Aalborg University Centre
Institute of Electronic Systems
M. Sc. Thesis in Computer Science

An Extended Bisimulation Induced by a Preorder on Actions

by
Bent Thomsen

Abstract

In this paper we investigate how a preorder on actions induces a preorder on processes in a model of concurrent systems based on labelled transition systems.

We use an extension of bisimulation [Par 81, Mil 83] to induce a preorder on processes from the preorder on actions.

The language for defining processes is taken to be an extract of current variants of CCS [Mil 80].

Sound and complete proof systems for the induced preorders on processes are given for three sublanguages: one for finite nondeterministic processes, one for general finite terms and one for regular expressions.

A preorder on processes expressing concreteness relationship between partial specifications are induced by a particular preorder on actions. This is investigated and used to verify the concurrent alternating bit protocol [Koy 85].

The concept of abstract interpretation [Cou 79], wellknown from traditional data flow analysis, is introduced to the framework of CCS, and it is shown how to use this concept together with an extension of the notion of bisimulation. Finally methods for determining properties of processes by abstract interpretation are presented.

Preface

The work presented in this paper represents my M. Sc. thesis in computer science. It is the physical result of a working period of about five months. During this period Dr. Kim G. Larsen has been my supervisor.

The presentation demands a knowledge of current research within the field of concurrency as in [Mil 80] and [Mil 83].

References in this paper are given by [Author Year].

Propositions, Definitions, Lemmas and Theorems are separately numbered sequentially in each chapter. Those proven in appendix have the same numbers as in the report.

Acknowledgements

It is hard to express my gratitude to the people who have helped me during these past months, but I want to thank all of them very much.

A special thank to my superviser Dr. Kim G. Larsen for his enthusiastic supervision and fruitful commenting.

I hope Lone Leth already knows my gratitude to her for all she has done for me.

Aalborg University Centre 12/1 1987.

Bent Thomsen.

Contents

Introduction

For many years investigations on modular or hierarchical approaches to development, implementation and verification of concurrent systems have been going on.

One branch of concurrent system development is protocol implementation and verification.

A protocol is a behaviour implementing data transmission from a sender a to a receiver b via a channel or a medium k. The channel is assumed to be faulty in various numbers of ways.

One family of protocols is the alternating bit protocols. The protocols consist of a number of communicating finite state machines, including a sender and a receiver process. The sender process receives data from a user, adds a sequence number and sends it to the receiver process, which answers by sending back an acknowledgement. The receiver process receives the message, peels off the sequence number and outputs the message to a user. The sequence number of new messages takes the values 0 and 1 alternately. The purpose of the sequence number is to detect loss of messages or acknowledgements. To recover from loss of messages or acknowledgements the sender process retransmits the message. There are at least two possible ways of holding information about the sequence number; the one being a variable, demanding for a value parsing mechanism, the other being a change of state in the sender and receiver processes. We choose the last since we do not consider value parsing in this thesis. Also the message sent is abstracted to a signal of sending since the actual message is inessential.

We now turn our attention to a specific member of the family of alternating bit protocols called the concurrent alternating bit protocol. This protocol was investigated by C. P. J. Koymans and J. C. Mulder in [Koy 85].

The concurrent alternating bit protocol (CABP) has the architecture displayed in figure .1.

a takes in a message and sends it via k to b. b outputs the message and sends an acknowledgement to c which transmits it to d via l. d communicates the acknowledgement to a. The channels k and l are assumed to be faulty in the sense that a message may be corrupted i.e. changed into an error message or it may be lost. Together a, k and b implement a sender/receiver process called p, and c, l and d a control process called q. In [Koy 85] it is verified that the

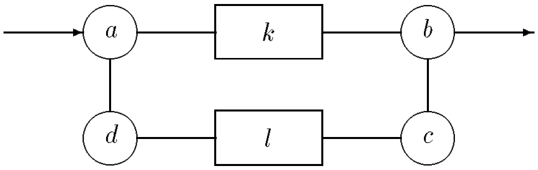


Figure 0.1: Architecture of the CABP.

process p enjoyes the property:

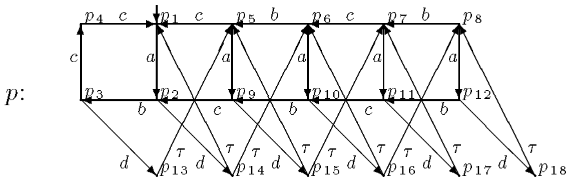


Figure 0.2: Figure 2.2 [Koy 85]

and q the property:

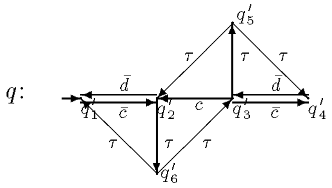


Figure 0.3: Figure 2.3 [Koy 85]

Consider the processes p and q. Together they implement CABP which is verified in [Koy 85] using process algebra. But when p and q are knit together over 80% of p and q cannot be reached i.e. in the context of q certain parts of p cannot be reached and also certain parts of q cannot be reached in the context of p.

It would be nice if we could specify p and q as p' and q':

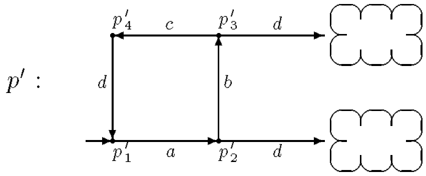


Figure 0.4: Figure 2.4 [Koy85]

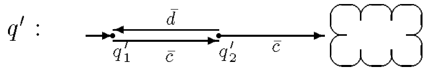


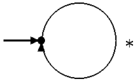
Figure 0.5: Figure 2.5 [koy85]

Here ☁ may be thought of as an area where any behaviour may be encountered. When we knit p' and q' together we are prohibited from entering the ☁ , and the behaviour is like that of p and q knit together.

We want to formalize the idea of using ☁ in what may be called partial specification of processes. In [Koy 85] a method called modularization was developed in the framework of process algebra [Gla 85], but the concept of modules seems ad hoc and seems only to be adequate to treat the example.

Perhaps the idea of using ☁ could be fit into the use of environment information presented in [Lar 86b] Here we are much more informal about what environment the process is to be in as long as it prohibits the process from entering the ☁ . This approach is much more like a creative programming process, we guess where to put in ☁ , then knit the system together and try if any ☁ may be reached. In that case the system is not fully specified and we have to concretize the specification.

To take the place of ☁ we want a process able to do an action which may represent any action. Call the action $*$ and define $\mathcal{U}$ as the process:

$\mathcal{U}$:  i.e. $\mathcal{U}$ can do $*$ and then become it self again.

In a way $*$ may be thought of as being a stronger (more informative) action than any other action. This yields a preorder on actions. The idea of using a preorder on actions seems more general than just to fit the above example. Another example where a preorder may be used is when actions are sets of "actions" with subset inclusion as ordering.

The purpose of this thesis is to formalize the idea of preorders on actions and extend it to processes. We shall develop a framework characterizing the preorder on processes induced by the preorder on actions and we shall show how this may be used in hierarchical development of concurrent systems. One example being the above system called CABP. Also we shall show how to introduce data flow analysis methods into hierarchical development of concurrent systems using the method of abstract interpretation.

Overview

Chapter 1 contains the description of our model of concurrent systems. The formalization of processes simulating and bisimulating each other is presented, and their extension to process systems with ordering upon actions is defined. The most refined preorder being able to take deadlock properties into account.

Chapter 2 contains the syntax and semantics of a language for defining process. The language being an extract of current variants of CCS [Mil 80].

In chapter 3 three sound and complete proof systems for the preorder introduced in chapter 1 are presented. The one system being for finite nondeterministic terms, the second the extension to general finite terms, and the third being for regular expressions.

In chapter 4 we show how the ordering introduced in the introduction may be used to formalize and solve the problem of partial specification and how it may be used to verify the concurrent alternating bit protocol (CABP).

Chapter 5 contains a larger and more theoretical use of the framework. Here the framework of abstract interpretation [Cou 77], known from data flow analysis of sequential and functional programming, is introduced such that approximations of concurrent processes may be used to make statements about processes. An example of the use of this is stable analysis, answering when processes are stable. This may be used to infer, that if p and q are observational equivalent and both stable, then p and q are congruent [Mil 80]. Also the proof system of chapter 3 may be used together with the framework presented in chapter 5.

Chapter 6 is the conclusion. We make a review of goals achieved and future work to be done.

Chapter 1

A model of concurrency and implementation orders

In recent years strong efforts have gone into giving semantic models for concurrent systems. Especially the desire for applying the methods of denotational semantics, known and well established for sequential programs, has been tried but without or with little success. The desire for a denotational model is partly based on its ability of decompositionality, but a major defect with this model seems to be that it only takes the input/output behaviour of programs into account, without taking notice of the intermediate states the program may pass through. Although this problem has been dealt with (see e.g. [Smy 78]), the solution does not seem appropriate.

Instead the method of operational semantics, with a notion of concurrent systems being processes or machines able to perform actions or offering experiments, together with labelled transition systems has shown successful in giving models for concurrent systems.

The work presented in this thesis follows up the line of defining processes, concurrent or nondeterministic, by the set of experiments they offer to an observer. We use the model of labelled transition systems, which is a simple model of nondeterminism based upon the primitive notion of state and transition. The simple notion of labelled transition systems has proven a very good model for operational semantics of programming languages [Plo 81].

Definition 1.1 (Definition 1.4 in [Plo 81])
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.

For $(s, a, t) \in \longrightarrow$ we shall write $s \xrightarrow{a} t$ which may be interpreted as in state s the system may perform an a action and in doing so evolve to a state t. We use the usual abbreviations as e.g. $s \xrightarrow{a}$ for $\exists t \in St.s \xrightarrow{a} t$ and $s \overset{a}{\not\rightarrow}$ for $\neg \exists t \in St.s \xrightarrow{a} t$.

Also we shall identify the state of a process by the process, yielding a transition system $P = (Pr, Act, \longrightarrow)$ modelling the operational semantics of a system of processes. Now if p and q are two processes of P, we say that p is simulated by q or q simulates p if every derivation of p can be simulated by a derivation of q in such a way that the simulation properties are maintained.

Definition 1.2 *A* <u>*simulation*</u> *R is a binary relation on Pr such that whenever pRq and $a \in Act$ then:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.q \xrightarrow{a} q' \;\&\; p'Rq'$

A process q is said to simulate a process p if and only if there is a simulation R such that pRq and then we write $p \leq q$.

Now for $R \subseteq Pr^2$ we can define $\mathcal{S}(R)$ as:

Definition 1.3 $(p,q) \in \mathcal{S}(R)$ *iff:*

1. $\forall a \in Act.p \xrightarrow{a} p' \Rightarrow \exists q'.q \xrightarrow{a} q' \;\&\; p'Rq'$

$\mathcal{S}$ is easy seen to be a monotone endofunction on the complete lattice of binary relations (over Pr) under subset inclusion. Standard fixed point results, due to Tarski [Tar 55], yield that a maximal fixed point for $\mathcal{S}$ exists and is defined as $\bigcup\{R \mid R \subseteq \mathcal{S}(R)\}$. This maximal fixed point equals $\leq$.

Proposition 1.1 $\leq$ *is a preorder on* Pr^2

PROOF: $Id_{Pr} = \{(p,p) \mid p \in Pr\}$ is a simulation, and composition of simulations yields a simulation. □

Composition of relations $R, S \subseteq Pr^2$ is taken to be

$$R \circ S = \{(p_1, p_3) \mid \exists p_2.(p_1,p_2) \in R \;\&\; (p_2,p_3) \in S\}.$$

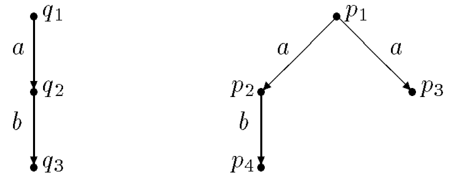
Note how this is opposite of function composition. The above yields the elegant proof technique named Park's Induction Principle by L. Cardelli [Car 81]:

Definition 1.4 $p \leq q$ *iff* $\exists R \subseteq Pr^2.(p,q) \in R \;\&\; R \subseteq \mathcal{S}(R)$

So when we have to prove $p \leq q$ it is sufficient and necessary to find a simulation containing (p,q). Throughout this thesis we shall write: "To see that $R \subseteq function(R)\ldots$", where $function$ is e.g. $\mathcal{S}$. By this we actually mean: "To see that $(p,q) \in function(R)$ whenever $(p,q) \in R \ldots$".

An equivalence relation on Pr may be obtained by $p \simeq q$ iff $p \leq q$ and $q \leq p$. However, this equivalence does not preserve deadlock properties as may be seen from the following example:

Example 1.1 (Example 2.1-14 [Lar 86a])



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. But p can perform an a-action and reach a state where no b-action is possible whereas q cannot. To obtain an equivalence that preserves deadlock properties the notion of bisimulation is introduced. Bisimulation is originally due to D. Park [Par 81] and was investigated by R. Milner [Mil 83].

Definition 1.5 *A binary relation R on Pr is a <u>bisimulation</u> iff whenever pRq and $a \in Act$ then:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.q \xrightarrow{a} q' \;\&\; p'Rq'$
2. $q \xrightarrow{a} q' \Rightarrow \exists p'.p \xrightarrow{a} p' \;\&\; q'Rp'$

Two processes p and q are said to be simulating each other if there exists a bisimulation containing (p, q), and then we write $p \sim q$.

Now for $R \subseteq Pr^2$ define $\mathcal{B}(R)$ as:

Definition 1.6 $(p, q) \in \mathcal{B}(R)$ *iff:*

1. $\forall a \in A.p \xrightarrow{a} p' \Rightarrow \exists q'.q \xrightarrow{a} q' \;\&\; p'Rq'$
2. $\forall a \in A.q \xrightarrow{a} q' \Rightarrow \exists p'.p \xrightarrow{a} p' \;\&\; q'Rp'$

Also $\mathcal{B}$ is easy seen to be a monotone endofunction on the complete lattice of binary relations (over Pr) under subset inclusion. Thus there exists a maximal fixed point. This fixed point equals $\sim$. We refer to [Lar 86a] for a proof of when $\sim$ coincides with strong equivalence defined in [Mil 80]. But the following proposition is easy established:

Proposition 1.2 $\sim$ *is an equivalence on Pr*

PROOF: Id_{Pr} is a bisimulation, and composition of bisimulations are easy seen to be bisimulations, and if R is a bisimulation then

$$R^T = \{(p, q) \mid (q, p) \in R\}$$

is a bisimulation. □

If the set of actions *Act* is equiped with a preorder $\sqsubseteq_A$, this preorder may have the intuition that $a \sqsubseteq_A b$ holds whenever what a can do for an observer b can do at least as well.

One may think of this as for example the light switches in a car. If a is the parking light and b is the normal light then switching on the normal light also turns on the parking light, so the normal light can do as well as the parking light. Of course the normal light is more informative than the parking light.

The preorder on actions may be extended to processes by an extension of the notion of simulation and bisimulation discussed above. Remember that a process q simulated a process p if all the derivations of p could be simulated by derivations of q. The extended simulation has the intuition that all the derivations of p can be simulated by derivations of q which can do at least as well for an observer e.g.

Definition 1.7 $p \leq_{\mathcal{E}} q$ *iff* $p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$

Definition 1.8 *An* <u>*extended simulation*</u> *R is a binary relation on Pr such that whenever pRq and $a \in Act$ then:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$

A process q is said to simulate, in the extended sense, a process p if and only if there exists a simulation R with pRq and in this case we write $p \leq_{\mathcal{E}} q$.

Now for $R \subseteq Pr^2$ we can define $\mathcal{ES}(R)$ as:

Definition 1.9 $(p,q) \in \mathcal{ES}(R)$ *iff:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$

Proposition 1.3 *R is an extended simulation iff $R \subseteq \mathcal{ES}(R)$*

As for simulation and bisimulation the fact that $\mathcal{ES}$ is monotone upon the complete lattice of binary relations (over Pr) under inclusion is easy verified. Thus there exists a maximal fixed point given by $\cup\{R \mid R \subseteq \mathcal{ES}(R)\}$ using standard fixed point results, originally due to Tarski [Tar 55]. This maximal fixed point equals $\leq_{\mathcal{E}}$. As with the notion of simulation q may extended simulate p without having the same deadlock properties. Also extended bisimulation enjoys the property of being a preorder:

Proposition 1.4 *$\leq_{\mathcal{E}}$ is a preorder on Pr^2*

Proof: As for simulation. □

To obtain a preorder which preserves deadlock properties we may, as with simulation, extend the notion of extended simulation to extended bisimulation. The intuition in the notion of extended bisimulation is that a process p is extended bisimulated by a process q if all the first actions of p can be matched by actions of q which for an observer can do at least as well. Also whenever q can do an action a, p has to match a in an approximative way by an action b such that $b \sqsubseteq_A a$. Also the processes have to have the same potentiality after performing the actions. This may be stated as:

Definition 1.10 *A binary relation R on Pr is an extended bisimulation whenever pRq and $a \in Act$ then:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$
2. $q \xrightarrow{a} q' \Rightarrow \exists p'.\exists b'.p \xrightarrow{b} p' \;\&\; b \sqsubseteq_A a \;\&\; p'Rq'$

If there exists a relation R such that R is an extended bisimulation and pRq holds we write $p \sqsubseteq q$. Now for $R \subseteq Pr^2$ we can define $\mathcal{EB}(R)$ as:

Definition 1.11 $(p,q) \in \mathcal{EB}(R)$ *iff:*

1. $\forall a \in Act.p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$
2. $\forall a \in Act.q \xrightarrow{a} q' \Rightarrow \exists p'.\exists b'.p \xrightarrow{b} p' \;\&\; b \sqsubseteq_A a \;\&\; p'Rq'$

Proposition 1.5 *R is an extended bisimulation iff $R \in \mathcal{EB}(R)$*

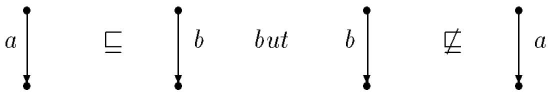
$\mathcal{EB}$ may be shown to be a monotone endofunction upon the complete lattice of binary relations over Pr. And by classic lattice theory $\mathcal{EB}$ has a maximal fixed point which equals $\sqsubseteq$. Also $\sqsubseteq$ is a preorder:

Proposition 1.6 *$\sqsubseteq$ is a preorder on Pr^2*

PROOF: As for simulation. □

One may wonder why extended bisimulation only yields a preorder when bisimulation yields an equivalence. This is due to the use of $\sqsubseteq_A$ in the predicate, a simple example shows why we cannot hope for an equivalence:

Example 1.2 *if $a \sqsubseteq_A b$ then*



This is analogous to the development in [Hen 84] where divergence is taken into account. A process is said to diverge if it can perform an endless sequence of internal/unobservable actions. In [Hen 84] bisimulation is extended such that $p \sqsubseteq q$ iff whatever action p can do q can match and if p and q do not diverge, then if q performs an action p has to match it.

The idea of $\sqsubseteq$ may be further refined. We think of $a \sqsubseteq_A b$ as a relation specifying whether b for an observer can do as well as a. In a way this states that b contains more information than a. In $p \sqsubseteq q$ we insisted on p matching every move of q perhaps by a less informative action, but this constraint can be too strong. If b is an action that in a way is too informative, like $\top$ in denotational semantics [Sto 77], we may loosen the second condition on $\sqsubseteq$ by cutting out such actions. This may be done by excluding b from the set of actions p has to match by giving only the set M of actions which has to be matched. And we arrive at the definition of M-bisimulation.

Definition 1.12 *A set M is downwardsclosed iff $\forall a \in M.\ b \sqsubseteq_A a \Rightarrow b \in M$.*

Definition 1.13 *Let M be a downwardsclosed set. An <u>M-bisimulation</u> R is a binary relation on Pr such that whenever pRq:*

1. $p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$
2. $\forall a \in M.q \xrightarrow{a} q' \Rightarrow \exists p'.\exists b'.p \xrightarrow{b} p' \;\&\; b \sqsubseteq_A a \;\&\; p'Rq'$

Two processes p and q are said to be in M-bisimulation if there exists an M-bisimulation containing (p, q) and we write $p \sqsubseteq_M q$. The need for M being downwardsclosed has proven essential for the theoretical development to work. Note that we may always downwardsclose a set B by $CL(B) = \{a \mid \exists b \in B.a \sqsubseteq_A b\}$. M may be seen as an environment where we can cut out uninteresting actions, for example when they are useless because they contain too much (possible inconsistent) information. Now for $R \subseteq Pr^2$ we can define $M\mathcal{B}(R)$ as:

Definition 1.14 *$(p, q) \in M\mathcal{B}(R)$ iff:*

1. $\forall a \in Act.p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b'.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; p'Rq'$
2. $\forall a \in M.q \xrightarrow{a} q' \Rightarrow \exists p'.\exists b'.p \xrightarrow{b} p' \;\&\; b \sqsubseteq_A a \;\&\; p'Rq'$

Proposition 1.7 *R is an M-bisimulation iff $R \subseteq M\mathcal{B}(R)$*

The fact that $M\mathcal{B}(R)$ is a monotone endofunction on the complete lattice of binary relations on Pr under inclusion is easy established. Thus $M\mathcal{B}$ has a maximal fixed point given by: $\bigcup\{R \mid R \subseteq M\mathcal{B}(R)\}$. Also this fixed point coincides with $\sqsubseteq_M$.

$\sqsubseteq_M$ actually extends $\sqsubseteq_A$ to processes:

Proposition 1.8 $\sqsubseteq_M$ *is a preorder on* Pr

PROOF: $\sqsubseteq_M$ is reflexive since $Id = \{(p,p) \mid p \in Pr\}$ is easy seen to be an M-bisimulation. $\sqsubseteq_M$ is transitive since if $p \sqsubseteq_M q$ and $q \sqsubseteq_M r$ then there exist M-bisimulations R and Q such that $(p,q) \in R$ and $R \subseteq M\mathcal{B}(R)$ and $(q,r) \in Q$ and $Q \subseteq M\mathcal{B}(Q)$. If M is downwardsclosed we may infer that the composition $R \circ Q$ is an M-bisimulation containing (p,r). To see that $R \circ Q \subseteq M\mathcal{B}(R \circ Q)$ observe that if $(p,r) \in R \circ Q$ then if $p \xrightarrow{a} p'$ then there exist q, q' and b such that $q \xrightarrow{b} q'$ with $a \sqsubseteq_A b$ and $(p',q') \in R$. Also if $q \xrightarrow{b} q'$ there exist r' and c such that $r \xrightarrow{c} r'$ with $b \sqsubseteq_A c$ and $(q',r') \in Q$ i.e. $(p',r') \in R \circ Q$. Also if $r \xrightarrow{a} r'$, $a \in M$, then there exist q, q' and b such that $b \sqsubseteq_A$ a and $(r',q') \in Q$, since M is downwardsclosed $b \in M$ and there exist p' and c such that $c \sqsubseteq_A b$ and $p \xrightarrow{c} p'$ with $(p',q') \in R$ i.e. $(p',r') \in R \circ Q$. □

A very convenient concept in proofs of M-bisimulation is the notion of a relation being an M-bisimulation upto '$\sqsubseteq_M$'. A similar definition of bisimulation upto '$\sim$' is given in [Mil 83].

Definition 1.15 *R is an M-bisimulation upto '*$\sqsubseteq_M$*' iff* $\sqsubseteq_M \circ R \circ \sqsubseteq_M$ *is an M-bisimulation.*

Proposition 1.9 *R is an M-bisimulation upto '*$\sqsubseteq_M$*' iff*

$$R \subseteq M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M)$$

PROOF: if R is an M-bisimulation upto '$\sqsubseteq_M$' then

$$\sqsubseteq_M \circ R \circ \sqsubseteq_M \subseteq M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M)$$

Clearly $R \subseteq \sqsubseteq_M \circ R \circ \sqsubseteq_M$, since $Id \subseteq \sqsubseteq_M$ and $R = Id \circ R \circ Id$.
Also if

$$R \subseteq M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M)$$

then

$$\sqsubseteq_M \circ R \circ \sqsubseteq_M \subseteq \sqsubseteq_M \circ M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M) \circ \sqsubseteq_M$$

but

$$\sqsubseteq_M \subseteq M\mathcal{B}(\sqsubseteq_M)$$

so

$$\sqsubseteq_M \circ R \circ \sqsubseteq_M \subseteq M\mathcal{B}(\sqsubseteq_M) \circ M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M) \circ M\mathcal{B}(\sqsubseteq_M)$$

but $M\mathcal{B}$ is monotone so

$$\sqsubseteq_M \circ R \circ \sqsubseteq_M \subseteq M\mathcal{B}(\sqsubseteq_M \circ \sqsubseteq_M \circ R \circ \sqsubseteq_M \circ \sqsubseteq_M) = M\mathcal{B}(\sqsubseteq_M \circ R \circ \sqsubseteq_M)$$

since $\sqsubseteq_M$ is transitive. □

The conveniency of this concept may be seen in proofs where we can construct a relation R, which almost is an M-bisimulation except for certain closure properties. The concept of M-bisimulation upto '$\sqsubseteq_M$' provides these closure properties and ensures the existence of an M-bisimulation. We may now see the notion of simulation, bisimulation, extended simulation, extended bisimulation and M-bisimulation as an evolution:

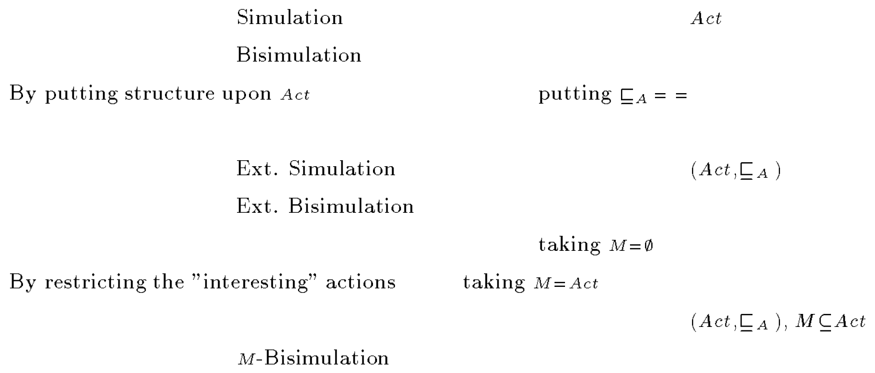

Simulation
Act
Bisimulation
By putting structure upon Act
putting $\sqsubseteq_A = =$
Ext. Simulation
$(Act, \sqsubseteq_A)$
Ext. Bisimulation
taking $M=\emptyset$
By restricting the "interesting" actions
taking $M=Act$
$(Act, \sqsubseteq_A)$, $M \subseteq Act$
M-Bisimulation

Figure 1.1:

Note how M-bisimulation does not need a definition of M-simulation since we just take $M = \emptyset$. This shows M-bisimulation as the most abstract of the notions considered and we therefore investigate its properties by giving sound and complete proof systems for it over a language for defining processes or machines. The language for defining machines is simply the word algebra T_Σ over the signature Σ of operators for machines. This yields a way of defining machines by their constituents.

A very interesting connection between two preorders upon the same universe is expressed by:

Proposition 1.10 *Let $\sqsubseteq_A$ and $\leq_A$ be preorders over the same set of actions A. Let $M, N \subseteq A$, then:*

$$N \subseteq M \; \& \; \sqsubseteq_A \subseteq \leq_A \; \& \, p \sqsubseteq_M q \;\Rightarrow\; p \leq_N q$$

Proof: The relation $R = \{(p,q) \mid p \sqsubseteq_M q\}$ is an N-bisimulation with respect to $\leq_A$. To see that $R \subseteq N\mathcal{B}(R)$ observe that if $p \xrightarrow{a} p'$ then there exist q' and b such that $q \xrightarrow{b} q'$ and $a \sqsubseteq_A b$ and $p' \sqsubseteq_M q$ but since $\sqsubseteq_A \subseteq \leq_A$ then

$(p', q') \in R$. Also if $q \xrightarrow{a} q'$, $a \in N$, then there exist p' and b such that $p \xrightarrow{b} p'$ and $b \sqsubseteq_A a$, since $a \in M$, and $p' \sqsubseteq_M q'$ but since $\sqsubseteq_A \subseteq \leq_A$ then $(p', q') \in R$. □

By this proposition it is possible to relate two different orderings on actions to orderings on processes.

Chapter 2

The language for defining processes

The language for defining processes is taken to be the free Σ-algebra T_Σ over a signature Σ, including the set of actions Act, a set of variables
$X = \{x, x_i, .., y, z, ..\}$ and operators for nondeterminism, communication and recursion.

The model is extensional in the sense that concurrency is unobservational and therefore, as will be seen later, communication may be exchanged by action-prefixing and nondeterminism.

Let $\Sigma = Act \cup \{nil, +, (\ldots)[f], |_g, \mu x.\} \cup X$. Thus Σ is a set of operators. nil and every $x \in X$ being constants, every $a \in Act$ a unary operator for action prefixing, $\mu x.$ a unary operator for recursive binding of x, $+$ a binary operator for nondeterminism, $|_g$ a binary operator for communication with the possibility of interleaving, one for every binary monotone partial function $g : Act^2 \hookrightarrow Act$ and $(\ldots)[f]$, an n-ary operator for every n-ary monotone partial function
$f : Act^n \hookrightarrow Act$.

By a monotone partial function we mean a function such that

$$a \sqsubseteq_A \ b \ \& \ f(a) \ defined \ \Rightarrow f(b) \ defined \ \& \ f(a) \sqsubseteq_A \ f(b).$$

Terms p in T_Σ are generated by the following abstract grammar:

$$p \ ::= \ nil \Bigm| a.p_1 \Bigm| p_1 + p_2 \Bigm| (p_1 \ldots p_n)[f] \Bigm| p_1 \,|_g\, p_2 \Bigm| x \Bigm| \mu x.p$$

As it may be seen from the above definition, Σ resembles an extract or extension of current variants over CCS [Mil 80].

In the notation $(\ldots)[f]$ we follow the idea of P. Aczel [Acz 84].

As an example f may be instantiated to an identity function on a subset B of Act, obtaining the restriction operator $\lceil B$, known from SCCS [Mil 83]. Also

f may be instantiated to a monotone endofunction $\Phi : Act \to Act$ yielding a renaming function.

Definition 2.1 *The operational semantics of the language is taken to be the smallest family of relations* $\{\xrightarrow{a} \subseteq T_\Sigma^2 \mid a \in Act\}$ *satisfying:*

$$\begin{array}{lrl}
ACT\!\rightarrow\!: & & a.p \xrightarrow{a} p \\[1ex]
SUM\!\rightarrow\!: & \underline{\textit{if}} & p_1 \xrightarrow{a} p' \\
 & \underline{\textit{or}} & p_2 \xrightarrow{a} p' \\
 & \underline{\textit{then}} & p_1 + p_2 \xrightarrow{a} p' \\[1ex]
FUN\!\rightarrow\!: & \underline{\textit{if}} & \forall i.p_i \xrightarrow{a_i} p_i' \;\&\; c \simeq f(a_1 \ldots a_n) \\
 & \underline{\textit{then}} & (p_1 \ldots p_n)[f] \xrightarrow{c} (p_1' \ldots p_n')[f] \\[1ex]
COM\!\rightarrow\!: & \underline{\textit{if}} & (p_1 \xrightarrow{c} p_1' \;\&\; p =' p_1' \mid_g p_2) \\
 & \underline{\textit{or}} & (p_2 \xrightarrow{c} p_2' \;\&\; p' = p_1 \mid_g p_2') \\
 & \underline{\textit{or}} & (p_1 \xrightarrow{a} p_1' \;\&\; p_2 \xrightarrow{b} p_2' \;\&\; p' = p_1' \mid_g p_2' \;\&\; c \simeq g(a,b)) \\
 & \underline{\textit{then}} & p_1 \mid_g p_2 \xrightarrow{c} p' \\[1ex]
REC\!\rightarrow\!: & \underline{\textit{if}} & p[\mu x.p/x] \xrightarrow{a} p' \\
 & \underline{\textit{then}} & \mu x.p \xrightarrow{a} p'
\end{array}$$

Here $c \simeq f(a_1 \ldots a_n)$ *and* $c \simeq g(a,b)$ *means true if* $f(a_1 \ldots a_n)(g(a,b))$ *is defined and* c *is instantiated to the value of* $f(a_1 \ldots a_n)(g(a,b))$*, false otherwise.*

In the above definition $p[q/x]$ means simultaneous substitution of q on every occurrence of x in p taking account of change of bound variables. A formal definition of substitution will be given in chapter 3 (definition 3.3).Also $p[\mu x.p/x]$ in $REC\!\rightarrow$ may be refered to as unfolding the recursive process $\mu x.p$ once.

