses feasible. In January 1986 the Laboratory of Foundations of Computer Science was founded in Edinburgh. The laboratory will be engaged in developing systems for computer assisted formal reasoning in general including a wide collection of computer based tools for design and analyses of concurrent systems. In relation to the very ambitious projects undertaken by this new laboratory the following thesis provides what can be seen as a modest start. More precisely: a PROLOG-system for deciding parameterized bisimulation equivalence

between CCS-processes is developed and implemented. It is shown that parameterized bisimulation equivalence is polynomial-time decidable. For this reason we expect our system — or similar systems — to be of potential use even for larger examples.

# ABSTRACT

In recent years several equivalences between nondeterministic and concurrent processes have been proposed in order to capture different notions of the extensional behaviour of a process. Usually the equivalences are congruences wrt. the process constructing operations in order to support hierarchic development and verification of systems. With the purpose of achieving more flexible hierarchic development methods we suggest parameterizing the equivalences with information about contexts.

We carry this suggestion out in full for the bisimulation equivalence, which we parameterize with a special type of context information called environments. The resulting parameterized equivalence is shown to have a large number of pleasant properties including a useful characterization of the information ordering on environments and a construction for producing the maximal environment identifying any two given processes.

Based on an investigation of how contexts transform environments it is shown how to reduce parameterized equivalence problems over composite processes to parameterized equivalence problems involving only the inner components of the processes. These results constitute the main tools provided by this thesis for hierarchic verification of systems.

All the results obtained for the parameterized bisimulation equivalence are extended to a similarly parameterized version of <u>weak</u> bisimulation equivalence. A worked example demonstrates the use of these extensions in correctness proofs.

Complete proof systems for the parameterized bisimulation equivalence for various combinations of the process and environment system are presented, extending existing proof systems for (unparameterized) bisimulation equivalence.

Finally, a PROLOG system for constructing bisimulations over CCS expressions has been implemented, verified and demonstrated.

# ACKNOWLEDGEMENTS

It is hard to express sufficiently how much I owe to my supervisor Robin Milner: his guidance, advise and constant encouragement and enthusiasm have been all-important factors in the making of this thesis.

Thanks are also due to Colin Stirling for suggesting the modal characterization in section 2.3 and for his constant support especially during the long search for a proof of the Main Theorem 2.4-20.

I am also grateful to Tatsuya Hagino for his willingness to discuss and comment on my work and for his expert assistance on Prolog.

Thank you so much to my wife Merete for being the anchor in my life and to my daughter Mia who made my stay in Edinburgh extra special.

The work presented in this thesis has been supported by a fellowship from Aarhus University, Denmark.

# DECLARATION

I hereby declare that this thesis has been composed by myself, that the work reported has not been presented for any university degree before, and that the work is my own, with the exception of the modal characterization in section 2.3 which was **proved by Colin Stirling.** A preliminary version of chapter 2 was accepted for the "12th Colloquium on Automata, Languages and Programming", Naplion, Greece, July 1985 as "A Context Dependent Equivalence between Processes" /La85/ and was selected in extended form for a special issue of TCS to appear.

Kim Guldstrand Larsen

# CONTENTS

# CHAPTER 1

# INTRODUCTION

## BACKGROUND

A major goal in the area of concurrent and sequential systems is to achieve semantic theories which support hierarchic and modular design and verification of systems. That is to say, given only the specification of components (not their implementation) it should be possible to deduce whether the components in a particular context or configuration will implement (or satisfy) some overall specification.

For sequential systems such theories are by now well-established. Perhaps most well-known is the theory of Denotational Semantics, founded by Scott and Strachey, which successfully has been used for describing the semantics of many sequential programming languages and systems /Gor79,Stoy77/. In Denotational Semantics, programs are basically modelled as computable functions from the domain of input values to the domain of output values. Also, the semantics of a composite program is expressed in terms of the semantics of its components thus satisfying the requirement of modularity.

However, for concurrent systems this semantic theory is inadequate. A concurrent system may have many interesting properties which cannot be described by an input-output function semantics (e.g. liveness, deadlock). Indeed, the purpose of a concurrent system may be entirely different from that of computing a function; e.g. an operating system which, despite it being non-terminating, normally is regarded as being a useful system. Even if we were to only consider the input-output function behaviour of concurrent systems, the requirement of modularity would fail to hold: there is simply no way of predicting the input-output behaviour of a concurrent system from the input-output behaviours of its components. In order to determine the systems overall behaviour, it seems that further information about possible intermediate states of the subcomponents is needed.

Concurrent systems are obviously more difficult to design and analyse than sequential ones, because they can exhibit very complicated behaviours. For this reason the requirement of modularity becomes a must for any semantic theory for concurrent systems. Though many new theories have been proposed recently, there is, as yet, no general agreement as to what a suitable theory is. A main disagreement seems to be whether the theory should be <u>intensional</u> in the sense that concurrency is a basic notion modelled in terms of <u>causal independence</u> and <u>dependence</u> of events or <u>extensional</u> in the sense that concurrency is viewed as unobservable and therefore indistinguishable from a <u>non-deterministic interleaving</u> of events. Representatives of the intensional approach are Petri Net /Pet80/, Event Structures /W80/ and Mazurkiewicz Traces /Maz77/.

Spurred on by the success of the Scott-Strachey approach for sequential languages, the notion of <u>power-domains</u> - a domain theoretic equivalent to powersets - was introduced /Pl76,Smy78/ in order to allow for non-deterministic computations. Based on powerdomains a notion of resumptions /P176/ (which contains information about the intermediate states of a non-deterministic computation) was used by Milne and Milner /MMi179/ to give an interleaving based model of a system of processes and process constructions. However, the model led to many unwanted identifications and was therefore abondoned in favour of an operational-based semantics. Out of this early research grew the calculus CCS /Mi180/ intended to serve the same purpose for concurrent computation as the lambda calculus does for sequential computation.

The operational semantics of CCS is given in terms of a labelled transition system /K75,P181/ describing the observation, or action, capabilities of processes and the resulting dynamic evolution of processes. Based on the operational semantics several equivalences and preorders have in recent years been proposed in order to capture different aspects of the extensional behaviour of a process. This results in semantic theories where both the requirements to a concurrent system (the specification) and its final realization (the implementation) can be expressed in the same formalism, e.g. CCS. The only difference, if any, in the two descriptions will be their computational feasibility in whatever model of computation of computation that is used. Based on the preorder and equivalence of the theory, the correctness of the implementation with respect to the specification can be stated and proved. Often the various theories provides (complete) algebraic laws useful for proving such correctness assertions. To achieve the goal of modularity great care is normally taken to ensure that the preorders and equivalences are substitutive with respect to the various process constructing operations.

The following is a short account of some of the abstracting equivalences and preorders which have been proposed recently. Generally all the equivalences and preorders are based on some idea of observation and how to use the result of an observation to either distinguish or identify processes.

<u>String or Trace equivalence:</u> This is the traditional language-theoretic equivalence where two processes are identified if they permit or accept the same sequences of observations. The equivalence has been used as the basis for a model of CSP /Ho81/. Unfortunately the equivalence does <u>not</u> preserve deadlock properties, and is therefore normally considered inadequiate.

<u>Failure equivalence:</u> In order to repair the deficiency of trace equivalence with respect to preservation of deadlock the failure equivalence was introduced /HoBroR84/. In addition to traces (= sequences of observations) of a process, also the set of observations which may fail (= deadlock) after each trace is taken into account.

<u>Testing equivalence:</u> /NiHen82,Ni85/. Here the equivalence of processes is determined by what tests a process can pass. A test $t$ is itself a process and applying $t$ to a process $p$ is a simple execution of $t$ in parallel with $p$, i.e. $p|t$. Then $p$ can pass $t$ in two different ways:

$p$ <u>may</u> $t$ $\Leftrightarrow$ "$p|t$ may, in some execution, perform the action <u>success</u>"

$p$ <u>must</u> $t$ $\Leftrightarrow$ "$p|t$ must, in every execution, perform the action <u>success</u>"

The two ways of passing tests give rise to the following two preorders:

$p \sqsubseteq_1 q \Leftrightarrow p$ <u>may</u> $t \Rightarrow q$ <u>may</u> $t$

$p \sqsubseteq_2 q \Leftrightarrow p$ <u>must</u> $t \Rightarrow q$ <u>must</u> $t$

Observational equivalence: This equivalence requires a strong relationship between the intermediate "states" of two processes in order for them to be considered equivalent. As a result the observational equivalence is more discriminating than any of the equivalences previously mentioned. Basically, two processes are observational equivalent if they have the same set of potential (first) observations and moreover can remain observational equivalent after the observation. The notion of observational equivalence was originally introduced by Robin Milner /Mil80/ as the intersection of a decreasing $\omega$-chain of (binary) relations. However, it turns out that the functional $\mathbb{F}$ used in constructing this chain is not continuous and the observational equivalence will therefore in general not be a fixed-point of $\mathbb{F}$. For this reason a slightly stronger equivalence (bisimulation equivalence), being the maximal fixed-point of $\mathbb{F}$, was introduced by David Park /P81B/ and later investigated by Michael Sanderson /San82/ and Robin Milner /Mil83/.

Comparisons of (some of) the above equivalences and their operational implications can be found in /BroR83/ and /Ni85/.

Recently, attempts have been made to give an alternative characterization of the abstract behaviours of processes in terms of the (modal) properties they enjoy. In this approach properties can be seen as providing the specifications, and the correctness of an implementation with respect to a specification is determined by the satisfaction relation between processes and properties. Based on the set of properties enjoyed (satisfied) by a process this approach also generates (in the obvious way) an equivalence (and preorder) between processes. Many of the preorders and equivalences mentioned previously have been shown to be generated by some set of modal properties /HenMil83,Pn85,BlTr85,BroR83,GrSif84,GrSif85,Mil81/.

In order for this approach to provide the required modularity, sound and complete (compositional) proof systems for the satisfiability problem have been given for various combinations of process system (some subset of CCS) and property domain /St83,St85,St84,W85,W85B/.

## MOTIVATION

The motivation for the work presented in this thesis is the possibility of achieving more flexible and easy-to-use hierarchic development methods for concurrent systems by <u>parameterizing</u> the equivalences with <u>information about contexts</u>. This idea of using information about contexts has proved successful in other connections: In /BK83,BKPn84/ a similar technique lead to decomposability of temporal logic specifications, and in /St84/ a relativized (with respect to information about other parallel components) satisfaction relation is used in order to obtain a sound and complete (compositional) proof system for CCS <u>with</u> concurrent composition.

Now consider the following hierarchic development method, the so-called <u>stepwise refinement method</u>: A specification, <u>SPEC</u>, of some desired non-deterministic or concurrent process has been given. The task is to find an implementable version of SPEC, <u>IMP</u>, such that IMP $\equiv$ SPEC ( $\equiv$ being the equivalence under consideration). Using the stepwise refinement method IMP is constructed in the following way. First decide on which process construction, <u>C</u>, to use and write down a sub-specification, <u>SUBSPEC</u>, such that C[SUBSPEC] $\equiv$ SPEC. Now find - using the stepwise refinement method recursively if SUBSPEC is not computationally feasible already - an implementation <u>SUBIMP</u> of SUBSPEC, i.e. SUBIMP $\equiv$ SUBSPEC. Then taking IMP to be C[SUBIMP] will clearly give an implementation of SPEC under the assumption that $\equiv$ is a congruence.

Looking carefully at the stepwise refinement method as stated above we notice that it requires SUBIMP and SUBSPEC to be proved congruent, i.e. interchangeable in <u>any</u> context and not just interchangeable in the context C in which they actually are going to be placed. We are therefore brought to prove more than seems necessary. Moreover, the subspecification SUBSPEC may have to specify behaviour which is not at all relevant in the context C. Again it seems that we are imposing a stronger requirement than necessary.

In order to reduce this work, we will <u>parameterize</u> the equivalence $\underline{=}$ with <u>information</u> about contexts. The required proof of SUBIMP $\underline{=}$ SUBSPEC can then be replaced by a proof of the more specific SUBIMP $\underline{=}_e$ SUBSPEC where e is information about the context C. Now assume that all the possible information relevant to parameterizing our equivalence $\underline{=}$ is collected in a <u>domain of information</u> $\underline{I}$. Then for any context C we may associate a subset <u>Inf</u>(C) of $\underline{I}$ defined by:

$$e \in \underline{Inf}(C) \overset{\Delta}{\Leftrightarrow} \forall p,q \in Pr.\ p \underline{=}_e q \Rightarrow C[p] \underline{=} C[q]$$

where Pr is the set of processes. Thus any $e \in \underline{Inf}(C)$ can be seen as <u>valid</u> information about C and can as such be used in the proof of SUBIMP $\underline{=}_e$ SUBSPEC. However, not all elements of <u>Inf</u>(C) contain the same amount of information about C. In particular if $e,f \in \underline{Inf}(C)$ such that $\underline{=}_f \subseteq \underline{=}_e$ we would consider e as being more (or more accurate, not less) informative than f since e agrees more closely to the equivalence induced by C: namely that of "interchangeability in the context C". Thus we define the preorder $\prec$ on information as follows:

$$f \prec e \overset{\Delta}{\Leftrightarrow} \underline{=}_f \subseteq \underline{=}_e$$

We shall denote the opposite ordering of $\prec$ by $\sqsubseteq$, and read $e \sqsubseteq f$ as "f is at least as <u>discriminating</u> as e".

Now define for any information $e \varepsilon \underline{I}$ the set of contexts $\underline{Con}(e)$ of which $e$ is valid information, i.e.:

$$\underline{Con}(e) = \{ C \mid e \varepsilon \underline{Inf}(C) \}$$

Let us assume that the domain of information $\underline{I}$ does not exceed the expressive power of contexts, in the sense that incompatible information can be distinguished by some context. Then the following is easily shown to hold:

$$e \sqsubseteq f \quad \Leftrightarrow \quad \underline{Con}(e) \subseteq \underline{Con}(f)$$

i.e. $e$ is at least as informative as $f$ if and only if any context for which $e$ is valid information $f$ is also valid information. As such, if there exists an element $U$ in $\underline{I}$ such that $\equiv_U = \equiv$ then $U$ will be a member of $\underline{Inf}(C)$ for any context $C$, since $\equiv$ is a congruence. Thus $U$ will be the maximal element under $\sqsubseteq$ or equivalently for all elements $e$ of $\underline{I}$, $\equiv_U \subseteq \equiv_e$.

Let us now return to the stepwise refinement method. As already mentioned SUBIMP may itself have been obtained by a stepwise refinement. I.e. for some context $D$ SUBIMP is D[SUBSUBIMP] where SUBSUBIMP is an implementation of SUBSUBSPEC with D[SUBSUBSPEC] $\equiv$ SUBSPEC. However, by using the parameterized equivalence we only have to prove SUBIMP $\equiv_e$ SUBSPEC so the above can be replace by taking SUBIMP as D[SUBSUBIMP] where D[SUBSUBIMP] $\equiv_e$ D[SUBSUBSPEC] and D[SUBSUBSPEC] $\equiv_e$ SUBSPEC. When $C$ is a context and $e$ is information then we define $\underline{Inf}^+(C,e) \subseteq \underline{I}$ as:

$$d \varepsilon \underline{Inf}^+(C,e) \quad \overset{\Delta}{\Leftrightarrow}$$
$$\forall p,q \varepsilon Pr.\ p \equiv_d q \quad \Rightarrow \quad C[p] \equiv_e C[q]$$

(Note that $\underline{Inf}^+$ generalizes $\underline{Inf}$ since $\underline{Inf}(C) = \underline{Inf}^+(C,U)$). Then, in order to obtain a proof of D[SUBSUBIMP] $\equiv_e$ D[SUBSUBSPEC] it should be enough to prove SUBSUBIMP $\equiv_d$ SUBSUBSPEC for some $d \varepsilon \underline{Inf}^+(D,e)$.

So far we have tried to motivate the idea of parameterizing process equivalences with information about contexts, by indicating its use in the stepwise refinement method. However, much is still left vague by the above description. First of all, what is "information about contexts" and secondly, how is this information used in parameterizing existing equivalences ? Once these two questions have been answered we must provide ways of deducing when some information e is valid with respect to a context C or more generally when $e \varepsilon \underline{\mathrm{Inf}}^{+}(C,d)$ for a context C and information d. In case there exists a minimal discriminating element, $\underline{\mathrm{min}}(C,d)$, in $\underline{\mathrm{Inf}}^{+}(C,d)$ we can reduce this problem to:

$$\underline{\mathrm{min}}(C,d) \sqsubseteq e$$

since $\underline{\mathrm{Inf}}^{+}(C,d)$ is upward closed under $\sqsubseteq$. Note, that this reduction emphasizes the importance of the ordering $\sqsubseteq$. As an analogy to Dijkstra's weakest precondition /Dij76/, we could term the element $\underline{\mathrm{min}}(C,d)$ the weakest inner information of d under C, and view contexts as weakest inner information transformers.

Assume that the equivalence, $\underline{=}$, considered is property generated, i.e. two processes are equivalent if they enjoy the same properties. Then, already at this early stage, we can give some indication as to what a parameterized version of $\underline{=}$ could be. Intuitively a context relates properties of processes placed inside it to outside properties of the combined process. If an (inner) property is not related to any non-trivial (outer) property under C it should not matter whether an inner process of C had that property or not. Thus, it seems that an appropriate information domain $\underline{I}$ simply consists of sets of properties, with two processes being equivalent with respect to a set of properties A if they enjoy the same properties of A.

## OVERVIEW

The main object of this thesis is to find and investigate suitable parameterized versions of the bisimulation equivalence /P81B,Mil83/.

It is well-known that bisimulation equivalence can be generated from a set of modal properties /HenMil83/, hence, by the remarks from the previous section, we can obtain a first parameterized version of bisimulation equivalence by simply using sets of modal properties as parameters. In the next chapter (chapter 2) we shall parameterize the bisimulation equivalence with another type of information called environments. First we give a short description of how to model processes and their operational behaviour in terms of labelled transition systems. We present and investigate the (abstracting) notions of simulation and bisimulation. The operational behaviour of environments is also described in terms of a labelled transition system. Intuitively, an environment is thought of as consuming (in a limited manner) actions produced by the inner processes. Based on environment as action consumers a notion of parameterized bisimulation and the parameterized bismulation equivalence it generates is introduced and investigated. It turns out that this parameterized bisimulation equivalence has all the properties expected in the last section. A modal characterization of the parameterized bisimulation equivalence is given showing an agreement between the two versions (environment contra sets of modal properties as parameters) of parameterized bisimulation equivalence. Finally, we present two main theorems. The first theorem gives a useful and simple characterization of the discrimination ordering, $\sqsubseteq$, between environments. The second theorem shows that there for any two processes exists a maximal environment (with respect to the simulation ordering) under which the two processes are identified.

In chapter 3 we look more closely at the way contexts translate information. In order to make this investigation easier and more general we give an abstract semantic account of contexts as action transducers. As an example it is shown how the standard CCS-contexts can be expressed in this formalism. In case the information is given as sets of modal properties we can for any context $C$ define a function $I_C$ which maps (desired) "outer" properties of $C[p]$ to "inner" sufficient and necessary properties of p. Extending $I_C$ to sets of modal properties gives the desired weakest inner information transformer. The function $I_C$ can also be used as a basis for complete, compositional proof systems similar to those recently given in /St83,St84,St85,W85,W85B/. For information given as environments slightly weaker results are obtained depending on the structure of the environment system.

In chapter 4 we present complete axiomatizations of the (environment) parameterized bisimulation equivalence for various combinations of the process and environment system.

Chapter 5 extends the definition and properties of (environment) parameterized bisimulation equivalence to the weak bisimulation equivalence, $\approx$, /Mil83/. A main problem in performing the extension is that $\approx$ is not preserved by all contexts - especially not sum-contexts. This makes the existence of weakest inner information (regardless of how the parameterization is done) impossible in general. Therefore conditions on the operational behaviour of contexts ensuring preservation of $\approx$ is given. All the standard CCS-contexts except sum-contexts satisfy these conditions. Finally, the parameterized weak bisimulation equivalence is used in proving the correctness of a simple scheduler (a simplification of the scheduler presented in /Mil80/).

In chapter 6 the complexity and implementation of the (environment) parameterized bisimulation problem is investigated. For general CCS-processes the problem is undecidable. However, for regular processes and environments the (restricted) problem is shown to be solvable in polynomial time, a surprising result considering that inequality of regular expressions is PSPACE-complete /GJ79/. The polynomial complexity result is obtained by a polynomial time reduction to a GENERALIZED PARTITIONING problem, for which a polynomial time algorithm has been designed in /KaSm83/. The GENERALIZED PARTITIONING problem is used in /KaSm83/ to show that the weak bisimulation equivalence problem can be decided in polynomial time for regular processes. Finally, an alternative decision procedure for bisimulation equivalence is implemented in PROLOG. A formal correctness proof of the implementation is given. A large subset of CCS and its operational semantics is also implemented in PROLOG. The usefulness of the resulting system is demonstrated through several examples.

# CHAPTER 2

# PARAMETERIZED BISIMULATION

In this chapter we shall parameterize the bisimulation equivalence /Mil80,Mil83,P81B/ with a special type of information called environments. First, in section 2.1, we give a short description of how to view processes and their behaviour as labelled transition systems. We define and investigate the notions of simulation and bisimulation together with the (simulation) preorder and (bisimulation) equivalence they generate.

In section 2.2 we introduce the concept of environments as elements of a labelled transition system. An environment consumes actions produced by an inner process. However, an environment's ability to consume actions may be limited, hence only part of the inner process' behaviour will be exploited by the environment. Using environments as parameters we then define and investigate a notion of parameterized bisimulation and the parameterized (bisimulation) equivalence it generates.

In section 2.3 we present a modal characterization of the parameterized bisimulation equivalence pointed out to us by Colin Stirling. The characterization extends in a natural way the existing modal characterizations of the simulation preorder and the (unparameterized) bisimulation equivalence, /HenMil83/.

In sections 2.4 and 2.5 we present two Main Theorems. The first theorem gives an important and simple characterization of the discrimination ordering, $\sqsubseteq$, between environments. The theorem simply says that the discrimination ordering is nothing more that the simulation preorder from section 2.1. Though easy to state the theorem was by no means easy to prove: only after several months search a proof was found. Unfortunately, the proof found only applies to environments satisfying certain finiteness conditions (the image-finiteness condition). Whether the theorem holds for general environments is left as an open problem. However, we prove that the present proof cannot be extended (in a direct way) to general environments.

The second theorem shows constructively that for any two processes there exist - in a sufficiently large environment system - a maximal environment (with respect to the simulation preorder) under which the two processes are equivalent. Thus the question of equivalence in an environment can be reduced to a question of simulation. It turns out that we can extend any environment system to a Heyting Algebra under the simulation ordering. We indicate briefly how to use this extended system as the interpretation for more complex formulas than merely (parameterized) equivalences between processes.

## 2.1 PROCESSES, SIMULATION AND BISIMULATION

### 2.1.1 Labelled Transition Systems.

A major goal in the area of concurrency is to achieve semantic theories that support hierarchic development and modular decomposition of programs. That is to say, given only the specification of a programs components (not their implementation) one should be able to deduce whether the program will implement (or satisfy) some overall specification.

For a sequential language a suitable semantic theory would be a theory of state-functions computed by programs written in that language. This is the view taken in Denotational Semantics /Gor79,Stoy77/. However, when concurrency is introduced this semantic theory is no longer adequate because of our modularity requirement: there is simply no way to predict the state-function behaviour of a concurrent program from the state-function behaviour of its components.

Thus, new semantic theories are needed, and in recent years a variety of such have been put forward. Underlying many of the proposed theories is the model of labelled transition systems /K75/. Labelled transition systems are a simple model of nondeterminism based on the two primitive notions of state and transition. In spite of (or maybe because of) their simplicity, labelled transition systems have proved an extremely general model for defining operational semantics of programming languages (see /Pl81,Pl82/).

By varying the definition of transition one can obtain a whole range of semantic descriptions, ranging from very concrete to more abstract. Also, various preorders and equivalences between nondeterministic programs, based on labelled transition systems, have been defined in order to abstract even further, /Bro83B, Bro83, NiHen82, Ni85, HoBroR81, Mil80, Mil81/.

Definition 2.1-1: A labelled transition system is a structure $(St, Act, \longrightarrow)$, where $St$ is a set of states (or configurations), $Act$ is a set of actions (or labels or operations) and $\longrightarrow \subseteq St \times Act \times St$ is the transition relation. □

Notation 2.1-2: For $(s,a,t) \in \longrightarrow$ we shall usually write $s \xrightarrow{a} t$ which is to be interpreted "in the state $s$ the system can perform the action $a$ and in doing so reach the state $t$". Often we shall write $s \xrightarrow{a}$ as an abbreviation for $\exists t \in St.\ s \xrightarrow{a} t$. Thus $s \xrightarrow{a}$ reads: "in the state $s$ the system can perform the action $a$".
Occasionally we shall extend $\longrightarrow$ to strings of actions using the following definition: $s \xrightarrow{a1..an} t$ iff there exists $s_i$, $0 \leqslant i \leqslant n$, such that $s \xrightarrow{a1} s_1 \xrightarrow{a2} s_2 \cdots s_{n-1} \xrightarrow{an} t$.
For complements of $s \xrightarrow{a} t$, $s \xrightarrow{a}$ resp. $s \xrightarrow{a1..an} t$ we shall use the notation $s \not\xrightarrow{a} t$, $s \not\xrightarrow{a}$ resp. $s \not\xrightarrow{a1..an} t$. For $s \in St$ and $a \in Act$, $s_a \subseteq St$ is the set of a-successors of s, i.e. $s_a = \{ t \in St \mid s \xrightarrow{a} t \}$. □

Definition 2.1-3: Let $R$ be a binary relation over the set $St$. Then $R$ is image-finite iff for each element $s$ of $St$ the set $\{ t \mid sRt \}$ is finite. □

Definition 2.1-4: We shall say that a labelled transition system is image-finite in case for all actions $a$ the binary relation $\xrightarrow{a} = \{ (s,t) \mid s \xrightarrow{a} t \}$ is image-finite. □

### 2.1.2 Processes, Simulation and Bisimulation.

As argued in the previous section we will model processes and their operational behaviour by labelled transition systems. We shall in this section introduce the general notions of simulation and bisimulation as means of abstracting the operational behaviour of a process, and we shall state some of their properties. For more detailed treatments and motivation we refer the reader to /Mil71,Mil80,Mil83,HenMil83/.

Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ be the labelled transition system modelling the operational semantics of a system of processes. We shall alternatively refer to the transition relation, $\longrightarrow$, of $\mathbb{P}$ as the <u>derivation relation</u>. Now, let p and q be two processes of $\mathbb{P}$. We then say that q <u>simulates</u> p or p <u>is simulated by</u> q if every derivation of p can be simulated by a derivation of q in such a way that the simulation property is maintained. We can formalize this by the following:

<u>Definition 2.1-5:</u> A <u>simulation</u> R is a binary relation on Pr such that whenever pRq and $a \in Act$ then:

$$\text{(i)} \quad p \xrightarrow{a} p' \Rightarrow \exists q'.\ q \xrightarrow{a} q' \ \& \ p' R q'$$

A process q is said to simulate a process p if and only if there exists a simulation R with pRq. In this case we write $p \leqslant q$. □

Now for $R \subseteq Pr^2$ we can define $\mathbb{S}(R) \subseteq Pr^2$ as the set of pairs (p,q) satisfying for all $a \in Act$ the clause (i) above. With this definition we can state the following properties:

<u>Proposition 2.1-6:</u> $R \subseteq Pr^2$ is a simulation iff $R \subseteq \mathbb{S}(R)$. □

<u>Proposition 2.1-7:</u> $\mathbb{S}$ is a monotonic endofunction on the complete lattice of binary relations (over Pr) under inclusion. □

Using the standard fixed-point result, originally due to Tarski /Ta55/, this implies:

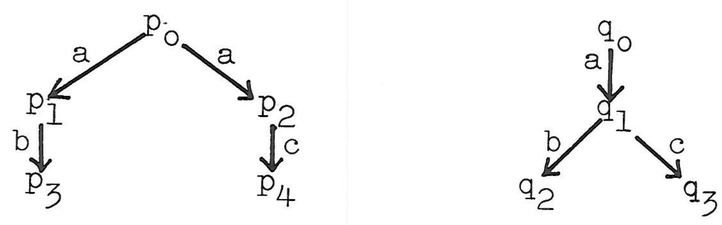
Proposition 2.1-8: $\mathbb{S}$ has a maximal fixed-point given by $\cup\{R \mid R \subseteq \mathbb{S}(R)\}$. Moreover $\leqslant$ equals this maximal fixed-point. $\square$

Proposition 2.1-9: $\leqslant$ is a preorder on $Pr^2$.

Proof: Show that $Id_{Pr}$ is a simulation and that composition of simulations yields a simulation. The proposition will then follow from the definition of $\leqslant$. $\square$

Note that the above definition of the simulation ordering admits an elegant proof technique: to show that $p \leqslant q$ it is sufficient and necessary to find a simulation containing $(p,q)$.

Example 2.1-10: Let $\mathbb{P}$ be given by the diagram below:




Then $R = \{(p_0,q_0),(p_1,q_1),(p_2,q_1),(p_3,q_2),(p_4,q_3)\}$ is a simulation. Thus $p_0 \leqslant q_0$. On the other hand $q_0 \not\leqslant p_0$. Assume namely that $R$ is a simulation containing $(q_0,p_0)$, then either $(q_1,p_1)$ or $(q_1,p_2)$ must be in R. However, in the former case $q_1 \xrightarrow{c}$ but $p_1 \overset{c}{\not\to}$ so if R is to be a simulation $(q_1,p_1)$ cannot be in R. Similarly it can be argued that $(q_1,p_2)$ is not in R. Therefore if R is a simulation it cannot contain $(q_0,p_0)$. $\square$

Definition 2.1-11: Let $\mathbb{F}$ be a function on a complete lattice D with greatest lower bound (glb), $\sqcap$, and least upper bound (lub), $\sqcup$. Then $\mathbb{F}$ is continuous iff for every increasing sequence $x_1 \sqsubseteq x_2 \sqsubseteq \ldots \sqsubseteq x_n \sqsubseteq \ldots$, of D elements $\mathbb{F}(\sqcup_n x_n) = \sqcup_n \mathbb{F}(x_n)$. $\mathbb{F}$ is anticontinuous iff for every decreasing sequence, $x_1 \sqsupseteq x_2 \sqsupseteq \ldots \sqsupseteq x_n \sqsupseteq \ldots$, of D elements $\mathbb{F}(\sqcap_n x_n) = \sqcap_n \mathbb{F}(x_n)$. $\square$

Now, if $\mathbb{S}$ is anticontinouous on the complete lattice of binary relations (with $\cap$ as glb) it follows from classical fixed-point theory that the maximal fixed-point, $\lesssim$ , is given as:

$$\lesssim \; = \cap_n \; \mathbb{S}^n(Pr^2)$$

where $\mathbb{S}^o = Id$ and $\mathbb{S}^{n+1} = \mathbb{S}^n \circ \mathbb{S}$. A sufficient condition for $\mathbb{S}$ to be anticontinuous is that the transition system $\mathbb{P}$ is image-finite (see definition 2.1-4).

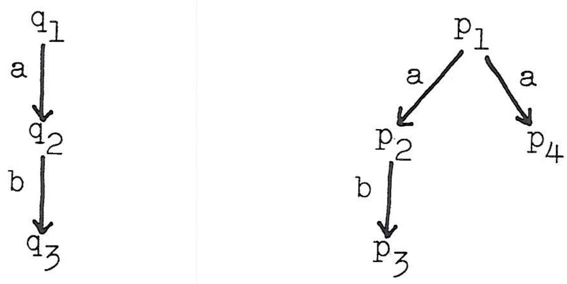
<u>Theorem 2.1-12:</u> If $\mathbb{P}$ is image-finite then $\mathbb{S}$ is anticontinuous.

<u>Proof:</u> Let $R_1 \supseteq R_2 \supseteq \ldots \supseteq R_n \supseteq \ldots$ be a decreasing sequence of binary relations over Pr. We must prove $\mathbb{S}(\cap_n R_n) = \cap_n \mathbb{S}(R_n)$. The "$\subseteq$"-direction follows directly from the monotonicity of $\mathbb{S}$ and $\cap_n R_n \subseteq R_i$ for all $i \varepsilon \omega$. For the "$\supseteq$"-direction let $(p,q) \varepsilon \cap_n \mathbb{S}(R_n)$ and let $p \xrightarrow{a} p'$. We must find a matching move for q such that $(p',q') \varepsilon \cap_n R_n$. Now $(p,q) \varepsilon \cap_n \mathbb{S}(R_n)$ iff for all $n \varepsilon \omega$, $(p,q) \varepsilon \mathbb{S}(R_n)$. Thus for all n there exists some $q_n$ such that $q \xrightarrow{a} q_n$ and $(p',q_n) \varepsilon R_n$. By image-finiteness of $\mathbb{P}$ this means that there exists a $q'$ such that $q \xrightarrow{a} q'$ and $(p',q') \varepsilon R_n$ for infinitely many $n \varepsilon \omega$. Since $R_n$ is decreasing in n, $(p', q') \varepsilon R_n$ for all $n \varepsilon \omega$ and thus $(p',q') \varepsilon \cap_n R_n$. By symmetry we conclude that $(p,q) \varepsilon \mathbb{S}(R_n)$. □

<u>Corollary 2.1-13:</u> If $\mathbb{P}$ is image-finite then $\lesssim \; = \cap_n \mathbb{S}^n(Pr^2)$. □

Now, two processes p and q could be considered equivalent if they simulate each other, i.e. $p \simeq q$ iff $p \lesssim q$ and $q \lesssim p$. However, this equivalence does not preserve deadlock properties as is demonstrated in the following example (see also /Mil80/).

<u>Example 2.1-14:</u> Let $\mathbb{P}$ be given by the diagram below:




Then $R_1 = \{(q_i, p_i) \mid i=1,2,3\}$ and $R_2 = \{(p_i, q_i) \mid i=1,2,3\} \cup \{(p_4, q_2)\}$ are both simulations. Thus $p \leqslant q$ and $q \leqslant p$. However, p can perform an a-action and reach a state where a b-action is impossible, whereas q cannot. Thus, p and q have different deadlock properties. $\square$

To obtain an equivalence that does preserve deadlock properties the notion of bisimulation is introduced. Under this notion, two processes are considered equivalent if they have the same set of potential first actions and can remain having equal potentiality during the course of execution. More formally we have:

<u>Definition 2.1-15:</u> A binary relation R on Pr is a bisimulation iff both $R$ and $R^T = \{(p,q) \mid (q,p) \varepsilon R\}$ are simulations. Two processes, p and q, are said to be bisimulation equivalent iff there exists a bisimulation $R$ with $pRq$. In this case we write $p \sim q$. $\square$

Now for $R \subseteq Pr^2$ define $\overline{\mathbb{S}}(R)$, $\mathbb{B}(R) \subseteq Pr^2$ as:

$$\overline{\mathbb{S}}(R) = \left(\mathbb{S}(R^T)\right)^T \qquad \text{and} \qquad \mathbb{B}(R) = \mathbb{S}(R) \cap \overline{\mathbb{S}}(R)$$

Then we have the following properties:

<u>Proposition 2.1-16:</u> $R \subseteq Pr^2$ is a bisimulation iff $R \subseteq \mathbb{B}(R)$.

<u>Proof:</u> By proposition 2.1-6 and definition of bisimulation. $\square$

Proposition 2.1-17: $\mathbb{B}$ is a monotonic endofunction on the complete lattice of binary relations over Pr.

Proof: By proposition 2.1-7 and the fact that $\cap$ and $(-)^T$ are monotonic functions. $\square$

Proposition 2.1-18: $\mathbb{B}$ has a maximal fixed-point which equals $\sim$. $\square$

Proposition 2.1-19: $\sim$ is an equivalence relation.

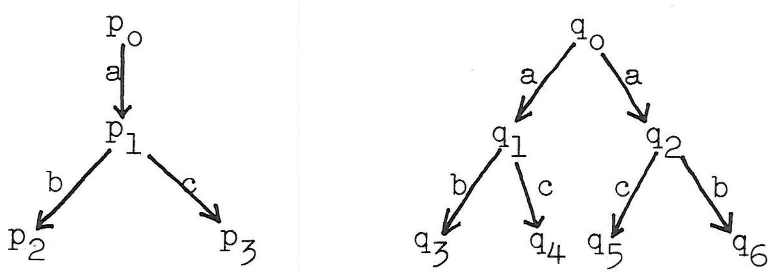
Proof: $Id_{Pr}$ is a bisimulation. Bisimulations are closed under composition and $(-)^T$. $\square$

Proposition 2.1-20: If $\mathbb{P}$ is image-finite then $\mathbb{B}$ is anticontinuous. Thus $\sim = \cap_n \mathbb{B}^n(Pr^2)$ where $\mathbb{B}^o = Id$ and $\mathbb{B}^{n+1} = \mathbb{B}^n \circ \mathbb{B}$.

Proof: From theorem 2.1-12 $\mathbb{S}$ is anticontinuous when $\mathbb{P}$ is image-finite. Both $\cap$ and $(-)^T$ are anticontinuous so the proposition follows since composition preserves anticontinuity. $\square$

As for simulation the definition of bisimulation equivalence provides an elegant proof technique due to proposition 2.1-18. This was first pointed out by David Park. To prove that $p \sim q$ it is sufficient and necessary to find a bisimulation containing $(p,q)$.

Example 2.1-21: Let $\mathbb{P}$ be given by the diagram below:




Then $R = \{(p_o,q_o),(p_1,q_1),(p_1,q_2),(p_2,q_3),(p_2,q_6),(p_3,q_4),(p_3,q_5)\}$ is a bisimulation with $p_o R q_o$. Thus $p_o \sim q_o$. In example 2.1-14, $R_1 \neq R_2^T$ so there is no reason to conclude

$p_1 \sim q_1$. In fact it can be shown that the two processes, $p_1$ and $q_1$ of example 2.1-14 are not bisimulation equivalent. □

The above example gives some indication of the relationship between the simulation ordering $\leqslant$ and the bisimulation equivalence $\sim$. The following proposition shows that $\sim$ is smaller than $\simeq$.

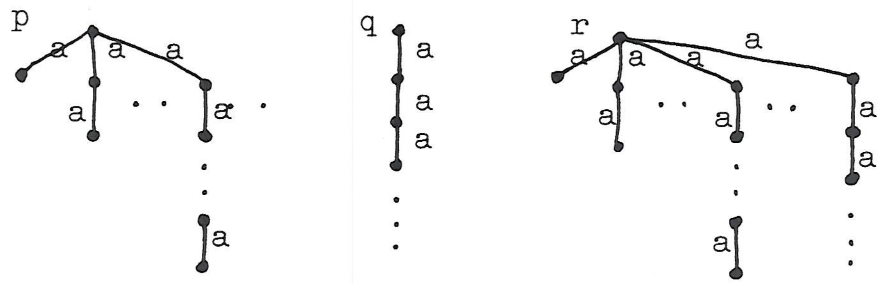
<u>Proposition 2.1-22:</u> If $p \sim q$ then $p \simeq q$.

<u>Proof:</u> $p \sim q$ iff there exists a bisimulation B containing $(p,q)$. Since obviously $\mathbb{B}(R) \subseteq \mathbb{S}(R)$ for all binary relations R, B is also a simulation. Thus $p \leqslant q$. Since $B^T$ is also a bisimulation and thus a simulation also $q \leqslant p$ and hence $p \simeq q$. □

Besides being an equivalence, $\sim$ has been shown to be a congruence wrt. all of the standard CCS-constructions /Mil80/. Obviously this is an essential property if hierarchic development of systems is to be possible. From the results of next chapter it will follow that $\sim$ indeed is a congruence wrt. any "natural" construction.

In Robin Milner's original work on CCS /Mil80/, $\leqslant$ and $\sim$ were defined as $\leqslant = \bigcap_{n \in \omega} \mathbb{S}^n(Pr^2)$ and $\sim = \bigcap_{n \in \omega} \mathbb{B}^n(Pr^2)$. However, unless $\mathbb{P}$ is image-finite, neither $\leqslant$ nor $\sim$ will in general be fixed-points if these definitions are used. The definitions given here in terms of simulations and bisimulations are due to David Park /P81B/ and - besides defining fixed-points - have the distinct advantages of providing useful proof techniques. Obviously the originally suggested definitions of $\leqslant$ and $\sim$ yield coarser relations than the versions suggested by David Park.

Example 2.1-23: Let p, q and r be processes with the following behaviour:




i.e $p=\sum_{n\varepsilon\omega} a^n$, $q=a^{\omega}$ and $r=p+q$. Then it is easily verified that for all $n\varepsilon\omega$, $q\lesssim^n p$ and $r\sim^n p$ where $\lesssim^n=\mathbb{S}^n(Pr^2)$ and $\sim^n=\mathbb{B}^n(Pr^2)$. However, $q\not\lesssim p$ and $r\not\sim p$. For the former assume namely that $q\lesssim p$. Then for some $k\varepsilon\omega$, $a^{\omega}\lesssim a^k$. But this implies that for all $n\varepsilon\omega$, $a^{\omega}\lesssim^n a^k$ which is false when $n>k$. A similar argument applies in the latter case. □

## 2.1.3 Modal Characterizations.

Matthew Hennessy and Robin Milner showed in /HenMil83/ that both $\lesssim$ and $\sim$ can alternatively be characterized by identifying a process with the properties it enjoys. For image-finite processes the relevant properties are formulas from the following modal languages: Let the language $\underline{M}$ (of formulas) be the least set such that:

(i) $Tr\,\varepsilon\, M$
(ii) $F\wedge G\,\varepsilon\, M$ whenever $F,G\,\varepsilon\, M$
(iii) $\neg F\,\varepsilon\, M$ whenever $F\,\varepsilon\, M$
(iv) $\langle a\rangle F\,\varepsilon\, M$ whenever $a\,\varepsilon\, Act$ and $F\,\varepsilon\, M$

Let $\underline{L}$ be the sublanguage of M consisting of the formulas not containing $\neg$. In /HenMil83/ the authors define a satisfaction relation $\models\subseteq Pr\times M$ as the least relation such that:

(i) $p\models Tr$ for $p\varepsilon Pr$
(ii) $p\models F\wedge G$ iff $p\models F$ and $p\models G$

(iii) $p \models \neg F$ iff $p \not\models F$

(iv) $p \models \langle a \rangle F$ iff $\exists p'.\ p \xrightarrow{a} p'\ \&\ p' \models F$

Now define for $p \varepsilon Pr$ the following two sets:

$$M(p) = \{F \varepsilon M \mid p \models F\} \quad \text{and} \quad L(p) = \{F \varepsilon L \mid p \models F\}$$

Then $\leqslant$ and $\sim$ have the following characterizations:

<u>Lemma 2.1-24:</u> If $\mathbb{P}$ is image-finite then:

(i) $p \sim q$ iff $M(p) = M(q)$

(ii) $p \leqslant q$ iff $L(p) \subseteq L(q)$

<u>Proof:</u> See /HenMil83/. □

By extending the modal languages with an <u>infinite</u> conjunction the above modal characterizations can be shown to hold for image-infinite process systems as well, /Mil84/.

Recently, complete proof systems for correctness assertions of the form $p \models F$ have been given for various subsets and variations of CCS /St83,St84,St85,W85,W85B/, with special emphasis on obtaining compositional proof systems. In the next chapter we will indicate how complete compositional proof systems for new languages could be obtained.

## 2.2 PARAMETERIZED BISIMULATION

The previous section shows us that $\sim$ is a property-generated equivalence. As such we can apply the general procedure suggested in the previous chapter to obtain our first parameterized version of $\sim$: as parameters we use sets of modal properties from M and for $A \subseteq M$, $\sim_A$ is simply defined as:

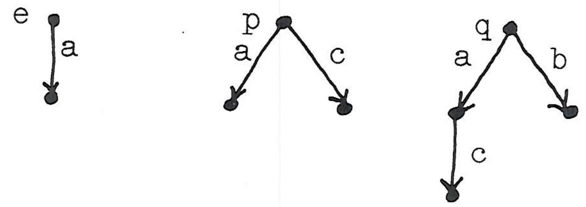
$$p \sim_A q \quad \text{iff} \quad M(p) \cap A \;=\; M(q) \cap A$$

In this section we shall define a parameterized version of $\sim$ based entirely on operational considerations similar to the definitions of $\lesssim$ and $\sim$ in 2.1-5 and 2.1-15. The operational definition will give us a simple and elegant proof technique similar to the proof techniques for $\lesssim$ and $\sim$. In the next section it will be demonstrated that this parameterized version of $\sim$ agrees with the above parameterized version of $\sim$ based on subsets of M as parameters.

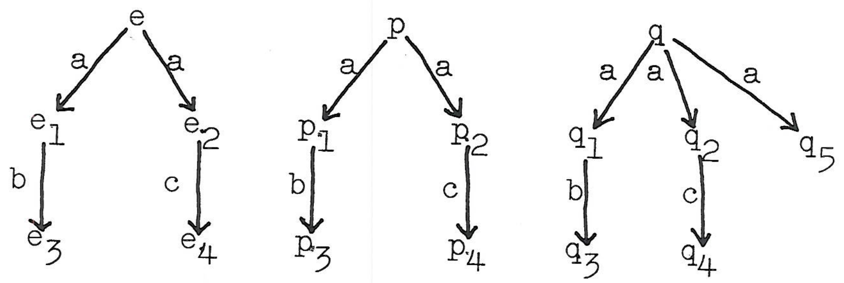
Following our initial motivation from chapter 1, $\sim$ is to be parameterized with (partial) information about contexts so that proofs of interchangability of processes can be simplified. For this purpose we shall introduce the notion of <u>environments</u> as a mean of representing such partial information about what behaviour (of an inner process) a context is able to "explore".

Operationally we take the view that an environment is an object with the ability to <u>consume</u> actions produced by an inner process. However, an environment's ability to consume actions might be limited, so if $p \xrightarrow{a} p'$ but e is an environment which cannot consume the action a, then the derivation $p \xrightarrow{a} p'$ will never be considered when p is executed in e. Similar to the assumption that a process can change after having produced (performed) an action we shall assume that an environment may change after having consumed an action. Thus environments and their behaviour can be described by a labelled transition system $\mathbb{E} = (Env, Act, \Rightarrow)$, where Env is the set of environments, Act is the set of actions (identical to the set of actions used in the transition system of processes) and $\Rightarrow$ is a subset of $Env \times Act \times Env$ called the consumption relation. $e \overset{a}{\Rightarrow} e'$ is to be read: "e may consume the action a and in doing so become the environment e'".

Let us now approach the question of how to parameterize $\sim$ with environments. Let e be an environment and let p and q be processes with behaviours given by the following:




In the environment e only a-actions can be consumed and after the consumption of one a-action e will change into an environment which is capable of consuming no actions at all. It therefore seems natural to expect p and q to be equivalent in e, i.e. $p \sim_e q$. As the next example let us consider the following slightly more complicated behaviours:




In order to determine whether $p \sim_e q$ we consider in turn all the possible ways e can consume an action. Let us consider the one consumption $e \overset{a}{\Rightarrow} e1$. For this particular consumption only a-derivatives of p and q will be examined. However, in order for $p \sim_e q$ to hold, for each a-derivative $q'$ of q ($q_5$ say) p must have a matching a-derivative $p'$ (here $p_2$) in the sense that $p' \sim_{e1} q'$. Similarly q must have a match (under e1) for each a-derivative of p.

Following this procedure the reader should be able to convince herself that p and q ought to be equivalent in e. Similarly, it can be argued that p and q should be distinguished in the following environment f:

$$\begin{tikzcd} & f \arrow[d, "a"] & \\ & f_1 \arrow[dl, "c"'] \arrow[dr, "b"] & \\ f_2 & & f_3 \end{tikzcd}$$

To satisfy the intuition indicated above we define a parameterized version of $\sim$ such that two processes, p and q, are considered equivalent in an environment e if they have the same set of potential first actions <u>that can be consumed by e</u> and they remain having equal potentiality during the course of execution <u>under all environment changes of e</u>. More formally we define the parameterized version of $\sim$ as follows:

<u>Definition 2.2-1:</u> Let $\mathbb{E} = (Env, Act, \Rightarrow)$ be an environment system. Then an <u>$\mathbb{E}$-parameterized bisimulation</u>, R, is an Env-indexed family of binary relations, $R_e \subseteq Pr^2$ for $e \in Env$, such that whenever $pR_e q$ the following holds:

For all $a \in Act$ if $e \overset{a}{\Rightarrow} e'$ then

$$\begin{array}{ll} \text{(i)} & p \xrightarrow{a} p' \Rightarrow \exists q'.\ q \xrightarrow{a} q' \ \&\ p' R_{e'} q' \qquad (*) \\ \text{(ii)} & q \xrightarrow{a} q' \Rightarrow \exists p'.\ p \xrightarrow{a} p' \ \&\ p' R_{e'} q' \end{array}$$

Two processes p and q are said to be equivalent in an environment e iff there exists an $\mathbb{E}$-parameterized bisimulation, R, such that $pR_e q$. In this case we write $p \sim_e q$. □

Since we shall be dealing with Env-indexed families and operations on such extensively in the following we adopt the following convenient notations. For Env-indexed families R and S let:

- $R \subseteq S$ iff for all $e \in Env$, $R_e \subseteq S_e$.
- $R \cap S$ is the Env-indexed family with

$(R \cap S)_e = R_e \cap S_e$.

- $R \cup S$ is the Env-indexed family with $(R \cup S)_e = R_e \cup S_e$.

Now, for R an Env-indexed family of binary relations over Pr, let $\mathbb{B}(R)$ be the Env-indexed family of binary relations over Pr such that $\mathbb{B}(R)_e$ is the set of pairs (p,q) satisfying (*) above. Then the following properties hold:

Proposition 2.2-2: An Env-indexed family R is an $\mathbb{E}$-parameterized bisimulation iff $R \subseteq \mathbb{B}(R)$. □

Proposition 2.2-3: $\mathbb{B}$ is a monotonic endofunction on the complete lattice of Env-indexed families of binary relations over Pr (ordered by componentwise inclusion). □

Then, using the standard fixed-point result /Ta55/, we get:

Proposition 2.2-4: $\mathbb{B}$ has a maximal fixed-point given as $\cup\{R \mid R \subseteq \mathbb{B}(R)\}$. Moreover this maximal fixed-point equals the Env-indexed family $\{\sim_e \mid e \in \mathrm{Env}\}$. □

Proposition 2.2-5: For all $e \in \mathrm{Env}$, $\sim_e$ is an equivalence relation.

Proof: Show that the Env-indexed family of relations Id, with $\mathrm{Id}_e$ being the identity relation on Pr, is an $\mathbb{E}$-parameterized bisimulation. Show that composition and converse of $\mathbb{E}$-parameterized bisimulations (composition and converse taken componentwise) are $\mathbb{E}$-parameterized bisimulations. The proposition will then follow from the definition of parameterized bisimulation equivalence. □

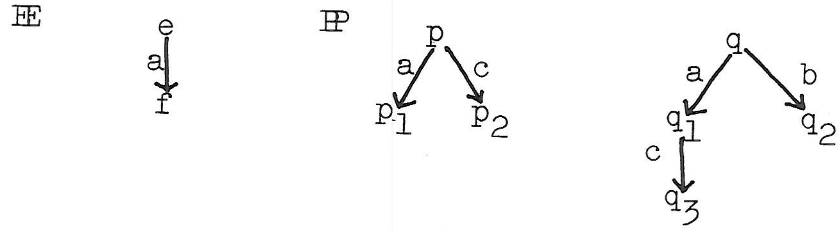
As expected in chapter 1, $\sim_e$ is for all environments e a weaker (and thus perhaps easier to prove) equivalence than the original (unparameterized) bisimulation equivalence:

Proposition 2.2-6: For all $e \in Env$ and all $p, q \in Pr$, if $p \sim q$ then also $p \sim_e q$.

Proof: Take for all $e \in Env$, $R_e = \sim$. Then $R$ is an $\mathbb{E}$-parameterized bisimulation. □

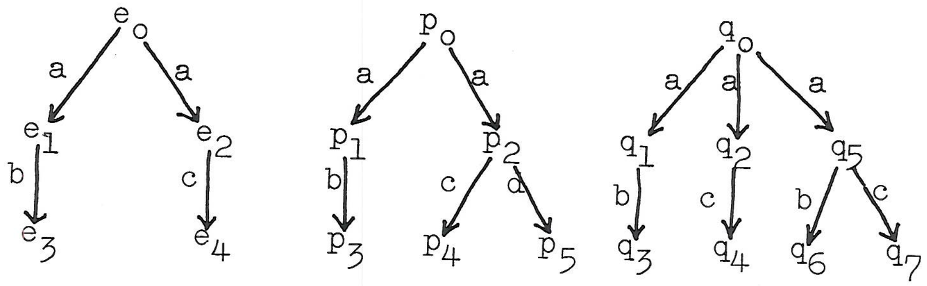
Note that proposition 2.2-4 provides us with a useful proof technique: to show that $p \sim_e q$ simply find an $\mathbb{E}$-parameterized bisimulation, $R$, such that $pR_eq$.

Example 2.2-7: Let us verify that our initial expectation is fulfilled. So let $\mathbb{E}$ and $\mathbb{P}$ be given by the diagrams below:




Then the Env-indexed family with $R_e = \{(p,q)\}$ and $R_f = \{(p_1,q_1)\}$ is a parameterized bisimulation. Thus, as expected, $p \sim_e q$. □

Example 2.2-8: Let $\mathbb{E}$ and $\mathbb{P}$ be given by the diagrams below:




Then the Env-indexed family R with:

$$R_{eo} = \{(p_o,q_o)\} \qquad R_{e3} = \{(p_3,q_6),(p_3,q_3)\}$$
$$R_{e1} = \{(p_1,q_1),(p_2,q_2),(p_1,q_5)\} \qquad R_{e4} = \{(p_4,q_7),(p_4,q_4)\}$$
$$R_{e2} = \{(p_2,q_2),(p_1,q_1),(p_2,q_5)\}$$

is a parameterized bisimulation. Thus $p_o \sim_{eo} q_o$. Note, that $p_o \not\sim q_o$. □

To insure anticontinuity of $\mathbb{B}$ only image-finiteness of the process system $\mathbb{P}$ is required:

Proposition 2.2-9: If $\mathbb{P}$ is image-finite then $\mathbb{B}$ is anticontinuous.

Proof: Let $R_1 \supseteq R_2 \supseteq \ldots \supseteq R_n \supseteq \ldots$ be a decreasing sequence of Env-indexed families. We must prove $\mathbb{B}(\bigcap_n R_n) = \bigcap_n \mathbb{B}(R_n)$. The "$\subseteq$"-direction follows directly from monotonicity of $\mathbb{B}$ and $\bigcap_n R_n \subseteq R_i$ for all $i \varepsilon \omega$. For the "$\supseteq$"-direction let $(p,q) \varepsilon \left[\bigcap_n \mathbb{B}(R_n)\right]_e$. We must show $(p,q) \varepsilon \left[\mathbb{B}(\bigcap_n R_n)\right]_e$. So let $e \overset{a}{\Longrightarrow} e'$ and $p \xrightarrow{a} p'$. We must find a matching move for $q$ such that $(p', q') \varepsilon \left[\bigcap_n R_n\right]_{e'} = \bigcap_n \left[(R_n)_{e'}\right]$. Now, $(p,q) \varepsilon \left[\bigcap_n \mathbb{B}(R_n)\right]_e$ iff for all $n \varepsilon \omega$ $(p,q) \varepsilon \mathbb{B}(R_n)_e$. Thus, for all $n \varepsilon \omega$ there exists a $q_n$ sucht that $q \xrightarrow{a} q_n$ and $(p', q_n) \varepsilon (R_n)_{e'}$. Under the assumption of $\mathbb{P}$ being image-finite there exists a $q'$ such that $q \xrightarrow{a} q'$ and $(p', q') \varepsilon (R_n)_{e'}$ for infinitely many n. Since $(R_n)_{e'}$ is decreasing in n, $(p', q') \varepsilon (R_n)_{e'}$ for all n and thus $(p', q') \varepsilon \bigcap_n \left[(R_n)_{e'}\right]$. By symmetry $(p,q) \varepsilon \left[\mathbb{B}(\bigcap_n R_n)\right]_{e'}$. □

Corollary 2.2-10: If $\mathbb{P}$ is image-finite then $\bigcap_{n \varepsilon \omega} \sim^n$ is the maximal fixed-point of $\mathbb{B}$, where for all $e \varepsilon$Env, $(\sim^0)_e = Pr^2$ and for $n \varepsilon \omega$, $\sim^{n+1} = \mathbb{B}(\sim^n)$. □

A particularly simple environment system is that of <u>language environments</u>, $\mathbb{L}$, consisting of (all) <u>deterministic</u> environments.

Definition 2.2-11: $\mathbb{L} = (\mathcal{P}(Act^*), Act, \Longrightarrow)$ is the labelled transition system, where $\Longrightarrow$ is the smallest relation satisfying for all $L \subseteq Act^*$ and $a \varepsilon Act$:

$$\partial L/\partial a \neq \emptyset \quad \Rightarrow \quad L \overset{a}{\Longrightarrow} \partial L/\partial a$$

where $\partial L/\partial a = \{w \mid aw \varepsilon L\}$. □

Obviously a language environment has at most one derivative for any action, and is thus deterministic. Also:

Lemma 2.2-12: $\mathbb{L}$ is image-finite. $\square$

Now let for $L \subseteq Act^*$, $L^p$ denote the prefixed closure of $L$, i.e.:

$$u \in L^p \Leftrightarrow \exists v \in Act^*.\ uv \in L$$

then the following properties are easily shown to hold:

Lemma 2.2-13:

(i) $L^p = \emptyset \Leftrightarrow L = \emptyset$

(ii) $(-)^p$ is monotonic wrt. $\subseteq$.

(iii) $L \subseteq L^p$

(iv) $\partial(L^p)/\partial a = [\partial L/\partial a]^p$ $\square$

We can now give a simple characterization of simulation between language environments based on their prefixed closures:

Theorem 2.2-14: For language environments L and M: $L \lesssim M$ iff $L^p \subseteq M^p$.

Proof: "$\Leftarrow$": We show that $S = \{(L,M) \mid L^p \subseteq M^p\}$ is a simulation. So let $(L,M) \in S$ and assume $L \stackrel{a}{\Longrightarrow} L'$. Then $L' = \partial L/\partial a \neq \emptyset$. By lemma 2.2-13 (ii) and lemma 2.2-13 (iv), $\emptyset \neq (\partial L/\partial a)^p \subseteq (\partial M/\partial a)^p$ and hence by lemma 2.2-13 (i), $\partial M/\partial a \neq \emptyset$. Thus, $M \stackrel{a}{\Longrightarrow} \partial M/\partial a$ and obviously $(\partial L/\partial a, \partial M/\partial a) \in S$.
"$\Rightarrow$": Assume $L^p \not\subseteq M^p$. Then for some string v, $v \in L^p$ but $v \notin M^p$. Since $M^p$ is prefixed closed also $vu \notin M^p$ for any extension, vu, of v. By induction on $|v|$ it is now easily shown that $L \stackrel{v}{\Longrightarrow}$ but $M \stackrel{v}{\not\Longrightarrow}$. Thus $L \not\lesssim M$.

$\square$

Recall from chapter 1 the definition of the discrimination ordering between environments:

$$e \sqsubseteq f \Leftrightarrow \sim_f \subseteq \sim_e$$

In some environment systems there are <u>minimal</u> and <u>maximal environments</u> wrt. $\sqsubseteq$:

<u>Lemma 2.2-15:</u>

(i) If e is an environment such that for all $a \varepsilon Act$, $e \overset{a}{\not\Longrightarrow}$ then e is minimal wrt. $\sqsubseteq$. Actually $\sim_e = Pr^2$.

(ii) If e is an environment such that for all $a \varepsilon Act$, $e \overset{a}{\Longrightarrow} e$ then e is maximal wrt. $\sqsubseteq$. Moreover $\sim_e = \sim$. We shall call any environment with this property a <u>universal environment</u>. □

As a corollary of this lemma it follows that $\emptyset$ is a minimal language environment and $Act^*$ is a universal language environment. We shall later, in section 2.4, vastly improve our knowledge about $\sqsubseteq$.

## 2.3 MODAL CHARACTERIZATION OF PARAMETERIZED BISIMULATION

In this section we shall present a modal characterization of the environment parameterized bisimulation equivalence pointed out to us by Colin Stirling. Let us first recall the standard characterization results for $\sim$ and $\lesssim$ given in section 2.1.3. Provided $\mathbb{P}$ is image-finite the following holds:

$$\text{(A)}\quad p \sim q \;\Leftrightarrow\; M(p) = M(q)$$
$$\text{(B)}\quad p \lesssim q \;\Leftrightarrow\; L(p) \subseteq L(q)$$

Now, $p \sim_e q$ means that p and q are equivalent when executed in the restricted environment e; i.e. only certain behaviours of p and q are being examined in e. From the characterization result (A) we expect a characterization of $\sim_e$ to be of the form:

$$p \sim_e q \;\Leftrightarrow\; M(p) \cap H(e) = M(q) \cap H(e)$$

where H(e) is a set of formulas corresponding to properties of processes which can be examined by e. From lemma 2.2-15 we know two things about H already. First, if e is the totally inactive environment, then $p \sim_e q$ holds for all p and q. Thus, we expect H(e) in this case to have the same effect on M(p) for all processes p. Secondly, if e is a universal environment, then $p \sim_e q$ iff $p \sim q$. Thus, we expect $H(e) = M$ in this case. We now offer H:

<u>Definition 2.3-1:</u> For $F \in L$ define $F^+ \subseteq M$ inductively as:

$$\text{(i)}\quad Tr^+ = \{Tr, \neg Tr\}$$
$$\text{(ii)}\quad (F \wedge G)^+ = \{C \wedge D, \neg(C \wedge D) \mid C \in F^+ \text{ and } D \in G^+\}$$
$$\text{(iii)}\quad (\langle a \rangle F)^+ = \{\langle a \rangle C, \neg \langle a \rangle C \mid C \in F^+\}$$ □

Thus, $F^+$ is simply the set of formulas derived from F by inserting arbitrary negations. We extend $(-)^+$ to sets of L-formulas by defining for $X \subseteq L$, $X^+ = \cup\{F^+ \mid F \in X\}$.

We can now state the Modal Characterization Theorem:

Theorem 2.3-2: Provided $\mathbb{P}$ is an image-finite transition system then for all $p, q \varepsilon Pr$ and $e \varepsilon Env$:

$$p \sim_e q \Leftrightarrow M(p) \cap L(e)^+ = M(q) \cap L(e)^+ \qquad \square$$

Hence, the set $H(e)$ is simply $L(e)^+$. Intuitively this seems correct since $L(e)^+$ only contains formulas based on what e can perform and thus detect. It also matchs the two things we know already. If e is the empty environment then $L(e)^+ = \{Tr, \neg Tr, Tr \wedge Tr, Tr \wedge \neg Tr, \ldots\}$ and if e is the universal environment then $L(e)=L$ and therefore clearly $L(e)^+=M$. We now outline the proof of theorem 2.3-2:

Proof: "$\Rightarrow$": Suppose $p \sim_e q$. We prove by induction on F that $F \varepsilon M(p) \cap L(e)^+$ iff $F \varepsilon M(q) \cap L(e)^+$. We consider only the cases $F = \neg G$ and $F = \langle a \rangle G$ leaving the two simpler cases to the reader:

$F = \neg G$: If $\neg G \varepsilon M(p) \cap L(e)^+$ an easy argument shows that $G \varepsilon L(e)^+$ and $G \not\varepsilon M(p)$. Thus $G \not\varepsilon M(p) \cap L(e)^+$ and therefore by the induction hypothesis $G \not\varepsilon M(q) \cap L(e)^+$. Since $G \varepsilon L(e)^+$ $G \not\varepsilon M(q)$ and thus $\neg G \varepsilon M(q)$. Hence, $\neg G \varepsilon M(q) \cap L(e)^+$.

$F = \langle a \rangle G$: If $\langle a \rangle G \varepsilon M(p) \cap L(e)^+$ an easy argument shows that there exists a $C \varepsilon L$ such that $\langle a \rangle C \varepsilon L(e)$ and $G \varepsilon C^+$. Hence, $e \overset{a}{\Longrightarrow} e'$ with $e' \models C$ for some $e'$. Also $p \overset{a}{\longrightarrow} p'$ with $p' \models G$ for some $p'$. However, $p \sim_e q$. Hence $q \overset{a}{\longrightarrow} q'$ with $p' \sim_{e'} q'$ for some $q'$. We know $G \varepsilon C^+ \subseteq L(e')^+$ and $G \varepsilon M(p')$. So by induction hypothesis $G \varepsilon M(q')$. Hence $\langle a \rangle G \varepsilon M(q)$ and finally $\langle a \rangle G \varepsilon M(q) \cap L(e)^+$.