In the notion $|_g$ we generalize the communication operator $|$ form CCS. By instantiating g to a function such that $g(a, \overline{a}) = \tau$ and $g(a,b) = undefined$ if $a \neq \overline{b}$ we obtain $|$.

The inclusion of $|_g$ with the operational semantics $COM\!\rightarrow$ shows that we have chosen to model concurrency asynchronously. Processes may communicate if they perform actions a and b and $g(a,b)$ is defined, otherwise they only can proceed interleaved.

If Act has an ablean group structure with composition function $f : Act^2 \hookrightarrow Act$, we could model the synchronous communication operator $\times$ known from SCCS [Mil 83],with the notion $(\ldots)[f]$.

As was shown in [Mil 83] it is possible to derive $|$ from $\times$ if we add an unobservable action "1" to the set of actions and extend the notion of M-bisimulation to take this into account, obtaining a notion of "weak"-M-bisimulation. It is a field for further study if and how such a theory may be developed but we do not investigate it further in this thesis.

If we only want to model synchronism we may exclude $|_g$ from the language and $COM \rightarrow$ from the semantics.

By the introduction of g in $|_g$ we have the ability of obtaining much more interesting communication operators by taking Act to a set with much more structure. An example would be $Act \simeq BAct + Act \times Act$ where $BAct$ is a set of basic actions as e.g. $\{a, b, c, \ldots, \overline{a}, \overline{b}, \overline{c}, \ldots\}$. We then instantiate $|_g$ with $g = \Theta \circ In_2 \circ (,)$ where $(,)$ is tupling and Θ is the isomorphism folding into a recursive defined domain. This communication operator gives a kind of composite action, composed of basic actions involved in communication during evaluation of the program. Of cause we may use the above g in $(\ldots)[f]$ in the synchronous case.

2.1 A general scheme for transformations

Often we want to define a transformation $\mathcal{F}$ from the language for defining processes to some domain D of interest, for example defining the free variables in processes by a transformation $\mathcal{F} : T_\Sigma \rightarrow \mathcal{P}(X)$.

We want the function to be defined structurally i.e. the result of taking $\mathcal{F}$ upon a compound construct depends on it constituents.

This is done by giving functions for every operator of Σ. So we have to define functions and constants:

Definition 2.2

$$\begin{array}{rcl} g_+ & \in & D \times D \rightarrow D \\ g_{nil} & \in & D \\ g_{\mu x.} & \in & D \rightarrow D \\ g_{|_\mathrm{g}} & \in & D^2 \rightarrow D \\ g_{(\ldots)[\mathrm{f}]} & \in & D^n \rightarrow D \\ g_x & \in & D \text{ one for each } x \in X \\ g_a & \in & D \text{ one for each } a \in Act \end{array}$$

Then we can define $\mathcal{F} : T_\Sigma \rightarrow \mathcal{P}(X)$ structurally by:

$$\begin{array}{rcl} \mathcal{F}(nil) & = & g_{nil} \\ \mathcal{F}(p_1 + p_2) & = & g_+(\mathcal{F}(p_1), \mathcal{F}(p_2)) \\ \mathcal{F}(\mu x.p) & = & g_{\mu x.}(\mathcal{F}(p)) \\ \mathcal{F}(p_1 \mid_g p_2) & = & g \mid_\mathrm{g}(\mathcal{F}(p_1), \mathcal{F}(p_2)) \end{array}$$

$$\begin{aligned}
\mathcal{F}((p_1 \ldots p_n)[f]) &= g_{(\ldots)[\mathrm{f}]}(\mathcal{F}(p_1) \ldots \mathcal{F}(p_n)) \\
\mathcal{F}(x) &= g_x \\
\mathcal{F}(a.p) &= g_a(\mathcal{F}(p))
\end{aligned}$$

Note how this scheme resembles denotational definitions.

Chapter 3

Complete proof systems

In this chapter we present sound and complete proof systems for three sublanguages Σ^{nf}, Σ^{f} and Σ^{r} of Σ.

We cannot hope for sound and complete proof systems for the whole Σ since $p_1 \mid_g p_2$ and $(p_1 \ldots p_n)[f]$ allow communication to be introduced into recursive defined terms and by this a coding of a Turing machine may be possible (see e.g. [Mil 83]). Therefore we restrict our attention to three proof systems of interest.

The first is for closed finite terms with only action-prefixing and nondeterminism, the second for closed finite terms without recursion and the third for regular expressions. The second is only a conservative (to be explained later) extension of the first. In general these are of interest since the second and third system may be used to express a lot of interesting concurrent systems within, systems either involving only closed finite expressions or systems involving only regular expressions or systems with recursion without using communication within recursion. Examples are protocols in data transmission.

3.1 Finite terms

In this part we present sound and complete proof systems for finite terms. We do this in three steps, first we present a proof system for nondeterminism, secondly we extend this by the operator $\mid_g$ for communication and thirdly we introduce the function scheme $(\ldots)[f]$.

Prefix and nondeterminism

First let us define the transition system of nondeterministic finite terms $NFT = (P_{nf}, Act, \longrightarrow)$ where P_{nf} is given by the following abstract syntax:

$$p \ ::= \ nil \ \Big| \ a.p_1 \ \Big| \ p_1 + p_2$$

Thus $\Sigma^{nf} = \Sigma \setminus (\{(\ldots)[f], |_g, \mu x.\} \cup X)$.
The operational semantics is the smallest family of relations satisfying $ACT\!\rightarrow$ and $SUM\!\rightarrow$ of definition 2.1

Proposition 3.1 *$\sqsubseteq_M$ is a precongruence with respect to the operators of Σ^{nf}*

PROOF: By proposition 1.8 $\sqsubseteq_M$ is a preorder.
To show that $\sqsubseteq_M$ is preserved by prefixing and nondeterminism we establish two M-bisimulations:

$$R_1 = \{(a.p_1, b.p_2) \mid a \sqsubseteq_A b \;\&\; p_1 \sqsubseteq_M p_2\} \cup \sqsubseteq_M$$

$$R_2 = \{(p_1 + p_2, q_1 + q_2) \mid p_1 \sqsubseteq_M q_1 \;\&\; p_2 \sqsubseteq_M q_2\} \cup \sqsubseteq_M$$

R_1 is more general than needed for showing preservation of $\sqsubseteq_M$ by prefixing, but we obtain this by choosing $b = a$.
The full proof is presented in appendix A. □

Proposition 3.2 *The following holds for all $p_1, p_2, p_3 \in T_{\Sigma^{nf}}$:*

$$\begin{array}{rcl} p_1 + (p_2 + p_3) & \simeq_M & (p_1 + p_2) + p_3 \\ p_1 + p_2 & \simeq_M & p_2 + p_1 \\ p_1 + p_1 & \simeq_M & p_1 \\ p_1 + nil & \simeq_M & p_1 \end{array}$$

Here $\simeq_M$ means that both $\sqsubseteq_M$ and $\sqsupseteq_M$ hold.

PROOF: The proof is similar to previous axiomations of bisimulation see e.g. [Hen 85].
The proof consists of establishing 8 M-bisimulations:
The relations

$$R_1 = \{(p_1 + (p_2 + p_3), (p_1 + p_2) + p_3) \mid p_1, p_2, p_3 \in Pr\} \cup Id$$

$$R_2 = \{(p_1 + p_2, p_1 + p_2) \mid p_1, p_2 \in Pr\} \cup Id$$

$$R_3 = \{(p_1 + p_1, p_1) \mid p_1 \in Pr\} \cup Id$$

$$R_4 = \{(p_1 + nil, p_1) \mid p_1 \in Pr\} \cup Id$$

are M-bisimulations.
Also all R_i^{-1}, $i \in \{1 \ldots 4\}$, are M-bisimulations.

The full proof is presented in appendix A. □

We now present the proof system S_{nf} of the precongruence $\sqsubseteq_M$ over P_{nf}:

We write $\vdash p_1 \sqsubseteq_M p_2$ if $p_1 \sqsubseteq_M p_2$ is provable by the rules of table 3.1.

SUM	S1	$p_1 + (p_2 + p_3) =_M (p_1 + p_2) + p_3$
	S2	$p_1 + p_2 =_M p_2 + p_1$
	S3	$p_1 + p_1 =_M p_1$
	S4	$p_1 + nil =_M p_1$
PREORD	P1	$p \sqsubseteq_M p$
	P2	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_3}{p_1 \sqsubseteq_M p_3}$
	P3	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}{p_1 =_M p_2}$
	P4	$\dfrac{p_1 =_M p_2}{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}$
PRECONG	C1	$\dfrac{p_1 \sqsubseteq_M p_2}{a.p_1 \sqsubseteq_M b.p_2} \quad , a \sqsubseteq_A b$
	C2	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_3 \sqsubseteq_M p_4}{p_1 + p_3 \sqsubseteq_M p_2 + p_4}$
ANNIHIL		$\dfrac{b \notin M}{nil \sqsubseteq_M b.p}$
CONS		$\dfrac{p_1 \sqsubseteq_M p_2}{p_1 \sqsubseteq_N p_2} \quad , N \subseteq M$

Table 3.1: Proof system for nondeterminism.

Proposition 3.3 *$\sqsubseteq_M$ satisfies the axiom ANNIHIL of table 3.1*

Proof: The relation $R = \{(nil, b.p) \mid b \notin M\}$ is an M-bisimulation. To see that $R \subseteq M\mathcal{B}(R)$ observe that $nil \overset{a}{\not\rightarrow}$ for any a. Also $b.p \overset{a}{\not\rightarrow}$ for any $a \in M$. i.e. both conditions of $\sqsubseteq_M$ hold trivially. □

Theorem 3.1 *The proof system S_{nf} of table 3.1 is sound*

Proof: By proposition 3.2 $\sqsubseteq_M$ satisfies S1-S4. By proposition 1.8 $\sqsubseteq_M$ is a preorder. By proposition 3.1 $\sqsubseteq_M$ is a precongruence and by proposition 3.3 ANNIHIL is satisfied. Also proposition 1.10 with $\leq_A = \sqsubseteq_A$ ensures that $\sqsubseteq_M$ satisfies CONS. □

We now turn to prove the completeness of the proof system S_{nf}. In this proof we introduce a normalform similar to that of [Hen 85]. A term is said to be in normalform if it can be written as:

$$p = \sum_{i=1}^{n} a_i.p_i \text{ where } p_i \text{ is in normalform}$$

We use $\sum_{i=1}^{n} a_i.p_i$ to describe the sum $a_1.p_1 + \ldots + a_n.p_n$, $n > 0$ and *nil* if $n = 0$, knowing the notion is unambiguous because of S1-S3 of table 3.1.

Proposition 3.4 *Every term p in Σ^{nf} has a normalform $nf(p)$ such that $\vdash p =_M nf(p)$*

PROOF: By structural induction (see appendix A). □

Proposition 3.5 *If p and q are in normal form then:*

$$p \sqsubseteq_M q \Rightarrow \vdash p \sqsubseteq_M q$$

PROOF: By induction on the size of p and q using that $p \xrightarrow{a} p'$ iff $a.p'$ is a subterm of p. Also in the case $a \notin M$ we use ANNIHIL.
For the full proof see appendix A. □

Theorem 3.2 *If $p \sqsubseteq_M q$ then $\vdash p \sqsubseteq_M q$*

PROOF: Assume $p \sqsubseteq_M q$. By proposition 3.4 there exist $nf(p)$ and $nf(q)$ such that:

$$\vdash p =_M nf(p) \text{ and } \vdash q =_M nf(q) \tag{3.1}$$

By theorem 3.1 (soundness) we have

$$p \sqsubseteq_M nf(p) \text{ and } nf(p) \sqsubseteq_M p \text{ and } q \sqsubseteq_M nf(q) \text{ and } nf(q) \sqsubseteq_M q \tag{3.2}$$

Since $\sqsubseteq_M$ is a preorder, $p \sqsubseteq_M q$ together with 3.2 yields:

$$nf(p) \sqsubseteq_M nf(q)$$

By proposition 3.5 we get

$$\vdash nf(p) \sqsubseteq_M nf(q)$$

By (3.1) and P2 of table 3.1 we get

$$\vdash p \sqsubseteq_M nf(p) \text{ and } \vdash nf(q) \sqsubseteq_M q$$

giving

$$\vdash p \sqsubseteq_M q$$

□

This shows that the proof system S_{nf} is sound and complete. In fact the proof system of table 3.1 induces the least Σ^{nf}-precongruence satisfying the axioms S1-S4.

Communication

We now extend the language Σ^{nf} with the operator $|_g$ for communication.

Let $CFT = (P_{cf}, Act, \longrightarrow)$ be the transition system of finite terms with communication, where P_{cf} is given by the following abstract syntax:

$$p \ ::= \ nil \ \Big| \ a.p_1 \ \Big| \ p_1 + p_2 \ \Big| \ p_1 \,|_g\, p_2$$

Thus $\Sigma^{cf} = \Sigma \setminus (\{(\ldots)[f], \mu x.\} \cup X) = \Sigma^{nf} \cup \{|_g\}$.

The operational semantics is the smallest family of relations satisfying $ACT\!\rightarrow$, $SUM\!\rightarrow$ and $COM\!\rightarrow$ of definition 2.1.

Proposition 3.6 $\sqsubseteq_M$ *is a precongruence with respect to the operators of* Σ^{cf}*, provided* g *in* $|_g$ *is monotone.*

PROOF: As in proposition 3.1 with the extension that the relation

$$R = \{(p_1 \,|_g\, p_2, q_1 \,|_g\, q_2) \mid p_i \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_i, i \in \{1,2\}\}$$

is an M-bisimulation.
(The proof of this is presented in appendix A.) □

Remember that $CL(B) = \{a \mid \exists b \in B. a \sqsubseteq_A \ b\}$.

Proposition 3.7 *if* $p = \sum_{i=1}^{n} a_i.p_i$ *and* $q = \sum_{j=1}^{m} b_j.q_j$ *then*

$$p \,|_g\, q \simeq_M \sum_{(i,j) \in \{(i,j) | g(a_i,b_j) \ defined\}} g(a_i,b_j).(p_i \,|_g\, q_j) + \sum_{i=1}^{n} a_i.(p_i \,|_g\, q) + \sum_{j=1}^{m} b_j.(p \,|_g\, q_j)$$

PROOF: The relation

$$R = \{(p \,|_g\, q, \sum_{(i,j) \in \{(i,j) | g(a_i,b_j) \ defined\}} g(a_i,b_j).(p_i \,|_g\, q_j) + \sum_{i=1}^{n} a_i.(p_i \,|_g\, q) + \sum_{j=1}^{m} b_j.(p \,|_g\, q_j)) \mid p = \sum_{i=1}^{n} a_i.p_i \ \& \ q = \sum_{j=1}^{m} b_j.q_j\}$$

is an M-bisimulation.
Also R^{-1} is an M-bisimulation.
(See appendix A.) □

This proposition shows that communication may be exchanged by action-prefixing and nondeterminism.

Functions

We now extend the language Σ^{cf} with the operator $(\ldots)[f]$ for the general function scheme.

Let $FT = (P_f, Act, \longrightarrow)$ be the transition system of finite terms, where P_f is given by the following abstract syntax:

$$p \ ::= \ nil \ \Big| \ a.p_1 \ \Big| \ p_1 + p_2 \ \Big| \ (p_1 \ldots p_n)[f] \ \Big| \ p_1 \,|_g\, p_2$$

Thus $\Sigma^f = \Sigma^{ct} \cup \{(\ldots)[f]\}$.

The operational semantics is the smallest family of relations satisfying $ACT\!\rightarrow$, $SUM\!\rightarrow$, $COM\!\rightarrow$ and $FUN\!\rightarrow$ of definition 2.1

Proposition 3.8 $\sqsubseteq_M$ *is a precongruence with respect to the operators of* Σ^f *provided* g *in* $|_g$ *and* f *in* $(\ldots)[f]$ *are monotone.*

Proof: As in proposition 3.6 extended with the proof that the relation

$$R = \{((p_1 \ldots p_n)[f], (q_1 \ldots q_n)[f]) \mid p_i \sqsubseteq_{CL(f^{-1}(M)\downarrow i)} q_i\}$$

is an M-bisimulation.
(The proof of this is presented in appendix A.) □

Proposition 3.9 *The following holds for* $p_1 \ldots p_n \in T_{\Sigma^f}$*:*

$$\begin{array}{rcl}
(p_1 \ldots p_n)[f] & \simeq_M & nil \text{ if } p_i \equiv nil \text{ for some } i \leq n \\
(a_1.p_1 \ldots a_n.p_n)[f] & \simeq_M & \begin{cases} f(a_1 \ldots a_n).(p_1 \ldots p_n)[f] \\ \text{if } f(a_1 \ldots a_n) \text{ is defined} \\ \\ nil \quad \text{otherwise} \end{cases} \\
(p_1 \ldots p_i + q_i \ldots p_n)[f] & \simeq_M & (p_1 \ldots p_i \ldots p_n)[f] + \\
 & & (p_1 \ldots q_i \ldots p_n)[f]
\end{array}$$

Proof: By establishing 8 M-bisimulations. The relations

$$R_1 = \{((p_1 \ldots p_n)[f], nil) \mid p_i \equiv nil, \ i \leq n\}$$

$$\begin{array}{c} R_2 = \{((a_1.p_1 \ldots a_n.p_n)[f], f(a_1 \ldots a_n).(p_1 \ldots p_n)[f]) \\ \mid f(a_1 \ldots a_n) \ \text{ is defined}\} \cup Id \end{array}$$

$$R_3 = \{((a_1.p_1 \ldots a_n.p_n)[f], nil) \mid f(a_1 \ldots a_n) \text{ is undefined}\}$$

$$R_4 = \{((p_1 \ldots p_i + q_i \ldots p_n)[f], (p_1 \ldots p_i \ldots p_n)[f] + (p_1 \ldots q_i \ldots p_n)[f])\} \cup Id$$

SUM	S1	$p_1 + (p_2 + p_3) =_M (p_1 + p_2) + p_3$	
	S2	$p_1 + p_2 =_M p_2 + p_1$	
	S3	$p_1 + p_1 =_M p_1$	
	S4	$p_1 + nil =_M p_1$	
PREORD	P1	$p \sqsubseteq_M p$	
	P2	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_3}{p_1 \sqsubseteq_M p_3}$	
	P3	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}{p_1 =_M p_2}$	
	P4	$\dfrac{p_1 =_M p_2}{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}$	
PRECONG	C1	$\dfrac{p_1 \sqsubseteq_M p_2}{a.p_1 \sqsubseteq_M b.p_2}$	$, a \sqsubseteq_A b$
	C2	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_3 \sqsubseteq_M p_4}{p_1 + p_3 \sqsubseteq_M p_2 + p_4}$	
	C3	$\dfrac{p_i \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_i}{p_1 \mid_g p_2 \sqsubseteq_M q_1 \mid_g q_2}$	$, i \in \{1, 2\}$
	C4	$\dfrac{p_i \sqsubseteq_{CL(f^{-1}(M)\downarrow i)} p'_i}{(p_1 \ldots p_n)[f] \sqsubseteq_M (p'_1 \ldots p'_n)[f]}$	,for all $i \leq n$
ANNIHIL		$\dfrac{b \notin M}{nil \sqsubseteq_M b.p}$	
CONS		$\dfrac{p_1 \sqsubseteq_M p_2}{p_1 \sqsubseteq_N p_2}$	$, N \subseteq M$

Table 3.2: Proof system for finite terms (Continued on next page).

COM
$$\frac{p \equiv \sum_{i=1}^{n} a_i.p_i \;\; q \equiv \sum_{j=1}^{m} b_j.q_j}{\begin{array}{c} p|_g q =_M \sum_{(i,j)\in\{(i,j)|g(a_i,b_j)\ defined\}} g(a_i,b_j).(p_i|_g q_j) + \\ \sum_{i=1}^{n} a_i.(p_i|_g q) + \sum_{j=1}^{m} b_j.(p|_g q_j) \end{array}}$$

FUN F1
$$\frac{p_i \equiv nil}{(p_1 \ldots p_n)[f] =_M nil}$$

F2
$$(a_1.p_1 \ldots a_n.p_n)[f] =_M \begin{cases} f(a_1 \ldots a_n).(p_1 \ldots p_n)[f] \\ \text{if } f(a_1 \ldots a_n) \text{ is defined} \\ \\ nil \;\; \text{otherwise} \end{cases}$$

F3
$$\begin{array}{c} (p_1 \ldots p_i + q_i \ldots p_n)[f] =_M (p_1 \ldots p_i \ldots p_n)[f] + \\ (p_1 \ldots q_i \ldots p_n)[f] \end{array}$$

Table 3.2: Proof system for finite terms.

Also the relations R_i^{-1}, $i \in \{1 \ldots 4\}$ are M-bisimulations.
(see appendix A.) □

This shows that $(\ldots)[f]$ may be exchanged by action-prefixing and nondeterminism.

In table 3.2 we present a proof system S_f of the precongruence $\sqsubseteq_M$ over P_f.

We write $\vdash p_1 \sqsubseteq_M \; p_2$ if $p_1 \sqsubseteq_M \; p_2$ is provable by the rules of table 3.2.

Theorem 3.3 *The proof system of table 3.2 is sound, i.e.*

$$\vdash p_1 \sqsubseteq_M \; p_2 \;\Rightarrow\; p_1 \sqsubseteq_M \; p_2$$

Proof: By theorem 3.1 and proposition 3.6, 3.7, 3.8 and 3.9. □

We now turn to prove the completeness of the proof system S_f.

We do this by a similar proof technique as presented in [Hen 85], but we have to take into account that $\sqsubseteq_M$ is only a preorder, not an equivalence.

Definition 3.1 *Let $\Sigma \subseteq \Sigma'$. Let R be a relation over T_Σ and R' a relation over $T_{\Sigma'}$. Then R' is a conservative extension of R if $R' \cap T_{\Sigma'} \subseteq R$.*

Lemma 3.1 *Let $\Sigma \subseteq \Sigma'$ and let R and R' be preorders over T_Σ and $T_{\Sigma'}$ such that $R \subseteq R'$.*
Let S be a preorder over $T_{\Sigma'}$ such that:

1. *S is a conservative extension of R*
2. *$R' \subseteq S$*

3. *For each $p \in T_{\Sigma'}$ there exists a normalform $nf(p)$ in T_Σ such that $(p, nf(p)) \in R'$ and $(nf(p), p) \in R'$*

Then $R' = S$.

(In spirit this is the extension lemma of [Hen 85] extended to take account of preorders).

PROOF: Suppose $(p, q) \in S$. Then:

$$\text{from 3: } (p, nf(p)) \in R' \text{ and } (nf(q), q) \in R'$$

$$\text{from 2: } (nf(p), nf(q)) \in S$$

$$\text{from 1: } (nf(p), nf(q)) \in R$$

Therefore $(p, q) \in R'$ since $R \subseteq R'$. □

In the following theorem the precongruence induced by S1-S4 shall be named $\sqsubseteq^1_M$, and the precongruence induced by S1-S4, COM and F1-F3 shall be named $\sqsubseteq^2_M$.

Theorem 3.4 *$\sqsubseteq_M$ is the least precongruence which satisfies the axioms S1-S4, COM and F1-F3 of table 3.2*

PROOF: We apply lemma 3.1 with $\Sigma = \Sigma^{nf}$ and $\Sigma' = \Sigma^f$, $R = \sqsubseteq^1_M$,$R' = \sqsubseteq^2_M$ and $S = \sqsubseteq_M$.
As it has been shown in proposition 3.2, 3.7 and 3.9, $\sqsubseteq_M$ satisfies the axioms S1-S4, COM and F1-F3 of table 3.2. This establishes hypothesis 2 of lemma 3.1 Also by using the axioms COM and F1-F3 of table 3.2 every occurrence of $|_g$ and $(\ldots)[f]$ can be eliminated from terms in Σ^f. This establishes hypothesis 3. It remains to show that $\sqsubseteq_M$ is a conservative extension of $\sqsubseteq^1_M$. Let $\sqsubseteq^{'}_M$ be the precongruence satisfying $ACT \rightarrow$ and $SUM \rightarrow$. From theorem 3.1 and 3.2 we know $\vdash p_1 \sqsubseteq^1_M p_2$ iff $p_1 \sqsubseteq^{'}_M p_2$. A simple proof by structural induction will establish that for all $p_1, p_2 \in \Sigma^{nf} : p_1 \sqsubseteq_M p_2 \Rightarrow p_1 \sqsubseteq^{'}_M p_2$. □

Since table 3.2 induces the least precongruence the above theorem establishes the desired result:

Theorem 3.5 *The proof system S_f of table 3.2 is sound and complete i.e:*

$$\vdash p_1 \sqsubseteq_M p_2 \textit{ iff } p_1 \sqsubseteq^{'}_M p_2.$$

PROOF: By theorem 3.3 and 3.4 and the above remark. □

3.2 Regular expressions

Let us define the transition system of regular expressions $RE = (P_r, Act, \longrightarrow)$ where P_r is given by the following abstract syntax:

$$p \ ::= \ nil \Bigm| a.p_1 \Bigm| p_1 + p_2 \Bigm| x \Bigm| \mu x.p$$

The operational semantics is the smallest family of relations satisfying $ACT\rightarrow$, $SUM\rightarrow$ and $REC\rightarrow$ of definition 2.1.

x and $\mu x.$ introduce the wellknown concept of free and bound variables in expressions.

Definition 3.2 *We define the set of free variables $free(p)$ inductively:*

$$\begin{aligned}
free(nil) &= \emptyset \\
free(a.p) &= free(p) \\
free(p_1 + p_2) &= free(p_1) \cup free(p_2) \\
free(x) &= \{x\} \\
free(\mu x.p) &= free(p) \setminus \{x\}
\end{aligned}$$

Note how $\mu x.$ binds free occurrences of x in p in $\mu x.p$. A variable that is not free is called bound. Note too how $free$ also may be seen as an instance of the general scheme of definition 2.2. A variable may be seen as a place holder, and we may substitute a variable with an expression. We will use the notation $p[\bar{r}/\bar{x}]$, where $\bar{x} = (x_1 \ldots x_n)$ are free variables in p and $\bar{r} = (r_1 \ldots r_n)$ are n expressions, as the act of simultaneous substituting r_i for x_i in p taking account of change of bound variables.

This may be formalized in the following definition:

Definition 3.3 *$p[\bar{r}/\bar{x}]$ is defined structurally on p as follows:*

$$\begin{aligned}
nil[\bar{r}/\bar{x}] &\equiv nil \\
(a.p)[\bar{r}/\bar{x}] &\equiv a.(p[\bar{r}/\bar{x}]) \\
(p_1 + p_2)[\bar{r}/\bar{x}] &\equiv p_1[\bar{r}/\bar{x}] + p_2[\bar{r}/\bar{x}] \\
y[\bar{r}/\bar{x}] &\equiv \begin{cases} r_i & \textit{if } y \equiv x_i \\ y & \textit{if } y \notin \bar{x} \end{cases} \\
(\mu y.p)[\bar{r}/\bar{x}] &\equiv \begin{cases} \mu y.p[\bar{r}/\bar{x}] \;\; \textit{if } y \textit{ not in } \bar{x} \textit{ nor free in } \bar{r} \\ \\ \mu z.(p[z/y][\bar{r}/\bar{x}]) \\ \textit{otherwise for some } z \textit{ not in } \bar{x} \textit{ nor free in } \mu y.p \textit{ or } \bar{r} \end{cases}
\end{aligned}$$

This definition is similar to the definition of substitution in [Mil 81] and has resemblance to substitution in the λ-calculus (see e.g. [Bar 85]).

In the above definition $\equiv$ means syntactic equivalence upto change of bound variables.

We now turn to investigate some useful properties of substitution; but first we introduce the concept of guarded variables, which divides the set of free variables of terms into two groups: guarded and unguarded variables:

Definition 3.4 *We define the set of unguarded variables $UG(p)$ inductively:*

$$\begin{array}{rcl} UG(nil) & = & \emptyset \\ UG(a.p) & = & \emptyset \\ UG(p_1 + p_2) & = & UG(p_1) \cup UG(p_2) \\ UG(x) & = & \{x\} \\ UG(\mu x.p) & = & UG(p) \setminus \{x\} \end{array}$$

A variable which is not unguarded is said to be guarded.

Properties of substitution

Property 3.1 *Whenever $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ then*

<u>either</u> *for some $p'' : p \xrightarrow{a} p''$ and $p' = p''[\bar{r}/\bar{x}]$*

<u>or</u> *for some $x_i \in UG(p) : r_i \xrightarrow{a} p'$*

PROOF: by induction on inferences observing the structure of p. (see appendix A.) □

Property 3.2 *Whenever $p' : p \xrightarrow{a} p'$ then $p[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$*

PROOF: by induction on inferences observing the structure of p. (see appendix A.) □

Property 3.3 *Whenever $x_i \in UG(p)$ and $r_i \xrightarrow{a} p'$ then $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$*

PROOF: by induction on inferences observing the structure of p. (see appendix A.) □

Property 3.4 *If no x_i is free in p then $p[\bar{r}/\bar{x}] \equiv p$*

PROOF: by structural induction. (see appendix A.) □

Property 3.5 *If $\bar{x}$ and $\bar{y}$ are disjoint then:*

$$p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$$

PROOF: by structural induction. (see appendix A.) □

Before presenting a proof system characterizing $\sqsubseteq_M$ over P_r we have to extend $\sqsubseteq_M$ to take account of the possibility of free variables in the expressions.

There are basically two possible strategies for doing this, the one being defined as:

$$p \sqsubseteq_M^{'} q \text{ iff } \forall \bar{r}.p[\bar{r}/\bar{x}] \sqsubseteq_M q[\bar{r}/\bar{x}]$$

where $\bar{x}$ contains all the free variables in p and q and the r_i's in $\bar{r}$ are closed. This strategy may be called instantiation.

The other strategy consists of extending the function $M\mathcal{B}$ to take account of free variables, the extension ensuring that the two processes have the same set of unguarded variables i.e:

Definition 3.5 $(p, q) \in M\mathcal{B}''(R)$ *iff:*

1. $UG(p) = UG(q)$
2. $\forall a \in Act.p \xrightarrow{a} p' \Rightarrow \exists q'.\exists b \in Act.q \xrightarrow{b} q' \;\&\; a \sqsubseteq_A b \;\&\; (p', q') \in R$
3. $\forall a \in M.q \xrightarrow{a} q' \Rightarrow \exists p'.\exists b \in Act.p \xrightarrow{b} p' \;\&\; b \sqsubseteq_A a \;\&\; (p', q') \in R$

This strategy may be called refining, and a relation R such that $R \subseteq M\mathcal{B}''(R)$ is called a refined M-bisimulation. We write $p \sqsubseteq_M^{''} q$ if there exists a refined M-bisimulation containing (p, q).

Despite the different nature of the two strategies they are equal in expressability:

Proposition 3.10

$$p \sqsubseteq_M^{''} q \textit{ iff } p \sqsubseteq_M^{'} q$$

PROOF:

$\Rightarrow$ We show that the relation

$$R = \{(p[\bar{r}/\bar{x}], q[\bar{r}/\bar{x}]) \mid p \sqsubseteq_M^{''} q\} \cup Id$$

is an M-bisimulation.

$\Leftarrow$ We prove this by showing

$$p \not\sqsubseteq_m^{''} q \Rightarrow \not\sqsubseteq_m^{'} q$$

For the full proof see appendix A. □

From now on we shall use M-bisimulation for a refined M-bisimulation, since the results of refined M-bisimulation holds for closed finite terms and since M-bisimulation and refined M-bisimulation coincides on closed finite terms.