"$\Leftarrow$": We show that the E-indexed family R with:

$$R_e = \{(p,q) \mid M(p) \cap L(e)^+ = M(q) \cap L(e)^+\}$$

is a parameterized bisimulation. Assume not. Then for some e, p and q $pR_eq$ but:

$$e \overset{a}{\Longrightarrow} e' \text{ and } p \overset{a}{\longrightarrow} p' \text{ and } \forall q'.\ q \overset{a}{\longrightarrow} q' \Rightarrow \neg(p' R_{e'} q')$$

Using the image-finiteness assumption for $\mathbb{P}$ let $\{q_1,\ldots,q_n\} = \{q' \mid q \xrightarrow{a} q'\}$. If this set is empty $\langle a\rangle Tr \varepsilon M(p) \cap L(e)^+$ but $\langle a\rangle Tr \not\varepsilon M(q) \cap L(e)^+$, contradicting $pR_e q$. Otherwise $\exists A_1,\ldots,A_n \varepsilon M$ and $\exists B_1,\ldots,B_n \varepsilon L$ such that:

(i) $\forall i.\ A_i \varepsilon B_i^+$

(ii) $\forall i.\ B_i \varepsilon L(e')$

(iii) $\forall i.\ p' \models A_i$ and $q_i \not\models A_i$

Clearly $B_1 \wedge \ldots \wedge B_n \varepsilon L(e')$ and by definition $A_1 \wedge \ldots \wedge A_n \varepsilon (B_1 \wedge \ldots \wedge B_n)^+$. We know $p \models \langle a\rangle (A_1 \wedge \ldots \wedge A_n)$ whereas $q \not\models \langle a\rangle (A_1 \wedge \ldots \wedge A_n)$. Moreover $\langle a\rangle (B_1 \wedge \ldots \wedge B_n) \varepsilon L(e)$ and $\langle a\rangle (A_1 \wedge \ldots \wedge A_n) \varepsilon (\langle a\rangle (B_1 \wedge \ldots \wedge B_n))^+$. However this contradicts $pR_e q$. $\square$

It is worth noticing that the above theorem establishes an agreement between the environment parameterized version of $\sim$ from definition 2.2-1 and the general idea from chapter 1 of parameterizing property generated equivalences with subsets of properties.

## 2.4 CHARACTERIZATION OF $\sqsubseteq$

In this and the next section we shall present two main theorems about the parameterized bisimulation equivalence.

The first theorem gives a characterization of the discrimination ordering under the assumption of image-finiteness. The characterization will be very useful when we axiomatize parameterized equivalence problems in chapter 4. Moreover, the characterization proved to be quite a technical challenge despite its obvious appearence: only after several months search a proof was found.

The second theorem shows constructively that for any two processes there exist a maximal (wrt. $\leqslant$) environment under which the two processes are equivalent. As such the theorem gives a way of reducing parameterized equivalence problems to problems of simulation and can therefore be used as the basis for an axiomatization of parameterized equivalence problems. It turns out that an (sufficiently rich) environment system forms a Heyting Algebra under $\leqslant$. Thus we can use environment systems as the interpretation for an intuitionistic propositional logic where the atomic propositions are equalities between processes.

### 2.4.1 Preliminary Definitions.

In order to enable the various constructions in the proofs of the two main theorems certain minimal structure on the transition systems involved is required.

Let $\mathbb{T} = (T, Act, \rightarrow)$ be a labelled transition system. We say that $\mathbb{T}$ is closed under action prefixing, summation resp. join if whenever $a \varepsilon Act$, $(t_i)_{i \varepsilon I}$ is some indexed family of states and $t$ is a state then there exist an element $a.t$, $\sum_{i \varepsilon I} t_i$ resp. $\&_{i \varepsilon I} t_i$ in $T$ with the operational semantics of $\mathbb{T}$ satisfying:

(a) $a.t \xrightarrow{b} t'$ <u>iff</u> $t'=t$ and $a=b$

(b) $\sum_{i\varepsilon I} t_i \xrightarrow{a} t'$ <u>iff</u> $\exists i\varepsilon I.\ t_i \xrightarrow{a} t'$

(c) $\underset{i\varepsilon I}{\&} t_i \xrightarrow{a} t'$ <u>iff</u> $\exists (t_i')_{i\varepsilon I}.$

$$\left(\forall i\varepsilon I.\ t_i \xrightarrow{a} t_i' \wedge t' = \underset{i\varepsilon I}{\&} t_i'\right)$$

We shall say that $\mathbb{T}$ is closed under finite sums (joins) if (b) ((c)) only holds for finite index sets, I. We shall use the following abbreviations:

$$\mathbb{O} = \sum_{i<0} t_i \qquad t_0 + t_1 = \sum_{i<2} t_i$$

$$U = \underset{i<0}{\&} t_i \qquad t_0 \,\&\, t_1 = \underset{i<2}{\&} t_i$$

By (b) we see that $\mathbb{O}$ has no actions at all, which means that $\mathbb{O}$ as an environment is minimal in the sense of lemma 2.2-18. By (c) it follows that $U \xrightarrow{a} U$ for all actions a. Thus U is a universal environment in the sense of lemma 2.2-18.

It turns out that $\Sigma$ and & are very special constructions wrt. the simulation ordering $\leqslant$.

<u>Lemma 2.4-1:</u> Let $\mathbb{T} = (T, Act, \longrightarrow)$ be closed under summation. Then $\sum_{i\varepsilon I} t_i$ is the least upper bound of $(t_i)_{i\varepsilon I}$ wrt. $\leqslant$.

<u>Proof:</u> We must prove that (a) $\forall i\varepsilon I.\ t_i \leqslant \sum_{i\varepsilon I} t_i$ and (b) $\left(\forall i\varepsilon I.\ t_i \leqslant t\right) \Rightarrow \sum_{i\varepsilon I} t_i \leqslant t$.

(a) follows from the fact that the set $S = \left\{(t_j, \sum_{i\varepsilon I} t_i) \mid j\varepsilon I\right\} \cup Id_T$ is a simulation. Similarly (b) follows from the fact that $S = \left\{(\sum_{i\varepsilon I} t_i, t) \mid \forall i\varepsilon I.\ t_i \leqslant t\right\} \cup \leqslant$ is a simulation. $\square$

<u>Lemma 2.4-2:</u> Let $\mathbb{T} = (T, Act, \longrightarrow)$ be closed under join. Then $\underset{i\varepsilon I}{\&} t_i$ is the greatest lower bound of $(t_i)_{i\varepsilon I}$ wrt. $\leqslant$.

Proof: We must show (a) $\forall i\varepsilon I.\ \underset{i\varepsilon I}{\&} t_i \leqslant t_i$ and (b) $(\forall i\varepsilon I.\ t \leqslant t_i) \Rightarrow t \leqslant \underset{i\varepsilon I}{\&} t_i$. (a) follows from the fact that $S = \{(\underset{i\varepsilon I}{\&} t_i, t_j) \mid j\varepsilon I\}$ is a simulation. (b) follows from the fact that $S = \{(t, \underset{i\varepsilon I}{\&} t_i) \mid \forall i\varepsilon I. t \leqslant t_i\}$ is a simulation. $\square$

All three constructions - action prefixing, summation and join - are monotonic wrt. $\leqslant$.

Lemma 2.4-3: Let $\mathbb{T}$ be closed under action prefixing, summation resp. join. Then whenever $t_i, s_i \varepsilon T$ for $i\varepsilon I$, $t, s\varepsilon T$ and $a\varepsilon Act$ the following holds:

$$\begin{array}{ll} \text{(i)} & t \leqslant s \Leftrightarrow a.t \leqslant a.s \\ \text{(ii)} & (\forall i\varepsilon I.\ t_i \leqslant s_i) \Rightarrow \sum_{i\varepsilon I} t_i \leqslant \sum_{i\varepsilon I} s_i \\ \text{(iii)} & (\forall i\varepsilon I.\ t_i \leqslant s_i) \Rightarrow \underset{i\varepsilon I}{\&} t_i \leqslant \underset{i\varepsilon I}{\&} s_i \end{array}$$

Proof: (i) follows directly from the operational semantics of action prefixing. (ii) and (iii) follows from lemma 2.4-1 and lemma 2.4-2. $\square$

Lemma 2.4-4: Let $\mathbb{P}$ be a process system and let $\mathbb{E}$ be an environment system closed under summation. Then:

$$[\forall i\varepsilon I.\ p \sim_{e_i} q] \Rightarrow p \sim_{\sum_{i\varepsilon I} e_i} q$$

Proof: Follows directly from the operational semantics of $\Sigma$. $\square$

From a later theorem the reverse direction will follow as a corollary. Thus if $\mathbb{E}$ is closed under summation $\sim_e$ will be continuous in e since:

$$\sim_{\sum_{i\varepsilon I} e_i} = \bigcap_{i\varepsilon I} \sim_{e_i}$$

Lemma 2.4-5: Let $\mathbb{P}$ be a process system closed under summation and let $\mathbb{E}$ be an environment system. Then:

$$\left[\forall i \varepsilon I.\ p_i \sim_e q_i\right] \Rightarrow \sum_{i \varepsilon I} p_i \sim_e \sum_{i \varepsilon I} q_i$$

$$\left[\forall i \varepsilon I.\ p_i \sim_e q_i\right] \Rightarrow \mathop{\&}_{i \varepsilon I} p_i \sim_e \mathop{\&}_{i \varepsilon I} q_i$$

Proof: Again directly from semantics of $\Sigma$ and &. □

For this lemma the reverse directions do not hold in general. The definitions of simulation and bisimulation (definitions 2.1-5 and 2.1-15) enables us only to compare (the behaviour of) processes or environments from the same transition system. However, the two notions are easily generalized so that comparison of processes or environments from different transition systems is possible.

Definition 2.4-6: Let $\mathbb{E} = (E, Act, \rightarrow_E)$ and $\mathbb{F} = (F, Act, \rightarrow_F)$ be two transition systems over the same set of actions, Act. A generalized simulation between $\mathbb{E}$ and $\mathbb{F}$ is a relation $R \subseteq E \times F$ such that whenever eRf and $a \varepsilon Act$ then:

$$\text{(i)} \quad e \xrightarrow{a}_E e' \Rightarrow \exists f'.\ f \xrightarrow{a}_F f' \ \&\ e'Rf'$$

If $R \subseteq E \times F$ is a generalized simulation such that eRf we write $e \leqslant f$. □

Definition 2.4-7: Let $\mathbb{E}$ and $\mathbb{F}$ be two transition systems over the same action set, Act. Then $R \subseteq E \times F$ is a generalized bisimulation between $\mathbb{E}$ and $\mathbb{F}$ if R is a generalized simulation between $\mathbb{E}$ and $\mathbb{F}$ and $R^T$ is a generalized simulation between $\mathbb{F}$ and $\mathbb{E}$. If $R \subseteq E \times F$ is a generalized bisimulation such that eRf we write $e \sim f$. □

Note that the notions of simulation (bisimulation) and generalized simulation (bisimulation) between $\mathbb{E}$ and $\mathbb{E}$ coincide. We shall therefore simply use the term simulation (bisimulation) instead of the more cumbersome generalized simulation (bisimulation). Using the new notion of generalized simulation we can relate the processes and environments in a parameterized equivalence:

Lemma 2.4-8: If $p \sim_e q$ and $e \leqslant q$ then $e \leqslant p$.

Proof: Show that $S = \{(e,p) \mid \exists q \varepsilon Pr.\ p \sim_e q \wedge e \leqslant q\}$ is a generalized simulation between $\mathbb{E}$ and $\mathbb{P}$. $\square$

Definition 2.4-9: Let $\mathbb{E} = (E, Act, \rightarrow_E)$ and $\mathbb{F} = (F, Act, \rightarrow_F)$ be two transition systems over the same action set, Act. Then $\mathbb{F}$ is an extension of $\mathbb{E}$ provided $E \subseteq F$ and $\rightarrow_F \cap (E \times Act \times E) = \rightarrow_E$. $\square$

Note if $\mathbb{F}$ is an extension of $\mathbb{E}$ then $Id_E$ is a generalized bisimulation between $\mathbb{E}$ and $\mathbb{F}$.

## 2.4.2 Characterization of $\sqsubseteq$.

Let $\mathbb{P}$ and $\mathbb{E}$ be the systems of processes and environments under consideration. Definition 2.2-1 then gives us a notion of equivalence between processes of $\mathbb{P}$ relative to environments of $\mathbb{E}$. Based on an environment's ability to distinguish between processes we can define the discrimination ordering $\sqsubseteq$ as:

$$e \sqsubseteq f \quad \Leftrightarrow \quad \sim_f \subseteq \sim_e$$

We shall in this section show that provided $\mathbb{E}$ is image-finite and $\mathbb{P}$ is sufficiently rich, $\sqsubseteq$ is nothing more than the simulation ordering $\leqslant$.

Already at this point certain things indicate that this is the right characterization of $\sqsubseteq$: As a first weak indication lemma 2.2-18, lemma 2.4-1 and lemma 2.4-2 shows that minimality and maximality wrt $\sqsubseteq$ and $\leqslant$ coincide. More substantial evidence is given by the modal characterization of parameterized equivalence in theorem 2.3-2 which shows that for image-finite process systems:

$$p \sim_e q \quad \text{iff} \quad M(p) \cap L(e)^+ = M(q) \cap L(e)^+$$

By the modal characterization of $\leqslant$ (lemma 2.1-24) we know that $e \leqslant f$ iff $L(e) \subseteq L(f)$ provided the environment system is image-finite. Since $(-)^+$ clearly is monotonic wrt. $\subseteq$, $e \leqslant f$ therefore implies $L(e)^+ \subseteq L(f)^+$ and hence - by the modal characterization above - that $p \sim_e q$ is more likely to hold than $p \sim_f q$ or equivalently $e \sqsubseteq f$. Thus for image-finite processes and environment systems $e \leqslant f$ implies $e \sqsubseteq f$. This result is easily generalized to image-infinite systems:

Theorem 2.4-10: $e \leqslant f$ implies $e \sqsubseteq f$.

Proof: Prove that the Env-indexed family R, with $R_e = \{(p,q) \mid \exists f.\ e \leqslant f \wedge p \sim_f q\}$ is an $\mathbb{E}$-parameterized bisimulation. Then if $e \leqslant f$ and $p \sim_f q$ we have $pR_e q$ and thus $p \sim_e q$. □

Proving the reverse direction however turns out to be far more involved and difficult as already hinted. Therefore, as a warming-up exercise, let us give a direct proof of the reverse implication in the simple case when the environment system is that of language environments, $\mathbb{L}$, see definition 2.2-11.

Obviously the system of processes $\mathbb{P}$ must be sufficiently rich (wrt. $\mathbb{L}$). If $\mathbb{P}$ only contains one process all language environments will be the same wrt. $\sqsubseteq$, but of course not wrt. $\leqslant$.

Theorem 2.4-11: Let $\mathbb{P}$ contain an inactive process $\mathbb{O}$ and be closed under action prefixing. Let L and M be two language environments. Then $L \sqsubseteq M$ implies $L \leqslant M$.

Proof: Assume $L \not\leqslant M$. By theorem 2.2-17 $L^p \not\subseteq M^p$, thus for some string $u \in L^p$ but for all extensions, uv, of u $uv \notin M^p$. Since $M^p$ is prefixed closed $u \neq \varepsilon$. Thus u is of the form wa for some $w \in Act^*$ and $a \in Act$. Define for $u \in Act^*$

the process $\bar{u}$ inductively as: $\bar{\varepsilon} = \mathbb{O}$ and $\overline{au} = a.\bar{u}$. Then - by induction on $|w|$ - it is easily shown that $\bar{w} \sim_M \overline{wa}$ but $\bar{w} \not\sim_L \overline{wa}$. Thus $L \not\sqsubseteq M$. ◻

Let us now return to the general problem, where $\mathbb{P}$ and $\mathbb{E}$ are arbitrary process and environment systems. We want to prove that whenever $e \sqsubseteq f$ then also $e \leq f$ or equivalently that $e \not\leq f$ implies $e \not\sqsubseteq f$, which is the same as:

(1) $e \not\leq f$ implies $\exists\, p,q \varepsilon Pr.\; p \sim_f q \wedge p \not\sim_e q$

Thus, we must construct or at least prove existence of a pair of processes , p and q, distinguished by e but not f. Assuming image-finiteness of $\mathbb{E}$, $e \not\leq f$ holds if and only if for some $n \varepsilon \omega$ $e \not\leq^n f$. Thus, we may attempt constructing the processes p and q required in (1) inductively in n:

For <u>n=0</u> no construction is needed since $e \not\leq^0 f$ is false.

If <u>$e \not\leq^1 f$</u> then $e \stackrel{a}{\Longrightarrow}$ and $f \stackrel{a}{\not\Longrightarrow}$ for some action a. Hence, by simply taking $p = a.\mathbb{O}$ and $q = \mathbb{O}$ the conclusion in (1) is fulfilled.

If <u>$e \not\leq^n f$</u> for some $n > 1$, then for some $a \varepsilon Act$ and $e' \varepsilon Env$, $e \stackrel{a}{\Longrightarrow} e'$ such that whenever $f \stackrel{a}{\Longrightarrow} f'$ then $e' \not\leq^{n-1} f'$.

$$\begin{tikzcd}
e \arrow[d, "a"'] & \not\leq^n & & f \arrow[dl, "a"'] \arrow[dr, "a"] & \\
e' & \not\leq^{n-1} & f_1 & \dots & f_k
\end{tikzcd}$$

Let $\{f_1, \dots, f_k\}$ be the set of all a-derivatives of f. Then we may apply the induction hypothesis to all the pairs $(e', f_1), \dots, (e', f_k)$ constructing k pairs of processes $(p_1, q_1), \dots, (p_k, q_k)$ such that $p_i \sim_{fi} q_i$ but $p_i \not\sim_{e'} q_i$ for all $i = 1..k$. The task is then to uniformly construct the required processes p and q distinguished by e but not f from the 2·k processes $p_1,\ldots,p_k,q_1,\ldots,q_k$. However, from the knowledge of $e \not\lesssim^n f$ and $e' \not\lesssim^{n-1} f_1,\ldots,e' \not\lesssim^{n-1} f_k$ alone, it seems impossible to find such a <u>uniform/general</u> construction, though we succeeded in finding applicable constructions for all the <u>instances</u> of e and f we considered.

Therefore, the construction has been divided into two stages: a <u>prestage</u> where e and f are transformed into two environments with a <u>stronger</u> relationship than merely $\not\lesssim$ and a <u>construction stage</u> where the two transformed environments are used as the basis of the construction of p and q. Let $\underline{P}$ be the predicate on pairs of environments which describes the desired relationship between the transformed environments. Assume $\underline{P}$ satisfies the following properties:

$$(2) \quad \underline{P}(e,f) \Rightarrow e \not\lesssim f$$
$$(3) \quad e \not\lesssim f \Rightarrow \exists\, e',f'.\ e' \lesssim e \wedge f \lesssim f' \wedge \underline{P}(e',f')$$
$$(4) \quad \underline{P}(e,f) \Rightarrow \exists\, p,q.\ p \sim_f q \wedge p \not\sim_e q$$

then we can conclude that (1) also holds:

> Let e and f be environments such that $e \not\lesssim f$. Then by (3) there exist environments $e'$ and $f'$ such that $e' \lesssim e$, $f \lesssim f'$ and $\underline{P}(e',f')$. Apply (4) to $e'$ and $f'$ gives processes p and q such that $p \sim_{f'} q$ and $p \not\sim_{e'} q$. However, since $e' \lesssim e$ and $f \lesssim f'$ and we already know $\lesssim \subseteq \sqsubseteq$ (theorem 2.4-10) also $p \sim_f q$ and $p \not\sim_e q$.

Note, that by $\lesssim \subseteq \sqsubseteq$, if (4) is to hold then $\underline{P}(e,f)$ implies $e \not\lesssim f$. So if $\underline{P}$ satisfies (3) and (4), (2) is automatically satisfied too.

In the above strategy the choice of the predicate $\underline{P}$ is obviously the key factor. On the one hand, we want $\underline{P}$ as strong as possible, in order to make the construction in

(4) as easy as possible. From past experience we know that we want $\underline{P}(e,f)$ to be stronger than simple $e \not\lessapprox f$. On the other hand $\underline{P}$ cannot be to strong since the transformation in (3) is to be possible too.

The present proof of (1) requires $\mathbb{E}$ to be image-finite. We shall later see what is required in order to extend the proof to image-infinite systems. Also $\mathbb{P}$ must obviously have a certain richness in order for (1) to hold. Thus we shall in the following assume that $\mathbb{E}$ is image-finite and that $\mathbb{P}$ is closed under action prefixing and finite sums. Also, for technical reasons we shall assume that $\mathbb{E}$ is closed under action prefixing and finite sums and that for all $e \varepsilon Env$ and $a \varepsilon Act$ there exist an environment $e_{-a} \varepsilon Env$ such that $e_{-a} \stackrel{b}{\Rightarrow} f$ iff $b \neq a$ and $e \stackrel{b}{\Rightarrow} f$. Note that $e_{-a} \stackrel{a}{\not\Rightarrow}$. Fortunately, an environment system can always be extended to a system with these properties, and clearly if (1) holds in the extended environment system it will be even more true in the original one.

Let us first state the definition of the predicate $\underline{P} \subseteq Env^2$:

<u>Definition 2.4-12:</u>

$\underline{P}_0(e,f)$ always false

$\underline{P}_n(e,f)$ iff

$\exists a \varepsilon Act.\ \exists e_0,..,e_{m-1},f_0,..,f_{m-1},g \varepsilon Env.$

(i) $e = a.(e_0+...+e_{m-1})$ ;

(ii) $f = a.f_0 + ... + a.f_{m-1} + g$ ;

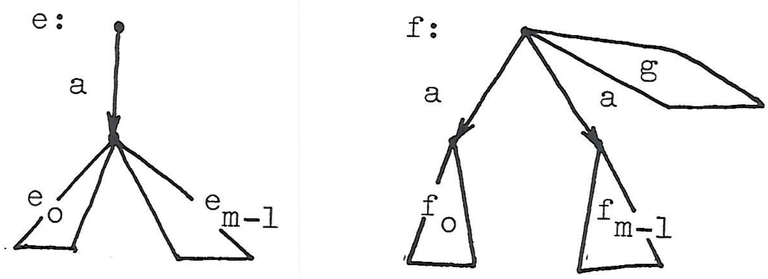
(iii) $g \stackrel{a}{\not\Rightarrow}$ ;

(iv) $\forall i<m. \exists k<n.\ \underline{P}_k(e_i,f_i)$ ;

(v) $\forall i,j<m.\ i \neq j \Rightarrow e_i \not\lessapprox e_j$ ;

$\underline{P}(e,f)$ iff $\exists n \geqslant 0.\ \underline{P}_n(e,f)$ □

Thus for $\underline{P}_n(e,f)$ to hold e and f must have the following form:




where the $e_i$'s are mutually incompatible under $\leqslant$, for all i $\underline{P}_k(e_i,f_i)$ holds for some $k<n$ and $g \not\xrightarrow{a}$ .

We state without proofs the following properties of $\underline{P}$.

<u>Lemma 2.4-13:</u> $\emptyset = \underline{P}_0 \subseteq \underline{P}_1 \subseteq \ \ldots \ \subseteq \underline{P}_n \subseteq \ \ldots$ □

<u>Lemma 2.4-14:</u> For all $n \varepsilon \omega$ and $e, f \varepsilon Env$:

$$\underline{P}_n(e,f) \Rightarrow e \not\leqslant^n f$$

<u>Proof:</u> By induction on n. □

<u>Lemma 2.4-15:</u> If $\underline{P}(e,f)$ then $e = a.e'$ for some $a \varepsilon Act$ and $e' \varepsilon Env$. □

We want to show that $\underline{P}$ enjoys the following two properties:

$$\text{(A)} \quad e \not\leqslant f \quad \Rightarrow \exists e', f'.\ e' \leqslant e \wedge f \leqslant f' \wedge \underline{P}(e', f')$$
$$\text{(B)} \quad \underline{P}(e,f) \Rightarrow \exists p, q.\ p \sim_f q \wedge p \not\sim_e q$$

Property (A):

In order to obtain property (A) we need to prove a stronger result (stronger due to clause (4) below):

Theorem 2.4-16: Let $e_0, f_0, \ldots, e_{m-1}, f_{m-1}$ be $m \geqslant 0$ pairs of environments such that:

$$\forall i<m.\ e_i \not\leqslant^n f_i$$

Then there exists $h \leqslant m$ pairs of environments $e'_0, f'_0, \ldots, e'_{h-1}, f'_{h-1}$ such that:

$$\begin{aligned}
&(1)\ \forall j<h.\ \underline{P}_n(e'_j, f'_j)\\
&(2)\ \forall j<h. \exists i<m.\ e'_j \leqslant e_i\\
&(3)\ \forall i<m. \exists j<h.\ f_i \leqslant f'_j\\
&(4)\ \forall i,j<h.\ i \neq j \Rightarrow e'_i \not\leqslant e'_j
\end{aligned}$$

□

Applying theorem 2.4-16 to a single pair of environments gives the following corollary from which property (A) trivially follows.

Corollary 2.4-17: Let e and f be environments such that $e \not\leqslant^n f$. Then there exists $e'$ and $f'$ such that $\underline{P}_n(e', f')$, $e' \leqslant e$ and $f \leqslant f'$.

□

Proof (of theorem 2.4-16): The proof is by induction on n with an inner induction on m:

Base n=0: Trivial since $e_i \not\leqslant^0 f_i$ is false.

Step: As our induction hypothesis we assume the theorem is true for all k<n. We prove the induction step using a subinduction on m.

Subbase m=0: Then $e_0, f_0, \ldots, e_{m-1}, f_{m-1}$ is the empty set. Take $e'_0, f'_0, \ldots, e'_{h-1}, f'_{h-1}$ to be the empty set as well trivially satisfies the theorem.

Subbase′ m=1: Let e, f be such that $e \not\leqslant^n f$. Then:

$$\exists a. \exists e'. \left(e \overset{a}{\Longrightarrow} e' \ \&\ \forall f'.\ f \overset{a}{\Longrightarrow} f'.\ e' \not\leqslant^{n-1} f'\right)$$

Let $\{f_0,\ldots,f_{k-1}\} = \{f' \mid f \xrightarrow{a} f'\}$ (using the image-finite property) then for all $i<k$, $e' \not\lesssim^{n-1} f_i$. Thus we can apply the induction hypothesis to the k pairs $e', f_0, \ldots e', f_{k-1}$ to obtain $h \leqslant k$ pairs $e_0^+, f_0^+, \ldots, e_{h-1}^+, f_{h-1}^+$ such that:

(a) $\forall i<h.\ \underline{P}_{n-1}(e_i^+, f_i^+)$

(b) $\forall i<h.\ e_i^+ \leqslant e'$

(c) $\forall i<k. \exists j<h.\ f_i \leqslant f_j^+$

(d) $\forall i,j<h.\ i \neq j \Rightarrow e_i^+ \not\leqslant e_j^+$

Now take:

$$e^+ = a.(e_0^+ + \ldots + e_{h-1}^+)$$

$$f^+ = a.f_0^+ + \ldots + a.f_{h-1}^+ + f_{-a}$$

then $e^+$ and $f^+$ satisfies (1)-(4) for e,f. Clearly $\underline{P}_n(e^+, f^+)$ by the definition of $e^+$ and $f^+$ and (a). (2) is $e^+ \leqslant e$ which holds by (b). (3) is $f \leqslant f^+$ which holds by (c) and the definition of $f^+$. (4) is trivial since we have only one pair.

<u>End Subbase'</u>

<u>Substep:</u> As our Sub-Induction Hypothesis we assume the theorem is true for $k \leqslant n$ when we have at most m-1 pairs of environments. As our Sub Induction Step we must prove the theorem true for $k \leqslant n$ when we have at most m pairs of environments. So let $e_0, f_0, \ldots, e_{m-1}, f_{m-1}$ be m pairs of environments such that:

$$\forall i<m.\ e_i \not\lesssim^n f_i$$

By the Sub Induction Hypothesis we can apply the theorem to $e_0, f_0, \ldots, e_{m-2}, f_{m-2}$ to obtain $h \leqslant m-1$ pairs of environments $e_0^+, f_0^+, \ldots, e_{h-1}^+, f_{h-1}^+$ such that:

(a) $\forall i<h.\ \underline{P}_n(e_i^+, f_i^+)$

(b) $\forall i<h.\ \exists j<m-1.\ e_i^+ \leqslant e_j$

(c) $\forall j<m-1.\ \exists i<h.\ f_j \leqslant f_i^+$

(d) $\forall i,j<h.\ i \neq j \Rightarrow e_i^+ \not\leqslant e_j^+$

We can also apply the theorem (using the subbase′) to the single pair $e_{m-1}, f_{m-1}$ to obtain a pair $e^+, f^+$ such that:

(e) $\underline{P}_n(e^+, f^+)$

(f) $e^+ \leqslant e_{m-1}$

(g) $f_{m-1} \leqslant f^+$

If $e^+$ does not simulate or is not simulated by any of the environments $e_0^+, \ldots, e_{h-1}^+$ then the set:

$$e_0^+, f_0^+, \ldots, e_{h-1}^+, f_{h-1}^+, e^+, f^+$$

will clearly make the theorem hold for $e_0, f_0, \ldots, e_{m-1}, f_{m-1}$. Otherwise assume $e^+$ is simulated by $e_0^+$ say. Since $\underline{P}_n(e_0^+, f_0^+)$ and $\underline{P}_n(e^+, f^+)$ lemma 2.4-14 and $e^+ \leqslant e_0^+$ gives:

$$e_0^+ \not\leqslant^n f_0^+ \qquad \text{and} \qquad e_0^+ \not\leqslant^n f^+$$

Since $e_0^+$ is of the form a.g (by lemma 2.4-15) we have:

$$e_0^+ \not\leqslant^n f_0^+ + f^+$$

Now, by the Sub-Induction Hypothesis we can apply the theorem to the $h \leqslant m-1$ pairs $e_0^+, f_0^+ + f^+, e_1^+, f_1^+, \ldots, e_{h-1}^+, f_{h-1}^+$ to obtain $p \leqslant h < m$ pairs:

$$e_0^{++}, f_0^{++}, \ldots, e_{p-1}^{++}, f_{p-1}^{++}$$

such that:

(h) $\forall i<p.\ \underline{P}_n(e_i^{++}, f_i^{++})$

(i) $\forall i<p.\ \exists j<h.\ e_i^{++} \leqslant e_j^+$

(j) $\exists i<p.\ f_0^+ + f^+ \leqslant f_i^{++}$ and

$\forall j.0<j<h.\ \exists i<p.\ f_j^+ \leqslant f_i^{++}$

(k) $\forall i,j<p.\ i \neq j \Rightarrow e_i^{++} \not\leqslant e_j^{++}$

We claim that the pairs $e_0^{++}, f_0^{++}, \dots, e_{p-1}^{++}, f_{p-1}^{++}$ will make the theorem hold for $e_0, f_0, \dots, e_{m-1}, f_{m-1}$. We only need to check (2) and (3) since (1)$\Leftrightarrow$(h) and (4)$\Leftrightarrow$(k). Now (2) follows from (i) and (b) and transitivity of $\leqslant$. (3) follows from (c) and (j) using transitivity of $\leqslant$ together with the fact $f_0^+ \leqslant f_0^+ + f^+$.

The case when $e^+$ simulates some $e_j^+$ is similar.

<u>End Substep.</u>

<u>End Step.</u> $\square$

<u>Property (B).</u>

We prove the following stronger theorem:

<u>Theorem 2.4-18:</u> If $\underline{P}_n(e,f)$ then there exists p and r such that:

(1) $p \sim_f p + r$

(2) $e \leqslant r$

(3) $e \not\leqslant p$

(4) $p \leqslant e$

(5) $r \leqslant e$

$\square$

Then property (B) is easily obtained as a corollary:

<u>Corollary 2.4-19:</u> If $\underline{P}(e,f)$ then there exists p and q such that $p \sim_f q$ but $p \not\sim_e q$.

<u>Proof:</u> $\underline{P}(e,f)$ implies $\underline{P}_n(e,f)$ for some $n \geqslant 0$. Thus theorem 2.4-18 gives p and r with properties (1)-(5). Now, taking $q = p + r$ will give the corollary. $p \sim_f q$ is simply (1) and (2) and (3) together with lemma 2.4-8 gives $p \not\sim_e q$.

$\square$

Proof of theorem 2.4-18: The proof is done by induction on n.

Base n=0: Trivial since $\underline{P}_0(e,f)$ is false.

Step: As induction hypothesis we assume the theorem is true for all $k<n$. We must prove the theorem true for n as well. So let e and f be environments such that $\underline{P}_n(e,f)$. Thus:

$$\exists\, a.\ \exists\, e_0,\ldots,e_{m-1},f_0,\ldots,f_{m-1},g.$$

$$\begin{array}{ll}
\text{(i)} & e = a.(e_0 + \ldots + e_{m-1}) \ ; \\
\text{(ii)} & f = a.f_0 + \ldots + a.f_{m-1} + g \ ; \\
\text{(iii)} & \forall i<m.\ \exists\, k<n.\ \underline{P}_k(e_i,f_i) \ ; \\
\text{(iv)} & g \overset{a}{\not\Rightarrow} \ ; \\
\text{(v)} & \forall i,j<m.\ i\neq j \ \Rightarrow\ e_i \not\leqslant e_j
\end{array}$$

By induction hypothesis there exists pairs $p_0,r_0,\ldots,p_{m-1},r_{m-1}$ such that:

$$\begin{array}{ll}
\text{(a)} & p_i \sim_{f_i} p_i + r_i \\
\text{(b)} & e_i \leqslant r_i \\
\text{(c)} & e_i \not\leqslant p_i \\
\text{(d)} & p_i \leqslant e_i \\
\text{(e)} & r_i \leqslant e_i
\end{array}$$

Now let for $i<m$ $q_i = p_i + r_i$. Then taking:

$$\begin{aligned}
p = {} & a.(p_0 + q_1 + \ldots + q_{m-1}) + \\
& a.(q_0 + p_1 + \ldots + q_{m-1}) + \\
& \quad \vdots \\
& a.(q_0 + q_1 + \ldots + p_{m-1}) \ ;
\end{aligned}$$

and $$r = a.(q_0 + q_1 + \ldots + q_{m-1}) \ ;$$

will make the theorem hold for e and f. To see this let us check that the properties (1)-(5) holds for p and r.

(1) $p \sim_f p+r$: The only way this could be false is by $f \overset{a}{\Rightarrow} f_i$ and $p + r \overset{a}{\Rightarrow} \sum_{j<m} q_j$. However:

$$p \xrightarrow{a} q_0 + \ldots + p_i + \ldots + q_{m-1}$$

will match $p + r'$s move, since $q_i \sim_{fi} p_i$ by (a). (and $p \sim_e q$ and $p' \sim_e q'$ implies $p + p' \sim_e q + q'$)

<u>(2) $e \leqslant r$ :</u> I.e. $a.(e_0 + \ldots + e_{m-1}) \leqslant a.(q_0 + \ldots + q_{m-1})$. This follows from (b) $(e_i \leqslant r_i)$ and $r_i \leqslant q_i$ .

<u>(3) $e \nleqslant p$:</u> If m=o then $e = a.\mathbb{O}$ and $p = \mathbb{O}$ and clearly $e \nleqslant p$. Otherwise we must prove that for all $j < m$:

$$e_0 + \ldots + e_{m-1} \nleqslant q_0 + \ldots + p_j + \ldots + q_{m-1}$$

This will follow from $e_j \nleqslant q_0 + \ldots + p_j + \ldots + q_{m-1}$ which, since $e_j$ has the form $a.e'_j$, will follow from:

(x) $\forall i < m.\ i \neq j \Rightarrow e_j \nleqslant q_i$

(y) $e_j \nleqslant p_j$

(y) is simply (c). To see (x) assume $e_j \leqslant q_i$ for some $i \neq j$. I.e. $e_j \leqslant p_i + r_i$. Then from (d) and (e) we have $e_j \leqslant e_i$ which contradicts $\underline{P}_n(e,f)$ clause (5).

<u>(4) $p \leqslant e$ :</u> Again if m=0 the clause follows easily. Otherwise we must show that for all $j < m$:

$$q_0 + \ldots + p_j + \ldots + q_{m-1} \leqslant e_0 + \ldots + e_{m-1}$$

However, this follows trivially since $p_i \leqslant e_i$ and $r_i \leqslant e_i$ by (d) and (e).

<u>(5) $r \leqslant e$ :</u> We must show that:

$$q_0 + \ldots + q_{m-1} \leqslant e_0 + \ldots + e_{m-1}$$

Again this follows from (d) and (e).

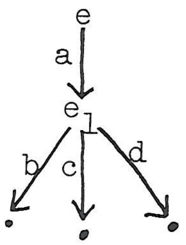
<u>End Step.</u> □

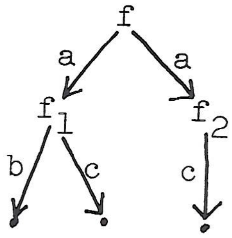
Having now proved that $\underline{P}$ enjoys the two properties (A) and (B) we can state the following <u>Main Theorem</u>:

Theorem 2.4-20: If $\mathbb{E}$ is image-finite and $\mathbb{P}$ is closed under action prefixing and finite summations then for all environments e and f:

$$e \sqsubseteq f \quad \Leftrightarrow \quad e \leqslant f\,.$$ □

Example 2.4-21: Let e and f be environments with the following behaviours:




Obviously, $e \not\leqslant^2 f$. We want to use the constructions of theorem 2.4-16 and theorem 214-18 to find processes, p and q, distinguished by e but not f.

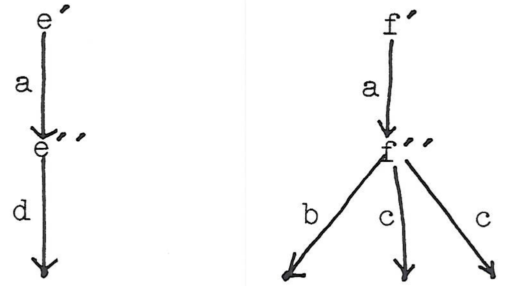
First we apply theorem 2.4-16 to find transformed environments, $e'$ and $f'$, such that $e' \leqslant e$, $f \leqslant f'$ and $\underline{P}_2(e',f')$. Obviously, $e_1 \not\leqslant^1 f_1$ and $e_1 \not\leqslant^1 f_2$, so we first apply theorem 2.4-16 to find transformed environments $e_1'', f_1'$ and $e_2'', f_2'$ such that $e_i'' \leqslant e_1$ and $f_i \leqslant f_i'$ for $i=1,2$.

For $i=1$ $e_1 \overset{d}{\Rightarrow} \mathbb{O}$ but $f_1 \overset{d}{\not\Rightarrow}$. Thus $e_1''=d.\mathbb{O}$ and $f_1' = f_1 = b.\mathbb{O} + c.\mathbb{O}$ are the transformed environments.

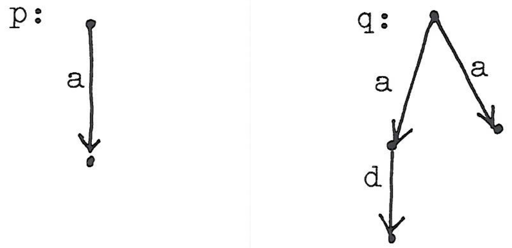
Similarly for $i=2$, $e_2''=d.\mathbb{O}$ and $f_2' = f_2 = c.\mathbb{O}$ are the transformed environments.

In order to obtain pairs of environments making theorem 2.4-16 true for $e_1,f_1;e_1,f_2$ we must combine $e_1'',f_1'$ and $e_2'',f_2'$. We note that $e_1'' \leqslant e_2''$ thus we must apply theorem 2.4-16 to the pair $e_2'', f_1' + f_2'$; i.e. $d.\mathbb{O}, b.\mathbb{O} + c.\mathbb{O} + c.\mathbb{O}$. This gives $d.\mathbb{O}, b.\mathbb{O} + c.\mathbb{O} + c.\mathbb{O}$ (no changes) as the pair of environments making 2.4-16 true for $e_1,f_1;e_1,f_2$.

To obtain a pair of environments making 2.4-16 true for e, f we apply the construction of the subbase', giving $a.d.\mathbb{O}, a.(b.\mathbb{O} + c.\mathbb{O} + c.\mathbb{O})$ as the transformed environments $e'$ and $f'$:

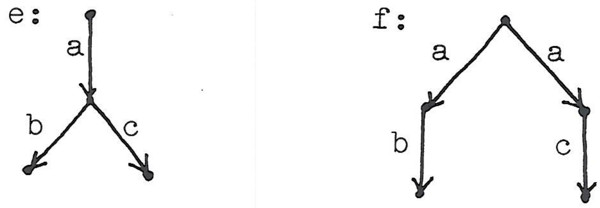



We can now apply theorem 2.4-18 to $e',f'$ in order to obtain a pair of processes distinguished by $e'$ (and hence e) but not $f'$ (and hence not f). For $e'',f''$ we find that $p'=\mathbb{O}$ and $r'=d.\mathbb{O}$ will make theorem 2.4-18 hold. Hence for $e',f'$ the pair $p=a.p'=a.\mathbb{O}$ and $r=a.(p'+r')=a.(\mathbb{O}+d.\mathbb{O})$ makes 2.4-18 hold. Thus the processes, p and q, distinguished by e but not f are:

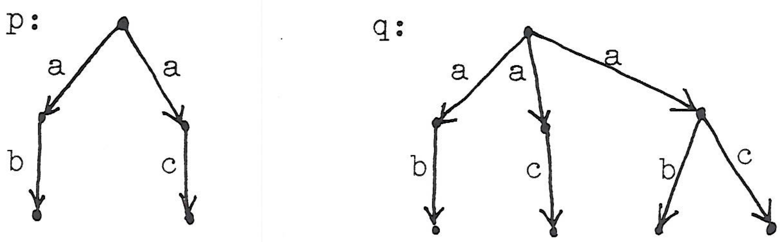



□

<u>Example 2.4-22:</u> Let e and f be environments with the following behaviours:




Obviously $e \not\leq^2 f$. Moreover $\underline{P}_2(e,f)$ so we can apply the construction in theorem 2.4-18 directly to obtain processes, p and q, distinguished by e but not f:

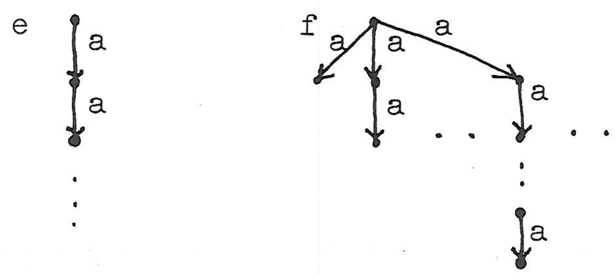



□

## 2.4.3 Extension to image-infinite case ?

A natural next step at this point would be to generalize the main theorem 2.4-20 to include the image-infinite cases as well. However, we shall show that as far as the present proof technique is concerned an extension is impossible. More precisely: we will show that even with a generalization of the predicate $\underline{P}$ to include image-infinite environments the property (A) fails to hold. I.e. there exist environments e and f such that $e \not\leq f$ but there are no transforms $e'$ and $f'$ such that $e' \leq e$, $f \leq f'$ and $\underline{P}(e',f')$. Thus either a new predicate $\underline{P}$ with the properties (A) and (B) or a totally new proof technique is needed. However, as far as this thesis is concerned the extension of the main theorem 2.4-20 to image-infinite cases is left as an open problem.

Let us first see why property (A) does not hold in the image-infinite case with the present definition of $\underline{P}$. For this purpose consider the following two environments:




From example 2.1-23 we know that $e \not\leq f$ but $e \leq^n f$ for all $n \varepsilon \omega$. Now assume $e'$ and $f'$ are transformed versions of e and f, i.e. $e' \leq e$, $f \leq f'$ and $\underline{P}(e',f')$. I.e. for some $n \varepsilon \omega$ , $\underline{P}_n(e',f')$ which by lemma 2.4-15 implies $e' \not\leq^n f'$. However, this contradicts $e' \leq e$, $f \leq f'$ and $e \leq^n f$ for all $n \varepsilon \omega$.

A possible reason for the above failure might be that for image-infinite environments the definition of $\underline{P}$ is not continuous and $\underline{P}$ is therefore not a fixed-point of its own definition. However - as we shall show - extending $\underline{P}$ to be a fixed-point of its definition will not make (A) hold for the above environments e and f.

Definition 2.4-23: Let $\mathbb{R} : \mathcal{P}(Env^2) \rightarrow \mathcal{P}(Env^2)$ be defined as:

$$
\begin{array}{l}
(e,f) \in \mathbb{R}(R) \quad \text{iff} \\
\quad \exists a \in Act.\ \exists (e_i, f_i)_{i \in I}.\ \exists g. \\
\qquad \text{(i)} \quad e = a.\sum_{i \in I} e_i \quad ; \\
\qquad \text{(ii)} \quad f = \sum_{i \in I} a.f_i + g \quad ; \\
\qquad \text{(iii)} \quad g \overset{a}{\not\Rightarrow} \quad ; \\
\qquad \text{(iv)} \quad \forall i \in I.\ (e_i, f_i) \in R \quad ; \\
\qquad \text{(v)} \quad \forall i,j \in I.\ i \neq j \Rightarrow e_i \not\leqslant e_j \quad ;
\end{array}
$$

□

It is easily shown that $\mathbb{R}$ is monotonic on $\mathcal{P}(Env^2)$ and as such has a least fixed-point, $\mu\mathbb{R}$. We shall use this least fixed-point as our generalized predicate $\underline{P}$.

Now, define the dual of $\mathbb{R}$, $\overline{\mathbb{R}}$, as $\overline{\mathbb{R}}(R) = \left(\mathbb{R}(R^c)\right)^c$. Using $\neg p \vee q \equiv p \Rightarrow q$, $\overline{\mathbb{R}}$ satisfies:

$$
\begin{array}{l}
(e,f) \in \overline{\mathbb{R}}(R) \quad \text{iff} \\
\quad \forall a \in Act.\ \forall (e_i, f_i)_{i \in I}.\ \forall g. \\
\qquad \underline{\text{If}} \quad \text{(i)} \quad e = a.\sum_{i \in I} e_i \quad ; \\
\qquad \qquad \text{(ii)} \quad f = \sum_{i \in I} a.f_i + g \quad ; \\
\qquad \qquad \text{(iii)} \quad g \overset{a}{\not\Rightarrow} \quad ; \\
\qquad \underline{\text{then}} \ \text{(iv)} \quad \exists i \in I.\ (e_i, f_i) \in R \quad \underline{\text{or}} \\
\qquad \qquad \text{(v)} \quad \exists i,j \in I.\ i \neq j \wedge e_i \leqslant e_j \quad ;
\end{array}
$$

Obviously $\overline{\mathbb{R}}$ is monotonic since $\mathbb{R}$ is. Also, if R is a fixed-point of $\mathbb{R}$, $R^c$ is a fixed-point of $\overline{\mathbb{R}}$. Thus if $\nu\overline{\mathbb{R}}$ is the maximal fixed-point of $\overline{\mathbb{R}}$ then $\nu\overline{\mathbb{R}} = (\mu\mathbb{R})^c$. Note, since $\mu\mathbb{R}$ is a least (pre) fixed-point, if $\mathbb{R}(R) \subseteq R$ then $\mu\mathbb{R} \subseteq R$. Also, since $\nu\overline{\mathbb{R}}$ is a maximal (post) fixed-point, if $R \subseteq \overline{\mathbb{R}}(R)$ then $R \subseteq \nu\overline{\mathbb{R}}$.

In order to show that the environments $a^\omega$ and $\sum_{n\varepsilon\omega} a^n$ cannot be transformed into environments with the relationship $\underline{P}$ (i.e $\mu\overline{\mathbb{R}}$) we show the following lemmas:

<u>Lemma 2.4-24:</u> If $(e,f)\varepsilon\,\mu\overline{\mathbb{R}}$ then $e \not\leqslant f$.

<u>Proof:</u> This is equivalent to: if $e \leqslant f$ then $(e,f)\varepsilon\,\nu\overline{\mathbb{R}}$. Let $R = \{(e,f) \mid e \leqslant f\}$. We show that R is a postfixed-point of $\overline{\mathbb{R}}$. Thus let $(e,f)\varepsilon R$ and assume:

$$- \; e = a.\sum_{i\varepsilon I} e_i$$
$$- \; f = \sum_{i\varepsilon I} a.f_i + g$$
$$- \; g \overset{a}{\not\Rightarrow}$$

for some $a\varepsilon Act$, $(e_i,f_i)_{i\varepsilon I}$ and g. We must show that either:

$$- \; \exists i\varepsilon I.\ (e_i,f_i)\varepsilon R$$

or

$$- \; \exists i,j.\ i \neq j \wedge e_i \leqslant e_j$$

Since $e \leqslant f$ and $g \overset{a}{\not\Rightarrow}$ there must exist $i\varepsilon I$ such that $\sum_{i\varepsilon I} e_i \leqslant f_i$ and hence $e_i \leqslant f_i$. Thus $(e_i,f_i)\varepsilon R$. □

<u>Lemma 2.4-25:</u> For all f, $(a^\omega,f) \not\in \underline{P}$.

<u>Proof:</u> Since $\underline{P} = \mu\overline{\mathbb{R}}$ this is the same as for all f, $(a^\omega,f)\varepsilon\,\nu\overline{\mathbb{R}}$. This follows from the fact that $R = \{(a^\omega,f) \mid f\varepsilon Env\}$ is a postfixed-point of $\overline{\mathbb{R}}$. So let $(a^\omega,f)\varepsilon R$ and assume:

$$- \; a^\omega = a.\sum_{i\varepsilon I} e_i$$
$$- \; f = \sum_{i\varepsilon I} a.f_i + g$$
$$- \; g \overset{a}{\not\Rightarrow}$$

Obviosly $|I| = 1$ with $a^\omega = a.a^\omega$, $f = a.f' + g$ and $g \overset{a}{\not\Rightarrow}$.
Thus all we have to show is $(a^\omega,f')\in R$ which is trivial. □

Lemma 2.4-26: Assume $e \leqslant a^\omega$ and for some $f \in Env$ that $(e,f) \in \underline{P}$. Then for some $\lambda \in \omega+1$, $e = a^\lambda$.

Proof: The above is equivalent to: if $e \neq a^\lambda$ and $e \leqslant a^\omega$ then for all $f \in Env$, $(e,f) \in \nu\overline{\mathbb{R}}$. Thus we simply show that $R = \{(e,f) \mid f \in Env \wedge \forall\lambda \in \omega+1.\ e \neq a^\lambda \wedge e \leqslant a^\omega\}$ is a postfixed-point of $\overline{\mathbb{R}}$. Thus let $(e,f) \in R$ and assume:

- $e = b.\sum_{i \in I} e_i$
- $f = \sum_{i \in I} b.f_i + g$
- $g \overset{b}{\not\Rightarrow}$

for some $b \in Act$, $(e_i,f_i)_{i \in I}$ and $g$. We must show that either:

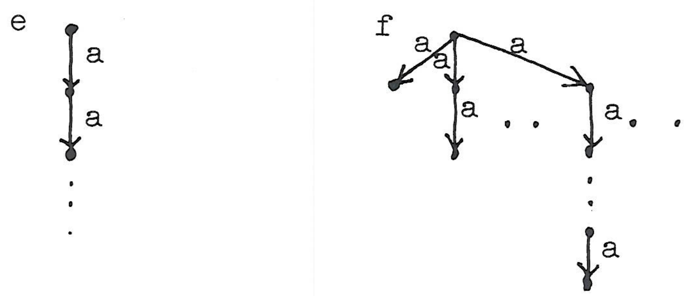
$$(1)\ \exists i \in I.\ (e_i,f_i) \in R$$

or

$$(2)\ \exists i,j \in I.\ i \neq j \wedge e_i \leqslant e_j$$

Obviously, since $e \leqslant a^\omega$, $b=a$. Assume that (1) does not hold. I.e. for all $i \in I$ there exist some $\lambda_i \in \omega+1$ such that $e_i = a^{\lambda_i}$. If $|I| = 0$ then $e=a$ and thus $(e,f) \notin R$ which is a contradiction. If $|I| = 1$ then $e=a.a^{\lambda_1}$ and therefore $(e,f) \notin R$. Again a contradiction. If $|I| > 1$ consider $e_1 = a^{\lambda_1}$ and $e_2 = a^{\lambda_2}$ then obviously $e_i \leqslant e_j$ iff $\lambda_i \leqslant \lambda_j$. Hence (2) holds. Thus either (1) or (2) holds. □

We are now ready to prove that there are no transforms corresponding to the two environments:




Theorem 2.4-27: Let $e = a^\omega$ and $f = \sum_{n \in \omega} a^n$. Then there are no environments $e'$ and $f'$ such that $e' \leqslant e$, $f \leqslant f'$ and $\underline{P}(e',f')$.

Proof: Assume $e'$ and $f'$ are such that $e' \leqslant e$, $f \leqslant f'$ and $\underline{P}(e',f')$. By lemma 2.4-26, $e'=a^\lambda$ for some $\lambda \in \omega+1$. Since $f \leqslant f'$ obviously $f' \xRightarrow{a^n}$ for all $n \in \omega$. Thus, since $\underline{P}(e',f')$ implies $e' \not\leqslant f'$ (lemma 2.4-24), $e'=a^\omega$. However, by lemma 2.4-25 $\underline{P}(a^\omega,f')$ does not hold for any $f'$. Thus, we have obtained a contradiction. □

Theorem 2.4-27 shows that the technique used in proving the Main Theorem 2.4-20 for the image-finite case does not generalize to the image-infinite cases. However, it does not show that the Main Theorem 2.4-20 is false in the image-infinite case. This is still an open problem (which the author conjectures to be true).

As a matter of fact, even though we cannot find transforms of the two environments $e=a^\omega$ and $f=\sum_{n\in\omega} a^n$ it is quite easy to find processes, p and q, distinguished by e but not f: take namely $p=\sum_{n\in\omega} a^n + a^\omega$ and $q=\sum_{n\in\omega} a^n$ then it is easily shown that p and q are identified under f but not e.

## 2.5 MAXIMAL ENVIRONMENT

We shall now show that for any two processes, p and q, there exists - in a sufficiently large environment system - a maximal (wrt. $\leqslant$) environment, $/p,q/$, under which p and q are equivalent. This means that a parameterized equivalence problem $p \sim_e q$ can be reduced to the simulation problem $e \leqslant /p,q/$. With the maximal environment construction, $/p,q/$, we can reformulate theorem 2.4-20 from the previous section as:

> whenever $e \not\leqslant f$ then there exist processes p and q such that $f \leqslant /p,q/$ but $e \not\leqslant /p,q/$.

Thus - provided the conditions of theorem 2.4-20 is met - this says that the maximal environments, $/p,q/$, are "dense" in $\mathbb{E}$.

Obviously, for $/p,q/$ to exist in general the environment system $\mathbb{E}$ needs to have a certain richness relative to the system of processes $\mathbb{P}$: let $\mathbb{E}$ consist of the four environments $\{U, a.\mathbb{0}, b.\mathbb{0}, \mathbb{0}\}$ and let $\mathbb{P}$ contain the two processes $p=a.a.\mathbb{0}$ and $q=a.\mathbb{0}$ (with the obvious operational semantics) then clearly both $a.\mathbb{0}$ and $b.\mathbb{0}$ identify p and q, whereas U does not. Thus, in $\mathbb{E}$ there is no maximal environment under which p and q are identified.

Let us now give an informal description of the behaviour of $/p,q/$. The description consists of three cases depending on the behaviour of p and q:

If $p \overset{a}{\not\rightarrow}$ and $q \overset{a}{\not\rightarrow}$ then we can safely let $/p,q/ \overset{a}{\Rightarrow}$ without distinguishing p and q. To obtain maximality we let $/p,q/ \overset{a}{\Longrightarrow} U$.

If $p \overset{a}{\rightarrow}$ and $q \overset{a}{\not\rightarrow}$ or $p \overset{a}{\not\rightarrow}$ and $q \overset{a}{\rightarrow}$ we cannot allow $/p,q/ \overset{a}{\Rightarrow}$ since this would lead to p and q being distinguished in $/p,q/$.

If both $p \overset{a}{\rightarrow}$ and $q \overset{a}{\rightarrow}$ we allow $/p,q/ \overset{a}{\Rightarrow}$. Clearly if only $/p,q/ \overset{a}{\Rightarrow} \mathbb{0}$ p and q will be identified in $/p,q/$. However this will in most cases not give maximality. Thus let us assume $/p,q/ \overset{a}{\Longrightarrow} e$ for some e. What bounds on e will ensure equivalence of p and q in $/p,q/$. Obviously, for the equivalence to hold there must exist a total surjective relation $\sigma \subseteq p_a \times q_a$ such that whenever $(p',q') \varepsilon \sigma$ then $p' \sim_e q'$. Thus for all $(p',q') \varepsilon \sigma$ we must have $e \leqslant /p',q'/$ or equivalently $e \leqslant \underset{(p',q')\varepsilon\sigma}{\&} /p',q'/$.
Thus, if $e \leqslant \underset{(p',q')\varepsilon\sigma}{\&} /p',q'/$ for some total surjective relation $\sigma \subseteq p_a \times q_a$, then $/p,q/ \overset{a}{\Longrightarrow} e$ will maintain equivalence of p and q. To obtain maximality of $/p,q/$ we let $/p,q/ \overset{a}{\Longrightarrow} \underset{(p',q')\varepsilon\sigma}{\&} /p',q'/$ for <u>all</u> total surjective relations $\sigma \subseteq p_a \times q_a$ (using lemma 2.4-1 and lemma 2.4-4 in a justification). We can now formally define the environment system in which these maximal environments

exist.

Definition 2.5-1: Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ be a system of processes. Then define the environment system $\mathbb{E}(\mathbb{P}) = (E_P, Act, \Longrightarrow)$ as the transition system where $E_P$ is the smallest set such that:

(i) $p, q \in Pr \Rightarrow /p,q/ \in E_P$

(ii) $\left(\forall i \in I.\ e_i \in E_P\right) \Rightarrow \underset{i \in I}{\&} e_i \in E_P$

(iii) $\left(\forall i \in I.\ e_i \in E_P\right) \Rightarrow \sum_{i \in I} e_i \in E_P$

(iv) $a \in Act,\ e \in E_P \Rightarrow a.e \in E_P$

and $\Longrightarrow$ is the smallest relation on $E_P \times Act \times E_P$ such that:

(a) $a.e \overset{a}{\Longrightarrow} e$

(b) $$\frac{\left[e_i \overset{a}{\Longrightarrow} e_i'\right]_{i \in I}}{\underset{i \in I}{\&} e_i \overset{a}{\Longrightarrow} \underset{i \in I}{\&} e_i'}$$

(c) $$\frac{e_i \overset{a}{\Longrightarrow} e_i'}{\sum_{i \in I} e_i \overset{a}{\Longrightarrow} e_i'} \quad ;\ i \in I$$

(d) $$\frac{p_a = \emptyset \qquad q_a = \emptyset}{/p,q/ \overset{a}{\Longrightarrow} U}$$

(e) $$\frac{p_a \neq \emptyset \qquad q_a \neq \emptyset \qquad \sigma \in p_a \longleftrightarrow q_a}{/p,q/ \overset{a}{\Longrightarrow} \underset{(p',q') \in \sigma}{\&} /p',q'/}$$

where for any two sets A and B $A \longleftrightarrow B$ is the set of all total surjective relations between A and B, i.e. $\sigma \in A \longleftrightarrow B$ iff $\sigma \subseteq A \times B$ and $\forall a \in A. \exists b \in B.\ (a,b) \in \sigma$ and $\forall b \in B. \exists a \in A.\ (a,b) \in \sigma$.

□

From the above definition $\mathbb{E}(\mathbb{P})$ is clearly seen to be closed under action prefixing, summation and join (se section 2.4.1). Also, if $\mathbb{P}$ is image-finite then $/p,q/$ is an image-finite environment for all processes p and q (since there are finitely many total and surjective relations between $p_a$ and $q_a$). $/p,q/$ is easily seen to satisfy the following:

<u>Proposition 2.5-2:</u>

$$/p,q/ \equiv \sum_{a:q_a=p_a=\emptyset} a.U \quad + \sum_{\substack{a:p_a\neq\emptyset \\ q_a\neq\emptyset}} \; \sum_{\sigma\varepsilon p_a \leftrightarrow q_a} a.\; \overset{\&}{(p', q')\varepsilon\sigma} /p',q'/$$

where $\equiv$ is the <u>direct equivalence</u> in the sense of /Mil80/. I.e. $e\equiv f$ iff $\forall a\varepsilon \mathrm{Act}.\forall g\varepsilon \mathrm{Env}.\ e\overset{a}{\Longrightarrow}g \Leftrightarrow f\overset{a}{\Longrightarrow}g$. □

We can now verify that $/p,q/$ indeed is a maximal environment identifying p and q. I.e. if e is an environment from <u>any</u> environment system such that $p\sim_e q$ then $e\leqslant/p,q/$, where $\leqslant$ is the generalized simulation of definition 2.4-6. First, however, let us show that p and q are actually identified in the environment $/p,q/$:

<u>Theorem 2.5-3:</u> $p\sim_{/p,q/}q$ .

<u>Proof:</u> We show that the family R with:

$$R_e = \{(p,q)\mid e\leqslant/p,q/\} \text{ for } e\varepsilon E_P$$

is an $\mathbb{E}(\mathbb{P})$-parameterized bisimulation. Thus let $(p,q)\varepsilon R_e$, $e\overset{a}{\Longrightarrow}f$ and $p\overset{a}{\longrightarrow}p'$. Since $e\leqslant/p,q/$ also $/p,q/\overset{a}{\Longrightarrow}$. Since $p\overset{a}{\Longrightarrow}$, $p_a\neq\emptyset$ and therefore also $q_a\neq\emptyset$. This means that $/p,q/$ only has a-moves caused by the (e)-rule. Thus for some $\sigma\varepsilon p_a\leftrightarrow q_a$, $/p,q/\overset{a}{\Longrightarrow} \overset{\&}{(p^+, q^+)\varepsilon\sigma}/p^+,q^+/$ with $f\leqslant \overset{\&}{(p^+, q^+)\varepsilon\sigma}/p^+,q^+/$. Since $\sigma$ is total and surjective $(p',q')\varepsilon\sigma$ for some $q'\varepsilon q_a$. We must show that $(p',q')\varepsilon R_f$ or equivalently that $f\leqslant/p',q'/$. However this is trivial since $\overset{\&}{(p^+,q^+)\varepsilon\sigma}/p^+,q^+/\leqslant/p',q'/$. □

Theorem 2.5-4: Let $\mathbb{E}$ be any environment system and let e be any environment of $\mathbb{E}$ such that $p \sim_e q$. Then $e \leqslant /p,q/$.

Proof: We show that the relation:

$$S = \Big\{ (e, \underset{i\varepsilon I}{\&} f_i) \mid \forall i \varepsilon I.\ f_i = U \text{ or } \big(f_i = /p_i, q_i/ \text{ and } p_i \sim_e q_i\big) \Big\}$$

is a simulation between $\mathbb{E}$ and $\mathbb{E}(\mathbb{P})$. Obviously if $p \sim_e q$ then $(e, /p,q/) \varepsilon S$ and thus – provided S is a simulation – $e \leqslant /p,q/$.

Let $(e, \underset{i\varepsilon I}{\&} f_i) \varepsilon S$ and let $e \overset{a}{\Rightarrow} e'$. Let J be the set of indices, j, of I such that $f_j = /p_j, q_j/$. Then we need to find a move $\underset{j\varepsilon J}{\&} f_j \overset{a}{\Rightarrow} g$ such that $(e', g) \varepsilon S$. Since for all $j \varepsilon J$, $p_j \sim_e q_j$, either $p_j \overset{a}{\rightarrow}$ and $q_j \overset{a}{\rightarrow}$ or $p_j \overset{a}{\not\rightarrow}$ and $q_j \overset{a}{\not\rightarrow}$. Thus we write J as $J' \cup J''$ where $J'$ is the subset of J such that the former is the case and $J''$ is the subset of J where the latter is the case. For $j \varepsilon J''$ we have by (d) $/p_j, q_j/ \overset{a}{\Rightarrow} U$. Thus we have reduced the problem to find a move $\underset{j\varepsilon J'}{\&} /p_j, q_j/ \overset{a}{\Rightarrow} g'$ such that $(e', g') \varepsilon S$. For $j \varepsilon J'$ there must exist some total surjective relation $\sigma_j \varepsilon (p_j)_a \leftrightarrow (q_j)_a$ such that whenever $(p_j', q_j') \varepsilon \sigma_j$ then $p_j' \sim_{e'} q_j'$. By (e):

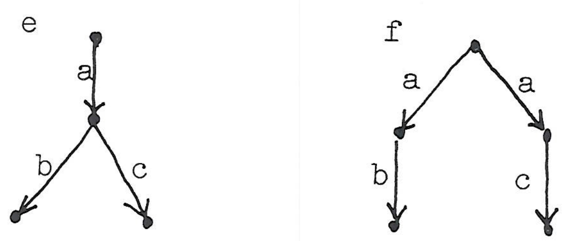
$$/p_j, q_j/ \overset{a}{\Rightarrow} \underset{\sigma_j}{\&} /p_j', q_j'/$$

and thus by the rule for join:

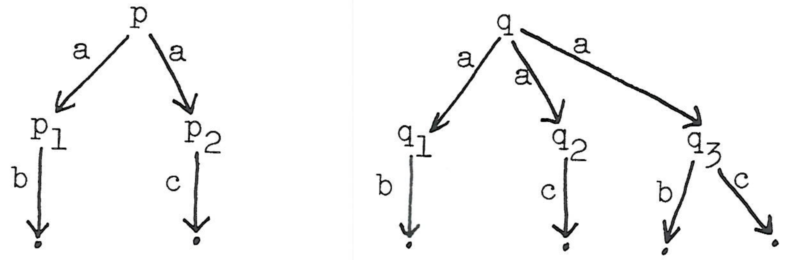
$$\underset{j\varepsilon J'}{\&} /p_j, q_j/ \overset{a}{\Rightarrow} \underset{j\varepsilon J'}{\&}\ \underset{\sigma_j}{\&} /p_j', q_j'/$$

which is the matching move. $\square$

Example 2.5-5: Recall example 2.4-22, where e and f are the environments:



and p and q are the processes:




We want to show that $p \sim_f q$ but $p \not\sim_e q$. By theorems 2.4-20, 2.5-4 and 2.4-10 we know that it is necessary and sufficient to show that $f \leqslant /p,q/$ and $e \not\leqslant /p,q/$. Let us therefore calculate $/p,q/$ using proposition 2.5-2. During this calculation we find:

$$/p_1,q_1/ \simeq U \qquad /p_2,q_1/ \simeq \{b,c\}^c.U$$
$$/p_1,q_2/ \simeq \{b,c\}^c.U \qquad /p_2,q_2/ \simeq U$$
$$/p_1,q_3/ \simeq \{c\}^c.U \qquad /p_2,q_3/ \simeq \{b\}^c.U$$

where for $m \subseteq Act$ and $e \varepsilon E_P$ $m.e$ is an abbreviation for $\sum_{a \varepsilon m} a.e$. It is then easily calculated that:

$$/p,q/ \simeq \{a\}^c.U + a.\{b\}^c.U + a.\{c\}^c.U$$

from which it is obvious that $e \not\leqslant /p,q/$ and $f \leqslant /p,q/$. □

We state without proof the following algebraic properties of $/p,q/$:

<u>Proposition 2.5-6:</u>

(i) $/a.p,b.q/ \simeq \begin{cases} \{a,b\}^c.U & ; a \neq b \\ \{a\}^c.U + a./p,q/ & ; \text{otherwise} \end{cases}$

(ii) $/p,p/ \simeq U$

(iii) $/p,q/ \simeq /q,p/$

(iv) $/p_1,q_1/ \ \& \ /p_2,q_2/ \leqslant /p_1+p_2,q_1+q_2/$

(v) $/p_1,q_1/ \ \& \ /p_2,q_2/ \leqslant /p_1\&p_2,q_1\&q_2/$ □

More complete laws than (iv) and (v) can be obtained by introducing sumforms.

Not only does the maximal environment construction provide a way of deciding parameterized equivalence problems it also allow us to consider more complex questions; e.g. the Horn Clause:

"Is it true that whenever
$p_1=q_1$ and ... and $p_n=q_n$
in an environment then also
$p=q$ "

is equivalent to:

$$/p_1,q_1/ \ \& \ \ldots \ \& \ /p_n,q_n/ \ \leqslant \ /p,q/$$

To deal with even more complex problems with possible nested implications we can extend $\mathbb{E}(\mathbb{P})$ to a <u>Heyting Algebra</u> (see /Go79,Da81/) by introducing an implication construction, $\rightarrow$ , being the right adjoint to &. We shall in the following briefly indicate how to extend $\mathbb{E}(\mathbb{P})$ and demonstrate its potential use. However, a more complete investigation is left as future work.

The extended environment system $\mathbb{E}(\mathbb{P})_{\rightarrow} = (E_P, Act, \Rightarrow)$ is obtained by adding an implication construct, $\rightarrow$. Thus we add the rule:

$$\text{(v)} \quad e,f \varepsilon E_P \quad \Rightarrow \quad (e \rightarrow f) \varepsilon E_P$$

The operational semantics of $(e \rightarrow f)$ is given by the following rules very similar to the rules (d) and (e) for /p,q/:

$$\text{(f)} \quad \frac{e_a = \emptyset}{(e \rightarrow f) \overset{a}{\Longrightarrow} U}$$

$$\text{(g)} \quad \frac{e_a \neq \emptyset \qquad f_a \neq \emptyset \qquad \tau \varepsilon e_a \longrightarrow f_a}{(e \rightarrow f) \overset{a}{\Longrightarrow} \underset{(e',f')\varepsilon\tau}{\&} (e' \rightarrow f')}$$

where for two sets A and B $A \longrightarrow B$ is the set of functions from A to B.

Similar to the proofs of theorem 2.5-3 and 2.5-4 it can be shown that $(e \to f)$ is the maximal environment wrt. $\leqslant$ such that:

$$e \,\&\, (e \to f) \leqslant f$$

Thus $\mathbb{E}(\mathbb{P})_{\to}$ is a Heyting Algebra with $\Phi$ as zero, & as conjuncion and $\to$ as the <u>relative pseudo-complement</u> (see /Go79/). As such the following (among many other) property holds:

$$e \leqslant f \quad \text{iff} \quad (e \to f) \simeq U$$

Define $\neg e = (e \to \Phi)$ then:

$$\neg U \simeq \Phi \simeq \neg(e \to e)$$

and $$\neg \Phi = (\Phi \to \Phi) \simeq U$$

We can now use $\mathbb{E}(\mathbb{P})_{\to}$ to "interpret" an intuitionistic propositional logic with connectives $\vee, \wedge, \supset$ and $\neg$ and with environments and equalities of processes as atomic propositions. The semantics of a sentence, $\varphi$, is an environment $[\![\varphi]\!]$ defined inductively as:

$$\begin{aligned}
[\![ e ]\!] &= e \\
[\![ p=q ]\!] &= /p,q/ \\
[\![ \varphi \wedge \psi ]\!] &= [\![\varphi]\!] \,\&\, [\![\psi]\!] \\
[\![ \varphi \vee \psi ]\!] &= [\![\varphi]\!] + [\![\psi]\!] \\
[\![ \varphi \supset \psi ]\!] &= [\![\varphi]\!] \to [\![\psi]\!] \\
[\![ \neg \varphi ]\!] &= [\![\varphi]\!] \to \Phi = \neg [\![\varphi]\!]
\end{aligned}$$

We say that a sentence $\varphi$ is <u>valid</u> in $\mathbb{E}(\mathbb{P})_{\to}$ iff $[\![\varphi]\!] \simeq U$ in which case we write $\models \varphi$. Thus, by the property above:

$$e \leqslant f \quad \text{iff} \quad \models e \supset f$$

Since $p \sim_e q$ iff $e \leqslant /p,q/$ we also have:

$$p \sim_e q \quad \text{iff} \quad \models e \supset (p=q)$$

Since $\mathbb{E}(\mathbb{P})_{\rightarrow}$ is a Heyting Algebra all the theorems of Intuitionistic Propositional Logic are valid in $\mathbb{E}(\mathbb{P})_{\rightarrow}$. Also, Modus Ponens preserves validity in $\mathbb{E}(\mathbb{P})_{\rightarrow}$ (if $[\![\varphi]\!] \simeq [\![\varphi \supset \psi]\!] \simeq$ U then by above property $[\![\varphi]\!] \leqslant [\![\psi]\!]$. Thus $[\![\psi]\!] \simeq$ U). Thus, we know that (among many other) the following are valid sentences:

$$\begin{array}{ll} \text{(i)} & \models \left[(\varphi \supset \psi) \wedge (\psi \supset \delta)\right] \supset (\varphi \supset \delta) \\ \text{(ii)} & \models \left[(\varphi \supset \psi) \wedge (\delta \supset \psi)\right] \supset (\varphi \vee \delta \supset \psi) \\ \text{(iii)} & \models \left[(\psi \supset \varphi) \wedge (\psi \supset \delta)\right] \supset (\psi \supset \varphi \wedge \delta) \end{array}$$

Let us indicate how these valid sentences can help us in formulating interesting properties of parameterized equivalence:

If we in (i) let $\varphi = e$, $\psi = f$ and $\delta = (p=q)$ we get the instance:

$$\models \left[(e \supset f) \wedge (f \supset (p=q))\right] \supset (e \supset (p=q))$$

which means that:

$$\boxed{e \leqslant f \quad \text{and} \quad p \sim_f q \quad \text{implies} \quad p \sim_e q}$$

In (ii) let $\varphi = e_1$, $\delta = e_2$ and $\psi = (p=q)$ then we get the instance:

$$\models \left[(e_1 \supset (p=q)) \wedge (e_2 \supset (p=q))\right] \supset ((e_1 \vee e_2) \supset (p=q))$$

which "translated" gives lemma 2.4-4:

$$\boxed{p \sim_{e_1} q \quad \text{and} \quad p \sim_{e_2} q \quad \text{implies} \quad p \sim_{e_1+e_2} q}$$

Since the reverse implication of (ii) is also a theorem of IPL we also have:

$$\boxed{p \sim_{e_1+e_2} q \quad \text{implies} \quad p \sim_{e_1} q \quad \text{and} \quad p \sim_{e_2} q}$$

In (iii) let $\psi = e$, $\varphi = (p_1 = q_1)$ and $\delta = (p_2 = q_2)$ then we get the instance:

$$\models \left[ (e \supset (p_1 = q_1)) \wedge (e \supset (p_2 = q_2)) \right] \supset \left( e \supset ((p_1 = q_1) \wedge (p_2 = q_2)) \right)$$

From proposition 2.5-6 (v) we know that:

$$\models \left[ (p_1 = q_1) \wedge (p_2 = q_2) \right] \supset (p_1 \& p_2 = q_1 \& q_2)$$

thus by (i):

$$\models \left[ (e \supset (p_1 = q_1)) \wedge (e \supset (p_2 = q_2)) \right] \supset \left( e \supset (p_1 \& p_2 = q_1 \& q_2) \right)$$

which means:

$$\boxed{p_1 \sim_e q_1 \quad \text{and} \quad p_2 \sim_e q_2 \quad \text{implies} \quad p_1 \& p_2 \sim_e q_1 \& q_2}$$

From proposition 2.5-6 (iv) we know that:

$$\models ((p_1 = q_1) \wedge (p_2 = q_2)) \supset (p_1 + p_2 = q_1 + q_2)$$

Thus:

$$\models \left[ (e \supset (p_1 = q_1)) \wedge (e \supset (p_2 = q_2)) \right] \supset \left( e \supset (p_1 + p_2 = q_1 + q_2) \right)$$

which says nothing more than lemma 2.4-5:

$$\boxed{p_1 \sim_e q_1 \quad \text{and} \quad p_2 \sim_e q_2 \quad \text{implies} \quad p_1 + p_2 \sim_e q_1 + q_2}$$

Obviously, none of the above derived properties of parameterized equivalence are new or could not have been just as easily established by other means. However, it might be that there are other theorems of IPL which would bring new insight into the parameterized bisimulation equivalence. This remains a subject for future work.

# CHAPTER 3

# CONTEXTS

So far we have put forward two parameterized versions of the bisimulation equivalence, $\sim$; one version - mentioned in section 2.2 - parameterized with subsets of the modal property domain M, and another version - studied at length in the last chapter - which uses environments as parameters. The Modal Characterization Theorem (theorem 2.3-2) demonstrates an agreement between the two versions in the sense that parameterizing $\sim$ with environments is the same as parameterizing $\sim$ with certain subsets of M.

Now recall the initial motivation from chapter 1 and especially the stepwise refinement method described in that chapter. According to this we want parameterized congruence laws, which for any given context C and information i (in our case the information i is given either as an environment or as a set of modal properties) will describe some information j such that for all processes p and q the following holds:

(1) $\qquad p \sim_j q \quad \Rightarrow \quad C[p] \sim_i C[q]$

Moreover, in order to make the proof $p \sim_j q$ as easy as possible we will prefer j to be as weak as possible with respect to the discrimination ordering (i.e. $\sim_j$ is as weak as possible).

As an analogy to Dijkstra's weakest precondition /Dij76/, we shall call the weakest information j satisfying (1) for the weakest inner information of i under C. The purpose of this chapter is to investigate the existence of such weakest inner information when the information used is either an environment or a set of modal formulas.

However, before the above investigation can be undertaken a deeper understanding of contexts as autonomous semantic objects is needed. In section 3.1 we describe contexts semantically as action transducers. This description enables us to derive the operational behaviour of a combined process, C[p], from the behaviours of the context C and the inner process p. As an example it is shown how a class of CCS-contexts is represented in this framework.

In section 3.3 we consider contexts as transformers of modal properties. It is shown, that for any context C there exist a function $I_C$ which maps "outer" properties to "inner" sufficient and necessary properties, i.e. for any property F and process p $C[p] \models F$ iff $p \models I_C(F)$. Extending $I_C$ to sets of modal properties turns out to give the desired weakest inner information transformer associated with C.

In section 3.4 we investigate contexts as environment transformers. In this case slightly weaker results are obtained: given a context C and an environment e we search for environments f such that for all processes p and q:

$$(2) \qquad p \sim_f q \quad \Rightarrow \quad \langle C,p\rangle \equiv_e \langle C,q\rangle$$

where $\langle C,p\rangle \equiv_e \langle C,q\rangle$ informally means that $C[p] \sim_e C[q]$ with C interacting identically with p and q. The existence of weakest (wrt. the discrimination ordering) environments satisfying (2) depends heavily on the structure of the environment system. For environment system closed under a non-swallowing[+] context system there always exists a weakest environment satisfying (2). For environment systems not closed, we give conditions sufficient for ensuring the existence. Finally, a denotational semantics of CCS-contexts in terms of how they transform language environments is given.

---

+ The notion of non-swallowing context systems will be defined later. Informally it means that a context cannot consume an (inner) action without producing an (outer) action.

## 3.1 OPERATIONAL SEMANTICS OF CONTEXTS

### 3.1.1 Context Systems.

We shall in this section study contexts as abstract semantic objects/agents on the same footing as processes and environments. This will make the problem of how contexts translate environments/subsets of modal formulas much easier to deal with as we shall see in the following sections.

If C is a context and p is a process, then we want C[p] to be a process whose behaviour can be derived from the behaviours of p and C. But what is the behaviour of a context ? Informally, in the behaviour of the process C[p] the context C acts as an interface between an external environment experimenting on the combined process C[p] and the internal process p in the sense that C consumes actions produced by the internal process p in order to produce actions for the external environment. Thus, we shall semantically describe contexts as action transducers (similar to the concepts of transducers from Automata Theory -- see for example /AU72/ vol 1).

If $p \xrightarrow{a} p'$, and C by consuming the a-action can produce a b-action, we will expect C[p] to be able to produce a b-action. Similar to the assumptions made about processes and environments it seems reasonable to assume that a context may change as a result of consuming and producing actions. This is reflected in the way we expect the process C[p] to change: if C can change to C′ after having consumed the action a and produced the action b, we will expect $C[p] \xrightarrow{b} C'[p']$.

In order to obtain a sufficiently general notion of contexts, which will enable us to express the operational behaviour of all the standard CCS-contexts, we shall allow a context to produce actions on its own without the need for consuming any actions produced by an internal process. Also, for reasons of symmetry, we shall allow a context to consume inner actions without producing any actions for the environment. Thus, processes and environments can be viewed as two extreme types of contexts: processes correspond to contexts which totally ignore the internal process and environments correspond to contexts which never produces any actions. If C can produce the action b and change to C′ in doing so without consuming any inner actions, we will expect $C[p] \xrightarrow{b} C'[p]$; i.e. the internal process p is unaffected. On the other hand, if $p \xrightarrow{a} p'$ and C can consume the action a changing to C′ without producing any outer actions, the process C[p] can change to the process C′[p′] without producing anything. Thus, if $C'[p'] \xrightarrow{b} q$ then also $C[p] \xrightarrow{b} q$. We shall assume that a context can always produce nothing by consuming nothing.

Formally, the <u>operational semantics of contexts</u> is described by a labelled transition system of the form $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$, where <u>Con</u> is the set of contexts, <u>Act</u> is the set of actions, $\underline{Act_0} = A \cup \{0\}$ where $\underline{0}$ is a distinguished <u>no-action</u> symbol $(0 \notin A)$, and $\longmapsto$ is the <u>transduction relation</u> satisfying $(C,(0,0),C) \in \longmapsto$ for all contexts C.

For $(C,(a,b),C') \in \longmapsto$ we will usually write $C \overset{b}{\underset{a}{\longmapsto}} C'$ which for $a,b \in Act$ is to be read: "the context C can by consuming an inner action a produce the outer action b and become the context C′ in doing so".

For $b \in Act$, $C \overset{b}{\underset{0}{\longmapsto}} C'$ is to be interpreted: "C may produce the outer action b without consuming any inner action and become the context C′ in doing so".

Similarly, for $a \in Act$, $C \overset{0}{\underset{a}{\longmapsto}} C'$ is to be read: "C may consume the inner action a without producing any outer action

action and become the context $C'$ in doing so".

### 3.1.2 Contexts and Processes.

We now know what the operational behaviour of contexts is. It remains therefore only to formalize how the behaviour of a combined process, $C[p]$, can be derived from the behaviours of $C$ and $p$. First let us extend the transduction relation to a relation over $Con \times Act_0^* \times Act_0^* \times Con$ in the natural way: For $u, v \in Act_0^*$ and $C, C' \in Con$ define $C \overset{v}{\underset{u}{\longmapsto}} C'$ iff $|u| = |v|$ and $u = a_1 \ldots a_n$, $v = b_1 \ldots b_n$ and for some contexts $C_1, \ldots, C_{n-1}$: $C \overset{b1}{\underset{a1}{\longmapsto}} C_1 \overset{b2}{\underset{a2}{\longmapsto}} C_2 \overset{b3}{\underset{a3}{\longmapsto}} \cdots C_{n-1} \overset{bn}{\underset{an}{\longmapsto}} C'$.
Then define the relation $\longmapsto\!\!\!\!\!\!\twoheadrightarrow \subseteq Con \times Act^* \times Act^* \times Con$ as:

$$C \overset{y}{\underset{x}{\longmapsto\!\!\!\!\twoheadrightarrow}} C' \quad \Leftrightarrow$$

$$\exists u, v \in Act_0^*.\ \bar{u} = x \wedge \bar{v} = y \wedge C \overset{v}{\underset{u}{\longmapsto}} C'$$

where $\bar{\_} : Act_0^* \rightarrow Act^*$ is defined inductively as: $\bar{\varepsilon} = \varepsilon$ and $\overline{au} = \bar{u}$ if $a = 0$ and $\overline{au} = a\,\bar{u}$ otherwise. (Thus $\bar{\_}$ simply cancels all occurrences of 0 in a string).

We can now introduce the concept of a process system being closed under a context system in order to formally express how the behaviour of $C[p]$ is derived from the behaviours of $C$ and $p$:

<u>Definition 3.1-1:</u> A <u>process system</u> $\mathbb{P} = (Pr, Act, \rightarrow)$ is <u>closed</u> under a context system $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$ with respect to the map $\_[\_] : Con \times Pr \rightarrow Pr$ if whenever $p, q \in Pr$, $b \in Act$ and $C \in Con$ the following holds:

$$\text{(i)}\quad C[p] \overset{b}{\rightarrow} q \quad \Leftrightarrow$$

$$\exists u \in Act^*.\ \exists p' \in Pr.\ \exists C' \in Con.$$

$$C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\twoheadrightarrow}} C' \quad \&$$

$$p \overset{u}{\rightarrow} p' \quad \&$$

$$q = C'[p']$$

where $\longrightarrow$ has been extended to strings over Act as defined in notation 2.1-2. $\square$

We shall later show that any process system can be extended to a closed system under a given context system.

Lemma 3.1-2: For all contexts $C, C'' \varepsilon Con$, $u_1, u_2, v \varepsilon Act^*$:

(i) $C \overset{u1\,u2}{\underset{v}{\longmapsto\!\!\!\!\!\rightarrow}} C'' \quad \Leftrightarrow$

$\exists v_1, v_2 \varepsilon Act^*. \exists C' \varepsilon Con.\ v = v_1 v_2 \wedge C \overset{u1}{\underset{v1}{\longmapsto\!\!\!\!\!\rightarrow}} C' \overset{u2}{\underset{v2}{\longmapsto\!\!\!\!\!\rightarrow}} C''$

(ii) $C \overset{v}{\underset{u1\,u2}{\longmapsto\!\!\!\!\!\rightarrow}} C'' \quad \Leftrightarrow$

$\exists v_1, v_2 \varepsilon Act^*. \exists C' \varepsilon Con.\ v = v_1 v_2 \wedge C \overset{v1}{\underset{u1}{\longmapsto\!\!\!\!\!\rightarrow}} C' \overset{v2}{\underset{u2}{\longmapsto\!\!\!\!\!\rightarrow}} C''$

Proof: Direct from the definition of $\longmapsto\!\!\!\!\!\rightarrow$. $\square$

We can now extend condition (i) of definition 3.1-1 to strings:

Lemma 3.1-3: Let $\mathbb{P}$ be a process system closed under the context system $\mathbb{C}$. Then for all $p, q \varepsilon Pr$, $v \varepsilon Act^+$ and $C \varepsilon Con$:

$$C[p] \xrightarrow{v} q \quad \Leftrightarrow$$

$$\exists u \varepsilon Act^*. \exists p' \varepsilon Pr. \exists C' \varepsilon Con.$$
$$C \overset{v}{\underset{u}{\longmapsto\!\!\!\!\!\rightarrow}} C' \quad \&$$
$$p \xrightarrow{u} p' \quad \&$$
$$q = C'[p']$$

Proof: Induction on $|v|$ with (i) for the base case ($|v|=1$) and use lemma 3.1-2 in the induction step. $\square$

Note that the above lemma does not hold for $v=\varepsilon$ (especially not the "$\Leftarrow$"-direction). The next lemma says that if a process system is closed wrt. two different maps $\_[\_]$, $\_\langle\_\rangle : Con \times Pr \longrightarrow Pr$ then there is a very strong connection between the two maps:

Lemma 3.1-4: Let $\mathbb{P}$ be closed under $\mathbb{C}$ wrt. $\_[\_]:Con \times Pr \to Pr$ and $\_\langle\_\rangle:Con \times Pr \to Pr$ then for all $p \varepsilon Pr$ and $C \varepsilon Con$: $C[p] \sim C\langle p\rangle$.

Proof: Show that $R = \{(C[p], C\langle p\rangle) \mid p\varepsilon Pr,\ C\varepsilon Con\}$ is a bisimulation using clause (i) of definition 3.1-1. □

We can now verify that our expectations for the behaviour of $C[p]$ in terms of the behaviours of C and p indeed has been fulfilled by the above defintion:

Proposition 3.1-5: Let $\mathbb{P}$ be a process system closed under the context system $\mathbb{C}$. Then for all $p,p',q\varepsilon Pr$, $a,b\varepsilon Act$ and $C,C'\varepsilon Con$ the following holds:

$$\text{(i)} \quad p \xrightarrow{a} p' \ \&\ C \overset{b}{\underset{a}{\longmapsto}} C' \ \Rightarrow\ C[p] \xrightarrow{b} C'[p']$$

$$\text{(ii)} \quad C \overset{b}{\underset{0}{\longmapsto}} C' \ \Rightarrow\ C[p] \xrightarrow{b} C'[p]$$

$$\text{(iii)} \quad p \xrightarrow{a} p' \ \&\ C \overset{0}{\underset{a}{\longmapsto}} C' \ \&\ C'[p'] \xrightarrow{b} q \ \Rightarrow\ C[p] \xrightarrow{b} q$$

Proof: Direct from definition 3.1-1 (i) and the definition of $\longmapsto\!\!\!\!\!\!\gg$. □

The next definition and proposition shows that any process system can be extended to a closed system under a given context system:

Definition 3.1-6: Let $\mathbb{P} = (Pr, Act, \to)$ be a process system and let $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$ be a context system. Then we define $\mathbb{P}_{\mathbb{C}}$ to be the process system $(Pr_{Con}, Act, \to)$ where $Pr_{Con}$ is the smallest set satisfying:

(a) $Pr \subseteq Pr_{Con}$

(b) $p\varepsilon Pr_{Con}\ \&\ C\varepsilon Con \Rightarrow (C,p)\varepsilon Pr_{Con}$

and $\to$ is the smallest relation on $Pr_{Con} \times Act \times Pr_{Con}$ satisfying for $p,p'\varepsilon Pr$, $q,q'\varepsilon Pr_{Con}$ and $C,C'\varepsilon Con$:

(i) $$\frac{p \xrightarrow{a} p'}{p \xrightarrow{a} p'}$$

(ii) $$\frac{q \xrightarrow{u} q' \qquad C \overset{b}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C'}{(C,q) \xrightarrow{b} (C',q')} \quad ; \quad b\varepsilon Act,\ u\varepsilon Act^*$$

□

Proposition 3.1-7: $\mathbb{P}_{\mathbb{C}}$ is closed under $\mathbb{C}$ with $\_[\_]:Con \times Pr_{Con} \to Pr_{Con}$ defined as:

$$C[p] \quad = \quad (C,p)$$

Proof: That condition (i) of definition 3.1-1 is satisfied follows directly from the definition of $\_[\_]$ and rule (ii) of definition 3.1-6.

□

We can now prove the longstanding claim that any "natural" process construction preserves bisimulation equivalence, $\sim$, provided "natural" is interpreted as: "can operationally be described by a context system". We shall in the next section show that all the standard CCS-constructions are indeed "natural" in this sense and as such preserve $\sim$. However, as we shall demonstrate later, there are ("unnatural") constructions whose operational behaviour cannot be described by any context system.

Theorem 3.1-8: Let $\mathbb{P}$ be a process system closed under a context system $\mathbb{C}$. Then, whenever $p \sim q$ and $C$ is a context, also $C[p] \sim C[q]$.

Proof: We prove that the relation:

$$R = \{(C[p],C[q]) \mid p \sim q\}$$

is a bisimulation. So let $C[p] \xrightarrow{b} r$. By definition 3.1-1 (i) then $C \overset{b}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C'$ and $p \xrightarrow{u} p'$ with $r = C'[p']$ for some $C', r'$ and $u$. Since $p \sim q$, $q \xrightarrow{u} q'$ for some $q'$ with $p' \sim q'$. Again by 3.1-1 (i), $C[q] \xrightarrow{b} C'[q']$ which is the matching move.

□

### 3.1.3 Contexts and Environments.

So far we have described how to derive the operational behaviour of a combined process, $C[p]$, from the behaviour of the inner process, $p$, and the behaviour of the context $C$. However, contexts are semantically viewed as interfaces between external environments and internal processes. Thus, an execution of a combined process, $C[p]$, in an environment, $e$, may - from the internal process' point of view - alternatively be viewed as an execution of $p$ in a <u>combined environment</u>, $e[C]$.

But what is the behaviour of this combined environment, $e[C]$, in terms of the behaviour of the outer environment, $e$, and the behaviour of the context $C$ ? Our answer to this is completely dual to the answer given for the behaviour of a combined process. Thus, we define the (dual) notion of an environment system being closed under a context system.

<u>Definition 3.1-9:</u> An environment system $\mathbb{E} = (Env, Act, \Rightarrow)$ is closed under a context system $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$ with respect to the map $\_[\_]: Env \times Con \longrightarrow Env$ if whenever $e, f \varepsilon Env$, $b \varepsilon Act$ and $C \varepsilon Con$ the following holds:

$$
\begin{aligned}
\text{(i)} \quad & e[C] \overset{b}{\Rightarrow} f \quad \Leftrightarrow \\
& \exists u \varepsilon Act^* . \exists e' \varepsilon Env . \exists C' \varepsilon Con . \\
& \qquad e \overset{u}{\Rightarrow} e' \quad \& \\
& \qquad C \underset{b}{\overset{u}{\longmapsto}} C' \quad \& \\
& \qquad f = e'[C']
\end{aligned}
$$

where $\Rightarrow$ has been extended to strings over Act as defined in notation 2.1-2.

□

As a dual to lemma 3.1-3 we can extend the condition (i) in the above definition to strings:

Lemma 3.1-10: Let $\mathbb{E}$ be an environment system closed under the context system $\mathbb{C}$. Then for all $e, f \varepsilon Env$, $v \varepsilon Act^+$ and $C \varepsilon Con$:

$$
\begin{array}{l}
e[C] \overset{v}{\Longrightarrow} f \quad \Leftrightarrow \\
\exists u \varepsilon Act^* . \exists e' \varepsilon Env . \exists C' \varepsilon Con . \\
\qquad e \overset{u}{\Longrightarrow} e' \quad \& \\
\qquad C \underset{v}{\overset{u}{\longmapsto}} C' \quad \& \\
\qquad f = e'[C']
\end{array}
$$
□

As a dual to proposition 3.1-5 we have:

Proposition 3.1-11: Let $\mathbb{E}$ be an environment system closed under the context system $\mathbb{C}$. Then for all $e, e', f \varepsilon Env$, $a, b \varepsilon Act$ and $C, C' \varepsilon Con$ the following holds:

(i) $e \overset{b}{\Longrightarrow} e' \;\&\; C \underset{a}{\overset{b}{\longmapsto}} C' \;\Rightarrow\; e[C] \overset{a}{\Longrightarrow} e'[C']$

(ii) $C \underset{a}{\overset{0}{\longmapsto}} C' \;\Rightarrow\; e[C] \overset{a}{\Longrightarrow} e[C']$

(iii) $e \overset{b}{\Longrightarrow} e' \;\&\; C \underset{0}{\overset{b}{\longmapsto}} C' \;\&\; e'[C'] \overset{a}{\Longrightarrow} f \;\Rightarrow\; e[C] \overset{a}{\Longrightarrow} f$ □

Again as a dual we can extend any environment system to a closed system under a given context system:

Definition 3.1-12: Let $\mathbb{E} = (Env, Act, \Longrightarrow)$ be an environment system and let $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$ be a context system. Then we define $\mathbb{E}_{\mathbb{C}}$ to be the environment system $(Env_{Con}, Act, \Longrightarrow)$ where $Env_{Con}$ is the smallest set satisfying:

(a) $Env \subseteq Env_{Con}$

(b) $e \varepsilon Env_{Con} \;\&\; C \varepsilon Con \;\Rightarrow\; (e, C) \varepsilon Env_{Con}$

and $\Longrightarrow$ is the smallest relation on $Env_{Con} \times Act \times Env_{Con}$ satisfying for $e, e' \varepsilon Env$, $f, f' \varepsilon Env_{Con}$ and $C, C' \varepsilon Con$:

$$
\text{(i)} \quad \frac{e \overset{a}{\Longrightarrow} e' \quad (\text{in } Env)}{e \overset{a}{\Longrightarrow} e' \quad (\text{in } Env_{Con})}
$$

$$\text{(ii)} \quad \frac{f \overset{u}{\Rightarrow} f' \qquad C \overset{u}{\underset{b}{\mapsto\!\!\!\Rightarrow}} C'}{(f,C) \overset{b}{\Rightarrow} (f',C')}$$ □

<u>Proposition 3.1-13:</u> $\mathbb{E}_{\mathbb{C}}$ is closed under $\mathbb{C}$ with $\_[\_]:\mathrm{Env}_{\mathrm{Con}} \times \mathrm{Con} \longrightarrow \mathrm{Env}_{\mathrm{Con}}$ given as:

$$e[C] = (e,C)$$ □

## 3.1.4 Composing Contexts.

If $C[p]$ is to be a process whenever C is a context and p is a process, then given a second context D, $D[C[p]]$ must also be a process. In some sense, the two layers of contexts surrounding p act as one single combined context. In order to express this formally we may assume that there is a binary <u>composition</u>, $\circ$, on contexts such that:

$$D[C[p]] = D \circ C[p]$$

Since then:

$$\begin{aligned} E \circ (D \circ C)[p] &= E[D[C[p]]] \\ &= (E \circ D) \circ C[p] \end{aligned}$$

it seems natural to assume that $\circ$ is associative.

The question is now: what is the behaviour of $D \circ C$ in terms of the behaviours of D and C ? The most straightforward way of combining behaviours of contexts seems to be the following:

<u>Definition 3.1-14:</u> Let $\mathbb{C} = (\mathrm{Con}, \mathrm{Act}_o \times \mathrm{Act}_o, \mapsto\!\!\!\Rightarrow)$ be a context system. Then $\circ : \mathrm{Con} \times \mathrm{Con} \longrightarrow \mathrm{Con}$ is a <u>context composition</u> iff $\circ$ is associative and for all $C,D,E \in \mathrm{Con}$ and $a,c \in \mathrm{Act}_o$ the following holds:

(i) $\quad C \circ D \xmapsto[a]{c} E \quad \Leftrightarrow$

$$\exists b \varepsilon Act_o . \exists D', C' \varepsilon Con.$$
$$C \xmapsto[b]{c} C' \quad \&$$
$$D \xmapsto[a]{b} D' \quad \&$$
$$E = C' \circ D' \qquad \square$$

In order to insure $C[D[p]] = C \circ D[p]$ we define the following notion of closure:

Definition 3.1-15: A process system $\mathbb{P}$ is said to be closed under a context system $\mathbb{C}$ with composition $\circ$ iff $\mathbb{P}$ is closed under $\mathbb{C}$ and for all $p \varepsilon Pr$ and $C, D \varepsilon Con$, $C[D[p]] = C \circ D[p]$. $\square$

We can extend the condition (i) in definition 3.1-14 to strings over Act:

Lemma 3.1-16: Let $\mathbb{C}$ be a context system with composition $\circ$ . Then for all $x, z \varepsilon Act^*$, $C, D, E \varepsilon Con$:

$$C \circ D \xmapsto[x]{z}\!\!\!\!\!\rightarrow E \quad \Leftrightarrow$$
$$\exists y \varepsilon Act^* . \exists C', D' \varepsilon Con.$$
$$C \xmapsto[y]{z}\!\!\!\!\!\rightarrow C' \quad \&$$
$$D \xmapsto[x]{y}\!\!\!\!\!\rightarrow D' \quad \&$$
$$E = C' \circ D'$$

Proof: "$\Rightarrow$": Easy by the definition of $\mapsto\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow$ and condition (i) of definition 3.1-14.

"$\Leftarrow$": Let $D \xmapsto[x]{y}\!\!\!\!\!\rightarrow D'$ and $C \xmapsto[y]{z}\!\!\!\!\!\rightarrow C'$. By definition of $\mapsto\!\!\!\!\!\rightarrow\!\!\!\!\!\rightarrow$ then for some $u, v', v, w \varepsilon Act_o^*$:

$$D \xmapsto[u]{v'} D' \quad \&$$
$$C \xmapsto[v]{w} C' \quad \& \quad \bar{u} = x, \ \bar{v} = \bar{v}' = y, \ \bar{w} = z$$

Unfortunately, we cannot compose D's and C's move directly since there is no guarantee that $v = v'$. However, if we can find $u', v'', w' \varepsilon Act_0^*$ such that:

$$D \overset{v''}{\underset{u'}{\longmapsto}} D' \quad \&$$

$$C \overset{w'}{\underset{v'}{\longmapsto}} C' \quad \& \quad \bar{u}'=x, \ \bar{v}''=y, \ \bar{w}'=z$$

then by applying (i) repeatedly we get:

$$C \circ D \overset{w'}{\underset{u'}{\longmapsto}} C' \circ D'$$

and hence by definition of $\longmapsto\!\!\!\!\!\rightarrow$:

$$C \circ D \overset{z}{\underset{x}{\longmapsto\!\!\!\!\rightarrow}} C' \circ D'$$

By definition of context systems we can always add O-moves into a transduction, i.e. if $D \overset{v1\ v2}{\underset{u1\ u2}{\longmapsto}} D'$ with $|vi| = |ui|$ i=1,2, then also $D \overset{v1\ O\ v2}{\underset{u1\ O\ u2}{\longmapsto}} D'$.
Thus, if y=b1...bn, then by adding O's we can for any $k > |v|$ obtain:

$$D \overset{O^k\, b1\, O^k\, b2 \ldots bn\, O^k}{\underset{u'}{\longmapsto}} D'$$

for some u' (dependent of k) and similar for any $l > |v'|$

$$C \overset{w'}{\underset{O^l\, b1\, O^l\, b2 \ldots bn\, O^l}{\longmapsto}} C'$$

for some w'. Thus by taking $l = k > \max\{|v|, |v'|\}$ we obtain the desired common v'' as $O^k\, b1\, O^k\, b2 \ldots bn\, O^k$. □

Now let us assume $\mathbb{P}$ is a process system closed under a context system $\mathbb{C}$ with composition $\circ$. Then for all $p \varepsilon Pr$ and $C, D \varepsilon Con$ $C[D[p]] = C \circ D[p]$. By definition 3.1-1 and lemma 3.1-3 we have:

$$C[D[p]] \xrightarrow{b} q$$

<u>iff</u> $\quad [\exists C'.\ C \overset{b}{\underset{\varepsilon}{\longmapsto}} C' \ \&\ q = C'[D[p]]\ ]$

or

$[\exists u \varepsilon Act^+ . \exists v \varepsilon Act^* . \exists C', D' \varepsilon Con . \exists p' \varepsilon Pr.$

$$C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \quad \&$$

$$D \overset{u}{\underset{v}{\longmapsto\!\!\!\!\rightarrow}} D' \quad \&$$

$$p \xrightarrow{v} p' \quad \&$$

$$q = C'[D'[p']] \quad ]$$

and by definition 3.1-1 and lemma 3.1-16:

$$C \circ D[p] \xrightarrow{b} q$$

<u>iff</u> $\exists u,v \varepsilon Act^*. \exists C', D' \varepsilon Con. \exists p' \varepsilon Pr.$

$$C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \quad \&$$

$$D \overset{u}{\underset{v}{\longmapsto\!\!\!\!\rightarrow}} D' \quad \&$$

$$p \xrightarrow{v} p' \quad \&$$

$$q = C' \circ D'[p']$$

From this it follows that in general it is not possible for $C[D[p]]$ and $C \circ D[p]$ to have the same behaviour: If $C \overset{b}{\underset{\varepsilon}{\longmapsto\!\!\!\!\rightarrow}} C'$ then in $C[D[p]]$ D and p are left unaffected whereas D and p may change in $C \circ D[p]$ in case D has a move of the form $D \overset{\varepsilon}{\underset{v}{\longmapsto\!\!\!\!\rightarrow}} D'$. Thus, it seems that if there is to exist any closed process systems wrt. a context system $\mathbb{C}$ with composition, $\circ$, the contexts of $\mathbb{C}$ must have the property that they never produce a no-action, 0, from a real action, i.e. for all $a \varepsilon Act_0$, all $C, C' \varepsilon Con$:

$$C \overset{0}{\underset{a}{\longmapsto}} C' \quad \Rightarrow \quad a=0 \quad \& \quad C=C'$$

(Note, that the reverse implication is always satisfied by the definition of a context system). Fortunately, we shall later see that all CCS-contexts have this property. We call a context with this property <u>non-swallowing</u>.

Now as a dual to definition 3.1-15 we could define the notion of an environment system being closed under a context system with a composition. However, this would impose the following dual restriction on contexts: for all $a \varepsilon Act_0$ and all $C, C' \varepsilon Con$:

$$C \overset{a}{\underset{0}{\longmapsto}} C' \quad \Rightarrow \quad a=0 \quad \& \quad C=C'$$

i.e. if a context is producing an (real) action it must have consumed some (real) action. Since this restriction is not fulfilled by all CCS-contexts we shall not introduce this dual notion. However, we can manage without it: if $\mathbb{E}$ is an environment system closed under a context system $\mathbb{C}$ in the sense of definition 3.1-9 and $\mathbb{C}$ moreover is equipped with a composition, $\circ$, there is a sufficiently strong relationship between combined environments of the forms $(e[C])[D]$ and $e[C\circ D]$.

Lemma 3.1-17: Let $\mathbb{E}$ be an environment system closed under a context system $\mathbb{C}$. Then whenever $f,e,e'\varepsilon Env$, $u\varepsilon Act^*$ and $C,C'\varepsilon Con$ the following holds:

$$\text{(i)}\quad e\leqslant f \;\Rightarrow\; e[C]\leqslant f[C]$$
$$\text{(ii)}\quad \left[e\overset{u}{\Longrightarrow}e' \;\&\; C\overset{u}{\underset{\varepsilon}{\longmapsto\!\!\!\!\twoheadrightarrow}}C'\right] \Rightarrow \quad e'[C']\leqslant e[C]$$

Proof: (i) Show that $S=\{(e[C],f[C])\mid e\leqslant f\}$ is a simulation using definition 3.1-9.
(ii) Assume $e'[C']\overset{b}{\Longrightarrow}f$. Then by definition 3.1-9, $e'\overset{v}{\Longrightarrow}e''$, $C'\overset{v}{\underset{b}{\longmapsto\!\!\!\!\twoheadrightarrow}}C''$ with $f=e''[C'']$ for some $e''$, $C''$ and v. Obviously $e\overset{uv}{\Longrightarrow}e'$ and by lemma 3.1-2, $C\overset{uv}{\underset{b}{\longmapsto\!\!\!\!\twoheadrightarrow}}C''$. Thus by definition 3.1-9, $e[C]\overset{b}{\Longrightarrow}f$ as well. □

Lemma 3.1-18: Let $\mathbb{E}$ be an environment system closed under the context system $\mathbb{C}$ and let $\circ$ be a composition for $\mathbb{C}$. Then, whenever $e\varepsilon Env$, $C,D\varepsilon Con$ the following holds:

$$(e[C])[D]\simeq e[C\circ D]$$

Proof: "$\leqslant$": We prove that:

$$S=\{((e[C])[D],e[C\circ D])\mid e\varepsilon Env,\ C,D\varepsilon Con\}$$

is a simulation. Assume $(e[C])[D]\overset{b}{\Longrightarrow}f$. Then either:

$$\text{(a)}\quad D\overset{\varepsilon}{\underset{b}{\longmapsto\!\!\!\!\twoheadrightarrow}}D' \;\&\; f=(e[C])[D']$$

for some $D'$ or:

$$\text{(b)}\quad e\overset{v}{\Longrightarrow}e' \;\&\; C\overset{v}{\underset{u}{\longmapsto\!\!\!\!\twoheadrightarrow}}C' \;\&$$
$$D\overset{u}{\underset{b}{\longmapsto\!\!\!\!\twoheadrightarrow}}D' \;\&\; f=(e'[C'])[D']$$

for some $e',C',D'$ and $v\varepsilon Act^*$, $u\varepsilon Act^+$.

In (a), $C\circ D\overset{\varepsilon}{\underset{b}{\longmapsto\!\!\!\!\twoheadrightarrow}}C\circ D'$ since $C\overset{0}{\underset{0}{\longmapsto}}C$. Thus, since $e\overset{\varepsilon}{\Longrightarrow}e$, $e[C\circ D]\overset{b}{\Longrightarrow}e[C\circ D']$ which is the matching move.

In (b), $C \circ D \overset{v}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}} C' \circ D'$. Thus, since $e \overset{v}{\Longrightarrow} e'$, $e[C \circ D] \overset{b}{\Longrightarrow} e'[C' \circ D']$ which is the matching move.

"$\geqslant$": We prove that:

$$S = \{ (e[C \circ D], f[D]) \mid e[C] \leqslant f \}$$

is a simulation. So assume $e[C \circ D] \overset{b}{\Longrightarrow} f$. Then:

$$e \overset{v}{\Longrightarrow} e' \ \ \& \ \ C \overset{v}{\underset{u}{\mapsto\!\!\!\twoheadrightarrow}} C' \ \ \& \ \ D \overset{u}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}} D' \ \ \& \ \ f = e'[C' \circ D']$$

for some $e', C', D'$ and $v, u \in Act^*$.

If $u = \varepsilon$ then by lemma 3.1-17 (ii), $e'[C'] \leqslant e[C] \leqslant f$. Since $D \overset{\varepsilon}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}} D'$, $f[D] \overset{b}{\Longrightarrow} f[D']$ which is a matching move.

If $u \neq \varepsilon$ then by lemma 3.1-10, $e[C] \overset{u}{\Longrightarrow} e'[C']$. Since $e[C] \leqslant f$, $f \overset{u}{\Longrightarrow} f'$ for some $f'$ with $e'[C'] \leqslant f'$. Since $D \overset{u}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}} D'$, $f[D] \overset{b}{\Longrightarrow} f'[D']$ which is the matching move. $\square$

## 3.2 C C S

In this section the syntax and operational semantics of CCS-processes and -contexts will be introduced formally. For more motivation and a full treatment of CCS (-processes) the reader is refered to /Mil80/, in particular chapters 5 and 7. As the main results of the section it is shown that CCS-contexts are equipped with a composition and that CCS-processes are closed under CCS-contexts with this composition.

The system of CCS-processes+ is closed under action-prefixing together with binary summation and join. Beyond this, CCS-processes are build up from a number of operators one of which is the parallel operator, $\underline{|}$. The $|$ operator represents the parallel composition of two processes, enabling communication to occur between them, and at the same time allowing their behaviours to interleave freely. Together with the $|$ operator a structure on the action set Act is introduced: it is assumed that Act is a disjoint union of three sets $\Delta$, $\bar{\Delta}$ and a singleton $\{1\}$ . The two sets, $\Delta$ and $\bar{\Delta}$, are isomorphic and for $a \varepsilon \Delta$ $(a \varepsilon \bar{\Delta})$, $\bar{a} \varepsilon \bar{\Delta}$ $(\bar{a} \varepsilon \Delta)$ is the <u>complementary</u> action <u>where</u> $\bar{}$ denotes both isomorphisms. Hence, whenever $a \varepsilon \Delta \cup \bar{\Delta}$ , $\bar{\bar{a}} = a$. Communication of two processes in parallel may then take place if they can perform complementary actions. As a result of the communication the combined system will produce a 1-action (a so-called "silent" or "internal" action).

Another class of operators is the <u>restriction operators</u>, $\restriction S$ for $S \subseteq Act$, which restrict a process' actions to a set S. Normally it is assumed that $1 \varepsilon S$ and that S is closed under $\bar{}$. A restriction operator is useful for ensuring that certain communications of processes composed by the $|$ operator occur internally.

---

\+ The system of processes defined here is really a slight extension of the standard CCS processes as defined in /Mil80/ due to the presence of the &-operator.

The last class of operators is the renaming operators, $[\Phi]$, where $\Phi$ is a function Act $\longrightarrow$ Act. A renaming operator relabels an inner process' actions according to a function $\Phi$: Act $\longrightarrow$ Act. Normally it is assumed that $\Phi$ preserves 1 and $\bar{\ }$. For reasons which will be explained later we shall assume that $\Phi$ is co-image finite, i.e. for all $a \varepsilon$ Act the set $\{b \mid \Phi b = a\}$ is finite.

Using the above six operators processes with quite complex behaviours can be defined, but the behaviours will in all cases be finite. In order to obtain processes with infinite behaviours a form of recursion is introduced: when x is a variable and p is a process with x as a possible free variable, $\mu x.p$ is a process which behaves as a solution to the equation $x \sim p$.

We can now introduce the syntax of CCS process expression; $PE_{CCS}$:

$$\boxed{\begin{array}{l} p ::= \ \mathbb{O} \mid x \mid a.p \mid p+p' \mid p\,\&\,p' \mid \\ \qquad\quad p \mid p' \mid p\lceil S \mid p[\Phi] \mid \mu x.p \end{array}}$$

where $x \varepsilon$ Var (a set of variables), $a \varepsilon$ Act (the set of actions), $S \subseteq$ Act and $\Phi$ is a co-image finite function Act $\longrightarrow$ Act.

In $\mu x.p$ the prefix $\mu x$ binds every free occurrence of x in p. The concepts of free and bound variables are defined as usual. $p\{q/x\}$ stands for the substitution of the expression q for the variable x in the expression p. The definition of substitution is as usual with bound variables of p being renamed when capturing of free variables of q can occur (see /Mil82/).

In order to obtain an image-finite process system a syntactic restriction is imposed on $\mu x.p$, that x is guarded in p: every free occurrence of x in p is within some subexpression a.q of p.

Let $\underline{P}_{CCS}$ be all closed CCS process expressions. Then we define the process system $\mathbb{P}_{CCS}$ as the transition system $(P_{CCS}, Act, \longrightarrow \cap (P_{CCS} \times Act \times P_{CCS}))$, where $\longrightarrow$ is the smallest relation on $PE_{CCS} \times Act \times PE_{CCS}$ satisfying the following rules:

| | | |
|---|---|---|
| ACT | $a.p \xrightarrow{a} p$ | |
| SUM | $\dfrac{p_1 \xrightarrow{a} p_1'}{p_1 + p_2 \xrightarrow{a} p_1'}$ | $\dfrac{p_2 \xrightarrow{a} p_2'}{p_1 + p_2 \xrightarrow{a} p_2'}$ |
| JOIN | $\dfrac{p_1 \xrightarrow{a} p_1' \qquad p_2 \xrightarrow{a} p_2'}{p_1 \,\&\, p_2 \xrightarrow{a} p_1' \,\&\, p_2'}$ | |
| PAR | $\dfrac{p_1 \xrightarrow{a} p_1'}{p_1 \mid p_2 \xrightarrow{a} p_1' \mid p_2}$ | $\dfrac{p_2 \xrightarrow{a} p_2'}{p_1 \mid p_2 \xrightarrow{a} p_1 \mid p_2'}$ |
| | $\dfrac{p_1 \xrightarrow{a} p_1' \qquad p_2 \xrightarrow{\bar{a}} p_2'}{p_1 \mid p_2 \xrightarrow{1} p_1' \mid p_2'}$ | |
| REST | $\dfrac{p \xrightarrow{a} p'}{p \restriction S \xrightarrow{a} p' \restriction S}$ ; $a \in S$ | |
| REN | $\dfrac{p \xrightarrow{a} p'}{p[\Phi] \xrightarrow{\Phi a} p'[\Phi]}$ | |
| REC | $\dfrac{p\{\mu x.p / x\} \xrightarrow{a} q}{\mu x.p \xrightarrow{a} q}$ | |

A CCS-context is a process expression with free variables contained in the singleton set $\{[\,]\}$ (thus we assume there is a distinguished variable $[\,]$). Our goal is to make

CCS-processes closed under CCS-contexts with a combined process, $C[p]$, simply being the process obtained by substituting p for the place-holding variable, $[\,]$, in C; i.e. $C[p] = C\{p/[\,]\}$.

However, if this goal is to be achieved we cannot accept all process expressions with free variables contained in $\{[\,]\}$ as contexts. In particular we must avoid expressions of the form $[\,]\&[\,]$ and $[\,]|[\,]$ : the obvious semantics of the context $[\,]\&[\,]$ is $[\,]\&[\,] \overset{a}{\underset{a}{\mapsto}} [\,]\&[\,]$ for all $a \varepsilon Act_o$. Now consider a combined process of the form $([\,]\&[\,])[p]$ then by definition 3.1-1 and the above semantics of $[\,]\&[\,]$ the behaviour of $([\,]\&[\,])[p]$ must satisfy:

$$([\,]\&[\,])[p] \xrightarrow{b} q \quad \Leftrightarrow$$
$$\exists p'.\ p \xrightarrow{b} p' \quad \& \quad q = ([\,]\&[\,])[p']$$

However, if we insist that $C[p]$ is given by $C\{p/[\,]\}$ then the above becomes:

$$p\,\&\,p \xrightarrow{b} q \quad \Leftrightarrow$$
$$\exists p'.\ p \xrightarrow{b} p' \quad \& \quad q = p'\,\&\,p'$$

which is false in general, since the two instances of p in $p\,\&\,p$ might choose different b-derivatives.

Also, to avoid the above situations ($[\,]\&[\,]$, $[\,]|[\,]$) to occur during an execution, we shall not allow $[\,]$ to occur inside a recursion (this restriction can be loosened slightly so to allow certain expressions with $[\,]$ ocurring inside a recursion as contexts; e.g. $\mu x.(a.x + [\,])$).
The grammar specifying CCS-contexts, $\underline{C}_{CCS}$, is as follows:

$$\boxed{\begin{array}{lllllll} C ::= & p \mid [\,] \mid a.C \mid C + D \mid \\ & p\,\&\,C \mid C\,\&\,p \mid C \,|\, p \mid p \,|\, C \mid \\ & C\!\upharpoonright\! S \mid C[\Phi] \end{array}}$$

where $a \in Act$, $S \subseteq Act$, $\Phi$ is a co-image finite function $Act \longrightarrow Act$ and $p \in P_{CCS}$. We can now define the context system $\mathbb{C}_{CCS}$ as the transition system $(C_{CCS}, Act_0 \times Act_0, \longmapsto)$ where $\longmapsto$ is the smallest relation on $C_{CCS} \times Act_0 \times Act_0 \times C_{CCS}$ satisfying the following rules:

| | |
|---|---|
| NOACT | $C \underset{0}{\overset{0}{\longmapsto}} C$ |
| CONST | $\dfrac{p \xrightarrow{a} p'}{p \underset{0}{\overset{a}{\longmapsto}} p'}$ |
| ID | $[\,] \underset{a}{\overset{a}{\longmapsto}} [\,]$ |
| ACT | $a.C \underset{0}{\overset{a}{\longmapsto}} C$ |
| SUM | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C'}{C + D \underset{a}{\overset{b}{\longmapsto}} C'}$ ; $b \neq 0$ $\qquad \dfrac{D \underset{a}{\overset{b}{\longmapsto}} D'}{C + D \underset{a}{\overset{b}{\longmapsto}} D'}$ ; $b \neq 0$ |
| JOIN | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C' \quad p \xrightarrow{b} p'}{C \,\&\, p \underset{a}{\overset{b}{\longmapsto}} C' \,\&\, p'}$ |
| PAR | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C'}{C \mid p \underset{a}{\overset{b}{\longmapsto}} C' \mid p}$ $\qquad \dfrac{p \xrightarrow{b} p'}{C \mid p \underset{0}{\overset{b}{\longmapsto}} C \mid p'}$ |
| | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C' \quad p \xrightarrow{\bar{b}} p'}{C \mid p \underset{a}{\overset{1}{\longmapsto}} C' \mid p'}$ |
| REST | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C'}{C \restriction S \underset{a}{\overset{b}{\longmapsto}} C' \restriction S}$ ; $b \in S$ |
| REN | $\dfrac{C \underset{a}{\overset{b}{\longmapsto}} C'}{C[\Phi] \underset{a}{\overset{\Phi b}{\longmapsto}} C'[\Phi]}$ ; $b \neq 0$ |

The operational semantics of $p \,\&\, C$ and $p \mid C$ are given by rules symmetric to JOIN and PAR.

Now, let $\Phi_P$ be the endofunction on $\mathcal{P}(PE_{CCS} \times Act \times PE_{CCS})$ defined by the rules for $\longrightarrow$. I.e if $R \subseteq PE_{CCS} \times Act \times PE_{CCS}$ then $(p,a,p') \varepsilon \Phi_P(R)$ iff there is some rule with $p \xrightarrow{a} p'$ as conclusion and such that if $\longrightarrow$ is replaced by R the premisses of the rule holds. Then $\Phi_P$ is monotonic wrt. $\subseteq$ and $\longrightarrow$ is the smallest fixed-point of $\Phi_P$. As such if R is another relation over $PE_{CCS} \times Act \times PE_{CCS}$ closed under the rules, i.e. $\Phi_P(R) \subseteq R$, then $\longrightarrow \subseteq R$. This gives us a way of proving properties of $\longrightarrow$. (similar to the bisimulation proof technique).
It is easily seen that all the rules of $\longrightarrow$ are finitary. Consequently $\Phi_P$ is continuous (for more information about inductive definitions we refer the reader to /A83/). Thus, $\longrightarrow \; = \cup_{n \varepsilon \omega} \longrightarrow_n$ where $\longrightarrow_o = \emptyset$ and $\longrightarrow_{n+1} = \Phi_P(\longrightarrow_n)$. This allow us to prove properties of $\longrightarrow$ by "the number of rules applied".

Similarly, an endofunction, $\Phi_C$, on $\mathcal{P}(C_{CCS} \times Act_o \times Act_o \times C_{CCS})$ can be derived from the rules of $\longmapsto$, such that $\longmapsto$ is the least fixed-point of $\Phi_C$. All the rules of $\longmapsto$ are finitary. Hence, $\Phi_C$ is continuous and $\longmapsto \; = \cup_{n \varepsilon \omega} \longmapsto_n$ with $\longmapsto_o = \emptyset$ and $\longmapsto_{n+1} = \Phi_C(\longmapsto_n)$.

We can now prove some properties of $\longrightarrow$ and $\longmapsto$ :

<u>Proposition 3.2-1:</u> For all CCS process expressions, p, the set $\{(a,p') \mid p \xrightarrow{a} p'\}$ is finite.

<u>Proof:</u> By structure on p. The only non-trivial case is the recursion-case, i.e. when p is of the form $\mu x.r$. Since x is guarded in r it is easily shown - by structure on r - that $r\{\mu x.r/x\} \xrightarrow{a} q$ iff for some $r'$, $r \xrightarrow{a} r'$ and $q = r'\{\mu x.r/x\}$. Since r by the induction hypothesis is supposed to have finitely many derivatives so has $r\{\mu x.r/x\}$ and hence $\mu x.r$. $\square$

For the above proposition to hold it is crucial that the guardedness condition for recursive definitions is fulfilled, e.g. for the process $\mu x.(a.\mathbb{O} \mid x)$, the proposition fails to hold.

Corollary 3.2-2: The process system $\mathbb{P}_{CCS}$ is image-finite. $\square$

Proposition 3.2-3: For all CCS contexts $C, C'$ and $a \in Act_0$:

$$C \overset{0}{\underset{a}{\mapsto}} C' \quad \Rightarrow \quad a=0 \quad \& \quad C=C'$$

Proof: By structure of C. $\square$

Proposition 3.2-4: For all CCS-contexts C and actions $a \in Act_0$ the set $\{(b,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is finite.

Proof: By structure on C using the previous proposition 3.2-1. We prove three cases leaving the rest to the reader:

CONST: $C=p$: Then the set $\{(b,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to either $\emptyset$ (if $a \neq 0$) or $\{(b,p') \mid p \xrightarrow{b} p'\}$ which by proposition 3.2-1 is finite.

JOIN: $C=D\,\&\,p$: Then the set $\{(b,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to $\{(b,D'\,\&\,p') \mid D \overset{b}{\underset{a}{\mapsto}} D' \;\&\; p \xrightarrow{b} p'\}$ which is finite since $\{(b,p') \mid p \xrightarrow{b} p'\}$ is finite by proposition 3.2-1 and $\{(b,D') \mid D \overset{b}{\underset{a}{\mapsto}} D'\}$ is finite by induction hypothesis.

REN: $C=D[\Phi]$: Then the set $\{(b,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to $\{(\Phi b, D'[\Phi]) \mid D \overset{b}{\underset{a}{\mapsto}} D'\}$ which is easily seen to be finite by the induction hypothesis. $\square$

Proposition 3.2-5: For all CCS-contexts C and actions $b \in Act_0$ the set $\{(a,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is finite.

Proof: For b=0 the above set is just the singleton $\{(0,C)\}$ by proposition 3.2-3. For $b \in Act$ the proof is by induction on the structure of C. We prove three cases leaving the rest to the reader.

CONST: $C=p$: Then the set $\{(a,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to $\{(0,p') \mid p \xrightarrow{b} p'\}$ which as a consequence of proposition 3.2-1 is finite.

JOIN: $C=D \,\&\, p$: Then the set $\{(a,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to $\{(a,D' \,\&\, p') \mid D \overset{b}{\underset{a}{\mapsto}} D' \;\&\; p \xrightarrow{b} p'\}$ which is finite by induction hypothesis and proposition 3.2-1.

REN: $C=D[\Phi]$: Then the set $\{(a,C') \mid C \overset{b}{\underset{a}{\mapsto}} C'\}$ is equal to $\{(a,D'[\Phi]) \mid \exists c\varepsilon Act.\ D \overset{c}{\underset{a}{\mapsto}} D' \;\&\; b=\Phi c\}$ or:

$$\bigcup_{c\varepsilon Act.\,\Phi c=b} \{(a,D'[\Phi]) \mid D \overset{c}{\underset{a}{\mapsto}} D'\}$$

For each c the corresponding set is finite by the induction hypothesis. By the co-image finiteness of $\Phi$ there are only finitely many $c\varepsilon Act$ such that $\Phi c=b$. Thus the full set is finite.

□

Let $\#: Act^* \times Act^* \longrightarrow \mathcal{P}(Act^*)$ be the shuffling operator defined by:

$$\varepsilon \# y = \{y\}$$
$$x \# \varepsilon = \{x\}$$

$$ax \# by = \begin{cases} a(x \# by) \cup b(ax \# y) \cup 1(x \# y) \;; & \text{if } a=\bar{b} \\ a(x \# by) \cup b(ax \# y) \;; & \text{otherwise} \end{cases}$$

with action prefixing generalized to sets of strings.

Proposition 3.2-6: The following equivalences hold for CCS-contexts, when $v\varepsilon Act^+$:

(i) $p \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \Leftrightarrow u=\varepsilon \;\&\; \exists p'.\ p \xrightarrow{v} p' \;\&\; p'=C'$

(ii) $[\,] \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \Leftrightarrow v=u \;\&\; [\,]=C'$

(iii) $a.C \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \Leftrightarrow \exists w.\ C \overset{w}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \;\&\; v=aw$

(iv) $C+D \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} E \Leftrightarrow C \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} E$ or $D \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} E$

(v) $C \,\&\, p \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \Leftrightarrow \exists C'', p'.\ C \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C'' \;\&\; p \xrightarrow{v} p' \;\&\; C'=C'' \,\&\, p'$

(vi) $C \mid p \overset{v}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C' \Leftrightarrow \exists C'', p', x, y.\ C \overset{x}{\underset{u}{\mapsto\!\!\twoheadrightarrow}} C'' \;\&\; p \xrightarrow{y} p' \;\&$
$v\varepsilon x \# y \;\&\; C'=C'' \mid p'$

(vii) $C\lceil S \overset{v}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \Leftrightarrow v \varepsilon S^* \ \& \ \exists C''.\ C \overset{v}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C'' \ \& \ C' = C''\lceil S$

(viii) $C[\Phi] \overset{v}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \Leftrightarrow \exists C'', w.\ C \overset{w}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C'' \ \& \ v = \Phi w \ \& \ C' = C''[\Phi]$

Proof: From the definition of $\longmapsto\!\!\!\!\rightarrow$ and the rules for $\longmapsto$.

□

Proposition 3.2-7: For all CCS-contexts C and $b \varepsilon Act$ the set $\{(u, C') \mid C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C'\}$ is finite.

Proof: By proposition 3.2-3 and definition of $\longmapsto\!\!\!\!\rightarrow$, $|u| < 1$ and $C \overset{b}{\underset{u}{\longmapsto}} C'$. By proposition 3.2-4 we then conclude that the set is finite.

Note, that the opposite proposition does not hold. I.e. it is not in general true that the set $\{(u, C') \mid C \overset{u}{\underset{a}{\longmapsto\!\!\!\!\rightarrow}} C'\}$ is finite for a CCS-context C and action a. The reason is that the opposite proposition to 3.2-3 does not hold for CCS-contexts.

□

We can now prove that $\mathbb{C}_{CCS}$ is equipped with a composition, which is nothing more than substitution.

Proposition 3.2-8: Let $\circ : C_{CCS} \times C_{CCS} \rightarrow C_{CCS}$ be defined by:

$$C \circ D = C\{D/[\,]\}$$

Then $\circ$ is a composition for $\mathbb{C}_{CCS}$.

Proof: We must verify the conditions of definition 3.1-14. Obviously $\circ$ is associative by properties of substitution. It remains to show that for all $C, D, E \varepsilon C_{CCS}$, and $a, c \varepsilon Act_o$:

$$\begin{aligned} & C \circ D \overset{c}{\underset{a}{\longmapsto}} E \Leftrightarrow \\ & \exists b \varepsilon Act_o.\ \exists D', C' \varepsilon C_{CCS}. \\ & \qquad C \overset{c}{\underset{b}{\longmapsto}} C' \quad \& \\ & \qquad D \overset{b}{\underset{a}{\longmapsto}} D' \quad \& \\ & \qquad E = C' \circ D' \end{aligned}$$

This is easily proved by the structure of C using properties of substitution. The details are routine.

□

Theorem 3.2-9: $\mathbb{P}_{CCS}$ is closed under $\mathbb{C}_{CCS}$ with $\circ$, by defining the map $\_[\_]$ as $C[p] = C\{p/[\,]\}$.

Proof: We must verify the conditions of definition 3.1-15 and definition 3.1-1. Obviously, by properties of substitution, $C[D[p]] = C\circ D[p]$. That:

$$C[p] \xrightarrow{b} q \quad \Leftrightarrow$$

$$\exists u.\exists p'.\exists C'.$$

$$C \underset{u}{\overset{b}{\longmapsto}} C' \;\&\; p \xrightarrow{u} p' \;\&\; q = C'[p']$$

is shown by induction on C using properties of substitution and proposition 3.2-6. The details are routine. □

As a corollary to theorem 3.1-8 and the above theorem 3.2-9 we can conclude that all the CCS operations preserve $\sim$.

Corollary 3.2-10: Let $p, q, p_1, p_2, q_1$ and $q_2$ be CCS-processes such that $p \sim q$, $p_1 \sim q_1$, $p_2 \sim q_2$. Then:

(i) $a.p \sim a.q$

(ii) $p_1 + p_2 \sim q_1 + q_2$

(iii) $p_1 \,\&\, p_2 \sim q_1 \,\&\, q_2$

(iv) $p_1 \mid p_2 \sim q_1 \mid q_2$

(v) $p\lceil S \sim q\lceil S$

(vi) $p[\Phi] \sim q[\Phi]$

where $S \subseteq Act$ and $\Phi$ is a co-image finite function $Act \rightarrow Act$.

Proof: Let us just prove (iii). The remaining clauses are proved similarly. By definition of $\_[\_]$ and theorem 3.1-8:

$$p_1 \,\&\, p_2 = (p_1 \,\&\, [\,])[p_2] \sim (p_1 \,\&\, [\,])[q_2] = p_1 \,\&\, q_2$$

and:

$$p_1 \,\&\, q_2 = ([\,] \,\&\, q_2)[p_1] \sim ([\,] \,\&\, q_2)[q_1] = q_1 \,\&\, q_2$$

Hence, by transitivity of $\sim$, $p_1 \,\&\, p_2 \sim q_1 \,\&\, q_2$. □

## 3.3 CONTEXTS AS MODAL PROPERTY TRANSFORMERS

In this section we shall investigate how contexts transform modal properties. More specifically, the following two problems will be treated:

A: Assume we want to construct a process $r$ such that $r$ satisfies some given property $F \in M$ and such that $r$ is a combined process of the form $C[p]$ where $C$ is a given context. We shall constructively show that there exists a property $G \in M$ (depending on $C$ and $F$) such that a necessary and sufficient condition for $C[p]$ to satisfy $F$ is that $p$ satisfies $G$. The construction of $G$ from $C$ and $F$ can be used as the basis for complete, decompositional proof systems of correctness assertions, $p \models F$, similar to those recently presented in /St84,St85,W85,W85B/. Our construction is actually a generalization of the decomposition of assertions given in /W85B/.