Let us now turn to characterize $\sqsubseteq_M$ for regular expressions. In table 3.3 we present the proof system S_{rt} of the precongruence $\sqsubseteq_M$ over P_r.

Note the interesting properties of R4 and R5. These rules state that $\mu x.p_1$ is the least prefix point and the greatest postfix point of the recursive equation

SUM	S1	$p_1 + (p_2 + p_3) =_M (p_1 + p_2) + p_3$
	S2	$p_1 + p_2 =_M p_2 + p_1$
	S3	$p_1 + p_1 =_M p_1$
	S4	$p_1 + nil =_M p_1$
PREORD	P1	$p \sqsubseteq_M p$
	P2	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_3}{p_1 \sqsubseteq_M p_3}$
	P3	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}{p_1 =_M p_2}$
	P4	$\dfrac{p_1 =_M p_2}{p_1 \sqsubseteq_M p_2 \; p_2 \sqsubseteq_M p_1}$
PRECONG	C1	$\dfrac{p_1 \sqsubseteq_M p_2 \; p_3 \sqsubseteq_M p_4}{p_1[p_3/x] \sqsubseteq_M p_2[p_4/x]}$
	C2	$\dfrac{p_1 \sqsubseteq_M p_2}{\mu x.p_1 \sqsubseteq_M \mu x.p_2}$
REC	R1	$\mu x.p =_M \mu y.(p[y/x]), y$ not free in $\mu x.p$
	R2	$\mu x.p =_M p[\mu x.p/x]$
	R3	$\mu x.(p + x) =_M \mu x.p$
	R4	$\dfrac{p_1[p_2/x] \sqsubseteq_M p_2}{\mu x.p_1 \sqsubseteq_M p_2}$,x guarded in p_1
	R5	$\dfrac{p_2 \sqsubseteq_M p_1[p_2/x]}{p_2 \sqsubseteq_M \mu x.p_1}$,x guarded in p_1
ANNIHIL		$\dfrac{b \notin M}{nil \sqsubseteq_M b.p}$
CONS		$\dfrac{p_1 \sqsubseteq_M p_2}{p_1 \sqsubseteq_N p_2}$ $, N \subseteq M$
PREFIX		$\dfrac{a \sqsubseteq_A b}{a.p \sqsubseteq_M b.p}$

Table 3.3: Proof system for nondeterminism.

$x = p_1$ provided x is guarded in p_1. Together R4 and R5 state that $\mu x.p_1$ is the unique solution, as was also shown in [Mil 81].

First we prove the soundness of the proof system.

We shall write $\vdash p_1 \sqsubseteq_M\ p_2$ if $p_1 \sqsubseteq_M\ p_2$ is provable by the rules from table 3.3.

Proposition 3.11 *$\sqsubseteq_M$ is a precongruence with respect to the operators of Σ^r. i.e. $\sqsubseteq_M$ satisfies C1 and C2 of table 3.3*

PROOF: The relation

$$R_1 = \{(p_1[p_3/x], p_2[p_4/x]) \mid p_1 \sqsubseteq_M\ p_2\ ,\ p_3 \sqsubseteq_M\ p_4\} \cup \sqsubseteq_M$$

is an M-bisimulation, and the relation

$$\begin{aligned} R_2 = \{(p[\mu x.p_1/x], p[\mu x.p_2/x]) \\ \mid p_1 \sqsubseteq_M\ p_2\ ,\ p_1, p_2, p \in \Sigma^r\ ,\ free(p) \subseteq \{x\}\} \cup \sqsubseteq_M \end{aligned}$$

is an M-bisimulation. The result follows by taking $p \equiv x$. We have chosen only to prove the case where $free(p) \subseteq \{x\}$, the result easy generalizes to general open expressions.

To see that $R_2 \subseteq M\mathcal{B}(R_2)$ we use induction on the length of inferences. (For the full proof see appendix A.) □

Proposition 3.12 *The following holds for all $p_1, p_2, p_3 \in T_{\Sigma^r}$:*

$$\begin{aligned} p_1 + (p_2 + p_3) &\simeq_M (p_1 + p_2) + p_3 \\ p_1 + p_2 &\simeq_M p_2 + p_1 \\ p_1 + p_1 &\simeq_M p_1 \\ p_1 + nil &\simeq_M p_1 \end{aligned}$$

Here $\simeq_M$ means that both $\sqsubseteq_M$ and $\sqsupseteq_M$ hold.

PROOF: As for proposition 3.1

$$UG(p_1 + (p_2 + p_3)) = UG(p_1) \cup UG(p_2) \cup UG(p_3) = UG((p_1 + p_2) + p_3)$$

etc. □

Proposition 3.13 *If $a \sqsubseteq_A\ b$ then $a.p \sqsubseteq_M\ b.p$ for all $p \in T_{\Sigma^r}$*

PROOF: The relation

$$R = \{(a.p, b.p) \mid a \sqsubseteq_A\ b\} \cup Id$$

is an M-bisimulation.
To see this observe that $a.p \xrightarrow{a} p$ which $b.p$ can match by $b.p \xrightarrow{b} p$ and by definition of $R : a \sqsubseteq_A\ b$ and $(p, p) \in Id \subseteq R$. Also if $b \in M$ $a.p$ can match $b.p$, otherwise the result holds trivially. □

Proposition 3.14 *If $b \notin M$ then $nil \sqsubseteq_M\ b.p$ for all $p \in T_{\Sigma^r}$.*

Proof: The relation

$$R = \{(nil, b.p) \mid b \notin M\}$$

is an M-bisimulation, since $UG(nil) = \emptyset = UG(b.p)$ and $nil \overset{a}{\not\to}$ and $b.p \overset{a}{\not\to}$ for any $a \in M$. □

Proposition 3.15 *$\mu x.p$ has exactly the derivations of its unfold i.e.*

$$\mu x.p \simeq_M p[\mu x.p/x]$$

Proof: The relation

$$R = \{(\mu x.p, p[\mu x.p/x]) \mid p \in T_{\Sigma^r}\} \cup Id$$

is an M-bisimulation.
To see that $R \subseteq M\mathcal{B}(R)$ observe that
if $\mu x.p \xrightarrow{a} p'$ then this is due to $p[\mu x.p/x] \xrightarrow{a} p'$.
Also if $p[\mu x.p/x] \xrightarrow{a} p', a \in M$ then by $REC\!\to$ $\mu x.p \xrightarrow{a} p'$ which obviously is the matching move.
Also R^{-1} is an M-bisimulation. □

Proposition 3.16 *If y is not free in $\mu x.p$ then $\mu x.p \simeq_M \mu y.(p[y/x])$ i.e. $\sqsubseteq_M$ satisfies R1 of table 3.3*

Proof: Let $p_1 = \mu x.p$ and $p_2 = \mu y.(p[y/x])$. By proposition 3.15 $p_1 \simeq_M p[p_1/x]$ and $p_2 \simeq_M p[p_2/x]$. We now show that

$$R = \{(q[p_1/x], q[y/x][p_2/y]) \mid q \in T_{\Sigma^r}\}$$

is an M-bisimulation. The result follows by taking $q \equiv p$ and applying proposition 3.15.
To see that $R \subseteq M\mathcal{B}(R)$ we use induction on the number of inferences.
The full proof is presented in appendix A. □

Proposition 3.17 *If x is guarded in p_1 then*
if $p_1[p_2/x] \sqsubseteq_M\ p_2$ then $\mu x.p_1 \sqsubseteq_M\ p_2$
and
if $p_2 \sqsubseteq_M\ p_1[p_2/x]$ then $p_2 \sqsubseteq_M\ \mu x.p_1$.
i.e. $\sqsubseteq_M$ satisfies R4 and R5 of table 3.3.

Proof: Since $\mu x.p \simeq_M p[\mu x.p/x]$ it is enough to show that if $p_1 \sqsubseteq_M\ p[p_1/x]$ & $p[p_2/x] \sqsubseteq_M\ p_2$ & $p_1 \sqsubseteq_M\ p_2$ then $p[p_1/x] \sqsubseteq_M\ p[p_2/x]$.
Let

$$R = \{(q[p_1/x], q[p_2/x]) \mid q \in T_{\Sigma^r}\}$$

We wish to show that R is an M-bisimulation upto '$\sqsubseteq_M$ ' and the result follows by taking $q = p$ and applying proposition 1.7.
The full proof is presented in appendix A. □

Proposition 3.18 *$\mu x.p \simeq_M \mu x.(p+x)$ i.e. $\sqsubseteq_M$ satisfies R3 of table 3.3*

PROOF: The relation

$$R = \{(q[\mu x.p/x], q[\mu x.(p+x)/x]) \mid q \in T_{\Sigma^r}\}$$

is an M-bisimulation. The result follows by taking $q \equiv x$.
To see that $R \subseteq M\mathcal{B}(R)$ we use induction on the number of inferences.
The full proof is presented in appendix A. □

By these propositions we arrive at the desired result:

Theorem 3.6 *The proof system S_{rt} is sound i.e.*

$$p \sqsubseteq_M q \Rightarrow \vdash p \sqsubseteq_M q$$

PROOF: By propositions 3.11-3.17.By proposition 1.8 we know that $\sqsubseteq_M$ is a preorder, and proposition 1.10 shows that CONS is sound with $\leq_A = \sqsubseteq_A$. □

We now turn to prove the completeness of the proof system. But first we need a lemma which resembles the properties 3.1-3.5.

Lemma 3.2 1. *if $\bar{x}$ is not free in p then $\vdash p[\bar{r}/\bar{x}] =_M p$*

2. *if $\bar{x}$ and $\bar{y}$ are disjoint then $\vdash p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] =_M p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$*

PROOF:

1. if $\bar{x}$ is not free in p then $p[\bar{q}/\bar{x}] \equiv p$, by definition of substitution and by P1: $\vdash p =_M p$
2. We prove this by structural induction on p (see appendix A).

□

We now prove a theorem which in spirit is similar to theorem 5.7 of [Mil 81], but the details as well as the overall result are quite different. The result of this theorem together with an equational characterization analogous to theorem 5.8 of [Mil 81] yields the completeness proof of the system of table 3.3. Also the completeness proof is similar in spirit to that of theorem 5.9 in [Mil 81] but in detail it is rather different.

Theorem 3.7 *Let $\bar{x} = (x_1 \dots x_m)$ and $\bar{y} = (y_1 \dots y_n)$ be distinct variables, and let $\bar{q} = (q_1 \dots q_m)$ and $\bar{q}' = (q'_1 \dots q'_m)$ be expressions with free variables in $(\bar{x}, \bar{y})$ in which each x_i is guarded. Then there exist expressions $\bar{p} = (p_1 \dots p_m)$ and $\bar{p}' = (p'_1 \dots p'_m)$ with free variables in $\bar{y}$ such that*

$$\vdash p_i \sqsubseteq_M q_i[\bar{p}/\bar{x}] \text{ and } \vdash p'_i \sqsubseteq_M q'_i[\bar{p}'/\bar{x}]$$

and moreover this and $\vdash q_i \sqsubseteq_M q'_i$ implies for $(i \leq m)$

$$\vdash p_i \sqsubseteq_M p'_i$$

Proof: By induction on m.

Basis For $m = 1$ we choose $p_1 \equiv \mu x_1.q_1$ and $p'_1 \equiv \mu x_1.q'_1$. The first result follows immediately from R2 and R4, the second follows from C2.

Step Assume the result for m. Now let $\bar{q} = (q_1 \ldots q_m)$ and q_{m+1} and $\bar{q}' = (q'_1 \ldots q'_m)$ and q'_{m+1} be expressions with free variables in $(\bar{x}, x_{m+1}, \bar{y})$ in which x_i is guarded $(i \leq m+1)$. We first find expressions $\bar{p} = (p_1 \ldots p_m)$ and p_{m+1} and $\bar{p}' = (p'_1 \ldots p'_m)$ and p'_{m+1} such that for $(i \leq m+1)$:

$$\begin{aligned} &\vdash p_i \sqsubseteq_M \; q_i[\bar{p}/\bar{x}, p_{m+1}/x_{m+1}] \text{ and} \\ &\vdash p'_i \sqsubseteq_M \; q'_i[\bar{p}'/\bar{x}, p'_{m+1}/x_{m+1}] \end{aligned} \tag{3.3}$$

For this purpose, first set

$$\begin{aligned} r_{m+1} &\equiv \mu x_{m+1}.q_{m+1} \text{ and} \\ r'_{m+1} &\equiv \mu x_{m+1}.q'_{m+1} \end{aligned} \tag{3.4}$$

and for $(i \leq m)$:

$$\begin{aligned} r_i &\equiv q_i[r_{m+1}/x_{m+1}] \text{ and} \\ r'_i &\equiv q'_i[r'_{m+1}/x_{m+1}] \end{aligned} \tag{3.5}$$

Since r_i and r'_i have free variables in $(\bar{x}, \bar{y})$ with $\bar{x}$ guarded, by induction there are expressions $\bar{p} = (p_1 \ldots p_m)$ and $\bar{p}' = (p'_1 \ldots p'_m)$ with free variables in $\bar{y}$ such that:

$$\begin{aligned} &\vdash p_i \sqsubseteq_M \; r_i[\bar{p}/\bar{x}] \text{ and} \\ &\vdash p'_i \sqsubseteq_M \; r'_i[\bar{p}'/\bar{x}] \end{aligned} \tag{3.6}$$

If we choose

$$\begin{aligned} p_{m+1} &\equiv r_{m+1}[\bar{p}/\bar{x}] \text{ and} \\ p'_{m+1} &\equiv r'_{m+1}[\bar{p}'/\bar{x}] \end{aligned} \tag{3.7}$$

we may rewrite (3.6) using (3.5) to:

$$\begin{aligned} &\vdash p_i \sqsubseteq_M \; q_i[r_{m+1}/x_{m+1}][\bar{p}/\bar{x}] \text{ and} \\ &\vdash q'_i[r'_{m+1}/x_{m+1}][\bar{p}'/\bar{x}] \sqsubseteq_M \; p'_i \end{aligned} \tag{3.8}$$

if we appeal to lemma 3.2 and use (3.7) we obtain for $(i \leq m)$:

$$\begin{aligned} &\vdash p_i \sqsubseteq_M \; q_i[\bar{p}/\bar{x}, p_{m+1}/x_{m+1}] \text{ and} \\ &\vdash q'_i[\bar{p}'/\bar{x}, p'_{m+1}/x_{m+1}] \sqsubseteq_M \; p'_i \end{aligned} \tag{3.9}$$

To obtain (3.3) for $i = m+1$, we deduce from (3.4) and (3.7) since x_{m+1} is not in $\bar{p}$ nor $\bar{p}'$:

$$\begin{aligned} p_{m+1} &\equiv \mu x_{m+1}(q_{m+1}[\bar{p}/\bar{x}]) \text{ and} \\ p'_{m+1} &\equiv \mu x_{m+1}(q'_{m+1}[\bar{p}'/\bar{x}]) \end{aligned} \tag{3.10}$$

and hence by R2, P4 and lemma 3.2 we obtain: (since x_{m+1} is not free in $\bar{p}$ nor $\bar{p}'$)

$$\begin{aligned} &\vdash p_{m+1} \sqsubseteq_M \ q_{m+1}[\bar{p}/\bar{x}, p_{m+1}/x_{m+1}] \text{ and} \\ &\vdash q'_{m+1}[\bar{p}'/\bar{x}, p'_{m+1}/x_{m+1}] \sqsubseteq_M \ p'_{m+1} \end{aligned} \tag{3.11}$$

For the second part, observe that the induction yields for $(i \leq m)$:

$$\vdash r_i \sqsubseteq_M \ r'_i \ \Rightarrow \vdash p_i \sqsubseteq_M \ p'_i \tag{3.12}$$

If $\vdash q_i \sqsubseteq_M \ q'_i$ $(i \leq m+1)$ then also

$$\vdash r_{m+1} \sqsubseteq_M \ r'_{m+1} \tag{3.13}$$

by C2 and this yields

$$\vdash r_i \sqsubseteq_M \ r'_i \tag{3.14}$$

by C1 so $\vdash p_i \sqsubseteq_M \ p'_i$ $(i \leq m)$ to see $\vdash p_{m+1} \sqsubseteq_M \ p'_{m+1}$ use (3.13) and (3.14) and C1.

□

Note how this theorem extends R4 and R5 of table 3.3 to systems of simultaneous recursive processes.

Theorem 3.8 *For any expression p, with free variables in $\bar{y}$, there exist expressions $p_1 \ldots p_k$ $(k \geq 1)$ with free variables in $\bar{y}$ satisfying k equations:*

$$\vdash p_i = \sum_{j=1}^{n_i} a_{ij} . p_{f(i,j)} + \sum_{j=1}^{m_i} y_{g(i,j)} \quad (k \geq 1)$$

moreover

$$\vdash p = p_1$$

Proof: By structural induction on p.
(The full proof is presented in appendix A.) □

Theorem 3.9 *If $p \sqsubseteq_M \ p'$ then $\vdash p \sqsubseteq_M \ p'$*

Proof: Let p and p' have free variables in $\bar{y}$. By theorem 3.8 there are equations $\vdash p =_M p_1$ and $\vdash p' =_M p'_1$ and

$$p_i =_M \sum_{j=1}^{m_i} a_{ij} . p_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)} \quad (i \leq k)$$

$$p'_i =_M \sum_{j=1}^{m'_i} a'_{ij}.p_{f'(i,j)} + \sum_{j=1}^{n'_i} y_{g'(i,j)} \quad (i \leq k)$$

Now let $I = \{(i,i') \mid p_i \sqsubseteq_M \ p'_i\}$. Clearly $(1,1) \in I$ since $\vdash p =_M p_1$ and $\vdash p' =_M p'_1$ and by soundness (Theorem 3.6). This implies $p_1 \sqsubseteq_M \ p$ and $p' \sqsubseteq_M \ p'_1$ and $p \sqsubseteq_M \ p'$ so by transitivity of $\sqsubseteq_M$ $p_1 \sqsubseteq_M \ p'_1$. Moreover p_i and p'_i must have equal sets of unguarded variables and every move a of p_i can be matched by a move b of p'_i with $a \sqsubseteq_A \ b$. Also if $b \in M$ then the b-action of p'_i may be matched by the a-action of p_i and if $b \notin M$ then p_i does not have to match b. So the following holds

1. There exists a relation
   $J_{ii'} = \{(j,j') \mid a_{ij} \sqsubseteq_A \ a'_{i'j'} \ \& \ (f(i,j), f'(i',j')) \in I\}$
2. There exists a set $J^c_{ii'} = \{j' \in 1 \ldots m'_{i'} \mid \neg\exists j.(j,j') \in J_{ii'}\}$
3. $\vdash \sum_{j=1}^{n_i} y_{g(i,j)} = \sum_{j'=1}^{n'_i} y_{g'(i',j')}$

Moreover $J_{ii'}$ is a total and surjective relation between $\{1 \ldots m_i\}$ and $\{1 \ldots m'_{i'}\} \setminus J^c_{ii'}$. Note also for all $j' \in J^c_{ii'}$ that $a'_{i'j'} \notin M$ since there otherwise would have been a matching a_{ij} and $(j,j') \in J_{ii'}$.

We now consider the formal equations, two for each $(i,i') \in I$

$$x_{ii'} = \sum_{(j,j') \in J_{ii'}} a_{ij}.x_{f(i,j)f'(i',j')} + \sum_{j=1}^{n_i} y_{g(i,j)}$$

$$x_{ii'} = \sum_{(j,j') \in J_{ii'}} a'_{i'j'}.x_{f(i,j)f'(i',j')} + \sum_{j' \in J^c} a'_{i'j'}.p'_{f'(i',j')} + \sum_{j'=1}^{n'_i} y_{g'(i',j')}$$

where $x_{ii'}$ is not in $\bar{y}$

These equations are provable satisfied when instantiated to p_i and $p'_{i'}$.

To see this the typical equations become:

$$p_i = \sum_{(j,j') \in J_{ii'}} a_{ij}.p_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)}$$

$$p'_{i'} = \sum_{(j,j') \in J_{ii'}} a'_{i'j'}.p'_{f'(i',j')} + \sum_{j' \in J^c} a'_{i'j'}.p'_{f'(i',j')} + \sum_{j'=1}^{n'_i} y_{g'(i',j')}$$

The first is provable since J is total and the second since J is total on $\{1 \ldots m'_{i'}\}$, and J^c covers the rest. The right hand sides differ at most by repeated summands which may be eliminated by S1, S2 and S3.

Now let

$$q_i \equiv \sum_{(j,j')\in J_{ii'}} a_{ij}.x_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)}$$

$$q'_{i'} \equiv \sum_{(j,j')\in J_{ii'}} a'_{i'j'}.x_{f'(i',j')} + \sum_{j\in J^c} a'_{i'j'}.p'_{f'(i',j')} + \sum_{j'=1}^{n'_i} y_{g'(i',j')}$$

Since J is total and surjective we know that

$$\vdash \sum_{(j,j')\in J} a_{ij}.x_{f(i,j)} \sqsubseteq_M \sum_{(j,j')\in J} a'_{i'j'}.x_{f'(i',j')} \tag{3.15}$$

by repeated use of PREFIX and C2.

By the remark that for $j \in J^c_{ii'} : a'_{i'j'} \notin M$ we know that

$$\vdash nil =_M \sum_{j\in J^c_{ii'}} nil \sqsubseteq_M \sum_{j\in J^c_{ii'}} a'_{i'j'}.p'_{f'(i',j')}$$

by ANNIHIL, S4 and C2 used repeatedly.

So by this and (3.15) we know

$$\vdash q_{ii'} \sqsubseteq_M q'_{ii'}$$

again by using C2 and S4 to eliminate *nil* in every summand.

Also we know that

$$\vdash p_i \sqsubseteq_M q_i[\bar{p}/\bar{x}]$$

and

$$\vdash q'_{i'}[\bar{p}'/\bar{x}] \sqsubseteq_M p'_{i'}$$

so by theorem 3.7 we know

$$\vdash p_i \sqsubseteq_M p'_{i'} \text{ for every } (i,i') \in I$$

especially $\vdash p_1 \sqsubseteq_M p'_1$ which proves the theorem.

3.3 An alternative characterization of recursive processes

In this part we present an alternative to the rules R4 and R5 of table 3.3 for regular expressions, characterizing $\mu x.p$ as the least postfixed point and the greatest prefixed point of the equation $x = p$, provided x is guarded in p.

The alternative characterization resembles the characterization of least fixed points in denotational semantics [Sto 77].

We want to characterize $\mu x.p$ by its unfoldings and we show that under certain conditions we may do so.

Definition 3.6 *An action $\top_A \in Act$ is a top action iff $\forall a \in Act. a \sqsubseteq_A \top_A$.*

Definition 3.7 $\top_p = \mu x.\top_A.x$

Proposition 3.19 *If M does not contain $\top_A$ then for all closed $p : p \sqsubseteq_M \top_p$*

Proof: The relation

$$R = \{(p, \top_p) \mid \text{p is closed}\}$$

is an M-bisimulation.
To see that $R \subseteq M\mathcal{B}(R)$ observe that $UG(p) = \emptyset$ since p is closed, also $UG(\top_p) = \emptyset$ by definition 3.4. If $p \xrightarrow{a} p'$ then $\mu x.\top_A.x \xrightarrow{\top_A} \mu x.\top_A.x$ by $REC\rightarrow$ and by definition of substitution. Clearly $a \sqsubseteq_A \top_A$ and $(p', \top_p) \in R$. Since $\top_A$ is not in M, and it is the only action of $\top_p$ the second condition of $M\mathcal{B}$ holds trivially. □

This shows that under the condition that Act has a top action $\top_A$ and M does not contain this top action, we have a greatest process $\top_p$.

Throughout this section we shall assume that Act contains a top action and M does not contain this top action.

The above definition of $\top_p$ may be used in defining the unfoldings of $\mu x.p$.

Definition 3.8

$$p^{n+1}[\top_p/x] = \begin{cases} \top_p & \textit{if } n = 0 \\ p[p^n[\top_p/x]/x] & \textit{if } n > 0 \end{cases}$$

Proposition 3.20 *For all $n \in \omega$. $p^{n+1}[\top_p/x] \sqsubseteq_M p^n[\top_p/x]$*

Proof: By induction on n.
The case 0 holds by proposition 3.19.
To see the case for $n+1$ assume $p^n[\top_p/x] \sqsubseteq_M p^{n-1}[\top_p/x]$. Since $\sqsubseteq_M$ is a precongruence with respect to the operators of Σ^r by proposition 3.11:$p \sqsubseteq_M p$ and the induction hypothesis implies $p^{n+1}[\top_p/x] \sqsubseteq_M p^n[\top_p/x]$. □

This shows that $(p^n[\top_p/x])_n$ is a decreasing chain in P_r.

Proposition 3.21 *if $free(p) \subseteq \{x\}$ then $\mu x.p \sqsubseteq_M p^n[\top_p/x]$ for all n*

Proof: By induction on n.
The case $n = 0$ holds by proposition 3.19.
To see the case for $n+1$, assume that $\mu x.p \sqsubseteq_M p^n[\top_p/x]$. Since $\sqsubseteq_M$ is a precongruence $p \sqsubseteq_M p$ and the above implies $p[\mu x.p/x] \sqsubseteq_M p[p^n[\top_p/x]/x]$. By proposition 3.15 $\mu x.p \sqsubseteq_M p^{n+1}[\top_p/x]$. □

This proposition shows that $\mu x.p$ is a lower bound for the chain $(p^n[\top_p/x])_n$, provided that $\mu x.p$ is closed.

But we want to do better than that, we want to show that $\mu x.p$ is the greatest lower bound of the chain $(p^n[\top_p/x])_n$.

For the theoretical development to work we have to characterize $\sqsubseteq_M$ in a different way. We do this by defining a decreasing series of preorders over P_r and showing that under certain conditions $\sqsubseteq_M$ coincides with the greatest lower bound of this chain.

Definition 3.9 *$p \sqsubseteq^0_M\ q$ is always true.*
$p \sqsubseteq^{n+1}_M\ q$ iff:

1. $\forall a \in Act.p \xrightarrow{a} p' \ \Rightarrow\ \exists q'.\exists b.q \xrightarrow{b} q' \ \&\ a \sqsubseteq_A\ b \ \&\ p' \sqsubseteq^n_M\ q'$

2. $\forall a \in Act.q \xrightarrow{a} q' \ \Rightarrow\ \exists p'.\exists b.p \xrightarrow{b} p' \ \&\ b \sqsubseteq_A\ a \ \&\ p' \sqsubseteq^n_M\ q'$

$p \sqsubseteq'_M\ q$ iff $\forall n \geq 0.p \sqsubseteq^n_M\ q$ i.e. $\sqsubseteq'_M\ = \bigcap_{n=1}^{\infty} \sqsubseteq^n_M$

This states that $\sqsubseteq^0_M\ = P_r^2$ and that $\sqsubseteq^{n+1}_M\ = M\mathcal{B}(\sqsubseteq^n_M\ )$

Definition 3.10 *A process system $P = (pr, Act, \longrightarrow)$ is said to be <u>imagefinite</u> iff:*

1. *$\forall p.\forall a.\{p' \mid \exists b \sqsupseteq a.p \xrightarrow{b} p'\}$ is finite and*

2. *$\forall q.\forall a \in M.\{q' \mid \exists b \sqsubseteq a.q \xrightarrow{b} q'\}$ is finite*

This definition is an extension of the imagefinite condition found in e.g. [Mil 80] saying p is imagefinite if $\{p' \mid p \xrightarrow{a} p'\}$ is finite. The above definition takes the ordering on actions into account.

We now turn to show when $\sqsubseteq_M$ and $\sqsubseteq'_M$ coincide.

Definition 3.11 *A function F on a complete lattice D is anticontinuous iff for every decreasing chain $x_1 \sqsupseteq x_2 \sqsupseteq x_3 \sqsupseteq \ldots x_n \sqsupseteq \ldots$ of D elements $F(\sqcap_n x_n) = \sqcap_n F(x_n)$.*

It follows by classic fix point theory that if a function is anticontinuous on a complete lattice then the maximal fix point is: $\sqcap_n F^n(D)$.

Proposition 3.22 *If P is imagefinite then $M\mathcal{B}$ is anticontinuous.*
Thus $\sqsubseteq_M = \bigcap_n M\mathcal{B}^n(P_r)$ where $M\mathcal{B}^0 = Id$ and $M\mathcal{B}^{n+1} = M\mathcal{B}^n \circ M\mathcal{B}$

Proof: Let $R_1 \supseteq R_2 \supseteq R_3 \supseteq \ldots R_n \supseteq \ldots$ be a decreasing chain of binary relations over P_r. We must prove $M\mathcal{B}(\bigcap_n R_n) = \bigcap_n M\mathcal{B}(R_n)$.
The $' \subseteq'$-direction follows directly from monotonicity of $M\mathcal{B}$ and $\bigcap_n R_n \subseteq R_i$ for all $i \in \omega$.
For the $' \supseteq'$-direction let $(p, q) \in \bigcap_n M\mathcal{B}(R_n)$. If $p \xrightarrow{a} p'$ we must find a matching move for q i.e. b and q' such that $q \xrightarrow{b} q'$ & $a \sqsubseteq_A\ b$ with $(p', q') \in \bigcap_n R_n$. Thus for all n there exist b_n and q'_n such that $q \xrightarrow{b_n} q'_n$ & $a \sqsubseteq_A b$ & $(p', q'_n) \in R_n$.

By the first imagefinite clause there are only finitely many q_n's. This means that there exist b and q' such that $q \xrightarrow{b} q'$ & $a \sqsubseteq_A b$ & $(p', q') \in R_n$ for infinitely many $n \in \omega$. Since R_n is decreasing in n $(p', q') \in R_n$ for all $n \in \omega$ and thus $(p', q') \in \bigcap_n R_n$.
Also if $q \xrightarrow{a} q'$, $a \in M$ we must find b and p' such that $p \xrightarrow{b} p'$ and $b \sqsubseteq_A a$ and $(p', q') \in \bigcap_n R_n$. Now $(p, q) \in \bigcap_n M\mathcal{B}(R_n)$ iff $\forall n \in \omega.(p, q) \in M\mathcal{B}(R_n)$. Thus for all n there exist p'_n and b_n such that $p \xrightarrow{b_n} p'_n$ with $b_n \sqsubseteq_A a$ and $(p'_n, q) \in R_n$. By the second imagefinite clause there are only finitely many p_n's. This means that there exist b and p' such that $p \xrightarrow{b} p'$ & $b \sqsubseteq_A a$ & $(p', q') \in R_n$ for infinitely many $n \in \omega$. Since R_n is decreasing in n $(p', q') \in R_n$ for all $n \in \omega$ and thus $(p', q') \in \bigcap_n R_n$. □

This proposition shows that $\sqsubseteq_M = \sqsubseteq'_M$ provided P is imagefinite. Throughout this section we shall assume that P is imagefinite.

Before presenting the alternative characterization of $\mu x.p$ we need a few properties of $\sqsubseteq^n_M$.

Proposition 3.23 *For all $n \in \omega$. $p \sqsubseteq^n_M q$ implies $r[p/x] \sqsubseteq^n_M r[q/x]$ for all $r \in P_r$*

PROOF: By induction on n.
The case $n = 0$ is trivial.
To prove $p \sqsubseteq^{n+1}_M q$ implies $r[p/x] \sqsubseteq^{n+1}_M r[q/x]$. Assume the case for n and assume $p \sqsubseteq^{n+1}_M q$. Then $p \sqsubseteq^n_M q$ holds since $\sqsubseteq^{n+1}_M \subseteq \sqsubseteq^n_M$ so $r[p/x] \sqsubseteq^n_M r[q/x]$.
If $r[p/x] \xrightarrow{a} r'$ then by property 3.1:

either $r \xrightarrow{a} r''$ and $r' = r''[p/x]$. Then also $r[q/x] \xrightarrow{a} r''[q/x]$ and $a \sqsubseteq_A a$ and $r''[p/x] \sqsubseteq^n_M r''[q/x]$ which obviously is the matching move.

or x is unguarded in r and $p \xrightarrow{a} p'$ then $q \xrightarrow{b} q'$ with $a \sqsubseteq_A b$ and $p' \sqsubseteq^n_M q'$ which is the matching move.

Also if $r[q/x] \xrightarrow{a} r'$, $a \in M$ then by property 3.1:

either $r \xrightarrow{a} r''$ and the case is as above.

or x is unguarded in r and $q \xrightarrow{a} q'$. Then $p \xrightarrow{b} p'$ with $b \sqsubseteq_A a$ and $p' \sqsubseteq^n_M q'$ which is the matching move.

□

Proposition 3.24 *For all $n \in \omega.p^n[\top_P/x] \sqsubseteq^n_M \mu x.p$, whenever x is guarded in p*

PROOF: By induction on n.
The case $n = 0$ is trivial.

To prove the case for $n+1$, assume that $p^n[\top_p/x] \sqsubseteq_M^n \ \mu x.p$. To see that $p^{n+1}[\top_p/x] \sqsubseteq_M^{n+1} \ \mu x.p$ observe that if $p^{n+1}[\top_p/x] \xrightarrow{a} p'$ then since x is guarded in p we know from property 3.1 that $p \xrightarrow{a} p''$ and $p' \equiv p''[p^n[\top_p/x]/x]$. By property 3.2 we know that $p[\mu x.p/x] \xrightarrow{a} p''[\mu x.p/x]$ so by $REC \rightarrow$ we know $\mu x.p \xrightarrow{a} p''[\mu x.p/x]$. This is the matching move since by induction $p^n[\top_p/x] \sqsubseteq_M^n \ \mu x.p$ and clearly $p'' \sqsubseteq_M^n \ p''$ so $p''[p^n[\top_p/x]/x] \sqsubseteq_M^n \ p''[\mu x.p/x]$, and also $a \sqsubseteq_A \ a$.
Also if $\mu x.p \xrightarrow{a} p'$, $a \in M$. Then this is due to $p[\mu x.p/x] \xrightarrow{a} p'$. Again since x is guarded in p we know from property 3.1 that $p \xrightarrow{a} p''$ and $p' \equiv p''[\mu x.p/x]$. From property 3.2 we know that $p^{n+1}[\top_p/x] = p[p^n[\top_p/x]/x] \xrightarrow{a} p''[p^n[\top_p/x]/x]$ which clearly is the matching move since $a \sqsubseteq_A \ a$ and $p'' \sqsubseteq_M^n \ p''$ and by the induction hypothesis $p^n[\top_p/x] \sqsubseteq_M^n \ \mu x.p$ which implies
$p''[p^n[\top_p/x]/x] \sqsubseteq_M^n \ p''[\mu x.p/x]$. □

Actually this would work as well with any other process than $\top_p$ in $p^n[\top_p/x]$, since $p^0[q/x] \sqsubseteq_M^0 \ \mu x.p$ holds trivially. But we prefer $\top_p$ since $(p^n[\top_p/x])_n$ then is a decreasing chain in P_r.

This shows that $p^n[\top_p/x]$ and $\mu x.p$ is indistinguishable upto the n'th move.

Our first attempt of characterizing $\mu x.p$ by its unfoldings is therefore the following inference rule:

Proposition 3.25 *The inference rule*

$$\frac{\forall n.q \sqsubseteq_M \ p^n[\top_p/x]}{q \sqsubseteq_M \ \mu x.p} \quad , x \text{ guarded in } p$$

is sound provided P is imagefinite.

Proof: If P is imagefinite $q \sqsubseteq_M \ p^n[\top_p/x]$ implies $\forall m.q \sqsubseteq_M^m \ p^n[\top_p/x]$ by proposition 3.22, i.e. $\forall n.\forall m.q \sqsubseteq_M^m \ p^n[\top_p/x]$ especially for $m = n$ i.e. $\forall n.q \sqsubseteq_M^n \ p^n[\top_p/x]$. By proposition 3.24 $\forall n.p^n[\top_p/x] \sqsubseteq_M^n \ \mu x.p$ so by transitivity of $\sqsubseteq_M^n : \forall n.q \sqsubseteq_M^n \ \mu x.p$. Since P is imagefinite this states that $q \sqsubseteq_M \ \mu x.p$ which proves the proposition. □

This rule states that $\mu x.p$ is the greatest lower bound of the chain $(p^n[\top_p/x])_n$. The above rule resembles the Approximation Induction Principle (AIP) from process algebra investigated in [Gla 85].

But we want to do better than the above infinitary inference rule. Since the above theoretical development only works for imagefinite process systems we may use this to restrict the number of unfoldings of $\mu x.p$ necessary to do the inference.

Definition 3.12
A set S is said to be $\longrightarrow$-closed iff $p \in S$ and $p \xrightarrow{a} p' \ \Rightarrow \ p' \in S$.

Proposition 3.26 *If $S \subseteq Pr$ is $\longrightarrow$-closed then for all binary relations R over Pr:*

$$M\mathcal{B}(R \cap S^2) \cap S^2 = M\mathcal{B}(R) \cap S^2$$

PROOF: The $\subseteq$-direction follows from the monotonicity of $M\mathcal{B}$. To see that $M\mathcal{B}(R) \cap S^2 \subseteq M\mathcal{B}(R \cap S^2) \cap S^2$, let $(p, q) \in M\mathcal{B}(R) \cap S^2$ and prove $(p, q) \in M\mathcal{B}(R \cap S^2)$. If $p \xrightarrow{a} p'$ then $q \xrightarrow{a} q'$ with $a \sqsubseteq_A \; b$ and $(p', q') \in R$. Since S is $\longrightarrow$-closed $(p', q') \in S^2$ so $(p', q') \in R \cap S^2$ and $(p, q) \in M\mathcal{B}(R \cap S^2)$. Also if $q \xrightarrow{a} q'$, $a \in M$ then similar arguments yield the desired result. □

Proposition 3.27 *If $S \subseteq Pr$ is $\longrightarrow$-closed and $\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^{n+1} \; \cap S^2$ then for all $m \geq n$:*

$$\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^m \; \cap S^2 = \sqsubseteq_M \; \cap S^2$$

PROOF: By induction on $m - n$ using the above proposition.

$m - n = 0$ Holds trivial. $\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M \; \cap S^2$ holds since

$$\sqsubseteq_M^n \; \cap S^2 \supseteq \sqsubseteq_M^{n+1} \; \cap S^2 \supseteq \ldots \supseteq \ldots \sqsubseteq_M \; \cap S^2$$

$m + n + 1$ Assume $\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^m \; \cap S^2$ then by proposition 3.26 and the assumption $\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^{n+1} \; \cap S^2$ we get:

$$\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^{n+1} \; \cap S^2 = M\mathcal{B}(\sqsubseteq_M^n \;) \cap S^2 =$$

$$M\mathcal{B}(\sqsubseteq_M^n \; \cap S^2) \cap S^2 = M\mathcal{B}(\sqsubseteq_M^m \; \cap S^2) \cap S^2 = \sqsubseteq_M^{m+1} \; \cap S^2$$

□

Proposition 3.28 *Let $S \subseteq_{fin} Pr$ be a $\longrightarrow$-closed set. Then for all $(p, q) \in S^2$:*

$$p \sqsubseteq_M \; q \;\; \textit{iff} \;\; p \sqsubseteq_M^N \; , \; N \geq \mid S \mid^2$$

PROOF: The $\Rightarrow$-direction is obvious. For the $\Leftarrow$-direction consider the decreasing chain:

$$\sqsubseteq_M^0 \; \cap S^2 \supseteq \sqsubseteq_M^1 \; \cap S^2 \supseteq \ldots \sqsubseteq_M^n \; \cap S^2 \supseteq \ldots \supseteq \sqsubseteq_M \; \cap S^2$$

Since S is finite $\sqsubseteq_M^i \; \cap S^2$ must be finite for all i and $\sqsubseteq_M \; \cap S^2$ must be finite. Since the chain is decreasing there must exist a smallest n such that $\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^{n+1} \; \cap S^2$. This number n must be smaller than or equal to $\mid S \mid^2$ since in a set of size $\mid S \mid$ there cannot be more than $\mid S \mid^2$ combinations, so when $n \geq \mid S \mid^2$:

$$\sqsubseteq_M^n \; \cap S^2 = \sqsubseteq_M^{n+1} \; \cap S^2 = \sqsubseteq_M^{|S|^2} \; \cap S^2 = \sqsubseteq_M \; \cap S^2$$

By proposition 3.27.

□

To get an inference rule we apply the above proposition to the set $S = Der(p) \cup Der(q)$ where

$$Der(p) = \{p' \mid \exists p_1 \ldots p_n . \exists a_1 \ldots a_{n+1} . p \xrightarrow{a_1} p_1 \xrightarrow{a_2} \ldots \xrightarrow{a_n} p_n \xrightarrow{a_{n+1}} p'\}$$

S has exactly the necessary properties i.e. S is finite and $\longrightarrow$-closed. To give an upper bound on $|Der(p) \cup Der(q)$ we use the following function [Lar 86a]: $ND : Pr \rightarrow N$:

Definition 3.13

$$\begin{aligned} ND(nil) &= 1 \\ ND(x) &= 1 \\ ND(a.p) &= 1 + ND(p) \\ ND(p+q) &= ND(p) + ND(q) \\ ND(\mu x.p) &= ND(p) \end{aligned}$$

The upper bound for $\mu x.p$ is justified by the $1-1$ correspondence between $\mu x.p$ and p.

We are now able to state the inference rule:

Proposition 3.29 *The inference rule:*

$$\frac{q \sqsubseteq_M \; p^N[\top_p/x]}{q \sqsubseteq_M \; \mu x.p} \quad , N \geq (ND(p) + ND(q))^2 \text{ , } x \textit{ guarded in } p$$

is sound if P is imagefinite.

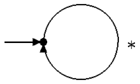
Proof: Assume $q \sqsubseteq_M \; p^N[\top_p/x]$. By proposition 3.24 $p^N[\top_p/x] \sqsubseteq_M^N \mu x.p$ for all N, especially for $N \geq (ND(p) + ND(q))^2$ and by proposition 3.28 $p^N[\top_p/x] \sqsubseteq_M \; \mu x.p$ which by transitivity of $\sqsubseteq_M$ implies $q \sqsubseteq_M \; \mu x.p$. □

The theoretical development above resembles the development of an alternative proof system for regular behaviours in [Lar 86a], although the development presented here is for processes whereas in [Lar 86a] it is for environments.

Chapter 4

Partial specifications

In this chapter we describe how the framework of chapter 1-3 may be used in hierarchical development of systems. We use this in an instantiation of the general theory called partial specification, since this allows one to partially specify subcomponents of systems and by knitting the partial specifications together obtaining a fully specified system. Specially we verify the concurrent alternating bit protocol CABP [Koy 85].

As mentioned in the introduction we want to be able to have an action $*$ which may take the place of any other action, a kind of action stronger than any other action. Also this action should be used in a process $\mathcal{U}$ with the characteristica $\mathcal{U}$: 

$\mathcal{U}$ is meant to take the place of in diagrams as:

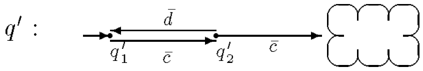
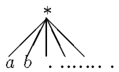


Figure 4.1: Partial specification of q.

yielding a process $q' = \mu x.\bar{c}.(\bar{c}.\mathcal{U} + \bar{d}.x)$ written in the language of chapter 2.

Formally $\mathcal{U}$ is defined as $\mathcal{U} = \mu x. * .x$ Since we want all other actions to be normal as in CCS [Mil 80], we take $Act = \{a, b, \ldots, \bar{a}, \bar{b}, \ldots, \tau\} \cup \{*\}$ with the ordering $a \sqsubseteq_A b$ iff $a = b$ or $b = *$. This ordering may be illustrated as: 

This ordering is extended to processes by $\sqsubseteq_M$ with $M = Act \setminus \{*\}$ since we want $*$-actions to be as strong as possible.

We need to extend the usual operations of CCS [Mil 80] to the new action set.

4.1 Instantiations of $|_g$ and $(\ldots)[f]$

We want to instantiate g in $|_g$ and f in $(\ldots)[f]$ such that we obtain the well-known operators for communication and restriction, with the extension to take $*$-actions into account in such a way that we cannot restrict $*$-actions away.

To obtain a restriction operator we instantiate f by an identity function $f : B \rightarrow B$ where B is a subset of Act, yielding the restriction operator $\lceil B$. To take $*$-actions into account we insist that B contains $*$.

For the communication operator we instantiate $|_g$ with the function g defined by the following table:

g	a	b	$\ldots$	$\bar{a}$	$\bar{b}$	$\ldots$	τ
a	u	u	$\ldots$	τ	u	$\ldots$	u
b	u	u	$\ldots$	u	τ	$\ldots$	u
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
$\bar{a}$	τ	u	$\ldots$	u	u	$\ldots$	u
$\bar{b}$	u	τ	$\ldots$	u	u	$\ldots$	u
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
τ	u	u	$\ldots$	u	u	$\ldots$	u

where u means undefined.

Table 4.1: $g : Act^2 \hookrightarrow Act$

As it may be seen from the table above $|_g$ really extends $|$ [Mil 80] to take $*$-actions into account.

As it may be seen from both $\lceil B$ and $|_g$ the functions have the property of being strict with respect to $*$-actions i.e. $f(*) = *$. In fact this is the requirement for f and g to be monotone partial functions fulfilling the definition of chapter 2. Also this constraint is enough to ensure that $CL(f^{-1}(M)\downarrow i) \subseteq M$ and that $M \cup CL(g^{-1}(M)\downarrow i) \subseteq M$, ensuring that the instantiations may be used with the proof systems of chapter 3.

From now on we shall write $|_g$ infix as $|$ and $\lceil B$ postfix.

4.2 General properties of partial specification

Before turning to our example we shall present a series of properties which may be used in partial specification of processes.

In the following $\sim$ is the bisimulation equivalence defined in chapter 1.

Proposition 4.1 *if $p \sim q$ then $p \sqsubseteq_M \ q$ and $q \sqsubseteq_M \ p$*

PROOF: From the definition of $\mathcal{B}$ and $M\mathcal{B}$ it is easy to see that $\mathcal{B}(R) \subseteq M\mathcal{B}(R)$ for all $R \subseteq Pr^2$. Since $\mathcal{B}(R)$ obviously is symmetric, also $R^{-1} \subseteq \mathcal{B}(R^{-1}) \subseteq M\mathcal{B}(R^{-1})$, which yields the second result. □

Definition 4.1 *p is robust iff $p \xrightarrow{*} p'$ is impossible for any p'.*

Definition 4.2 *p is concrete iff all derivations, including p, are robust.*

Note how these definitions resemble the definitions of rigid and stable in [Mil 80], where the ability of performing τ-actions is investigated.

Proposition 4.2 *if q is concrete and $p \sqsubseteq_M \ q$ then p is concrete*

PROOF: Since $p \sqsubseteq_M \ q$ there exists an M-bisimulation R including (p, q) such that $R \subseteq M\mathcal{B}(R)$. To see that $p \xrightarrow{*} p'$ is impossible observe that if $p \xrightarrow{*} p'$ then there exist q', b such that $* \sqsubseteq_A \ b$ and $(p', q') \in R$. But $* \sqsubseteq_A \ b$ iff $b = *$ and $q \overset{*}{\not\rightarrow}$ since q is robust so $p \overset{*}{\not\rightarrow}$. Also all derivatives of p are robust since all derivatives of q are especially those $q' \in (p', q') \in R$. □

This proposition yields an excellent test of concreteness. We just have to find a concrete process q and show $p \sqsubseteq_M \ q$, then we know that p is concrete. In chapter 5 we shall see another method of concreteness analysis, not involving another process.

Proposition 4.3 *if q is concrete and $p \sqsubseteq_M \ q$ then $p \sim q$*

PROOF: By proposition 4.2 p is concrete if q is concrete and $p \sqsubseteq_M \ q$. If $p \sqsubseteq_M \ q$ there exists a relation R such that $R \subseteq M\mathcal{B}(R)$ containing (p, q). Now let $SR = (Der(p) \times Der(q)) \cap R$, where

$$Der(p) = \{p' \mid \exists p_1 \ldots p_n \exists a_1 \ldots a_{n+1} . p \xrightarrow{a_1} p_1 \longrightarrow \ldots \xrightarrow{a_n} p_n \xrightarrow{n+1} p'\}$$

SR is the smallest M-bisimulation containing and concerning (p, q) and their derivatives. Then SR is also an M-bisimulation. To see this let $(p'', q'') \in SR$. To be a member of $M\mathcal{B}(SR)$ it is necessary that

1. $\forall a \in Act . p'' \xrightarrow{a} p' \ \Rightarrow \ \exists q' \exists b . q'' \xrightarrow{b} q' \ \& \ a \sqsubseteq_A \ b \ \& \ (p', q') \in SR$

2. $\forall a \in M \subseteq Act . q'' \xrightarrow{a} q' \ \Rightarrow \ \exists p' \exists b . p'' \xrightarrow{b} p' \ \& \ b \sqsubseteq_A \ a \ \& \ (p', q') \in SR$

Since both p and q are concrete and SR only concerns p and q and their derivatives, which we assume to be concrete, $a \sqsubseteq_A \ b$ specializes to $a = b$. And clearly $(p'', q'') \in \mathcal{B}(SR)$ i.e.

1. $\forall a \in Act . p'' \xrightarrow{a} p' \ \Rightarrow \ \exists q' . q'' \xrightarrow{a} q' \ \& \ (p', q') \in SR$

2. $\forall a \in Act.q'' \xrightarrow{a} q' \;\Rightarrow\; \exists p'.p'' \xrightarrow{a} p' \;\&\; (p',q') \in SR$

Taking $a \in Act$ in clause 2 does not violate $a \in M$ since all actions of q'' and its derivatives are different from $*$ i.e. included in M. □

This shows how this instance of $\sqsubseteq_M$ extends $\sim$.

4.3 Verifying the CABP

In this part we sketch how to verify that the CABP meets its specification, i.e. is equivalent to $spec = \mu x.a.b.\tau.\tau.x$

In [Koy 85] it was proven, using process algebra, that the system could be partioned into two processes p and q behaving like:

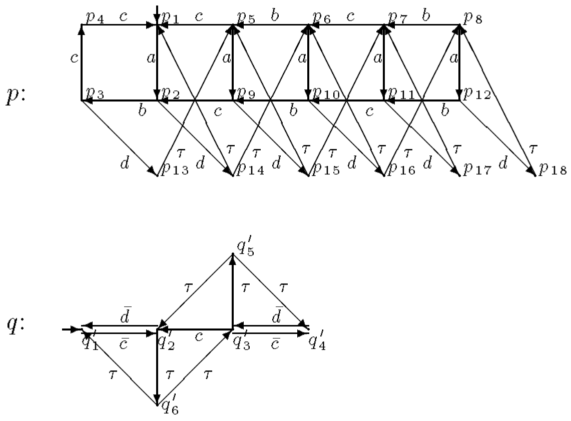


Figure 4.2:

The processes p and q may be defined in the language of chapter 2 with the instantiations of f defined in part 4.1. The partial specifications p' and q' of p and q may be illustrated as in figure 4.3.

These processes may be defined as:

$$p' = \mu x.a.(b.(c.d.x + d.\mathcal{U}) + d.\mathcal{U}) where\, \mathcal{U} = \mu x. * .x$$

$$q' = \mu x.\bar{c}.(\bar{d}.x + \bar{c}.\mathcal{U})$$

We could prove that $p \sqsubseteq_M p'$ and $q \sqsubseteq_M q'$ by using the proof systems of chapter 3. But since the relations:

$$R_1 = \{(p_i, p_i') \mid i \in [1,4]\} \cup \{p_i, p_5'\} \cup \{p_i, p_6'\}$$

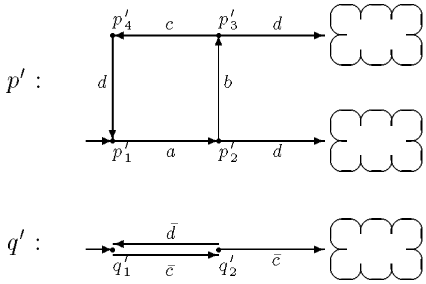


Figure 4.3:

and

$$R_2 = \{(q_i, q'_i) \mid i \in [1,2]\} \cup \{q_i, q'_3\}$$

are easy seen to be M-bisimulations containing (p, p') and (q, q'), we may draw the conclusion directly.

By C3 and C4 of table 3.2 we then know that

$$(p \mid q)\lceil B \sqsubseteq_M \; (p' \mid q')\lceil B, \; where \; B = \{a, b, \tau, *\}$$

We now prove that $(p' \mid q')\lceil B \sqsubseteq_M \; spec$ by use of the proof systems of chapter 3.

$(p' \mid q')\lceil B$

$=_M$ $((\mu x.a.(b.(c.d.x + d.\mathcal{U}) + d.\mathcal{U} \mid \mu y.\bar{c}.(\bar{d}.y + \bar{c}.\mathcal{U}))\lceil B$

by definition of p' and q'.

$=_M$ $((a.b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U}))\lceil B$

by R2 of table 3.3 and definition of substitution.

$=_M$ $(a.((b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U})) +$
$\bar{c}.((a.(b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U})) \mid (\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B$

by COM of table 3.2.

$=_M$ $(a.((b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B +$
$(\bar{c}.((a.(b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U})) \mid (\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B$

by F3 of table 3.2.

$=_M$ $(a.((b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B + nil$

by F2 of table 3.2.

$$=_M \quad a.(((b.(c.d.p' + d.\mathcal{U}) + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B)$$

by S4 and F2 of table 3.2.

$$\begin{aligned}=_M \quad & a.((b.((c.d.p' + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U}))) + d.(\mathcal{U} \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U}))) + \\ & \bar{c}.((b.(c.d.p' + d.\mathcal{U})) \mid (\bar{d}.q' + \bar{c}.\mathcal{U})) + \bar{c}(d.\mathcal{U} \mid (\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B)\end{aligned}$$

by COM of table 3.2.

$$=_M \quad a.b.(((c.d.p' + d.\mathcal{U}) \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U})))\lceil B)$$

by F3 to move $\lceil B$ over +, F2 four times and S4 three times
to eliminate *nil*'s.

$$\begin{aligned}=_M \quad & a.b.((\tau.(d.p' \mid (\bar{d}.q' + \bar{c}.\mathcal{U})) + c.(d.p' \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U}))) + \\ & d.(\mathcal{U} \mid (\bar{c}.(\bar{d}.q' + \bar{c}.\mathcal{U}))))\lceil B)\end{aligned}$$

by COM of table 3.2.

$$=_M \quad a.b.\tau.((d.p' \mid (\bar{d}.q' + \bar{c}.\mathcal{U}))\lceil B)$$

by F3, F2 and S4.

$$=_M \quad a.b.\tau.\tau.(p' \mid q')\lceil B$$

by arguments similar to the last two operations.

By R4 of R5 of table 3.2 we know that $(p' \mid q')\lceil B \sqsubseteq_M$ *spec*. By P2 we then know that $(p \mid q)\lceil B \sqsubseteq_M$ *spec*. Both $(p \mid q)\lceil B$ and *spec* are concrete, so by proposition 4.3 we know that $(p \mid q)\lceil B \sim$ *spec*, which is the desired result.

As it may be seen from the above proofs the method of partial specification yields an elegant proof technique when modularizing development and verification of implementations meeting their specifications.

Chapter 5

Data flow analysis

5.1 Introduction

In this chapter we describe how structure upon the set of actions may be used to impose a data flow analysis view and how this may be used to find properties of processes. Since data flow analysis for concurrent or nondeterministic programs, to our knowledge, has not yet been investigated, this may be seen as a first attempt to develop such a theory and not as a final answer to how data flow analysis for concurrent programming has to be done. In the development we try to extend the known methods from sequential programming to concurrent programming.

In data flow analysis we are interested in detecting dynamic aspects of programs by static analysis. The static analysis normally takes place at compile time such that these analyses may be used to justify program transformations, i.e. substitutions of subparts of a system with other subparts which in the context of the system yield the same behaviour, justified by the analysis.

Normally the information collected is in terms of approximations of sets (of states or values) instead of the real sets, since the information otherwise might not be computable.

It is essential that the information is correct i.e. yields no information that does not hold.

In traditional data flow analysis (see e.g. [Aho 77]) one has put on a "practical" view, concerning how profitable the analyses are and how to implement the analyses by "fast" algorithms not concerning correctness, or the methods have been ad hoc. Abstract interpretation [1] is a framework for describing correct data flow analyses. In [Cou 77] P. Cousot and R. Cousot showed how one could ensure "correctness" between a data flow analysis and a nonstandard semantics called the static semantics, by means of a pair (α, γ) of adjoined functions.

[1] For a general introduction to abstract interpretation see for example [Nie 85].

The semantics model was given as flow charts of sequential programs and the static semantics was taken to concern sets of values instead of the real values, traversing all branches of the flow chart. The "correctness" in [Cou 77, Cou 79] is called safeness since it relates to one data flow analysis being safe with respect to another, i.e. not describing a smaller set than the precise set expressed by the static semantics. The data flow analysis was a semantics using approximations of sets i.e. values describing sets. In [Nie 82] F. Nielson showed how to extend the work of P. Cousot and R. Cousot to a denotational semantics model by defining a series of nonstandard semantics. In [Nie 82] the term collecting semantics, which we shall use, is used for the static semantics of [Cou 77, Cou 79]. Also the correctness condition is reformulated such that it states that the analysis must yield information consistent with the standard semantics. It was shown how one could induce a data flow analysis by means of the static (collecting) semantics, and in fact we can construct a correct data flow analysis if it is safe with respect to the collecting semantics, relying on the correctness between the collecting semantics and the standard semantics.

We shall use the term consistent for both safeness and correctness. This will be justified in the following.

We may apply the method of abstract interpretation to the language for defining processes with the intuition:

Agents offer experiments in some universe of actions, abstracted agents offer experiments in another universe of abstract actions, so that experiments on abstract agents give information on the actual agents. So applied to the language for defining processes the approximation is in terms of approximations of sets of actions.

In the last section of this chapter we will describe an analysis which may give answer to when processes are stable [Mil 80]. This yields information about processes being weak-congruent or observational congruent [Mil 80].

Also an analysis of concreteness will be described. This analysis has its use together with the methods presented in chapter 4.

5.2 The framework

To express consistency of a data flow analysis with respect to the standard semantics we follow the line of [Nie 82] by building a series of (non-standard) semantics consistent with one another in a way to be defined later.

Let $P_{SS} = (Pr_{SS}, Act, \longrightarrow_{SS})$ where Pr_{SS} is given by the following abstract syntax:

$$p \ ::= \ nil \ \Big| \ a.p_1 \ \Big| \ p_1 + p_2 \ \Big| \ (p_1 \ldots p_n)[f] \ \Big| \ p_1 \,|_g\, p_2 \ \Big| \ x \ \Big| \ \mu x.p$$

for all $a \in Act$, $f : Act^n \hookrightarrow Act$, $g : Act^2 \hookrightarrow Act$, and $\longrightarrow_{SS}$ is the smallest family $\{\xrightarrow{a}_{SS} \subseteq Pr_{SS}^2 \mid a \in Act\}$ of relations satisfying:

$$\begin{array}{lrl}
ACT{\rightarrow_{SS}}: & & a.p \xrightarrow{a}_{SS} p \\
 & & \\
SUM{\rightarrow_{SS}}: & \underline{\text{if}} & p_1 \xrightarrow{a}_{SS} p' \\
 & \underline{\text{or}} & p_2 \xrightarrow{a}_{SS} p' \\
 & \underline{\text{then}} & p_1 + p_2 \xrightarrow{a}_{SS} p' \\
 & & \\
FUN{\rightarrow_{SS}}: & \underline{\text{if}} & \forall i.p_i \xrightarrow{a_i}_{SS} p'_i \;\&\; c \simeq f(a_1 \ldots a_n) \\
 & \underline{\text{then}} & (p_1 \ldots p_n)[f] \xrightarrow{c}_{SS} (p'_1 \ldots p'_n)[f] \\
 & & \\
COM{\rightarrow_{SS}}: & \underline{\text{if}} & (p_1 \xrightarrow{c}_{SS} p'_1 \;\&\; p ='\, p'_1 \mid_g p_2) \\
 & \underline{\text{or}} & (p_2 \xrightarrow{c}_{SS} p'_2 \;\&\; p' = p_1 \mid_g p'_2) \\
 & \underline{\text{or}} & (p_1 \xrightarrow{a}_{SS} p'_1 \;\&\; p_2 \xrightarrow{b}_{SS} p'_2 \;\& \\
 & & p' = p'_1 \mid_g p'_2 \;\&\; c \simeq g(a,b)) \\
 & \underline{\text{then}} & p_1 \mid_g p_2 \xrightarrow{c}_{SS} p' \\
 & & \\
REC{\rightarrow_{SS}}: & \underline{\text{if}} & p[\mu x.p/x] \xrightarrow{a}_{SS} p' \\
 & \underline{\text{then}} & \mu x.p \xrightarrow{a}_{SS} p'
\end{array}$$

This process system will be called the standard semantics SS of the language for defining processes. Throughout this chapter $c \simeq f(a_1 \ldots a_n)$ and $c \simeq g(a,b)$ are defined as in chapter 2 and $p[\mu x.p/x]$ has its normal meaning.

It is not obvious how a static (collecting) semantics for nondeterministic programs should be, but the nondeterministic operator + behaves in a way like the if-then-else construct of sequential programming, we therefore pursue by treating + almost as we would treat the if-then-else construct. i.e. traverse over both branches if they lead to the same state (process), and collect both values.

Let $P_{CS} = (Pr_{CS}, \mathcal{P}(Act), \longrightarrow_{CS})$ where Pr_{CS} is given by the following abstract syntax:

$$p \;::=\; nil \;\Big|\; A.p_1 \;\Big|\; p_1 + p_2 \;\Big|\; (p_1 \ldots p_n)[f] \;\Big|\; p_1 \mid_g p_2 \;\Big|\; x \;\Big|\; \mu x.p$$

for all $A \in \mathcal{P}(Act)$, $f : \mathcal{P}(Act)^n \hookrightarrow \mathcal{P}(Act)$, $g : \mathcal{P}(Act)^2 \hookrightarrow \mathcal{P}(Act)$, and $\longrightarrow_{CS}$ is the smallest family $\{\xrightarrow{A}_{CS} \subseteq Pr^2_{CS} \mid A \in \mathcal{P}(Act)\}$ of relations satisfying:

$$\begin{array}{lrl}
ACT{\rightarrow_{CS}}: & & A.p \xrightarrow{A}_{CS} p
\end{array}$$

$$\begin{array}{lrl}
SUM{\rightarrow_{CS}}: & \underline{\text{if}} & p_1 \xrightarrow{A}_{CS} p' \\
 & \underline{\text{or}} & p_2 \xrightarrow{A}_{CS} p' \\
 & \underline{\text{or}} & (A = B \cup C \;\&\; p_1 \xrightarrow{B}_{CS} p' \;\&\; p_2 \xrightarrow{C}_{CS} p') \\
 & \underline{\text{then}} & p_1 + p_2 \xrightarrow{A}_{CS} p'
\end{array}$$

$$\begin{array}{lrl}
FUN{\rightarrow_{CS}}: & \underline{\text{if}} & \forall i.p_i \xrightarrow{A_i}_{CS} p'_i \;\&\; C \simeq f(A_1 \ldots A_n) \\
 & \underline{\text{then}} & (p_1 \ldots p_n)[f] \xrightarrow{C}_{CS} (p'_1 \ldots p'_n)[f]
\end{array}$$

$$\begin{array}{lrl}
COM{\rightarrow_{CS}}: & \underline{\text{if}} & (p_1 \xrightarrow{C}_{CS} p'_1 \;\&\; p =' p'_1 \mid_g p_2) \\
 & \underline{\text{or}} & (p_2 \xrightarrow{C}_{CS} p'_2 \;\&\; p' = p_1 \mid_g p'_2) \\
 & \underline{\text{or}} & (p_1 \xrightarrow{A}_{CS} p'_1 \;\&\; p_2 \xrightarrow{B}_{CS} p'_2 \;\& \\
 & & p' = p'_1 \mid_g p'_2 \;\&\; C \simeq g(A, B)) \\
 & \underline{\text{then}} & p_1 \mid_g p_2 \xrightarrow{C}_{CS} p'
\end{array}$$

$$\begin{array}{lrl}
REC{\rightarrow_{CS}}: & \underline{\text{if}} & p[\mu x.p/x] \xrightarrow{A}_{CS} p' \\
 & \underline{\text{then}} & \mu x.p \xrightarrow{A}_{CS} p'
\end{array}$$

This process system will be called the collecting semantics CS of the language for defining processes. We shall assume $\mathcal{P}(Act)$ ordered by subset inclusion $\subseteq$.