B: Recall the parameterized version of $\sim$ where the parameters simply are subsets, $A$, of the property domain $M$, with $\sim_A$ defined by:

$$p \sim_A q \quad \Leftrightarrow \quad M(p) \cap A \ = \ M(q) \cap A$$

Given a context $C$ and a set $A \subseteq M$ we want to reduce the parameterized equivalence problem, $C[p] \sim_A C[q]$, to a parameterized equivalence problem involving the inner processes: i.e. we want to find a set $B \subseteq M$ such that for all $p$ and $q$:

$$(*) \qquad p \sim_B q \quad \Rightarrow \quad C[p] \sim_A C[q]$$

In order to make the proof of $p \sim_B q$ as easy as possible we prefer $B$ as small as possible wrt. the discrimination ordering, $\sqsubseteq$, between sets of modal properties. Using the construction from problem A it turns out that we can find a set $B \subseteq M$ such that for all processes $p$ and $q$:

$$(**) \qquad p \sim_B q \quad \Leftrightarrow \quad C[p] \sim_A C[q]$$

Obviously, this set B is the (desired) least discriminating set satisfying (*).

We shall for the remainder of this section assume that $\mathbb{P}$ is a process system closed under a context system $\mathbb{C}$. In order to make the construction in $\underline{A}$ possible the following finiteness restriction on contexts is imposed:

(F) Whenever C is a context and $b \varepsilon Act$, the set:

$$\left\{(u,C')\varepsilon Act^* \times Con \mid C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\twoheadrightarrow}} C' \right\}$$

is finite.

Note, that by proposition 3.2-7 all CCS-contexts satisfy the above restriction. By extending the modal language M with an infinite conjunction the construction of $\underline{A}$ can be generalized to arbitrary context systems.

Definition 3.3-1: For a context C define the transformer $I_C : M \longrightarrow M$ mapping "outer" properties to "inner" properties inductively as:

(1) $I_C(Tr) = Tr$

(2) $I_C(\langle b \rangle F) = \bigvee_{C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\twoheadrightarrow}} D} \langle u \rangle I_D(F)$

(3) $I_C(F \wedge G) = I_C(F) \wedge I_C(G)$

(4) $I_C(\neg F) = \neg I_C(F)$

where $F \vee G$ is an abbreviation for $\neg(\neg F \wedge \neg G)$ and for $u \varepsilon Act^*$ and $F \varepsilon M$, $\langle u \rangle F \varepsilon M$ is defined inductively as: $\langle \varepsilon \rangle F = F$ and $\langle au \rangle F = \langle a \rangle \langle u \rangle F$. Also $\bigwedge_{i \varepsilon \emptyset} F_i = Tr$ by convention. □

Note, that our finiteness restriction (F) on contexts ensures that the above definition is welldefined: especially that the disjunction in (2) is finite and thus expressible in M.

Our next theorem shows that $I_C(F)$ is the construction required in A, i.e. a sufficient and necessary condition for a property F to hold of $C[p]$ is that $I_C(F)$ holds of p:

Theorem 3.3-2: $C[p] \models F$ iff $p \models I_C(F)$ .

Proof: By structure on F.

F=Tr: Since $I_C(\mathrm{Tr}) = \mathrm{Tr}$ this clearly holds.

F=<b>G:

$C[p] \models \langle b\rangle G$

iff (defn $\models$)

$\exists q.\ C[p] \overset{b}{\longmapsto} q \ \&\ q \models G$

iff (IH, defn 3.1-1)

$\exists C', p', u.\ C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \ \&\ p \overset{u}{\rightarrow} p' \ \&\ p' \models I_{C'}(G)$

iff (defn $\models$)

$\exists C'.\ C \overset{b}{\underset{u}{\longmapsto\!\!\!\!\rightarrow}} C' \ \&\ p \models \langle u\rangle I_{C'}(G)$

iff (defn 3.3-1 (2))

$p \models I_C(\langle b\rangle G)$

F=G∧G′:

$C[p] \models G \wedge G'$

iff (defn $\models$)

$C[p] \models G$ and $C[p] \models G'$

iff (IH)

$p \models I_C(G)$ and $p \models I_C(G')$

iff (defn $\models$)

$p \models I_C(G) \wedge I_C(G')$

iff (defn 3.3-1 (3))

$p \models I_C(G \wedge G')$

F=¬G:

$C[p] \models \neg G$

iff (defn $\models$)

$C[p] \not\models G$

iff (IH)

$p \not\models I_C(G)$

iff (defn $\models$)

$p \models \neg I_C(G)$ iff (defn 3.3-1 (4)) $p \models I_C(\neg G)$ □

Proposition 3.3-3: For CCS-contexts the following holds:

$$(1)\quad I_{[\,]}(F) \equiv F$$

$$(2)\quad I_p(F) \equiv \begin{cases} Tr & ;\ p \models F \\ \neg Tr & ;\ \text{otherwise} \end{cases}$$

$$(3)\quad I_{a.C}(\langle b\rangle F) \equiv \begin{cases} \neg Tr & ;\ b \neq a \\ I_C(F) & ;\ \text{otherwise} \end{cases}$$

$$(4)\quad I_{C+D}(\langle b\rangle F) \equiv I_C(\langle b\rangle F) \vee I_D(\langle b\rangle F)$$

$$(5)\quad I_{C\restriction S}(\langle b\rangle F) \equiv \begin{cases} \neg Tr & ;\ b \notin S \\ I_C(\langle b\rangle F) & ;\ \text{otherwise} \end{cases}$$

$$(6)\quad I_{C[\Phi]}(\langle b\rangle F) \equiv \bigvee_{a.\Phi a = b} I_C(\langle a\rangle F)$$

$$(7)\quad I_{C|p}(\langle b\rangle F) \equiv \bigvee_{C \overset{b}{\underset{u}{\mapsto}} C'} \langle u\rangle I_{C'|p}(F) \;\vee\; \bigvee_{p \xrightarrow{u} p'} I_{C|p'}(F) \;\vee\; \Big[ \bigvee_{\substack{C \overset{c}{\underset{u}{\mapsto}} C' \\ p \xrightarrow{\bar{c}} p'}} \langle u\rangle I_{C'|p'}(F) \Big]_{b=1}$$

$$(8)\quad I_{C\&p}(\langle b\rangle F) \equiv \bigvee_{\substack{C \overset{b}{\underset{u}{\mapsto}} C' \\ p \xrightarrow{b} p'}} \langle u\rangle I_{C'\&p'}(F)$$

where $F \equiv G$ iff $\forall p \varepsilon Pr.\ p \models F \Leftrightarrow p \models G$.

Proof: By structure of F using definition 3.3-1 and proposition 3.2-6. □

Example 3.3-4: (From /St83/) Using the above proposition 3.3-3 let us verify that:

$$a.p + b.q \models \langle a\rangle Tr \wedge \langle b\rangle Tr \wedge \neg\langle c\rangle Tr$$

By theorem 3.3-2 it is sufficient and necessary to prove that:

$$b.q \models I_{a.p+[]}\left[\langle a\rangle Tr \wedge \langle b\rangle Tr \wedge \neg\langle c\rangle Tr\right]$$

We calculate, using proposition 3.3-3 and definition 3.3-1:

$$I_{a.p+[]}\left[\langle a\rangle Tr \wedge \langle b\rangle Tr \wedge \neg\langle c\rangle Tr\right]$$

$$= I_{a.p+[]}(\langle a\rangle Tr) \wedge I_{a.p+[]}(\langle b\rangle Tr) \wedge I_{a.p+[]}(\neg\langle c\rangle Tr)$$

$$\equiv \left(I_{a.p}(\langle a\rangle Tr) \vee I_{[]}(\langle a\rangle Tr)\right) \wedge$$
$$\left(I_{a.p}(\langle b\rangle Tr) \vee I_{[]}(\langle b\rangle Tr)\right) \wedge$$
$$\neg\left(I_{a.p}(\langle c\rangle Tr) \vee I_{[]}(\langle c\rangle Tr)\right)$$

$$\equiv (Tr \vee \langle a\rangle Tr) \wedge (\neg Tr \vee \langle b\rangle Tr) \wedge \neg(\neg Tr \vee \langle c\rangle Tr)$$

$$\equiv \langle b\rangle Tr \wedge \neg\langle c\rangle Tr$$

Thus, we must prove:

$$b.q \models \langle b\rangle Tr \wedge \neg\langle c\rangle Tr$$

By theorem 3.3-2 it is sufficient and necessary to prove that:

$$q \models I_{b.[]}\left[\langle b\rangle Tr \wedge \neg\langle c\rangle Tr\right]$$

We calculate again:

$$I_{b.[]}\left[\langle b\rangle Tr \wedge \neg\langle c\rangle Tr\right]$$

$$= I_{b.[]}(\langle b\rangle Tr) \wedge \neg I_{b.[]}(\langle c\rangle Tr)$$

$$\equiv Tr \wedge \neg(\neg Tr)$$

$$\equiv Tr$$

Obviously, $q \models Tr$. This concludes the proof. □

According to theorem 3.3-2, definition 3.3-1 gives a <u>uniform</u> and <u>universal</u> way of translating modal properties of a combined process into sufficient and necessary properties of the inner process. As such we have the basis for a complete axiomatization of correctness assertions, $p \models F$, as long as the process constructions operationally can be described as contexts. The axiomatization would simply have a rule of the form:

$$\frac{p \vdash I_C(F)}{C[p] \vdash F}$$

for each ("basic") context. For an acceptable system it still remains to find an expression for $I_C(F)$, uniform in F and structurally defined in C without any explicit reference to the operational behaviour of C. However, we know what the expression should be semantically and have thus a guide for our search.

From theorem 3.3-2 a solution to the second problem, B, is easily obtained. Extending $I_C$ to subsets (of modal formulas) in the usual way we have the following lemma:

Lemma 3.3-5: Let C be a context and B a subset of M. Then for all processes p and q:

$$p \sim_{I_C(B)} q \quad \Leftrightarrow \quad C[p] \sim_B C[q]$$

Proof: $p \sim_{I_C(B)} q$ iff $M(p) \cap I_C(B) = M(q) \cap I_C(B)$ iff $\forall F \varepsilon B.\ p \models I_C(F) \Leftrightarrow q \models I_C(F)$ iff (thm 3.3-2) $\forall F \varepsilon B.\ C[p] \models F \Leftrightarrow C[q] \models F$ iff $C[p] \sim_B C[q]$. □

From the above lemma it follows immediately that $A = I_C(B)$ gives the least discriminating set of formulas such that whenever p and q are processes then:

$$p \sim_A q \quad \Rightarrow \quad C[p] \sim_B C[q]$$

Corollary 3.3-6: Let C be a context and B a subset of M. Then for all processes p and q:

(i) $p \sim_{I_C(B)} q \quad \Rightarrow \quad C[p] \sim_B C[q]$

Moreover, if A is a subset of M such that (i) holds, then A is more discriminating than $I_C(B)$. I.e. whenever p and q are processes, then:

(ii) $p \sim_A q \quad \Rightarrow \quad p \sim_{I_C(B)} q$ □

<u>Example 3.3-7:</u> Consider the CCS-context:

$$C = (\mu x.a.x \mid [\,])\restriction\{a\}$$

We want to prove that $C[p] \sim C[q]$ for all processes p and q (and thus $C[p] \sim C[\mathbb{O}] \sim \mu x.a.x$ for all processes p). We first note that the operational behaviour of C is given by:

$$C \xmapsto[0]{a} C \qquad \text{and} \qquad C \xmapsto[a]{a} C$$

Now $C[p] \sim C[q]$ iff $C[p] \sim_M C[q]$ so by lemma 3.3-5 a necessary and sufficient condition is:

$$p \sim_{I_C(M)} q$$

We prove by structure that for all formulas F either $I_C(F) \equiv Tr$ or $I_C(F) \equiv \neg Tr$. The only interesting case is when F is of the form $\langle b\rangle G$:

If $a \neq b$ then $I_C(\langle b\rangle G) = \neg Tr$. Otherwise $I_C(\langle a\rangle G) = \langle a\rangle I_C(G) \vee I_C(G)$. By induction hypothesis either $I_C(G) \equiv Tr$ or $I_C(G) \equiv \neg Tr$. In the former case $I_C(\langle a\rangle G) \equiv Tr$. Otherwise $I_C(\langle a\rangle G) \equiv \langle a\rangle \neg Tr \vee \neg Tr \equiv \neg Tr$, since $\langle a\rangle \neg Tr \equiv \neg Tr$.

Thus $I_C(M) \subseteq \{F \mid \forall p.\ p \models F \vee \forall p.\ p \not\models F\}$ and therefore always $p \sim_{I_C(M)} q$. □

## 3.4 CONTEXTS AS ENVIRONMENT TRANSFORMERS

In this section we shall investigate how contexts transform environments. More specifically, we are interested in the following problem:

> Given a context, C, and an (outer) environment, e, we want to find an (inner) environment, f, such that for all processes p and q:
>
> (*) $\qquad p \sim_f q \quad \Rightarrow \quad C[p] \sim_e C[q]$
>
> Preferably the environment, f, described is as small as possible wrt. the discrimination ordering $\sqsubseteq$.

From the results of the previous section and the modal characterization result of section 2.3, f will satisfy (*) iff:

$$I_C\big(L(e)^+\big) \;\sqsubseteq\; L(f)^+$$

However, we know very little about the discrimination ordering between sets of modal properties so the above condition will be difficult to verify in general. Instead we would like a condition based directly on the operational behaviours of e,f and C and ideally a condition of the form:

$$\min(C,e) \sqsubseteq f$$

where min(C,e) is a minimal environment wrt. $\sqsubseteq$ satisfying (*). Such a condition should be simple to check since (for image-finite environments) we know by theorem 2.4-20 that $\sqsubseteq = \leqslant$.

Now, by the very definition of parameterized bisimulation (definition 2.2-1), in the antecedent of (*), f must interact identically with p and q, whereas the equivalence $C[p] \sim_e C[q]$ may hold by C interacting differently with p and q (see example 3.3-7 for such a situation). For this reason we expect the behaviour of min(C,e) - when and if it exists - to be extremely complicated. We shall therefore instead look for a weakest environment f (wrt. $\sqsubseteq$ ) such that for all processes p and q:

$$(**) \qquad p \sim_f q \;\Rightarrow\; \langle C,p\rangle \equiv_e \langle C,q\rangle$$

where $\langle C,p\rangle \equiv_e \langle C,q\rangle$ roughly means that $C[p] \sim_e C[q]$ with C interacting identically with p and q. Thus any environment, f, satisfying (**) will also satisfy (*).

We shall call the weakest environment (wrt. $\sqsubseteq$ ) satisfying (**) the weakest inner environment of e under C, and use the notation $wie_{\mathbb{E}}(C,e)$. The questions to be investigated in the following are then: "When does $wie_{\mathbb{E}}(C,e)$ exist ?" and if it does exist: "What is its behaviour ?" Clearly, the answers will depend upon the environment system, $\mathbb{E}$, in question.

For environment system, $\mathbb{E}$, closed under a non-swallowing context system $\mathbb{C}$ it turns out that we can find an environment f such that for all processes p and q:

$$(***) \qquad p \sim_f q \;\Leftrightarrow\; \langle C,p\rangle \equiv_e \langle C,q\rangle$$

In this case f is obviously a suitable choise for $wie_{\mathbb{E}}(C,e)$.

For cases when $\mathbb{E}$ is not closed under $\mathbb{C}$ we give various sufficient conditions which will ensure existence of $wie_{\mathbb{E}}(C,e)$. It is shown that language environments, $\mathbb{L}$, satisfies these conditions wrt. (a subset of) CCS-contexts.

+ If C does not interact identically with p and q we would not expect to be able to reduce an equivalence between C p and C q to an equivalence between p and q anyhow.

## 3.4.1 Wie for Closed Environment Systems.

First let us formally define the (parameterized) relation, $\equiv$, used in (**).

Definition 3.4-1: Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ be a process system and let $\mathbb{C} = (Con, Act_o \times Act_o, \longmapsto)$ be a context system. Then define the process system $\underline{\mathbb{P}\text{-}\mathbb{C}}$ as $(Con \times Pr, Con \times Act \times Act^*, \longrightarrow)$, where for all $C, C', C'' \varepsilon Con$, $p, p' \varepsilon Pr$, $b \varepsilon Act$ and $u \varepsilon Act^*$, $\longrightarrow$ satisfies:

$$\langle C,p\rangle \xrightarrow{(C'',b,u)} \langle C',p'\rangle \quad \Leftrightarrow$$
$$C''=C' \ \& \ C \overset{b}{\underset{u}{\longmapsto}} C' \ \& \ p \overset{u}{\longrightarrow} p'$$ □

The intuition is that we encode information about the interaction between C and p in the labelling of derivations of $\langle C,p\rangle$ (following a suggestion by Peter Aczel).

Definition 3.4-2: Let $\mathbb{E} = (Env, Act, \Longrightarrow)$ be an environment system and let $\mathbb{C} = (Con, Act_o \times Act_o, \longmapsto)$ be a context system. Then define the environment system $\underline{\mathbb{E}\text{-}\mathbb{C}}$ as $(Env, Con \times Act \times Act^*, \Longrightarrow)$, where for all $e, e' \varepsilon Env$, $C \varepsilon Con$, $b \varepsilon Act$ and $u \varepsilon Act^*$, $\Longrightarrow$ satisfies:

$$e \xRightarrow{(C,b,u)} e' \quad \Leftrightarrow \quad e \overset{b}{\Longrightarrow} e'$$ □

Since $\mathbb{E}\text{-}\mathbb{C}$ is an environment system over the same action set as $\mathbb{P}\text{-}\mathbb{C}$ we have the notion of an $\mathbb{E}\text{-}\mathbb{C}$-parameterized bisimulation (definition 2.2-1) over $\mathbb{P}\text{-}\mathbb{C}$. We shall write $\langle C,p\rangle \equiv_e \langle C,q\rangle$ iff there is an $\mathbb{E}\text{-}\mathbb{C}$-parameterized bisimulation, R, over $\mathbb{P}\text{-}\mathbb{C}$ such that $(\langle C,p\rangle, \langle C,q\rangle) \varepsilon R_e$.

By the construction of the action set and the restrictions made on the derivation relation of $\mathbb{P}\text{-}\mathbb{C}$ it is clear that if $\langle C,p\rangle \equiv_e \langle C,q\rangle$, then C must interact identically with p and q. Thus, we might have a situation where $C[p] \sim_e C[q]$ but not $\langle C,p\rangle \equiv_e \langle C,q\rangle$.

Example 3.4-3: Recall example 3.3-7. That is, let C be a context with the operational behaviour given by the two rules:

$$C \overset{a}{\underset{0}{\mapsto}} C \qquad \text{and} \qquad C \overset{a}{\underset{a}{\mapsto}} C$$

Then we know from 3.3-7 that $C[a.\mathbb{O}] \sim_U C[\mathbb{O}]$. However, in the above equivalence C does not interact identically with $a.\mathbb{O}$ and $\mathbb{O}$: in the behaviour of $C[\mathbb{O}]$ the transduction $C \overset{a}{\underset{a}{\mapsto}} C$ is never used whereas it can be used in the behaviour of $C[a.\mathbb{O}]$. For this reason we would expect $\langle C, a.\mathbb{O}\rangle \not\equiv_U \langle C,\mathbb{O}\rangle$. To verify this, note that $U \xRightarrow{(C,a,a)} U$ and $\langle C, a.\mathbb{O}\rangle \xrightarrow{(C,a,a)} \langle C,\mathbb{O}\rangle$ (since $C \overset{a}{\underset{a}{\mapsto}} C$ and $a.\mathbb{O} \xrightarrow{a} \mathbb{O}$) but $\langle C,\mathbb{O}\rangle \xrightarrow{(C,a,a)}\!\!\!\!\!\!\!\!/\;\;\;$ (since $\mathbb{O} \overset{a}{\not\rightarrow}$). $\square$

On the other hand if $\langle C,p\rangle \equiv_e \langle C,q\rangle$ has been established then $C[p] \sim_e C[q]$ will also hold:

Theorem 3.4-4: Let $\mathbb{P}$ be closed under $\mathbb{C}$. Then whenever $\langle C,p\rangle \equiv_e \langle C,q\rangle$ also $C[p] \sim_e C[q]$.

Proof: It is easily shown that the Env-indexed family, R, with:

$$R_e = \{(C[p],C[q]) \mid \langle C,p\rangle \equiv_e \langle C,q\rangle\}$$

is an $\mathbb{E}$-parameterized bisimulation. $\square$

If $\mathbb{E}$ is closed under $\mathbb{C}$ and $\mathbb{C}$ is non-swallowing, then for any context C and environment e, we can find an environment f such that for all processes p and q:

$$p \sim_f q \quad \Leftrightarrow \quad \langle C,p\rangle \equiv_e \langle C,q\rangle$$

Not surprisingly, it turns out that a suitable choice for f is simply the combined environment $e[C]$ (see definition 3.1-9).

Theorem 3.4-5: Let $\mathbb{E}$ be closed under $\mathbb{C}$. Then whenever $C\in Con$, $p,q\in Pr$ and $e\in Env$ the following holds:

(1) $\quad p \sim_{e[C]} q \quad \Rightarrow \quad \langle C,p\rangle \equiv_e \langle C,q\rangle$

If $\mathbb{C}$ moreover is non-swallowing then also:

$$(2) \quad \langle C,p\rangle \equiv_e \langle C,q\rangle \Rightarrow p \sim_{e[C]} q \qquad \square$$

Note that the system of CCS-contexts, $\mathbb{C}_{CCS}$, is non-swallowing.

<u>Corollary 3.4-6:</u> If $\mathbb{E}$ is closed under $\mathbb{C}$ and $\mathbb{C}$ is non-swallowing then for all contexts $C$ and environments $e$, we can define $wie_{\mathbb{E}}(C,e) = e[C]$.

<u>Proof (of theorem 3.4-5):</u>
(1) We show that $R$ with $R_e = \{(\langle C,p\rangle,\langle C,q\rangle) \mid p \sim_{e[C]} q\}$ is an $\mathbb{E}$-$\mathbb{C}$-parameterized bisimulation.
So let $(\langle C,p\rangle,\langle C,q\rangle)\varepsilon R_e$. Assume $e \xRightarrow{(C',b,u)} e'$ and $\langle C,p\rangle \xrightarrow{(C'',b,u)} \langle C',p'\rangle$. Then $e \overset{b}{\Rightarrow} e'$ (in $\mathbb{E}$), $C''=C'$ $C \overset{b}{\underset{u}{\mapsto}} C'$ and $p \xrightarrow{u} p'$. There are two cases to consider:

<u>$u=\varepsilon$:</u> Then $p=p'$ and by lemma 3.1-17 $e'[C'] \leqslant e[C]$. Thus also $p \sim_{e'[C']} q$. Obviously, $\langle C,q\rangle \xrightarrow{(C',b,\varepsilon)} \langle C',q\rangle$ is a matching move.

<u>$u \neq \varepsilon$:</u> Then by lemma 3.1-10 $e[C] \overset{u}{\Rightarrow} e'[C']$. Since $p \sim_{e[C]} q$, $q \xrightarrow{u} q'$ with $p' \sim_{e'[C']} q'$ for some $q'$. Hence, $\langle C,q\rangle \xrightarrow{(C',b,u)} \langle C',q'\rangle$ which is a matching move.

(2) Recall that a context $C$ is non-swallowing iff $C \overset{0}{\underset{a}{\mapsto}} C' \Rightarrow a=0$ & $C=C'$. We show that $R$ with:

$$R_f = \{(p,q) \mid \exists C.\exists e.\ f=e[C] \ \&\ \langle C,p\rangle \equiv_e \langle C,q\rangle\}$$

is an $\mathbb{E}$- parameterized bisimulation. So let $(p,q)\varepsilon R_{e[C]}$. Assume $e[C] \overset{b}{\Rightarrow} f$ and $p \xrightarrow{b} p'$. Then for some $u\varepsilon Act^*$, $e'\varepsilon Env$ and $C'\varepsilon Con$, $e \overset{u}{\Rightarrow} e'$, $C \overset{u}{\underset{b}{\mapsto}} C'$ and $f=e'[C']$. Since $C$ is non-swallowing $|u| \geqslant 1$.
Then in $\mathbb{E}$-$\mathbb{C}$ $e \xRightarrow{(C',u,b)} e'$ and in $\mathbb{P}$-$\mathbb{C}$ $\langle C,p\rangle \xrightarrow{(C',u,b)} \langle C',p'\rangle$ (we have actually extended $\Rightarrow$ and $\rightarrow$ to be labelled with elements of $Con \times Act^* \times Act^*$ in the obvious way)

Since $\langle C,p\rangle \equiv_e \langle C,q\rangle$, therefore $\langle C,q\rangle \xrightarrow{(C',u,b)} \langle C',q'\rangle$ with $\langle C',p'\rangle \equiv_{e'} \langle C',q'\rangle$ for some $q'$ such that $q \xrightarrow{b} q'$. This is obviously a matching move for q. $\square$

It is important to realize that the second part of theorem 3.4-5 only holds provided $\mathbb{C}$ is non-swallowing. Let namely:

$$C_0 \overset{0}{\underset{a}{\mapsto}} C_1 \overset{a}{\underset{a}{\mapsto}} C_2$$

$$e_0 \overset{a}{\Rightarrow} e_1$$

then both $\langle C_0,a.\mathbb{O}\rangle$ and $\langle C_0,\mathbb{O}\rangle$ has no moves at all. Hence trivially $\langle C_0,a.\mathbb{O}\rangle \equiv_{e_0} \langle C_0,\mathbb{O}\rangle$. However, $e_0[C_0] \overset{a}{\Rightarrow}$, and therefore $a.\mathbb{O} \not\sim_{e_0[C_0]} \mathbb{O}$.

## 3.4.2 Wie for General Environment Systems.

In the previous section we showed that $wie_{\mathbb{E}}(C,e)$ always exists provided the environment system $\mathbb{E}$ is closed under the context system $\mathbb{C}$, and $\mathbb{C}$ is non-swallowing. If $\mathbb{E}$ is not closed under $\mathbb{C}$ the weakest inner environment may not exist. We shall in this section give (sufficient) conditions which will insure existence of $wie_{\mathbb{E}}(C,e)$ in such cases.

Our strategy is very simple: first close $\mathbb{E}$ under $\mathbb{C}$ (which is assumed to be non-swallowing) giving the extension $\mathbb{E}_{\mathbb{C}}$ (see definition 3.1-12). From the previous section we know that $wie_{\mathbb{E}_{\mathbb{C}}}(C,e)$ exists and is simply $e[C]$. Since $\mathbb{E}_{\mathbb{C}}$ is an extension of $\mathbb{E}$, $wie_{\mathbb{E}}(C,e)$ exists iff there is a smallest environment, f, of $\mathbb{E}$ with respect to $\sqsubseteq$ such that $e[C] \sqsubseteq f$.

Now assume we can find a smallest (wrt. $\leqslant$) environment f of $\mathbb{E}$ such that $e[C] \leqslant f$. We shall use the notation <u>$ba_{\mathbb{E}}(C,e)$</u> (<u>b</u>est <u>a</u>pproximation) for this environment.

Since $\leqslant\subseteq\sqsubseteq$ (theorem 2.4-10) we always have $e[C]\sqsubseteq ba_{\mathbb{E}}(C,e)$.

If moreover $\mathbb{E}_{\mathbb{C}}$ is image-finite, then by the Main Theorem 2.4-20, $\sqsubseteq\subseteq\leqslant$. Hence if $g$ is any environment of $\mathbb{E}$ such that $e[C]\sqsubseteq g$ then by the property of $ba_{\mathbb{E}}(C,e)$ also $ba_{\mathbb{E}}(C,e)\sqsubseteq g$. Thus $ba_{\mathbb{E}}(C,e)$ is the smallest environment of $\mathbb{E}$ wrt. $\sqsubseteq$ such that $e[C]\sqsubseteq ba_{\mathbb{E}}(C,e)$ and we can therefore take $wie_{\mathbb{E}}(C,e) = ba_{\mathbb{E}}(C,e)$. Note, that if the Main Theorem 2.4-20 should extend to image-infinite cases, we can in all cases take $wie_{\mathbb{E}}(C,e)$ to be $ba_{\mathbb{E}}(C,e)$.

What remains to be done now is to find conditions which will ensure image-finiteness of $\mathbb{E}_{\mathbb{C}}$ and existence of $ba_{\mathbb{E}}(C,e)$. For the former the following will suffice:

<u>Lemma 3.4-6:</u> If $\mathbb{E}$ is image-finite and for all contexts, $C$, of $\mathbb{C}$ and actions $b\varepsilon Act$ the set $\{(u,C') \mid C\overset{u}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}}C'\}$ is finite, then $\mathbb{E}_{\mathbb{C}}$ is image-finite.

<u>Proof:</u> Directly from lemma 3.1-10. $\square$

Unfortunately not all CCS-contexts have the above property, especially not contexts involving the $|$ operator: let $C = (\mu x.a.x \mid [\,])$ then obviously for any $n\varepsilon\omega$: $C\overset{ba^n}{\underset{b}{\mapsto\!\!\!\twoheadrightarrow}}C$ which violates the above property. However, for CCS-contexts with no occurrences of $|$ the property can be shown to hold. What we really need in order to allow all CCS-contexts, is to extend the Main Theorem 2.4-10 to image-infinite cases. However - as we have mentioned earlier - such an extension is left as an open problem (which we conjecture to hold).

For existence of $ba_{\mathbb{E}}(C,e)$ it suffices that $\mathbb{E}$ is closed under &:

<u>Lemma 3.4-7:</u> If $\mathbb{E}$ is closed under & then:

$$ba_{\mathbb{E}}(C,e) \simeq \mathop{\&}_{f\varepsilon Env.\, e[C]\leqslant f} f$$

Proof: Follows directly from the greatest lower bound property of & wrt. $\leqslant$. □

Now let $(L_i)_{i\varepsilon I}$ be any family of language environments. Then:

$$\mathop{\&}_{i\varepsilon I} L_i \sim \bigcap_{i\varepsilon I} L_i^p$$

since it is easily shown that $\bigcap_{i\varepsilon I} L_i^p$ is a greatest lower bound (wrt. $\leqslant$) of $(L_i)_{i\varepsilon I}$ using the characterization of $\leqslant$ for language environments given in theorem 2.2-17. Thus, $\mathbb{L}$ is closed under & and from the previous lemma $ba_{\mathbb{L}}(C,L)$ therefore always exists.

As a simple generalization of theorem 2.2-17 it can be shown that if e is any environment and L is any language environment, then:

$$e \leqslant L \quad \Leftrightarrow \quad D(e) \subseteq L^p$$

where D(e) is the "language" of e, defined by:

$$D(e) = \left\{ u\varepsilon Act^* \mid e \overset{u}{\Longrightarrow} \right\}$$

(Note, D(e) is always prefixed closed). Hence, from lemma 3.4-7 and proposition 3.4-8 it follows that for C a context and L a language environment:

$$\begin{aligned} ba_{\mathbb{L}}(C,L) &\simeq \mathop{\&}_{L[C] \leqslant M} M \\ &\simeq \bigcap_{L[C] \leqslant M} M^p \\ &\simeq D\big(L[C]\big) \end{aligned}$$

Using lemma 3.1-10 we have:

$$\begin{aligned} D\big(L[C]\big) &= \{\varepsilon\} \cup \left\{ u\varepsilon Act^+ \mid \exists v\varepsilon Act^* .\ L \overset{v}{\Longrightarrow} \ \&\ C \overset{v}{\underset{u}{\longmapsto\!\!\!\twoheadrightarrow}} \right\} \\ &= \left\{ u\varepsilon Act^* \mid \exists v\varepsilon L^p .\ C \overset{v}{\underset{u}{\longmapsto\!\!\!\twoheadrightarrow}} \right\} \end{aligned}$$

Thus, we can simply define:

Definition 3.4-9: $ba_{\mathbb{L}}(C,L) = \{u \in Act^* \mid \exists v \in L^p.\ C \overset{v}{\underset{u}{\longmapsto\!\!\!\!\twoheadrightarrow}}\}$ □

From this definition it is easily shown that $ba_{\mathbb{L}}(C,L)$ satisfies the following:

Proposition 3.4-10:

(i) $ba_{\mathbb{L}}(C,\emptyset) = \emptyset$

(ii) $ba_{\mathbb{L}}(C, \bigcup_i L_i) = \bigcup_i ba_{\mathbb{L}}(C,L_i)$

(iii) $ba_{\mathbb{L}}(C, \bigcap_i L_i) \subseteq \bigcap_i ba_{\mathbb{L}}(C,L_i)$

(iv) $ba_{\mathbb{L}}(C \circ D,L) = ba_{\mathbb{L}}(D, ba_{\mathbb{L}}(C,L))$ □

For CCS-contexts the following holds:

Proposition 3.4-11:

(i) $ba_{\mathbb{L}}(C,L) = \{\varepsilon\}$ if $[\,] \notin free(C)$, $L \neq \emptyset$

(ii) $ba_{\mathbb{L}}([\,],L) = L^p$

(iii) $ba_{\mathbb{L}}(a.C,L) = ba_{\mathbb{L}}(C, \partial L/\partial a)$

(iv) $ba_{\mathbb{L}}(C + D,L) = ba_{\mathbb{L}}(C,L) \cup ba_{\mathbb{L}}(D,L)$

(v) $ba_{\mathbb{L}}(C \,\&\, p,L) = ba_{\mathbb{L}}(C, D(p) \cap L^p)$

(vi) $ba_{\mathbb{L}}(C \mid p,L) = ba_{\mathbb{L}}\left(C, \{u \mid (u \# D(p)) \cap L^p \neq \emptyset\}\right)$

(vii) $ba_{\mathbb{L}}(C \lceil S,L) = ba_{\mathbb{L}}(C, L^p \cap S^*)$

(viii) $ba_{\mathbb{L}}(C[\Phi],L) = ba_{\mathbb{L}}(C, \Phi^{-1}(L^p))$

where $\#$ and $\Phi^{-1}$ have been extended to sets of strings in the obvious ways.

Proof: Direct from definition 3.4-9 and proposition 3.2-6. □

Example 3.4-12: We want to show:

$$[\mu x.(a.b.x) \mid \mu x.(\bar{a}.w.\bar{b}.x)] \lceil \{w,1\} \sim$$
$$[\mu x.(a.b.x) \mid \mu x.(\bar{a}.w.\bar{b}.x + \bar{b}.\mathbb{O})] \lceil \{w,1\}$$

Let $C = [\mu x.(a.b.x) \mid [\,]] \restriction \{w,1\}$. Then it is sufficient to prove that:

$$\mu x.(\bar{a}.w.\bar{b}.x) \sim_{ba_{\mathbb{L}}(C,Act^*)} \mu x.(\bar{a}.w.\bar{b}.x + \bar{b}.\mathbb{O})$$

So let us calculate $ba_{\mathbb{L}}(C,Act^*)$ using proposition 3.4-11.

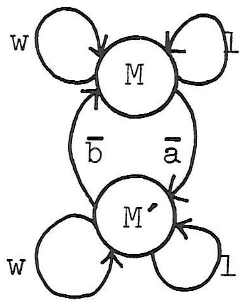
$$ba_{\mathbb{L}}(C,Act^*) =_{(vii)}$$

$$ba_{\mathbb{L}}([\mu x.(a.b.x) \mid [\,]], \{w,1\}^*) =_{(vi)}$$

$$\{u \mid (u \# (ab)^{*p}) \cap \{w,1\}^* \neq \emptyset\} =$$

$$[(\{w,1\}^*.\bar{a}.\{w,1\}^*.\bar{b})^*]^p$$

Let M denote the above language. Then the behaviour of M is given by the following diagram:




It is easily verified that R, with:

$$R_M = \{(\mu x.(\bar{a}.w.\bar{b}.x)\ ,\ \mu x.(\bar{a}.w.\bar{b}.x + \bar{b}.\mathbb{O}))\}$$

$$R_{M'} = \{(w.\bar{b}.\mu x.(\bar{a}.w.\bar{b}.x)\ ,\ w.\bar{b}.\mu x.(\bar{a}.w.\bar{b}.x + \bar{b}.\mathbb{O}))\ ;$$
$$(\bar{b}.\mu x.(\bar{a}.w.\bar{b}.x)\ ,\ \bar{b}.\mu x.(\bar{a}.w.\bar{b}.x + \bar{b}.\mathbb{O}))\}$$

$$R_L = \emptyset\ ;\quad L \neq M \text{ and } L \neq M'$$

is an $\mathbb{L}$-parameterized bisimulation. □

## 3.5 CONCLUDING REMARKS

In this chapter we have studied contexts as objects which semantically behaves like action transducers. This view has enabled us to define the behaviour of a combined process, C[p], from the behaviours of the context C and the inner process p.

As an example a class of CCS-contexts - being certain CCS-process expressions with free variables contained in $\{[\,]\}$ - has been described operationally, and it has been shown that the behaviour of a CCS-process of the form $C\{p/[\,]\}$ is exactly that expected of the combined process C[p].

In section 3.3 it is shown how contexts transform modal properties: under certain finiteness conditions (satisfied by all CCS-contexts) on the context C, a property transformer $I_C$ has been defined such that for any property F and process p:

$$C[p] \models F \quad \Leftrightarrow \quad p \models I_C(F)$$

Furthermore for all p,q Pr and $A \subseteq M$:

$$p \sim_{I_C(A)} q \quad \Leftrightarrow \quad C[p] \sim_A C[q]$$

which shows how to reduce a parameterized equivalence problem involving combined processes to a parameterized equivalence problem involving only the inner processes.

For the environment-parameterized version of $\sim$, a slightly weaker result has been obtained in section 3.4 (weaker maybe because environments are less expressive than sets of modal properties): for environment systems closed under a non-swallowing context system (satisfied by all CCS-contexts) there exists an environment transformer, $wie_{\mathbb{E}}(C,\_)$, such that for any $p,q \in Pr$ and $e \in Env$:

$$p \sim_{wie_{EE}(C,e)} q \Leftrightarrow \langle C,p\rangle \equiv_e \langle C,q\rangle$$

where $\langle C,p\rangle \equiv_e \langle C,q\rangle$ roughly means that $C[p] \sim_e C[q]$ with C interacting identically with p and q. The transformer $wie_{EE}(C,\_)$ is simply the map $wie_{EE}(C,\_)$: $e \mapsto e[C]$.

For environment systems not closed under the context system, conditions have been given which ensure the existence of an environment transformer, $wie_{EE}(C,\_)$, such that for any $p,q \in Pr$ and $e \in Env$, $wie_{EE}(C,e)$ is the weakest (wrt. $\sqsubseteq$ ) environment such that:

$$p \sim_{wie_{EE}(C,e)} q \Rightarrow \langle C,p\rangle \equiv_e \langle C,q\rangle$$

Our notion of (action) transduction as the semantics of contexts has strong similarities to the <u>causality relation</u>, $\rightarrow$, defined in /San82/: For contexts C,D and actions a,b /San82/ defines:

$C \xrightarrow[a]{b} D$ iff whenever a proof of $p \xrightarrow{a} q$ is given it is possible to construct a proof of $C[p] \xrightarrow{b} D[q]$.

$C \xrightarrow{b} D$ iff it is always possible to construct a proof of $C[p] \xrightarrow{b} D[p]$ for any process p.

However, the causality relation in /San82/ is defined and investigated only for (a subset of our) CCS-contexts, and is used for finding conditions ensuring unique solutions to equations of the form $C[p] \approx p$, where $\approx$ is the weak bisimulation equivalence (see also chapter 5). In contrast to this we have been working with a <u>general</u> and <u>abstract</u> notion of context (of which CCS-contexts is an example). Thus our results hold for any (future) process construction as long as the construction can be described operationally as an action transducer (=context).
Normally a process construction, O, is introduced semantically by a (finite) set of <u>inferencerules</u> describing

the behaviour of combined processes of the form $O(p)$ (or $O(p_1,\ldots,p_n)$ if $O$ is an n-ary process construction). As such there is no a priori guarantee that $O$ can be described as a context. In fact it is very easy in this way to introduce constructions which can not be described as contexts; e.g. let the semantics of $O$ be given by the following rule:

$$\frac{p \xrightarrow{a} p' \qquad b \in sort(p)}{O(p) \xrightarrow{a} O(p')}$$

where sort(p) is the set of all actions occurring in the syntax of p. The only possible semantics of $O$ as a context is $O \underset{a}{\overset{a}{\longmapsto}} O$ and thus we should have $O(p) \xrightarrow{a} O(p')$ whenever $p \xrightarrow{a} p'$. However, this is not true since $O$ makes certain demands to the syntax (structure) of the inner process p. It seems that for a process construction to be describable as a context, it must only exploit the inner process' ability to produce actions and not its structure.

An interesting future problem would be to find conditions on the type of inference rules allowed for a construction in order to ensure describability as a context. The conditional behaviour rules examined in /Sim85/ seems a good candidate for such conditions. It is also interesting to note that a set of MEIJE-SCCS contexts (called architectural expressions) is introduced in the above paper which is very similar to the CCS-contexts studied in section 3.2: an architectural expression is a process expression such that every free variable occurs at most once and outside the scope of recursive definitions.

An obvious limitation in our work is that only unary contexts have been considered. A natural extension would be to consider n-ary contexts as well, where intuitively an n-ary context produces an external action by consuming (up to) n inner actions. Thus, the operational semantics of a set of n-ary contexts, $C_n$, could be described by a transduction relation with the following functionality: $\mapsto \subseteq C_n \times Act_o^n \times Act_o \times C_n$. With this extension we should be able to describe the + and & operator as dyadic contexts with the following operational semantics:

$$+ \xmapsto[(a,0)]{a} P_1^2 \qquad + \xmapsto[(0,a)]{a} P_2^2$$

$$\& \xmapsto[(a,a)]{a} \&$$

where:

$$P_1^2 \xmapsto[(a,0)]{a} P_1^2 \qquad P_2^2 \xmapsto[(0,a)]{a} P_2^2$$

Such an extension is left for future work.

Since the operational behaviour of contexts is described by a transition system of the form $\mathbb{C} = (Con, Act_o \times Act_o, \mapsto)$ we can apply the general notion of bisimulation equivalence, $\sim$, to $\mathbb{C}$. The modal property transformer associated with a context suggest another equivalence, $\sim_1$, between contexts:

$$C \sim_1 D \quad \Leftrightarrow \quad I_C \equiv I_D$$

where $I_C \equiv I_D$ iff $\forall F \varepsilon M. \{ p \mid p \models I_C(F)\} = \{p \mid p \models I_D(F)\}$. Finally, we have an equivalence, $\sim_2$, between contexts based on their extensionality. I.e.:

$$C \sim_2 D \quad \Leftrightarrow \quad \forall p.\ C[p] \sim D[p]$$

An interesting (future) problem is to determine the relationship between these three equivalences. Provided the assumptions for theorem 2.3-2 and theorem 3.3-2 hold it is easy to show that $\sim_1 = \sim_2$. It is also easy to prove that $\sim \subseteq \sim_2$, whereas the inclusion $\sim_2 \subseteq \sim$ - not unexpectedly - seems hard to prove. Maybe a technique similar to the one used for the Main Theorem in section 2.4.2 can be used.

# CHAPTER 4

# COMPLETE PROOF SYSTEMS

In this chapter we shall present complete proof systems (or inference systems) for the (environment) parameterized equivalence problem, $p \sim_e q$, for various combinations of the environment and process systems.

In section 4.1 a complete proof system for finite environments and processes is given, extending the complete axiomatization for the corresponding unparameterized equivalence problem in /HenMil83/. It is also shown how to derive a (relative) complete proof system for language environments and finite processes.

In sections 4.2 and 4.3 two alternative complete proof systems for regular environments and processes are presented. The first system extends the complete system for the corresponding unparameterized equivalence problem in /Mil82/. The second system is based on a reduction of parameterized equivalences involving regular environments and processes to corresponding parameterized equivalences, where the environments are finite. The reduction defined is similar to the results concerning Moore experiments on finite automata /Mo56,Con71,Sal66/.

For reasons of notational convenience we shall throughout the remainder of this chapter use a linearised version, $e \models p = q$, for $p \sim_e q$. The notation suggests that an environment acts as an assumption (made about an outer context) under which two processes are equivalent.

## 4.1 COMPLETE PROOF SYSTEMS FOR FINITE AND DETERMINISTIC BEHAVIOURS

First let us define the two transition systems of finite processes and environments, $\mathbb{P}_f$ and $\mathbb{E}_f$: Let $\mathbb{P}_f = (P_f, Act, \longrightarrow)$ where $P_f$ consists of the following terms:

$$p ::= \mathbb{O} \mid a.p \mid p + p'$$

and the operational semantics $(\longrightarrow)$ is the standard one (see section 3.2). Let $\mathbb{E}_f$ be $\mathbb{P}_f$ extended with a universal environment U, i.e.: $\mathbb{E}_f = (E_f, Act, \Longrightarrow)$ where $E_f = P_f \cup \{U\}$ and $\Longrightarrow \; = \; \longrightarrow \cup \{(U,a,U) \mid a \varepsilon Act\}$.

We recall the complete axiomatization of the unparameterized bisimulation equivalence for $\mathbb{P}_f$ given in /HenMil83/.

Theorem 4.1-1: The bisimulation equivalence $\sim$ over $\mathbb{P}_f$ is exactly the congruence induced by the following axioms:

$$\begin{array}{ll}
(A1) & p + (q + r) = (p + q) + r \\
(A2) & p + q = q + p \\
(A3) & p + p = p \\
(A4) & p + \mathbb{O} = p
\end{array}$$

□

In the proof of the above theorem it is used that any process, p, (of $\mathbb{P}_f$) can be (provably) brought into sumform: an expression p is on sumform iff for some $a_0, \ldots, a_{n-1} \varepsilon Act$ and $p_0, \ldots, p_{n-1} \varepsilon P_f$, p is of the form:

$$p = a_0.p_0 + \ldots + a_{n-1}.p_{n-1}$$

where for all $i<n$, $p_i$ is on sumform as well. By convention $p = \mathbb{O}$ if $n=0$. Note that by (A1)-(A3) the above notation is unambiguous up to provable equivalence.

We now present the proof system, $S_{ff}$, for parameterized equivalence over $\mathbb{E}_f$ and $\mathbb{P}_f$:

| | | |
|---|---|---|
| SUM | S1. | $U \vdash p+(q+r) = (p+q)+r$ |
| | S2. | $U \vdash p+q = q+p$ |
| | S3. | $U \vdash p+p = p$ |
| | S4. | $U \vdash p+\emptyset = p$ |
| EQUIV | E1. | $e \vdash p = p$ |
| | E2. | $\dfrac{e \vdash p = q}{e \vdash q = p}$ |
| | E3. | $\dfrac{e \vdash p = q \qquad e \vdash q = r}{e \vdash p = r}$ |
| CONG | C1. | $\dfrac{U \vdash p = q}{U \vdash a.p = a.q}$ |
| | C2. | $\dfrac{e \vdash p = q}{a.e \vdash a.p = a.q}$ |
| | C3. | $\dfrac{e \vdash p = q}{e \vdash r+p = r+q}$ |
| CONS | | $\dfrac{e \leqslant f \qquad f \vdash p = q}{e \vdash p = q}$ |
| NIL | | $\emptyset \vdash p = q$ |
| COMB | | $\dfrac{e \vdash p = q \qquad f \vdash p = q}{e+f \vdash p = q}$ |
| ANNIHIL | | $\dfrac{a \neq b}{b.e \vdash a.p = \emptyset}$ |

( The system $\underline{S}_{ff}$ )

We shall write $e \vdash_F p = q$ if $e \vdash p = q$ is provable in $\underline{S}_{ff}$.

<u>Theorem 4.1-2:</u> (Soundness of $\underline{S}_{ff}$)
For all $e \varepsilon E_f$ and $p,q \varepsilon P_f$:

$$e \vdash_F p = q \quad \text{implies} \quad e \models p = q$$

<u>Proof:</u> We must show that each axiom of $\underline{S}_{ff}$ is valid and that each rule of $\underline{S}_{ff}$ preserves validity.

For S1-S4 use soundness of the system in theorem 4.1-1 and the fact that $\sim_U = \sim$. For E1-E3 appeal to proposition 2.2-5. All the rules of <u>CONG</u> are of the general form:

$$\frac{wie(C,e) \vdash p = q}{e \vdash C[p] = C[q]}$$

Hence preservation of validity follows from the general parameterized congruence law, theorem 3.4-5 and theorem 3.4-4. For <u>CONS</u> appeal to theorem 2.4-10. Obviously $\Phi$ is a minimal environment. Hence <u>NIL</u> is sound. For <u>COMB</u> use lemma 2.4-4. Validity of <u>ANNIHIL</u> is immediate. □

<u>Example 4.1-3:</u> Recall examples 2.4-22 and 2.4-32 where $e=a.b.\Phi + a.c.\Phi$, $p = a.b.\Phi + a.c.\Phi$ and $q = a.b.\Phi + a.c.\Phi + a.(b.\Phi + c.\Phi)$. We want to establish $e \vdash_F p = q$:

$$\dfrac{\dfrac{\vdots}{a.b.\Phi \vdash p = q} \qquad \dfrac{\dfrac{\dfrac{\dfrac{\dfrac{c \neq b}{c.\Phi \vdash b.\Phi = \Phi}\;\text{ANNIHIL}}{c.\Phi \vdash c.\Phi = b.\Phi + c.\Phi}\;\text{C3,S4}}{a.c.\Phi \vdash a.c.\Phi = a.(b.\Phi + c.\Phi)}\;\text{C2}}{a.c.\Phi \vdash p + a.c.\Phi = p + a.(b.\Phi + c.\Phi)}\;\text{C3}}{a.c.\Phi \vdash p = q}\;\text{S3,E3,CONS}}{e \vdash p = q}\;\text{COMB}$$

□

As it stands the proof systeym $\underline{S}_{ff}$ is actually only <u>relative</u> complete wrt. true assertions of the form, $e \leqslant f$, where e and f are finite environments. However, these assertions are easily axiomatized as indicated below:

Theorem 4.1-4: The simulation ordering, $\lesssim$, over $\mathbb{E}_f$ is exactly the substitutive preorder induced by the following axioms:

$$
\begin{array}{ll}
(A1) & e \lesssim U \\
(A2) & e + (f + g) \simeq (e + f) + g \\
(A3) & e + f \simeq f + e \\
(A4) & e + e \simeq e \\
(A5) & e + \mathbb{0} \simeq e \\
(A6) & e \lesssim e + f
\end{array}
$$

($t_1 \simeq t_2$ is an abbreviation for the two rules $t_1 \lesssim t_2$ and $t_2 \lesssim t_1$).

Proof: Validity of the axioms (A2)-(A5) follows from theorem 4.1-1 and the fact that $\sim \subseteq \lesssim$. Validity of (A1) and (A6) is immediate. By proposition 2.1-9 we know that $\lesssim$ is a preorder. Lemma 2.4-3 ensures that $\lesssim$ is substitutive.

For completeness assume $e \lesssim f$. If $f=U$ then $\vdash e \lesssim f$ follows from (A1). If $e=U$ then also $f=U$ (otherwise $e \not\lesssim f$) and hence again $\vdash e \lesssim f$ by (A1). If neither e nor f is U we can find sumforms $e^+$ and $f^+$ such that:

$$\vdash e \simeq e^+ \qquad \text{and} \qquad \vdash f \simeq f^+$$

where $e^+ = \sum_n a_i.e_i$ and $f^+ = \sum_m b_j.f_j$. We prove by induction on the size of $e^+$ that $e^+ \lesssim f^+$ implies $\vdash e^+ \lesssim f^+$.

$|e^+| = 0$: Then $e^+ = \mathbb{0}$ and $\vdash \mathbb{0} \lesssim f^+$ follows from (A6) and (A5).

$|e^+| > 0$: Consider the first term of $e^+$, $a_1.e_1$. Then $a_1.e_1 \lesssim f^+$. Thus for some $f'$, $f^+ \xrightarrow{a_1} f'$ and $e_1 \lesssim f'$. But $f'$ must be $f_j$ for some $j<m$, with $b_j = a_1$, and by induction $\vdash e_1 \lesssim f_j$. By substitutiveness of $\lesssim$ then $\vdash a_1.e_1 \lesssim a_1.f_j$, and hence using (A6) and (A4) $\vdash a_1.e_1 \lesssim a_1.f_j + f^+ \simeq f^+$. Thus we can obtain $\vdash a_i.e_i \lesssim f^+$ for all $i<n$ and it follows therefore that $\vdash e^+ \lesssim f^+$. $\square$

Now add the axioms and inferencerules for the above axiomatization of $\leqslant$ over $\mathbb{E}_f$ to $\underline{S}_{ff}$ and we obtain a genuine complete proof system, $\underline{S}_{ff}^+$.

Theorem 4.1-5: (Completeness of $\underline{S}_{ff}^+$)
For all $e \varepsilon E_f$ and $p,q \varepsilon P_f$:

$$e \models p = q \quad \text{implies} \quad e \vdash_F^+ p = q$$

where $\vdash_F^+$ means provability in the extended system, $\underline{S}_{ff}^+$.

Proof: For e=U, $e \vdash_F^+ p = q$ follows immediately since $\underline{S}_{ff}^+$ is an extension of the system in theorem 4.1-1. Thus if $\vdash p = q$ follows from (A1)-(A4) of theorem 4.1-1 together with congruence properties then $U \vdash_F^+ p = q$. Otherwise $(e \neq U)$, e can be brought on sumform, i.e.:

$$\vdash e \simeq e^+$$

where $e^+ = \sum_p c_k \cdot e_k$. Using (S1)-(S4), EQUIV and CONG with $e = U$ we can (provably) transform p and q to sumforms, $p^+$ and $q^+$, i.e.:

$$U \vdash_F^+ p = p^+ \qquad \text{and} \qquad U \vdash_F^+ q = q^+$$

with $p^+ = \sum_n a_i \cdot p_i$ and $q^+ = \sum_m b_j \cdot q_j$. By the transitivity rule of EQUIV and CONS clearly:

$$e \vdash_F^+ p = q \quad \text{iff} \quad e^+ \vdash_F^+ p^+ = q^+$$

So if we can establish $e^+ \vdash_F^+ p^+ = q^+$ we are done. The proof of this is done by induction on the size of $e^+$.

$|e^+|=0$: Then $e^+ = \Phi$ and $\Phi \vdash_F^+ p^+ = q^+$ is immediate from NIL.

$|e^+|=1$: Then $e^+ = c_1 \cdot e_1$ for some $c_1, e_1$. If $a_1 \neq c_1$ then by ANNIHIL $c_1 \cdot e_1 \vdash_F^+ a_1 \cdot p_1 = \Phi$ and hence $c_1 \cdot e_1 \vdash_F^+ p^+ = a_2 \cdot p_2 + \dots + a_{n-1} \cdot p_{n-1}$ by EQUIV. Repeating this procedure we can cancel out all terms of $p^+$ not prefixed with $c_1$. Thus we get:

$$c_1 \cdot e_1 \vdash^+_F p^+ = p^{++}$$

and similarly for $q^+$:

$$c_1 \cdot e_1 \vdash^+_F q^+ = q^{++}$$

where $p^{++}$ is of the form $\sum_{n'} c_1 \cdot p_i'$ and similarly $q^{++}$ is of the form $\sum_{m'} c_1 \cdot q_j'$.
By soundness $c_1 \cdot e_1 \models p^{++} = q^{++}$. If $p^{++} = \emptyset$ then also $q^{++} = \emptyset$ and so from reflexivity we have $c_1 \cdot e_1 \vdash^+_F p^{++} = q^{++}$. Otherwise let $c_1 \cdot p_1'$ be a term of $p^{++}$. Then - by the very definition of parameterized bisimulation - $q^{++} \xrightarrow{c_1} q'$ for some $q'$ with $e_1 \models p_1' = q_1'$. But $q'$ must be $q_j'$ for some $j<m'$. By induction hypothesis then:

$$e_1 \vdash^+_F p_1' = q_j'$$

(C2) $$c_1 \cdot e_1 \vdash^+_F c_1 \cdot p_1' = c_1 \cdot q_j'$$

(C3) $$c_1 \cdot e_1 \vdash^+_F q^{++} + c_1 \cdot p_1' = q^{++} + c_1 \cdot q_j'$$

(SUM) $$c_1 \cdot e_1 \vdash^+_F q^{++} + c_1 \cdot p_1' = q^{++}$$

By repeating this procedure for all $i<n'$ we get $c_1 \cdot e_1 \vdash^+_F q^{++} + p^{++} = q^{++}$ and by symmetry $c_1 \cdot e_1 \vdash^+_F p^{++} = q^{++}$ and hence $c_1 \cdot e_1 \vdash^+_F p^+ = q^+$.

<u>$|e^+|>1$:</u> Split $e^+$ up into two smaller subterms and apply the induction hypothesis to them. Use <u>COMB</u> to get the result for $e^+$. □

A proof system, $\underline{S}_{fl}$, for parameterized equivalence for <u>finite processes</u> and <u>language environments</u> is given below. The system is sound and relative complete wrt. true assertions of the form $M \subseteq L$, where $M$ and $L$ are languages over Act. $\underline{S}_{fl}$ is very similar to $\underline{S}_{ff}$ and the completeness proof (which we ommit) is analogous.

Note: there is obviously no rule corresponding to <u>COMB</u> of $\underline{S}_{ff}$ in $\underline{S}_{fl}$. The two rules, <u>NIL</u> and <u>ANNIHIL</u>, of $\underline{S}_{ff}$, are replaced by a single rule, <u>ANNIHIL</u>, in $\underline{S}_{fl}$.

| | | |
|---|---|---|
| <u>SUM</u> | S1. | $Act^* \vdash p + (q + r) = (p + q) + r$ |
| | S2. | $Act^* \vdash p + q = q + p$ |
| | S3. | $Act^* \vdash p + p = p$ |
| | S4. | $Act^* \vdash p + \Phi = p$ |
| <u>EQUIV</u> | E1. | $L \vdash p = p$ |
| | E2. | $\dfrac{L \vdash p = q}{L \vdash q = p}$ |
| | E3. | $\dfrac{L \vdash p = q \qquad L \vdash q = r}{L \vdash p = r}$ |
| <u>CONG</u> | C1. | $\dfrac{\partial L/\partial a \vdash p = q}{L \vdash a.p = a.q}$ |
| | C2. | $\dfrac{L \vdash p = q}{L \vdash r + p = r + q}$ |
| <u>CONS</u> | | $\dfrac{M^p \subseteq L^p \qquad L \vdash p = q}{M \vdash p = q}$ |
| <u>ANNIHIL</u> | | $\dfrac{\partial L/\partial a = \emptyset}{L \vdash a.p = \Phi}$ |

( The system $\underline{S}_{fl}$ )

## 4.2 A COMPLETE PROOF SYSTEM FOR REGULAR BEHAVIOURS

Let us define the two transition systems of regular processes and environments, $\mathbb{P}_r$ and $\mathbb{E}_r$ : $\mathbb{P}_r = (P_r, Act, \rightarrow)$ where $P_r$ consists of the following terms:

$$p ::= \mathbb{O} \mid x \mid a.p \mid p+q \mid \mu x.p$$

where $x \in Var$ and $a \in Act$. The operational semantics ($\rightarrow$) of $\mathbb{P}_r$ is the standard one (see section 3.2). However, in contrast to the notion of recursion introduced for CCS in section 3.2, we shall not insist on the guardedness restriction here.

The system of regular environments, $\mathbb{E}_r$, is simply $\mathbb{P}_r$ extended with a universal environment. I.e. $\mathbb{E}_r = (E_r, Act, \Rightarrow)$ where $E_r = P_r \cup \{U\}$ and $\Rightarrow = \rightarrow \cup \{(U,a,U) \mid a \in Act\}$. Let $P_r^c$ resp. $E_r^c$ be the set of closed process expressions resp. closed environment expressions and let $\mathbb{P}_r^c$ and $\mathbb{E}_r^c$ be the corresponding restricted transition systems. We want to axiomatize the parameterized equivalence problem for $\mathbb{P}_r^c$ and $\mathbb{E}_r^c$. However, it seems necessary to widen the axiomatization to allow for general process expressions over $\mathbb{P}_r$. For this reason we refine the notion of parameterized bisimulation (similar to the refinement of bisimulation in /Mil82/) in order to take account of the possibility of free variables in a process expression. Let UG(p) be the set of unguarded variables in the process expression p. We then define:

Definition 4.2-1: Let R be an $E_r^c$-indexed family of binary relations over $P_r$. Then R is a refined parameterized bisimulation if R is a parameterized bisimulation and whenever $pR_eq$ then $UG(p) = UG(q)$. We write $e \models p = q$ if there exists a refined parameterized bisimulation, R, with $pR_eq$. □

Note, that for closed process expressions the notion of refined parameterized bisimulation coincides with that of parameterized bisimulation. It is easily shown that propositions 2.2-2 - 2.2-6, 2.2-9 extend to refined parameterized bisimulation in the obvious ways. We shall throughout the remainder of this chapter use the term parameterized bisimulation for refined parameterized bisimulation.

### 4.2.1 Properties of $\mathbb{P}_r$ and $\mathbb{E}_r$.

Before presenting any proof systems let us state some fundamental properties of the derivation relation $\longrightarrow$ in $\mathbb{P}_r$. Since $\mathbb{E}_r$ is a simple extension of $\mathbb{P}_r$ it is easily shown that all these properties hold for the consumption relation, $\Longrightarrow$, of $\mathbb{E}_r$ as well.

Let $p\{\bar{r}/\bar{x}\}$, where $\bar{r} = (r_1,\ldots,r_m)$ and $\bar{x} = (x_1,\ldots,x_m)$, stand for the <u>simultaneous substitution</u> of expressions $\bar{r}$ for variables $\bar{x}$ in the expression $p$. Let $p \equiv q$ if $p$ and $q$ are expressions equal up to renaming of bound variables. Then the following is easily shown to hold:

(P1) Whenever $p\{\bar{q}/\bar{x}\} \xrightarrow{a} r$ then either for some $p'$:

$$p \xrightarrow{a} p' \text{ and } r \equiv p'\{\bar{q}/\bar{x}\}$$

or for some $i \leq m$:

$$x_i \varepsilon UG(p) \text{ and } q_i \xrightarrow{a} r$$

(P2) Whenever $x_i \varepsilon UG(p)$ and $q_i \xrightarrow{a} r$ then:

$$p\{\bar{q}/\bar{x}\} \xrightarrow{a} r$$

(P3) Whenever $p \xrightarrow{a} p'$ then for some $r$:

$$p\{\bar{q}/\bar{x}\} \xrightarrow{a} r \text{ and } r \equiv p'\{\bar{q}/\bar{x}\}$$

If all $q_i$'s are closed expressions, then $\equiv$ can be replaced by simple syntactic equality in (P1) and (P3), since no renaming of bound variables of p in $p\{\bar{q}/\bar{x}\}$ is needed in this case.

From the operational behaviour of $\mu x.p$ it now follows that:

(P4) Whenever $\mu x.p \xrightarrow{a} r$ then for some $p'$:
$$p \xrightarrow{a} p' \quad \text{and} \quad r \equiv p'\{\mu x.p/x\}$$

(P5) Whenever $p \xrightarrow{a} p'$ then for some r:
$$\mu x.p \xrightarrow{a} r \quad \text{with} \quad r \equiv p'\{\mu x.p/x\}$$

Again we can replace $\equiv$ with simple equality if $\mu x.p$ is a closed expression.

As a slightly stronger result than (P4) and (P5) it can be shown that there is a 1-1 correspondence between derivatives of p and derivatives of $\mu x.p$. From this it follows by structural induction that $\mathbb{P}_r$ is image-finite and for all processes p of $\mathbb{P}_r$ the set $\{p' \mid \exists s \varepsilon Act^*.\ p \xrightarrow{s} p'\}$ is finite.

The properties (P1)-(P5) only determines derivatives of processes from $\mathbb{P}_r$ up to "$\equiv$". For this reason the following concept of parameterized bisimulation up to "$\equiv$" is often useful: (see /Mil83/ for an analogous notion of bisimulation up to "$\sim$"). An $E_r^c$-indexed family of binary relations over $P_r$, R, is a parameterized bisimulation up to "$\equiv$" if and only if $\equiv' \circ R \circ \equiv'$ is a parameterized bisimulation, where $(\equiv')_e = \equiv$ for all $e \varepsilon E_r^c$. If R is a parameterized bisimulation up to "$\equiv$" and $pR_eq$ then by the reflexivity of $\equiv$ it follows that $p \sim_e q$. A necessary and sufficient condition for R to be a parameterized bisimulation up to "$\equiv$" is that $R \subseteq \mathbb{B}(\equiv' \circ R \circ \equiv')$ ( a condition we shall be using repeatedly in the following).

Finally, we shall need a few basic properties of substitution:

(P6) If no $x_i$ is free in p then $p\{\bar{r}/\bar{x}\} \equiv p$.

(P7) If $\bar{x}$ and $\bar{y}$ are disjoint then:

$$p\{\bar{q}/\bar{x}\}\{\bar{r}/\bar{y}\} \equiv p\{\bar{q}\{\bar{r}/\bar{y}\}/\bar{x}, \bar{r}/\bar{y}\}$$

## 4.2.2 The proof system $\underline{S_M}$.

Let us start by recalling the complete proof system, here called $\underline{S_M}$, for the <u>un</u>parameterized equivalence problem over $\mathbb{P}_r$ given in /Mil82/.

<u>EQUIV</u>

E1. $p = p$

E2. $$\frac{p = q}{q = p}$$

E3. $$\frac{p = q \qquad q = r}{p = r}$$

<u>CONG</u>

C1. $$\frac{p = q \qquad \bar{r} = \bar{r}'}{p\{\bar{r}/\bar{x}\} = q\{\bar{r}'/\bar{x}\}}$$

C2. $$\frac{p = q}{\mu x.p = \mu x.q}$$

<u>SUM</u>

S1. $p + q = q + p$

S2. $p + (q + r) = (p + q) + r$

S3. $p + p = p$

S4. $p + \mathbb{0} = p$

<u>REC</u>

R1. $\mu x.p = \mu y.p\{y/x\}$ ; y not free in $\mu x.p$

R2. $\mu x.p = p\{\mu x.p/x\}$

R3. $\mu x.(p + x) = \mu x.p$

R4. $$\frac{p = q\{p/x\}}{p = \mu x.q} \quad ; \quad x \notin UG(q)$$

( The system $\underline{S_M}$ )

We shall write $\vdash_M p = q$ if and only if $p = q$ is provable in $\underline{S}_M$. The completeness proof of $\underline{S}_M$ is based on the following two important theorems (see /Mil82/):

Theorem 4.2-2: (Unique Solution of Equations)
Let $\bar{x} = (x_1,\ldots,x_m)$ and $\bar{y} = (y_1,\ldots,y_n)$ be distinct variables, and $\bar{p} = (p_1,\ldots,p_m)$ expressions with free variables in $(\bar{x},\bar{y})$ in which each $x_i$ is guarded. Then there exist expressions $\bar{r} = (r_1,\ldots,r_m)$ with free variables in $\bar{y}$ such that:

$$\vdash_M r_i = p_i\{\bar{r}/\bar{x}\} \qquad (i \leq m)$$

Moreover, if the above also holds for expressions $\bar{r}' = (r_1',\ldots,r_m')$ with free variables in $\bar{y}$, then:

$$\vdash_M r_i' = r_i \qquad (i \leq m)$$

□

Theorem 4.2-3: (Equational Characterization in $\underline{S}_M$)
For any expression p, with free variables in $\bar{y}$, there exist expressions $p_1,\ldots,p_h$ $(h \geq 1)$ with free variables in $\bar{y}$, satisfying h equations:

$$\vdash_M p_i = \sum_{j=1}^{m(i)} a_{ij} \cdot p_{f(i,j)} + \sum_{j=1}^{n(i)} y_{g(i,j)} \qquad (i \leq h)$$

and moreover:

$$\vdash_M p = p_1$$

□

The complete proof system $\underline{S}_M$ is closely analogous to that of Salomaa /Sal66/ for equality of regular sets of words. A close comparison of $\underline{S}_M$ with Salomaa's system is made in /Mil82/.

### 4.2.3 Wie and its properties.

We are searching for an extension of Milner's system, $\underline{S}_M$, which will be sound and complete wrt. parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_r^c$. It turns out that in the final extended system most of the rules of $\underline{S}_M$ are used directly with only minor changes. The only two rules of $\underline{S}_M$ which requires more careful alterations are the congruence rule, C1, and the recursion rule R4.

We notice that in $p\{\bar{r}/\bar{x}\}$, p acts as an m-ary ($\bar{x} = (x_1,\ldots,x_m)$) context with $r_1,\ldots,r_m$ as inner processes. In light of the previous chapters results it seems therefore natural to replace C1 with a parameterized congruence law of the form:

$$\frac{e \vdash p = p' \qquad \overline{\mathrm{wie}(p,e)} \vdash \bar{r} = \bar{r}'}{e \vdash p\{\bar{r}/\bar{x}\} = p'\{\bar{r}'/\bar{x}\}}$$

where $\overline{\mathrm{wie}(p,e)}$ is the weakest (wrt. $\sqsubseteq$) m-tuple of environments which will make the above rule sound (if we make the additional requirement that p and p' must interact identically with $\bar{r}$ and $\bar{r}'$). Since our results from chapter 3 only applies to unary contexts a special treatment is needed.

The recursion rule, R4, gives conditions which ensures that a recursive equation has a unique solution. In the extended system, R4, will be replaced by a more general rule ensuring unique solutions to recursive equations <u>in an environment</u>. This new rule will also be using the wie-construct.

Now for $x \varepsilon Var$, $p \varepsilon P_r$ and $e \varepsilon E_r^c$ we define $\underline{\mathrm{wie}_x(p,e)} \varepsilon E_r^c$ as follows:

$$\mathrm{wie}_x(p,e) = \sum_{f \in I_x(p,e)} f$$

where $I_x(p,e) = \{f \mid \exists s \in Act^*.\ e \xRightarrow{s} f \ \& \ (\exists p'.\ p \xrightarrow{s} p' \ \& \ x \in UG(p'))\}$

Note, that since e has only finitely many derivatives, $I_x(p,e)$ is finite. Thus $\mathrm{wie}_x(p,e)$ is indeed expressible in $E_r^c$. The intuition behind the set $I_x(p,e)$ is loosely that $f \in I_x(p,e)$ if and only if when executing $p\{q/x\}$ in e it is possible to reach a situation where q may be executed in f. With this definition of $\mathrm{wie}_x(p,e)$ it is easily shown that the following algebraic properties hold:

Proposition 4.2-4:

(i) $\mathrm{wie}_x(\Phi,e) \simeq \Phi$

(ii) $\mathrm{wie}_x(y,e) \simeq \begin{cases} \Phi & ; \text{ if } x \neq y \\ e & ; \text{ otherwise} \end{cases}$

(iii) $\mathrm{wie}_x(p,\Phi) \simeq \Phi$

(iv) $\mathrm{wie}_x(p,U) \simeq \begin{cases} U & ; \text{ if } x \text{ is free in } p \\ \Phi & ; \text{ otherwise} \end{cases}$

(v) $\mathrm{wie}_x(p+q,e) \simeq \mathrm{wie}_x(p,e) + \mathrm{wie}_x(q,e)$

(vi) $\mathrm{wie}_x(p,e+f) \simeq \mathrm{wie}_x(p,e) + \mathrm{wie}_x(p,f)$

(vii) $\mathrm{wie}_x(a.p\ ,\ b.e) \simeq \begin{cases} \mathrm{wie}_x(p,e) & ; \text{ if } a=b \\ \Phi & ; \text{ otherwise} \end{cases}$

(viii) $\mathrm{wie}_x(\mu y.p\ ,\ e) \sim \mathrm{wie}_x(p\{\mu y.p\,/\,y\}\ ,\ e)$

(ix) $\mathrm{wie}_x(p\ ,\mu y.e) \sim \mathrm{wie}_x(p\ ,\ e\{\mu y.e\,/\,y\})$ □

Proposition 4.2-5: $\mathrm{wie}_x(p,e)$ is monotonic in e with respect to $\leqslant$. □

Lemma 4.2-6: (Derivations Lemma)

If $p \xrightarrow{a} p'$ and $e \overset{a}{\Rightarrow} e'$ then $wie_x(p',e') \leq wie_x(p,e)$.

Proof: Follows from $I_x(p',e') \subseteq I_x(p,e)$. $\square$

Lemma 4.2-7: (Substitution Lemma)

$$wie_x(p\{\bar{r}/\bar{x}\}, e) \simeq \sum_{i \leq m} wie_x(r_i, wie_{x_i}(p,e)) \quad [+ wie_x(p,e)]_{x \notin \bar{x}}$$

Proof: Show, using (P1)-(P3) and $wie_x(p,e) \simeq wie_x(q,e)$ if $p \equiv q$, that:

$$I_x(p\{\bar{r}/\bar{x}\}, e) = \bigcup_{i \leq m} I_x(r_i, wie_{x_i}(p,e)) \quad [\cup I_x(p,e)]_{x \notin \bar{x}}$$

$\square$

Lemma 4.2-8: If $x \neq y$, $wie_y(p,f) \leq e$ and $wie_y(q,e) \leq e$ then:

$$wie_x(p\{\mu y.q/y\}, f) \leq wie_x(p,f) + wie_x(q,e)$$

Proof: Let $g \in I_x(p\{\mu y.q/y\}, f)$. I.e. for some $s \in Act^*$, some $g$ and $r$:

$$f \overset{s}{\Rightarrow} g \quad \text{and} \quad p\{\mu y.q/y\} \xrightarrow{s} r \quad \text{with } x \in UG(r)$$

We prove by induction on $|s|$ that $g \leq wie_x(p,f) + wie_x(q,e)$. By the least upper bound property of summation the lemma will then follow.

Basis, $s=\varepsilon$: Then $g$ is $f$, $r = p\{\mu y.q/y\}$ and $x \in UG(r)$. Now, $x \in UG(r)$ iff either $x \in UG(p)$ or $y \in UG(p)$ and $x \in UG(q)$. Thus, also $x \in UG(p\{q/y\})$. Obviously $p\{q/y\} \xrightarrow{\varepsilon} p\{q/y\}$ so we have:

$$f \leq wie_x(p\{q/y\}, f)$$

$$\text{(4.2-7)} \quad \simeq wie_x(p,f) + wie_x(q, wie_y(p,f))$$

$$(4.2\text{-}5) \quad \leq \mathrm{wie}_x(p,f) + \mathrm{wie}_x(q,e)$$

Step, $s=a\,s'$: Then for some $h$ and $r'$: $f \overset{a}{\Rightarrow} h \overset{s'}{\Rightarrow} g$ and $p\{\mu y.q/y\} \overset{a}{\rightarrow} r' \overset{s'}{\rightarrow} r$. By (P1)-(P5) either:

(A) For some $r''$, $p \overset{a}{\rightarrow} r''$ and $r' \equiv r''\{\mu y.q/y\}$

or (B) $y \in UG(p)$ and for some $r''$:
$q \overset{a}{\rightarrow} r''$ and $r' \equiv r''\{\mu y.q/y\}$

We will show that in both cases $\mathrm{wie}_y(r'',h) \leq e$ (and of course $\mathrm{wie}_y(q,e) \leq e$) in order to invoce the induction hypothesis. Clearly $g \in I_x(r',h)$. So:

$$\begin{aligned}
g &\leq \mathrm{wie}_x(r',h) \\
&\simeq \mathrm{wie}_x(r''\{\mu y.q/y\}, h) \\
(\mathrm{IH}) \quad &\leq \mathrm{wie}_x(r'',h) + \mathrm{wie}_x(q,e) \\
(4.2\text{-}6) \quad &\leq \begin{cases} \mathrm{wie}_x(p,f) + \mathrm{wie}_x(q,e) & ;\ \text{in (A)} \\ \mathrm{wie}_x(q,f) + \mathrm{wie}_x(q,e) & ;\ \text{in (B)} \end{cases}
\end{aligned}$$

But in (B) $f \leq \mathrm{wie}_y(p,f) \leq e$ so by lemma 4.2-5:

$$\leq \mathrm{wie}_x(p,f) + \mathrm{wie}_x(q,e)$$

in both (A) and (B). It remains to verify that $\mathrm{wie}_y(r'',h) \leq e$ in both (A) and (B). In (A) we have from the Derivation Lemma 4.2-6 that:

$$\mathrm{wie}_y(r'',h) \leq \mathrm{wie}_y(p,f) \leq e$$

In (B) we have $f \leq e$, since $y \in UG(p)$ and $\mathrm{wie}_y(p,f) \leq e$. Thus by Derivation Lemma 4.2-6 and monotonicity 4.2-5:

$$\begin{aligned}
\mathrm{wie}_y(r'',h) &\leq \mathrm{wie}_y(q,f) \\
&\leq \mathrm{wie}_y(q,e) \leq e
\end{aligned}$$

□

Corollary 4.2-9: (Recursion Lemma)
If $x \neq y$ and $\mathrm{wie}_y(q,e) \leq e$ then $\mathrm{wie}_x(\mu y.q\,,\,e) \simeq \mathrm{wie}_x(q,e)$

Proof: Using proposition 4.2-4 (viii) and the Substitution Lemma 4.2-7 we have:

$$\begin{aligned}
&\mathrm{wie}_x(\mu y.q\ ,\ e)\\
\simeq\ &\mathrm{wie}_x(q\{\mu y.q/y\}\ ,\ e)\\
\simeq\ &\mathrm{wie}_x(q,e)\ +\ \mathrm{wie}_x(\mu y.q\ ,\ \mathrm{wie}_y(q,e)\ )\\
\geqslant\ &\mathrm{wie}_x(q,e)
\end{aligned}$$

To prove $\mathrm{wie}_x(\mu y.q\ ,\ e) \leqslant \mathrm{wie}_x(q,e)$ we apply the previous lemma 4.2-8 with p=y and f=e. Obviously then the condition $\mathrm{wie}_y(p,f) \leqslant e$ is fulfilled so we can conclude:

$$\begin{aligned}
&\mathrm{wie}_x(\mu y.q\ ,\ e)\\
\simeq\ &\mathrm{wie}_x(y\{\mu y.q/y\}\ ,\ e)\\
\leqslant\ &\mathrm{wie}_x(y,e)\ +\ \mathrm{wie}_x(q,e)\\
\simeq\ &\mathrm{wie}_x(q,e)
\end{aligned}$$

□

## 4.2.4 The proof system $\underline{S}_{rr}$ and its soundness.

We can now present the proof system $\underline{S}_{rr}$ for parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_r^c$ (see next page). As we predicted previously most of the rules of $\underline{S}_{rr}$ are carried over from $\underline{S}_M$ (or even $\underline{S}_{ff}$), with a few minor changes. Only the rules C1 and R4 seem to need further justifications. In C1 $\mathrm{wie}_{\bar{x}}(p,e) \vdash \bar{r} = \bar{r}'$ is an abbreviation for the m assertions $\mathrm{wie}_{x_i}(p,e) \vdash r_i = r_i'$ (i≤m).

**EQUIV**

E1. $$e \vdash p = p$$

E2. $$\frac{e \vdash p = q}{e \vdash q = p}$$

E3. $$\frac{e \vdash p = q \qquad e \vdash q = r}{e \vdash p = r}$$

**CONG**

C1. $$\frac{e \vdash p = p' \qquad wie_{\bar{x}}(p,e) \vdash \bar{r} = \bar{r}'}{e \vdash p\{\bar{r}/\bar{x}\} = p'\{\bar{r}'/\bar{x}\}}$$

C2. $$\frac{U \vdash p = p'}{U \vdash \mu x.p = \mu x.p'}$$

**CONS**

$$\frac{e \leqslant f \qquad f \vdash p = q}{e \vdash p = q}$$

**NIL**

$$\frac{UG(p) = UG(q)}{\emptyset \vdash p = q}$$

**COMB**

$$\frac{e \vdash p = q \qquad f \vdash p = q}{e + f \vdash p = q}$$

**ANNIHIL**

$$\frac{a \neq b}{b.e \vdash a.p = \emptyset}$$

**SUM**

S1. $$U \vdash p + q = q + p$$

S2. $$U \vdash p + (q + r) = (p + q) + r$$

S3. $$U \vdash p + p = p$$

S4. $$U \vdash p + \emptyset = p$$

**REC**

R1. $$U \vdash \mu x.p = \mu y.p\{y/x\}$$ ; y not free in p.

R2. $$U \vdash \mu x.p = p\{\mu x.p / x\}$$

R3. $$U \vdash \mu x(p + x) = \mu x.p$$

R4. $$\frac{e \vdash p = q\{p/x\} \qquad wie_{x}(q,e) \leqslant e}{e \vdash p = \mu x.q} \quad ; \ x \notin UG(q)$$

( The system $\underline{S}_{rr}$ )

We shall write $e \vdash_R p = q$ iff $e \vdash q = q$ is provable in $\underline{S}_{rr}$ <u>using all true assertions of the form $e \leqslant f$ as axioms</u>. The following theorem proves the validity of C1.

<u>Theorem 4.2-10:</u> (Substitution Theorem)
Let $\bar{x} = (x_1,\ldots,x_m)$, $\bar{r} = (r_1,\ldots,r_m)$ and $\bar{r}' = (r_1',\ldots,r_m')$. If $e \models p = p'$ and $wie_{x_i}(p,e) \models r_i = r_i'$ for $i \leqslant m$ then:

$$e \models p\{\bar{r}/\bar{x}\} = p'\{\bar{r}'/\bar{x}\}$$

<u>Proof:</u> It suffice to prove that the $E_r^c$-indexed family, R, with:

$$R_e = \{(p\{\bar{r}/\bar{x}\}, p'\{\bar{r}'/\bar{x}\}) \mid e \models p = p' \ \& \ \forall i \leqslant m.\ wie_{x_i}(p,e) \models r_i = r_i' \}$$

is a parameterized bisimulation up to "$\equiv$".
Let $(p\{\bar{r}/\bar{x}\}, p'\{\bar{r}'/\bar{x}\}) \varepsilon R_e$. Then $UG(p)=UG(p')$ and for all $i \leqslant m$, $UG(r_i)=UG(r_i')$. Hence, $UG(p\{\bar{r}/\bar{x}\})=UG(p'\{\bar{r}'/\bar{x}\})$. Since $p \equiv p'$ implies $UG(p)=UG(p')$ it follows that whenever $(p,p') \varepsilon \equiv \circ R_e \circ \equiv$ then $UG(p)=UG(p')$. Now, let $e \overset{a}{\Rightarrow} f$ and $p\{\bar{r}/\bar{x}\} \xrightarrow{a} q$. By (P1) either:

(A) for some $p_+$, $p \xrightarrow{a} p_+$ and $q \equiv p_+\{\bar{r}/\bar{x}\}$

or (B) for some $i \leqslant m$, $x_i \varepsilon UG(p)$ and $r_i \xrightarrow{a} q$

We must find a matching move in $\equiv \circ R_f \circ \equiv$ for $p'\{\bar{r}'/\bar{x}\}$ in both cases.

<u>(A):</u> Since $e \models p = p'$, $p' \xrightarrow{a} p_+'$ for some $p_+'$ with $f \models p_+ = p_+'$. By (P3) then for some $q'$, $p'\{\bar{r}'/\bar{x}\} \xrightarrow{a} q'$ with $q' \equiv p_+'\{\bar{r}'/\bar{x}\}$. In order for $(q,q') \varepsilon \equiv \circ R_f \circ \equiv$ it suffice to prove $(p_+\{\bar{r}/\bar{x}\}, p_+'\{\bar{r}'/\bar{x}\}) \varepsilon R_f$. However, this will follow if $wie_{x_i}(p_+,f) \models r_i = r_i'$ for all $i \leqslant m$. But by the Derivation Lemma 4.2-6, $wie_{x_i}(p_+,f) \leqslant wie_{x_i}(p,e)$ and by assumptions $wie_{x_i}(p,e) \models r_i = r_i'$ for all $i \leqslant m$. Thus $wie_{x_i}(p_+,f) \models r_i = r_i'$ follows.

<u>(B):</u> Now $x_i \varepsilon UG(p)$ implies $e \varepsilon I_{x_i}(p,e)$ and thus $e \leqslant wie_{x_i}(p,e)$.

Thus we have $e \models r_i = r_i'$. Hence, for some $q'$, $r_i' \xrightarrow{a} q'$ with $f \models q = q'$ or equivalently $f \models x_i\{\bar{q}/\bar{x}\} = x_i\{\bar{q}'/\bar{x}\}$ where $\bar{q} = (q,\dots,q)$ and $\bar{q}' = (q',\dots,q')$. Since

$$wie_{x_j}(x_i,f) = \begin{cases} \Phi & ; \text{ if } i \neq j \\ f & ; \text{ otherwise} \end{cases}$$

we have for all $j \leq m$, $wie_{x_j}(x_i,f) \models q = q'$ and hence $(q,q') = (x_i\{\bar{q}/\bar{x}\}, x_i\{\bar{q}'/\bar{x}\}) \varepsilon R_f$. Since $e \models p = p'$ also $x_i \varepsilon UG(p)$. By (P2) therefore $p'\{\bar{r}'/\bar{x}\} \xrightarrow{a} q'$. The above shows that this is the matching derivation. □

The rule R4 claims that provided $wie_x(q,e) \leq e$, then the <u>parameterized</u> recursive equation $e \models p = q\{p/x\}$ has exactly one solution, $\mu x.q$. The condition $wie_x(q,e) \leq e$ express an <u>invariant</u> property of e wrt. q similar to the wellknown loop-invariant for sequential while-programs. It is easily shown that without this condition R4 will become invalid:

<u>Example 4.2-11:</u> Let $e = a.b.\Phi$, $q = a.x$, $p_0 = b.\Phi + a.b.\Phi$ and $p_1 = a.a.\Phi$. Then it is easily shown that:

$$e \models p_i = q\{p_i/x\} \qquad i=0,1$$

but

$$e \not\models p_0 = p_1$$

□

From $e \models p = q\{p/x\}$ and $wie_x(q,e) \leq e$ it follows by repeated use of the Substitution Theorem 4.2-10 and <u>CONS</u> that for all $n \varepsilon \omega$:

$$e \models p = q^n\{p/x\}$$

where $q^1 = q$ and $q^{n+1} = q\{q^n/x\}$. Since x is guarded in q we expect $q^n\{p/x\}$ to converge to $\mu x.q$ and hence that $e \models p = \mu x.q$. This is formally verified in the following:

Theorem 4.2-12: (Invariant Theorem)
If $x \notin UG(q)$, $e \models p = q\{p/x\}$ and $wie_x(q,e) \leqslant e$ then $e \models p = \mu x.q$ .

Proof: From soundness of $R_2$ and CONS it is enough to show that if $e \models p_o = q\{p_o/x\}$ and $e \models p_1 = q\{p_1/x\}$ then $e \models p_o = p_1$. Thus, let R be the $E_r^c$-indexed family given by:

$$R_f = \{(p_o', p_1') \mid \exists r. \ f \models p_i' = r\{p_i/x\} \quad (i=o,1) \ \& \ wie_x(r,f) \leqslant e \ \& \ x \notin UG(r)\}$$

We want to show that R is a parameterized bisimulation. Since $(p_o,p_1) \varepsilon R_e$ (choose $r=q$) we will then have $e \models p_o = p_1$.

Note, that $UG(p_i') = UG(r\{p_i/x\}) = UG(r)$ since x is guarded in r. Thus $UG(p_o) = UG(p_1)$.

It remains to prove that $R \subseteq \mathbb{B}(R)$. So let $(p_o', p_1') \varepsilon R_f$, $f \overset{a}{\Rightarrow} g$ and $p_o' \xrightarrow{a} p_o''$. Since $f \models p_o' = r\{p_o/x\}$ and x is guarded in r it follows from (P1) and $\equiv \subseteq \sim$ that $r \xrightarrow{a} r'$ for some $r'$ with $g \models p_o'' = r'\{p_o/x\}$. Using (P3) also $r\{p_1/x\} \xrightarrow{a} \equiv r'\{p_1/x\}$, and since $f \models p_1' = r\{p_1/x\}$ therefore $p_1' \xrightarrow{a} p_1''$ for some $p_1''$ with $g \models p_1'' = r'\{p_1/x\}$. We shall prove that this is a matching move for $p_1'$. From the Derivation Lemma 4.2-6 it follows that $wie_x(r',g) \leqslant wie_x(r,f) \leqslant e$. Thus using the Substitution Theorem 4.2-10:

$$g \models p_i'' = r'\{r\{p_i/x\}/x\} \qquad i=o,1$$

or by properties of substitution:

$$g \models p_i'' = r'\{r/x\}\{p_i/x\} \qquad i=o,1$$

Note, that $wie_x(r'\{r/x\},g) \simeq wie_x(r, wie_x(r',g)) \leqslant wie_x(r,e) \leqslant e$, by the Substitution Lemma 4.2-7 and monotonicity. Since obviously x is guarded in $r'\{r/x\}$ therefore $(p_o'', p_1'') \varepsilon R_g$. $\square$

We can now state the soundness of $\underline{S}_{rr}$:

**Theorem 4.2-13:** (Soundness of $\underline{S}_{rr}$)
For all $e \varepsilon E_r^c$ and $p,q \varepsilon P_r$:

$$e \vdash_R p = q \quad \text{implies} \quad e \models p = q$$

**Proof:** We must show that each axiom of $\underline{S}_{rr}$ is valid and that each rule of $\underline{S}_{rr}$ preserves validity.

For $C_2$,S1-S4 and R1-R3 soundness follows from the soundness of $\underline{S}_M$ and $\sim_U = \sim$ . For E1-E3 appeal to proposition 2.2-5. C1 preserves validity by the previous Substitution Theorem 4.2-10. For CONS appeal to theorem 2.4-10; NIL is valid since $\Phi$ is obviously a minimal environment; and for COMB use lemma 2.4-4. Validity of ANNIHIL is immediate. Finally, R4 preserves validity by the previous Invariant Theorem 4.2-12. □

## 4.2.5 Restricted completeness of $\underline{S}_{rr}$.

In order to obtain a completeness result for $\underline{S}_{rr}$ we shall extend the Unique Solution Theorem 4.2-2 (used in the completeness proof of $\underline{S}_M$) to systems of recursive, parameterized equations. Just as theorem 4.2-2 is a generalization of the rule R4 of $\underline{S}_M$, so will its extension be a generalization of R4 of $\underline{S}_{rr}$.

**Theorem 4.2-14:** (Unique Solution of Parameterized Equations)
Let $\bar{x} = (x_1,\ldots,x_m)$ and $\bar{y} = (y_1,\ldots,y_n)$ be distinct variables. Let $\bar{p} = (p_1,\ldots,p_m)$ be expressions with free variables in $(\bar{x},\bar{y})$ in which each $x_i$ is guarded. Let $\bar{e} = (e_1,\ldots,e_m)$ be (closed) environment expressions such that for all $i,j \leqslant m$, $\mathrm{wie}_{x_j}(p_i,e_i) \leqslant e_j$. Then there exist expressions $\bar{r} = (r_1,\ldots,r_m)$ with free variables in $\bar{y}$ such that:

$$e_i \vdash_R r_i = p_i\{\bar{r}/\bar{x}\} \qquad (i \leqslant m)$$

Moreover, $\bar{r}$ is unique up to provable equivalence, i.e. if $\bar{r}' = (r_1',\ldots,r_m')$ with free variables in $\bar{y}$ also satisfies the m equations then:

$$e_i \vdash_R r'_i = r_i \qquad (i \leqslant m) \qquad \square$$

The proof of the above theorem is closely analogous to the proof of the theorem 4.2-2 in /Mil82/ except for the additional difficulties caused by the parameterization of the equations. To cope with these special difficulties we shall repeatedly appeal to the properties established in section 4.2.3.

<u>Proof (of theorem 4.2-14):</u>
The proof is by **induction on m:**

<u>For m=1</u> take $r_1 = \mu x_1 . p_1$. Then from R2 and <u>CONS</u> clearly $e_1 \vdash_R r_1 = p_1\{r_1/x_1\}$. Since by assumption, $\mathrm{wie}_{x_1}(p_1, e_1) \leqslant e_1$, if $e_1 \vdash_R r'_1 = p_1\{r'_1/x_1\}$ then by R4, $e_1 \vdash_R r'_1 = \mu x_1 . p_1$ and hence $e_1 \vdash_R r'_1 = r_1$.

<u>Step:</u> Assume the result holds for m and let $\bar{p} = (p_1, \ldots, p_m)$ and $p_{m+1}$ be expressions with free variables in $(\bar{x}, x_{m+1}, \bar{y})$ in which each $x_i$ $(i \leqslant m+1)$ is guarded, and let $\bar{e} = (e_1, \ldots, e_m)$ and $e_{m+1}$ be (closed) environment expressions such that for all $i, j \leqslant m+1$ $\mathrm{wie}_{x_j}(p_i, e_i) \leqslant e_j$. We first deal with <u>existence</u> of expressions $\bar{r} = (r_1, \ldots, r_m)$ and $r_{m+1}$ such that:

$$(1) \qquad e_i \vdash_R r_i = p_i\{\bar{r}/\bar{x}, r_{m+1}/x_{m+1}\} \qquad (i \leqslant m+1)$$

For this purpose, first set:

$$(2) \qquad q_{m+1} = \mu x_{m+1} . p_{m+1}$$

$$(3) \qquad q_i = p_i\{q_{m+1}/x_{m+1}\} \qquad (i \leqslant m)$$

Obviously each $q_i$ has free variables in $(\bar{x}, \bar{y})$ with $\bar{x}$ guarded. In order to appeal to the induction hypothesis we prove that $\bar{e}$ is indeed invariant wrt. $\bar{q}$, i.e. for all $i, j \leqslant m$, $\mathrm{wie}_{x_j}(q_i, e_i) \leqslant e_j$. We calculate:

$$
\begin{array}{lll}
(4) & \mathrm{wie}_{x_j}(q_i, e_i) = & \\
& \mathrm{wie}_{x_j}(p_i\{q_{m+1}/x_{m+1}\}, e_i) \simeq & (4.2\text{-}7) \\
& \left.\begin{array}{l}\mathrm{wie}_{x_j}(p_i, e_i) + \\ \quad \mathrm{wie}_{x_j}(q_{m+1}, \mathrm{wie}_{x_{m+1}}(p_i, e_i))\end{array}\right\} \leqslant & \begin{array}{l}(\text{assum},) \\ (4.2\text{-}5)\end{array} \\
& e_j + \mathrm{wie}_{x_j}(q_{m+1}, e_{m+1}) = & \\
& e_j + \mathrm{wie}_{x_j}(\mu x_{m+1} \cdot p_{m+1}, e_{m+1}) \simeq & (4.2\text{-}9) \\
& e_j + \mathrm{wie}_{x_j}(p_{m+1}, e_{m+1}) \leqslant & (\text{assum}) \\
& e_j &
\end{array}
$$

Now we can apply the induction hypothesis to $\bar{q} = (q_1, \ldots, q_m)$ and $\bar{e}$ to obtain expressions $\bar{r} = (r_1, \ldots, r_m)$ such that:

$$(5) \qquad e_i \vdash_R r_i = q_i\{\bar{r}/\bar{x}\} \qquad (i \leqslant m)$$

Now take $r_{m+1} = q_{m+1}\{\bar{r}/\bar{x}\}$ and rewrite (5) using (3):

$$(6) \qquad e_i \vdash_R r_i = p_i\{q_{m+1}/x_{m+1}\}\{\bar{r}/\bar{x}\} \qquad (i \leqslant m)$$

which by distinctness of $x_{m+1}$ and $\bar{x}$ and (P7) gives:

$$(7) \qquad e_i \vdash_R r_i = p_i\{\bar{r}/\bar{x}\,,\, q_{m+1}\{\bar{r}/\bar{x}\}/x_{m+1}\} \qquad (i \leqslant m)$$

which by definition of $r_{m+1}$ is nothing more than:

$$(8) \qquad e_i \vdash_R r_i = p_i\{\bar{r}/\bar{x}\,,\, r_{m+1}/x_{m+1}\} \qquad (i \leqslant m)$$

Now, $r_{m+1} = q_{m+1}\{\bar{r}/\bar{x}\} = (\mu x_{m+1} \cdot p_{m+1})\{\bar{r}/\bar{x}\} = \mu x_{m+1} \cdot (p_{m+1}\{\bar{r}/\bar{x}\})$ since $x_{m+1}$ is neither in $\bar{x}$ nor free in $\bar{r}$. By R2 then

$$(9) \qquad U \vdash_R r_{m+1} = p_{m+1}\{\bar{r}/\bar{x}\}\{r_{m+1}/x_{m+1}\}$$

and since $x_{m+1}$ is not free in $\bar{r}$ and $e_{m+1} \leqslant U$:

$$(10) \qquad e_{m+1} \vdash_R r_{m+1} = p_{m+1}\{\bar{r}/\bar{x}\,,\, r_{m+1}/x_{m+1}\}$$

as required (we are actually using $p \equiv q$ implies $U \vdash_R p = q$ - which follows from R1).

For <u>uniqueness</u> assume that (1) is also satisfied by expression $\bar{r}' = (r_1', \ldots, r_m')$ and $r_{m+1}'$ with free variables in $\bar{y}$. Then by (P6) and (P7) (and $p \underline{=} q$ implies $U \vdash_R p = q$):

(11) $$e_{m+1} \vdash_R r_{m+1}' = p_{m+1}\{\bar{r}'/\bar{x}\}\{r_{m+1}'/x_{m+1}\}$$

Now $x_{m+1}$ is guarded in $p_{m+1}\{\bar{r}'/\bar{x}\}$ and:

(12) $$\begin{aligned} & wie_{x_{m+1}}(p_{m+1}\{\bar{r}'/\bar{x}\}, e_{m+1}) \simeq \\ & \left.\begin{aligned} & \sum_{i \leq m} wie_{x_{m+1}}(r_i', wie_{x_i}(p_{m+1}, e_{m+1})) \\ & + \ wie_{x_{m+1}}(p_{m+1}, e_{m+1}) \end{aligned}\right\} \simeq \quad \begin{array}{l} x_{m+1} \text{ is} \\ \text{not free} \\ \text{in } r_i' \\ (i \leq m) \end{array} \\ & wie_{x_{m+1}}(p_{m+1}, e_{m+1}) \leq \\ & e_{m+1} \end{aligned}$$

So by the recursion rule R4 we have:

(13) $$e_{m+1} \vdash_R r_{m+1}' = \mu x_{m+1} \cdot (p_{m+1}\{\bar{r}'/\bar{x}\})$$

Again let $q_{m+1} = \mu x_{m+1} \cdot p_{m+1}$. Since $x_{m+1}$ is not in $\bar{x}$ and not free in $\bar{r}'$:

(14) $$e_{m+1} \vdash_R r_{m+1}' = q_{m+1}\{\bar{r}'/\bar{x}\}$$

Since $wie_{x_{m+1}}(p_i, e_i) \leq e_{m+1}$ we can by the congruence rule C1 replace $r_{m+1}'$ with $q_{m+1}\{\bar{r}'/\bar{x}\}$ in the equation for $r_i'$. I.e.:

(15) $$e_i \vdash_R r_i' = p_i\{\bar{r}'/\bar{x}, q_{m+1}\{\bar{r}'/\bar{x}\}/x_{m+1}\} \qquad (i \leq m)$$

or by (P7):

(16) $$e_i \vdash_R r_i' = p_i\{q_{m+1}/x_{m+1}\}\{\bar{r}'/\bar{x}\} \qquad (i \leq m)$$

Now let $q_i = p_i\{q_{m+1}/x_{m+1}\}$ for $i \leq m$. Then:

(17) $$e_i \vdash_R r_i' = q_i\{\bar{r}'/\bar{x}\} \qquad (i \leq m)$$

We want to apply the induction hypothesis to $\bar{e}$ and $\bar{q} = (q_1,\ldots,q_m)$. So we calculate for $i,j\leqslant m$:

$$\begin{array}{lll}
(18) & \mathrm{wie}_{x_j}(q_i,e_i) = & \\
& \mathrm{wie}_{x_j}(p_i\{\mu x_{m+1}\cdot p_{m+1}\}, e_i) \leqslant & (4.2\text{-}8) \\
& \mathrm{wie}_{x_j}(p_i,e_i) + \mathrm{wie}_{x_j}(p_{m+1},e_{m+1}) \leqslant & (\text{assum}) \\
& e_j &
\end{array}$$

Thus by induction hypothesis we have:

$$(19) \qquad e_i \vdash_R r_i' = r_i \qquad (i\leqslant m)$$

By 4.2-9 we have $\mathrm{wie}_{x_i}(q_{m+1},e_{m+1}) = \mathrm{wie}_{x_i}(\mu x_{m+1}\cdot p_{m+1},e_{m+1}) \simeq \mathrm{wie}_{x_i}(p_{m+1},e_{m+1}) \leqslant e_i$. So we can substitute $\bar{r}'$ for $\bar{r}$ in (14) obtaining:

$$(20) \qquad e_{m+1} \vdash_R r_{m+1}' = q_{m+1}\{\bar{r}/\bar{x}\}$$

and hence by definition of $r_{m+1}$:

$$(21) \qquad e_{m+1} \vdash_R r_{m+1}' = r_{m+1}$$

which completes the proof. □

$\underline{S}_{rr}$ is obviously an extension of $\underline{S}_M$ in the sense that if $\vdash_M p = q$ then $U \vdash_R p = q$: for every application of a rule or axiom of $\underline{S}_M$ in the proof of $\vdash_M p = q$ simply use the corresponding rule of $\underline{S}_{rr}$ with the environment e instantiated to U (note that with this instantiation the invariant condition in R4 of $\underline{S}_{rr}$ becomes trivially true). The equational characterization theorem 4.2-3 therefore generalizes to $\underline{S}_{rr}$ in the following way:

<u>Theorem 4.2-15:</u> (Equational Characterization in $\underline{S}_{rr}$)
For any expression p with free variables in $\bar{y}$, there exist expressions $p_1,\ldots,p_h$ $(h\geqslant 1)$ with free variables in $\bar{y}$, satisfying h equations:

$$U \vdash_R p_i = \sum_{j=1}^{m(i)} a_{ij}\cdot p_{f(i,j)} + \sum_{j=1}^{n(i)} y_{g(i,j)} \qquad (i \leqslant h)$$

and moreover:

$$U \vdash_R p = p_1$$

$\square$

Unfortunately we have only been able to prove a restricted completeness result for $\underline{S}_{rr}$: if $e \models p = q$ and $e$ is deterministic then also $e \vdash_R p = q$. We shall in the next section show how to extend $\underline{S}_{rr}$ to a complete proof system. Whether $\underline{S}_{rr}$ itself is complete or not is left as an open problem.

An environment e is deterministic if e=U or there exist environment expression $e_1,\ldots,e_k$ satisfying k equations:

$$e_i \sim \sum_{j=1}^{o(i)} b_{ij}\cdot e_{h(i,j)} \qquad (i \leqslant k)$$

and moreover:

$$e \sim e_1$$

such that for all $i \leqslant k$ and all $j,j' \leqslant o(i)$ if $b_{ij} = b_{ij'}$ then $j=j'$. Thus if $b.e_j$ and $b.e_{j'}$ are summands of the righthand side of the equation for $e_i$, then $j=j'$.

Theorem 4.2-16: (Restricted Completeness for $\underline{S}_{rr}$)
If e is deterministic and $e \models p = p'$ then $e \vdash_R p = p'$

Proof: If e=U then the theorem follows by the completeness theorem for $\underline{S}_M$, $\sim\, = \,\sim_U$ and $\vdash_M p = q$ implies $U \vdash_R p = q$. Otherwise, there exist k equations such that:

$$e_i \sim \sum_{j=1}^{o(i)} b_{ij}\cdot e_{h(i,j)} \qquad (i \leqslant k)$$

with $e \sim e_1$ and for all $i \leqslant k$, $j,j' \leqslant o(i)$, if $b_{ij} = b_{ij'}$ then $j=j'$. By theorem 4.2-15 there are provable equations $U \vdash_R p = p_1$, $U \vdash_R p' = p'_1$ and

$$U \vdash_R p_i = \sum_{j=1}^{m(i)} a_{ij} \cdot p_{f(i,j)} + \sum_{j=1}^{n(i)} y_{g(i,j)} \qquad (i \leqslant \ell)$$

$$U \vdash_R p'_i = \sum_{j=1}^{m'(i)} a'_{ij} \cdot p'_{f'(i,j)} + \sum_{j=1}^{n'(i)} y_{g'(i,j)} \qquad (i \leqslant \ell')$$

Now let $I = \{(i_1,i_2,i_3) \mid e_{i_3} \models p_{i_1} = p'_{i_2}\}$. Then obviously $(1,1,1) \varepsilon I$. For $(i_1,i_2,i_3) \varepsilon I$ define:

$$J_{i_1 i_2 i_3} = \{(j_1,j_2,j_3) \mid a_{i_1 j_1} = a'_{i_2 j_2} = b_{i_3 j_3} \;\&\; (f(i_1,j_1); f'(i_2,j_2); h(i_3,j_3)) \varepsilon I\}$$

Note, that for all $j_3 \leqslant o(i_3)$, $J_{i_1 i_2 i_3} \cap \{(j_1 j_2 j_3) \mid j_1 \leqslant m(i_1) \;\&\; j_2 \leqslant m'(i_2)\}$ gives a <u>total surjective</u> relationship between:

$$\{j_1 \mid j_1 \leqslant m(i_1) \;\&\; a_{i_1 j_1} = b_{i_3 j_3}\}$$

and $\{j_2 \mid j_2 \leqslant m'(i_2) \;\&\; a_{i_2 j_2} = b_{i_3 j_3}\}$

(This is a direct consequence of the definition of parameterized bisimulation). We now consider the following formal equations, one for each $(i_1,i_2,i_3) \varepsilon I$:

$$(*) \qquad e_{i_3} \vdash X_{i_1 i_2 i_3} = \sum_{(j_1 j_2 j_3) \varepsilon J_{i_1 i_2 i_3}} a_{i_1 j_1} \cdot X_{f(i_1 j_1) f'(i_2 j_2) h(i_3 j_3)} + \sum_{j=1}^{n(i_1)} y_{g(i_1 j)}$$

where the $X_{i_1 i_2 i_3}$ are not in $\bar{y}$.

First, we claim that the formal equations are satisfied when each $X_{i_1 i_2 i_3}$ is instantiated to $p_{i_1}$. To see this note that the typical equation becomes:

$$e_{i_3} \vdash p_{i_1} = \sum_{(j_1 j_2 j_3) \varepsilon J_{i_1 i_2 i_3}} a_{i_1 j_1} \cdot p_{f(i_1 j_1)} + \sum_{j=1}^{n(i)} y_{g(i_1 j)}$$

which is provable in $\underline{S}_{rr}$: using the already proven equation for $p_{i_1}$ in U we can use <u>ANNIHIL</u> and <u>COMB</u> (or <u>NIL</u>) to cancel out all terms on the righthand side not relevant in $e_{i_3}$. By the totality of $J_{i_1 i_2 i_3}$ the result of this will give an equation for $p_{i_1}$ which is identical to the one above except for a difference in the way summands are repeated.

Second, by the surjectivity of $J_{i_1 i_2 i_3}$, it can be argued that the formal equations are satisfied when each $X_{i_1 i_2 i_3}$ is instantiated to $p'_{i_2}$. Let us write the equation (*) for $(i_1 i_2 i_3) \varepsilon I$ as:

$$e_{i_3} \vdash X_{i_1 i_2 i_3} = RS_{i_1 i_2 i_3}$$

We want to appeal to the Unique Solution Theorem 4.2-14 for this system of parameterized equations. Obviously each $X_{j_1 j_2 j_3}$ in $RS_{i_1 i_2 i_3}$ is guarded. We must verify that for each $(i_1, i_2, i_3)$, $(i'_1, i'_2, i'_3) \varepsilon I$:

$$\text{wie}_{X_{i'_1 i'_2 i'_3}}\left(RS_{i_1 i_2 i_3}, e_{i_3}\right) \leqslant e_{i'_3}$$

By the form of $RS_{i_1 i_2 i_3}$ and the equation for $e_{i_3}$:

$$\mathrm{wie}_{X_{i_1' i_2' i_3'}}\left(RS_{i_1 i_2 i_3}, e_{i_3}\right)$$

$$\simeq \sum \left\{ e_{h(i_3,j)} \mid j \leqslant o(i_3) \ \& \ b_{i_3 j} . X_{i_1' i_2' i_3'} \text{ is a summand of } RS_{i_1 i_2 i_3} \right\}$$

$$\simeq \sum \left\{ e_{h(i_3,j)} \mid j \leqslant o(i_3) \ \& \ \exists (j_1 j_2 j_3) \varepsilon J_{i_1 i_2 i_3} .\ a_{i_1 j_1} = b_{i_3 j} \ \& \ (f(i_1 j_1), f'(i_2, j_2), h(i_3 j_3)) = (i_1', i_2', i_3') \right\}$$

Assume the above set contains $e_{h(i_3,j)}$. Then for some $(j_1 j_2 j_3) \varepsilon J_{i_1 i_2 i_3}$ $a_{i_1 j_1} = b_{i_3 j}$ and $(i_1', i_2', i_3') = (f(i_1 j_1), f'(i_2 j_2), h(i_3 j_3))$. By definition of J, $b_{i_3 j} = b_{i_3 j_3} = a_{i_1 j_1}$ and hence by determinism, $j = j_3$. Hence, $e_{h(i_3,j)} = e_{h(i_3,j_3)} = e_{i_3'}$. Thus as required:

$$\mathrm{wie}_{X_{i_1' i_2' i_3'}}\left(RS_{i_1 i_2 i_3}, e_{i_3}\right) \leqslant e_{i_3'}$$

Thus, uniqueness of solutions to the formal parameterized equations (*), follows from the Unique Solution Theorem 4.2-14. I.e. for all $(i_1, i_2, i_3) \varepsilon I$:

$$e_{i_3} \vdash_R p_{i_1} = p'_{i_2}$$

and especially:

$$e_1 \vdash_R p_1 = p'_1$$

□

### 4.2.6 The proof system $\underline{S}_{rr}^{M}$.

In the above proof, the determinism of the environment e is absolutely necessary for the condition $wie_{X_{i_1'i_2'i_3'}}(RS_{i_1 i_2 i_3}, e_{i_3}) \leqslant e_{i_3'}$ to hold, and hence necessary for the subsequent appeal to the unique solution theorem 4.2-14 to be valid. We have not been able to generalize the restricted completeness theorem for $\underline{S}_{rr}$ to non-deterministic environments nor have we been able to find any counter-examples for such a generalization. The (full) completeness of $\underline{S}_{rr}$ is as such an open problem.

However, as we shall see in this section, $\underline{S}_{rr}$ can be extended to a fully complete proof system. The extended system is based on the fact that any parameterized equivalence problem, $e \models p = q$, is equivalent to a problem, $e^D \models p^C = q^C$, where $e^D$ is a deterministic version of e (obtained by "tagging" identically labelled "branches" in e) and $p^C$ and $q^C$ are "multiplied" versions of p and q. In order to perform the "tagging" and "multiplication" operations we shall assume that the action set, Act, satisfies the following equation:

$$Act = Act_B + Act \times N$$

where N is the set of natural numbers and $Act_B$ is some set of basic actions (if Act does not satisfy this equation already we can always find an extension that does).

For $a \in Act$ and $i \in N$ let $a^i \in Act$ denote the action inr(a,i). For any $S \subseteq_{fin} N$ we now inductively define the following two syntactic operations:

$$(\_)^S : P_r \longrightarrow P_r$$

$$\uparrow_S(\_) : E_r \longrightarrow E_r$$

$$\begin{aligned}
\mathbb{O}^S &= \mathbb{O}\\
x^S &= x\\
(a.p)^S &= \sum_{i\varepsilon S} a^i.p^S\\
(p+q)^S &= p^S + q^S\\
(\mu x.p)^S &= \mu x.(p^S)
\end{aligned}$$

$$\begin{aligned}
\uparrow_S \mathbb{O} &= \mathbb{O}\\
\uparrow_S x &= x\\
\uparrow_S (a.e) &= \begin{cases} b.\uparrow_S e & ; \text{ if for some } i\in S\ b^i = a\\ \mathbb{O} & ; \text{ otherwise}\end{cases}\\
\uparrow_S (e+f) &= \uparrow_S e + \uparrow_S f\\
\uparrow_S (\mu x.e) &= \mu x.(\uparrow_S e)\\
\uparrow_S U &= \begin{cases} \mathbb{O} & ; \text{ if } S=\emptyset\\ U & ; \text{ otherwise}\end{cases}
\end{aligned}$$

Obviously $(\_)^S$ is a copying operation and $\uparrow_S(\_)$ is a de-tagging operation (in some sense the inverse of $(\_)^S$).

An easy induction on size shows that $(\_)^S$ and $\uparrow_S(\_)$ distribute over substitution in the following sense:

$$(p\{r/x\})^S = p^S\{r^S/x\}$$
$$\uparrow_S(e\{f/x\}) = \uparrow_S e\{\uparrow_S f/x\}$$

Hence, by induction on the number of rules applied, it can be shown that the operational behaviours of $p^S$ and $\uparrow_S e$ have the following characterizations:

<u>Lemma 4.2-17:</u> $p^S \xrightarrow{a} r$ iff for some $i\varepsilon S$, $b\varepsilon Act$ and $q\,\varepsilon P_r$: $a=b^i$, $r=q^S$ and $p \xrightarrow{b} q$. □

<u>Lemma 4.2-18:</u> $\uparrow_S e \overset{a}{\Longrightarrow} f$ iff for some $i\varepsilon S$ and $g\varepsilon E_r$: $f=\uparrow_S g$ and $e \overset{a^i}{\Longrightarrow} g$. □

We then have the following theorem:

<u>Theorem 4.2-19:</u> $\uparrow_S e \models p = q$ iff $e \models p^S = q^S$

<u>Proof:</u> "⇒": We show that the indexed family, R, with:

$$R_e = \{(p^S, q^S) \mid \uparrow_S e \models p = q\}$$

is a parameterized bisimulation. Since $UG(p)=UG(p^S)$ obviously whenever $(p^S,q^S)\varepsilon R_e$ then $UG(p^S)=UG(q^S)$. Now let $e \overset{a}{\Rightarrow} f$ and $p^S \xrightarrow{a} r$. Then for some $i\varepsilon S$, $p'$ and $b$: $a=b^i$, $r=p'^S$ and $p \xrightarrow{b} p'$. Thus $e \overset{b^i}{\Longrightarrow} f$ and hence $\uparrow_S e \overset{b}{\Rightarrow} \uparrow_S f$. Since $\uparrow_S e \models p = q$, $q \xrightarrow{b} q'$ with $\uparrow_S f \models p' = q'$ for some $q'$. Thus also $q^S \xrightarrow{b^i} q'^S$ which is the matching move.

"$\Leftarrow$": We show that the family, R, with:

$$R_f = \{(p,q) \mid \exists\, e.\, \uparrow_S e=f \ \ \&\ \ e \models p^S = q^S\}$$

is a parameterized bisimulation. Since $UG(p)=UG(p^S)$ obviously $UG(p)=UG(q)$ whenever $(p,q)\varepsilon R_f$. Now let $f \overset{a}{\Rightarrow} g$ and $p \xrightarrow{a} p'$. Then for some $e,e'$ and $i\varepsilon S$: $f=\uparrow_S e$, $g=\uparrow_S e'$ and $e \overset{a^i}{\Longrightarrow} e'$. Since $p \xrightarrow{a} p'$ also $p^S \xrightarrow{a^i} p'^S$, and since $e \models p^S = q^S$, $q^S \xrightarrow{a^i} r$ with $e' \models p'^S = r$ for some $r$. However, $r= q'^S$ for some $q'$ with $q \xrightarrow{a} q'$. This is obviously a matching move. □

To obtain a complete proof system we simply add the following (macro) rule, $\underline{M}$, to $\underline{S}_{rr}$:

$$\boxed{\underline{M} \qquad \frac{e \vdash p^S = q^S}{\uparrow_S e \vdash p = q}\ ;\ S\subseteq_{fin} N}$$

By the above theorem 4.2-19 this rule is obviously sound. Now, let $\underline{S}^M_{rr}$ denote the extended system and write $e \vdash_{RM} p = q$ iff $e \vdash p = q$ is provable in $\underline{S}^M_{rr}$ using all true assertions of the form $e \leqslant f$ as axioms. We then have the following completeness result:

<u>Theorem 4.2-20:</u> (Completeness of $\underline{S}^M_{rr}$)
If $e \models p = q$ then $e \vdash_{RM} p = q$.

<u>Proof:</u> For $e=U$ the theorem follows from the restricted completeness theorem, 4.2-16, for $\underline{S}_{rr}$. Otherwise $e$ has an equational characterization (using theorem 4.2-3 and soundness of $\underline{S}_M$):

$$(1) \qquad e_i \sim \sum_{j=1}^{o(i)} b_{ij}.e_{h(i,j)} \qquad (i \leq k)$$

with $e \sim e_1$. Now, let $e_1^+, \ldots, e_k^+$ be expressions satisfying the following <u>derived</u> system of equations:

$$(2) \qquad e_i^+ \sim \sum_{j=1}^{o(i)} b_{ij}^{j}.e_{h(i,j)}^+ \qquad (i \leq k)$$

and let $e^+ = e_1^+$. By the structure of the derived system $e^+$ is obviously deterministic. Let $S = \{1, \ldots, \max\{o(i) \mid i \leq k\}\}$. Then, by the definition of $\uparrow_S(\_)$ and since $e \sim f$ implies $\uparrow_S e \sim \uparrow_S f$, $\uparrow_S e_1^+, \ldots, \uparrow_S e_k^+$ will satisfy the original equations (1). By uniqueness (theorem 4.2-2 and soundness and completeness of $\underline{S}_M$) therefore $e_i \sim \uparrow_S e_i^+$ for all $i \leq k$ and especially $e_1 \sim \uparrow_S e_1^+$. Since $\sim \subseteq \leq$ we can therefore conclude from theorem 4.2-19 that:

$$e \models p = q \quad \text{iff} \quad e^+ \models p^S = q^S$$

Since $e^+$ is deterministic we can apply the restricted completeness theorem, 4.2-16, giving:

$$e^+ \vdash_{RM} p^S = q^S$$

Now, use the new rule <u>M</u> to obtain:

$$\uparrow_S e^+ \vdash_{RM} p = q$$

and finally, by <u>CONS</u>, since $e \leq \uparrow_S e^+$:

$$e \vdash_{RM} p = q$$

□

<u>Example 4.2-21:</u> Let us illustrate the completeness proof above with an example. Let $e = \mu x.(a.b.x + a.c.\emptyset)$, $p = \mu x.(a.b.x + a.c.\emptyset)$ and $q = \mu x.(a.b.x + a.c.\emptyset + a.\emptyset)$. We want to prove $e \vdash p = q$. Obviously the environment $e$ is <u>not</u> deterministic and the restricted completeness proof of $\underline{S}_{rr}$ is therefore not applicable. However, let:

$$e' = \mu x.(a_1.b_1.x + a_2.c_1.\emptyset)$$

$$p' = \mu x.\big( a_1.(b_1.x + b_2.x) + a_2.(b_1.x + b_2.x) + a_1.(c_1.\mathbb{O} + c_2.\mathbb{O}) + a_2.(c_1.\mathbb{O} + c_2.\mathbb{O}) \big)$$

$$q' = \mu x.\big(a_1.(b_1.x + b_2.x) + a_2.(b_1.x + b_2.x) + a_1.(c_1.\mathbb{O} + c_2.\mathbb{O}) + a_2.(c_1.\mathbb{O} + c_2.\mathbb{O}) + a_1.\mathbb{O} + a_2.\mathbb{O} \big)$$

Then it is easily seen that $\uparrow_{\{1,2\}} e' = e$, $p^{\{1,2\}} = p'$ and $q^{\{1,2\}} = q'$. Hence, by theorem 4.2-19, $e \models p = q$ iff $e' \models p' = q'$. Since $e'$ is obviously deterministic, we can apply the restricted completeness proof for $\underline{S}_{rr}$ to $e' \models p' = q'$. □

An obvious way of demonstrating full completeness of the system $\underline{S}_{rr}$ would be to prove that the new rule $\underline{M}$ is a derived rule in $\underline{S}_{rr}$, i.e. to prove that:

$$e \vdash_R p^S = q^S \quad \text{implies} \quad \uparrow_S e \vdash_R p = q$$

However, an attempt of proving this by the obvious induction on the number of rules applied for $e \vdash_R p^S = q^S$ with a case-analysis on the last rule applied fails on the rule E3 of $\underline{S}_{rr}$ (it does not seem possible to appeal to the induction hypothesis in this case). Thus, full completeness of $\underline{S}_{rr}$ remains open.

By the definition of $e \vdash_{RM} p = q$ it follows that $\underline{S}_{rr}^M$ is only complete relative to true assertions of the form $e \leqslant f$, where $e, f \varepsilon E_r^c$. However, a complete proof system for these assertions is easily derived from the proof system for $\sim, \underline{S}_M$, and thus a genuine complete proof system for parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_r^c$ can be obtained.

## 4.3 AN ALTERNATIVE PROOF SYSTEM FOR REGULAR BEHAVIOURS

In this section we shall present an alternative axiomatization of parameterized bisimulation over $\mathbb{P}_r$ and $\mathbb{E}_r^c$. The proof system is based on a reduction of parameterized equivalences involving regular environments and processes to parameterized equivalences where the environment is finite. This reduction corresponds closely to the results which hold for Moore exprriments on finite automatas (see /Mo56,Con71/), and the final proof system is analogous to Salomaa's (alternative) proof system, F3, for equalities between regular expressions /Sal66/.

First, we claim that a proof system consisting of $\underline{S}_M$ with all equalities being parameterized with U, and the rules CONG, CONS, NIL, COMB and ANNIHIL of $\underline{S}_{ff}$ will give a sound and complete proof system for parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_f$. The completeness proof is closely analogous to the proof of theorem 4.1-4 , the only difference is that an equational characterization instead of a sumform (as in 4.1-4) for the processes has to be used. The proof proceeds - as the proof for 4.1-14 - by induction on the size of the sumform for the environment. We leave it to the reader to formally verify the details involved. Let $\underline{S}_{rf}$ denote this proof system. We shall in the following extend $\underline{S}_{rf}$ to a complete proof system for parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_r^c$. The extended system is based on the following way of approximating a recursive environment expression with non-recursive ones:

Definition 4.3-1: For all $n \in \omega$ define the (syntactic) function $\underline{app}_n : E_r \rightarrow E_r$ inductively as follows:

$$app_o\, f = \Phi$$

and for $n > 0$:

$$\begin{aligned}
app_n \mathbb{O} &= \mathbb{O} \\
app_n x &= x \\
app_n(f+g) &= app_n f + app_n g \\
app_n(a.g) &= a.(app_{n-1} g) \\
app_n(\mu x.f) &= [app_n f]_x^n
\end{aligned}$$

where for an expression g we define $g_x^o = \mathbb{O}$ and $g_x^{n+1} = g\{g_x^n/x\}$ . □

Obviously for any $n \in \omega$ and any expression f, $app_n f$ is a non-recursive expression, and if f is closed so is $app_n f$. The idea is that $app_n f$ is a (finite) non-recursive n'th approximation of f with respect to $\leqslant$. This is formally justified by the following lemmas:

Lemma 4.3-2: For all $e \in E_r$: $app_n e \leqslant e$ .

Proof: By induction on the structure of e. For the recursion case use that whenever $e \leqslant e'$ then $f\{e/x\} \leqslant f\{e'/x\}$. □

Lemma 4.3-3: For all $e \in E_r$: $e \leqslant^n app_n e$ .

Proof: By the structure of e. All cases except the recursion case is trivial. For $e = \mu x.f$ we have:

$$app_n e = app_n(\mu x.f) = [app_n f]_x^n$$

Let us prove by induction on k that:

$$(*) \qquad \mu x.f \leqslant^k [app_n f]_x^k \qquad \text{for } k \leqslant n$$

The base case, k=0, is trivial. For the induction step assume (*) holds for all $j<k$, and let $\mu x.f \overset{a}{\Rightarrow} g$. I.e. by (P4), for some $f'$, $f \overset{a}{\Rightarrow} f'$ with $g \equiv f'\{\mu x.f/x\}$ . By the structural induction hypothesis we have $f \leqslant^n app_n f$ and thus, since $k \leqslant n$, $f \leqslant^k app_n f$ . Hence $app_n f \overset{a}{\Rightarrow} f''$ for some $f''$ with $f' \leqslant^{k-1} f''$. By (P3) then:

$$[app_n f]_x^k = (app_n f)\{[app_n f]_x^{k-1}/x\} \overset{a}{\Rightarrow} h$$

where $h \equiv f''\{[app_n\, f]_x^{k-1}/x\}$. We claim that this is a matching move. To see this note that $f' \lesssim^{k-1} f''$ and by induction hypothesis $\mu x.f \lesssim^{k-1} [app_n\, f]_x^{k-1}$. Since $f \lesssim^n g$ and $f' \lesssim^n g'$ implies $f\{f'/x\} \lesssim^n g\{g'/x\}$ we conclude:

$$g \simeq f'\{\mu x.f/x\} \lesssim^{k-1} f''\{[app_n\, f]_x^{k-1}/x\} \simeq h \qquad \square$$

Combining lemma 4.3-2 and 4.3-3 we have $e \simeq^n app_n\, e$. Due to the possibility of unguarded recursion the stronger relationship $e \sim^n app_n\, e$ fails to hold.

The consequence law, theorem 2.4-10, can be refined by introducing indices:

<u>Lemma 4.3-4:</u> Whenever $p \sim_f^n q$ and $e \lesssim^n f$ then also $p \sim_e^n q$.

<u>Proof:</u> An easy induction on n. $\square$

Since $\mathbb{P}_r$ is image-finite we can conclude the following as an easy corollary:

$$p \sim_e q \quad \Leftrightarrow \quad \forall n \varepsilon \omega.\ p \sim_e^n q \quad \Leftrightarrow \quad \forall n \varepsilon \omega.\ p \sim_{app_n e} q$$

Hence as a first attempt of extending $\underline{S}_{rf}$ we might add the following <u>infinitary</u> rule:

$$\frac{app_0 e \vdash p = q \qquad app_1 e \vdash p = q \quad \ldots\ldots \quad app_n e \vdash p = q \quad \ldots}{e \vdash p = q}$$

However, this rule can be replaced by a <u>finitary</u> one, since – as we shall show in the following – only finitely many approximations of e needs to be considered. To see this, let for $S \subseteq P_r$ and $U \subseteq E_r^c$, $S_U$ be the $E_r^c$-indexed family of binary relations over $P_r$ defined by:

$$(S_U)_e = \begin{cases} S \times S & ;\ \text{if } e \varepsilon U \\ \emptyset & ;\ \text{otherwise} \end{cases}$$

A set $S \subseteq P_r$ is $\rightarrow$-closed iff whenever $p \varepsilon S$ and $p \xrightarrow{a} p'$ then $p' \varepsilon S$. Similarly, a set $U \subseteq E_r^c$ is $\Rightarrow$-closed iff whenever $e \varepsilon U$ and $e \overset{a}{\Rightarrow} e'$ then $e' \varepsilon U$.

Lemma 4.3-5: If $S \subseteq P_r$ is $\rightarrow$-closed and $U \subseteq E_r^c$ is $\Rightarrow$-closed then for all $E_r^c$-indexed families of binary relations over $P_r$, $R$:

$$\mathbb{B}(R \cap S_U) \cap S_U = \mathbb{B}(R) \cap S_U$$

Proof: Only the "$\supseteq$"-direction is non-trivial. Since $(S_U)_e = \emptyset$ for $e \not\varepsilon U$ we only need to prove:

$$\left[\mathbb{B}(R) \cap S_U\right]_e \subseteq \left[\mathbb{B}(R \cap S_U) \cap S_U\right]_e$$

for $e \varepsilon U$. Let $(p,q) \varepsilon \left[\mathbb{B}(R) \cap S_U\right]_e$ with $e \varepsilon U$. It suffice to prove $(p,q) \varepsilon \mathbb{B}(R \cap S_U)_e$. So let $e \overset{a}{\Rightarrow} f$ and $p \xrightarrow{a} p'$. Then $q \xrightarrow{a} q'$ with $(p',q') \varepsilon R_f$ for some $q'$. Since $S$ is $\rightarrow$-closed $(p',q') \varepsilon S \times S$ and since $U$ is $\Rightarrow$-closed $(S_U)_f = S \times S$. Thus $(p',q') \varepsilon (R \cap S_U)_f$ and hence by symmetry $(p,q) \varepsilon \mathbb{B}(R \cap S_U)_e$. $\square$

Lemma 4.3-6: If $S \subseteq P_r$ is $\rightarrow$-closed and $U \subseteq E_r^c$ is $\Rightarrow$-closed and $\sim^n \cap S_U = \sim^{n+1} \cap S_U$ then for all $m \geqslant n$, $\sim^n \cap S_U = \sim^m \cap S_U = \sim \cap S_U$.

Proof: An easy induction on $m-n$ using the previous lemma 4.3-5. $\square$

The following theorem is closely analogous to the theorem for finite automata which says that any two distinguishable states of a finite automata with n states can be distiguished by some experiment of length at most n-1 (see /Mo56,Con71/).

Theorem 4.3-7: Let $S \subseteq_{fin} P_r$ be $\rightarrow$-closed and $U \subseteq_{fin} E_r^c$ be $\Rightarrow$-closed. Then for all $(p,q) \varepsilon S \times S$ and $e \varepsilon U$:

$$p \sim_e q \quad \Leftrightarrow \quad p \sim_e^N q$$

when $N \geqslant |S| \cdot |U| - |U|$.

Proof: "$\Rightarrow$": obvious.