Assume *AppAct* is a complete lattice such that $\sqcup$ is defined. Let $P_{AS} = (Pr_{AS}, AppAct, \longrightarrow_{AS})$ where Pr_{AS} is given by the following abstract syntax:

$$p \;::=\; nil \;\Big|\; l.p_1 \;\Big|\; p_1 + p_2 \;\Big|\; (p_1 \ldots p_n)[f] \;\Big|\; p_1 \mid_g p_2 \;\Big|\; x \;\Big|\; \mu x.p$$

for all $l \in AppAct$, $f : AppAct^n \hookrightarrow AppAct$, $g : AppAct^2 \hookrightarrow AppAct$, and $\longrightarrow_{AS}$ is the smallest family $\{\xrightarrow{l}_{AS} \subseteq Pr^2_{AS} \mid l \in AppAct\}$ of relations satisfying:

$$\begin{array}{lrl}
ACT{\rightarrow}_{AS}: & & l.p \xrightarrow{l}_{AS} p
\end{array}$$

$$\begin{array}{lrl}
SUM{\rightarrow}_{AS}: & \underline{\text{if}} & p_1 \xrightarrow{l}_{AS} p' \\
 & \underline{\text{or}} & p_2 \xrightarrow{l}_{AS} p' \\
 & \underline{\text{or}} & (l = m \sqcup n \;\&\; p_1 \xrightarrow{m}_{CS} p' \;\&\; p_2 \xrightarrow{n}_{CS} p') \\
 & \underline{\text{then}} & p_1 + p_2 \xrightarrow{l}_{AS} p'
\end{array}$$

$$\begin{array}{lrl}
FUN{\rightarrow}_{AS}: & \underline{\text{if}} & \forall i.p_i \xrightarrow{l_i}_{AS} p'_i \;\&\; m \simeq f(l_1 \ldots l_n) \\
 & \underline{\text{then}} & (p_1 \ldots p_n)[f] \xrightarrow{m}_{AS} (p'_1 \ldots p'_n)[f]
\end{array}$$

$$\begin{array}{lrl}
COM{\rightarrow}_{AS}: & \underline{\text{if}} & (p_1 \xrightarrow{m}_{AS} p'_1 \;\&\; p =' p'_1 \,|_g\, p_2) \\
 & \underline{\text{or}} & (p_2 \xrightarrow{m}_{AS} p'_2 \;\&\; p' = p_1 \,|_g\, p'_2) \\
 & \underline{\text{or}} & (p_1 \xrightarrow{n}_{AS} p'_1 \;\&\; p_2 \xrightarrow{l}_{AS} p'_2 \;\&\; \\
 & & p' = p'_1 \,|_g\, p'_2 \;\&\; m \simeq g(n, l)) \\
 & \underline{\text{then}} & p_1 \,|_g\, p_2 \xrightarrow{m}_{AS} p'
\end{array}$$

$$\begin{array}{lrl}
REC{\rightarrow}_{AS}: & \underline{\text{if}} & p[\mu x.p/x] \xrightarrow{l}_{AS} p' \\
 & \underline{\text{then}} & \mu x.p \xrightarrow{l}_{AS} p'
\end{array}$$

This process system will be called the approximating semantics AS of the language for defining processes.

Note how CS is just an instance of AS with $AppAct = \mathcal{P}(Act)$ and $\sqcup = \cup$, but since CS is essential we want to single it out.

5.3 Correctness and safeness on actions

To express correctness of an analysis AS with respect to the standard semantics SS we may use a function $\beta : Act \rightarrow AppAct$. The intuition is that if $\beta(a) = Appa$ then $Appa$ describes properties of a.

In traditional data flow analysis one analysis, the collecting semantics CS, is in a way the most precise data flow analysis, since it gives a trace of all possible values (actions).

Normally CS takes its values in the powerset of standard values. Applied to our model of concurrency it takes its actions in $\mathcal{P}(Act)$. The correctness of CS with respect to SS is expressed by $\sigma : Act \rightarrow \mathcal{P}(Act)$ where $\sigma(a) = \{a\}$.

To express safeness we may use the ideas of abstract interpretation [Nie 85] [Cou 77] [Cou 79]. Safeness is to say that one data flow analysis is more approximative than another, i.e. it is not as precise, but it will not give contradicting information.

Safeness is usually expressed relative to the collecting semantics by a function $\alpha : \mathcal{P}(Act) \rightarrow AppAct$.

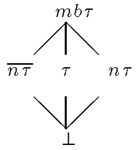
If an approximating analysis AS is safe with respect to CS we may rely on the correctness between CS and SS to ensure correctness of AS with respect to SS. This is in a way closest in spirit to the original framework of P. Cousot and R. Cousot [Cou 77] [Cou 79].

These relationships may be illustrated as:

$$\begin{tikzcd}
\mathcal{P}(Act) \arrow[rr, dash, "\alpha"] & & AppAct \\
& Act \arrow[ul, dash, "\sigma"] \arrow[ur, dash, "\beta"'] &
\end{tikzcd}$$

Figure 5.1: (in spirit Figure 2 of [Nie 86]).

For a more concrete example suppose $Act = A \cup \overline{A} \cup \{\tau\}$, where $A = \{a, b, c, ...\}$ and $\overline{A}$ is a set of conames. Assume we only have one unary function f to instantiate $FUN \!\rightarrow_{SS}$ with. We have to find functions $g : \mathcal{P}(Act) \hookrightarrow \mathcal{P}(Act)$ and $h : L \hookrightarrow L$, such that they simulate the behaviour of $f : Act \hookrightarrow Act$. Concerning the collecting semantics we assume $\mathcal{P}(Act)$ ordered by subset inclusion $\subseteq$, and the function as some monotone function $g : \mathcal{P}(Act) \hookrightarrow \mathcal{P}(Act)$

Finally we must consider the approximating semantics AS where we assume that *AppAct* is some complete lattice L and the function is some monotone function $h : L \hookrightarrow L$. A possible choice of L is: 

i.e. the complete lattice with elements $\perp$, $\overline{n\tau}$, τ, $n\tau$ and $mb\tau$ ordered by $\perp \sqsubseteq l \sqsubseteq mb\tau$ where l is $\overline{n\tau}$, τ or $n\tau$. $\perp$ seems artificial in the above example but it is kept for historical reasons. Note, however, that the intuition "more approximative" in the ordering used in data flow analysis differs from the standard intuition: "more defined" and that the ordering may seem back to front by the first glance.

With this assumption figure 5.1 specializes to figure 5.2 and we shall now explain α, β, γ and σ.

Readers familiar with abstract interpretation will notice the analogy with

$$\begin{tikzcd}
\mathcal{P}(Act) \arrow[loop left, "g"] \arrow[rr, bend right, "\alpha"'] & & AppAct \arrow[ll, bend right, "\gamma"'] \arrow[loop right, "h"] \\
& Act \arrow[ul, "\sigma"] \arrow[ur, "\beta"'] \arrow[loop left, "f"] &
\end{tikzcd}$$

Figure 5.2:

the rule of sign test (see e.g. [Cou 77]).

The functions β and σ are termed representation functions since the intuition is that $\beta(a) \in AppAct$ and $\sigma(a) \in \mathcal{P}(Act)$ are properties that best describe $a \in Act$. Thus it is natural to put

$$\begin{aligned}
\beta(a) &= n\tau \text{ if } a \in A \\
\beta(\tau) &= \tau \\
\beta(\overline{a}) &= \overline{n\tau} \text{ if } \overline{a} \in \overline{A}
\end{aligned}$$

and $\sigma(a) = \{a\}$

The correctness of h with respect to f may then be expressed by the condition that:

$$\beta(a) \sqsubseteq l \ \Rightarrow\ \beta(f(a)) \sqsubseteq h(l)$$

which says that:

whenever l correctly describes a, also $h(l)$ correctly describes $f(a)$

Similar σ may be used to express correctness of g with respect to f.

The relationship between $\mathcal{P}(Act)$ and L is expressed using the framework of abstract interpretation [Cou 79]. The fundamental ingredient is a pair (α, γ) of abstraction and concretization functions.

The intuition with γ is to formalize the intuitive meaning of properties in $AppAct$, so one has $\gamma(n\tau) = A$, $\gamma(\bot) = \emptyset$, $\gamma(mb\tau) = Act$ etc.

Given a set of actions B the intention with α is that $\alpha(B)$ is a best safe description of B in *AppAct*, so one would express e.g.

$$\alpha(\{\overline{a},\overline{b}\}) = \overline{n\tau} \text{ and } \alpha(\{a,\overline{b}\}) = mb\tau$$

This indicates that "B is safely described by l" means

$$B \subseteq \gamma(l)$$

So $\{\overline{a},\overline{b}\}$ is safely described by $\overline{n\tau}$ and $mb\tau$, but $\gamma(\overline{n\tau})$ is a proper subset of $\gamma(mb\tau)$, and $\overline{n\tau}$ is therefore a better property.

It is most convenient if we can use $\overline{n\tau} \sqsubseteq mb\tau$ to deduce that we should prefer $\overline{n\tau}$, so we have to relate the partial orders $\sqsubseteq$ and $\subseteq$.

This is captured by the adjoinedness condition [Cou 79] :

1. α and γ are monotone
2. $\forall B : B \subseteq (\gamma \circ \alpha)(B)$
3. $\forall l : (\alpha \circ \gamma)(l) \sqsubseteq l$

The relationship between g and h may be illustrated as:

$$\begin{tikzcd}
Appact \arrow[r, "h"] \arrow[dd, "\gamma"'] & Appact \\
 & \rotatebox{90}{$\sqsubseteq$} \\
\mathcal{P}(App) \arrow[r, "g"] & \mathcal{P}(App) \arrow[u, "\alpha"']
\end{tikzcd}$$

Figure 5.3:

Generally we may use two complete lattices L and M, and the adjoinedness condition of $\alpha : L \to M$ and $\gamma : M \to L$ may be expressed as:

$$\forall l \in L : \forall m \in M.\alpha(l) \sqsubseteq m \Leftrightarrow l \subseteq \gamma(m)$$

5.4 Extending correctness and safeness to process systems

We now turn to extend the correctness and safeness conditions to process systems so that we are able to make statements like AS is safe and correct with respect to SS and CS.

The extension of the correctness condition may be given by a bisimulation-like construct. Note how these constructs extend definition 2.4-6 in [Lar 86a].

Let $R \subseteq Pr_{SS} \times Pr_{AS}$ be a relation such that whenever $(p, q) \in R$ then:

1. $\forall a \in Act . p \xrightarrow{a}_{SS} p' \Rightarrow \exists q' \exists l \in AppAct . q \xrightarrow{l}_{AS} q' \ \& \ \beta(a) \sqsubseteq l \, \& \ (p', q') \in R$
2. $\forall l \in AppAct . q \xrightarrow{l}_{AS} q' \Rightarrow \exists p' \exists a \in Act . p \xrightarrow{a}_{SS} p' \ \& \ \beta(a) \sqsubseteq l \, \& \ (p', q') \in R$

We say that R is a correctness relation, and we write $p \ \underline{corr}_\beta \ q$ if there exists a correctness relation containing (p, q).

Now for $R \subseteq Pr_{SS} \times Pr_{AS}$ we define $CORR_\beta(R) \subseteq Pr_{SS} \times Pr_{AS}$ as:

$(p, q) \in CORR_\beta(R)$ iff:

1. $\forall a \in Act . p \xrightarrow{a}_{SS} p' \Rightarrow \exists q' \exists l \in AppAct . q \xrightarrow{l}_{AS} q' \ \& \ \beta(a) \sqsubseteq l \, \& \ (p', q') \in R$
2. $\forall l \in AppAct . q \xrightarrow{l}_{AS} q' \Rightarrow \exists p' \exists a \in Act . p \xrightarrow{a}_{SS} p' \ \& \ \beta(a) \sqsubseteq l \, \& \ (p', q') \in R$

$CORR_\beta$ is easy seen to be a monotone endofunction upon the complete lattice of relations over $Pr_{SS} \times Pr_{AS}$ ordered by subset inclusion. Thus by classic lattice theory $CORR_\beta$ has a maximal fixed point. This fixed point equals $\underline{corr}_\beta$. We may straighten the condition $\beta(a) \sqsubseteq l$ to $\beta(a) = l$ obtaining a restricted relation, but $\beta(a) \sqsubseteq l$ seems more in the spirit of abstract interpretation.

The correctness condition of CS with respect to SS may be expressed by σ, but we prefer to use $a \in A$ instead of $\{a\} \subseteq A$ even though they are equivalent.

Let $R \subseteq Pr_{SS} \times Pr_{CS}$ be a relation such that whenever $(p, q) \in R$ then:

1. $\forall a \in Act . p \xrightarrow{a}_{SS} p' \Rightarrow \exists q' . \exists A \in \mathcal{P}(Act) . q \xrightarrow{A}_{CS} q' \ \& \ a \in A \ \& \ (p', q') \in R$
2. $\forall A \in \mathcal{P}(Act) . q \xrightarrow{A}_{CS} q' \Rightarrow \exists p' . \exists a \in Act . p \xrightarrow{a}_{SS} p' \ \& \ a \in A \ \& \ (p', q') \in R$

Now for $R \subseteq Pr_{SS} \times Pr_{CS}$ we define $CORR_\sigma(R) \subseteq Pr_{SS} \times Pr_{CS}$ as:

$(p, q) \in CORR_\sigma(R)$ iff:

1. $\forall a \in Act . p \xrightarrow{a}_{SS} p' \Rightarrow \exists q' . \exists A \in \mathcal{P}(Act) . q \xrightarrow{A}_{CS} q' \ \& \ a \in A \ \& \ (p', q') \in R$
2. $\forall A \in \mathcal{P}(Act) . q \xrightarrow{A}_{CS} q' \Rightarrow \exists p' . \exists a \in Act . p \xrightarrow{a}_{SS} p' \ \& \ a \in A \ \& \ (p', q') \in R$

The maximal fixed point of this monotone endofunction is $\underline{corr}_{\sigma}$.

Also the safeness condition between $\mathcal{P}(Act)$ and $AppAct$ may be extended to process systems.

Let $R \subseteq Pr_{CS} \times Pr_{AS}$ be a relation such that whenever $(p, q) \in R$ then:

1. $\forall A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \Rightarrow \exists q' \exists l \in AppAct.q \xrightarrow{l}_{AS} q' \,\&\, \alpha(A) \sqsubseteq l \,\&\, (p', q') \in R$
2. $\forall l \in AppAct.q \xrightarrow{l}_{AS} q' \Rightarrow \exists p' \exists A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \,\&\, A \subseteq \gamma(l) \,\&\, (p', q') \in R$

We say that R is a safeness relation, and we write $p\ \underline{safe}^{\sigma}_{\alpha}\ q$ if there exists a safeness relation R containing (p, q).

Now for $R \subseteq Pr_{CS} \times Pr_{AS}$ we define $SAFE^{\sigma}_{\alpha}(R) \subseteq Pr_{CS} \times Pr_{AS}$ as:

$(p, q) \in SAFE^{\sigma}_{\alpha}(R)$ iff:

1. $\forall A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \Rightarrow \exists q' \exists l \in AppAct.q \xrightarrow{l}_{AS} q' \,\&\, \alpha(A) \sqsubseteq l \,\&\, (p', q') \in R$
2. $\forall l \in AppAct.q \xrightarrow{l}_{AS} q' \Rightarrow \exists p' \exists A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \,\&\, A \subseteq \gamma(l) \,\&\, (p', q') \in R$

The maximal fixed point of this monotone endofunction is $\underline{safe}^{\sigma}_{\alpha}$.

To take expressions with free variables into account we may use the technique of refining the functionals $CORR_{\sigma}$, $CORR_{\beta}$ and $SAFE^{\sigma}_{\alpha}$ in the same way as we did for M-bisimulation in chapter 3. We may refine $SAFE^{\sigma}_{\alpha}$, $CORR_{\beta}$ and $CORR_{\sigma}$ by adding the clause $UG(p) = UG(q)$, where $UG(p)$ is the set of unguarded variables in p.

$(p, q) \in SAFE^{\sigma}_{\alpha}(R)$ iff:

1. $UG(p) = UG(q)$
2. $\forall A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \Rightarrow \exists q' \exists l \in AppAct.q \xrightarrow{l}_{AS} q' \,\&\, \alpha(A) \sqsubseteq l \,\&\, (p', q') \in R$
3. $\forall l \in AppAct.q \xrightarrow{l}_{AS} q' \Rightarrow \exists p' \exists A \in \mathcal{P}(Act).p \xrightarrow{A}_{CS} p' \,\&\, A \subseteq \gamma(l) \,\&\, (p', q') \in R$

Unfortunately this only works for regular expressions, as the definition of unguardedness given in chapter 3 (definition 3.4)is only given for regular expressions.

The naive extension of UG by $UG((p_1 \dots p_n)[f]) = \bigcup_{i=1}^{n} UG(p_i)$ does not work, since the substitution property will <u>not</u> in general hold, i.e it is not guaranteed that $p_1[q_1/x]\ \underline{safe}^{\sigma}_{\alpha}\ p_2[q_2/x]$ holds whenever $p_1\ \underline{safe}^{\sigma}_{\alpha}\ p_2$ and $q_1\ \underline{safe}^{\sigma}_{\alpha}\ q_2$

To see this let:

$$\begin{aligned} p_1 &= x[a \to \bar{a}] \\ q_1 &= x[id] \\ p_2 &= a.nil \\ q_2 &= n\tau.nil \end{aligned}$$

where $[a \rightarrow \bar{a}]$ means the renaming function which sends every name into its coname. Then it is obvious that $p_1 \sqsubseteq_M q_1$ and $p_2 \sqsubseteq_M q_2$, but

$$a.nil[a \rightarrow \bar{a}] \ \underline{safe}^{\sigma}_{\alpha} \ n\tau.nil[id]$$

does not hold.

In fact this problem is a general problem of substitution in non-regular contexts. The solution seems to be to extend the definition of unguardedness to take account of the presence of $[f]$ in expressions (see e.g. the remark in [Lar 86a] p. 166). We will not pursue this any further in this thesis but only turn our attention to open expressions if they are regular.

5.5 Inducing data flow analysis

If we have a function $f : Act^n \hookrightarrow Act$ which we may use to instantiate $FUN{\rightarrow}_{SS}$, then this function may be extended to $\bar{f} : \mathcal{P}(Act)^n \hookrightarrow \mathcal{P}(Act)$ in the obvious way $\bar{f}(A_1 \ldots A_n) = \{f(a_1 \ldots a_n) \mid a_i \in A_i\}$, and similar for g in $COM{\rightarrow}_{SS}$.

Also if we have defined a representation function $\beta : Act \rightarrow AppAct$ we may extend this to an abstraction function
$\alpha : \mathcal{P}(Act) \rightarrow AppAct$ by $\alpha(A) = \sqcup\{\beta(a) \mid a \in A\}$.

α is called a lower adjoined if there exists a function γ such that (α, γ) is an adjoined pair. There need not exist such a γ but if it does it is uniquely determined by the formula [Cou 79]:

$$\gamma(m) = \sqcup\{l \mid \alpha(l) \sqsubseteq m\}$$

Finally, γ is called an upper adjoined if there exists an α such that (α, γ) is a pair of adjoined functions and then α is uniquely determined by

$$\alpha(l) = \sqcap\{m \mid l \sqsubseteq \gamma(m)\}$$

Also β may be given in terms of α by $\beta = \alpha \circ \sigma$ assuming that $\sigma(a)$ is $\{a\}$. Composition of partial functions is taken to be:

$$(f \circ g)(x) = \begin{cases} f(g(x)) \text{ if } g(x) \text{ is defined} \\ \text{undefined otherwise} \end{cases}$$

The above constructs may be used in what may be termed an induced data flow analysis. The induction will be given as an instance of the general scheme of definition 2.2.

First we may relate SS to CS by an instance of the general scheme of definition 2.2. by giving a function $\mathcal{F}_c : Pr_{SS} \rightarrow Pr_{CS}$, given inductively by:

Definition 5.1

$$\mathcal{F}_c(nil) \quad = \quad nil$$

$$\begin{array}{rcl}
\mathcal{F}_c(a.p) & = & \sigma(a).\mathcal{F}_c(p) \;=\; \{a\}.\mathcal{F}_c(p) \\
\mathcal{F}_c(p_1 + p_2) & = & \mathcal{F}_c(p_1) + \mathcal{F}_c(p_2) \\
\mathcal{F}_c(p_1 \mid_g p_2) & = & \mathcal{F}_c(p_1) \mid_{\bar{g}} \mathcal{F}_c(p_2) \\
\mathcal{F}_c((p_1 \ldots p_n)[f]) & = & (\mathcal{F}_c(p_1) \ldots \mathcal{F}_c(p_n))[\bar{f}] \\
\mathcal{F}_c(x) & = & x \\
\mathcal{F}_c(\mu x.p) & = & \mu x.\mathcal{F}_c(p)
\end{array}$$

To take account for substitution we may use the following lemma:

Lemma 5.1

$$\mathcal{F}_c(p[q/x]) = \mathcal{F}_c(p)[\mathcal{F}_c(q)/x]$$

PROOF: By structural induction on p using the definition of substitution (definition 3.3). □

Lemma 5.2 *if* $\mathcal{F}_c(p) \xrightarrow{A}_{CS} p''$ *then* $p'' = \mathcal{F}_c(p')$ *and* $p \xrightarrow{a}_{SS} p'$ *with* $a \in A$

PROOF: By induction on the number of inferences.
(The full proof is presented in appendix B.) □

Proposition 5.1 p $\underline{corr}_\sigma$ $\mathcal{F}_c(p)$, *p closed and finite.*

PROOF: The relation

$$R = \{(p, \mathcal{F}_c(p)) \mid p \in Pr^f_{SS}\}$$

is a correctness relation.
To see that $R \subseteq CORR_\sigma(R)$ we proceed by structural induction using lemma 5.2 in the cases: $p = (p_1 \ldots p_n)[f]$ and $p = p_1 \mid_g p_2$.
(The full proof is presented in appendix B.) □

Proposition 5.2 p $\underline{corr}_\sigma$ $\mathcal{F}_c(p)$, *p regular.*

PROOF: The relation

$$R = \{(p, \mathcal{F}_c(p)) \mid p \in Pr^r_{SS}\}$$

is a correctness relation.
To see that $R \subseteq CORR_\sigma(R)$ we proceed by induction on inferences using lemma 5.1 when p has the form $\mu x.p$.
(The full proof is presented in appendix B.) □

The relationship between CS and AS may be given by a function $\mathcal{F}_a : Pr_{CS} \rightarrow Pr_{AS}$ defined inductively by:

Definition 5.2

$$
\begin{array}{rcl}
\mathcal{F}_a(nil) & = & nil \\
\mathcal{F}_a(A.p) & = & \alpha(A).\mathcal{F}_a(p) \\
\mathcal{F}_a(p_1 + p_2) & = & \mathcal{F}_a(p_1) + \mathcal{F}_a(p_2) \\
\mathcal{F}_a(p_1 \mid_h p_2) & = & \mathcal{F}_a(p_1) \mid_{\alpha \circ h \circ \gamma_2} \mathcal{F}_a(p_2) \\
\mathcal{F}_a((p_1 \ldots p_n)[g]) & = & (\mathcal{F}_a(p_1 \ldots \mathcal{F}_a(p_n))[\alpha \circ g \circ \gamma_n] \\
\mathcal{F}_a(x) & = & x \\
\mathcal{F}_a(\mu x.p) & = & \mu x.\mathcal{F}_a(p)
\end{array}
$$

Here γ_n means $\lambda v_1 \ldots v_n.(\gamma(v_1) \ldots \gamma(v_n))$.

Actually this is a general scheme when relating two approximating semantics AS_1 and AS_2.

The definition of $\mathcal{F}_a((p_1 \ldots p_n)[g]) = (\mathcal{F}_a(p_1) \ldots \mathcal{F}_a(p_n))[\alpha \circ g \circ \gamma_n]$ will be the best choice of a function approximating g due to the adjoinedness condition upon (α, γ). In particular it will be better than

$$\mathcal{F}_a((p_1 \ldots p_n)[h]) = (\mathcal{F}_a(p_1) \ldots \mathcal{F}_a(p_n))[\lambda v_1 \ldots v_n.\top]$$

where $\top = \sqcup AppAct$, as this would yield a rather uninformative analysis.

Lemma 5.3

$$\mathcal{F}_a(p[q/x]) = \mathcal{F}_a(p)[\mathcal{F}_a(q)/x]$$

Proof: By structural induction on p using definition 3.3 of substitution. □

Lemma 5.4 *if $\mathcal{F}_a(p) \xrightarrow{l}_{AS} p''$ then $p'' = \mathcal{F}_a(p')$ and $p \xrightarrow{A}_{CS} p'$ with $\alpha(A) \sqsubseteq l$*

Proof: By induction on the number of inferences. □

That AS is safe with respect to CS is expressed by

Proposition 5.3 *$p \; \underline{safe}^{\sigma}_{\alpha} \; \mathcal{F}_a(p)$, p closed and finite.*

Proof: The relation

$$R = \{(p, \mathcal{F}_a(p)) \mid p \in Pr^f_{CS}\}$$

is a safeness relation.
To see that $R \subseteq SAFE^{\sigma}_{\alpha}(R)$ we proceed by structural induction using lemma 5.4 in the cases: $p = (p_1 \ldots p_n)[f]$ and $p = p_1 \mid_g p_2$.
(The proof of the nontrivial cases is presented in appendix B.) □

Proposition 5.4 *$p \underline{safe^{\sigma}_{\alpha}} \mathcal{F}_a(p)$, p regular.*

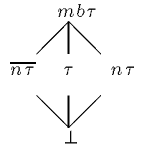
Proof: The relation

$$R = \{(p, \mathcal{F}_a(p)) \mid p \in Pr^{r}_{CS}\}$$

is a safeness relation.
To see that $R \subseteq SAFE^{\sigma}_{\alpha}(R)$ we use induction on the number of inferences using lemma 5.3 when p has the form $\mu x.p$. □

From the above we see that by giving either $\alpha : \mathcal{P}(Act) \to AppAct$ or $\beta : Act \to AppAct$ we may induce a consistent data flow analysis, in the first case relying on the correctness of CS and in the second by using
$\alpha(A) = \sqcup\{\beta(a) \mid a \in A\}$.

5.6 Stable analysis

As shown in part 5.3 we may describe properties of $Act = A \cup \bar{A} \cup \{\tau\}$ by the lattice L: 

and we may use the function $\beta(a) = n\tau$ if $a \in A$, $\beta(\bar{a}) = \overline{n\tau}$ if $\bar{a} \in \bar{A}$, $\beta(\tau) = \tau$ to induce a data flow analysis.

This analysis may be used to test whether a process p is stable or not. If only p yields $\overline{n\tau}$ or $n\tau$ experiments the process is stable, if the process yields τ experiments it is not stable, and if it yields an $mb\tau$ experiment we can not be certain whether it is stable or not, and a more precise analysis is necessary.

The analysis of stableness is useful for deciding congruence of processes:
If p and q are stable then if $p \approx q$ then $p \approx^c q$ where $\approx$ is observational equivalence and $\approx^c$ is congruence [Mil 80].

Let $g : Act^2 \hookrightarrow Act$ be the partial function satisfying the following diagram shown in table 5.1:

When we instantiate $|_g$ with g we obtain the wellknown operator $|$ for communication from CCS [Mil 80]. Also let $f_1 : Act \hookrightarrow Act$ be the function:

$$f_1(x) = \begin{cases} x \text{ if } x \neq c \text{ and } x \neq \bar{c} \\ \text{unknown otherwise} \end{cases}$$

When we instantiate $(\ldots)[f]$ with f_1 we obtain the restriction operator $\backslash c$ from CCS.

g	a	b	$\ldots$	$\bar{a}$	$\bar{b}$	$\ldots$	τ
a	u	u	$\ldots$	τ	u	$\ldots$	u
b	u	u	$\ldots$	u	τ	$\ldots$	u
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
$\bar{a}$	τ	u	$\ldots$	u	u	$\ldots$	u
$\bar{b}$	u	τ	$\ldots$	u	u	$\ldots$	u
$\vdots$	$\vdots$	$\vdots$		$\vdots$	$\vdots$		$\vdots$
τ	u	u	$\ldots$	u	u	$\ldots$	u

where u means undefined.

Table 5.1: $g : Act^2 \hookrightarrow Act$

Let $f_2 : Act \hookrightarrow Act$ be the function:

$$f_2(x) = \begin{cases} x & \text{if} \quad x \neq a \\ x & \text{if} \quad x \neq \bar{a} \\ c & \text{if} \quad x = a \\ \bar{c} & \text{if} \quad x = \bar{a} \end{cases}$$

When we instantiate $(\ldots)[f]$ with f_2 we obtain the renaming operator $[c/a]$. Let $f_3 : Act \hookrightarrow Act$ be the function:

$$f_3(x) = \begin{cases} x & \text{if} \quad x \neq b \\ x & \text{if} \quad x \neq \bar{b} \\ c & \text{if} \quad x = b \\ \bar{c} & \text{if} \quad x = \bar{b} \end{cases}$$

When we instantiate $(\ldots)[f]$ with f_3 we obtain the renaming operator $[c/b]$. Let $B_1 = \mu x.a.b.x$ and $B_2 = (B_1[f_2] \mid_g B_1[f_3])[f_1]$. $\mathcal{F}_a(\mathcal{F}_c(B_1)) = \mu x.n\tau.n\tau.x$ and since $\mathcal{F}_a(\mathcal{F}_c(B_1))$ does not offer any τ or $mb\tau$ experiment we may conclude that B_1 is stable.

$$\mathcal{F}_a(\mathcal{F}_c(B_2)) = (\mathcal{F}_a(\mathcal{F}_c(B_1))[\alpha \circ \bar{f}_2 \circ \gamma] \mid_{\alpha \circ \bar{g} \circ \gamma} \mathcal{F}_a(\mathcal{F}_c(B_1))[\alpha \circ \bar{f}_3 \circ \gamma])[\alpha \circ \bar{f}_1 \circ \gamma_2]$$

$$= (\mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_2 \circ \gamma] \mid_{\alpha \circ \bar{g} \circ \gamma} \mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_3 \circ \gamma])[\alpha \circ \bar{f}_1 \circ \gamma_2]$$

Since

$$(\mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_2 \circ \gamma] \mid_{\alpha \circ \bar{g} \circ \gamma} \mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_3 \circ \gamma])[\alpha \circ \bar{f}_1 \circ \gamma_2] \xrightarrow{\tau}$$

$$(n\tau.n\tau.x[\mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_2 \circ \gamma]/x][\alpha \circ \bar{f}_2 \circ \gamma] \mid_{\alpha \circ \bar{g} \circ \gamma}$$

$$n\tau.n\tau.x[\mu x.n\tau.n\tau.x[\alpha \circ \bar{f}_3 \circ \gamma]/x][\alpha \circ \bar{f}_3 \circ \gamma])[\alpha \circ \bar{f}_1 \circ \gamma_2]$$

we may conclude that $\mathcal{F}_a(\mathcal{F}_c(B_2))$ is not stable. This shows that although $B_1 \approx B_2$ where $\approx$ is observational equivalence [Mil 80] B_2 enjoys a property which B_1 does not. Hence B_1 and B_2 are not congruent.

In the above example it seems tedious to decide whether B_1 and B_2 enjoy different properties, but of cause this is because we only use the semantics rules in the investigation. We may wish to use algebraic laws to manipulate the terms and clearly terms in CS and AS are much easier to handle by algebraic laws than their similar SS terms.