"$\Leftarrow$": Consider the decreasing chain:

$$\sim^{o} \cap S_U \supseteq \sim^{1} \cap S_U \supseteq \ldots\ldots \supseteq \sim^{n} \cap S_U \supseteq \ldots\ldots \supseteq \sim \cap S_U$$

Let for $e \varepsilon U$, $C_e(n)$ be the number of equivalence classes of $(\sim^n \cap S_U)_e = \sim_e^n \cap (S \times S)$ and let $C_e(\infty)$ be the number of classes of $(\sim \cap S_U)_e$ ($\leqslant |S|$ since there can not be more classes than there are elements in S). Let:

$$C(n) = \sum_{e \varepsilon U} C_e(n) \quad (\leqslant \sum_{e \varepsilon U} |S| = |S| \cdot |U|)$$

then:

$$|U| = C(o) \leqslant C(1) \leqslant \ldots\ldots \leqslant C(\infty) \leqslant |S| \cdot |U|$$

Thus there must be a smallest N such that $C(N) = C(N+1)$ and hence $\sim^n \cap S_U = \sim^{n+1} \cap S_U$. We therefore have:

$$|U| = C(o) < C(1) < \ldots\ldots < C(N) \leqslant |S|\,|U|$$

and so $|U| + N \leqslant C(N) \leqslant |S| \cdot |U|$, implying $N \leqslant |S| \cdot |U| - |U|$. By the previous lemma 4.3-6 we conclude that for all $m \geqslant |S| \cdot |U| - |U|$, $\sim^m \cap S_U = \sim^{|S| \cdot |U| - |U|} \cap S_U = \sim \cap S_U$. Thus for all $(p,q) \varepsilon S \times S$, $e \varepsilon U$ and $m \geqslant |S| \cdot |U| - |U|$:

$$(p,q) \varepsilon \sim_e \quad \Leftrightarrow \quad (p,q) \varepsilon (\sim \cap S_U)_e \quad \Leftrightarrow$$
$$(p,q) \varepsilon (\sim^m \cap S_U)_e \quad \Leftrightarrow \quad (p,q) \varepsilon \sim_e^m$$

□

Corollary 4.3-8: Let $S \subseteq_{fin} P_r$ be $\rightarrow$-closed and $U \subseteq_{fin} E_r^c$ be $\Rightarrow$-closed. Then for all $(p,q) \varepsilon S \times S$ and $e \varepsilon U$:

$$p \sim_e q \quad \Leftrightarrow \quad p \sim_{app_N e} q$$

where $N \geqslant |S| \cdot |U| - |U|$.

□

It follows from this corollary that if we for all processes $p \varepsilon P_r$ and environments $e \varepsilon E_r^c$ can find <u>finite</u> and <u>closed</u> sets S and U, with $p \varepsilon S$ and $e \varepsilon U$, then we have a way of removing recursive environments in parameterized equivalences. But for $p \varepsilon P_r$ ($e \varepsilon E_r^c$) the set $DER(p) = \{p' \mid \exists s \varepsilon Act^*.\ p \overset{s}{\Rightarrow} p'\}$ ($DER(e) = \{e' \mid \exists s \varepsilon Act^*.\ e \overset{s}{\Rightarrow} e'\}$) has exactly these properties. The following function $ND: P_r \longrightarrow N$ gives an upper bound on $|DER(p)|$ :

$$\begin{aligned}
ND(\mathbb{O}) &= 1 \\
ND(x) &= 1 \\
ND(a.p) &= 1 + ND(p) \\
ND(p+q) &= ND(p) + ND(q) \\
ND(\mu x.p) &= ND(p)
\end{aligned}$$

The upper bound for $\mu x.p$ is justified since there is a 1-1 correspondance between derivatives of $\mu x.p$ and p. We therefore have the following theorem:

<u>Theorem 4.3-9:</u> For $p,q \varepsilon P_r$ and $e \varepsilon E_r^c$:

$$e \models p = q \quad \Leftrightarrow \quad app_N\, e \models p = q$$

where $N \geqslant (ND(p) + ND(q) - 1) \cdot ND(e)$.

<u>Proof:</u> Apply corollary 4.3-8 with $S = DER(p) \cup DER(q)$ and $U = DER(e)$. Note $|S| \leqslant ND(p) + ND(q)$ and $|U| \leqslant ND(e)$. □

Then adding the following finitary rule <u>A</u> to $\underline{S}_{rf}$ obviously results in a sound and complete proof system, $\underline{S}_{rf}^{A}$, for parameterized equivalence over $\mathbb{P}_r$ and $\mathbb{E}_r^c$.

$$\underline{A} \qquad \frac{app_N\, e \vdash p = q}{e \vdash p = q} \; ; \; N \geqslant (ND(p) + ND(q) - 1) \cdot ND(e)$$

## 4.4 CONCLUDING REMARKS

In this chapter we have offered complete axiomatizations of parameterized equivalence for various combinations of the process and environment system: the system $\underline{S}_{ff}^{+}$ is a complete proof system for finite behaviours, and $\underline{S}_{rr}^{M}$ and $\underline{S}_{rf}^{A}$ are (relative) complete proof systems for regular behaviours.

It is left as an open problem to decide whether the subsystem $\underline{S}_{rr}$ of $\underline{S}_{rr}^{M}$ is complete in itself or not. However, for the sake of completeness, instead of adding the macro-rule $\underline{M}$ to $\underline{S}_{rr}$, we could add a class of renaming-operators, $\_[\Phi]$, and axiomatize parameterized equivalence for the extended systems. It should then be possible to express the behaviours $p^{S}$ and $\uparrow_{S}e$ as renamed versions of p and e, and thus obtain the macro-rule $\underline{M}$ as a derived rule from the laws of renaming. Obviously several new problems have to be dealt with in this approach:

- The notion of an unguarded variable must be carefully revised in order to take account of the renamings that can affect the unguarded variable. A simple extension of UG by adding the naive rule $UG(p[\Phi]) = UG(p)$ will fail to make the congruence law hold. Instead UG(p) should be a set of pairs, $(x,\Phi)$, where x is a variable unguarded in p affected by the (total) renaming $\Phi$. (Obviously laws for combining renaming are required).

- The new definition of UG requires a revision of wie, such that the parameterized congruence law (theorem 4.2-10) remains valid.

- In order for the equational characterization, theorem 4.2-15, to extend, the rule R3 of $\underline{S}_{rr}$ must be changed so that unguarded variables inside a recursion and <u>inside a renaming "context"</u> can be removed; e.g. the variable x in $\mu x.(p + x[\Phi])$.

Finally, a whole new class of axiomatizations of parameterized equivalence can be obtained from the maximal environment construction in section 2.5. It is here shown that the parameterized equivalence problem:

$$p \sim_e q$$

is equivalent to the simulation problem:

$$e \leqslant /p,q/$$

where /p,q/ is the maximal environment identifying p and q. Thus, the problem of axiomatizing parameterized equivalence can be solved by an axiomatization of the (derived) simulation problems.

# CHAPTER 5

# PARAMETERIZED WEAK BISIMULATION

The bisimulation equivalence which we have studied so far assumes that every action is observable: a process cannot proceed without being observed. Let us now assume that there is a single, distinguished action $1 \varepsilon Act$, which is <u>unobservable</u> (Note that according to the operational semantics of CCS given in section 3.2, communication between processes in parallel gives rise to this unobservable action). We want a weakened version, $\approx$, of the bisimulation equivalence, $\sim$, which takes this into account; i.e. processes which only differ in the number of unobservable 1-actions (=delay) between observable actions should be identified. Thus we would expect $a.\mathbb{O} \approx a.1.\mathbb{O}$ to hold.

The standard way of defining $\approx$ (see /Mil80,Mil83/) is to apply the existing general notion of bisimulation (definition 2.1-15) to a <u>derived observational process system</u> $\mathbb{P}^{o} = (Pr, Act_{-1}^{*}, \rightarrow_{o})$ where <u>$Act_{-1}$</u> $= Act - \{1\}$ and <u>$\rightarrow_{o}$</u> (the <u>observational derivation relation</u>) is derived from $\rightarrow$ by absorbing any finite sequence of unobservable 1-actions between observable actions, i.e. for $s = (a_{o}, \ldots, a_{n-1}) \varepsilon Act_{-1}^{*}$:

$$p \xrightarrow{s}_0 p' \Leftrightarrow p \left(\xrightarrow{1}\right)^* \left(\xrightarrow{a_0}\right) \left(\xrightarrow{1}\right)^* \cdots \left(\xrightarrow{1}\right)^* \left(\xrightarrow{a_{n-1}}\right) \left(\xrightarrow{1}\right)^* p'$$

A bisimulation over the observational process system $\mathbb{P}^o$ is called a <u>weak</u> (or <u>observational</u>) <u>bisimulation</u> and we shall write $p \approx q$ whenever $(p,q)$ is contained in some weak bisimulation. From proposition 2.1-19 it follows that $\approx$ is an equivalence relation on Pr. We shall call $\approx$ the <u>weak bisimulation equivalence</u>.

The following easy result from /Mil83/ allows us to restrict s to range over sequences of observable actions of length at most 1. First, let $\tilde{\ }: Act^* \rightarrow Act^*_{-1}$ be the homomorphism generated by: $\tilde{a} = a$ for $a \neq 1$ and $\tilde{1} = \varepsilon$.

<u>Proposition 5.0-1:</u> $R \subseteq Pr \times Pr$ is a weak bisimulation if and only if, whenever $pRq$ and $a \varepsilon Act$, then:

(i) $p \xrightarrow{a} p' \Rightarrow \exists q'.\ q \xrightarrow{\tilde{a}}_0 q'$ & $p'Rq'$
(ii) $q \xrightarrow{a} q' \Rightarrow \exists p'.\ p \xrightarrow{\tilde{a}}_0 p'$ & $p'Rq'$

Since obviously $p \xrightarrow{s} p'$ implies $p \xrightarrow{\tilde{s}}_0 p'$ it follows that any bisimulation is also a weak bisimulation, and hence that $\sim \subseteq \approx$. □

Similarly, we shall call a simulation over $\mathbb{P}^o$ a <u>weak simulation</u> and write $p \lessapprox q$ whenever $(p,q)$ is contained in some weak simulation. From proposition 2.1-9 it follows that $\lessapprox$ is a preorder on Pr and we shall call $\lessapprox$ the <u>weak simulation ordering</u>.

The purpose of this chapter is to extend the notion of environment parameterization to <u>weak</u> bisimulation equivalence, $\approx$, and preferably in such a way that the results obtained in chapters 2 and 3 for the parameterized (strong) bisimulation equivalence extend as well. In particular we want to be able to reduce a parameterized (weak) equivalence problem of the form, $C[p] \approx_e C[q]$, to a parameterized (weak) equivalence problem involving only the inner processes p and q; i.e. we want to find an environment, f, (dependent on C and e) such that for all processes p and q:

$$(*) \qquad p \approx_f q \quad \Rightarrow \quad C[p] \approx_e C[q]$$

Preferably the described environment, f, is as small as possible wrt. the (weak) discrimination ordering $\subsetapprox$ (induced by the relative strength of the corresponding parameterized <u>weak</u> bisimulation equivalences).

Unfortunately, it will not in general be possible to perform the above reduction since $\approx$ is not a congruence wrt. all (CCS-) contexts (especially not wrt. sum contexts, $p+[\,]$, see /Mil80,HenMil83,Mil83/). To see this, assume that U is a universal environment ($\approx = \approx_U$) and that for all environments e, $\approx \subseteq \approx_e$. Then, if for environments e and contexts C we could describe an environment $f_{C,e}$ satisfying (*), the following would hold:

$$p \approx q \quad \Rightarrow \quad p \approx_{f_{C,U}} q \quad \Rightarrow$$
$$C[p] \approx_U C[q] \quad \Rightarrow \quad C[p] \approx C[q]$$

I.e. $\approx$ would, in contradiction to what we know, be a congruence wrt. all contexts.

There seems to be two ways out of this problem. One is to parameterize the congruence , $\approx^c$, induced by $\approx$ instead of parameterizing $\approx$ . However, $\approx^c$ is highly dependent on the context system considered, and it therefore seems very unlikely that we will be able to achieve any interesting results which will hold for arbitrary context systems. Also, there are context systems for which $\approx^c$ collapses down to $\sim$ (Remember that $\sim$ is a congruence wrt. all contexts according to theorem 3.1-8. Therefore for all context systems $\sim \subseteq \approx^c \subseteq \approx$). Hence, it seems that a <u>general</u> theory of parameterized weak congruence will simply reduce to that of parameterized (strong) bisimulation equivalence.

The other way of overcoming the above problem, which is the way we shall follow, is to parameterize $\approx$ but restrict our attention to contexts which preserve $\approx$ .

In section 5.1 we shall offer (sufficient) conditions on contexts, in terms of their operational semantics, which will ensure congruence of $\approx$ .

In section 5.2 we define the parameterized weak bisimulation equivalence and show how (some of) the results from chapter 2 for the parameterized (strong) bisimulation equivalence generalizes. In particular we show that the Characterization Theorem 2.4-20 ($\leqslant = \sqsubseteq$) generalizes to the weak case (i.e. $\lesssim = \underset{\approx}{\sqsubseteq}$).

In section 5.3 we study the relationship between (parameterized) strong and weak bisimulation equivalence. In particular we show that the inclusion $\sim \subseteq \approx$ generalizes to the parameterized versions (i.e. $\sim_e \subseteq \approx_e$ for all e) under certain conditions.

In section 5.4 we investigate how contexts (or more precisely: contexts satisfying the conditions of section 5.1) transform environments in the weak case, thus generalizing the results from section 3.4.

These generalizations are applied in section 5.5, where we prove the correctness of a Simple Scheduler using the parameterized weak bisimulation equivalence.

## 5.1 CONDITIONS ENSURING PRESERVATION OF $\approx$

For the remainder of this section we shall assume that $\mathbb{P} = (Pr, Act, \longrightarrow)$ is a process system closed under a context system $\mathbb{C} = (Con, Act_0 \times Act_0, \longmapsto)$ with respect to a map $\_[\_]: Con \times Pr \longrightarrow Pr$. We are looking for conditions (on $\mathbb{C}$ and/or $\mathbb{P}$) that will ensure preservation of $\approx$ with respect to all contexts of $\mathbb{C}$.

Similarly to the derivation of the observational process system $\mathbb{P}^0$ we can derive an <u>observational context system</u> $\mathbb{C}^0 = (Con, Act_{-1}^* \times Act_{-1}^*, \longmapsto_0)$ by defining the <u>observational transduction relation</u> $\longmapsto_0 \subseteq Con \times Act_{-1}^* \times Act_{-1}^* \times Con$ as:

$$C \overset{v}{\underset{u}{\longmapsto}}_0 C' \qquad \Leftrightarrow$$

$$\exists s, t \varepsilon Act^* .\ C \overset{t}{\underset{s}{\longmapsto\!\!\!\!\!\!\rightarrow}} C' \ \ \&\ \ \tilde{s} = u \ \ \&\ \ \tilde{t} = v$$

where $u, v \varepsilon Act_{-1}^*$ and $\longmapsto\!\!\!\!\!\!\rightarrow$ is defined in section 3.1.2.

As a first attempt towards conditions ensuring preservation of $\approx$, assume that the map $\_[\_]: Con \times Pr \longrightarrow Pr$ also provides a closure of $\mathbb{P}^0$ under $\mathbb{C}^0$. I.e. the <u>observational behaviour</u> of a combined process, $C[p]$, can be decomposed into and derived from the <u>observational behaviours</u> of the context $C$ and the inner process $p$. In particular if $C \overset{v}{\underset{u}{\longmapsto}}_0 C'$ and $p \overset{u}{\longrightarrow}_0 p'$ then $C[p] \overset{v}{\longrightarrow}_0 C'[p']$. Then, since $\approx$ is simply the bisimulation equivalence over $\mathbb{P}^0$, we would expect theorem 3.1-8 to generalize, thus implying that $\approx$ is preserved by all contexts of $\mathbb{C}$.

Indeed, with the right formal definition of closure, it is not difficult to prove that theorem 3.1-8 <u>does</u> generalize. However, requiring $\_[\_]$ to be a closure of $\mathbb{P}^0$ under $\mathbb{C}^0$ in the above sense is too strong a requirement since it rules out a large class of contexts which in fact do preserve $\approx$: namely, the class of guarding contexts of which the prefix-contexts of CCS provide an example. A context C is <u>guarding</u> iff whenever $C \overset{b}{\underset{u}{\mapsto}} C'$ ($b \varepsilon Act$, $u \varepsilon Act^*$) then $u = \varepsilon$ (i.e. an inner process is prevented from executing at once). To see why $\_[\_]$ is not in general an observational closure for such contexts, consider the (guarding) CCS-context a.b.[] and the CCS-process $1.\mathbb{O}$. Then $a.b.[] \overset{a}{\underset{\varepsilon}{\mapsto}}_o b.[]$ and $1.\mathbb{O} \xrightarrow{\varepsilon}_o \mathbb{O}$ but not $a.b.1.\mathbb{O} \xrightarrow{a}_o b.\mathbb{O}$. To accommodate for guarding contexts we therefore define the notion of <u>observational closure</u> as follows:

<u>Definition 5.1-1:</u> Let $\mathbb{P} = (Pr, Act, \rightarrow)$ be closed under $\mathbb{C} = (Con, Act_o \times Act_o, \mapsto)$ with respect to $\_[\_]$. Then, $\mathbb{P}$ is observationally closed under $\mathbb{C}$ (or $\_[\_]$ is an observational closure) iff whenever $p,q \varepsilon Pr$, $v \varepsilon Act^*_{-1}$ and $C \varepsilon Con$ either:

(i) C is guarding

or (ii) $C[p] \xrightarrow{v}_o q \Leftrightarrow$

$$\exists u \varepsilon Act^*_{-1}. \exists p' \varepsilon Pr. \exists C' \varepsilon Con.$$
$$C \overset{v}{\underset{u}{\mapsto}}_o C' \quad \&$$
$$p \xrightarrow{u}_o p' \quad \&$$
$$q = C'[p'] \qquad \square$$

We can now prove that with this definition of observational closure $\approx$ will be a congruence:

<u>Theorem 5.1-2:</u> If $\mathbb{P}$ is observationally closed under $\mathbb{C}$ then $\approx$ is preserved by all contexts of $\mathbb{C}$.

<u>Proof:</u> We prove that the relation:

$$R = \{(C[p], C[q]) \mid p \approx q\}$$

is a weak bisimulation using proposition 5.0-1. So let $(C[p], C[q]) \varepsilon R$ and assume $C[p] \xrightarrow{b} r$ ($b \varepsilon Act$). There are two cases to consider:

C is guarding: Then by definition 3.1-1, $C \overset{b}{\underset{u}{\mapsto\!\!\!\twoheadrightarrow}} C'$ and $p \xrightarrow{u} p'$ with $r=C'[p']$ for some $C',p'$ and u. Since C is guarding $u=\varepsilon$ and hence $p=p'$. Again by definition 3.1-1, $C[q] \xrightarrow{b} C'[q]$ and hence $C[q] \xrightarrow{\tilde{b}}_o C'[q]$. Obviously $(C'[p],C'[q]) \varepsilon R$.

C is not guarding: By definition 3.1-1, $C \overset{b}{\underset{u}{\mapsto\!\!\!\twoheadrightarrow}} C'$ and $p \xrightarrow{u} p'$ with $r=C'[p']$ for some $C',p'$ and u. Thus $C \overset{\tilde{b}}{\underset{\tilde{u}}{\mapsto\!\!\!\Rightarrow}}_o C'$ and $p \xrightarrow{\tilde{u}}_o p'$. Since $p \approx q$, $q \xrightarrow{\tilde{u}}_o q'$ with $p' \approx q'$ for some $q'$. Since in this case condition (ii) of definition 5.1-1 holds, $C[q] \xrightarrow{\tilde{b}}_o C'[q']$ which obviously is a matching move. □

Although observational closure is a sufficient condition for the preservation of $\approx$ , it is a condition which obviously is difficult to test given particular instances of process and context systems. In the following we shall therefore try to replace this (impractical) condition with conditions based on the operational semantics of the individual contexts and processes, similar in degree of complexity to the guarding condition.

First, let us from a few examples see which properties of contexts can lead to violation of the preservation of $\approx$.

1. A context may prevent the inner process from executing 1-actions and thus violate preservation of $\approx$; e.g. let C be the CCS-context $[\,]\lceil Act_{-1}$. Then $1.a.\emptyset \approx a.\emptyset$ but not $C[1.a.\emptyset] \approx C[a.\emptyset]$, since $C[1.a.\emptyset]$ is deadlocked whereas $C[a.\emptyset]$ is not.

2. By changing 1-actions performed by the inner process into observable actions the context may violate preservation of $\approx$. E.g. let C be the CCS-context $[\,][\Phi_b]$ where $b \varepsilon Act_{-1}$ and $\Phi_b(a)=a$ if $a \neq 1$ and $\Phi_b(a)=b$ otherwise. Then

$1.\mathbb{O} \approx \mathbb{O}$ but not $C[1.\mathbb{O}] \approx C[\mathbb{O}]$. Actually, if $\Phi$ is a 1-1 map with $\Phi(1) \varepsilon Act_{-1}$, then it is easily proved that for $p[\Phi] \approx q[\Phi]$ to hold we must require $p \sim q$. Thus, since $\sim \subsetneq \approx$ , $\approx$ is not preserved. Note also, that for this context $\approx^c$ collapses down to $\sim$.

3. Even if the inner process is allowed to perform 1-actions without these being made visible, the context can by changing during such a 1-transduction violate preservation of $\approx$. This is exactly what happens in a CCS sum-context, $p + [\,]$: during the 1-transduction $p + [\,] \overset{1}{\underset{1}{\mapsto}} [\,]$ a context change occurs (the process p is being discharged). To see why this violates preservation of $\approx$, note that $1.a.\mathbb{O} \approx a.\mathbb{O}$ but not $b.\mathbb{O} + 1.a.\mathbb{O} \approx b.\mathbb{O} + a.\mathbb{O}$, since $b.\mathbb{O} + a.\mathbb{O}$ has no matching move to $b.\mathbb{O} + 1.a.\mathbb{O} \overset{\varepsilon}{\Rightarrow}_0 a.0$.

From the above examples it follows that a context may violate preservation of $\approx$ if it in any way can detect or use 1-actions produced by an inner process. To avoid such contexts we introduce the following concept of <u>idle-preservation</u>:

<u>Definition 5.1-3:</u> A context C is idle-preserving iff for all $a \varepsilon Act$ and $C' \varepsilon Con$:

(i) $C \overset{a}{\underset{1}{\mapsto}} C' \Leftrightarrow a = 1 \ \& \ C = C'$

(ii) All C's derivatives are idle-preserving. □

Note, that the "$\Leftarrow$"-direction of (i) prevents contexts of type <u>1</u> from being idle-preserving. Similarly, the "$\Rightarrow$"-direction of (i) prevents contexts of type <u>2</u> and <u>3</u> from being idle-preserving.

To accommodate guarding contexts (which clearly cannot be idle-preserving) we define the following notion of asynchrony:

Definition 5.1-4: A context C is asynchronous iff:

(i) C is guarding or C is idle-preserving
(ii) All C's derivatives are asynchronous.

A context system $\mathbb{C}$ is said to be asynchronous iff all contexts of $\mathbb{C}$ are asynchronous. □

Example 5.1-5: Let C be an asynchronous CCS-context. Then the following CCS-contexts are easily shown to be asynchronous as well:

(i) Constant contexts; p.
(ii) Identity context; [].
(iii) Prefixing contexts; a.C.
(iv) Parallel contexts; $C|p$ and $p|C$ .
(v) Restriction contexts; $C\restriction S$ provided $1 \varepsilon S$.
(vi) Renaming contexts; $C[\Phi]$ provided $\Phi(1) = 1$

The following CCS-contexts are in general not asynchronous:

(vii) Sum contexts; $p + C$ and $C + p$
(viii) Join contexts; $p \& C$ and $C \& p$ □

The importance of asynchrony is due to the following theorem:

Theorem 5.1-6: If $\mathbb{P}$ is closed under $\mathbb{C}$, where $\mathbb{C}$ is an asynchronous, non-swallowing context system, then $\mathbb{P}$ is also observationally closed under $\mathbb{C}$. □

We give the proof of theorem 5.1-6 shortly. Let us first, using theorem 5.1-2, state the following immediate corollary:

Corollary 5.1-7: If $\mathbb{P}$ is closed under a non-swallowing asynchronous context system $\mathbb{C}$, then $\approx$ is preserved by all contexts of $\mathbb{C}$. □

Thus, it follows from example 5.1-5 that $\approx$ is preserved by all CCS-contexts except sum- and join-contexts as well as certain restriction- and renaming-contexts.

Even though asynchrony is a sufficient condition for $\approx$ to be preserved it is not a necessary one: consider the <u>delay-operator</u> $\delta$ from /Mil83/. For a process p, $\delta p$ is defined as $\delta p = \mu x.(1.x + p)$. As a context we define $\delta = \mu x.(1.x + [\,])$ with the following operational semantics:

$$\delta \overset{1}{\underset{0}{\longmapsto}} \delta \qquad\qquad \delta \overset{a}{\underset{a}{\longmapsto}} [\,]$$

Obviously, $\delta$ is neither guarding nor idle preserving, since $\delta \overset{1}{\underset{1}{\longmapsto}} [\,]$ violates the "$\Rightarrow$"-direction of (i) in definition 5.1-3. However, it is easily shown that $\delta$ nevertheless <u>does</u> preserve $\approx$ (see proposition 8.7 /Mil83/). Now, by modifying the operational semantics of $\delta$ slightly we can obtain an asynchronous delay-operator $\varrho$:

$$\varrho \overset{1}{\underset{0}{\longmapsto}} \varrho \qquad\qquad \varrho \overset{1}{\underset{1}{\longmapsto}} \varrho$$

$$\varrho \overset{a}{\underset{a}{\longmapsto}} [\,] \qquad a \neq 1$$

It would be interesting to see if the theory of ASCCS in /Mil83/ could be carried out using $\varrho$ instead of $\delta$. However, unlike $\delta$ it seems difficult to express $\varrho$ as a derived operator of CCS/SCCS (though results in /Sim85/ suggest that it should be possible).

By a similar modification of + we can introduce a new sum-context, $\oplus$, which is asynchronous and thus $\approx$-preserving (unlike +). The operational semantics of $\oplus$ is given by:

$$\frac{C \overset{b}{\underset{a}{\longmapsto}} C'}{C \oplus D \overset{b}{\underset{a}{\longmapsto}} C'}\ ;\ b \neq 1 \qquad\qquad \frac{C \overset{1}{\underset{a}{\longmapsto}} C'}{C \oplus D \overset{1}{\underset{a}{\longmapsto}} C' \oplus D}$$

with two symmetric rules for when D is executing.

It remains for us to prove theorem 5.1-6:

<u>Proof (of theorem 5.1-6):</u> We must prove that for all contexts C of $\mathbb{C}$ either C is guarding or C satisfies condition (ii) of definition 5.1-1. So assume C is not guarding. Thus, since C is assumed to be asynchronous, C must be idle-preserving. Let us prove that C satisfies condition (ii) of definition 5.1-1:

"$\Rightarrow$": Let $C[p] \overset{v}{\longmapsto}_o q$. Then for some $s \varepsilon Act^*$ with $\tilde{s} = v$ $C[p] \overset{s}{\longmapsto} q$. If $s = \varepsilon$ then $q = C[p]$ and obviously $C \underset{\varepsilon}{\overset{\varepsilon}{\longmapsto}}_o C$ and $p \overset{\varepsilon}{\longrightarrow}_o p$. Otherwise, by lemma 3.1-3, $C \underset{t}{\overset{s}{\mapsto\!\!\twoheadrightarrow}} C'$, $p \overset{t}{\longrightarrow} p'$ with $q = C'[p']$ for some $C', p'$ and $t \varepsilon Act^*$. Then by definition $C \underset{\tilde{t}}{\overset{v}{\longmapsto}}_o C'$ and $p \overset{\tilde{t}}{\longrightarrow}_o p'$ giving the "$\Rightarrow$"-direction.

"$\Leftarrow$": Assume $C \underset{u}{\overset{v}{\longmapsto}}_o C'$ and $p \overset{u}{\longrightarrow}_o p'$. Then by definition $C \underset{t}{\overset{s}{\mapsto\!\!\twoheadrightarrow}} C'$ and $p \overset{t'}{\longrightarrow} p'$ for some $s, t, t' \varepsilon Act^*$ where $\tilde{s} = v$ and $\tilde{t} = \tilde{t'} = u$. Since C is idle-preserving an easy argument shows that if $C \underset{t}{\overset{s}{\mapsto\!\!\twoheadrightarrow}} C'$ and $\tilde{t} = \tilde{t'}$, then for some $s'$ with $\tilde{s'} = \tilde{s}$ also $C \underset{t'}{\overset{s'}{\mapsto\!\!\twoheadrightarrow}} C'$. (Informally this simply means that we can insert and remove 1-transductions as we want when C is idle-preserving). <u>If $s' = \varepsilon$</u> then, since C is non-swallowing, also $t' = \varepsilon$ and $C = C'$, $p = p'$. Thus we have immediately $C[p] \overset{\varepsilon}{\longrightarrow}_o C'[p'] = C[p]$. <u>If $s' \neq \varepsilon$</u> it follows from lemma 3.1-3 that $C[p] \overset{s'}{\longrightarrow} C'[p']$, and hence by definition, $C[p] \overset{v}{\longrightarrow}_o C'[p']$.

□

## 5.2 PARAMETERIZED WEAK BISIMULATION

In this section we shall define an environment-parameterized version of the weak bisimulation equivalence, $\approx$ . We shall show that the results from chapter 2 for the parameterized (strong) bisimulation equivalence generalizes, and in particular that the Characterization Theorem 2.4-20 generalizes.

The definition of parameterized weak bisimulation is rather obvious: we simply apply the existing general definition of parameterized bisimulation (definition 2.2-1) to the derived observational process system $\mathbb{P}^o$ and a similarly derived observational environment system $\mathbb{E}^o = (Env, Act^*_{-1}, \Rightarrow_o)$, i.e. $\Rightarrow_o \subseteq Env \times Act^*_{-1} \times Env$ is derived from $\Rightarrow$ by absorbing any finite sequence of 1-moves (similar to the definition of $\rightarrow_o$).

Thus an <u>$\mathbb{E}$-parameterized weak bisimulation</u> over $\mathbb{P}$ is simply an $\mathbb{E}^o$-parameterized (strong) bisimulation over $\mathbb{P}^o$ . We shall write $p \approx_e q$ whenever $(p,q)$ is contained in the e-component of some $\mathbb{E}$-parameterized weak bisimulation.

With this definition it follows directly from propositions 2.2-5 and 2.2-6 that $\approx_e$ is an equivalence relation and that $\approx \subseteq \approx_e$ for all environments e.

As for parameterized (strong) bisimulation we can in the weak case define a <u>(weak) discrimination ordering</u>, $\precapprox$, on environments based on the relative strength of the corresponding parameterized weak bisimulation equivalence. Thus:

$$e \precapprox f \quad \Leftrightarrow \quad \approx_f \subseteq \approx_e$$

We shall in the following show that $\precapprox$ is fully characterized by the weak simulation ordering, $\lesssim$, under certain image-finiteness conditions. The inclusion $\lesssim \subseteq \sqsubseteq$ follows directly from the generally applicable theorem 2.4-10. This is in contrast to the Main Theorem 2.4-20 which, besides image-finiteness, assumes a certain structure of the process system $\mathbb{P}$. In particular $\mathbb{P}$ must be closed under action-prefixing, where actions are assumed to be <u>atomic</u>. Thus the operational semantics of a.p is fully described by the axiom $a.p \xrightarrow{a} p$. However, for an observational process system actions are <u>not</u> atomic; rather they are strings of atomic actions. As such, the operational semantics of (observational) action-prefixing is given by:

$$u.p \xrightarrow{u} p$$

$$\text{and} \qquad (uv).p \xrightarrow{u} v.p$$

where $u, v \varepsilon Act^{*}_{-1}$. We can therefore not a priori rely on the proof of theorem 2.4-20 to generalize to the weak case. Fortunately, as we shall see in the following, we can still obtain the desired generalization without having to redo the (long) proof of theorem 2.4-20.

Following /Mil80/ we define a process p to be <u>stable</u> iff $p \stackrel{1}{\not\rightarrow}$. If p and all p's derivatives are stable then we call p <u>rigid</u>. A <u>rigid process system</u> is one whose processes are all rigid. Similar definitions are made for environment and environment systems.

Given an environment system $\mathbb{E} = (Env, Act, \Rightarrow)$ we can derive a rigid environment system $@\mathbb{E} = (@Env, Act, \Rightarrow)$ where $@Env = \{@e \mid e \varepsilon Env\}$ and the consumption relation of $@\mathbb{E}$ is defined by:

$$\Rightarrow \; = \; \{(@e, a, @f) \mid a \neq 1 \;\; \& \;\; e \overset{a}{\Rightarrow}_{o} f\}$$

Obviously, this definition makes $@\mathbb{E}$ rigid. More important though is that the observational behaviour of e and @e are closely related.

**Lemma 5.2-1:** For all environments e of $\mathbb{E}$: $e \lesssim @e$ and $@e \lesssim e$. (Note, we are using a simple generalization of $\lesssim$ similar to definition 2.4-6 in order to allow comparisons of environments from different systems).

**Proof:** Prove that the two relations:

$$S_1 = \{(e, @f) \mid e \lesssim f\}$$
$$S_2 = \{(@e, e) \mid e \in Env\}$$

are (generalized) weak simulations using the fact that whenever $e \overset{\varepsilon}{\Rightarrow}_0 e'$ then $e' \lesssim e$. $\square$

Note, that it is not true (in general) that $@e \approx e$; e.g. $e = 1.\mathbb{O} + a.\mathbb{O}$.

For rigid environments and processes it is easily shown that weak simulation (bisimulation, parameterized bisimulation) coincide with the corresponding strong notion:

**Lemma 5.2-2:** For rigid environments e and f of $\mathbb{E}$:

$$e \leqslant f \quad \text{iff} \quad e \lesssim f$$ $\square$

**Lemma 5.2-3:** For rigid processes p and q of $\mathbb{P}$ and rigid environments e of $\mathbb{E}$:

$$p \approx_e q \quad \text{iff} \quad p \sim_e q$$ $\square$

Based on the previous three lemmas we can now prove the desired generalization of theorem 2.4-20.

**Theorem 5.2-4:** If $\mathbb{E}^0$ is an image-finite environment system and $\mathbb{P}$ is closed under action-prefixing and finite sums, then for all environments, e and f, of $\mathbb{E}$:

$$e \sqsubseteq f \quad \Rightarrow \quad e \lesssim f$$

**Proof:** Assume $e \not\lesssim f$. Then from lemma 5.2-1 and lemma 5.2-2 $@e \not\leqslant @f$. Since $\mathbb{E}^0$ is image-finite if and only if $@\mathbb{E}$ is image-finite we can apply the Main Theorem 2.4-20 obtaining processes p and q such that $p \sim_{@f} q$ but $p \not\sim_{@e} q$. From their constructions (p and q are only build from actions which either @f or @e can perform) p and q are obviously rigid. Thus, by lemma 5.2-3, $p \approx_{@f} q$ but $p \not\approx_{@e} q$. Since $\lesssim \subseteq \precsim$ , lemma 5.2-1 finally gives us $p \approx_{f} q$ but $p \not\approx_{e} q$, i.e. $e \not\subseteq f$. $\square$

## 5.3 RELATIONSHIPS BETWEEN (PARAMETERIZED) STRONG AND WEAK BISIMULATION

We devote this section to a study of the relationship between (parameterized) strong and weak bisimulation equivalence. As the main result of the section we shall show that the already known inclusion $\sim \subseteq \approx$ generalizes to the parameterized versions (i.e. $\sim_e \subseteq \approx_e$ for all environments e) under certain conditions. Also, we shall exhibit conditions under which the notions of (parameterized) strong and weak bisimulation equivalence will coincide. Finally, a more practical definition of parameterized weak bisimulation analogous to the alternative definition of weak bisimulation in proposition 5.0-1 is given.

In the previous section we demonstrated how to reduce weak simulation to strong simulation by introducing the notion of a derived rigid transition system. In order to obtain a similar reduction of weak bisimulation to strong bisimulation we shall introduce a slightly different derivation.

First, a process system $\mathbb{P} = (Pr, Act, \rightarrow)$ is said to have the compression property iff the following holds:

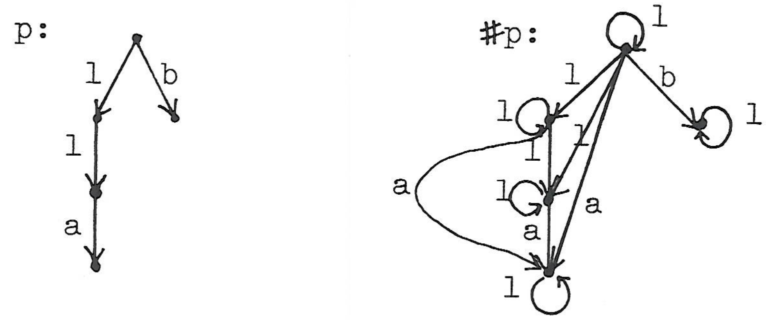
(i) Whenever $a \in Act_{-1}$ and $p \xrightarrow{1^n a 1^m} q$ with $n,m \geq 0$ then also $p \xrightarrow{a} q$.

(ii) Whenever $p \xrightarrow{1^n} q$ with $n \geq 0$ then also $p \xrightarrow{1} q$.

Now, for a process system $\mathbb{P} = (Pr, Act, \rightarrow)$ define the derived process system $\#\mathbb{P} = (\#Pr, Act, \rightarrow)$ where $\#Pr = \{\#p \mid p \in Pr\}$ and the derivation relation of $\#\mathbb{P}$ is defined by:

$$\rightarrow \; = \; \{(\#p, a, \#q) \mid p \xrightarrow{\tilde{a}}_{o} q\}$$

Example 5.3-1: The following two diagrams show the behaviour of a process p and the derived process $\#p$:




□

Proposition 5.3-2: For all process systems $\mathbb{P}$ the derived system $\#\mathbb{P}$ has the compression property.

Proof: Straightforward. □

It is easily shown that the observational behaviours of p and $\#p$ are closely related:

Proposition 5.3-3: For all processes p of $\mathbb{P}$ : $p \approx \#p$.

Proof: Show that the relation $R = \{(p,\#p) \mid p \varepsilon Pr\}$ is a (generalized) weak bisimulation (between $\mathbb{P}$ and $\#\mathbb{P}$ ) using the fact that $\#p \xrightarrow{s}_{o} \#q$ iff $p \xrightarrow{s}_{o} q$ for all $s \varepsilon Act^{*}_{-1}$. □

For process systems with the compression property it is easily shown that the notions of weak and strong bisimulation equivalence coincide:

Proposition 5.3-4: If $\mathbb{P}$ has the compression property then for all processes p and q of $\mathbb{P}$ : $p \approx q$ iff $p \sim q$ .

Proof: Then "$\Leftarrow$"-direction is already wellknown. For the "$\Rightarrow$"- direction show that the relation $R = \{(p,q) \mid p \approx q\}$ is a bisimulation using the compression property of $\mathbb{P}$ . □

From lemma 5.3-2, lemma 5.3-3 and 5.3-4 we can now immediately extract the desired reduction as a corollary:

Corollary 5.3-5: For all processes p and q of $\mathbb{P}$ :
$p \approx q$ iff $\# p \sim \# q$ . □

Let us now try to establish similar results for the parameterized versions of weak and strong bisimulation equivalence. We start by stating the following obvious negative result: it does not in general hold that $\sim_e \subseteq \approx_e$. To see this let:

$$e = 1.a.\mathbb{O}$$
$$p = a.\mathbb{O}$$
$$q = b.\mathbb{O}$$

then $p \sim_e q$ since neither p nor q can perform a 1-action. However, $p \not\approx_e q$ since $e \overset{a}{\Rightarrow}_o$, $p \overset{a}{\rightarrow}_o$ but $q \overset{a}{\not\Rightarrow}_o$. In order to guarantee the inclusion $\sim_e \subseteq \approx_e$ we shall impose restrictions on the operational behaviour of the environment e.

An environment e is (strongly) idle iff $e \overset{1}{\Rightarrow} e$ $(e \overset{1}{\Rightarrow} f \Leftrightarrow e = f)$ and all e's derivatives are also (strongly) idle. A (strongly) idle environment system is one whose environments are all (strongly) idle. Similar definitions are made for processes and process systems. Note, that our notion of idle differs from that in /Mil83/ where a process is idle if it initially can delay arbitrarly. Our notion of idleness requires that the process can delay arbitrarly throughout all of its execution and is as such more closely related to the concept of asynchrony in /Mil83/.

It is easy to prove that the following implications hold, and are strict; i.e. none of the reverse implications hold. We leave the verification of the implications to the reader:

$\mathbb{E}/\mathbb{P}$ is strongly idle $\Rightarrow$

$\mathbb{E}/\mathbb{P}$ has the compression property $\Rightarrow$

$\mathbb{E}/\mathbb{P}$ is idle

Proposition 5.3-6: If $\mathbb{E}$ is strongly idle, then for all processes p and q and environments e:

$$p \sim_e q \quad \Rightarrow \quad p \approx_e q$$

Proof: We show that the Env-indexed family R with $R_e = \{(p,q) \mid p \sim_e q\}$ for $e \in Env$, is a parameterized weak bisimulation using the easily established fact that, whenever e is strongly idle and $e \overset{t}{\Longrightarrow} e'$, then also $e \overset{s}{\Longrightarrow} e'$ for all $s \in Act^*$ such that $\tilde{t} = \tilde{s}$. $\square$

We can relax the strong idleness condition on $\mathbb{E}$ in the lemma above, if we at the same time impose an idleness constraint on the process system $\mathbb{P}$:

Proposition 5.3-7: If $\mathbb{E}$ and $\mathbb{P}$ are idle, then for all processes p and q and environments e:

$$p \sim_e q \quad \Rightarrow \quad p \approx_e q$$

Proof: Similar to the proof of lemma 5.3-6. Use the fact that if $p \overset{s}{\Longrightarrow}_0 p'$ and $e \overset{s}{\Longrightarrow}_0 e'$ then, by the idleness of $\mathbb{P}$ and $\mathbb{E}$, we can find a $t \in Act^*$ such that $\tilde{t} = s$ and $p \overset{t}{\longrightarrow} p'$ and $e \overset{t}{\Longrightarrow} e'$. $\square$

By imposing a slightly stronger constraint on the process system $\mathbb{P}$, we can actually make parameterized weak and strong bisimulation equivalence coincide (giving a parameterized analogue to lemma 5.3-4).

Proposition 5.3-8: If $\mathbb{E}$ is idle and $\mathbb{P}$ has the compression property then for all processes p and q and environments e:

$$p \sim_e q \quad \Leftrightarrow \quad p \approx_e q$$

Proof: "$\Rightarrow$": follows from lemma 5.3-7 since having the compression property implies being idle.
"$\Leftarrow$": Show that the Env-indexed family R with $R_e = \{(p,q) \mid p \approx_e q\}$ for $e \in Env$, is a parameterized bisimulation using the compression property. $\square$

Assuming the environment system $\mathbb{E}$ is idle, it follows from lemma 5.3-3, 5.3-2 and lemma 5.3-8 that:

$$p \approx_e q \quad \Leftrightarrow \quad \#p \sim_e \#q$$

thus giving us a parameterized generalization of corollary 5.3-5. From this observation the following alternative characterization of $\approx_e$ follows directly (using $\#p \xrightarrow{s} \#q$ iff $p \xRightarrow{s}_o q$ for $s \in Act^*_{-1}$).

Definition 5.3-9: Let $\Xi$ be the maximal Env-indexed family of binary relations on Pr such that the following holds: whenever $p \Xi_e q$ and $e \xRightarrow{a} e'$ for some $a \in Act$ then:

$$\text{(i)} \quad p \xRightarrow{\tilde{a}}_o p' \;\Rightarrow\; \exists q'.\; q \xRightarrow{\tilde{a}}_o q' \;\&\; p' \Xi_{e'} q'$$
$$\text{(ii)} \quad q \xRightarrow{\tilde{a}}_o q' \;\Rightarrow\; \exists p'.\; p \xRightarrow{\tilde{a}}_o p' \;\&\; p' \Xi_{e'} q' \quad \square$$

Proposition 5.3-10: Assume $\mathbb{E}$ is idle. Then for all processes p and q and environments e:

$$p \approx_e q \quad \Leftrightarrow \quad p \Xi_e q \quad \square$$

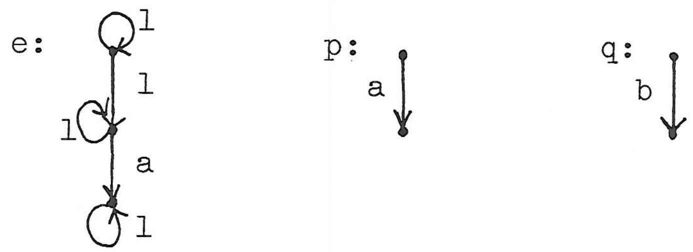
The alternative definition of $\approx_e$ is slightly more practical than the original one (see section 5.2) since we only need to consider single (observable or unobservable) "atomic" moves of environments and, for processes, moves where the observable contents is of length at most 1. However, to get an even simpler definition, analogous to definition 5.0-1, we would like to replace the observational moves in the antecedents of (i) and (ii) of definition 5.3-9 with single "atomic" moves:

Definition 5.3-11: Let $\approxeq$ be the maximal Env-indexed family of binary relations over Pr such that the following holds: whenever $p \approxeq_e q$ and $e \stackrel{a}{\Rightarrow} e'$ for some $a \in Act$ then:

$$\text{(i)} \quad p \xrightarrow{a} p' \Rightarrow \quad \exists q'.\ q \xrightarrow{\tilde{a}}_o q' \ \& \ p' \approxeq_{e'} q'$$

$$\text{(ii)} \quad q \xrightarrow{a} q' \Rightarrow \quad \exists p'.\ p \xrightarrow{\tilde{a}}_o p' \ \& \ p' \approxeq_{e'} q' \qquad \square$$

Since $p \xrightarrow{a} p'$ implies $p \xrightarrow{\tilde{a}}_o p'$ obviously $\underline{\equiv}_e \subseteq \approxeq_e$ always holds. However, the reverse inclusion does not hold in general even if e is idle. To see this let e,p and q be given by the following diagrams:




Then $p \approxeq_e q$ since neither p nor q can perform a 1-move. But it is easily seen that $p \not\underline{\equiv}_e q$. To ensure the inclusion $\approxeq_e \subseteq \underline{\equiv}_e$ we impose a stronger condition on the environment system $\mathbb{E}$:

Proposition 5.3-12: If $\mathbb{E}$ is strongly idle then for all processes p and q and environments e the following are equivalent:

(1) $p \approxeq_e q$

(2) $p \underline{\equiv}_e q$

(3) $p \approx_e q$

Proof: (2)$\Leftrightarrow$ (3) follows from lemma 5.3-10 and (2)$\Rightarrow$ (1) follows from the remarks above. For (1)$\Rightarrow$ (2) show, using the strong idleness of $\mathbb{E}$, that $\approxeq$ satisfies conditions (i) and (ii) of definition 5.3-9 and therefore that $\approxeq \subseteq \underline{\equiv}$ by the maximality of $\underline{\equiv}$. $\square$

## 5.4 CONTEXTS AS OBSERVATIONAL ENVIRONMENT TRANSFORMERS

In this section we shall investigate how contexts transform environments in the weak case, thus generalizing the results from section 3.4. More specifically we shall deal with the following (weak) reduction problem:

> Given a context C, and an (outer) environment, e, we want to find an (inner) environment, f, such that for all processes p and q the following holds:
>
> $$(*) \qquad p \approx_f q \quad \Rightarrow \quad C[p] \approx_e C[q]$$
>
> Preferably, the described (inner) environment, f, should be as small as possible with respect to the weak discrimination ordering, $\precapprox$.

Unfortunately, as we already have demonstrated, since $\approx$ is not preserved by all contexts, it will not in general be possible to find environments, f, satisfying (*). For this reason we shall only deal with the above reduction problem for non-swallowing and asynchronous contexts; i.e. contexts which from section 5.1 are known to preserve $\approx$.

As for the corresponding strong reduction problem in section 3.4 and for similar reasons, we shall consider a modified reduction problem where the condition (*) has been replaced by the following stronger condition on f:

$$(**) \qquad p \approx_f q \quad \Rightarrow \quad [C,p] \approxeq_e [C,q]$$

where $[C,p] \approxeq_e [C,q]$ informally means that $C[p] \approx_e C[q]$ with the context C interacting identically with the two processes p and q. We shall call the weakest environment with respect to $\precapprox$ satisfying (**) for the weakest inner observational environment of e under C, and use the notation $\mathrm{wioe}_{\mathrm{FE}}(C,e)$.

In the following we shall investigate the two questions: "When does $wioe_{\mathbb{E}}(C,e)$ exist ?" and if it does exist: "What is the behaviour of $wioe_{\mathbb{E}}(C,e)$ ?". We shall also deal with the relationship between $wie_{\mathbb{E}}(C,e)$ and $wioe_{\mathbb{E}}(C,e)$. Obviously the answers to these questions will depend upon the environment system, $\mathbb{E}$, in question.

For environment systems, $\mathbb{E}$, closed under a non-swallowing, idle-preserving context system, $\mathbb{C}$, we shall show that there exist an environment f such that for all processes p and q:

$$(***) \qquad p \approx_f q \quad \Leftrightarrow \quad [C,p] \approxeq_e [C,q]$$

provided e is strongly idle. In this case f is a suitable choise for $wioe_{\mathbb{E}}(C,e)$. If $\mathbb{E}$ is not closed under $\mathbb{C}$ we give sufficient conditions which will ensure existence of $wioe_{\mathbb{E}}(C,e)$.

## 5.4.1 Wioe for closed environment systems.

In order to define the parameterized relation $\approxeq$ used in (**) we introduce derived observational versions of the systems $\mathbb{P}$-$\mathbb{C}$ and $\mathbb{E}$-$\mathbb{C}$ defined in section 3.4.1.

Definition 5.4-1: Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ and $\mathbb{C} = (Con, Act_o \times Act_o, \longmapsto)$. Then we define the process system $\mathbb{P}$-$\mathbb{C}^o$ as $(Con\times Pr, Con\times Act^*_{-1}\times Act^*_{-1}, \longrightarrow_o)$ where for all $C,C',C''\varepsilon Con$, $p,p'\varepsilon Pr$ and $u,v \varepsilon Act^*_{-1}$, $\longrightarrow_o$ is defined by:

$$[C,p] \xrightarrow{(C'',v,u)}_o [C',p'] \quad \Leftrightarrow$$
$$\exists s,t\varepsilon Act^*.\ \tilde{s} = v \ \&\ \tilde{t} = u \ \&$$
$$\langle C,p\rangle \xrightarrow{(C'',s,t)} \langle C',p'\rangle$$

where $\longrightarrow$ is the derivation relation of $\mathbb{P}$-$\mathbb{C}$ extended to $(Con\times Pr)\times(Con\times Act^*\times Act^*)\times(Con\times Pr)$ in the obvious way. □

Definition 5.4-2: Let $\mathbb{E} = (Env, Act, \Rightarrow)$ and $\mathbb{C} = (Con, Act_o \times Act_o, \mapsto)$. Then we define the environment system $\mathbb{E}\text{-}\mathbb{C}^o$ as $(Env, Con \times Act^*_{-1} \times Act^*_{-1}, \Rightarrow_o)$ where for all $e, e' \in Env$, $C \in Con$ and $u, v \in Act^*_{-1}$, $\Rightarrow_o$ is defined by:

$$e \xrightarrow{(C,v,u)}_o e' \quad \Leftrightarrow \quad e \overset{v}{\Rightarrow}_o e' \text{ (in } \mathbb{E}^o\text{)} \qquad \square$$

Based on these two definitions we then define:

$$[C,p] \simeq_e [C,q]$$

if and only if there is an $\mathbb{E}\text{-}\mathbb{C}^o$-parameterized bisimulation, R, over $\mathbb{P}\text{-}\mathbb{C}^o$ such that $([C,p],[C,q]) \in R_e$. From the definition of $\mathbb{P}\text{-}\mathbb{C}^o$ we can prove the following useful lemma:

Lemma 5.4-3:

(i) If $[C,p] \xrightarrow{(C'',v,u)}_o [C',p']$
then $C'' = C'$ & $C \overset{v}{\underset{u}{\mapsto}}_o C'$ & $p \xrightarrow{u}_o p'$

(ii) If $C$ is non-swallowing and idle-preserving and $C \overset{v}{\underset{u}{\mapsto}}_o C'$ & $p \xrightarrow{u}_o p'$
then $[C,p] \xrightarrow{(C',u,v)}_o [C',p']$

Proof: (i) Follows directly from definition of $\rightarrow_o$ $(\mathbb{P}\text{-}\mathbb{C}^o)$ and $\rightarrow$ $(\mathbb{P}\text{-}\mathbb{C})$.

(ii) Since $C$ is idle-preserving, we can find $s, t \in Act^*$, $\tilde{s} = v, \tilde{t} = u$ such that $C \overset{s}{\underset{t}{\mapsto}} C'$ and $p \xrightarrow{t} p'$. Since $C$ is non-swallowing $s = \varepsilon$ implies $t = \varepsilon$. Thus, we always have $\langle C,p\rangle \xrightarrow{(C',s,t)} \langle C',p'\rangle$ and therefore $[C,p] \xrightarrow{(C',v,u)}_o [C',p']$. $\square$

Since lemma 5.4-3 (i) always holds, it is obvious that if $[C,p] \simeq_e [C,q]$ then $C$ must interact (observationally) identically with $p$ and $q$. It remains to prove that $[C,p] \simeq_e [C,q]$ also implies $C[p] \approx_e C[q]$.

Theorem 5.4-4: Let $\mathbb{P}$ be closed under a non-swallowing context system $\mathbb{C}$. Then for all contexts, C, processes p and q and strongly idle environments e:

$$[C,p] \gtrsim_e [C,q] \quad \Rightarrow \quad C[p] \approx_e C[q]$$

Proof: Since e is strongly idle, $C[p] \simeq_e C[q]$ if and only if $C[p] \approx_e C[q]$. Thus we show that the family, R, with:

$$R_e = \{(C[p],C[q]) \mid [C,p] \gtrsim_e [C,q]\} \quad ; \text{ e strongly idle}$$

$$R_e = \emptyset \quad ; \text{ otherwise}$$

satisfies the closure-condition in definition 5.3-11. So let e be strongly idle and $(C[p],C[q]) \varepsilon R_e$. Assume $e \overset{b}{\Rightarrow} e'$ and $C[p] \xrightarrow{b} r$. Since $\mathbb{P}$ is closed under $\mathbb{C}$, $C \overset{b}{\underset{u}{\models\!\!\!\gg}} C'$, $p \xrightarrow{u} p'$ for some $C', p'$ and $u \varepsilon Act^*$ with $r = C'[p']$. By definition of $\mathbb{P}$-$\mathbb{C}$ then $\langle C,p\rangle \xrightarrow{(C',b,u)} \langle C',p'\rangle$ and thus $[C,p] \xrightarrow{(C',\tilde{b},\tilde{u})}_o [C',p']$. By definition of $\mathbb{E}$-$\mathbb{C}^o$ obviously $e \xrightarrow{(C',\tilde{b},\tilde{u})}_o e'$. Hence, since $[C,p] \gtrsim_e [C,q]$, $[C,q] \xrightarrow{(C',\tilde{b},\tilde{u})}_o [C',q']$ with $[C',p'] \gtrsim_{e'} [C',q']$ for some $q'$. By definition of $\rightarrow_o$, $\langle C,q\rangle \xrightarrow{(C',s,t)} \langle C',q'\rangle$ for some $s,t \varepsilon Act^*$ with $\tilde{s}=\tilde{b}$ and $\tilde{t}=\tilde{u}$. Then by definition of $\rightarrow$ of $\mathbb{P}$-$\mathbb{C}$, $C \overset{s}{\underset{t}{\models\!\!\!\gg}} C'$ and $q \xrightarrow{t} q'$. Since C is non-swallowing, $s=\varepsilon$ implies $t=\varepsilon$, and thus always $C[q] \xrightarrow{s} C'[q']$ and hence $C[q] \xrightarrow{\tilde{b}}_o C'[q']$, which is a matching move. □

Theorem 5.4-5: Let $\mathbb{P}$ and $\mathbb{E}$ be closed under an asynchronous context system $\mathbb{C}$. Then for all contexts C, processes p and q and strongly idle environments e the following holds:

$$(1) \quad p \simeq_{e[C]} q \quad \Rightarrow \quad [C,p] \gtrsim_e [C,q]$$

If $\mathbb{C}$ moreover is non-swallowing and idle-preserving, then also:

$$(2) \quad [C,p] \gtrsim_e [C,q] \quad \Rightarrow \quad p \simeq_{e[C]} q$$

Proof:

(1): We show that R with:

$$R_e = \{([C,p];[C,q]) \mid p \approx_{e[C]} q\} \quad ; \text{ e strongly idle}$$

$$R_e = \emptyset \quad ; \text{ otherwise}$$

is an $\mathbb{E}$-$\mathbb{P}^o$-parameterized bisimulation. So let e be strongly idle and let $([C,p];[C,q]) \varepsilon R_e$. Assume $e \xrightarrow{(C',v,u)}_o e'$ and $[C,p] \xrightarrow{(C',v,u)}_o [C',p']$. By definitions and strong idleness of e, $e \overset{s}{\Rightarrow} e'$, $C \vdash\!\frac{s}{t}\!\!\gg C'$ and $p \overset{t}{\Rightarrow} p'$ for some $s,t \varepsilon Act^*$ with $\tilde{s}=v$ and $\tilde{t}=u$.

If $t \neq \varepsilon$: then since $\mathbb{E}$ is closed under $\mathbb{P}$, $e[C] \overset{t}{\Rightarrow} e'[C']$ or $e[C] \overset{\tilde{t}}{\Rightarrow}_o e'[C']$. Thus, $q \overset{\tilde{t}}{\Rightarrow}_o q'$ with $p' \approx_{e'[C']} q'$ for some $q'$, i.e. $q \overset{t'}{\Rightarrow} q'$ for some $t' \varepsilon Act^*$ with $\tilde{t'}=\tilde{t}=u$. Since C is asynchron and $t \neq \varepsilon$ it is easy to show that for some $s'$ with $\tilde{s'}=\tilde{s}$ also $C \vdash\!\frac{s'}{t'}\!\!\gg C'$. Hence $\langle C,q\rangle \xrightarrow{(C',s',t')} \langle C',q'\rangle$ and finally $[C,q] \xrightarrow{(C',v,u)}_o [C',q']$ which is the matching move.

If $t=\varepsilon$: then $p=p'$ and $e'[C'] \leqslant e[C]$ (implying $e'[C'] \lessapprox e[C]$). Hence also $p \approx_{e'[C']} q$. Obviously $q \overset{\varepsilon}{\Rightarrow} q$, so $\langle C,q\rangle \xrightarrow{(C',s,\varepsilon)} \langle C',q\rangle$ and thus $[C,q] \xrightarrow{(C',v,\varepsilon)}_o [C',q]$ which is the matching move.

(2): We show that R with:

$$R_f = \{(p,q) \mid \exists C. \exists E.\ f = e[C] \ \&\ [C,p] \simeq_e [C\ q]\}$$

is an $\mathbb{E}$- parameterized weak bisimulation.
So let $(p,q)\varepsilon R_{e[C]}$, $e[C] \overset{u}{\Rightarrow}_o f$ and $p \overset{u}{\Rightarrow}_o p'$. Then for some $C',e'$ and $v \varepsilon Act^*$, $e \overset{v}{\Rightarrow}_o e'$, $C \vdash\!\frac{v}{u}\!\!\gg_o C'$ with $f=e'[C']$. Since $\mathbb{C}$ is assumed non-swallowing and idle-preserving we can apply lemma 5.4-3 (ii) giving:

$$[C,p] \xrightarrow{(C',v,u)}_o [C',p']$$

Obviously, $e \xrightarrow{(C',v,u)}_o e'$ in $\mathbb{E}$-$\mathbb{C}^o$, so since $[C,p] \simeq_e [C,q]$:

$$[C,q] \xrightarrow{(C',v,u)}_o [C',q']$$

with $[C',p'] \simeq_{e'} [C',q']$. By lemma 5.4-3 (i) we then conclude that $q \overset{u}{\Rightarrow}_o q'$ which is the matching move. □

Corollary 5.4-6: Let $\mathbb{P}$ and $\mathbb{E}$ be closed under a non-swallowing and asynchronous context system $\mathbb{C}$. Then for all contexts C, processes p and q and strongly idle environments e the following holds:

$$p \approx_{e[C]} q \quad \Rightarrow \quad C[p] \approx_e C[q]$$

Proof: Direct from theorem 5.4-4 and theorem 5.4-5. □

Corollary 5.4-7: If $\mathbb{E}$ is closed under $\mathbb{C}$ and $\mathbb{C}$ is non-swallowing and idle-preserving then for all contexts C and strongly idle environments e, we can define:

$$\mathrm{wioe}_{\mathbb{E}}(C,e) = e[C]$$

Proof: Direct from theorem 5.4-5 (2). □

Corollary 5.4-7 also gives us information about the relationship between $\mathrm{wie}_{\mathbb{E}}(C,e)$ and $\mathrm{wioe}_{\mathbb{E}}(C,e)$: if $\mathbb{C}$ is non-swallowing and idle-preserving and e is strongly idle then $\mathrm{wie}_{\mathbb{E}}(C,e) \simeq \mathrm{wioe}_{\mathbb{E}}(C,e)$.

One thing that might worry the reader slightly, is that most of our results for the weak reduction problem requires the environment, e, to be strongly idle: a seemingly strong requirement. However, any environment can be transformed into a $\lessapprox$-equivalent (and thus $\sqsubseteq$-equivalent) strongly idle one, for which our results applies: let $\mathbb{E}$ be any environment system. Then $\#(@\mathbb{E})$ is strongly idle (an easy argument shows that if e is rigid then $\#e$ is strongly idle) and from lemmas 5.2-1 and 5.3-3 it follows that $e \lessgtr \#(@e)$ for all environments e.

## 5.4.2 Wioe for general environment systems.

In the previous section we dealt with the weak reduction problem for environment systems, $\mathbb{E}$, closed under the context system, $\mathbb{C}$.

In this section we shall give solutions to the problem for general strongly idle environment systems (not necessarily closed under $\mathbb{C}$). We shall offer (sufficient) conditions which will ensure the existence of $\mathrm{wioe}_{\mathbb{E}}(C,e)$ in such cases.

So let $\mathbb{E}$ be a (general) strongly idle environment system and let $\mathbb{C}$ be a non-swallowing, asynchronous context system. The weak reduction problem is for a given context $C$ and environment $e$ to find an environment $f$ such that:

$$(*) \qquad p \approx_f q \quad \Rightarrow \quad C[p] \approx_e C[q]$$

for all processes $p$ and $q$. Since $\mathbb{E}$ is not (necessarily) closed under $\mathbb{C}$ the results from previous section cannot be used. However, we can apply the following simple strategy: first close $\mathbb{E}$ under $\mathbb{C}$ (definition 3.1-12) giving the (closed) extension $\mathbb{E}_{\mathbb{C}}$. Then from the results of the previous section (Corollary 5.4-6) we know that

$$(**) \qquad p \approx_{e[C]} q \quad \Rightarrow \quad C[p] \approx_e C[q]$$

for all processes $p$ and $q$. If $\mathbb{C}$ moreover is idle-preserving, we know in fact that $e[C]$ is $\mathrm{wioe}_{\mathbb{E}_{\mathbb{C}}}(C,e)$

Now, assume we can find a smallest environment, $f$, in $\mathbb{E}$ with respect to $\leqslant$, such that $e[C] \leqslant f$. We shall use the notation $\underline{\mathrm{boa}_{\mathbb{E}}(C,e)}$ (<u>b</u>est <u>o</u>bservational <u>a</u>pproximation) for this environment. Since $\lesssim \subseteq \underset{\sim}{\sqsubseteq}$, $\mathrm{boa}_{\mathbb{E}}(C,e)$ would obviously be a solution to the weak reduction problem, i.e.:

$$(***) \qquad p \approx_{\mathrm{boa}_{\mathbb{E}}(C,e)} q \quad \Rightarrow \quad C[p] \approx_e C[q]$$

If moreover $\mathbb{C}$ is idle-preserving, and $(\mathbb{E}_{\mathbb{C}})^{o}$ is image-finite we can from the Generalized Main Theorem (5.2-4) simply take $\mathrm{wioe}_{\mathbb{E}}(C,e)$ to be $\mathrm{boa}_{\mathbb{E}}(C,e)$.

An easy argument shows that for strongly idle environments, $f$, $e \lesssim f$ if and only if $e \leqslant f$ (irrespective of whether $\underline{e}$ is strongly idle or not). Thus, $boa_{\mathbb{E}}(C,e)$ exists if and only if $ba_{\mathbb{E}}(C,e)$ does (see section 3.4.2 for definition of $ba_{\mathbb{E}}(C,e)$ ), in which case $boa_{\mathbb{E}}(C,e) \simeq ba_{\mathbb{E}}(C,e)$.

The system of language environments, $\mathbb{L}$, (see definition 2.2-11) is obviously not strongly idle and falls as such outside the scope of our results. We therefore introduce a new system, $\mathbb{L}_{si}$, of strongly idle language environments consisting of languages over $Act_{-1}$.

<u>Definition 5.4-8:</u> $\mathbb{L}_{si} = (\mathcal{P}(Act^*_{-1}), Act, \Rightarrow)$ is the environment system, where $\Rightarrow$ is the smallest relation satisfying for all $L \subseteq Act^*_{-1}$ and $a \in Act_{-1}$:

(i) $L \overset{1}{\Rightarrow} L$

(ii) $\partial L/\partial a \neq \emptyset \;\Rightarrow\; L \overset{a}{\Rightarrow} \partial L/\partial a$ $\square$

$\mathbb{L}_{si}$ is obviously strongly idle. Also $\mathbb{L}_{si}$ can be seen as a subsystem of $\mathbb{L}$. Let $\overline{\phantom{x}} : Act^* \rightarrow Act^*$ be defined by:

$$\overline{\varepsilon} = 1$$

$$\overline{a_1 \ldots a_n} = 1^* a_1 1^* \ldots 1^* a_n 1^* \quad ; n \geq 1$$

with the natural extension to sets of strings. Then for all $L \subseteq Act^*_{-1}$, the behaviour of L in $\mathbb{L}_{si}$ is strong equivalent to the behaviour of $\overline{L}$ in $\mathbb{L}$.