5.7 Using proof systems together with data flow analysis

We may extend the proof systems of chapter 3 to take the operational semantics of CS and AS into account. All that is required is that the axioms S1-S4 remain sound.

Proposition 5.5 *The axoims S1-S4 of S_r of S_f are sound for AS*

PROOF: The proof of proposition 3.2 and 3.12 has to be extended with the cases

S1 $(p_1+p_2)+p_3 \xrightarrow{l} p'$ and $l = m \sqcup n$ and $p_{+1i} \xrightarrow{m} p'$ and $p_{i+2} \xrightarrow{n} p'$ (here $+$ is modulo 3). But then $p_1 + (p_2+p_3) \xrightarrow{l} p'$ by using $SUM\!\rightarrow$ twice

S2 $p_1+p_2 \xrightarrow{l} p'$ and $l = m \sqcup n$ and and $p_1 \xrightarrow{n} p'$ and $p_2 \xrightarrow{n} p'$ but then $p_2+p_1 \xrightarrow{l} p'$ since $l = m \sqcup n = n \sqcup m$

S3 $p+p \xrightarrow{l} p'$. $l = l \sqcup l$

S4 $p+nil \xrightarrow{l} p'$. But $nil \overset{m}{\not\rightarrow}$ for any m so we cannot use $l = m \sqcup n$.

□

Note that the above proof also generalizes to CS since CS is just a particular instance of AS with $AppAct = \mathcal{P}(Act)$ and $\sqcup = \cup$. Once we have established the above proposition we may use the proof system in a constructive way:

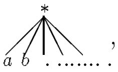
Proposition 5.6 *if p $\underline{corr}_\beta$ q & $q \sqsubseteq_{AppAct} q'$ then p $\underline{corr}_\beta$ q'*

PROOF: Obvious since every action l of q can be matched by q' by an action l' such that $l \sqsubseteq l'$. Note that this proposition would not hold if we insist on $a = \beta(l)$ in the definition of $\underline{corr}_\beta$. □

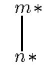
Proposition 5.7 *if p $\underline{safe}^\sigma_\alpha$ q & $p' \sqsubseteq_{\mathcal{P}(Act)} q'$ then p' $\underline{safe}^\sigma_\alpha$ q'*

PROOF: Obvious since every move A of p can be matched by a move $A' \subseteq A$ of p' and a move l' of q' such that $\alpha(A') \sqsubseteq l'$ since α is monotone. □

5.8 Concreteness test

Recall the method of partial specification described in chapter 4. If p and q were concrete, i.e. p and q and all their derivatives could not perform an $*$-action, and $p \sqsubseteq_M \; q$ where $M = Act \setminus \{*\}$ and $Act =$ , then $p \sim q$ where $\sim$ is the bisimulation equivalence [Par 81].

One method of infering that a process p is concrete is by proposition 4.2, if q is concrete and $p \sqsubseteq_M \; q$ then p is concrete. But how were we to know that q was concrete? We could run q testing it and seeing if an $*$-action is possible, but this seems as a rather expensive process. If we could make a static analysis disregarding all other actions and by this process have the answer, then a more attractive analysis has been reached. We shall use the framework of this chapter to this end yielding an analysis for concreteness.

For $Act = \{a, b, \ldots\} \cup \{*\}$ let $AppAct = \{n*, m*\}$ with the ordering $n* \sqsubseteq_{AppAct} m*$ i.e. $AppAct$ may be illustrated as in . $m*$ has the intuition that the action may be $*$, and $n*$ has the intuition that we by certainty know that the action is not $*$.

We may induce the analysis by $\beta : Act \to AppAct$ given by:

$$\beta(a) \quad = \quad \begin{cases} n* & \text{if } a \neq * \\ m* & \text{if } a = * \end{cases}$$

This analysis resembles the wellknown strictness analysis from functional programming where one is interested in information on when operators are strict. Then the parameter mechanism may be exchanged from call-by-name to call-by-value without changing the semantics of the language (see e.g. [Nie 86]).

Returning to the concreteness analysis this states that whenever only $n*$-actions are possible for the abstracted agent, then only actions in $M = Act \setminus \{*\}$ are possible for the real agent.

Instead of using $\alpha \circ \bar{f} \circ \gamma_n$ in $FUN \!\rightarrow_{AS}$ and $\alpha \circ \bar{g} \circ \gamma_2$ in $COM \!\rightarrow_{AS}$ for the communication operator we could use a more abstract function $h : AppAct^2 \to AppAct$ defined as:

$$h(a, b) \quad = \quad \begin{cases} n* & \text{if } a = n* \text{ and } b = n* \\ m* & \text{otherwise} \end{cases}$$

The fact that $\alpha \circ \bar{g} \circ \gamma_2 \sqsubseteq h$ is easy verified.

As an example of the use of the above analysis we may show that $spec = \mu x.a.b.\tau.\tau.x$ defined in chapter 4 is concrete.
By proposition 5.5 $spec \; \underline{corr}_\beta \; \mathcal{F}_a((\mathcal{F}_c spec))$ where

$$\mathcal{F}_a(\mathcal{F}_c(spec)) = \mathcal{F}_a(\mathcal{F}_c(\mu x.a.b.\tau.\tau.x)) = \mu x.\mathcal{F}_a(\mathcal{F}_c(a.b.\tau.\tau.x)) =$$

$$\mu x.\beta(a).\mathcal{F}_a(\mathcal{F}_c((b.\tau.\tau.x)) = \mu x.n * .n * .n * .n * .x$$

We may now "run" $\mathcal{F}_a(\mathcal{F}_c(spec))$ to check that it offers no $m*$- experiment and infer that $spec$ does not offer an $*$-experiment. Also we may see that $q' = \mu x.\bar{c}.(\bar{d}.x + \bar{d}.\mu y. * .y)$ is not concrete since

$$\mathcal{F}_a(\mathcal{F}_c(q')) = \mu x.n * .(n * .x + n * .\mu y.m * .y) \xrightarrow{n*}$$

$$(n * .x + n * .\mu y.m * .y)[\mathcal{F}_a(\mathcal{F}_c(p'))/x] \xrightarrow{n*} \mu y.m * .y \xrightarrow{m*} y[\mu y.m * .y]$$

i.e. $\mathcal{F}_a(\mathcal{F}_c(p'))$ offers an $m*$-experiment and therefore we can not be certain as to whether p' offers an $*$-experiment or not, and further analysis is necessary.

Conclusion

A thorough investigation of a preorder $\sqsubseteq_M$ on processes induced by a preorder $\sqsubseteq_A$ on actions has been presented. The preorder on processes is obtained by extending the notion of bisimulation to take the preorder on actions into account. Also certain "uninteresting" actions may be excluded from consideration by the notion of M-bisimulation.

A language for defining processes has been defined. The language resembles an extract and extension of current variations over CCS. Its semantics is operational and based on a labelled transition system.

We have algebraically characterized the preorder $\sqsubseteq_M$ by three sound and complete proof systems, one for finite terms describing nondeterminism and action-prefixing, the second, an extension of the first, introducing the notion of communication and function, and the third for regular expressions. These proof systems are of interest since many interesting concurrent systems can be expressed in the sublanguages they characterize.

We have instantiated the general theory to a specific instance called partial specification. We have shown that the concurrent alternating bit protocol CABP [Koy 85] meets its specification using partial specification, and it is our claim that our method is both more general and simpler than the method of modularization presented in [Koy 85].

Finally by an extension of the generalized bisimulation we have shown how the method of abstract interpretation may be introduced into hierarchical development of concurrent systems. We have shown how to induce safe and correct analyses and shown how these analyses may answer questions about processes, that otherwise would have demanded a run of the program.

Still, however, there are numerous fields for further investigation. An interesting future problem to investigate is to see if our work extends to that of "weak"-bisimulation equivalence, i.e. if a kind of "weak"-M-bisimulation exists and how it may be axiomatized. In that case it will also be interesting how the notion of divergence will influence on the preorder induced.

Another interesting field is how modal logic may be used to characterize the preorder presented in this thesis and the preorders described above. It is clear that the theory of modal logic offers a lot of interesting aspects not considered in this thesis, aspects such as distinguishing processes by the modal properties they enjoy. Work in this area is going on at the Institute of Electronic Systems, Aalborg University Centre (see e.g. [Hil 87, Ves 86]).

As it may be seen from the verification of the concurrent alternating bit protocol in chapter 4 it will be interesting to apply the theory of partial specifications to larger examples to see how this may shorten the developing time of large systems. But as it may be seen from the verification of the CABP using the proof systems of chapter 4 even modest specifications yield tedious proofs.

Therefore it will be interesting if an automatic system of M-bisimulation checking could be designed and implemented just as it was shown in [Lar 86a] for bisimulation and "weak"-bisimulation. It seems like it will at least be possible for the partial specification problem since this is just a slight extension of the notion of bisimulation, and since we may stop the search of a branch for further matches as soon as a pair $(p, \mathcal{U})$ has been encountered.

Finally, further investigations in how methods of data flow analysis may be applied to concurrent system development will be interesting. For example does a "constant propagation" of actions exist?, a "live variable"-analysis of actions? etc. Also the question about the chosen static (collecting) semantics is left open, is it intuitively the most precise as is the case for sequential programs? or does there exist one taking the nature of nondeterminism into account in another way?

Bibliography

[Acz 84] P. Aczel: "A Simple Version of SCCS and Its Semantics",
Unpublished notes, Edinburgh 1984.

[Aho 77] A. V. Aho & J. D. Ullman: "Principles of Compiler Design",
London, Addison Wesley 1977.

[Bar 85] H. P. Barendreght: "The Lambda Calculus – Its Syntax and Semantics",
North-Holland l985.

[Car 81] L. Cardelli: "Real Time Agents",
LNCS 104, Springer Verlag 1981.

[Cou 77] P. Cousot & R. Cousot: "Abstract Interpretation: a unified lattice model for static analysis of programs by construction of approximation of fixpoints",
In: Conf. Record of the 4th ACM symposium on Principles of Programming Languages 1977.

[Cou 79] P. Cousot & R. Cousot: "Systematic design of Program Analysis Frameworks",
In: Conf. Record of the 6th ACM symposium on Principles of Programming Languages 1979.

[Gla 85] R. J. Glabbeck: "Bounded Nondeterminism and the Approximation Induction Principle in Process Algebra",
Centre for Mathematics and Computer Science, The Netherlands 1985.

[Hen 84] M. Hennessy: "Axiomatising Finite Delay Operators",
Acta Informatica 21 (61-88), Springer-Verlag 1984.

[Hen 85] M. Hennessy & R. Milner: "Algebraic Laws for Nondeterminism and Concurrency",
Journal of the Association for Computing Machinery pp. 137-161, 1985.

[Hil 87] M. Hillerström: "Verification of CCS–processes",
M. Sc. Thesis, Aalborg University Centre, 1987.

[Koy 85] C. P. Koymans & J. C. Mulder: "A Modular Approach to Protocol Verification Using Process Algebra",
Logic Group Preprint Series No. 6, University of Utrecht 1985.

[Lar 86a] K. G. Larsen: "Context Dependent Bisimulation Between Processes",
Ph. D. Thesis, Edinburgh University 1986.

[Lar 86b] K. G. Larsen & R. Milner: "A Complete Protocol Verification Using Relativized Bisimulation",
Institute of Electronic Systems, Aalborg University Centre, R-86-12 September 1986.

[Mil 80] R. Milner: "A Calculus of Communicating Systems",
LNCS 12, Springer Verlag, 1980.

[Mil 81] R. Milner: "A Complete Inference System for a Class of Regular Behaviours",
University of Edinburgh 1981.

[Mil 83] R. Milner: "Calculi for Synchrony and Asynchrony",
Theoretical Computer Science 25 (1983) pp 269-310, North Holland.

[Nie 82] F. Nielson: "A Denotational Framework For Data Flow Analysis",
Acta Informatica, Springer Verlag, 1982.

[Nie 85] F. Nielson: "A Bibliography on Abstract Interpretation",
Institute of Electronic Systems, Aalborg University Centre 1985.

[Nie 86] F. Nielson: "Strictness Analysis And Denotational Abstract Interpretation",
Institute of Electronic Systems, Aalborg University Centre, R-86-9 1986.

[Par 81] D. Park: "Concurrency and Automata on Infinite Sequences",
LNCS 104, Springer Verlag, 1981.

[Plo 81] G. Plotkin: "A Structural Approach to Operational Semantics",
DAIMI FN-19 Aarhus University Computer Science Department 1981.

[Smy 78] M. Smyth: "Power Domains",
Journal of Computers and System Science Vol. 2 pp. 23-36, 1978.

[Sto 77] J. Stoy: "Denotational Semantics: The Scott-Strachey Approach to Programming Languages Theory",
The MIT Press 1977.

[Tar 55] A. Tarski: "A Lattice-Theoretical Fixpoint Theorem and Its Applications",
Pacific Journal of Math. 5, 1955.

[Ves 86] K. Vestmar & J. Olesen: "Specifikation og Implementation af Fuldautomatisk Verifikationsværktøj Baseret på en Operationel Semantik",
M. Sc. Thesis, Aalborg University Centre, 1986.

Appendix A

In this appendix we present the full proofs of propositions, lemmas and theorems which have been presented but not proven in full in chapter 3 of the thesis.

The proofs are presented here either because they are long, tedious and sometimes trivial or because they only are of interest for readers interested in the underlying theoretical development.

Proposition A.1 *(Proposition 3.1)*
$\sqsubseteq_M$ *is a precongruence with respect to the operators of* Σ^{nf}

PROOF:

$p = a.p$: We prove the more general inference rule $C1$ sound. The relation

$$R_1 = \{(a.p_1, b.p_2) \mid a \sqsubseteq_A \; b \;\&\; p_1 \sqsubseteq_M \; p_2\} \cup \sqsubseteq_M$$

is an M-bisimulation.
To see that $R_1 \subseteq M\mathcal{B}(R_1)$ observe that if $a.p_1 \xrightarrow{a} p_1$ then by $ACT\rightarrow$: $b.p_2 \xrightarrow{b} p_2$ and by definition of R_1: $a \sqsubseteq_A \; b$ and $(p_1, p_2) \in \sqsubseteq_M \subseteq R_1$.
Also if $b \in M$ then $b.p_2 \xrightarrow{b} p_2$ and by $ACT\rightarrow$: $a.p_1 \xrightarrow{a} p_1$ with $a \sqsubseteq_A \; b$ and $(p_1, p_2) \in R_1$ which is the matching move. If $b \notin M$ the case holds trivially.

$p = p_1 + p_2$: The relation

$$R_2 = \{(p_1 + p_2, q_1 + q_2) \mid p_1 \sqsubseteq_M \; q_1 \;\&\; p_2 \sqsubseteq_M \; q_2\} \cup \sqsubseteq_M$$

is an M-bisimulation. To see that $R_2 \subseteq M\mathcal{B}(R_2)$ observe that if

1. $p_1 + p_2 \xrightarrow{a} p'$ then
   either $p' = p_1'$ and $p_1 \xrightarrow{a} p_1'$. Then by definition of R_2 there exist q_1' and b such that $q_1 \xrightarrow{b} q_1'$ and $a \sqsubseteq_A \; b$ and $p_1' \sqsubseteq_M \; q_1'$. By $SUM\rightarrow$: $q_1 + q_2 \xrightarrow{b} q_1'$ which obviously is the matching move.
   or $p' = p_2'$ and $p_2 \xrightarrow{a} p_2'$. The case is similar.

2. $q_1 + q_2 \xrightarrow{a} q'$, $a \in M$ then

   either $q' = p_1'$ and $q_1 \xrightarrow{a} q_1'$. Then by definition of R_2 there exist
   p_1' and b such that $p_1 \xrightarrow{b} p_1'$ and $b \sqsubseteq_A a$ and $p_1' \sqsubseteq_M q_1'$. By
   $SUM \rightarrow$: $p_1 + p_2 \xrightarrow{b} q_2'$ which is the matching move.

   or $q' = q_2'$ and $q_2 \xrightarrow{a} q_1'$. The case is similar.

□

Proposition A.2 *(Proposition 3.2)*
The following holds for all $p_1, p_2, p_3 \in T_{\Sigma^{nf}}$:

$$\begin{aligned}
p_1 + (p_2 + p_3) &\simeq_M (p_1 + p_2) + p_3 \\
p_1 + p_2 &\simeq_M p_2 + p_1 \\
p_1 + p_1 &\simeq_M p_1 \\
p_1 + nil &\simeq_M p_1
\end{aligned}$$

Here $\simeq_M$ means that both $\sqsubseteq_M$ and $\sqsupseteq_M$ hold.

Proof: The relation

$$R_1 = \{((p_1 + p_2) + p_3, p_1 + (p_2 + p_3)) \mid p_1, p_2, p_3 \in Pr\} \cup Id$$

is an M-bisimulation.
To see that $R_1 \subseteq M\mathcal{B}(R_1)$ observe the following cases:

1. $(p_1 + p_2) + p_3 \xrightarrow{a} p'$

   i. $p' = p_1'$ and $p_1 \xrightarrow{a} p_1'$. Then by $SUM \rightarrow$: $p_1 + (p_2 + p_3) \xrightarrow{a} p_1'$ clearly $a \sqsubseteq_A a$ and $(p_1', p_1') \in Id \subseteq R_1$.

   ii. $p' = p_2'$ and $p_2 \xrightarrow{a} p_2'$. Then by $SUM \rightarrow$ twice: $p_1 + (p_2 + p_3) \xrightarrow{a} p_2'$ which obviously is the matching move.

   iii. $p' = p_3'$ and $p_3 \xrightarrow{a} p_3'$. As in ii.

2. $p_1 + (p_2 + p_3) \xrightarrow{a} p'$, $a \in M$. Similar arguments as in i yield the result.

Also the relation R_1^{-1} is an M-bisimulation.
The relation

$$R_2 = \{(p_1 + p_2, p_2 + p_1) \mid p_1, p_2 \in Pr\} \cup Id$$

is an M-bisimulation.
Also the relation R_2^{-1} is an M-bisimulation.
The relation

$$R_3 = \{(p_1 + p_1, p_1) \mid p_1 \in Pr\} \cup Id$$

is an M-bisimulation.
Also the relation R_3^{-1} is an M-bisimulation.
The relation

$$R_4 = \{(p_1 + nil, p_1) \mid p_1 \in Pr\} \cup Id$$

is an M-bisimulation.
To see that $R_4 \subseteq M\mathcal{B}(R_4)$ observe that $nil \overset{a}{\not\rightarrow}$ for all $a \in Act$ and therefore $p_1 + nil \xrightarrow{a} p'$ iff $p_1 \xrightarrow{a} p'$ which p obviously can match.
Also the relation R_4^{-1} is an M-bisimulation.
□

Proposition A.3 *(Proposition 3.4)*
Every term p in Σ^{nf} has a normalform $nf(p)$ such that $\vdash p =_M nf(p)$

Proof: By structural induction.

$p = nil$: $nf(nil) = nil$ because of P1: $\vdash nil \sqsubseteq_M nil$.

$p = a.p'$: Assume there exists $nf(p')$ such that $\vdash p' \sqsubseteq_M nf(p')$. Choose $nf(ap') = a.nf(p')$. Then clearly $nf(a.p')$ is in normalform by the assumption and C1 we $\vdash a.p' \sqsubseteq_M a.nf(p')$ since $a \sqsubseteq_A a$.

$p = p_1 + p_2$:
We may assume that $nf(p_1) = \sum_{i=1}^{n} a_i.p_i$ and $nf(p_2) = \sum_{i=n+1}^{m} a_i.p_i$. The induction hypothesis states that

$$\vdash p_i = nf(p_i),\ i \in \{1,2\}$$

From C2 we may infer

$$\vdash p_1 + p_2 = nf(p_1) + nf(p_2)$$

We choose $nf(p) = \sum_{i=1}^{m} a_i.p_i$ in such a way that

$$\vdash nf(p) = nf(p_1) + nf(p_2) \qquad (A.1)$$

and then use P2 to obtain the desired result. We only sketch the proof of (A.1) since it is tedious handwork. If $n = 0$ or $m = n$ we use S4 to eliminate nil, and in case of "duplicates" we use S3 to eliminate these. Also $\vdash nf(p) \sqsubseteq_M p$ may be proven by arguments as above yielding $\vdash p =_M nf(p)$.

□

Proposition A.4 *(Proposition 3.5)*
If p and q are in normal form then:

$$p \sqsubseteq_M q \Rightarrow \vdash p \sqsubseteq_M q$$

Proof: We prove this by induction on the size of p and q, noting that $p \xrightarrow{a} p'$ iff $a.p'$ is a subterm of p and the same holds for q. We therefore assume both p and q take the form $\sum_{i=1}^{n} a_i.p_i$ and $\sum_{j=1}^{m} b_j.q_j$.
Assume $p \sqsubseteq_M q$.
First consider the case $p = nil$, that is $n = 0$, since $p \sqsubseteq_M q$, q must be nil or $\sum_{j=1}^{m} b_j.q_j$ with all $b_j \notin M$. In the first case P1 yields the desired result and in the second case ANNIHIL and C1 and S4 yield the desired result.
Now consider the case $n > 0$.
Let $a_i p_i$ be a subterm of p. Thus if $p \xrightarrow{a_i} p'$ then for some q': $q \xrightarrow{c_j} q'$ with $a_i \sqsubseteq_A c_j$ and $p_i \sqsubseteq_M q'$. By the above remark q' must be q_j for some j and by induction $\vdash p_i \sqsubseteq_M q_j$ and by C1 $\vdash a_i.p_i \sqsubseteq_M c_j.q_j$.
Now let $c_i.q_i$ be a subterm of q. Thus if $q \xrightarrow{c_i} q'$ then if $c_i \in M$ then for some p': $p \xrightarrow{a_j} p'$ with $a_j \sqsubseteq_A c_i$ and $p' \sqsubseteq_M q_i$. But p' must be p_j for some j. By induction $\vdash p_j \sqsubseteq_M q_i$ and by C1: $\vdash a_j.p_j \sqsubseteq_M c_i.q_i$. If $c_i \notin M$ then by ANNIHIL $\vdash nil \sqsubseteq_M c_i.q_i$.
So for every i there exists a j such that $\vdash a_i.p_i \sqsubseteq_M c_j.q_j$ and for every j there exists an i such that $\vdash a_j.p_j \sqsubseteq_M c_i.q_i$ if $c_i \in M$ or $\vdash nil \sqsubseteq_M c_i.q_i$. otherwise. By repeated use of C2 we may now build up p and q term by term such that $\vdash p \sqsubseteq_M q$ holds using S3 to eliminate "duplicates" and S4 to eliminate nil's. □

Proposition A.5 *(Proposition 3.6)*
$\sqsubseteq_M$ is a precongruence with respect to the operators of Σ^{cf}, provided g in $|_g$ is monotone.

Proof: As in proposition 3.1 with the extension that the relation:

$$R = \{(p_1 |_g p_2, q_1 |_g q_2) \mid p_i \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_i, i \in \{1,2\}\}$$

is an M-bisimulation.
To see that $R \subseteq M\mathcal{B}(R)$ observe the following cases:

1. $p_1 |_g p_2 \xrightarrow{a} p'$

   i. $p' = p_1' |_g p_2$ and $p_1 \xrightarrow{a} p_1'$. Then by definition of R there exist q_1' and b such that $q_1 \xrightarrow{b} q_1'$ with $a \sqsubseteq_M b$ and $p_1' \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_1'$. By $COM \rightarrow$: $q_1 |_g q_2 \xrightarrow{b} q_1' |_g q_2$ which obviously is the matching move.

   ii. $p' = p_1 |_g p_2'$ and $p_2 \xrightarrow{a} p_2'$. The case holds by similar arguments as in i.

   iii. $p' = p_1' |_g p_2'$ and $p_1 \xrightarrow{b} p_1'$ and $p_2 \xrightarrow{c} p_2'$ and $a = g(b,c)$. By definition of R : $q_1 \xrightarrow{d} q_1'$ with $b \sqsubseteq_A d$ and $p_1 \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_1'$ and $q_2 \xrightarrow{e} q_2'$ with $c \sqsubseteq_A e$ and $p_2 \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q_2'$. By $COM \rightarrow$ and monotonicity of g: $q_1 |_g q_2 \xrightarrow{g(d,e)} q_1' |_g q_2'$ which obviously is the matching move.

2. $q_1 \mid_g q_2 \xrightarrow{a} q'$, $a \in M$

   i. $q' = q'_1 \mid_g q_2$ and $q_1 \xrightarrow{a} q'_1$. Then by definition of R there exist p'_1 and b such that $p_1 \xrightarrow{b} p'_1$ with $b \sqsubseteq_M a$ and $p'_1 \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q'_1$. By $COM\!\rightarrow$: $p_1 \mid_g p_2 \xrightarrow{b} p'_1 \mid_g p_2$ which obviously is the matching move.

   ii. $q' = q_1 \mid_g q'_2$ and $q_2 \xrightarrow{a} q'_2$. The case holds by similar arguments as in i.

   iii. $q' = q'_1 \mid_g q'_2$ and $q_1 \xrightarrow{b} q'_1$ and $q_2 \xrightarrow{c} q'_2$ and $a = g(b,c)$. $b, c \in g^{-1}(M)$ then by definition of R there exist d and p'_1 such that $p_1 \xrightarrow{d} p'_1$ with $d \sqsubseteq_A c$ and $p'_1 \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q'_1$. Also there exist e and p'_2 such that $p_2 \xrightarrow{e} p'_2$ with $e \sqsubseteq_A c$ and $p'_2 \sqsubseteq_{M \cup CL(g^{-1}(M)\downarrow i)} q'_2$. Since g is monotone we may infer from $COM\!\rightarrow$ that $p_1 \mid_g p_2 \xrightarrow{g(d,e)} p'_1 \mid_g p'_2$ which is the matching move.

□

Proposition A.6 *(Proposition 3.7)*
if $p = \sum_{i=1}^{n} a_i.p_i$ and $q = \sum_{j=1}^{m} b_j.q_j$ then

$$p \mid_g q \simeq_M \sum_{(i,j)\in\{(i,j) \mid g(a_i,b_j)\ defined\}} g(a_i,b_j).(p_i \mid_g q_j) +$$
$$\sum_{i=1}^{n} a_i.(p_i \mid_g q) + \sum_{j=1}^{m} b_j.(p \mid_g q_j)$$

PROOF: The relation

$$R = \{(p \mid_g q, \sum_{(i,j)\in\{(i,j) \mid g(a_i,b_j)\ defined\}} g(a_i,b_j).(p_i \mid_g q_j) +$$
$$\sum_{i=1}^{n} a_i.(p_i \mid_g q) + \sum_{j=1}^{m} b_j.(p \mid_g q_j))$$
$$\mid p = \sum_{i=1}^{n} a_i.p_i \ \&\ q = \sum_{j=1}^{m} b_j.q_j\}$$

is an M-bisimulation.
To see that $R \subseteq M\mathcal{B}(R)$ observe the following cases:

1. $p \mid_g q \xrightarrow{a} p'$

   i. $p' = p_i \mid_g q_j$ and $a = g(a_i, b_j)$ and $p \xrightarrow{a_i} p_i$ and $q \xrightarrow{b_j} q_j$. Then by definition of R and $ACT\!\rightarrow$ and $SUM\!\rightarrow$:
   $\sum_{(i,j)\in\{(i,j)\mid g(a_i,b_j)\ defined\}} g(a_i, b_j).(p_i \mid_g q_j) +$
   $\sum_{i=1}^{n} a_i.(p_i \mid_g q) + \sum_{j=1}^{m} b_j.(p \mid_g q_j) \xrightarrow{g(a_i,b_j)} p_i \mid_g q_j$
   which is the matching move.

   ii. $p' = p_i \mid_g q$ and $a = a_i$ and $p \xrightarrow{a_i} p_i$. Then by $ACT \rightarrow$ and $SUM\!\rightarrow \sum_{(i,j)\in\{(i,j)\mid g(a_i,b_j)\ defined\}} g(a_i, b_j).(p_i \mid_g q_j) + \sum_{i=1}^{n} a_i.(p_i \mid_g q) + \sum_{j=1}^{m} b_j.(p \mid_g q_j) \xrightarrow{a_i} p_i \mid_g q$ which is the matching move.

   iii. $p' = p \mid_g q_j$ and $a = b_j$ and $q \xrightarrow{b_j} q_j$. Similar to ii.

2. $\sum_{(i,j)\in\{(i,j)\mid g(a_i,b_j)\ defined\}} g(a_i, b_j).(p_i \mid_g q_j) +$
   $\sum_{i=1}^{n} a_i.(p_i \mid_g q) + \sum_{j=1}^{m} b_j.(p \mid_g q_j) \xrightarrow{a} q'$, $a \in M$

   i. $q' = p_i \mid_g q_j$ and $a = g(a_i, b_j)$. By definition of R and q and $\{(i,j) \mid g(a_i, b_j) defined\}$ we know that $p \xrightarrow{a_i} p_i$ and $q \xrightarrow{b_j} q_j$ and $g(a_i, b_j)$ is defined so by $COM\!\rightarrow$ $p \mid_g q \xrightarrow{g(a_i,b_j)} p_i \mid_g q_j$ which is the matching move.

   ii. As in 1.ii.

   iii. As in 1.ii.

Also similar arguments as above yield the result that R^{-1} is an M-bisimulation.
□

Proposition A.7 *(Proposition 3.8)*
$\sqsubseteq_M$ is a precongruence with respect to the operators of Σ^f provided g in $\mid_g$ and f in $(\ldots)[f]$ are monotone.

Proof: As in proposition 3.6 with the extension that the relation

$$R = \{((p_1 \ldots p_n)[f], (q_1 \ldots q_n)[f]) \mid p_i \sqsubseteq_{CL(f^{-1}(M)\downarrow i)} q_i\}$$

is an M-bisimulation.
To see that $R_3 \subseteq M\mathcal{B}(R_3)$ observe that if $(p_1 \ldots p_n)[f] \xrightarrow{a} p'$ then $p' = (p'_1 \ldots p'_n)[f]$ and $a = f(a_1 \ldots a_n)$ and $p_i \xrightarrow{a_i} p'_i$. By definition of R_3 there exist q'_i's and b_i's such that $q_i \xrightarrow{b_i} q'_i$ with $a_i \sqsubseteq_A b_i$ and $p'_i \sqsubseteq_{CL(f^{-1}(M)\downarrow i)} q'_i$. Since f is monotone $b = f(b_1 \ldots b_n)$ exists and $f(a_1 \ldots a_n) \sqsubseteq_A f(b_1 \ldots b_n)$. By $FUN\!\rightarrow$: $(q_1 \ldots q_n)[f] \xrightarrow{b} (q'_1 \ldots q'_n)[f]$ which is the matching move.
Also if $(q_1 \ldots q_n)[f] \xrightarrow{a} q'$, $a \in M$ then $q' = (q'_1 \ldots q'_n)[f]$ and
$q_i \xrightarrow{a_i} q'_i$, $a_i \in\sqsubseteq_{CL(f^{-1}(M)\downarrow i)}$. By definition of R_3 there exist p'_i's and b_i's such that $p_i \xrightarrow{b_i} p'_i$ with $b_i \sqsubseteq_A a_i$ and $p'_i \sqsubseteq_{CL(f^{-1}(M)\downarrow i)} q_i$. Again since f is monotone and all sets have to be downwardsclosed: $f(b_1 \ldots b_n) \sqsubseteq_A f(a_1 \ldots a_n)$. Clearly $f(a_1 \ldots a_n) \in M$ implies $f(b_1 \ldots b_n) \in M$. By $FUN\!\rightarrow$: $(p_1 \ldots p_n)[f] \xrightarrow{f(b_1 \ldots b_n)} (p'_1 \ldots p'_n)[f]$ which is the matching move. □

Proposition A.8 *(Proposition 3.9)*
The following holds for $p_1 \dots p_n \in T_{\Sigma^f}$*:*

$$\begin{array}{rcl}
(p_1 \dots p_n)[f] & \simeq_M & nil \ if \ p_i \equiv nil \ for \ some \ i \leq n \\
(a_1.p_1 \dots a_n.p_n)[f] & \simeq_M & \begin{cases} f(a_1 \dots a_n).(p_1 \dots p_n)[f] \\ if \ f(a_1 \dots a_n) \ is \ defined \\ \\ nil \quad otherwise \end{cases} \\
(p_1 \dots p_i + q_i \dots p_n)[f] & \simeq_M & (p_1 \dots p_i \dots p_n)[f] + \\
 & & (p_1 \dots q_i \dots p_n)[f]
\end{array}$$

Proof: The relation

$$R_1 = \{((p_1 \dots p_n)[f], nil) \mid p_i \equiv nil, \ i \leq n\}$$

is an M-bisimulation.
To see this observe that $(p_1 \dots p_n)[f] \overset{a}{\not\rightarrow}$ for any a since $f(a_1 \dots a_n)$ is not defined since a_i does not exist. Also $nil \overset{a}{\not\rightarrow}$ for any a.
The relation R_1^{-1} is also an M-bisimulation.

The relation

$$R_2 = \{((a_1.p_1 \dots a_n.p_n)[f], f(a_1 \dots a_n).(p_1 \dots p_n)[f]) \\ \mid f(a_1 \dots a_n) \ is \ defined\} \cup Id$$

is an M-bisimulation.
To see that $R_2 \subseteq M\mathcal{B}(R_2)$ observe that by $ACT\rightarrow$: $a_i.p_i \xrightarrow{a_i} p_i$ and by definition of R_2: $f(a_1 \dots a_n)$ is defined and by $FUN\rightarrow$: $(a_1.p_1 \dots a_n.p_n)[f] \overset{f(a_1 \dots a_n)}{\longrightarrow}$ $(p_1 \dots p_n)[f]$. Also by $ACT\rightarrow$: $f(a_1 \dots a_n).(p_1 \dots p_n)[f] \overset{f(a_1 \dots a_n)}{\longrightarrow} (p_1 \dots p_n)[f]$ which is the matching move.

Also R_2^{-1} is an M-bisimulation.

The relation

$$R_3 = \{((a_1.p_1 \dots a_n.p_n)[f], nil) \mid f(a_1 \dots a_n) \ is \ undefined\}$$

is an M-bisimulation.
To see this observe that the only action of $(a_1.p_1 \dots a_n.p_n)[f]$ should be $f(a_1 \dots a_n)$ so $(a_1.p_1 \dots a_n.p_n)[f] \overset{a}{\not\rightarrow}$ for any a, also $nil \overset{a}{\not\rightarrow}$ for any a.

Also R_3^{-1} is an M-bisimulation.

The relation

$$R_4 = \{((p_1 \dots p_i + q_i \dots p_n)[f], (p_1 \dots p_i \dots p_n)[f] + (p_1 \dots q_i \dots p_n)[f])\} \cup Id$$

is an M-bisimulation.
To see this observe that if $(p_1 \ldots p_i + q_i \ldots p_n)[f] \xrightarrow{a} p'$ then

either $p' = (p'_1 \ldots p'_i \ldots p'_n)[f]$ and $\forall i . p_i \xrightarrow{a_i} p'_i$ and $a = f(a_1 \ldots a_n)$ which by $FUN \rightarrow$: $(p_1 \ldots p_i \ldots p_n)[f]$ can match and the result follows from $SUM \rightarrow$.

or $p' = (p'_1 \ldots q'_i \ldots p'_n)[f]$ and $\forall j \neq i . p_j \xrightarrow{a_j} p_j$ and $q_i \xrightarrow{a_i} q'_i$ and $a = f(a_1 \ldots a_n)$ and the result follows by arguments as above.

Also R_4^{-1} is an M-bisimulation. □

Property A.1 *(Property 3.1)*
Whenever $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ *then*

either *for some* $p'' : p \xrightarrow{a} p''$ *and* $p' = p''[\bar{r}/\bar{x}]$

or *for some* $x_i \in UG(p) : r_i \xrightarrow{a} p'$

PROOF: By induction on inferences. Observe the possible form of p.

$p = nil$: The statement holds trivially since $nil[\bar{r}/\bar{x}] \equiv nil$.

$p = a.p_1$: By definition all variables in p are guarded since they are within a subterm $a.p_1$, since the only action of $a.p_1$ is a and $(a.p_1)[\bar{r}/\bar{x}] \equiv a.p_1[\bar{r}/\bar{x}]$ then $a.p_1 \xrightarrow{a} p_1$ and $p' = p_1[\bar{r}/\bar{x}]$.

$p = p_1 + p_2$: Assume the proposition holds for p_1 and p_2. Since $(p_1 + p_2)[\bar{r}/\bar{x}] \equiv p_1[\bar{r}/\bar{x}] + p_2[\bar{r}/\bar{x}]$ the proposition holds by induction and $SUM \rightarrow$.

$p = x_i$: Then by definition of substitution $x_i[\bar{r}/\bar{x}] \equiv r_i$, so $x_i[\bar{r}/\bar{x}] \xrightarrow{a} p'$ iff $r_i \xrightarrow{a} p'$ and the proposition holds.

$p = \mu x . p_1$: Since the only actions of $\mu x . p_1$ are the actions of $p_1[\mu x . p_1 / x]$: $(\mu x . p_1)[\bar{r}/\bar{x}] \xrightarrow{a} p'$ iff $(p_1[\mu x . p_1 / x])[\bar{r}/\bar{x}] \xrightarrow{a} p'$ which holds by induction.

□

We also need the opposite properties:

Property A.2 *(Property 3.2)*
Whenever $p' : p \xrightarrow{a} p'$ *then* $p[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$

PROOF: We prove this by induction on inferences. Observe the possible form of p.

$p = nil$: The case holds trivially since $nil \overset{a}{\not\rightarrow}$ for all a.

$p = a.q$: $a.q \xrightarrow{a} q$ by $ACT \rightarrow$, by definition of substitution $(a.q)[\bar{r}/\bar{x}] \equiv a.(q[\bar{r}/\bar{x}])$, and $a.(q[\bar{r}/\bar{x}]) \xrightarrow{a} q[\bar{r}/\bar{x}]$ by $ACT \rightarrow$, and the desired result holds.

$p = p_1 + p_2$: If $p_1 + p_2 \xrightarrow{a} p'$ then either $p_1 \xrightarrow{a} p'$ or $p_2 \xrightarrow{a} p'$ by a shorter inference. Thus either $p_1[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$ or $p_2[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$ by the induction hypothesis, by $SUM \rightarrow$: $p_1[\bar{r}/\bar{x}] + p_2[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$ and by definition of substitution $p[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$ which is the desired result.

$p = y$: Then the result holds trivially since $y \overset{a}{\not\rightarrow}$.

$p = \mu x.q$: If $\mu x.q \xrightarrow{a} p'$ then this is due to $q[\mu x.q/x] \xrightarrow{a} p'$ by a shorter inference. Thus by induction $q[\mu x.q/x][\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$. By $REC \rightarrow$ and definition of substitution we may infer that $(\mu x.q)[\bar{r}/\bar{x}] \xrightarrow{a} p'[\bar{r}/\bar{x}]$ which is the desired result.

□

Property A.3 *(Property 3.3)*
Whenever $x_i \in UG(p)$ and $r_i \xrightarrow{a} p'$ then $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$

Proof: We prove this by induction on inferences. Observe the possible form of p.

$p = nil$: The case holds trivially since $UG(nil) = \emptyset$

$p = a.q$: The case holds trivially since $UG(a.q) = \emptyset$

$p = p_1 + p_2$: If $x_i \in UG(p)$ then $x_i \in UG(p_1)$ or $x_i \in UG(p_2)$ or both. If $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ then either $p_1[\bar{r}/\bar{x}] \xrightarrow{a} p'$ or $p_2[\bar{r}/\bar{x}] \xrightarrow{a} p'$ by a shorter inference. Thus if $x_i \in UG(p_1)$ or $x_i \in UG(p_2)$ and $r_i \xrightarrow{a} p'$ then $p_1[\bar{r}/\bar{x}] \xrightarrow{a} p'$ or $p_2[\bar{r}/\bar{x}] \xrightarrow{a} p'$. By $SUM \rightarrow$ and definition of substitution $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ which is the desired result.

$p = y$: If $y \notin \bar{x}$ the result holds trivially. If $y \in \bar{x}$ then $y = x_i$ for some i and $UG(p) = \{x_i\}$. If $r_i \xrightarrow{a} p'$ then $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ by definition of substitution.

$\mu x.q$: If $x_i \in UG(p)$ then $x_i \in UG(q)$ and $x_i \not\equiv x$. If $r_i \xrightarrow{a} p'$ then $q[\bar{r}/\bar{x}] \xrightarrow{a} p'$ by induction and also $q[\bar{r}/\bar{x}][\mu x.q/x] \xrightarrow{a} p'$. By $REC \rightarrow$: $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ which is the desired result.

□

Property A.4 *(Property 3.4)*
If no x_i is free in p then $p[\bar{r}/\bar{x}] \equiv p$

Proof: By structural induction on p.

$p = nil$: $free(nil) = \emptyset$ and by definition of substitution $nil[\bar{r}/\bar{x}] \equiv nil$.

$p = a.q$: $free(p) = free(q)$. Assume $q[\bar{r}/\bar{x}] \equiv q$ then $(a.q)[\bar{r}/\bar{x}] \equiv a.(q[\bar{r}/\bar{x}])$ by definition of substitution and $a.(q[\bar{r}/\bar{x}]) \equiv a.q$ by induction.

$p = y$: If no x_i is free in p then $y \notin \bar{x}$ so $y[\bar{r}/\bar{x}] \equiv y$ by definition of substitution.

$p = p_1 + p_2$: Assume no x_i is free in p_1 nor p_2 and $p_1[\bar{r}/\bar{x}] \equiv p_1$ and $p_2[\bar{r}/\bar{x}] \equiv p_2$. Since $free(p) = free(p_1) \cup free(p_2)$ no x_i is free in p. Also $p_1[\bar{r}/\bar{x}] + p_2[\bar{r}/\bar{x}] \equiv (p_1 + p_2)[\bar{r}/\bar{x}]$ by definition of substitution, and also $p_1 + p_2 \equiv (p_1 + p_2)[\bar{r}/\bar{x}]$ by induction.

$p = \mu x.p_1$: Assume no x_i is free in p_1 and $p_1[\bar{r}/\bar{x}] \equiv p_1$. Since $free(p) = free(p_1) \setminus \{x\}$ and $(\mu x.p_1)[\bar{r}/\bar{x}] \equiv \mu x.p_1[\bar{r}/\bar{x}]$ if x is not in $\bar{x}$ nor free in $\bar{r}$ which by induction yields $\mu x.p_1[\bar{r}/\bar{x}] \equiv \mu x.p_1$. Otherwise $(\mu x.p_1) \equiv \mu z.(p_1[z/x][\bar{r}/\bar{x}])$ for some z not in $\bar{x}$ nor free in $\mu x.p_1$ or $\bar{r}$, and the result follows by the induction hypothesis applied to $p_1[z/x]$.

□

Property A.5 *(Property 3.5)*
If $\bar{x}$ and $\bar{y}$ are disjoint then:

$$p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$$

Proof: By structural induction on p.

$p = nil$: $nil[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv nil \equiv nil[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ by definition of substitution.

$p = a.q$: Assume $\bar{x}$ and $\bar{y}$ are disjoined and $q[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv q[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ then

$$\begin{array}{rcl} p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] & \equiv & a.(q[\bar{q}/\bar{x}][\bar{r}/\bar{y}]) \\ & & \quad \text{by definition of substitution} \\ & \equiv & a.(q[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]) \\ & & \quad \text{by induction} \\ & \equiv & (a.q)[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \\ & & \quad \text{by definition of substitution} \end{array}$$

$p = p_1 + p_2$: Assume $p_i[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p_i[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ $i \in \{1, 2\}$ then

$$\begin{array}{rcl} (p_1 + p_2)[\bar{q}/\bar{x}][\bar{r}/\bar{y}] & \equiv & p_1[\bar{q}/\bar{x}][\bar{r}/\bar{y}] + p_2[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \\ & & \quad \text{by definition of substitution} \\ & \equiv & p_1[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] + p_2[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \\ & & \quad \text{by induction} \\ & \equiv & (p_1 + p_2)[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \\ & & \quad \text{by definition of substitution} \end{array}$$

$p = x :$ If $x \in \bar{x}$ then $x[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv q_i[\bar{r}/\bar{y}]$, also $x[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv q_i[\bar{r}/\bar{y}]$. If $x \in \bar{y}$ similar arguments hold. If $x \notin \bar{x} \cup \bar{y}$ then $x[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv x \equiv x[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$.

$p = \mu x.q :$ Assume $q[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv q[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$. Then if x not in $\bar{x}$ nor in $\bar{y}$ and not free in $\bar{q}$ nor $\bar{r}$.

$$\begin{array}{rcl}
(\mu x.q)[\bar{q}/\bar{x}][\bar{r}/\bar{y}] & \equiv & \mu x.(q[\bar{q}/\bar{x}][\bar{r}/\bar{y}]) \\
 & & \text{by definition of substitution} \\
 & \equiv & \mu x.(q[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]) \\
 & & \text{by induction} \\
 & \equiv & (\mu x.q)[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \\
 & & \text{by definition of substitution} \\
 & & \text{Otherwise} \\
(\mu x.q)[\bar{q}/\bar{x}][\bar{r}/\bar{y}] & \equiv & (\mu z.q[z/x])[\bar{q}/\bar{x}][\bar{r}/\bar{y}]
\end{array}$$

for z not in $\bar{x}$ or $\bar{y}$ nor free in q or $\bar{q}$ or $\bar{r}$, and the result follows by induction on $q[z/x]$.

□

Definition A.1
We define the possible actions of a process by an inductively defined function $Names : T_{\Sigma^r} \to \mathcal{P}(Act)$*:*

$$\begin{array}{rcl}
Names(nil) & = & \emptyset \\
Names(a.p) & = & \{a\} \cup Names(p) \\
Names(p_1 + p_2) & = & Names(p_1) + Names(p_2) \\
Names(x) & = & \emptyset \\
Names(recp) & = & Names(p)
\end{array}$$

Note how this is an instance of the general function scheme of definition 2.1.

Proposition A.9 *(Proposition 3.10)*

$$p \sqsubseteq_M^{''} q \;\; \mathit{iff} \;\; p \sqsubseteq_M^{'} q$$

Proof: For the ⇒-direction assume $p \sqsubseteq_M^{''} q$. We have to prove that $\forall \bar{r}.p[\bar{r}/\bar{x}] \sqsubseteq_M q[\bar{r}/\bar{x}]$ holds. This is established by proving that the relation

$$R = \{(p[\bar{r}/\bar{x}], q[\bar{r}/\bar{x}]) \mid p \sqsubseteq_M^{''} q\} \cup Id$$

is an M-bisimulation. To see that $R \subseteq M\mathcal{B}(R)$ observe that if $p[\bar{r}/\bar{x}] \xrightarrow{a} p'$ then

either $p \xrightarrow{a} p''$ and $p' \equiv p''[\bar{r}/\bar{x}]$. But then we know from the definition of R that $p \sqsubseteq^{''}_{M} q$ such that $q \xrightarrow{b} q''$ with $a \sqsubseteq_A b$ and $p'' \sqsubseteq^{''}_{M} q''$ .By property 3.2 $q[\bar{r}/\bar{x}] \xrightarrow{b} q''[\bar{r}/\bar{x}]$ which obviously is the matching move.

or there exists $x_i \in \bar{x}$ which is unguarded in p and $r_i \xrightarrow{a} r_i'$ and $p' \equiv r_i'$. Since $UG(p) = UG(q)$ by $p \sqsubseteq^{''}_{M} q$ then $q[\bar{r}/\bar{x}] \xrightarrow{a} r_i$ by property 3.3, which is the matching move.

If $q[\bar{r}/\bar{x}] \xrightarrow{a} q'$,$a \in M$ then

either $q \xrightarrow{a} q''$ and $q' \equiv q''[\bar{r}/\bar{x}]$. Then we have from the definition of R that $p \sqsubseteq^{''}_{M} q$ such that $p \xrightarrow{b} p''$ with $b \sqsubseteq_A a$ and $p'' \sqsubseteq^{''}_{M} q''$. By property 3.2 $p[\bar{r}/\bar{x}] \xrightarrow{b} p''[\bar{r}/\bar{x}]$, which obviously is the matching move.

or there exists $x_i \in \bar{x}$ which is unguarded in q and $r_i \xrightarrow{a} r_i'$ and $q' \equiv r_i'$. Since $UG(p) = UG(q)$ by $p \sqsubseteq^{''}_{M} q$ then $q[\bar{r}/\bar{x}] \xrightarrow{a} r_i$ by property 3.3, which obviously is the matching move.

The $\Leftarrow$-direction is proved by showing $p \not\sqsubseteq^{''}_{M} q \Rightarrow p \not\sqsubseteq^{'}_{M} q$.
Assume $p \not\sqsubseteq^{''}_{M} q$. Then

1. $UG(p) \neq UG(q)$

2. There exists an a such that $p \xrightarrow{a} p'$ but there exist no b and q' such that $a \sqsubseteq_A b$ and $q \xrightarrow{b} q'$ with $p \sqsubseteq^{''}_{M} q$

3. Or there exists $a \in M$ such that $q \xrightarrow{a} q'$ but there exist no b and p' such that $b \sqsubseteq_A a$ and $p \xrightarrow{b} p'$ with $p \sqsubseteq^{''}_{M} q$.

Assume 1. Then take $\bar{r} = (a_1.nil \ldots a_n.nil)$ where $a_i \notin Names(p) \cup Names(q)$ and $a \sqsubseteq_A a_i$ for all $a \in Names(q)$.
There must exist $x_i \in UG(p)$,$x_i \notin UG(q)$ or $x_i \in UG(q)$,$x_i \notin UG(p)$, in the first case $p[\bar{r}/\bar{x}] \xrightarrow{a_i} nil$ but $q[\bar{r}/\bar{x}] \overset{b}{\not\rightarrow}$ such that $a_i \sqsubseteq_A b$. The second case is similar.

Assume 2. Take $\bar{r} = \overline{nil} = (nil \ldots nil)$ then $p \sqsubseteq^{'}_{M} q$ cannot hold either, since the only actions of $p[\overline{nil}/\bar{x}]$ are the actions of p which q cannot match.

Assume 3. The argument as in 2 yields the desired result with p and q exchanged.

□

Proposition A.10 *(Proposition 3.11)*
$\sqsubseteq_M$ *is a precongruence with respect to the operators of* Σ^r*. i.e.* $\sqsubseteq_M$ *satisfies C1 and C2 of table 3.3*

PROOF: The relation

$$R_1 = \{(p_1[p_3/x], p_2[p_4/x]) \mid p_1 \sqsubseteq_M \ p_2, p_3 \sqsubseteq_M \ p_4\} \cup \sqsubseteq_M$$

is an M-bisimulation.
To see that $R_1 \subseteq M\mathcal{B}(R_1)$ observe that if $p_1[p_3/x] \xrightarrow{a} p'$ then by property 3.1

<u>either</u> $p_1 \xrightarrow{a} p_1'$ and $p' \equiv p_1'[p_3/x]$ but by definition of $R_1 : p_2 \xrightarrow{b} p_2'$ with $a \sqsubseteq_A \ b$ and $p_1' \sqsubseteq_M \ p_2'$ also by property 3.2 $p_2[p_4/x] \xrightarrow{b} p_2'[p_4/x]$ which is the matching move

<u>or</u> x is unguarded in p_1 and $p_3 \xrightarrow{a} p_3'$ and $p' \equiv p_3'$. But by definition of $R_1 : p_4 \xrightarrow{b} p_4'$ with $a \sqsubseteq_A \ b$ and $p_3' \sqsubseteq_M \ p_4'$, and also x is unguarded in p_2 since $p_1 \sqsubseteq_A \ p_2$. So by property 3.3 $p_2[p_4/x] \xrightarrow{b} p_4'$ which is the matching move.

Also if $p_2[p_4/x] \xrightarrow{a} p'$, $a \in M$ then by property 3.1

<u>either</u> $p_2 \xrightarrow{a} p_2'$, $a \in M$ and $p' \equiv p_2'[p_4/x]$. By the definition of $R_1 : p_1 \xrightarrow{b} p_1'$ with $b \sqsubseteq_A \ a$ and $p_1' \sqsubseteq_M \ p_2'$. Then similar arguments as above yield the desired result

<u>or</u> x is unguarded in p_2 and p_1, and similar arguments as above yield the desired result.

The relation

$$R_2 = \{(p[\mu x.p_1/x], p[\mu x.p_2/x]) \mid p_1 \sqsubseteq_M \ p_2, \ p, p_1, p_2 \in \Sigma^r, free(p) \subseteq \{x\}\} \cup \sqsubseteq_M$$

is an M-bisimulation, and the result follows by taking $p \equiv x$.
To see that $R_2 \subseteq M\mathcal{B}(R_2)$ we shall show by induction on the length of inferences that if

1. $p[\mu x.p_1/x] \xrightarrow{a} p'$ then there exist p'' and b such that $p[\mu x.p_2/x] \xrightarrow{b} p''$ with $a \sqsubseteq_A \ b$ and $(p', p'') \in R_2$
2. $p[\mu x.p_2/x] \xrightarrow{a} p'$, $a \in M$, then there exist p'' and b such that $p[\mu x.p_1/x] \xrightarrow{b} p''$ with $b \sqsubseteq_A \ a$ and $(p'', p') \in R_2$

We only prove case 2, the other being similar but simpler.
Consider the possible form of p:

$p = x$: Then $\mu x.p_2 \xrightarrow{a} p'$, $a \in M$, so by a shorter inference $p_2[\mu x.p_2/x] \xrightarrow{a} p'$, $a \in M$, and by induction $p_2[\mu x.p_1/x] \xrightarrow{b} p''$ with $(p'', p') \in R$ and $b \sqsubseteq_A a$. But $p_1 \sqsubseteq_M p_2$ by definition of R_2, so $p_1[\mu x.p_1/x] \sqsubseteq_M p_2[\mu x.p_1/x]$ by the first part of this proposition, so $[\mu x.p_1/x] \xrightarrow{c} p'''$ with $c \sqsubseteq_A b$, $b \in M$. Since M is downwardsclosed $(p''', p'') \in \sqsubseteq_M \subseteq R_2$.
By $REC \rightarrow$: $p[\mu x.p_1] \xrightarrow{c} p'''$ which obviously is the matching move.

$p = q_1 + q_2$: Then either $q_1[\mu x.p_2/x] \xrightarrow{a} p'$, $a \in M$ or $q_2[\mu x.p_2/x] \xrightarrow{a} p'$, $a \in M$ and hence by induction $q_1[\mu x.p_1/x] \xrightarrow{b} p''$ or $q_2[\mu x.p_2/x] \xrightarrow{b} p''$ with $b \sqsubseteq_A a$ and $(p'', p') \in R_2$.
By $SUM \rightarrow$: $p[\mu x.p_1/x] \xrightarrow{b} p''$ which is the matching move.

$p = c.q$: Then $a = c$ and $p' \equiv q[\mu x.p_2/x]$ and by induction $p'' = q[\mu x.p_1/x]$ and $(p'', p') \in R_2$.

$p = nil$: Since p has no free occurrences of x
$(nil[\mu x.p_1/x], nil[\mu x.p_2/x]) \in R_2$

$p = \mu y.q$: $y \neq x$. Then we have, by assumption, that $\mu y.(q[\mu x.p_2/x]) \xrightarrow{a} p'$, $a \in M$, so by a shorter inference we have $q[\mu x.p_2/x][p[\mu x.p_2/x]/y] \xrightarrow{a} p'$, $a \in M$, which by property 3.5 may be rewritten as $q[p/y][\mu x.p_2/x] \xrightarrow{a} p'$, $a \in M$. So by induction, applied to the expression $q[p/y]$, we know that $q[p/y][\mu x.p_1/x] \xrightarrow{b} p''$ with $b \sqsubseteq_A a$ and $(p'', p') \in R_2$. By using property 3.5 $q[\mu x.p_1/x][p[\mu x.p_1/x]/y] \xrightarrow{b} p''$ and by $REC \rightarrow$: $p[\mu x.p_1/x] \xrightarrow{b} p''$ which is the matching move.

□

Proposition A.11 *(Proposition 3.16)*
If y is not free in $\mu x.p$ then $\mu x.p \simeq_M \mu y.(p[y/x])$
i.e. $\sqsubseteq_M$ satisfies $R1$ of table 3.3

PROOF:

Let $p_1 = \mu x.p$ and $p_2 = \mu y.p[y/x]$. By proposition 3.15 $p_1 = p[p_1/x]$ and $p_2 = p[y/x][p_2/y]$.
We may now show that

$$R = \{(q[p_1/x], q[y/x][p_2/y]) \mid q \in \Sigma^r\}$$

is an M-bisimulation.
The result follows by taking $q = p$ and applying proposition 3.15.
To see that $R \subseteq M\mathcal{B}(R)$ we use induction on the number of inferences.
Observe that if $q[p_1/x] \xrightarrow{a} q_1$ then

<u>**either**</u> $q \xrightarrow{a} q'$ and $q_1 \equiv q'[p_1/x]$, but also $q[y/x][p_2/y] \xrightarrow{a} q_2 = q'[y/x][p_2/y]$ and $(q_1, q_2) \in R$

or x is unguarded in q and $p_1 \xrightarrow{a} q_1$ then $p[p_1/x] \xrightarrow{a} q_1$ and by shorter inference $p[y/x][p_2/y] \xrightarrow{b} q_2$ with $a \sqsubseteq_A b$ and $(q_1, q_2) \in R$.
By $REC\!\rightarrow$: $p_2 \xrightarrow{b} q_2$ and since x is unguarded in q then $q[y/x][p_2/y] \xrightarrow{b} q_2$ which is the matching move.

Also if $q[y/x][p_2/y] \xrightarrow{a} q_1$,$a \in M$, then

either $q \xrightarrow{a} q'$, $a \in M$, and the case is obvious

or x is unguarded in q. Then y becomes unguarded in $q[y/x]$ and $p_2 \xrightarrow{a} q_1$ then $p[y/x][p_2/y] \xrightarrow{a} q_1$ and by a shorter inference $p[p_1/x] \xrightarrow{b} q_2$ with $b \sqsubseteq_A a$ and $(q_1, q_2) \in R$.
By $REC\!\rightarrow$: $p_2 \xrightarrow{b} q_2$ and since x is unguarded in q then $q[p_1/x] \xrightarrow{b} q_2$ which is the matching move.

□

Proposition A.12 *(Proposition 3.17)*
If x is guarded in p_1 then
if $p_1[p_2/x] \sqsubseteq_M p_2$ then $\mu x.p_1 \sqsubseteq_M p_2$
and
if $p_2 \sqsubseteq_M p_1[p_2/x]$ then $p_2 \sqsubseteq_M \mu x.p_1$.
i.e. $\sqsubseteq_M$ satisfies R4 and R5 of table 3.3

Proof: Since $\sqsubseteq_M$ satisfies $\mu x.p \simeq_M p[\mu x.p/x]$ it is enough to show that if $p_1 \sqsubseteq_M p[p_1/x]$ & $p[p_2/x] \sqsubseteq_M p_2$ & $p_1 \sqsubseteq_M p_2$ then $p[p_1/x] \sqsubseteq_M p[p_2/x]$.
Let $R = \{q[p_1/x], q[p_2/x] \mid q \in \Sigma^r\}$.
We wish to show that R is an M-bisimulation upto '$\sqsubseteq_M$', and the result follows by taking $q = p$, and applying proposition 1.9.
We prove that $R \subseteq M\mathcal{B}(R)$.
For arbitrary q let $q[p_1/x] \xrightarrow{a} q_1$ then

either $q \xrightarrow{a} q'$ and $q_1 = q'[p_1/x]$ but also $q[p_2/x] \xrightarrow{a} q_2 = q'[p_2/x]$ and $(q_1, q_2) \in R$

or x is unguarded in q and $p_1 \xrightarrow{a} q_1$.
Then $p[p_1/x] \xrightarrow{b} q_1'$, $a \sqsubseteq_A b$ with $(q_1, q_1') \in \sqsubseteq_M$. Since x is guarded in p then $p \xrightarrow{b} p'$ and $q_1' = p'[p_1/x]$. Also $p[p_2/x] \xrightarrow{b} q_2 = p'[p_2/x]$ and $p_2 \xrightarrow{c} p_2'$, $b \sqsubseteq_A c$ with $(q_2, p_2') \in \sqsubseteq_M$.
But $(p'[p_1/x], p'[p_2/x]) = (q_1', q_2) \in R$ which implies $(q_1, p_2') \in \sqsubseteq_M \circ R \circ \sqsubseteq_M$.
Also $q[p_2/x] \xrightarrow{c} p_2'$ since x is unguarded in q.

Also if $q[p_2/x] \xrightarrow{a} q_1$, $a \in M$ then

either $q \xrightarrow{a} q'$ and $q_1 = q'[p_2/x]$ but also $q[p_1/x] \xrightarrow{a} q_2 = q'[p_1/x]$ and $(q_1, q_2) \in R$

or x is unguarded in q and $p_2 \xrightarrow{a} q_1$, $a \in M$.
Then $p[p_2/x] \xrightarrow{b} q_1'$, $b \sqsubseteq_A a$ with $(q_1', q_1) \in \sqsubseteq_M$. But since x is guarded in p then $p \xrightarrow{b} p'$ and $q_1' = p'[p_2/x]$. Also $p[p_1/x] \xrightarrow{b} q_2 = p'[p_1/x]$ and $p_1 \xrightarrow{c} p_1'$, $c \sqsubseteq_A b$, $b \in M$ since M is downwardsclosed with $(p_1', q_2) \in \sqsubseteq_M$.
But $(p'[p_1/x], p'[p_2/x]) = (q_2, q_1') \in R$ which implies
$(p_1', q_1) \in \sqsubseteq_M \circ R \circ \sqsubseteq_M$.
Also $q[p_1/x] \xrightarrow{c} p_1'$ since x is unguarded in q.

□

Proposition A.13 *(Proposition 3.18)*
$\mu x.p \simeq_M \mu x.(p+x)$ i.e. $\sqsubseteq_M$ satisfies R3 of table 3.3

Proof: The relation

$$R = \{(q[\mu x.p/x], q[\mu x.(p+x)/x]) \mid q \in T_{\Sigma^r}\}$$

is an M-bisimulation. The result follows by taking $q \equiv x$.

To see that $R \subseteq M\mathcal{B}(R)$ we use induction on the number of inferences. Observe that if $q[\mu x.p/x] \xrightarrow{a} q'$ then

either $q \xrightarrow{a} q''$ and $q' = q''[\mu x.p/x]$. By property 3.1 then also $q[\mu x.(p+x)/x] \xrightarrow{a} q''[\mu x.(p+x)/x]$ which is the matching move.

or x is unguarded in q and $\mu x.p \xrightarrow{a} q'$. But then $q' = p'[\mu x.p/x]$ and this is due to $p[\mu x.p/x] \xrightarrow{a} p'[\mu x.p/x]$ and by a shorter inference. Thus the induction hypothesis: $p[\mu x.(p+x)/x] \xrightarrow{b} p'[\mu x.(p+x)/x]$ with $a \sqsubseteq_A b$ and $(p'[\mu x.p/x], p'[\mu x.(p+x)/x]) \in R$. By $SUM \rightarrow$: $p[\mu x.(p+x)/x] + x[\mu x.(p+x)/x] \xrightarrow{b} p'[\mu x.(p+x)/x]$ which by the definition of substitution yields: $(p+x)[\mu x.(p+x)/x] \xrightarrow{b} p'[\mu x.(p+x)/x]$ and by $REC\rightarrow$:
$\mu x.(p+x) \xrightarrow{b} p'[\mu x.(p+x)/x]$. Since x is unguarded in q:
$q[\mu x.(p+x)/x] \xrightarrow{b} p'[\mu x.(p+x)/x]$ which obviously is the matching move.