<u>Lemma 5.4-9:</u> For all environments e and all environments L of $\mathbb{L}_{si}$ the following are equivalent:

(i) $e \lessapprox L$

(ii) $e \lesssim L$

(iii) $D(e) \subseteq \overline{L^p}$

(iv) $\widetilde{D(e)} \subseteq L^p$

where $D(e) = \{ u \in Act^* \mid e \overset{u}{\Rightarrow} \}$ and $\sim$ cancels all occurrences of 1 in a string.

<u>Proof:</u> (i)$\Leftrightarrow$(ii) follows from the strong idleness of L. (ii)$\Leftrightarrow$(iii) Write $L{:}\mathbb{L}_{si}$ resp. $L{:}\mathbb{L}$ for L being viewed as an environment of $\mathbb{L}_{si}$ resp. $\mathbb{L}$. From the remark above the lemma $L{:}\mathbb{L}_{si} \sim \bar{L}{:}\mathbb{L}$ and hence $e \lesssim L{:}\mathbb{L}_{si}$ iff $e \lesssim \bar{L}{:}\mathbb{L}$. From section 3.4.2 $e \lesssim \bar{L}{:}\mathbb{L}$ is known to hold iff $D(e) \subseteq \bar{L}^p = \overline{L^p}$. (iii)$\Leftrightarrow$(iv) follows directly from $\widetilde{(\bar{L})} = L$ for all $L \subseteq Act^*_{-1}$ and $L \subseteq \overline{(\tilde{L})}$ for $L \subseteq Act^*$. $\square$

It is easily seen that for an $\mathbb{L}_{si}$-environment, M, $D(M) = \overline{M^p}$. Thus, it follows as a corollary from the above lemma that for all $\mathbb{L}_{si}$- environments L and M:

$$L \lesssim M \quad \Leftrightarrow \quad L^p \subseteq M^p$$

Hence, for any $\mathbb{L}_{si}$- environment L and context C it follows immediately, that:

$$boa_{\mathbb{L}_{si}}(C,L) \lesssim D(\widetilde{L[C]})$$

Using lemma 3.1-10 we have:

$$\begin{aligned}
D(\widetilde{L[C]}) &= \{\tilde{s} \mid s \varepsilon Act^* \;\&\; L[C] \overset{s}{\Rightarrow}\} \\
&= \{\varepsilon\} \cup \{\tilde{s} \mid s \varepsilon Act^+ \;\&\; \exists t \varepsilon Act^*.\; L \overset{t}{\Rightarrow} \;\&\; C \overset{t}{\underset{s}{\longmapsto}} \} \\
&\overset{+}{=} \{u \varepsilon Act^*_{-1} \mid \exists v \varepsilon Act^*.\; L \overset{v}{\Rightarrow}_o \;\&\; C \overset{v}{\underset{u}{\longmapsto}}_o \} \\
&\overset{++}{=} \{u \varepsilon Act^*_{-1} \mid \exists v \varepsilon L^p.\; C \overset{v}{\underset{u}{\longmapsto}}_o \}
\end{aligned}$$

where + holds since L is strongly idle, and ++ holds since $L \overset{v}{\Rightarrow}_o$ iff $v \varepsilon L^p$. Thus we can simply define:

<u>Definition 5.4-10:</u>

$$boa_{\mathbb{L}_{si}}(C,L) = \{u \varepsilon Act^*_{-1} \mid \exists v \varepsilon L^p.\; C \overset{v}{\underset{u}{\longmapsto}}_o \} \qquad \square$$

From this definition the following laws can be derived easily:

Proposition 5.4-11: For CCS-contexts the following holds:

(i) $\mathrm{boa}_{\mathbb{L}_{si}}(p,L) = \{\varepsilon\}$ ; $L \neq \emptyset$

(ii) $\mathrm{boa}_{\mathbb{L}_{si}}([\,],L) = L^p$

(iii) $\mathrm{boa}_{\mathbb{L}_{si}}(1.[\,],L) = L^p$

(iv) $\mathrm{boa}_{\mathbb{L}_{si}}(a.[\,],L) = (\partial L/\partial a)^p$ ; $a \neq 1$

(v) $\mathrm{boa}_{\mathbb{L}_{si}}(p\,|\,[\,],L) = \{u \varepsilon \mathrm{Act}^*_{-1} \mid (\widetilde{u \# \widetilde{D(p)}}) \cap L^p \neq \emptyset\}$

(vi) $\mathrm{boa}_{\mathbb{L}_{si}}([\,]\lceil S,L) = L^p \cap S^*$ ; $1 \varepsilon S$

(vii) $\mathrm{boa}_{\mathbb{L}_{si}}([\,][\Phi],L) = \Phi^{-1}(L^p)$ ; $\Phi(1) = 1$

(viii) $\mathrm{boa}_{\mathbb{L}_{si}}(C \circ D,L) \subseteq \mathrm{boa}_{\mathbb{L}_{si}}(D, \mathrm{boa}_{\mathbb{L}_{si}}(C,L))$

with "=" if C and D are idle-preserving.

Proof: We only prove the slightly more difficult (v), leaving the rest to the reader. From the discussion above and proposition 3.2-6 (vi) we have:

$$\begin{aligned}
\mathrm{boa}_{\mathbb{L}_{si}}(p|[\,],L) &= \{\varepsilon\} \cup \{\tilde{s} \mid s \varepsilon \mathrm{Act}^+ \;\&\; \exists t \varepsilon \overline{L^p}.\; p|[\,] \overset{t}{\underset{s}{\mapsto\!\!\!\Rightarrow}} \} \\
&= \{\tilde{s} \mid s \varepsilon \mathrm{Act}^* . \exists t \varepsilon \overline{L^p}.\; t \varepsilon (s \# D(p))\} \\
&= \{\tilde{s} \mid s \varepsilon \mathrm{Act}^* .\; (s \# D(p)) \cap \overline{L^p} \neq \emptyset\} \\
&= \{\tilde{s} \mid s \varepsilon \mathrm{Act}^* .\; (\widetilde{s \# D(p)}) \cap L^p \neq \emptyset\} \\
&\overset{+}{=} \{u \varepsilon \mathrm{Act}^*_{-1} \mid (\widetilde{u \# \widetilde{D(p)}}) \cap L^p \neq \emptyset\}
\end{aligned}$$

where + is justified by the equation $(\widetilde{s \# t}) = (\widetilde{\tilde{s} \# \tilde{t}})$. □

## 5.5 A SIMPLE SCHEDULER

In this section we shall prove the correctness of a simple scheduler using the parameterized weak bisimulation equivalence.

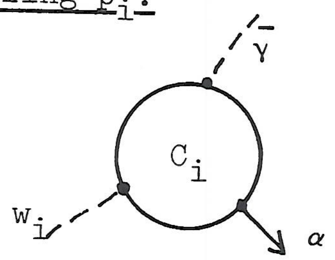
The scheduling problem we study is a simplified version of the scheduling problem in /Mil79B,Mil80/: we simply want to design a scheduler $S_n$ which will signal a set $\{p_1,\ldots,p_n\}$ of n agents in rotation starting with the agent $p_1$.

Suppose that $p_i$ is expecting to be signalled at label $w_i$. Then our scheduler should simply satisfy the constraint:

(1) $$\boxed{S_n \approx w_1.w_2.\ \ldots\ .w_n.S_n}$$

We could of course easily write a CCS-process with the above property directly, e.g. the process $\mu x.w_1.\ \ldots\ .w_n.x$ would suffice. However, we prefer to build $S_n$ as a ring of n identical cyclic cells with each cell in control of one agent.

The cell controlling $p_i$:




The cell's behaviour consists of an endless repetition of the following:

(i) Be enabled at $\alpha$ by the preceding cell.

(ii) Signal the waiting agent $p_i$ at $w_i$.

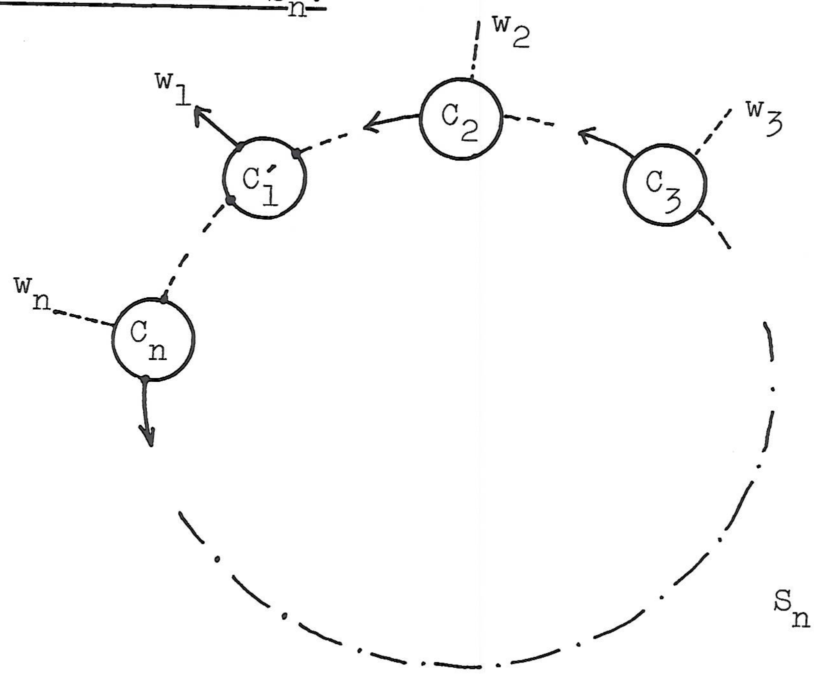
(iii) Enable the successor cell at $\bar{\gamma}$.

Thus we can simply define:

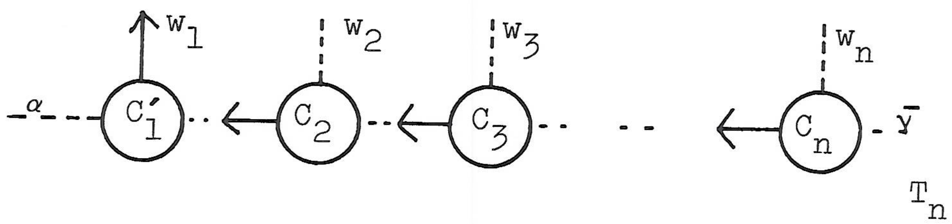
(2) $$\boxed{C_i = \mu x.(\alpha.w_i.\bar{\gamma}.x)}$$

The scheduler is now built as a ring from the cells $C_1,\ldots,C_n$ with the first cell $C_1$ being in state (ii) (in order for the scheduler to start).

<u>The scheduler $S_n$:</u>




In order to define $S_n$ we consider the following <u>rectified version</u> $T_n$ of $S_n$:




$T_n$ can be defined inductively on n as:

(3) $$\boxed{\begin{array}{l} T_1 = C_1' = w_1.\bar{\gamma}.C_1 \\ T_n = [T_{n-1}[\gamma \mapsto \delta] \mid C_n[\alpha \mapsto \delta]]\backslash\delta \quad ; \; n \geq 2 \end{array}}$$

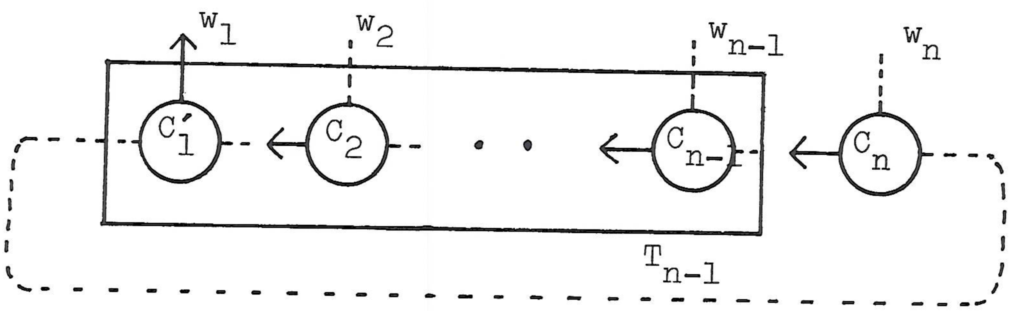
where for $a_1 \dots a_k \varepsilon Act_{-1}$ and $b_1 \dots b_k \varepsilon Act$ $(a_1 \mapsto b_1, \dots, a_k \mapsto b_k)$ is the renaming map $Act \longrightarrow Act$ defined by:

$$(a_1 \mapsto b_1, \dots, a_k \mapsto b_k)\ a \ = \begin{cases} b_i & ;\ 1 \leqslant i \leqslant k \wedge a = a_i \\ \overline{b_i} & ;\ 1 \leqslant i \leqslant k \wedge a = \overline{a_i} \\ a & ;\ \text{otherwise} \end{cases}$$

(In /Mil80/ the notation $b_1/a_1 \dots b_k/a_k$ is used for this map). Note, since $a_1 \dots a_k \varepsilon Act_{-1}$ we have $(a_1 \mapsto b_1, \dots, a_k \mapsto b_k)\ 1 = 1$. Thus the associated renaming context is idle-preserving.

For $a \varepsilon Act_{-1}$, $[\,]\backslash a$ is an abbreviation for the CCS-context $[\,]\restriction S - \{a, \bar{a}\}$. Since $1 \varepsilon S - \{a, \bar{a}\}$, $[\,]\backslash a$ is an idle-preserving context.

For n>1 we can then construct $S_n$ from $T_{n-1}$ and $C_n$ as illustrated below:




Formally we define:

(4) $$\boxed{S_n \ = \{\, T_{n-1}\left[\begin{smallmatrix}\alpha \mapsto \varrho \\ \gamma \mapsto \delta\end{smallmatrix}\right] \mid C_n\left[\begin{smallmatrix}\alpha \mapsto \delta \\ \gamma \mapsto \varrho\end{smallmatrix}\right] \}\backslash\delta\backslash\varrho}$$

Based on this definition of $S_n$ it is possible to prove directly, using the weak bisimulation proof technique, that the constraint (1) is satisfied. A defect with this direct approach is that it is not based on an analysis of any subsystems. This defect may not be serious for the present simple example, but for larger systems such a strategy would suffer a combinatorial explosion. In order for our proof techniques to be relevant for large (realistic) examples it is imperative that we can reason about the system in terms of its subsystems. For this reason we prefer to prove the correctness of $S_n$ inductively (on n). The overall effort in proving the correctness inductively will for this simple example actually increase, but it illustrates a technique which seems useful for larger systems. Further evidence of this potential usefulness for larger systems has recently been given by Robin Milner, who has successfully applied the (parameterized bisimulation) techniques of this thesis to the Alternating Bit Protocol.

Unfortunately $S_n$ does not lend itself to such an inductive proof since $S_{n-1}$ is <u>not</u> a substructure of $S_n$. An inductive proof seems much more likely to succeed for the rectified version $T_n$ since obviously $T_{n-1}$ is a substructure of $T_n$. But what should we prove (inductively) about $T_n$ ? Ideally we would like to prove $T_n \approx w_1.w_2. \dots .w_n.\bar{\gamma}.\alpha.T_n$. But this is not a valid equivalence: after the occurrence of $w_1$ $T_n$ is free to perform $\alpha$ at any time. In fact the full behaviour of $T_n$ is extremely complicated. However, we are only interested in the behaviour of $T_n$ as a component of the scheduler $S_{n+1}$, and it seems that in this context the behaviour of $T_n$ is indeed captured by the above equation. In the following we shall prove that there is a strongly idle language environment L such that:

(5) $$\boxed{T_n \approx_L w_1.w_2. \dots .w_n.\bar{\gamma}.\alpha.T_n \qquad (n \geq 1)}$$

and

(6) $$\boxed{boa_{\mathbb{L}_{si}}(TC_n, Act_{-1}^{*}) \subseteq L^p \qquad (n \geq 2)}$$

where

(7) $$\boxed{TC_n = \{[\,][^{\alpha \mapsto \varrho}_{\gamma \mapsto \delta}] \mid C_n[^{\alpha \mapsto \delta}_{\gamma \mapsto \varrho}]\}\backslash\delta\backslash\varrho \qquad (n \geq 2)}$$

From (5) and (6) it follows that $T_n$ and $w_1.w_2. \dots .w_n.\bar{\gamma}.\alpha.T_n$

are substitutive in $TC_{n+1}$. ($TC_{n+1}$ is idle-preserving and the reduction is therefore valid, see section 5.4.2). Thus for $n \geq 2$:

$$
\begin{aligned}
S_n &= \left\{ T_{n-1}\begin{bmatrix}\alpha \mapsto \varrho \\ \gamma \mapsto \delta\end{bmatrix} \mid C_n\begin{bmatrix}\alpha \mapsto \delta \\ \gamma \mapsto \varrho\end{bmatrix} \right\} \backslash \delta \backslash \varrho \\
&\approx \left\{ (w_1.w_2. \;\ldots\; .w_{n-1}.\bar{\gamma}.\alpha.T_{n-1})\begin{bmatrix}\alpha \mapsto \varrho \\ \gamma \mapsto \delta\end{bmatrix} \mid \right. \\
&\qquad\qquad \left. C_n\begin{bmatrix}\alpha \mapsto \delta \\ \gamma \mapsto \varrho\end{bmatrix} \right\} \backslash \delta \backslash \varrho
\end{aligned}
$$

Using the fixed point rule (R2 in the proof system $\underline{S}_{rr}$ of chapter 4), the Expansion Theorem (see /Mil80/ theorem 5.8), some simple laws for renaming, and the fact that parallel CCS-contexts preserve $\approx$ (see section 5.1) the following is easily established:

$$
\begin{aligned}
&\approx \left\{ w_1.w_2. \;\ldots\; .w_{n-1}.\bar{\delta}.\varrho.(T_{n-1}\begin{bmatrix}\alpha \mapsto \varrho \\ \gamma \mapsto \delta\end{bmatrix}) \mid \right. \\
&\qquad\qquad \left. \delta.w_n.\bar{\varrho}.(C_n\begin{bmatrix}\alpha \mapsto \delta \\ \gamma \mapsto \varrho\end{bmatrix}) \right\} \backslash \delta \backslash \varrho \\
&\approx w_1.w_2. \;\ldots\; .w_{n-1}.w_n.\left\{ (T_{n-1}\begin{bmatrix}\alpha \mapsto \varrho \\ \gamma \mapsto \delta\end{bmatrix} \mid \right. \\
&\qquad\qquad \left. C_n\begin{bmatrix}\alpha \mapsto \delta \\ \gamma \mapsto \varrho\end{bmatrix}) \backslash \delta \backslash \varrho \right\} \\
&= w_1.w_2. \;\ldots\; .w_n.S_n
\end{aligned}
$$

This verifies the correctness condition (1). It remains to exhibit the strongly idle language environment L and prove that it has the two required properties (5) and (6).

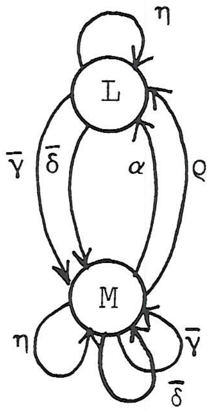
The unparameterized version of (5) fails to hold basically because $T_n$ can perform $\alpha$-actions in a very undisciplined manner: after each $w_1$-action $T_n$ will always be ready (at least after some 1-moves) to perform an $\alpha$. However, when $T_n$ is executed in the context $TC_{n+1}$ no $\alpha$ will occur before the first $\bar{\gamma}$ and before any new $\alpha$-action can occure $T_n$ must perform a $\bar{\gamma}$ first. This information about $TC_{n+1}$ is captured by the following strongly idle language environment L (we are using the standard notation for regular languages):

$$L = \left[\Delta^{*}_{-\alpha\gamma\delta\varrho} \cdot (\bar{\gamma} + \bar{\delta}) \cdot (\Delta_{-\alpha\gamma\delta\varrho} + \bar{\gamma} + \bar{\delta})^{*} \cdot (\alpha + \varrho)\right]^{*}$$

where for $a_1 \ldots a_k \varepsilon Act_{-1}$:

$$\Delta_{-a_1 \ldots a_k} = Act_{-1} - \{a_1 \ldots a_k, \bar{a}_1 \ldots \bar{a}_k\}$$

The behaviour of L is given by the following diagram:




where $\eta \varepsilon \Delta_{-\alpha\gamma\varrho\delta} \cup \{1\}$. From the diagram, and since $T_n$ cannot perform $\varrho$ or $\bar{\delta}$ actions, it follows that $T_n$'s undiciplined usages of $\alpha$ mentioned above are prohibited in L.

Let us first verify (6) using the laws for $boa_{\mathbb{L}_{si}}$ in proposition 5.4-11. Since $TC_n$ is built from idle preserving contexts we can decompose the calculation of $boa_{\mathbb{L}_{si}}(TC_n, Act_{-1})$ into stages. Using proposition 5.4-11 (vi) we have:

$$boa_{\mathbb{L}_{si}}([\,]\backslash\delta\backslash\varrho, Act^{*}_{-1}) = \Delta^{*}_{-\delta\varrho}$$

since $D(C_n\left[\begin{smallmatrix}\alpha\mapsto\delta\\ \gamma\mapsto\varrho\end{smallmatrix}\right]) = (\delta . w_n . \bar{\varrho})^{*p}$ we conclude from proposition 5.4-11 (v):

$$boa_{\mathbb{L}_{si}}([\,] \mid C_n\left[\begin{smallmatrix}\alpha\mapsto\delta\\ \gamma\mapsto\varrho\end{smallmatrix}\right], \Delta^{*}_{-\delta\varrho})$$

$$= \{u \varepsilon Act^{*}_{-1} \mid u \# (\delta . w_n . \bar{\varrho})^{*p} \cap \Delta^{*}_{-\delta\varrho} \neq \emptyset\}$$

$$= (\Delta^{*}_{-\delta\varrho} . \bar{\delta} . \Delta^{*}_{-\delta\varrho} . \varrho)^{*p} = N$$

From proposition 5.4-11 (vii) it follows that:

$$\text{boa}_{\mathbb{L}_{si}}([\ ][{}^{\alpha\mapsto\varrho}_{\gamma\mapsto\delta}]\ ,N)$$

$$= (\ \alpha\mapsto\varrho\ ,\ \gamma\mapsto\delta\ )^{-1}\ N$$

$$= (\Delta^*_{\underset{\alpha\gamma}{-\delta\varrho}}.(\bar{\gamma}+\bar{\delta}).\Delta^*_{\underset{\alpha\gamma}{-\delta\varrho}}.(\alpha+\varrho))^{*\ p}$$

Combining these three calculations we have:

$$\text{boa}_{\mathbb{L}_{si}}(TC_n,Act^*_{-1})$$

$$= (\Delta^*_{\underset{\alpha\gamma}{-\delta\varrho}}.(\bar{\gamma}+\bar{\delta}).\Delta^*_{\underset{\alpha\gamma}{-\delta\varrho}}.(\alpha+\varrho))^{*\ p}$$

$$\subseteq L^p$$

Thus condition (6) is satisfied. Let us now prove that (5) is satisfied by induction on n. For <u>n=1</u> we have immediately:

$$\begin{aligned} T_1 &= w_1.\bar{\gamma}.C_1 \\ &\approx w_1.\bar{\gamma}.(\alpha.w_1.\bar{\gamma}.C_1) \\ &= w_1.\bar{\gamma}.\alpha.T_1 \end{aligned}$$

For the induction step we shall use the following property of L.

(8)

$$\boxed{\begin{array}{ll} \text{boa}_{\mathbb{L}_{si}}(TD_n,L)\subseteq L^p & (n\geqslant 2) \\ \text{where} & \\ TD_n = \{\, [\ ][\gamma\mapsto\delta]\ \mid\ C_n[\alpha\mapsto\delta]\,\}\backslash\delta & \\ & (n\geqslant 2) \end{array}}$$

Now, assume (5) holds for $1\leqslant k<n$. Then, using property (8) we know that $T_{n-1}$ and $w_1.\ \ldots\ .w_{n-1}.\bar{\gamma}.\alpha.T_{n-1}$ are substitutive (up to L) in $TD_n$. Thus:

$$
\begin{aligned}
T_n &= \{T_{n-1}[\gamma \mapsto \delta] \mid C_n[\alpha \mapsto \delta]\}\backslash\delta \\
&\approx_L \quad (w_1 \cdot \,..\, \cdot w_{n-1} \cdot \bar{\gamma} \cdot \alpha \cdot T_{n-1})[\gamma \mapsto \delta] \mid \\
&\qquad\qquad\qquad C_n[\alpha \mapsto \delta]\}\backslash\delta \\
&\approx \{w_1 \cdot \,..\, \cdot w_{n-1} \cdot \bar{\delta} \cdot \alpha \cdot (T_{n-1}[\gamma \mapsto \delta]) \mid \\
&\qquad\qquad\qquad \delta \cdot w_n \cdot \bar{\gamma} \cdot (C_n[\alpha \mapsto \delta])\}\backslash\delta \\
&\approx \quad w_1 \cdot \,..\, \cdot w_{n-1}(\{\alpha \cdot (T_{n-1}[\gamma \mapsto \delta]) \mid \\
&\qquad\qquad\qquad w_n \cdot \bar{\gamma} \cdot (C_n[\alpha \mapsto \delta])\}\backslash\delta)
\end{aligned}
$$

Since $boa_{\mathbb{L}_{si}}(w_1 \cdot \,..\, \cdot w_{n-1} \cdot [\,], L) = L$, $boa_{\mathbb{L}_{si}}(w_1 \cdot \,..\, \cdot w_n \cdot [\,], L) = L$ and $L \overset{g}{\not\Rightarrow}$ we conclude further:

$$\approx_L \; w_1 \cdot \,...\, \cdot w_n \cdot \bar{\gamma} \cdot \alpha \cdot T_n$$

The last remaining proof obligation is the verification of (8). Again we can use the laws for $boa_{\mathbb{L}_{si}}$ from proposition 5.4-11. However, we prefer this time to appeal directly to the definition:

$$boa_{\mathbb{L}_{si}}(C,L) = \widetilde{D(L[C])}$$

Unfortunately, to determine the behaviour of $L[C]$ directly from definition 3.1-9 could prove quite a lengthy process since we are required to consider how $L$ can undergo strings of actions. However, the process can be shortened considerably by the following lemma:

Lemma 5.5-1: Let $\mathbb{E} = (Env, Act, \Rightarrow)$ be an environment system closed under a context system $\mathbb{C} = (Con, Act_0 \times Act_0, \mapsto)$ with respect to $\_[\_]$. Let $\mathbb{E}\langle\mathbb{C}\rangle = (Env \times Con, Act, \Rightarrow)$ be the environment system where $\Rightarrow$ is the smallest relation satisfying:

(i) $e \overset{b}{\Rightarrow} e' \;\&\; C \overset{b}{\underset{0}{\mapsto}} C' \;\Rightarrow\; e\langle C\rangle \overset{1}{\Rightarrow} e'\langle C'\rangle$

(ii) $e \overset{b}{\Rightarrow} e' \;\&\; C \overset{b}{\underset{a}{\mapsto}} C' \;\&\; a \neq 0 \;\Rightarrow\; e\langle C\rangle \overset{a}{\Rightarrow} e'\langle C'\rangle$

Then for all environments e of $\mathbb{E}$ and contexts C of $\mathbb{C}$:

$$e[C] \lesseqgtr e\langle C\rangle \qquad \square$$

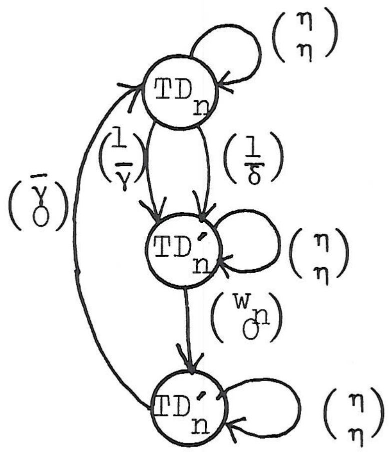
Since the behaviour of $L\langle C\rangle$ only requires considering single atomic actions of L it should be easier to determine than that of $L[C]$. Also, since $e \lesseqgtr f$ implies $\widetilde{D(e)} = \widetilde{D(f)}$ we have:

$$boa_{\mathbb{L}_{si}}(C,L) = \widetilde{D(L\langle C\rangle)}$$

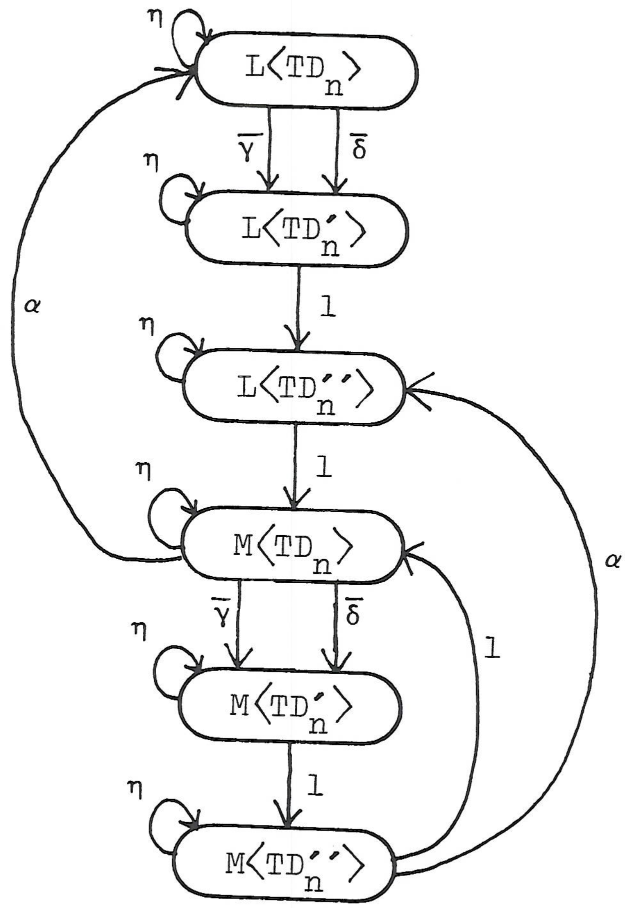
Now, let $TD'_n$ and $TD''_n$ be the following contexts:

$$TD'_n = \{[\,][\gamma \mapsto \delta] \mid (w_n . \bar{\gamma} . C_n)[\alpha \mapsto \delta]\} \backslash \delta$$
$$TD''_n = \{[\,][\gamma \mapsto \delta] \mid (\bar{\gamma} . C_n)[\alpha \mapsto \delta]\} \backslash \delta$$

Then the behaviour of $TD_n$ is easily seen to be described by the following diagram:




where $\eta \in \Delta_{-\gamma\delta} \cup \{1\}$. An arrow labelled $\binom{b}{a}$ between two contexts C and D indicates $C \overset{b}{\underset{a}{\mapsto}} D$. Based on the diagrams for L and $TD_n$ we can determine the behaviour of $L\langle TD_n\rangle$ using the above lemma 5.5-1.




From this diagram it follows immediately that:

$$\widetilde{D(L\langle TD_n\rangle)} \subseteq L^p$$

and hence that condition (8) is satisfied.

This example raises the question of what is the more advantageous: to use the algebraic laws for $boa_{\mathbb{L}_{si}}$ or to appeal directly to the definition of $boa_{\mathbb{L}_{si}}$. Obviously many more examples must be dealt with before this question can be answered.

# CHAPTER 6

# COMPLEXITY RESULTS & PROLOG IMPLEMENTATIONS

When applying the various notions of bisimulation (strong or weak, parameterized or unparameterized) to larger examples (see for example /Pr84/) the availability of automatic or semiautomatic tools becomes of increasing importance for the manageability of the problem. For this reason we shall in this chapter investigate the complexity and implementation of the various notions of bisimulation equivalence.

The (strong or weak, parameterized or unparameterized) equivalence problem is for general CCS-expressions undecidable: given the index i of a Turing Machine $M_i$ it is easy (but tedious) to effectively construct a CCS-expression $p_i$ such that $M_i$ does not halt on input i if and only if $p_i \approx \mathbb{O}$ ($p_i \sim \mu x.1.x$ iff $p_i \approx_U \mathbb{O}$ iff $p_i \sim_U \mu x.1.x$). This reduction actually shows that the various equivalences are not even recursively enumerable (r.e.) for general CCS-expressions.

From the finitary, complete proof systems in /HenMil83, Mil82/ and their parameterized extensions in chapter 4 it follows that, by restricting to finite or regular CCS-expressions, the unparameterized as well as the parameterized strong equivalence problem becomes r.e.

However, as a "complexity-bound" this can be improved drastically due to a result by Paris Kannellakis and Scott Smolka. In /KaSm83/ they show that the unparameterized strong and weak equivalence problems are both polynomial-time decidable for regular CCS-expressions (in terms of the size of the expressions). Given the highly recursive definition of bisimulation equivalence this result is rather surprising. In comparison the seemingly much simpler (traditional automata-theoretical) string or trace equivalence /Hoa81/, failure-equivalence /Bro83,HoBroR84/ and testing-equivalence /NiHen82,Ni85/ problems are all PSPACE-complete for regular CCS-processes and as such highly intractable (see /GJ79,KaSm83/). In section 6.1 we show how to extend this polynomial-time complexity result to the corresponding parameterized equivalences.

In section 6.2 we develop and verify the correctness of a PROLOG implementation for the strong equivalence problem. The implementation, which is easily modified to support the other notions of bisimulation equivalence, is a theorem prover in the following sense: given two processes p and q a procedure will construct a bisimulation (=proof) containing the pair (p,q) if $p \sim q$. If $p \not\sim q$ the procedure will terminate with failure. However, the termination is subject to the condition that the processes p and q have finite state-transition diagrams. Thus regular expressions (e.g. $\mu x.a.x$) or finite CCS-expressions over regular expressions (e.g. $[\mu x.a.x \mid \mathbb{O}]\lceil\{a,b\}$) are allowed, whereas CCS-expressions with a parallel, restriction or renaming operator occurring within the scope of a fixed-point operator will in general lead to non-termination (since such expressions have infinitely many derivatives).

A large subset of CCS and its operational semantics is also implemented in PROLOG. The usefulness of the resulting system is demonstrated through several examples including the simple scheduler from section 5.5 and the closed shop example /San82/.

Finally, in section 6.3, we comment on some existing alternative (semi-) automatic tools for proving bisimulation equivalences, and we discuss what properties future tools might/ought to have.

## 6.1 COMPLEXITY RESULTS

The polynomial-time results in /KaSm83/ are based on the following Generalized Partitioning problem. A partitioning of a set S consists of disjoint, nonempty subsets of S called blocks, whose union is S.

GENERALIZED PARTITIONING.

As input is given a finite set S, an initial partitioning of S $\Gamma_o = \{B_1, \ldots, B_p\}$ and k functions with $f_\ell : S \rightarrow \mathcal{P}(S)$ $(1 \leq \ell \leq k)$.

The problem is to find the coarsest partitioning $\Gamma_f = \{E_1, \ldots, E_q\}$ of S such that:

(i) $\Gamma_f$ is a refinement of $\Gamma_o$ (i.e. each block $E_i$ is a subset of some $B_j$)

(ii) For all blocks $E_i$, all $a, b \varepsilon E_i$, any function $f_\ell$ and any block $E_j$:

$$f_\ell(a) \cap E_j \neq \emptyset \quad \Leftrightarrow \quad f_\ell(b) \cap E_j \neq \emptyset \;^{+} \qquad \square$$

Obviously $\Gamma_f$ is unique if it exists. Existence of $\Gamma_f$ (which is left untreated in /KaSm83/) will follow if, for any two partitionings $\Gamma_1$ and $\Gamma_2$ satisfying (i) and (ii), we can find a partitioning $\Gamma_3$ also satisfying (i) and (ii) and moreover coarser than both $\Gamma_1$ and $\Gamma_2$:

Let $\Gamma = \{F_1, \ldots, F_r\}$ be a set of (not necessarily disjoint) blocks such that (i) and (ii) are satisfied and $\cup_{i \leq r} F_i = S$. Let $\equiv$ be the smallest equivalence on $\{1, \ldots, r\}$ such that $i \equiv j$ if $F_i \cap F_j \neq \emptyset$. Then let $\Gamma_\equiv$ be the set of blocks:

$$\Gamma_\equiv = \{\cup_{j \varepsilon [i]} F_j \mid i \leq r\}$$

where $[i]$ is the equivalence class containing i. Obviously $\Gamma_\equiv$ is coarser that $\Gamma$ and it is not difficult to see that $\Gamma_\equiv$ is a partitioning satisfying (i) and

+ Writing $a \xrightarrow{\ell} a'$ for $a' \in f_\ell(a)$ condition (ii) can be rewritten as:

$$\forall i,j. \forall a,b \in E_i . ((\forall a' \in E_j . a \xrightarrow{\ell} a' \Rightarrow \exists b' \in E_j . b \xrightarrow{\ell} b') \;\&\; (\forall b' \in E_j . b \xrightarrow{\ell} b' \Rightarrow \exists a' \in E_j . a \xrightarrow{\ell} a'))$$

(ii). Now let $\Gamma_1$ and $\Gamma_2$ be the two partitionings satisfying (i) and (ii). Then it follows that $\Gamma_3 = (\Gamma_1 \cup \Gamma_2)_{\equiv}$ will have all the properties required above.

For the following complexity analysis we shall assume that each function $f_\ell$ is effectively represented as a directed graph with node set S and a vertex from a to b iff $b \in f_\ell(a)$. Let $m_\ell$ be the number of vertices in the graph associated with $f_\ell$ (i.e. $m_\ell = \sum_{a \in S} |f_\ell(a)|$). We shall measure the size of an instance of GENERALIZED PARTITIONING as a pair (n,m), where n denotes $|S|$ and m is $\sum_{1 \leq \ell \leq k} m_\ell$ (i.e. the total number of vertices in the graphs associated with $f_1, \ldots, f_k$).

The restricted class of GENERALIZED PARTITIONING problems, for which the k functions are <u>deterministic</u> (i.e. $|f_\ell(a)| = 1$ for all $\ell$ and a), constitutes the well-studied class of PARTITIONING problems which is known to have an $O(k \cdot n \cdot \log n)$ solution (see /AHU74/ §4.13). The PARTITIONING problem has many applications. One important application is the minimalization of the number of states in a deterministic finite automata. In the following we shall see how the GENERALIZED PARTITIONING problem can be applied to solve the (strong) bisimulation equivalence problem.

For any <u>finite process system</u> $\mathbb{P} = (Pr, Act, \rightarrow)$ ($\mathbb{P}$ is finite if and only if Pr and Act are both finite sets) let $\underline{\Lambda_{\mathbb{P}}}$ be the GENERALIZED PARTITIONING problem consisting of the set Pr, the initial partitioning $\Gamma_0^{\mathbb{P}} = \{Pr\}$, and $|Act|$ functions, $f_a : Pr \rightarrow \mathcal{P}(Pr)$ for $a \in Act$, with $f_a(p) = \{p' \mid p \xrightarrow{a} p'\}$. Let $\Gamma_f^{\mathbb{P}}$ be the solution to $\Lambda_{\mathbb{P}}$. Then the following holds:

<u>Theorem 6.1-1:</u> For all processes p and q of $\mathbb{P}$, $p \sim q$ if and only if p and q belong to the same block of $\Gamma_f^{\mathbb{P}}$.

Proof: "$\Leftarrow$": We show that the relation $R \subseteq Pr \times Pr$ defined by:

$$p\,R\,q \quad \Leftrightarrow \quad \text{p and q belong to the same block of } \Gamma_f^{\mathbb{P}}.$$

is a bisimulation and thus $R \subseteq \sim$. Let $pRq$ and $p \xrightarrow{a} p'$. Assume $p' \varepsilon E_j$ where $E_j$ is some block of $\Gamma_f^{\mathbb{P}}$ (such a block exists). Thus $\emptyset \neq \{p'\} \subseteq f_a(p) \cap E_j$. From the closure properties of $\Gamma_f^{\mathbb{P}}$ it follows that $f_a(q) \cap E_j \neq \emptyset$, and hence that $q \xrightarrow{a} q'$ for some $q' \varepsilon E_j$.

"$\Rightarrow$": Let $Pr/\sim$ be the set of equivalence classes of Pr under $\sim$. $Pr/\sim$ is obviously a partitioning of Pr and it is easy to show that $Pr/\sim$ satisfies (i) and (ii). Thus, by definition, $\Gamma_f^{\mathbb{P}}$ is coarser than $Pr/\sim$ from which the "$\Rightarrow$"-direction follows immediately. □

The obvious solution to the GENERALIZED PARTITIONING problem is, starting from $\Gamma_o$, to repeatedly refine the blocks of the partition by the following method. Let $B_i$ be a block in the current partitioning, and let $f_\ell$ be one of the k functions. Examine $f_\ell(a) \subseteq S$ for all a in $B_i$. Now we partition $B_i$ so that two elements a and b are put in the same block if and only if $f_\ell(a)$ and $f_\ell(b)$ intersect the same set of blocks.

```
Γ:= Γo ;
REPEAT
   change := false ;
   FOR all blocks Bi of Γ , all fℓ
   DO  - Partition Bi with respect to
         fℓ into h≥1 new blocks Bi^1,...,Bi^h
       - Γ := (Γ - {Bi}) ∪ {Bi^1,...,Bi^h}
       - if h>1 set change := true
UNTIL  change = false
Γf := Γ
```

(figure 6.1-2)

Theorem 6.1-3: The algorithm in figure 6.1-2 solves the GENERALIZED PARTITIONING problem in $O(n\cdot(n+m))$ time.

Proof: The partial correctness of the algorithm follows fairly easy: at any stage during the execution the initial partition $\Gamma_0$ is coarser than the current one $\Gamma$. Thus $\Gamma_0$ is coarser than $\Gamma_f$. Obviously at exit of the outer loop $\Gamma$, and hence $\Gamma_f$, satisfies (i) and (ii). To prove that the final value of $\Gamma$ is endeed the coarsest refinement of $\Gamma_0$ satisfying (i) and (ii) use the following as a loop-invariant: if $\Gamma'$ is any partition satisfying (i) and (ii), then $\Gamma$ is coarser than $\Gamma'$. For the complexity (and total correctness) we note that the algorithm will terminate after at most n iterations of the outer loop since there can at most be n blocks. A slightly tricky use of the lexicographic sorting method from /AHU74/ makes it possible to perform each iteration in $O(n+m)$ time (see /KaSm83/). □

Corollary 6.1-4: Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ be a finite process system and let p and q be processes of $\mathbb{P}$. Then the strong bisimulation equivalence problem $p \sim q$ can be decided in $O(n\cdot(n+m) + M)$ time, where $n = |Pr|$, $m = |\longrightarrow|$ and M is the time required to compute the derived GENERALIZED PARTITIONING problem $\Lambda_{\mathbb{P}}$.

Proof: Note that for the derived GENERALIZED PARTITIONING problem $\Lambda_{\mathbb{P}}$ the following holds:

$$\sum_{a \in Act} \Big( \sum_{p \in Pr} |f_a(p)| \Big) = |\longrightarrow|$$

Thus the result follows directly from theorems 6.1-1 and 6.1-3. □

Since the regular process system $\mathbb{P}_r$ (see section 4.2) is not a finite process system we cannot apply the above corollary directly to $\mathbb{P}_r$. However, for any pair of processes p and q we can find a finite restriction of $\mathbb{P}_r$ containing p and q and all their derivatives.

Let $\mathbb{P} = (Pr, Act, \rightarrow)$ be a process system and let $Q$ be a $\rightarrow$-closed (see section 4.3) subset of $Pr$. Define the restricted process system $\underline{\mathbb{P}\lceil Q}$ as $(Q, Act_Q, \rightarrow_Q)$, where $Act_Q = \left\{ a \varepsilon Act \mid \exists q \varepsilon Q.\ q \xrightarrow{a} \right\}$ and $\rightarrow_Q = \rightarrow \cap (Q \times Act_Q \times Q) = \rightarrow \cap (Q \times Act \times Q)$. Since $Q$ is $\rightarrow$-closed it is easy to prove that whenever $p, q \varepsilon Q$ then $p \sim q$ in $\mathbb{P}$ iff $p \sim q$ in $\mathbb{P}\lceil Q$.

<u>Corollary 6.1-5:</u> Let p and q be closed regular process expressions. Then $p \sim q$ can be decided in $O(n^3)$ time where $n = ND(p) + ND(q)$ (see section 4.3).

<u>Proof:</u> Let $Q$ be the $\rightarrow$-closed set $DER(p) \cup DER(q)$, where $DER(p) = \left\{ p' \mid \exists s \varepsilon Act^*.\ p \xRightarrow{s} p' \right\}$. Then $p \sim q$ in $\mathbb{P}_r$ iff $p \sim q$ in $\mathbb{P}_r \lceil Q$. From section 4.3 we know that $|DER(p)| \leqslant ND(p)$, hence $|Q| \leqslant n$. Obviously any action which can be performed by any derivative of p must appear in the expression for p. Since $ND(p)$ is increased for each action occurring in p, $Act_Q$ $ND(p) + ND(q) = n$. A simple bound on $|\rightarrow_Q|$ is obtained from the following:

$$
\begin{aligned}
|\rightarrow_Q| &\leqslant |Q \times Act_Q \times Q| \\
&\leqslant |Q| \cdot |Act_Q| \cdot |Q| \\
&\leqslant n^3
\end{aligned}
$$

However, a tighter bound can be obtained by noticing that, for each derivative r of p, there is a bijection from the set $\left\{ (a,s) \mid r \xrightarrow{a} s \right\}$ to the occurrences of action symbols in the expression for p. Thus, for each $r \varepsilon DER(p)$ the size of the set $\left\{ (a,s) \mid r \xrightarrow{a} s \right\}$ can at most be $ND(p)$. Using this observation we get:

$$
\begin{aligned}
|\rightarrow_Q| &\leqslant \sum_{r \varepsilon Q} \left| \left\{ (a,s) \mid r \xrightarrow{a} s \right\} \right| \\
&\leqslant \sum_{r \varepsilon DER(p)} \left| \left\{ (a,s) \mid r \xrightarrow{a} s \right\} \right| + \\
&\qquad \sum_{r \varepsilon DER(q)} \left| \left\{ (a,s) \mid r \xrightarrow{a} s \right\} \right| \\
&\leqslant ND(p)^2 + ND(q)^2 \leqslant n^2
\end{aligned}
$$

Finally, (the effective graph representation of) $\Lambda_{\mathbb{P}_r \restriction Q}$ can be constructed in $O(n^2)$ time (see /KaSm83/ or the similar chart construction in /Mil82/). Thus it follows from corollary 6.1-4 that $p \sim q$ can be decided in $O(n \cdot (n + n^2) + n^2) = O(n^3)$ time. $\square$

<u>Corollary 6.1-6:</u> Let p and q be closed regular expressions. Then $p \approx q$ can be decided in $O(n^4)$ time, where $n = ND(p) + ND(q)$.

<u>Proof:</u> Let Q be the $\longrightarrow$-closed set $DER(p) \cup DER(q)$. Then, also $p \approx q$ in $\mathbb{P}_r$ iff $p \approx q$ in $\mathbb{P}_r \restriction Q$. From corollary 5.3-5 we know that $p \approx q$ in $\mathbb{P}_r \restriction Q$ iff $\#p \sim \#q$ in $\#(\mathbb{P}_r \restriction Q)$. By definition of $\#(\mathbb{P}_r \restriction Q)$ we have, $|\#Q| = |Q| \leqslant n$. Since the derivation relation, $\longrightarrow_{\#}$, of $\#(\mathbb{P}_r \restriction Q)$ is a subset of $\#Q \times Act_Q \times \#Q$ we have the simple bound, $|\longrightarrow_{\#}| \leqslant n^3$. An effective graph representation of $\#(\mathbb{P}_r \restriction Q)$ (and hence of $\Lambda_{\#(\mathbb{P}_r \restriction Q)}$) can be obtained from the effective representation of $\mathbb{P}_r \restriction Q$ using a "transitive-&-relexive closure" type operation, adding a derivation $(\#p, a, \#q)$ to $\longrightarrow_{\#}$ whenever $p \overset{\tilde{a}}{\longrightarrow}_0 q$. Constructing $\#(\mathbb{P}_r \restriction Q)$ from $\mathbb{P}_r \restriction Q$ can as such be done $O(n^3)$ time (see /AHU74/ for "transitive closure" algorithms.)

Thus it follows from corollary 6.1-4 that $\#p \sim \#q$, and hence $p \approx q$ can be decided in $O(n \cdot (n + n^3) + n^3) = O(n^4)$ time. $\square$

Let $\mathbb{E} = (Env, Act, \Longrightarrow)$ be a finite environment system and $\mathbb{P} = (Pr, Act, \longrightarrow)$ a finite process system. We want to reduce the parameterized strong bisimulation equivalence problem over $\mathbb{E}$ and $\mathbb{P}$ to a GENERALIZED PARTITIONING problem, $\Lambda_{\mathbb{E},\mathbb{P}}$.

By choosing the initial partition of $\Lambda_{\mathbb{E},\mathbb{P}}$ carefully we can obtain such a reduction: $\Lambda_{\mathbb{E},\mathbb{P}}$ consists of the set $Env \times Pr$, the initial partition $\Gamma_0^{\mathbb{E},\mathbb{P}} = \{\{e\} \times Pr \mid e \in Env\}$ and $|Act|$ functions, $f_a : Env \times Pr \longrightarrow \mathcal{P}(Env \times Pr)$ for $a \in Act$, with:

$$f_a((e,p)) = \{(e',p') \mid e \overset{a}{\Longrightarrow} e' \ \&\ p \overset{a}{\longrightarrow} p'\}$$

Let $\Gamma_f^{\mathbb{E},\mathbb{P}}$ be the solution to $\Lambda_{\mathbb{E},\mathbb{P}}$. Then the following holds:

<u>Theorem 6.1-7:</u> For all processes p and q of $\mathbb{P}$ and all environments e of $\mathbb{E}$, $p \sim_e q$ if and only if (e,p) and (e,q) belong to the same block of $\Gamma_f^{\mathbb{E},\mathbb{P}}$.

<u>Proof:</u>

"$\Leftarrow$": It suffices to show that the Env-indexed family R with:

$$p\, R_e\, q \;\Leftrightarrow\; \text{(e,p) and (e,q) belong to the same block of } \Gamma_f^{\mathbb{E},\mathbb{P}}$$

is a parameterized bisimulation.

Let $pR_eq$, $e \overset{a}{\Longrightarrow} e'$ and $p \overset{a}{\longrightarrow} p'$. Assume $(e',p') \varepsilon E_j$, where $E_j$ is some block of $\Gamma_f^{\mathbb{E},\mathbb{P}}$ (obviously such a block exists). Thus $\emptyset \neq \{(e',p')\} \subseteq f_a((e,p)) \cap E_j$. From the closure preperties of $\Gamma_f^{\mathbb{E},\mathbb{P}}$ it follows that $f_a((e,q)) \cap E_j \neq \emptyset$. Thus for some $(e'',q') \varepsilon E_j$, $e \overset{a}{\Longrightarrow} e''$ and $q \overset{a}{\longrightarrow} q'$. Since $\Gamma_f^{\mathbb{E},\mathbb{P}}$ is a refinement of the (carefully chosen) $\Gamma_o^{\mathbb{E},\mathbb{P}}$, $e''=e'$. Thus $p'\, R_{e'}\, q'$.

"$\Rightarrow$": Let $\underline{=}$ be the equivalence relation on Env×Pr defined by:

$$(e,p) \underline{=} (f,q) \;\Leftrightarrow\; e=f \ \&\ p \sim_e q$$

and let Env×Pr/$\underline{=}$ be the equivalence classes of Env×Pr under $\underline{=}$. Env×Pr/$\underline{=}$ is obviously a partition of Env×Pr finer than $\Gamma_o^{\mathbb{E},\mathbb{P}}$.

Now, assume (e,p) and (f,q) belong to the same block of Env×Pr/$\underline{=}$ and $(e',p') \varepsilon f_a((e,p)) \cap F_j$ where $F_j$ is some equivalence class of Env×Pr/$\underline{=}$. Thus $e \overset{a}{\Longrightarrow} e'$ and $p \overset{a}{\longrightarrow} p'$. By definition of $\underline{=}$, e=f and $p \sim_e q$. Thus $q \overset{a}{\longrightarrow} q'$ for some $q'$ with $p' \sim_{e'} q'$. Hence $(e',q') \underline{=} (e',p')$ and thus $(e',q') \varepsilon f_a((e,q)) \cap F_j$. By symmetry it follows that Env×Pr/$\underline{=}$ satisfies condition (ii) of the GENERALIZED

PARTITIONING problem. Thus, by definition, $\Gamma_f^{\mathbb{E},\mathbb{P}}$ is coarser than $Env \times Pr/\equiv$ ensuring the "$\Rightarrow$"-direction. $\square$

If we instead had chosen the perhaps more obvious $\{Env \times Pr\}$ as the initial partition for $\Lambda_{\mathbb{E},\mathbb{P}}$, theorem 6.1-7 would fail to hold. It is not hard to see that with this choice, two pairs (e,p) and (e,q) would belong to the same block of the final partition just in case $e \& p \sim e \& q$ (which is a weaker property than $p \sim_e q$).

Corollary 6.1-8: Let $\mathbb{P} = (Pr, Act, \longrightarrow)$ be a finite process system and let $\mathbb{E} = (Env, Act, \Longrightarrow)$ be a finite environment system. Then, for processes p and q of $\mathbb{P}$ and environments e of $\mathbb{E}$, $p \sim_e q$ can be decided in $O(n \cdot (n+m) + M)$ time, where $n = |Pr| \cdot |Env|$, $m = |\longrightarrow| \cdot |\Longrightarrow|$ and M is the time required to compute the derived GENERALIZED PARTITIONING problem $\Lambda_{\mathbb{E},\mathbb{P}}$.

Proof: If we can solve $\Lambda_{\mathbb{E},\mathbb{P}}$ in $O(n \cdot (n+m))$ time the corollary follows directly from theorem 6.1-7. For $\Lambda_{\mathbb{E},\mathbb{P}}$ it is easily seen that:

$$\sum_{a \in Act} \left( \sum_{(e,p) \in Env \times Pr} |f_a((e,p))| \right) = |\longrightarrow| \cdot |\Longrightarrow|$$

Thus the $O(n \cdot (n+m))$ complexity bound for $\Lambda_{\mathbb{E},\mathbb{P}}$ follows directly from theorem 6.1-3. $\square$

Corollary 6.1-9: Let p and q be closed regular process expressions and let e be a closed regular environment expression different from U. Then $p \sim_e q$ can be decided in $O((n_P \cdot n_E)^3)$ time, where $n_P = ND(p) + ND(q)$ and $n_E = ND(e)$.

Proof: The proof is very similar to the proof of corollary 6.1-5. Let $Q_P = DER(p) \cup DER(q)$ and $Q_E = DER(e)$. Then it is easily seen that $p \sim_e q$ in $\mathbb{P}_r$ and $\mathbb{E}_r$ (i.e. there is an $\mathbb{E}_r$-parameterized bisimulation R over $\mathbb{P}_r$ such that $(p,q) \in R_e$) iff $p \sim_e q$ in $\mathbb{P}_r \lceil Q_P$ and $\mathbb{E}_r \lceil Q_E$. Since $\mathbb{P}_r \lceil Q_P$ and $\mathbb{E}_r \lceil Q_E$ are finite we can apply the previous corollary 6.1-8. Let $\mathbb{P}_r \restriction Q_P = (Q_P, Act_P, \rightarrow_P)$ and $\mathbb{E}_r \restriction Q_E = (Q_E, Act_E, \Rightarrow_E)$. Then

$$|Q_P| \leqslant n_P$$
$$|Q_E| \leqslant n_E$$
$$|\rightarrow_P| \leqslant n_P^2$$
$$|\Rightarrow_E| \leqslant n_E^2$$

follows from arguments similar to those of corollary 6.1-5. It only remains to see how fast the GENERALIZED PARTITIONING problem $\Lambda_{\mathbb{E}_r \restriction Q_E, \mathbb{P}_r \restriction Q_P}$ can be constructed (or rather an effective graph representation of it). Since $\Lambda_{\mathbb{E}_r \restriction Q_E, \mathbb{P}_r \restriction Q_P}$ essentially is the "product" of $\Lambda_{\mathbb{E}_r \restriction Q_E}$ (size $(n_E, n_E^2)$ ) and $\Lambda_{\mathbb{P}_r \restriction Q_P}$ (size $(n_P, n_P^2)$ ) it can be constructed in $O(n_P^2 \cdot n_E^2)$ time. Thus it follows from corollary 6.1-8, that $p \sim_e q$ can be decided in $O(n_P \cdot n_E \cdot (n_P \cdot n_E + n_P^2 \cdot n_E^2) + n_P^2 \cdot n_E^2) = O((n_P \cdot n_E)^3)$ time. □

From the results of section 5.3 it follows that $p \approx_e q$ if and only if $\#p \sim_{\#(@e)} \#q$ ($\#(@e)$ is a strongly idle environment equivalent under $\lesssim$ to e). Thus, using a technique similar to the one for the proof of corollary 6.1-6, we can for regular processes p and q and regular environments e obtain a polynomial-time complexity result for the parameterized weak bisimulation equivalence problem $p \approx_e q$. (Note, that $\#\mathbb{P}$ and $\#(@\mathbb{E})$ can be obtained by "transitive-&-reflexive closure" type operations). More precisely, $p \approx_e q$ can be decided in $O((n_P \cdot n_E)^4)$ time, where $n_P$ and $n_E$ are as in corollary 6.1-9.

## 6.2 PROLOG IMPLEMENTATIONS

In this section we shall develop and verify an alternative decision procedure for the strong equivalence problem (the procedure is easily modified for other notions of bisimulation equivalence). In contrast to the polynomial time algorithm (figure 6.1-2) from the previous section, which computes the <u>maximal</u> bisimulation, the alternative procedure will for a given pair of processes try to construct a <u>minimal</u> bisimulation containing the pair. The procedure follows very closely the recursive definition of bisimulation and may involve backtracking in case the processes are non-deterministic. Thus, the time complexity of the procedure is essentially exponential. However, the previous section's polynomial time results only hold for <u>regular</u> CCS-expressions. By allowing parallel compositions of regular process expressions, an (extended) expression may have an exponential number of derivatives (in terms of the size of the expression), because of possible nesting of parallel operators. Thus the equivalence problem is likely to become hard anyway. (As an analogy, the string equivalence problem for regular expressions increases in complexity when the intersection operator is added - see /HU79/ exercise 13.32). The new alternative procedure is moreover extremely easy to implement in PROLOG, as we shall demonstrate in the following.

### 6.2.1 An operational-based inference system for bisimulation.

Let $\mathbb{P} = (Pr, Act, \rightarrow)$ be a given process system. We shall present an inference system for constructing bisimulations over $\mathbb{P}$ based on the derivation relation of $\mathbb{P}$. We shall prove both soundness and restricted completeness of the inference system. Also, we shall later see that the inference system can be represented directly in PROLOG.

Let

$$\text{bisim} \subseteq Pr \times Pr \times \mathcal{P}(Pr^2)$$

$$\left.\begin{array}{l}\text{closure}\\ \text{matchl}\\ \text{mathcr}\end{array}\right\} \subseteq Pr \times Pr \times \mathcal{P}(Pr^2) \times \mathcal{P}(Pr^2)$$

$$\left.\begin{array}{l}\text{matchl}^+\\ \text{matchr}^+\end{array}\right\} \subseteq Pr \times Pr \times \mathcal{P}(Act \times Pr) \times \mathcal{P}(Pr^2) \times \mathcal{P}(Pr^2)$$

be the smallest relations closed under the following rules (an informal explanation will be given after the rules).

B
$$\frac{\text{closure}(p,q,\{(p,q)\},C)}{\text{bisim}(p,q,C)}$$

C
$$\frac{\text{matchl}(p,q,B,C) \qquad \text{matchr}(p,q,C,D)}{\text{closure}(p,q,B,D)}$$

ML
$$\frac{\text{matchl}^+(p,q,M,B,C)}{\text{matchl}(p,q,B,C)} \;;\; M = \{(a,p') \mid p \xrightarrow{a} p'\}$$

MR
$$\frac{\text{matchr}^+(p,q,N,C,D)}{\text{matchr}(p,q,C,D)} \;;\; N = \{(a,q') \mid q \xrightarrow{a} q'\}$$

$ML^+$
$$\text{matchl}^+(p,q,\emptyset,B,B)$$

$$\frac{\text{matchl}^+(p,q,M,B,D)}{\text{matchl}^+(p,q,\{(a,p')\} \cup M,B,D)} \;;\; \begin{array}{l} q \xrightarrow{a} q' \\ (p',q') \varepsilon B \end{array}$$

$$\frac{\text{closure}(p',q',\{(p',q')\} \cup B,C) \qquad \text{matchl}^+(p,q,M,C,D)}{\text{matchl}^+(p,q,\{(a,p')\} \cup M,B,D)} \;;\; q \xrightarrow{a} q'$$

$MR^+$
$$\text{matchr}^+(p,q,\emptyset,B,B)$$

$$\frac{\text{matchr}^+(p,q,N,B,D)}{\text{matchr}^+(p,q,\{(a,q')\} \cup N,B,D)} \;;\; \begin{array}{l} p \xrightarrow{a} p' \\ (p',q') \varepsilon B \end{array}$$

$$\frac{\text{closure}(p',q',\{(p',q')\} \cup B,C) \qquad \text{matchr}^+(p,q,N,C,D)}{\text{matchr}^+(p,q,\{(a,q')\} \cup N,B,D)} \;;\; p \xrightarrow{a} p'$$

(figure 6.2-1)

Now, think of bisim as a (partial) function from its first two arguments to its third argument, closure, matchl and matchr as (partial) functions from their first three arguments to their last argument, and $matchl^+$ and $matchr^+$ as (partial) functions from their first four arguments to their last argument. Then, the intended meaning of the six relations can informally be described as follows:

- Given two processes p and q, bisim will try to "build" up a bisimulation C containing (p,q).

- Given two processes p and q and an approximate bisimulation B containing (p,q) ("approximate" in the sense that B is not yet known to be closed under $\mathbb{B}$, in particular it is unknown whether $(p,q) \varepsilon \mathbb{B}(B)$ or not), closure will try to extend B to a genuine bisimulation C.

- Given two processes p and q and an approximate bisimulation B containing (p,q), matchl will try to extend B to an approximate bisimulation C closed under $\mathbb{S}$ (i.e. C is a simulation), whereas matchr will try to extend B to an approximate bisimulation C closed under $\overline{\mathbb{S}}$.

From the definition of $\mathbb{S}$ it follows that the approximate bisimulation C constructed by matchl must be such that for each derivation (a,p′) of p (i.e. $p \xrightarrow{a} p'$) q has a match in C, i.e. $q \xrightarrow{a} q'$ for some q′ with $(p',q') \varepsilon C$. Obviously we would like to construct C by dealing with one derivation of p at a time. For this reason a refined version of matchl, $matchl^+$, augmented with a fifth argument for keeping track of which of p′s derivatives that are left to be dealt with, is introduced.

- Given two processes p and q, an approximate bisimulation B containing (p,q) and a subset M of p′s derivations such that q only remains to match those of p's derivations which are in M. Then $matchl^+$ will try to extend B to an approximate bisimulation closed under $\mathbb{SS}$.

Similarly a refined version, $matchr^+$, of matchr is introduced.

Note, that by letting M be the set of <u>all</u> of p's derivations ($M = \{(a,p') \mid p \xrightarrow{a} p'\}$) we can reduce matchl to $matchl^+$. This explains the rules ML and MR.

To see how to realize $matchl^+$, note that when M is empty we are done: simply take C to be B. Otherwise, let $(a,p')$ be a derivation of p in M. We remove $(a,p')$ from M observing the following two cases:

- Assume $q \xrightarrow{a} q'$ for some $q'$ with $(p',q') \varepsilon B$. In this case q already has a match in B for the derivation $(a,p')$ and we can simply remove $(a,p')$ from M.

- If q cannot match the derivation $(a,p')$ in B we extend B with a pair $(p',q')$ where $q \xrightarrow{a} q'$ (it may later be discovered that the chosen a-derivation $q'$ of q is not a match for $(a,p')$. Thus backtracking to this point may be necessary in order to replace the chosen $q'$ with another a-derivation of q). Obviously, q will then have a match for $(a,p')$ in the extended set. However, since the final extension C is required to be an approximate bisimulation itself, we "close" $B \cup \{(p',q')\}$ with respect to $(p',q')$ before dealing with the remaining derivations of M.

The above three cases (including $M = \emptyset$) correspond to the three rules of $ML^+$.

In the following we shall formalize the above informal descriptions: we shall prove that the inference system is <u>sound</u> in the sense that:

$$\boxed{\begin{array}{l} \text{bisim}(p,q,C) \quad \Rightarrow \\ \qquad (p,q)\varepsilon C \quad \& \\ \qquad C \subseteq \mathbb{B}(C) \end{array}}$$

Thus, if it can be derived form the rules that bisim(p,q,C), then we can conclude that $p \sim q$. We shall also indicate how, under certain finiteness assumptions, to prove the following <u>completeness</u> result:

$$\boxed{p \sim q \quad \Rightarrow \quad \exists C.\ \text{bisim}(p,q,C)}$$

Obviously, in order to prove the above soundness result it will be necessary to prove auxiliary properties about the other relations used in the system. Assume that the vague notion of an <u>approximate bisimulation</u> of a pair (p,q) is given by the following:

$$B - \{(p,q)\} \subseteq \mathbb{B}(B)$$

i.e. B would be a bisimulation if $(p,q)\varepsilon\mathbb{B}(B)$. Then according to their informal descriptions, closure, matchl and matchr ought to satisfy the following properties/verification conditions:

$$\boxed{\begin{array}{l} \text{closure}(p,q,B,D) \quad \Rightarrow \\ \qquad \left\{ \begin{array}{l} (p,q)\varepsilon B \quad \& \\ B - \{(p,q)\} \subseteq \mathbb{B}(B) \end{array} \right\} \Rightarrow \\ \qquad B \subseteq D \quad \& \quad D \subseteq \mathbb{B}(D) \end{array}}$$

$$
\begin{array}{l}
\mathrm{matchl}(p,q,B,C) \Rightarrow \\
\qquad \left[ \begin{array}{l} (p,q) \varepsilon B \quad \& \\ B - \{(p,q)\} \subseteq \mathbb{B}(B) \end{array} \right] \Rightarrow \\
\qquad \left\{ \begin{array}{l} B \subseteq C \quad \& \\ C - \{(p,q)\} \subseteq \mathbb{B}(C) \quad \& \\ (p,q) \varepsilon \mathbb{S}(C) \end{array} \right\}
\end{array}
$$

$$
\begin{array}{l}
\mathrm{matchr}(p,q,C,D) \Rightarrow \\
\qquad \left[ \begin{array}{l} (p,q) \varepsilon C \quad \& \\ C - \{(p,q)\} \subseteq \mathbb{B}(C) \end{array} \right] \Rightarrow \\
\qquad \left\{ \begin{array}{l} C \subseteq D \quad \& \\ D - \{(p,q)\} \subseteq \mathbb{B}(D) \quad \& \\ (p,q) \varepsilon \overline{\mathbb{S}}(D) \end{array} \right\}
\end{array}
$$

Note, that by thinking of closure, matchl and matchr as (partial) functions the above properties are verification conditions (or pre- and post-conditions) in the sense that the results of the functions are guaranteed to have certain properties provided the arguments to the functions satisfy certain constraints.

The six relations bisim, closure, matchl, ... is actually the fixed-point of the functional associated with the inference system figure 6.2-1 (see section 3.2 and /A83/). For this reason certain equivalences holds, in particular:

$$(1) \qquad \mathrm{bisim}(p,q,C) \Leftrightarrow \mathrm{closure}(p,q,\{(p,q)\},C)$$

and

$$(2) \qquad \mathrm{closure}(p,q,B,D) \Leftrightarrow \exists C.\ \mathrm{matchl}(p,q,B,C) \ \&\ \mathrm{matchr}(p,q,C,D)$$

If the verification condition for closure holds then the soundness theorem follows directly from (1) since $(p,q) \varepsilon \{(p,q)\}$ and $\{(p,q)\} - \{(p,q)\} = \emptyset \subseteq \mathbb{B}(\{(p,q)\})$. Similarly, if the verification conditions for matchl and matchr hold, then the verification condition for closure will follow from (2). However, from the rules of the inference system it is obvious that the six relations are mutually dependent. Thus, in order to complete the soundness proof an (simultaneous) induction proof is needed.

The induction principle associated with the inference system is straightforward (see /A83/): let Bis, Cl, Ml, Mr, $Ml^+$, $Mr^+$ be six relations (of the appropriate type) also closed under the rules of the inference system. Then, by the leastness of bisim, closure, matchl, matchr, $matchl^+$ and $matchr^+$ it follows that:

| | |
|---|---|
| $bisim \subseteq Bis$ | $matchr \subseteq Mr$ |
| $closure \subseteq Cl$ | $matchl^+ \subseteq Ml^+$ |
| $matchl \subseteq Ml$ | $matchr^+ \subseteq Mr^+$ |

For Cl, Ml, Mr it seems natural first to try the previous verification conditions for closure, matchl, matchr. Unfortunately, these verification conditions are, though true, too weak for the induction proof to go through. In order to obtain stronger conditions we shall introduce a much more liberal definition of an approximate bisimulation B for a pair (p,q), being simply $(p,q)\varepsilon B$.

We can now reveal the definitions of these stronger verification conditions Bis, Cl, Ml, Mr, $Ml^+$ and $Mr^+$:

$$Bis(p,q,C) \overset{\Delta}{\Leftrightarrow}$$

$$(p,q)\varepsilon\, C \quad \& \\ C \subseteq \mathbb{B}(C)$$

$$Cl(p,q,B,D) \overset{\Delta}{\Leftrightarrow}$$

$$(p,q)\varepsilon\, B \quad \Rightarrow$$

$$\begin{bmatrix} B \subseteq D \quad \& \\ D - (B - \{(p,q)\}) \subseteq \mathbb{B}(D) \end{bmatrix}$$

$$
\begin{aligned}
&\mathrm{Ml}(p,q,B,C) \;\overset{\Delta}{\Leftrightarrow}\\
&\qquad (p,q)\,\varepsilon\, B \;\Rightarrow\\
&\qquad \begin{bmatrix} B \subseteq C \quad \& \\ C - B \subseteq \mathbb{B}(C) \quad \& \\ (p,q)\;\varepsilon\; \mathbb{S}(C) \end{bmatrix}\\[1em]
&\mathrm{Mr}(p,q,C,D) \;\overset{\Delta}{\Leftrightarrow}\\
&\qquad (p,q)\,\varepsilon\, C \;\Rightarrow\\
&\qquad \begin{bmatrix} C \subseteq D \quad \& \\ D - C \subseteq \mathbb{B}(D) \quad \& \\ (p,q)\;\varepsilon\; \overline{\mathbb{S}}(D) \end{bmatrix}\\[1em]
&\mathrm{Ml}^{+}(p,q,M,B,C) \;\overset{\Delta}{\Leftrightarrow}\\
&\qquad \begin{bmatrix} (p,q)\,\varepsilon\, B \quad \& \\ M \subseteq \left\{(a,p') \mid p \xrightarrow{a} p'\right\} \quad \& \\ (p_M,q)\;\varepsilon\; \mathbb{S}(B) \end{bmatrix} \Rightarrow\\
&\qquad \begin{bmatrix} B \subseteq C \quad \& \\ C - B \subseteq \mathbb{B}(C) \quad \& \\ (p,q)\;\varepsilon\; \mathbb{S}(C) \end{bmatrix}\\[1em]
&\mathrm{Mr}^{+}(p,q,N,C,D) \;\overset{\Delta}{\Leftrightarrow}\\
&\qquad \begin{bmatrix} (p,q)\,\varepsilon\, C \quad \& \\ N \subseteq \left\{(a,q') \mid q \xrightarrow{a} q'\right\} \quad \& \\ (p,q_N)\;\varepsilon\; \overline{\mathbb{S}}(C) \end{bmatrix} \Rightarrow\\
&\qquad \begin{bmatrix} C \subseteq D \quad \& \\ D - C \subseteq \mathbb{B}(D) \quad \& \\ (p,q)\;\varepsilon\; \overline{\mathbb{S}}(D) \end{bmatrix}
\end{aligned}
$$

where for $M \subseteq \left\{(a,p') \mid p \xrightarrow{a} p'\right\}$, $p_M$ is defined by:

$$p_M = \sum \left\{ a.p' \mid p \xrightarrow{a} p' \wedge (a,p') \notin M \right\}$$

It is not difficult to show that Cl, Ml and Mr are indeed stronger than the previous verification conditions closure, matchl and matchr.

We can now prove that Bis, Cl, Ml, Mr, $Ml^+$ and $Mr^+$ are closed under the rules of the inference system, thus implying the following Soundness Theorem:

Theorem 6.2-2: $Bisim(p,q,C) \Rightarrow (p,q) \varepsilon C \;\;\&\;\; C \subseteq \mathbb{B}(C)$

Proof: We consider each rule in turn:

Rule B: We must prove:

$$Cl(p,q,\{(p,q)\},C) \Rightarrow Bis(p,q,C)$$

or

$$Cl(p,q,\{(p,q)\},C) \Rightarrow (p,q) \varepsilon C \;\;\&\;\; C \subseteq \mathbb{B}(C)$$

This follows immediately from the definition of Cl, $(p,q) \varepsilon \{(p,q)\}$ and $C - (\{(p,q)\} - \{(p,q)\}) = C$.

Rule C: We must prove:

$$(1) \quad [Ml(p,q,B,C) \;\;\&\;\; Mr(p,q,C,D)] \;\Rightarrow\; Cl(p,q,B,D)$$

Assuming the antecedent of (1) and the antecedent of the conclusion of (1) ( $(p,q) \varepsilon B$ ) we must prove:

$$(2) \quad \begin{array}{l} 1.\; B \subseteq D \;\;\& \\ 2.\; D - (B - \{(p,q)\}) \subseteq \mathbb{B}(D) \end{array}$$

Now, $(p,q) \varepsilon B$ together with $Ml(p,q,B,C)$ gives:

$$(3) \quad \begin{array}{l} 1.\; B \subseteq C \;\;\& \\ 2.\; C - B \subseteq \mathbb{B}(C) \;\;\& \\ 3.\; (p,q) \;\varepsilon\; \mathbb{S}(C) \end{array}$$

Since $B \subseteq C$ also $(p,q) \varepsilon C$. Thus, from $Mr(p,q,C,D)$ we can conclude:

$$(4) \quad \begin{array}{l} 1.\; C \subseteq D \;\;\& \\ 2.\; D - C \subseteq \mathbb{B}(D) \;\;\& \\ 3.\; (p,q) \;\varepsilon\; \overline{\mathbb{S}}(D) \end{array}$$

Obviously (3.1) and (4.1) gives (2.1). (2.2) can be rewritten as:

$$(2.2') \qquad (D - C) \cup (C - B) \cup \{(p,q)\} \subseteq \mathbb{B}(D)$$

From (3.2) and (4.2) and monotonicity of $\mathbb{B}$ it only remains to demonstrate:

$$\{(p,q)\} \subseteq \mathbb{B}(D)$$

From (3.3) it follows that $(p,q) \varepsilon \mathbb{S}\mathbb{S}(D)$. Thus, from (4.3) and $\mathbb{B}(D) = \mathbb{S}\mathbb{S}(D) \cap \overline{\mathbb{S}\mathbb{S}}(D)$ it follows that $(p,q) \varepsilon \mathbb{B}(D)$.

<u>Rule ML:</u> We must prove:

$$Ml^{+}(p,q,M,B,C) \Rightarrow Ml(p,q,B,C)$$

when $M = \{(a,p') \mid p \xrightarrow{a} p'\}$. Since $Ml^{+}(p,q,M,B,C)$ and $Ml(p,q,B,C)$ have the same conclusion it suffices to prove that the antecedent of $Ml(p,q,B,C)$ implies the antecedent of $Ml^{+}(p,q,M,^{B},C)$, i.e.:

$$(p,q) \varepsilon B \Rightarrow \left[\begin{array}{l} (p,q) \varepsilon B \quad \& \\ M \subseteq \{(a,p') \mid p \xrightarrow{a} p'\} \quad \& \\ (p_M, q) \varepsilon \mathbb{S}\mathbb{S}(B) \end{array}\right.$$

Only $(p_M,q) \varepsilon \mathbb{S}\mathbb{S}(B)$ does not follow immediately. However, $M = \emptyset$ so $p_M = \mathbb{O}$. Thus trivially $(p_M,q) \varepsilon \mathbb{S}\mathbb{S}(B)$.

<u>Rule Mr:</u> Similar to Ml.

<u>Rule $Ml^{+}$ 1:</u> We must prove:

$$Ml^{+}(p,q,\emptyset,B,B)$$

or equivalently:

$$\left[\begin{array}{l} (p,q) \varepsilon B \quad \& \\ \emptyset \subseteq \{(a,p') \mid p \xrightarrow{a} p'\} \quad \& \\ (p_{\emptyset}, q) \varepsilon \mathbb{S}\mathbb{S}(B) \end{array}\right] \Rightarrow$$

$$\left[\begin{array}{l} B \subseteq B \quad \& \\ \emptyset \subseteq \mathbb{B}(B) \quad \& \\ (p,q) \varepsilon \mathbb{S}\mathbb{S}(B) \end{array}\right]$$

which is trivially true since $(p,q)\varepsilon\mathbb{SB}(B)$ iff $(p_{\emptyset},q)\varepsilon\mathbb{SB}(B)$ .

Rule $M1^{+}$ 2: We must prove:

$$(0) \qquad M1^{+}(p,q,M,B,D) \Rightarrow M1^{+}(p,q,\{(a,p')\}\cup M,B,D)$$

when $q \xrightarrow{a} q'$ for some $q'$ with $(p',q')\varepsilon B$. Since $M1^{+}(p,q,M,B,D)$ and $M1^{+}(p,q,\{(a,p')\}\cup M,B,D)$ have the same conclusion it suffice to prove that the antecedent of $M1^{+}(p,q,\{(a,p')\}\cup M,B,D)$ implies the antecedent of $M1^{+}(p,q,M,B,D)$, i.e.:

$$\left[\begin{array}{l}(p,q)\varepsilon B \quad \& \\ M\cup\{(a,p')\}\subseteq\{(a,p')\mid p\xrightarrow{a}p'\} \quad \& \\ (p_{M\cup\{(a,p')\}},q)\varepsilon\mathbb{SB}(B)\end{array}\right] \Rightarrow$$

$$\left[\begin{array}{l}(p,q)\varepsilon B \quad \& \\ M\subseteq\{(a,p')\mid p\xrightarrow{a}p'\} \quad \& \\ (p_{M},q)\varepsilon\mathbb{SB}(B)\end{array}\right]$$

Only $(p_M,q)\varepsilon\mathbb{SB}(B)$ does not follow immediately. However, $p_M = p_{M\cup\{(a,p')\}} + a.p'$, and $(p_{M\cup\{(a,p')\}},q)\varepsilon\mathbb{SB}(B)$ by the antecedent. Since $q\xrightarrow{a}q'$ and $(p',q')\varepsilon B$ also $(a.p',q)\varepsilon\mathbb{SB}(B)$. Thus $(p_M,q)\varepsilon\mathbb{SB}(B)$.

Rule $M1^{+}$ 3: We must prove:

$$(0) \qquad \left[\begin{array}{l}C1(p',q',\{(p',q')\}\cup B,C) \quad \& \\ M1^{+}(p,q,M,C,D)\end{array}\right] \Rightarrow$$
$$M1^{+}(p,q,M\cup\{(a,p')\},B,D)$$

when $q\xrightarrow{a}q'$. Assume the antecedent of (0), i.e.:

$$(1) \qquad C1(p',q',\{(p',q')\}\cup B,C) \Leftrightarrow$$
$$\begin{array}{ll}1. & \{(p',q')\}\cup B\subseteq C \quad \& \\ 2. & C-((B\cup\{(p',q')\})-\{(p',q')\})\subseteq\mathbb{B}(C)\end{array}$$

since $(p',q')\subseteq\{(p',q')\}\cup B$ is trivially true, and:

(2) $\quad Ml^{+}(p,q,M,C,D) \quad \Leftrightarrow$

$$\text{(ant)} \quad \left[\begin{array}{ll} 1.\ (p,q) \in C & \& \\ 2.\ M \subseteq \{(a,p') \mid p \xrightarrow{a} p'\} & \& \\ 3.\ (p_M, q) \in \mathbb{S}(C) & \end{array}\right] \Rightarrow$$

$$\text{(concl)} \quad \left[\begin{array}{ll} 1.\ C \subseteq D & \& \\ 2.\ D - C \subseteq \mathbb{B}(D) & \& \\ 3.\ (p,q) \in \mathbb{S}(D) & \end{array}\right]$$

Also assume the antecedent of the conclusion of (0), i.e.:

(3) $\quad$ 1. $\quad (p,q) \in B \quad \&$

$\quad\quad$ 2. $\quad M \cup \{(a,p')\} \subseteq \{(a,p') \mid p \xrightarrow{a} p'\} \quad \&$

$\quad\quad$ 3. $\quad (p_{M\cup\{(a,p')\}}, q) \in \mathbb{S}(B)$

From the assumptions we must now prove:

(4) $\quad$ 1. $\quad B \subseteq D \quad \&$

$\quad\quad$ 2. $\quad D - B \subseteq \mathbb{B}(D) \quad \&$

$\quad\quad$ 3. $\quad (p,q) \in \mathbb{S}(D)$

First let us establish (2.ant): (2.ant.1) follows from (3.1) and (1.1). (2.ant.2) follows from (3.2). To see (2.ant.3), note that $p_M = p_{M\cup\{(a,p')\}} + a.p'$. Using (3.3) it suffice to prove that $(a.p',q) \in \mathbb{S}(C)$. However, $q \xrightarrow{a} q'$ and by (1.1) $(p',q') \in C$.

So we can now use (2.concl). Let us now prove (4). (4.1) follows from (1.1) and (2.concl.1). For (4.2) note that $D - B = (D - C) \cup (C - B) \subseteq (D - C) \cup (C - B')$ where $B' = (B \cup \{(p',q')\}) - \{(p',q')\}$. By (2.concl.2), (1.2) and monotonicity of $\mathbb{B}$ it follows that $D - B \subseteq \mathbb{B}(D)$. Finally, (4.3) is identical to (2.concl.3).

<u>Rules $Mr^{+}$ 1,2,3:</u> Similar to $Ml^{+}$ 1,2,3. $\square$

Using the induction principle associated with the inference system figure 6.2-1 once more, it is straightforward to prove that the following <u>finiteness</u> conditions hold:

$$\text{bisim}(p,q,C) \Rightarrow C \text{ is finite}$$

$$\text{closure}(p,q,B,C) \Rightarrow B \text{ is finite} \Rightarrow C \text{ is finite}$$

$$\text{matchl}(p,q,B,C) \Rightarrow B \text{ is finite} \Rightarrow C \text{ is finite}$$

$$\text{matchl}^+(p,q,M,B,C) \Rightarrow \begin{bmatrix} M \text{ is finite} \\ \& \\ B \text{ is finite} \end{bmatrix} \Rightarrow C \text{ is finite}$$

With similar finiteness conditions for matchr and matchr$^+$.

Since any bisimulation C containing (p,q) must also contain a pair for each derivative of p (and similarly a pair for each derivative of q), it follows that the inference system cannot be complete if the processes p and q have infinitely many derivatives. Similarly, from the fourth finiteness condition it follows that the processes p and q as well as their derivatives must have finitely many derivations (i.e. the set $\{(a,p') \mid p \xrightarrow{a} p'\}$ is finite) for the inference system to be complete.

Thus, we can at most hope for completeness for processes p and q with <u>finite state-transition diagrams</u> in the sense that $\mathbb{P}\restriction DER(p)$ and $\mathbb{P}\restriction DER(q)$ are finite transition systems. Fortunately the inference system turns out to be complete for all such processes. We give an outline of the completeness proof in the following, leaving the details to the reader.

As a first attempt we might try proving the following inclusions:

| | |
|---|---|
| $Bis \subseteq bisim$ | $Mr \subseteq matchr$ |
| $Cl \subseteq closure$ | $Ml^+ \subseteq matchl^+$ |
| $Ml \subseteq matchl$ | $Mr^+ \subseteq matchr^+$ |

However, the verification conditions Bis, Cl, Ml,... do not satisfy the previous finiteness conditions and the above inclusions are therefore not valid. Also, viewing bisim, closure, ... as (partial) functions, we shall only require the above inclusions to hold when the input-arguments satisfy the "pre-conditions" of the relevant verification condition. Thus, we shall be content with the following weaker type of implications to hold:

$$\begin{bmatrix} \text{ANT Cl}(p,q,B,C) \quad \& \\ \text{Cl}(p,q,B,C) \end{bmatrix} \Rightarrow$$
$$\exists C' \subseteq C.\ \text{closure}(p,q,B,C')$$

To prove the correctness of these implications we define for each relation a size function which measures the size of the (input) arguments given. The proof is then performed by induction on the size of the input-arguments.

For $p \varepsilon Pr$ we already have $DER(p) = \{p' \mid \exists s \varepsilon Act^*.\ p \xrightarrow{s} p'\}$. Now extend DER to subsets M of Act×Pr by $DER(M) = \{p' \mid \exists (a,p) \varepsilon M. \exists s \varepsilon Act^*.\ p \xrightarrow{s} p'\}$. Then define the following size functions:

$$\begin{aligned}
\#_{Bis}(p,q,C) &= |DER(p) \times DER(q)| \\
\#_{Cl}(p,q,B,C) &= |DER(p) \times DER(q) - B| + 1 \\
\#_{Ml}(p,q,B,C) &= |DER(p) \times DER(q) - B| + 1 \\
\#_{Ml^+}(p,q,M,B,C) &= |DER(M) \times DER(q) - B| \\
\#_{Mr}(p,q,B,C) &= |DER(p) \times DER(q) - B| + 1 \\
\#_{Mr^+}(p,q,N,B,C) &= |DER(p) \times DER(N) - B|
\end{aligned}$$

Note, that all the size functions only depend on the input-arguments. For $\#_{Cl}$ B is to be thought of as the part of the final bisimulation which have been established so far. Thus, $DER(p) \times DER(q) - B$ is the state space which remains to be investigated. Note, that $\#_{Ml^+}$ is independent of its first input-argument p.

Instead, the set of derivations M of p which remains to be matched by q is used.

Lemma 6.2-3: If p and q have finite state-transition diagrams, then for all $n \in \omega$ :

(i) $\begin{bmatrix} \mathrm{Bis}(p,q,C) & \& \\ \#_{\mathrm{Bis}}(p,q,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{bisim}(p,q,C')$

(ii) $\begin{bmatrix} \mathrm{ANT}(\mathrm{Cl}(p,q,B,C)) & \& \\ \mathrm{Cl}(p,q,B,C) & \& \\ \#_{\mathrm{Cl}}(p,q,B,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{closure}(p,q,B,C')$

(iii) $\begin{bmatrix} \mathrm{ANT}(\mathrm{Ml}(p,q,B,C)) & \& \\ \mathrm{Ml}(p,q,B,C) & \& \\ \#_{\mathrm{Ml}}(p,q,B,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{matchl}(p,q,B,C')$

(iv) $\begin{bmatrix} \mathrm{ANT}(\mathrm{Mr}(p,q,B,C) & \& \\ \mathrm{Mr}(p,q,B,C) & \& \\ \#_{\mathrm{Mr}}(p,q,B,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{matchr}(p,q,B,C')$

(v) $\begin{bmatrix} \mathrm{ANT}(\mathrm{Ml}^{+}(p,q,M,B,C)) & \& \\ \mathrm{Ml}^{+}(p,q,M,B,C) & \& \\ \#_{\mathrm{Ml}^{+}}(p,q,M,B,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{matchl}^{+}(p,q,M,B,C')$

(vi) $\begin{bmatrix} \mathrm{ANT}(\mathrm{Mr}^{+}(p,q,N,B,C)) & \& \\ \mathrm{Mr}^{+}(p,q,N,B,C) & \& \\ \#_{\mathrm{Mr}^{+}}(p,q,N,B,C) \leq n & \end{bmatrix} \Rightarrow \exists C' \subseteq C.\ \mathrm{matchr}^{+}(p,q,N,B,C')$

Proof: By induction on n with subinductions on $|M|$ and $|N|$ for (v) and (vi). $ML^{+}$ 3 (and similarly $MR^{+}$ 3) only needs to be used when q does not have a match for $(a,p')$ in B (otherwise $ML^{+}$ 2 is applicable). It is therefore easy to see that using the inference rules backwards once or twice will decrease the size of the input arguments and hence make the induction hypothesis applicable. □

From this lemma the following completeness result follows immediately:

Theorem 6.2-4: If p and q have finite state-transition diagrams then:

$$p \sim q \;\Rightarrow\; \exists C.\ \text{bisim}(p,q,C)$$

Proof: Since p and q have finite-state transition diagrams, $\#_{Bis}(p,q,C)\,\varepsilon\,\omega$ for all C. $p \sim q$ implies that Bis(p,q,C) holds for some C. Thus, the completeness theorem follows from lemma 6.2-3 (i). $\square$

The inference system in figure 6.2-1 is easily modified for weak bisimulation: simply change the sideconditions of $ML^+$ 2 and 3 (and similarly of $MR^+$ 2 and 3) to:

$$\begin{array}{c} q \xrightarrow{\tilde{a}}_{o} q' \\ (p',q')\,\varepsilon B \end{array} \qquad \text{and} \qquad q \xrightarrow{\tilde{a}}_{o} q'$$

Using proposition 5.0-1 soundness and (restricted) completeness can be proved for the modified system. Similarly, the inference system 6.2-1 can be extended to parameterized strong and weak bisimulation.

The inference system 6.2-1 can also be represented almost directly in PROLOG (see /CM81/), thus giving an (operational based) implementation for constructing bisimulations. Each of the six relations (bisim, closure, matchl, ... ) is represented as a PROLOG predicate and each rule of the inference system is represented as a Horn Clause with sideconditions (of $ML^+$ and $MR^+$) being included as part of the premisses. Sets and set-operations are represented as lists and operations on such.

```
bisim(P,Q,C) :- closure(P,Q,[[P,Q]],C).
closure(P,Q,B,D) :-
        matchl(P,Q,B,C), matchr(P,Q,C,D).
matchl(P,Q,B,C) :-
        derset(P,M), matchl+(P,Q,M,B,C).
matchr(P,Q,C,D) :-
        derset(Q,N), matchr+(P,Q,N,C,D).
matchl+(P,Q,[],B,B).
matchl+(P,Q,[[A,P']|M],B,D) :-
        der(Q,A,Q'), in([P',Q'],B), !,
        matchl+(P,Q,M,B,D).
matchl+(P,Q,[[A,P']|M],B,D) :-
        der(Q,A,Q'), closure(P',Q',[[P',Q']|B],C),
        matchl+(P,Q,M,C,D).
matchr+(P,Q,[],B,B).
matchr+(P,Q,[[A,Q']|N],B,D) :-
        der(P,A,P'), in([P',Q'],B), !,
        matchr+(P,Q,N,B,D).
matchr+(P,Q,[[A,Q']|N],B,D) :-
        der(P,A,P'), closure(P',Q',[[P',Q']|B],C),
        matchr+(P,Q,N,C,D).
```

(figure 6.2-5)

The cut symbol (!) in the second clause for $\text{matchl}^+$ (and similar $\text{matchr}^+$) optimizes the implementation slightly, in that it only allows the third clause for $\text{matchl}^+$ (and similarly for $\text{matchr}^+$) to be used in case q does <u>not</u> have a match for (a,p′) in B.

To complete the implementation clauses for the predicates <u>derset</u> and <u>der</u> must be given such that:

$$\text{derset}(p,M) \Leftrightarrow \text{'}M\text{'} = \{(a,p') \mid p \xrightarrow{a} p'\}$$

and

$$\text{der}(p,a,p') \Leftrightarrow p \xrightarrow{a} p'$$

where ′M′ is the set represented by M. <u>derset</u> is easily derived from <u>der</u> and in the next section we shall show how to represent (a large subset of) CCS and its operational semantics in PROLOG.

Due to the particular order (leftmost-depthfirst) in which PROLOG tries to satisfy goals, non-termination may occur. For example, by prefixing the clauses of figure 6.2-5 with the trivial clause:

```
bisim(P,Q,C) :- bisim(P,Q,C).
```

no goals involving the predicate bisim will terminate. Thus, our previous soundness and completeness theorems only demonstrate <u>partial correctness</u> of the PROLOG program figure 6.2-5. In order to obtain <u>total correctness</u> it must be proved that the PROLOG program always terminates given a goal of the form bisim(p,q,C), where p and q are processes with finite state-transition diagrams. However, given two such processes it is clear that the space of subgoals which is relevant for the goal bisim(p,q,C) is finite. Moreover, the clauses of the PROLOG program define an acyclic dependency between these subgoals (acyclic because the previously defined size functions decrease when the rules or clauses are used backwards). Thus, the leftmost-depthfirst search strategy used by PROLOG will always lead to termination. A more formal proof of termination may be obtained by employing the methods of /Fran84/.

### 6.2.2 CCS in PROLOG.

It is straightforward to represent (a subset of) CCS and its operational semantics in PROLOG. To each CCS process construction we simply introduce a corresponding PROLOG-operator. For obvious reasons we cannot always get the desired standard notation, so here is the PROLOG representation of CCS:

| Construction | PROLOG | Standard Notation |
|---|---|---|
| Inaction | nil | $\mathbb{O}$ |
| Prefix | a;p | $a.p$ |
| Summation | p + q | $p + q$ |
| Parallel | p / q | $p \mid q$ |
| Renaming | p-[a:=b] | $p[a \mapsto b]$ |
| Restriction | p\[a,b] | $p \upharpoonright \{a, \bar{a}, b, \bar{b}\}$ |
| Variable | var(x) | $x$ |
| Recursion | fix(var(x), p) | $\mu x.p$ |

To represent the notion of complementary actions in PROLOG two prefix operators in and out are introduced. Thus, an action is of the form:

action ::= atom | in(atom) | out(atom)

A special action is the atom tau, which represents the unobservable action 1.

In the "Prefix"-rule a can be any action, whereas in the "Renaming" and "Restriction" rules the variables a and b must be atoms. The operational semantics will automatically extend the Renaming/Restriction to all prefixes of the atoms.

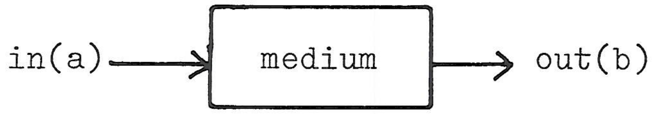
Recursion variables must be prefixed with the operator var in order to distinguish them from actions.

Parentheses are used to make parsing unambiguous; however, to avoid excessive use of parentheses the following operator precedence has been introduced:

Prefix > Restriction > Renaming > Summation > Parallel

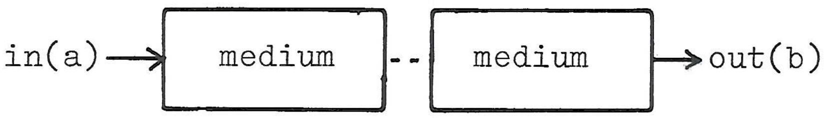
Often large systems will have many occurrences of some subcomponent (e.g. a memory consisting of many identical cells). To avoid having to write out in full the expression for this subcomponent for each occurrence, a <u>let-construct</u> for declaring abbreviations is available, e.g.:

```
let(medium, in(a);out(b);nil).
```




An already declared abbreviation can be used in the declaration of new ones; e.g.:

```
let(delaymed,
        (medium-[b:=c] /
         medium-[a:=c])\[a,b]  ).
```




We shall later see that <u>medium</u> and <u>delaymed</u> are weak bisimulation equivalent.

The derivation relation $\rightarrow$ for the above subset of CCS is represented as a PROLOG predicate <u>der</u> with a one-to-one correspondence between the inference rules for $\rightarrow$ and the PROLOG clauses for <u>der</u>; e.g.:

| <u>Inference rule</u> | <u>PROLOG clause</u> |
|---|---|
| $a.p \xrightarrow{a} p$ | `der(A;P , A , P).` |
| $\dfrac{p \xrightarrow{a} r}{p + q \xrightarrow{a} r}$ | `der(P+Q , A , R) :- der(P,A,R).` |
| $\dfrac{p\{\mu x.p/x\} \xrightarrow{a} q}{\mu x.p \xrightarrow{a} q}$ | `der(fix(var(X),P),A,Q) :- subst(fix(var(X),P),var(X),P,R), der(R,A,Q).` |

where subst is an auxiliary PROLOG predicate such that subst(S,var(X),U,V) holds iff $V = U\{S/var(X)\}$. By the way: it seems that many Structured Operational Semantics (see /P181/) have a direct implementation in PROLOG. The operational semantics of CCS is of course just an especially simple SOS.

### 6.2.3 Using the system.

Combining the representation of CCS in PROLOG from the previous section with the PROLOG-program for constructing (weak) bisimulations from section 6.2.1 results in a system for proving (weak) bisimulation equivalences between CCS processes. We shall demonstrate the usefulness of the system for weak bisimulation through three examples.

First, consider the two processes medium and delaymed declared in the previous section.

```
| ?- bisim(medium,delaymed).

1      medium
           delaymed       [2,4]

2      out(b);nil
           (nil-[b:=c]/out(b);nil-[a:=c])\[a,b]       [3]

3      nil
           (nil-[b:=c]/nil-[a:=c])\[a,b]       []

4      out(b);nil
           (out(b);nil-[b:=c]/medium-[a:=c])\[a,b]       [3,2]


yes
| ?-
```

We see that the goal bisim(medium,delaymed) succeeds, and hence that medium $\approx$ delaymed. The resulting bisimulation contains four (numbered) pairs of processes, (medium,delaymed) being one of them. The list of numbers following each pair indicates its successorpairs and is handy if one wants to check that the set of pairs really constitutes a bisimulation.

As our next example we consider the Simple Scheduler from section 5.5. We declare abbreviations for an individual cell, the scheduler of size 3 and its specification:

```
?- let(cell, fix(var(x), in(a);w;out(b);var(x)) ).


?- let(sch,    (  w;out(b);cell-[a:=c1]-[b:=c2]-[w:=w1] /
                           cell-[a:=c2]-[b:=c3]-[w:=w2] /
                           cell-[a:=c3]-[b:=c1]-[w:=w3]
               ) \[w1,w2,w3]    ).


?- let(spec, fix(var(x), w1;w2;w3;var(x)) ).
```

```
| ?- bisim(spec,sch).

spec
     sch

w2;w3;spec
     (cell-[a:=c1]-[b:=c2]-[w:=w1] /
      w;out(b);cell-[a:=c2]-[b:=c3]-[w:=w2] /
      cell-[a:=c3]-[b:=c1]-[w:=w3])\[w1,w2,w3]

w3;spec
     (cell-[a:=c1]-[b:=c2]-[w:=w1] /
      cell-[a:=c2]-[b:=c3]-[w:=w2] /
      w;out(b);cell-[a:=c3]-[b:=c1]-[w:=w3])\[w1,w2,w3]

spec
     (cell-[a:=c1]-[b:=c2]-[w:=w1] /
      cell-[a:=c2]-[b:=c3]-[w:=w2] /
      out(b);cell-[a:=c3]-[b:=c1]-[w:=w3])\[w1,w2,w3]

w3;spec
     (cell-[a:=c1]-[b:=c2]-[w:=w1] /
      out(b);cell-[a:=c2]-[b:=c3]-[w:=w2] /
      cell-[a:=c3]-[b:=c1]-[w:=w3])\[w1,w2,w3]

w2;w3;spec
     (out(b);cell-[a:=c1]-[b:=c2]-[w:=w1] /
      cell-[a:=c2]-[b:=c3]-[w:=w2] /
      cell-[a:=c3]-[b:=c1]-[w:=w3])\[w1,w2,w3]


yes
| ?-
```

The goal bisim(spec,sch) succeeds and hence spec$\approx$ sch as expected. Note, that the three abbreviations are also used in the display of the final bisimulation.

The final example we consider, in a slightly simplified version, comes from a set of Lecture Notes used by Robin Milner to accompany a course on CCS and involves the representation of a workshop comprising two men, a mallet and a hammer. In our simplified version a man can use either a hammer or a mallet to perform a job. gh and ph represent the actions of getting and putting a hammer, likewise gm and pm for mallet.

```
?- let(man,
        fix(var(x), injob;(in(gh);out(ph);outjob;var(x) +
                           in(gm);out(pm);outjob;var(x))
            )
       ).
```

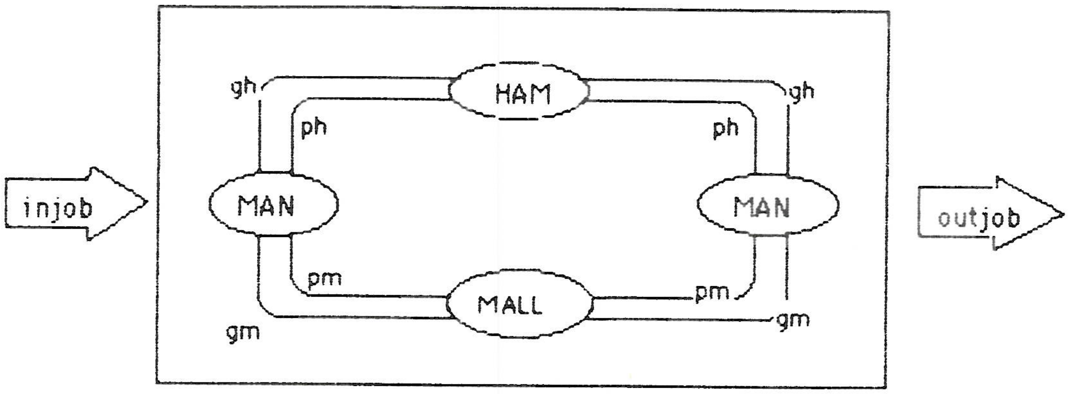
The behaviour of the hammer and mallet are extremely simple:

```
?- let(hammer,
        fix(var(x), out(gh);in(ph);var(x)) ).

?- let(mallet,
        fix(var(x), out(gm);in(pm);var(x)) ).
```

The two men together with the tools, the hammer and the mallet, is put together to form a CLOSEDSHOP as follows:

```
?- let(closedshop,
        ( man / man / hammer / mallet )\[injob,outjob]  ).
```




The specification for closedshop is given by the following process donothing:

```
?- let(one,
       fix(var(x), injob;outjob;var(x) +
                   outjob;injob;var(x) ) ).
?- let(donothing,
       injob;one  ).
```

The following shows that the goal bisim(donothing,closedshop) succeeds producing a bisimulation containing 23 pairs. Thus we can indeed conclude that donothing≈closedshop. A "handmade" proof of the closedshop example (in its full version) can be found in /San82/.

```
| ?- bisim(donothing,closedshop).

1      donothing
           closedshop      [2,6,16]

2      one
           (outjob;man /
            man /
            hammer /
            mallet)\[injob,outjob]      [3,1,15]
```

```
3      outjob;one
           (outjob;man /
            outjob;man /
            hammer /
            mallet)\[injob,outjob]      [2,4]

4      one
           (man /
            outjob;man /
            hammer /
            mallet)\[injob,outjob]      [3,1,5]

5      outjob;one
           (in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            outjob;man /
            hammer /
            mallet)\[injob,outjob]      [4,6,21,12]

6      one
           (in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            man /
            hammer /
            mallet)\[injob,outjob]      [3,1,7,22,13]

7      outjob;one
           (in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            hammer /
            mallet)\[injob,outjob]      [4,8,18,23,14]

8      outjob;one
           (in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            out(ph);outjob;man /
            in(ph);hammer  /
            mallet)\[injob,outjob]      [4,5,9]

9      outjob;one
           (out(pm);outjob;man /
            out(ph);outjob;man /
            in(ph);hammer /
            in(pm);mallet)\[injob,outjob]      [4,10,12]

10     outjob;one
         (outjob;man /
          out(ph);outjob;man /
          in(ph);hammer /
          mallet)\[injob,outjob]      [4,11,3]

11     one
         (man /
          out(ph);outjob;man /
          in(ph);hammer /
          mallet)\[injob,outjob]      [3,1,8,4]
```

```
12      outjob;one
           (out(pm);outjob;man   /
            outjob;man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [4,13,3]

13      one
           (out(pm);outjob;man /
            man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [3,1,14,2]

14      outjob;one
           (out(pm);outjob;man /
            in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [4,15,9]

15      outjob;one
           (outjob;man /
            in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            hammer /
            mallet)\[injob,outjob]      [4,16,10,20]

16      one
           (man /
            in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            hammer /
            mallet)\[injob,outjob]      [3,1,7,11,17]

17      one
           (man /
            out(pm);outjob;man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [3,1,18,4]

18      outjob;one
           (in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            out(pm);outjob;man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [4,5,19]

19      outjob;one
           (out(ph);outjob;man /
            out(pm);outjob;man /
            in(ph);hammer /
            in(pm);mallet)\[injob,outjob]      [4,20,21]

20      outjob;one
           (outjob;man /
            out(pm);outjob;man /
            hammer /
            in(pm);mallet)\[injob,outjob]      [4,17,3]
```

```
21      outjob;one
           (out(ph);outjob;man /
            outjob;man /
            in(ph);hammer /
            mallet)\[injob,outjob]     [4,22,3]

22      one
           (out(ph);outjob;man /
            man /
            in(ph);hammer /
            mallet)\[injob,outjob]     [3,1,23,2]

23      outjob;one
           (out(ph);outjob;man /
            in(gh);out(ph);outjob;man+in(gm);out(pm);outjob;man /
            in(ph);hammer /
            mallet)\[injob,outjob]     [4,15,19]
```

## 6.3 CONCLUDING REMARKS, FUTURE AND RELATED WORK

In the previous section 6.1 of this chapter we have shown that the various notions (parameterized/unparameterized, strong/weak) of bisimulation equivalence are all polynomial time decidable for processes with finite state-transition diagrams. Based on an alternative decision procedure, a PROLOG-system for constructing (parameterized/unparameterized, strong/weak) bisimulations for finite CCS expressions over regular expressions has been implemented (and verified) in section 6.2. This alternative decision procedure is related to a similar algorithm presented in /San 82/: both algorithms will, given two processes p and q, try to construct a <u>minimal</u> bisimulation containing the pair (p,q). However, the algorithm in /San82/ is significantly less general than ours: besides the necessary condition of p and q having finite state-transition diagrams, the process p must be rigid and deterministic (see /San82,Mil80/) and the process q must be non-divergent in the sense that none of its derivatives can perform an infinite sequence of 1-actions. Also, neither a correctness proof nor an implementation is provided in /San82/.

Though the PROLOG-system presented in section 6.2 is rather simple it serves the purpose of demonstrating the achievability and potential uses of automatic tools. However, lots of work remains to be done in developing more satisfactory future tools. One main disadvantage of the PROLOG-system presented is that it only allows processes with finite state-transition diagrams. In any realistic example this assumption is likely to be violated: Often process expressions are indexed or parameterized with elements from some infinite set (the natural numbers in the Simple Scheduler example in section 5.5, natural numbers and sets of natural numbers in the scheduling example of /Mil80/ chapter 3). In order to deal with such expressions the system must be able to prove properties about the parameters used. Depending on the parameters used and the complexity of properties the system is required to deal with, it may well turn out that the equivalence problem for indexed/parameterized process expressions becomes undecidable. Thus, for future systems, it might be more relevant to think in terms of checking and guiding equivalence proofs (à la LCF /GMW79/) instead of automatically producing such proofs.

A small, first system of this type has been developed in PROLOG by K.V.S. Prasad, /Pr?/. His system is quite similar to ours except that it instead of constructing bisimulations will check whether a given (by the user) binary relation on processes constitutes a (weak) bisimulation. Being essentially a proof checker (viewing a bisimulation as a proof) the system is able to deal with certain types of parameterized expressions. Parts of the correctness proof of a simple fault tolerant system /Pr84/ have been checked by the system.

Another proof checking system has been developed in Lisp by Nick Traub /Tr83/. In contrast to Prasad's and our systems, which both are based on the operational semantics of CCS, Traub's system allows the user to manipulate (CIRCAL) expressions using algebraic laws (for CIRCAL see /M82/).

Maybe future systems should support both equivalence proofs obtained by applying algebraic laws and equivalence proofs obtained by exhibiting appropriate bisimulations.

So far we have concentrated on systems for proving (weak) equivalences between processes. However, in order for a system to assist in (weak) parameterized equivalence proofs and support the associated proof methodology developed in this thesis, it seems necessary for the system also to know about the following:

- Contexts (and their operational semantics)
- How contexts transform environments.

It seems quite feasible to extend our PROLOG-system with such "information".

Finally, we will mention the possibility of having systems for verifying or assisting in verifying <u>partial properties of processes</u>, specifically <u>modal properties</u> of processes. Such a system could be either operationally based (i.e. using directly the definition of the satisfaction relation) or based on the proof systems which exist for various subsets of CCS, SCCS /St83,St85,W85,W85B/. However, it seems that the (socalled Hennessy-Milner) Modal Logic (see section 2.1.3) which is currently being used is, from a pragmatic point of view, not expressive enough. For instance will the satisfaction of any modal formula from this logic only depend on a (certain) finite part of the processes. Though, it seems that this deficiency can be remedied by adding recursion to the modal logic (à la Dexter Kozen's $L\mu$-calculus /Ko82/), more work is needed in finding a practically satisfactory logic.

# CHAPTER 7

# CONCLUSION & FUTURE WORK

A thorough investigation of a **parameterized version of** bisimulation equivalence has been presented in this thesis. The parameterized version proposed has been shown to enjoy a large number of pleasant properties and we are therefore confident that the version is indeed a natural one. It is hoped that the results proved in this thesis will provide a useful repertoire of techniques for making hierarchic verification of concurrent systems an easier task. The Simple Scheduler example considered demonstrates the intended use of the results presented. We believe that the techniques introduced will be especially useful for larger examples, where obviously the need for hierarchic decomposition is greater. Evidence of this potential usefulness for larger systems has recently been given by Robin Milner, who has indicated how to apply the techniques of this thesis to the Alternating Bit Protocol.

More specifically, the <u>main achievements of this thesis</u> are:

<u>1.</u> We have defined a parameterized version of bisimulation equivalence with so-called environments used as parameters. The resulting parameterized equivalence is

shown to have all the properties expected in chapter 1. As Main Theorems a characterization of the discrimination ordering between environments, and a maximal environment construction has been presented. Also, a modal characterization of parameterized bisimulation equivalence is given.

2. Results showing how contexts transform modal formulas and environments have been given. These results constitute the main tools provided by this thesis for hierarchic verification of concurrent systems. In order to facilitate the above investigation an abstract (and new) semantic account of contexts as action transducers has been introduced. Besides being of independent interest, this semantic account has made our results general in the sense that they are applicable to (almost) all process constructions.

3. The results from 1 and 2 have been extended to a similarly parameterized version of the (perhabs more interesting) <u>weak</u> bisimulation equivalence, $\approx$. The main obstacle in performing this extension has been that $\approx$ is not preserved by all contexts. However, based on the semantic description of contexts as action transducers, conditions insuring the preservation of $\approx$ have been given. These conditions ought also to be of independent interest. The intended use of the (extended) results in verification has been illustrated through an example.

4. Complete axiomatizations for parameterized bisimulation equivalence have been given for finite and regular processes and environments.

5. We have shown that parameterized bisimulation equivalence is polynomial time decidable for regular processes and environments, thus generalizing the existing polynomial time complexity result for (unparameterized) bisimulation equivalence.

6. Finally, a PROLOG system for constructing bisimulations over CCS expressions has been implemented, verified and demonstrated.

There are at least three main areas in which future work can be done. Having developed a theory of parameterized bisimulation equivalence it is imperative that we test it extensively through practical applications. Only this will enable us to determine whether the developed theory is succesfull in shortening correctness proofs. The Simple Scheduler considered in this thesis and the Alternating Bit Protocol investigated by Robin Milner indicate the potential usefulness of the theory but much more practical experience is obviously needed before any final judgement can be made. The Alternating Bit Protocol is a member (the simplest) of a whole class of protocols known as Sliding Window Protocols. These protocols therefore seem natural next candidates for our proof techniques. The process of gaining more practical experience would also help us in finding more advantageous ways of utilizing our results in correctness proofs and might even create a demand for results slightly different from those provided by this thesis. From the maximal environment construction and the weakest inner environment construction we know that the parameterized equivalence:

(*) $$C[p] \sim_e C[q]$$

can be reduced to the simulation problem:

(**) $$wie(C,e) \leqslant /p,q/$$

Using the algebraic laws presented in this thesis we might be able to calculate $wie(C,e)$ and $/p,q/$. However, the calculation of $/p,q/$ will depend on all of p's and q's behaviours regardless of whatever restrictions C may impose on p and q. Similarly, the calculation of $wie(C,e)$ is based on the full behaviour of C with no considerations of the restrictions the processes p and q may impose on C. Obviously, we would like to deduce the simulation in (**) without an explicit calculation of wie(C,e) and /p,q/. By replacing (**) with a parameterized equivalence $p \sim_f q$ where f is an environment satisfying wie(C,e)$\leqslant$f, the calculation of /p,q/ can be avoided. However, this still leaves the problem of deciding wie(C,e)$\leqslant$f without calculating wie(C,e). Our experience with the Simple Scheduler as well as the Alternating Bit Protocol suggests that this may more easily be done by appealing directly to the operational semantics of wie(C,e) (i.e. e[C]) instead of using the algebraic laws for wie(C,e). However, this remains to be confirmed by more examples.

Through more examples we may also find that certain types of environments are more useful than others. Judged by the few examples already investigated it seems that language environments are especially convenient and frequent. Also, it seems that the type of language environments we encounter in our correctness proofs are themselves special: they are almost universal language environments except for a few restrictions on certain key actions; e.g. the action a most occur before any b action and between any two a actions there are at least one occurrence of b. In order to emphasise these (key) restrictions it may well be more convenient to adopt some other notation for language environments than the regular expression notation used in this thesis. We expect some Linear Temporal Logic may prove useful for this purpose. However, irrespective of what notation is used, it is crucial to maintain an operational semantics of environments in order for the parameterized bisimulation technique to be at our disposal.

During the process of gaining more practical experience by applying our techniques to larger examples, the availability of computer assistance will become essential. This is another area for future work. Our PROLOG system provides a first such automatic tool but lots of work remains to be done in order to develop more satisfactory tools. At present the PROLOG system will simply terminate with failure when given two processes, p and q, not equivalent. This is rather uninformative. Obviously the user would like to be given a reason for why the processes are not equivalent so that proper alterations of either process can be done. From the modal characterization of bisimulation equivalence we know that there exist some modal formula $F$ such that $p \models F$ and $q \not\models F$ in case p and q are inequivalent. We may view $F$ as a reason for or an explanation of why p and q are not equivalent. It seems possible to extend the GENERALIZED PARTITIONING algorithm from section 6.1 so that it returns a modal formula $F$ with $p \models F$ and $q \not\models F$ when $p \not\sim q$ :

> Throughout the execution each block $B_j$ of the current partitioning is associated with a modal formula $F_j$ such that $p \models F_j$ for all p in $B_j$ and $p \not\models F_j$ whenever p is not in $B_j$. When (and if) the two processes p and q under consideration are seperated into two different blocks $B_i$ and $B_j$ (which will happen if $p \not\sim q$) we may simply return either of the modal formulas $F_i$ and $\neg F_j$. The single block of the initial partitioning is associated with the modal formula $Tr$. When, during execution, a block $B_i$ of the current partitioning is split into two blocks $B_i'$ and $B_i''$ with respect to some function $f_a$ and some block $B_j$ (i.e. $q \in B_i'$ iff $f_a(q) \cap B_j \neq \emptyset$ and $B_i'' = B_i - B_i'$) we associate with $B_i'$ and $B_i''$ the modal formulas $F_i' = F_i \wedge \langle a \rangle F_j$ and $F_i'' = F_i \wedge \neg \langle a \rangle F_j$. This will maintain the invariant property of the modal formulas.

Obviously, we are also interested in developing tools which can assist in parameterized equivalence proofs and support the associated proof methodology developed in this thesis.

It seems necessary for such a tool to know about contexts and their operational semantics and how to derive the operational behaviour of a combined environment $e[C]$ from those of e and C. It is quite feasible to extend our PROLOG system with such "information".

The motivation for context dependent equivalences is a general one and not only applicable to bisimulation equivalence. Thus, a third area for future work is concerned about extending the results of this thesis to other equivalences, especially the equivalences mentioned in chapter 1 (failure and testing equivalence). It seems natural to try and maintain the use of environments as parameters. The various alternative (and recursive) defintions of failure and testing equivalence given in /Ni85/ ought to be a useful guide for how precisely to define their parameterized versions. Other possibilities for future research include an extension of the Main Theorem 2.4-20 to image-infinite environments.

In conclusion, it has become clear that, while this thesis provides a thorough investigation of a parameterized bisimulation equivalence and indicates its use in correctness proofs, there is still future work to be done in applying the techniques and results of this thesis, in developing tools for computer aided verification and in extending the results of this thesis to other equivalences.

# REFERENCES


/A83/ P.Aczel: An Introduction to Inductive Definitions, North-Holland, In the Handbook of Mathematical Logic, ed. J. Barwice, 1983.

/AHU74/ Aho, Hopcroft and Ullman: The Design and Analysis of Computer Algorithms, Addison-Wesley, 1974.

/AU72/ Aho and Ullman: The Theory of Parsing, Tranlation, and Compiling, Prentice-Hall, Series in Automatic Computation, 1972.

/BK83/ Barringer and Kuiper: Towards the Hierarchical, Temporal Logic, Specification of Concurrent Systems, Presented at STL/SERC Workshop on the Analysis of Concurrent Systems, Cambridge, 1983.

/BKPn84/ Barringer, Kuiper and Pnueli: Now you may compose Temporal Logic Specifications, ACM Symposium on Theory of Computing, pp. 51-63, 1984.

/B-A82/ Ben-Ari: Principles of Concurrent Programming, Prentice-Hall International, 1982.

/BerKl84/ Bergstra and Klop: A Complete Inference System for Regular Processes with Silent Moves, Centre for Math. and Comp. Sc., Amsterdam Report CS-R8420,1984.

/BlTr85/ Bloom and Troeger: A Logical Characterization of Observation Equivalence, TCS vol. 35, no. 1, 1985.

/Bou84/ Boudol: Notes on Algebraic Calculi of Processes, INRIA-Shophia-Antipolis, 1984.

/Bro83/ S.Brookes: On the Relationship of CCS and CSP, LNCS 154, 1983.

/Bro83B/ S.Brookes: A Model for Communicating Sequential Processes, Ph.D Thesis, University of Oxford, 1983.

/BroR83/ S.Brookes and W.Rounds: Behavioural equivalence relations induced by programming logics, LNCS 154, pp. 97-108, 1983.

/Bro85/ S.Brookes: An axiomatic treatment of a Parallel Programming Language , To appear in: 1985 Logics of Programs Conference, Brooklyn, LNCS, 1985.

/Con71/ J.H.Conway: Regular Algebra and Finite Machines, Chapman and Hall, Math. Series, 1971.

/CM81/ Clocksin and Mellish: Programming in Prolog, Springer-Verlag, 1981.

/Da81/ D. Van Dalen: First Draft for Philosophical Logic, University Utrecht, Department of Mathematics, Preprint nr. 209, September 1981.

/Dij76/ E.Dijkstra: A discipline of programming, Prentice-Hall Series in Automatic Computation, 1976.

/EK74/ M.H.Emden and R.A.Kowalski: The Semantics of Predicate Logic as a Programming Language, Memo no 73, Edinburgh University, Artificial Intelligence.

/Fran84/ Francez, Grumberg, Katz and Pnueli: Proving Termination of PROLOG Programs.

/GJ79/ Garey and Johnson: Computers and Intractability, A Guide to the Theory of NP-Completeness, Freman & Co, Bell Laboratories, Murray Hill, New Jersey, 1979.

/GMW79/ M.Gordon, R.Milner and C.Wadsworth: Edinburgh LCF, LNCS 78, 1979.

/Go79/ R.Goldblatt: Topoi: The Categorical Analysis of Logic, North- Holland, 1979.

/Gor79/ M.Gordon: The Denotational Description of Programming Languages, Springer-Verlag, 1979.

/GrSif84/ S.Graf and J.Sifakis: A modal characterization of observational congruence on finite terms of CCS, LNCS 172, pp. 222-234, 1984.

/GrSif85/ S.Graf and J.Sifakis: A Logic for the Description of Nondeterministic Programs and Their Properties, Technical Report RR no 511, 38402, St. Martin D'Heres, 1985.

/Hen81/ M.Hennessy: A term model for Synchronous Processes, Internal Report, University of Edinburgh, CSR-77-81, 1981.

/Hen83/ M.Hennessy: A Model for Nondeterministic Machines, Internal Report, University of Edinburgh CSR-135-83, 1983.

/HenPl80/ M.Hennessy and G.Plotkin: A term model for CCS, Proceedings of 9th MFCS Conference, LNCS 88, 1980.

/HenMil80/ M.Hennessy and R.Milner: On Observing Nondeterminism and Concurrency, Proceedings of 7th ICALP, LNCS 85, 1980.

/HenMil83/ M.Hennessy and R.Milner: Algebraic Laws for Nondeterminism and Concurrency, Journal of the Association for Computing Machinery, pp. 137-161, 1985.

/HenSt84/ M.Hennessy and C.Stirling: The power of the future perfect in program logics, LNCS 176, pp. 301-311, 1984.

/HoBroR84/ C.Hoare, S.Brookes and A.Rounds: A Theory of Communicating Sequential Processes, Journal of the Association for Computing Machinery, pp. 560- , 1984.

/Ho78/ C.Hoare: Communicating Sequential Processes, CACM 21, vol 8, 1978.

/Ho81/ C.Hoare: A Model for Communicating Sequential Processes, Technical Monograph Prg-22, Computing Laboratory, University of Oxford, 1981.

/Ho84/ C.Hoare: Communicating Sequential Processes, Prentice-Hall, 1985.

/HU79/ J.Hopcroft, J.Ullman: Introducting to Automata Theory, Languages and Computation, Addison-Wesley, 1979.

/Jo81/ C.Jones: Development Methods for Computer Programs including a Notion of Interference, Ph.D Thesis, Wolfson College, 1981.

/Jo83/ C.Jones: Tentative Steps Toward a Development Method for Interfering Programs, TOPLAS 1983, vol 5, no 4, 1983.

/KaSm83/ P.C.Kannellakis and S.A.Smolka: CCS Expressions, finite state processes, and three problems of equivalence, 1983.

/K75/ R.Keller: A fundamental theorem of asynchronous parallel computation, LNCS 24, 1975.

/Ko82/ D.Kozen: Results on the Propositional $\mu$-Calculus, 9th ICALP, Aarhus, LNCS 140, 1982.

/La85/ K.G.Larsen: A Context Dependent Equivalence between Processes, 12th ICALP, LNCS 194, pp. 373-382, 1985. Full version to appear in TCS.

/MaPn83/ Z.Manna and A.Pnueli: How to cook a temporal proof system for your pet language, Proceedings of Principles of Programming Languages, pp. 141-154, 1983.

/MaPn82/ Z.Manna and A.Pnueli: Verification of concurrent programs: the temporal framework, in: The Correctness Problem in Computer Science, ed. Boyer and Moore, Academic Press, 1982.

/MaW84/ Z.Manna and P.Wolper: Synthesis of Communicating Processes from Temporal Logic Specifications, ACM TOPLAS, vol 6 no 1, 1984.

/Maz77/ A.Mazurkiewicz: Concurrent Processes and their Syntax, DAIMI-PB-78, Aarhus University, 1977.

/M82/ G.Milne: CIRCAL: A Calculus for Circuit Description, Integration 1, 2 and 3, 1983.

/M85/ G.Milne: Simulation and Verification: Related Techniques for Hardware Analysis, 7th International Symposium on CHDL, Tokyo, North-Holland, 1985.

/MMil79/ G.Milne and R.Milner: Concurrent Processes and their Syntax, Journal of ACM, vol 26, no 2, 1979.

/Mil71/ R.Milner: An Algebraic Definition of Simulation between Programs, in: Proceedings of 2nd International Conference on Artificial Intelligence, British Comp. Soc., 1971.

/Mil73B/ R.Milner: An Approach to the Semantics of Parallel Programs, Proceedings, Convegno di Information, March, Pisa, 1973.

/Mil75/ R.Milner: Processes: A Mathematical Model of Computing Agents, in: H.Rose, J.Shepherdson, Logic Colloquium '73, North-Holland, pp. 157-174, 1975.

/Mil78/ R.Milner: Synthesis of Communicating Behaviour, MFCS, LNCS 64, 1978.

/Mil80/ R.Milner: A Calculus of Communicating Systems, LNCS 92, 1980.

/Mil79/ R.Milner: Flowgraphs and Flow Algebra, JACM 26(4), 1979.

/Mil79B/ R.Milner: An Algebraic Theory for Synchronization, LNCS 67, 1979.

/Mil81/ R.Milner: A modal characterization of observable machine-behaviour, LNCS 112, 1981.

/Mil82/ R.Milner: A Complete Inference System for a Class of Regular Behaviours, Internal Report, University of Edinburgh, CSR-111-82, 1982.

/Mil83/ R.Milner: Calculi for Synchrony and Asynchrony, TCS 25, pp.267-310, North-Holland, 1983.

/Mil84/ R.Milner: Lectures on a Calculus for Communicating Systems, To appear in LNCS, Summerschool Marktoberdorf, 1984.

/Mo56/ E.F.Moore: Gedanken-experiments on Sequential Machines, in: Automata Studies, ed. C.Shannon, J.McCarthy, Princeton University Press, pp. 129-153, 1956.

/NiHen82/ R. de Nicola and M.Hennessy: Testing Equivalences for Processes, in: LNCS 154, 1983, Full version in TCS vol. 34, pp. 83-133, 1984.

/Ni85/ R. de Nicola: Testing Equivalences and Fully Abstract Models for Communicating Processes, Ph.D. Thesis, University of Edinburgh, 1985.

/OHo83/ E.Olderog and C.Hoare: Specification oriented semantics for communicating processes, LNCS 154, 1983.

/P81/ D.Park: A predicate transformer for weak fair iteration, Proceedings, 6th IBM Symposium on mathematical foundation of computer science, Hakene, Japan, 1981.

/P81B/ D.Park: Concurrency and automata on infinite sequences, LNCS 104, 1981.

/Pet80/ C.Petri: Concurrency, in: Net Theory and Applications, LNCS 84, 1980.

/Pl76/ G.Plotkin: A Powerdomain Construction, SIAM J. on Computing, no. 5, 1976.

/Pl81/ G.Plotkin: A Structural Approach to Operational Semantics, DAIMI-FN-19, Aarhus University, Computer Science Department, Denmark, 1981.

/Pl82/ G.Plotkin: An Operational Semantics for CSP, in Proceedings of the IFIP WG 2.2 Working Conference on Formal Description of Programming Concepts II, 1982.

/Pn85/ A.Pnueli: Linear and branching structures in the semantics and logics of reactive systems, 12th ICALP, LNCS 194, 1985.

/Pr84/ K.V.S.Prasad: Specification and Proof of a Simple Fault Tolerant System in CCS, Internal Report, University of Edinburgh, CSR-178-84, 1984.

/Pr?/ K.V.S.Prasad: Forthcoming Ph.D Thesis, University of Edinburgh.

/Sal66/ A.Salomaa: Two Complete Axiom Systems for the Algebra of Regular Events, JACM, vol 13, no 1, pp. 158-169, 1966.

/San82/ M.Sanderson: Proof Techniques for CCS, Ph.D Thesis, University of Edinburgh, CST-19-82, 1982.

/Sif82/ J.Sifakis: A unified approach for studying the properties of transition systems, TCS pp. 227-258, 1982.

/Sim85/ R. de Simone: Higher-level Synchronizing Devices in MEIJE-CCS, Rapports de Recherche, INRIA, no 360, jan 1985.

/St83/ C.Stirling: A Proof Theoretic Characterization of Observational Equivalence, in Proceedings of FCT-TCS Bangalore, 1983, to appear in TCS.

/St84/ C.Stirling: A Complete Proof System for a Subset of SCCS, LNCS 185, 1985. To appear in CAAP'85.

/St85/ C.Stirling: A Complete Compositional Modal Proof System for a Subset of CCS, 12th ICALP, LNCS 194, 1985. Full version to appear in TCS.

/Stoy77/ J.Stoy: Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory, The MIT Press, 1977.

/Smy78/ M.Smyth: Power Domains, Journal of Computers and Systems Science, Vol. 2, pp. 23-36, 1978.

/Ta55/ A.Tarski: A Lattice-Theoretical Fixpoint Theorem and its Applications, Pacific Jounal of Math. 5, 1955.

/Tr85/ N.Traub: A Lisp based CIRCAL Environment, Internal Report, University of Edinburgh, CSR-152-83, 1983.

/W82/ G.Winskel: Event Structure Semantics of CCS and related Languages, ICALP 82, LNCS 140, 1982.

/W85/ G.Winskel: A Complete Proof System for SCCS with Modal Assertions, Technical Report, Computer Laboratory, University of Cambridge, 1985.

/W85B/ G.Winskel: _On the Composition and Decomposition of Assertions_, Technical Report, Computer Laboratory, University of Cambridge, 1985.

1986

R-RAPPORTER

R 86-1 Ryszard Janicki:
A Formal Semantics for Concurrent Systems with a Priority Relation. (Revised Version.)

R 86-2 Hanns-Georg Leimer:
Triangulated Graphs With Marked Vertices

R 86-3 Ryszard Janicki & Eugeniusz Eberbach:
Fixedpoint Semantics of Recursive Co-routine Vectors

R 86-4 Johan Brøndum:
Digital fasemodulation

R 86-5 Gunther Johannsen:
Comparison of Two Models for Active Fibres

R 86-6 Lars Døvling Andersen & A.J.W. Hilton:
Extending Edge-Colourings of Complete Graphs and Independent Edges

R 86-7 Preben D. Vestergaard:
Finite and Infinite Graphs Whose Spanning Trees Are Pairwise Isomorphic

R 86-8 Paul Dalsgaard, Svend Danielsen & Ole Hansen:
Talestyret kommunikationsterminal - Statusrapport II

R 86-9 Finn V. Jensen & Kim G. Larsen:
Principles for Simultaneous Induction in Contextfree Languages.

R 86-9A Flemming Nielson:
Strictness Analysis and Denotational Abstract Interpretation

R 86-10 Lars Døvling Andersen, Herbert Fleischner & Bill Jackson:
Removable Edges in Cyclically 4-edge-connected Cubic Graphs

R 86-11 E. Susanne Christensen: Estimation in Exponential Families Based on a Dual Likelihoodfunction

1987

## R-RAPPORTER

R 87-1 Leif Kjær Jørgensen:
Some Probabilistic and Extremal Results on Subdivisions and Odd Subdivisions of Graphs

R 87-2 Paul Dalsgaard & Børge Lindberg:

R 87-3 Hanne Riis Nielson & Flemming Nielson:
Two-Level Semantics and Code Generation

R 87-4 Preben Dahl Vestergaard:
On Graphs with Prescribed Spanning Subgraphs