Also if $q[\mu x.(p+x)/x] \xrightarrow{a} q'$, $a \in M$ then

either $q \xrightarrow{a} q''$ and the case is obvious.

or x is unguarded in q and $\mu x.(p+x) \xrightarrow{a} q'$ and this is due to $(p+x)[\mu x.(p+x)/x] \xrightarrow{a} p'[\mu x.(p+x)/x] = q'$ Then

either $p[\mu x.(p+x)/x] \xrightarrow{a} q'$

<u>**or**</u> $x[\mu x.(p+x)/x] \xrightarrow{a} q'$ i.e. $\mu x.(p+x) \xrightarrow{a} q'$

In the first case we have by induction that $p[\mu x.p/x] \xrightarrow{a} q'' = p''[\mu x.p/x]$ with $b \sqsubseteq_A a$ and $(q', q'') \in R$ and by $REC \rightarrow$: $\mu x.p \xrightarrow{a} q''$, and since x is unguarded in q then $q[\mu x.p/x] \xrightarrow{b} q''$ which is the matching move. In the second case we may unfold at most a finite number and then arrive at the first case.

Also R^{-1} may be shown to be an M-bisimulation by similar arguments. □

Lemma A.1 *(Lemma 3.2)*

1. *if $\bar{x}$ is not free in p then* $\vdash p[\bar{r}/\bar{x}] =_M p$
2. *if $\bar{x}$ and $\bar{y}$ are disjoint then* $\vdash p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] =_M p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$

PROOF:

1. if $\bar{x}$ is not free in p then $p[\bar{q}/\bar{x}] \equiv p$, by definition of substitution and by P1: $\vdash p =_M p$
2. We prove this by structural induction on p

$p \equiv nil$: Then by the definition of substitution $p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv nil$ and also $p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv nil$ and by P1: $\vdash nil =_M nil$

$p \equiv a.p'$: By the definition of substitution $(a.p')[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv a.(p'[\bar{q}/\bar{x}][\bar{r}/\bar{y}])$ also $(a.p')[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv a.(p'[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}])$. By induction: $\vdash p'[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p'[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ and by C1 with $p_1 = p_2 = a.z$ where z is not in $\bar{x}, \bar{y}, p', \bar{q}$ or $\bar{r}$ the result follows using the rules of substitution.

$p \equiv p_1 + p_2$: By the definition of substitution $(p_1 + p_2)[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p_1[\bar{q}/\bar{x}][\bar{r}/\bar{y}] + p_2[\bar{q}/\bar{x}][\bar{r}/\bar{y}]$. Also $(p_1 + p_2)[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv p_1[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] + p_2[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$. By induction $\vdash p_1[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p_1[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ and $\vdash p_2[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p_2[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ and by C1 with $p_1 = p_2 \equiv v + w$ where v and w are not in $\bar{x}, \bar{y}, p', \bar{q}$ or $\bar{r}$ the result follows using the rules of substitution.

$p \equiv v$: By the definition of substitution $v[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv q_i[\bar{r}/\bar{y}]$ if $v \equiv x_i$ or else $v[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv v[\bar{r}/\bar{y}]$ otherwise, which is r_i if $v \equiv y_i$ or v otherwise. Also $v[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv q_i[\bar{r}/\bar{y}]$ if $v \equiv x_i$ or else $v[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv v[\bar{r}/\bar{y}]$ otherwise, which is r_i if $v \equiv y_i$ or v otherwise. In all cases P1 yields the desired result.

$p \equiv \mu z.p$: If z is in $\bar{x}, \bar{y}$ or free in $\bar{q}$ or $\bar{r}$ we first use R1 to obtain $\vdash \mu z.p =_M \mu v.(p[v/z])$ for a completely new variable v and the details are as follows: $(\mu z.p)[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv \mu z.(p[\bar{q}/\bar{x}][\bar{r}/\bar{y}])$ also $(\mu z.p)[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}] \equiv \mu z.(p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}])$ and by induction: $\vdash p[\bar{q}/\bar{x}][\bar{r}/\bar{y}] \equiv p[\bar{q}[\bar{r}/\bar{y}]/\bar{x}, \bar{r}/\bar{y}]$ so by C2 the we obtain the desired result.

□

Theorem A.1 *(Theorem 3.8)*
For any expression p, with free variables in $\bar{y}$, there exist expressions $p_1 \ldots p_k$ $(k \geq 1)$ with free variables in $\bar{y}$ satisfying k equations:

$$\vdash p_i = \sum_{j=1}^{n_i} a_{ij}.p_{f(i,j)} + \sum_{j=1}^{m_i} y_{g(i,j)} \quad (k \geq 1)$$

moreover

$$\vdash p = p_1$$

Proof: By structural induction on p. □

$p \equiv nil$: Choose $m_i = 0$ and $n_i = 0$ then with $k = 1$: $\vdash p =_M nil$ by P1.

$p \equiv a.p'$: By induction there exist k equations

$$\vdash p'_i =_M \sum_{j=1}^{m_i} a_{ij} p'_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)} \quad (k \geq 1)$$

and $\vdash p' =_M p'_1$.
By C1 with $p_1 = p_2 \equiv a.z$ with z not in $\bar{p}'$ nor in $\bar{y}$ and $p_3 = p_4 \equiv p'_i$ we obtain k equations such that $\vdash p =_M p_1$ and p_i are on the desired form namely by taking $p_1 = a.p'_1$ and $p_i = a.p'_i$. Clearly the p_i's are on the desired form, also $\vdash p =_M p_1$ holds since $\vdash p' =_M p'_1$ and $a \sqsubseteq_A a$.

$p \equiv p_1 + p_2$: Now p_1 and p_2 have free variables in $\bar{y}$ so by induction we have $2k$ equations: $p^1_1 \ldots p^1_k$ and $p^2_1 \ldots p^2_k$ satisfying $2k$ equations:

$$\vdash p^1_i =_M \sum_{j=1}^{m^1_i} a^1_{ij}.p^1_{f_1(i,j)} + \sum_{j=1}^{n^1_i} y_{g^1(i,j)}$$

$$\vdash p^2_i =_M \sum_{j=1}^{m^2_i} a^2_{ij}.p^2_{f_2(i,j)} + \sum_{j=1}^{n^2_i} y_{g^2(i,j)}$$

By C1 twice with $p_1 = p_2 \equiv z^1 + z^2$ and z^1 and z^2 not in $\bar{p}^1$ nor in $\bar{p}^2$ or in $\bar{y}$ we obtain

$$\vdash p^1_i + p^2_i =_M \sum_{j=1}^{m^1_i} a^1_{ij}.p^1_{f_1(i,j)} + \sum_{j=1}^{n^1_i} y_{g^1(i,j)} + \sum_{j=1}^{m^2_i} a^2_{ij}.p^2_{f_2(i,j)} + \sum_{j=1}^{n^2_i} y_{g^2(i,j)}$$

Let $p_i \equiv p^1_i + p^2_i$. We obtain k equations which by rearrangement by S1 and S2 may be brought on the desired form, moreover $\vdash p =_M p_1$.

$p \equiv x$: As for *nil* we may choose $m_i = 0$ and $n_i = 0$ then with $k = 1$ we obtain $\vdash p =_M x$ by P1.

$p \equiv \mu x.q$: Now q has free variables in $(x, \bar{y})$, so by induction we have expressions $q_1 \ldots q_p$ satisfying k equations:

$$\vdash q_i =_M \sum_{j=1}^{m_i} a_{ij} q_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)} + [x] \quad (k \geq 1)$$

in each summand x may or may not occurre; also $q \equiv q_1$. Now set

$$\vdash q'_1 =_M \sum_{j=1}^{m_1} a_{1j}.q_{f(1,j)} + \sum_{j=1}^{n_1} y_{g(1,j)}$$

So that either $\vdash q =_M q'_1$ or $\vdash q =_M q'_1 + x$. It follows by R2 and R3 that

$$\vdash p =_M q'_1[p/x] \tag{A.2}$$

Now set

$$p_i \equiv q_1[p/x] \;\; (i \leq k)$$

Then by instantiation $[p/x]$ of the equation we obtain using (A.2) for any summand x,

$$\vdash p_i =_M \sum_{j=1}^{m_i} a_{ij}.p_{f(i,j)} + \sum_{j=1}^{n_i} y_{g(i,j)} + [\sum_{j=1}^{m_1} a_{1j}.p_{f(1,j)} + \sum_{j=1}^{n_1} y_{g(1,j)}]$$

which by S1 and S2 may be rearranged to the desired form. Moreover $\vdash p =_M p_1$ follows from $\vdash q =_M q_1$ and expressions p_i are easy seen to have free variables in $\bar{y}$.

□

Appendix B

In this appendix we present the full proofs of propositions, lemmas and theorems which have been presented but not proven in full in chapter 5 of the thesis.

The proofs are presented here because they are long, tedious and sometimes trivial.

Lemma B.1 *(Lemma 5.2)*
If $\mathcal{F}_c(p) \xrightarrow{A}_{CS} p''$ *then* $p'' = \mathcal{F}_c(p')$ *and* $p \xrightarrow{a}_{SS} p'$ *with* $a \in A$

Proof: We prove this by induction on the number of inferences used to obtain: $\mathcal{F}_c(p) \xrightarrow{A}_{CS} p''$. Observe the possible structure of p:

$p = nil$**:** The lemma holds trivially since $\mathcal{F}_c(nil) \overset{A}{\not\rightarrow}_{CS}$ for any $A \in \mathcal{P}(Act)$

$p = a.p_1$**:** If $\mathcal{F}_c(a.p_1) \xrightarrow{A}_{CS} p''$ then by $ACT{\rightarrow}_{CS}$ and definition of $\mathcal{F}_c$: $A = \{a\}$ and $p'' = \mathcal{F}_c(p_1)$ and by $ACT{\rightarrow}_{SS}$: $a.p_1 \xrightarrow{a}_{SS} p_1$.

$p = p_1 + p_2$**:** If $\mathcal{F}_c(p_1 + p_2) \xrightarrow{A}_{CS} p''$ then by $SUM{\rightarrow}_{CS}$ and definition of $\mathcal{F}_c$:

either $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ by a shorter inference so $p'' = \mathcal{F}_c(p_1')$ and $p_1 \xrightarrow{a}_{SS} p_1'$ with $a \in A$ and by $SUM{\rightarrow}_{CS}$: $\mathcal{F}_c(p_1 + p_2) \xrightarrow{A}_{CS} \mathcal{F}_c(p_1)$

or $\mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$ and the case is similar.

or both $\mathcal{F}_c(p_1 + p_2) \xrightarrow{B}_{CS} p''$ and $\mathcal{F}_c(p_2) \xrightarrow{C}_{CS} p''$ and $A = B \cup C$ by shorter inferences so $p'' = \mathcal{F}_c(p_1')$ and $p_1 \xrightarrow{b}_{SS} p_1'$ with $b \in B$ and so $p'' = \mathcal{F}_c(p_2')$ and $p_2 \xrightarrow{c}_{SS} p_2'$ with $c \in C$ but then $p'' = \mathcal{F}_c(p_1') = \mathcal{F}_c(p_2')$ and an easy argument by structural induction establishes that $\mathcal{F}_c(p_1') = \mathcal{F}_c(p_2') \Rightarrow p_1 = p_2$ so by $SUM{\rightarrow}_{CS}$: we arrive at the desired result.

$p = (p_1 \ldots p_n)[f]$**:** If $\mathcal{F}_c((p_1 \ldots p_n)[f]) \xrightarrow{C}_{CS} p''$ then $\forall i.\mathcal{F}_c(p_i) \xrightarrow{A_i}_{CS} p_i''$ and by shorter inferences so for all i: $p_i'' = \mathcal{F}_c(p_i')$ and $p_i \xrightarrow{a_i}_{SS} p_i'$ with $a_i \in A_i$.

Also $p'' = (p''_1 \dots p''_n)[\bar{f}]$ and by induction and the definition of $\mathcal{F}_c$: $p'' = \mathcal{F}_c((p'_1 \dots p'_n)[f])$.

$p = p_1 \mid_g p_2$**:** If $\mathcal{F}_c(p_1 \mid_g p_2) \xrightarrow{C}_{CS} p''$ then

either $\mathcal{F}_c(p_1) \xrightarrow{C}_{CS} p''$ and $p'' = p''_1 \mid_{\bar{g}} \mathcal{F}_c(p_2)$ by a shorter inference. So $p''_1 = \mathcal{F}_c(p'_1)$ and $p_1 \xrightarrow{c}_{SS} p'_1$ with $c \in C$. By $COM{\rightarrow}_{CS}$ and definition of $\mathcal{F}_c$: $p'' = \mathcal{F}_c(p'_1 \mid_g p_2)$

or $\mathcal{F}_c(p_2) \xrightarrow{C}_{CS} p''$ and the case is similar.

or $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ and $\mathcal{F}_c(p_2) \xrightarrow{B}_{CS} p''$ and $C = \bar{g}(A, B)$ by a shorter inference. So $p''_1 = \mathcal{F}_c(p'_1)$ and $p_1 \xrightarrow{a}_{SS} p'_1$ with $a \in A$ and $p''_2 = \mathcal{F}_c(p'_2)$ and $p_2 \xrightarrow{b}_{SS} p'_2$ with $b \in B$ but then $c = g(a, b) \in \bar{g}(A, B) = \{g(a, b) \mid a \in A,\ b \in B\}$ and by $COM{\rightarrow}_{SS}$: $p_1 \mid_g p_2 \xrightarrow{c}_{SS} p'_1 \mid_g p'_2$ which is the matching move.

$p = x$**:** The case holds trivially since: $\mathcal{F}_c(x) \overset{A}{\not\rightarrow}_{CS}$ for any $A \in \mathcal{P}(Act)$.

$p = \mu x.p_1$**:** If $\mathcal{F}_c(\mu x.p_1) \xrightarrow{A}_{CS} p''$ then by $REC{\rightarrow}_{CS}$: $\mathcal{F}_c(p_1)[\mu x.\mathcal{F}_c(p_1)/x] \xrightarrow{A}_{CS} p''$ by a shorter inference. So $p_1[\mu x.p_1/x] \xrightarrow{a}_{SS} p'$ with $a \in A$ and $p'' = \mathcal{F}_c(p')$. By $REC{\rightarrow}_{SS}$: $\mu x.p_1 \xrightarrow{a}_{SS} p'$ which is the matching move.

□

Proposition B.1 *(Proposition 5.1)*
p $\underline{corr}_\sigma$ $\mathcal{F}_c(p)$, p closed and finite.

Proof: The relation

$$R = \{(p, \mathcal{F}_c(p)) \mid p \in Pr^f_{SS}\}$$

is a correctness relation.

To see that $R \subseteq CORR_\sigma(R)$ we proceed by structural induction on p.

$p = nil$**:** Since $nil \overset{a}{\not\rightarrow}_{SS}$ for any $a \in Act$ and $nil \overset{A}{\not\rightarrow}_{CS}$ for any $A \in \mathcal{P}(Act)$: $(nil, \mathcal{F}_c(nil)) \in R$

$p = a.p'$**:** Since $a.p' \xrightarrow{a}_{SS} p'$ by $ACT{\rightarrow}_{SS}$ and $\{a\}.\mathcal{F}_c(p') \xrightarrow{\{a\}} \mathcal{F}_c(p')$ by $ACT{\rightarrow}_{CS}$ and $a \in \{a\}$ and $(p', \mathcal{F}_c(p')) \in R$ by induction, which proves that: $(a.p', \{a\}.\mathcal{F}_c(p')) \in R$.

$p = p_1 + p_2$**:** if $p_1 + p_2 \xrightarrow{a}_{SS} p'$ then either $p_1 \xrightarrow{a}_{SS} p'$ or $p_2 \xrightarrow{a}_{SS} p'$ by assumption $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ with $(p', p'') \in R$ and $a \in A$ or $\mathcal{F}_c(p_2) \xrightarrow{A}_{CS}$

p''. By $SUM \rightarrow_{CS}$: $\mathcal{F}_c(p_1) + \mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$ which obviously is the matching move.

Also if $\mathcal{F}_c(p_1 + p_2) \xrightarrow{A}_{CS} p''$ then either $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ or $\mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$ or $A = B \cup C$ and $\mathcal{F}_c(p_1) \xrightarrow{B}_{CS} p''$ and $\mathcal{F}_c(p_2) \xrightarrow{C}_{CS} p''$. By induction $p_1 \xrightarrow{a}_{SS} p'$ with $a \in A$ and $(p', p'') \in R$ or $p_2 \xrightarrow{a}_{SS} p'$ with $a \in A$ and $(p', p'') \in R$ or $p_1 \xrightarrow{b}_{SS} p'_1$ and $p_2 \xrightarrow{c}_{SS} p'_2$ with $b \in B$ and $(p'_1, p'') \in R$ and $c \in C$ and $(p'_2, p'') \in R$.

In all cases $p_1 + p_2 \xrightarrow{d}_{SS} p'$ with $(p', p'') \in R$ and $d \in A$ by $SUM \rightarrow_{SS}$, and clearly this is the matching move.

$p = (p_1 \ldots p_n)[f]$**:** if $(p_1 \ldots p_n)[f] \xrightarrow{c}_{SS} (p'_1 \ldots p'_n)[f]$ then $c = f(a_1 \ldots a_n)$ and $\forall i.p_i \xrightarrow{a_i}_{SS} p'_i$ By induction there exist A_i's and p''_i's such that $\forall i.\mathcal{F}_c(p_i) \xrightarrow{A_i} p''_i$ with $a_i \in A_i$ and $(p'_i, p''_i) \in R$.
By $FUN \rightarrow_{SS}$: $(\mathcal{F}_c(p_1) \ldots \mathcal{F}_c(p_n))[\bar{f}] \xrightarrow{C}_{CS} (p''_1 \ldots p''_n)[\bar{f}]$ with $C = \{f(a_1 \ldots a_n) \mid a_i \in A_i\}$ and $((p'_1 \ldots p'_n)[f], (p''_1 \ldots p''_n)[\bar{f}]) \in R$ which is the matching move.

Also if $(\mathcal{F}_c(p_1) \ldots \mathcal{F}_c(p_n))[\bar{f}] \xrightarrow{C}_{CS} (p''_1 \ldots p''_n)[\bar{f}]$ then $C = \{f(a_1 \ldots a_n) \mid a_i \in A_i\}$ and $\forall i.\mathcal{F}_c(p_i) \xrightarrow{A_i}_{CS} p''_i$ and by induction there exist a_i's and p_i's such that $\forall p_i \xrightarrow{a_i}_{SS} p'_i$ and $a_i \in A_i$ and $(p'_i, p''_i) \in R$ so by $FUN \rightarrow_{SS}$: $(p_1 \ldots p_n)[f] \xrightarrow{c}_{SS} (p'_1 \ldots p'_n)[f]$ which obviously is the matching move.

$p = p_1 \mid_g p_2$**:** if $p_1 \mid_g p_2 \xrightarrow{c} p'$ then

1. $p_1 \xrightarrow{c} p'_1$ & $p' = p'_1 \mid_g p_2$. Then by induction $\mathcal{F}_c(p_1) \xrightarrow{C} p''_1$ with $c \in C$ and $(p'_1, p''_1) \in R$. By $COM \rightarrow_{CS}$: $\mathcal{F}_c(p_1 \mid_g p_2) \xrightarrow{C} p''_1 \mid_{\bar{g}} \mathcal{F}_c(p_2)$. By lemma 5.2: $p''_1 = \mathcal{F}_c(p'_1)$ and the induction hypothesis: $(p_2, \mathcal{F}_c(p_2)) \in R$ so $p''_1 \mid_{\bar{g}} \mathcal{F}_c(p_2) = \mathcal{F}_c(p'_1) \mid_{\bar{g}} \mathcal{F}_c(p_2) = \mathcal{F}_c(p_1 \mid_g p_2)$ which yields the matching move.

2. $p_2 \xrightarrow{c} p'_2$ & $p' = p_1 \mid_g p'_2$. As in 1.

3. $p_1 \xrightarrow{a} p'_1$ & $p_2 \xrightarrow{b} p'_2$ & $p' = p'_1 \mid_g p'_2$ & $c \simeq g(a, b)$. By induction $\mathcal{F}_c(p_1) \xrightarrow{A} p''_1$ with $a \in A$ and by lemma 5.2: $p''_1 = \mathcal{F}_c(p'_1)$ and $\mathcal{F}_c(p_2) \xrightarrow{B} p''_2$ with $b \in B$ and by lemma 5.2: $p''_2 = \mathcal{F}_c(p'_2)$. By $COM \rightarrow_{CS}$: $\mathcal{F}_c(p_1 \mid_g p_2) = \mathcal{F}_c(p_1) \mid_{\bar{g}} \mathcal{F}_c(p_2) \xrightarrow{C} p''_1 \mid_{\bar{g}} p''_2 =$
$\mathcal{F}_c(p'_1 \mid_g p'_2)$ with $C = \bar{g}(A, B)$ so $c \in C$ and clearly this is the matching move.

Also if $\mathcal{F}_c(p_1 \mid_g p_2) \xrightarrow{C} p''$ similar arguments yield the result.

□

Proposition B.2 *(Proposition 5.2)*
p $\underline{corr}_\sigma$ $\mathcal{F}_c(p)$, p regular.

PROOF: The relation

$$R = \{(p, \mathcal{F}_c(p)) \mid p \in Pr^r_{SS}\}$$

is a correctness relation. To see that $R \subseteq CORR_\sigma(R)$ we proceed by induction on inferences. Observe the possible form of p:

$p = nil$**:** Since $nil \overset{a}{\not\rightarrow}_{SS}$ for any $a \in Act$ and $nil \overset{A}{\not\rightarrow}_{CS}$ for any $A \in \mathcal{P}(Act)$: $(nil, \mathcal{F}_c(nil)) \in R$ also $UG(nil) = \emptyset = UG(\mathcal{F}_c(nil))$

$p = p_1 + p_2$**:** if $p_1 + p_2 \xrightarrow{a}_{SS} p'$ then either $p_1 \xrightarrow{a}_{SS} p'$ or $p_2 \xrightarrow{a}_{SS} p'$ by a shorter inference $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ with $(p', p'') \in R$ and $a \in A$ or $\mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$. By $SUM{\rightarrow}_{CS}$: $\mathcal{F}_c(p_1) + \mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$ which obviously is the matching move.

Also if $\mathcal{F}_c(p_1 + p_2) \xrightarrow{A}_{CS} p''$ then either $\mathcal{F}_c(p_1) \xrightarrow{A}_{CS} p''$ or $\mathcal{F}_c(p_2) \xrightarrow{A}_{CS} p''$ or $A = B \cup C$ and $\mathcal{F}_c(p_1) \xrightarrow{B}_{CS} p''$ and $\mathcal{F}_c(p_2) \xrightarrow{C}_{CS} p''$. By induction $p_1 \xrightarrow{a}_{SS} p'$ with $a \in A$ and $(p', p'') \in R$ or $p_2 \xrightarrow{a}_{SS} p'$ with $a \in A$ and $(p', p'') \in R$ or $p_1 \xrightarrow{b}_{SS} p'_1$ and $p_2 \xrightarrow{c}_{SS} p'_2$ with $b \in B$ and $(p'_1, p'') \in R$ and $c \in C$ and $(p'_2, p'') \in R$.

In all cases $p_1 + p_2 \xrightarrow{d}_{SS} p'$ with $('p, p'') \in R$ and $d \in A$ by $SUM{\rightarrow}_{SS}$, and clearly this is the matching move. Also $UG(p_1 + p_2) = UG(p_1) + UG(p_2) = UG(\mathcal{F}_c(p_1 + p_2))$.

$p = x$**:** $x \overset{a}{\not\rightarrow}_{SS}$ for all $a \in Act$, $x \overset{A}{\not\rightarrow}_{CS}$ for all $A \in \mathcal{P}(Act)$ and $UG(x) = UG(\mathcal{F}_c(x))$.

$p = \mu x.p$**:** if $\mu x.p \xrightarrow{a}_{SS} p'$ then $p[\mu x.p/x] \xrightarrow{a} p'$ and by a shorter inference: $\mathcal{F}_c(p[\mu x.p/x]) \xrightarrow{A}_{CS} p''$ with $a \in A$ and $(p', p'') \in R$. By lemma 5.1: $\mathcal{F}_c(p[\mu x.p/x]) = \mathcal{F}_c(p).[\mu x.\mathcal{F}_c(p)/x]$ and by $REC{\rightarrow}_{CS}$: $\mu x.\mathcal{F}_c(p) \xrightarrow{A}_{CS} p''$ which is the matching move.

Also if $\mathcal{F}_c(\mu x.p) \xrightarrow{A}_{CS} p''$ by a shorter inference $\mathcal{F}_c(p)[\mu x.\mathcal{F}_c(p)/x] \xrightarrow{A}_{CS} p''$. Then by induction $p[\mu x.p/x] \xrightarrow{a}_{SS} p'$ with $a \in A$ and $(p', p'') \in R$, so by $REC{\rightarrow}_{SS}$: $\mu x.p \xrightarrow{a}_{SS} p'$ which is the matching move.

Also $UG(\mu x.p) = UG(\mathcal{F}_c(\mu x.p))$ since $UG(\mu x.p) = UG(p) \setminus \{x\}$ and $UG(\mathcal{F}_c(\mu x.p)) = UG(\mu x.\mathcal{F}_c(p)) = UG(\mathcal{F}_c(p)) \setminus \{x\}$ and by the induction hypothesis $UG(p) = UG(\mathcal{F}_c(p))$.

□

Proposition B.3 *(Proposition 5.3)*
$p\ \underline{safe}^{\sigma}_{\alpha}\ \mathcal{F}_a(p)$, *$p$ closed and finite.*

PROOF: The relation

$$R = \{(p, \mathcal{F}_a(p)) \mid p \in Pr^{f}_{CS}\}$$

is a safeness relation. To see that $R \subseteq SAFE^{\sigma}_{\alpha}(R)$ we proceed by structural induction. The only nontrivial cases are:

$p = p_1 \mid_h p_2$**:** If $p_1 \mid_h p_2 \xrightarrow{C} p'$ then

1. $p_1 \xrightarrow{c} p'_1$ & $p' = p'_1 \mid_h p'_2$. Then by induction $(p_i, \mathcal{F}_a(p_i)) \in R$ so $\mathcal{F}_a(p_1) \xrightarrow{l} p''_1$ with $\alpha(C) \sqsubseteq l$ and $p''_1 = \mathcal{F}_a(p'_1)$ by lemma 5.4. By $COM\!\rightarrow_{AS}$ and $\mathcal{F}_a(p_1 \mid_h p_2) \xrightarrow{l} p''_1 \mid_{\alpha \circ h \circ \gamma_2} \mathcal{F}_a(p_2) = \mathcal{F}_a(p'_1 \mid_h p_2)$ which is the matching move.

2. $p_2 \xrightarrow{C} p'_2$ & $p' = p_1 \mid_h p'_2$. As in 1.

3. $p_1 \xrightarrow{A} p'_1$ & $p_2 \xrightarrow{B} p'_2$ & $p' = p'_1 \mid_h p'_2$ & $c \simeq h(A, B)$.
   By definition of $R : \mathcal{F}_a(p_1) \xrightarrow{l_1} p''_1$ with $\alpha(A) \sqsubseteq l_1$ and $(p''_1, \mathcal{F}_a(p'_1)) \in R$ and $\mathcal{F}_a(p_2) \xrightarrow{l_2} p''_2$ with $\alpha(B) \sqsubseteq l_2$ and $(p'_2, \mathcal{F}_a(p'_2)) \in R$. Since $C = h(A, B) :\ \alpha(C) \sqsubseteq \alpha \circ h \circ \gamma_2(l_1, l_2)$ by the adjoinedness condition on (α, γ). By $COM\!\rightarrow_{SS}$: $\mathcal{F}_a(p_1 \mid_h p_2) \xrightarrow{l} p''_1 \mid_{\alpha \circ h \circ \gamma_2} p''_2 = \mathcal{F}_a(p'_1 \mid_h p'_2)$ where $l = \alpha \circ h \circ \gamma_2(l_1, l_2)$ which clearly is the matching move.

Also if $\mathcal{F}_a(p_1 \mid_h p_2) \xrightarrow{l} p''$ then

1. $p'' = p''_1 \mid_{\alpha \circ h \circ \gamma_2} \mathcal{F}_a(p_2)$ & $\mathcal{F}_a(p_1) \xrightarrow{l} p''_1$. By induction $(p_i, \mathcal{F}_a(p_i)) \in R$, $i \in \{1, 2\}$, so $p_1 \xrightarrow{C} p'_1$ with $C \subseteq \gamma(l)$ and $p''_1 = \mathcal{F}_a(p'_1)$ by lemma 5.4, so $(p'_1, \mathcal{F}_a(p'_1)) \in R$. By $COM\!\rightarrow_{CS}$: $p_1 \mid_h p_2 \xrightarrow{C} p'_1 \mid_h p_2$ which is the matching move.

2. $p'' = \mathcal{F}_a(p_1) \mid_{\alpha \circ h \circ \gamma_2} p''_2$ & $\mathcal{F}_a(p_2) \xrightarrow{l} p''_2$. As in 1.

3. $\mathcal{F}_a(p_1) \xrightarrow{l_1} p''_1$ & $\mathcal{F}_a(p_2) \xrightarrow{l_2} p''_2$ & $p'' = p''_1 \mid_{\alpha \circ h \circ \gamma_2} p''_2$ & $l \simeq \alpha \circ h \circ \gamma_2(l_1, l_2)$. By assumption $p_1 \xrightarrow{A} p'_1$ with $A \subseteq \gamma(l_1)$ and $p''_1 = \mathcal{F}_a(p'_1)$ by lemma 5.4, so $(p'_1, \mathcal{F}_a(p'_1)) \in R$ $p_2 \xrightarrow{B} p'_2$ with $B \subseteq \gamma(l_2)$ and $p''_2 = \mathcal{F}_a(p'_2)$ by lemma 5.4, so $(p'_2, \mathcal{F}_a(p'_2)) \in R$. By the adjoinedness condition on $(\alpha, \gamma) :\ C = h(A, B) \subseteq \gamma(\alpha \circ h \circ \gamma_2(l_1, l_2))$.
   By $COM\!\rightarrow_{CS}$: $p_1 \mid_h p_2 \xrightarrow{C} p'_1 \mid_h p'_2$ which is the matching move.

$p = (p_1 \ldots p_n)[g]$ If $(p_1 \ldots p_n)[g] \xrightarrow{C}_{CS} (p'_1 \ldots p'_n)[g]$ then $\forall i . p_i \xrightarrow{A}_{CS} p'_i$ and $C = g(A_1 \ldots A_n)$, so by induction $\forall i . \mathcal{F}_a(p_i) \xrightarrow{l_i}_{AS} p''_i$ with $\alpha(A) \sqsubseteq l_i$ and

by lemma 5.4: $p''_i = \mathcal{F}_a(p'_i)$ so $(p'_i, p''_i) \in R$.
By $FUN{\rightarrow}_{AS}$: $(\mathcal{F}_a(p_1)\ldots\mathcal{F}_a(p_n))[h] \xrightarrow{l}_{AS} (p''_1 \ldots p''_n)[h]$ where $h = \alpha \circ g \circ \gamma_n$ with $l = (\alpha \circ g \circ \gamma_n)(l_1 \ldots l_n)$ clearly $\alpha(g(A_1 \ldots A_n)) \sqsubseteq (\alpha \circ g \circ \gamma_n)(l_1 \ldots l_n)$ and also $((p'_1 \ldots p'_n)[g], (p''_1 \ldots p''_n)) \in R$.

Also if $(\mathcal{F}_a(p_1)\ldots\mathcal{F}_a(p_n))[h] \xrightarrow{l}_{AS} (p''_1 \ldots p''_n)[h]$ then $\forall i.\mathcal{F}_a(p_i) \xrightarrow{l_i}_{AS} p''_i$ and $l = h(l_1 \ldots l_n)$. So by induction $\forall i.p_i \xrightarrow{A_i}_{CS} p'_i$ with $A_i \subseteq \gamma(l_i)$ and $(p'_i, p''_i) \in R$. By $FUN{\rightarrow}_{CS}$: $(p_1 \ldots p_n)[g] \xrightarrow{A}_{CS} (p'_1 \ldots p'_n)[g]$ with $A \subseteq \gamma(l)$ since $A = g(A_1 \ldots A_n) \sqsubseteq \alpha \circ g \circ [\gamma \ldots \gamma](l_1 \ldots l_n)$. And this obviously is the matching move.

□