Structural and Behavioural Equivalences of Networks

Joachim Parrow

Swedish Institute of Computer Science, Kista, Sweden
and
Uppsala University, Sweden

December 1990 (Revised July 1991)

Footnote on first page:
This is a revised and extended version of a paper that appeared under the same title in the Proceedings of the 17th Int. Colloquium on Automata, Languages and Programming, Warwick University, July 1990; published as Springer Verlag Lecture Notes of Computer Science volume 443 pages 540–552, 1990.

Running title: Structural and Behavioural Equivalences of Networks

Address for proofs: Joachim Parrow, SICS, Box 1263, S-16428 Kista, Sweden.

Special symbols: Since this copy is mathematically type-set, only a few of the less obvious symbols are listed below.

$\mathbf{0}$, $\mathbf{C}$, $\mathbf{D}$, $\mathbf{I}$, $\mathbf{L}$, $\mathbf{M}$. $\mathbf{S}$, $\mathbf{R}$	Boldface symbols, note $\mathbf{0}$ is zero
$\mathcal{A}$, $\mathcal{B}$, $\mathcal{E}$, $\mathcal{I}$, $\mathcal{J}$, $\mathcal{L}$, $\mathcal{M}$, $\mathcal{N}$, $\mathcal{P}$, $\mathcal{V}$	Script style symbols
α, β, λ, θ, σ, τ, ω, Λ	Greek letters
$\{$, $\}$, $($, $)$, $[$, $]$, $\langle$, $\rangle$, $[\![$, $]\!]$	brackets
$\times$, $\cup$, $\cap$, $\subseteq$, $\in$, $\emptyset$	Set symbols (product, union, intersection, subset, member, emptyset)
$=$, $\neq$, $\leq$, $\geq$, $\simeq$, $\not\simeq$	Various equivalence symbols etc.
$\longrightarrow$, $\Longrightarrow$	Transition arrows
$\Rightarrow$, $\Leftarrow$	Implication arrows
$\rightarrow$	Function arrow
$\mapsto$	Maps-to arrow
$\stackrel{x\frown y}{\longleftrightarrow}$	Chaining symbol (x and y may vary)
$\frown$	Linking symbol as in $\langle a\frown b\rangle$
$\vdash$	Turnstile
$\#$	Arity symbol
$\circ$	Function composition
$\bullet$	Bullet
$\mid$	Parallel operator
$\square$	Used to signal end of a theorem, lemma, definition or proof

Note that the transition arrows will come in superscripted variants. The superscript should be on top of the arrow, as in

$$\stackrel{\sigma}{\Longrightarrow}$$

There are some unnumbered simple figures in the text, these function as displayed formulae and should *not* be relocated.

Abstract

We define an algebraic language for networks of synchronously communicating asynchronous processes. A node in the network may have several ports; a port is either external to the whole network or connected through a link to another port. The language contains two types of operations: parallel composition of two networks, and interlinking of two external ports within a network. We interpret this language in two ways: first we give a *structural* semantics, where terms are mapped to graphs representing the structure of networks, and second we give a *behavioural* semantics, where terms are mapped to behaviour schemas. A schema corresponds to a behaviour parameterised on the behaviours of the network nodes. These semantics give rise to structural and behavioural equivalences. We compare the equivalences and give sound and complete axiomatisations.

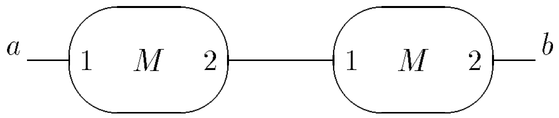
1 Introduction

Consider a network consisting of several nodes executing in parallel and synchronising over the links in the network. A description of such a network can be interpreted in two ways: either as a specification of the interconnection structure, or as a specification of the behaviour of the network. The former contains information about the nodes, ports, and links; the latter contains information about the states and external communication events. In this paper we will put forth a simple algebraic language for description of networks, and provide both a structural semantics and a behavioural semantics. The main interest will be to compare and axiomatise the equivalences induced by these semantics. Our companion paper [12] explores the expressive power of the same language.

We will be interested in a particular class of networks where each node may have several numbered *ports*. A port is either external to the network and has a unique name, or it is linked to another port in the network. A link connects exactly two ports and each port can be attached to at most one link. The nodes in a network will be of different *types*. There may be several occurrences of nodes of the same type in a network. For example, the network could be a piece of hardware where the nodes are logic circuits; there would then be one node type for each type of circuit. To conform with the terminology in our companion paper we will use the term *module* for "node type".

The structural semantics of the network is a graph which determines the modules of the nodes, the links between internal ports, and the names of external ports. As a simple example consider the following network $\mathcal{N}_1$:

$\mathcal{N}_1$:



$\mathcal{N}_1$ consists of two nodes labelled with the same module M and with two ports each. Port number 2 in the left hand node is linked to port number 1 in the right hand node; the remaining two ports are external with names a and b.

With a *behaviour* we will understand a rooted transition graph (not to be confused with the graph representing the structure of a network). A transition $q \xrightarrow{\alpha} q'$ in a behaviour means that from state q it is possible reach state q' through the action α. We assume that an *operational interpretation* assigns to each module a behaviour where the actions are sets of integers. The intended meaning is that the behaviour of a module determines the behaviour of the nodes labelled with that module, and that the actions represent synchronisation events on sets of ports. In a network, two interlinked ports may collaborate in an internal synchronisation involving both endpoints of the link. The actions of a network are sets of external port names; internal synchronisations are not counted in the actions. Continuing the example of $\mathcal{N}_1$ above, let the operational interpretation $\mathcal{I}_1$ assign the following behaviour to M:

$\mathcal{I}_1$:

$$\underset{M}{\bullet} \xrightarrow{\{1,2\}} \bullet$$

This transition graph means that M has one transition involving synchronisations on both ports. The behaviour of $\mathcal{N}_1$ in $\mathcal{I}_1$ is then the following, with the leftmost state as the root:

$$\bullet \xrightarrow{\{a,b\}} \bullet$$

Here $\mathcal{N}_1$ has one transition involving both external ports. The transition also involves a synchronisation over the link in the network, but this internal event is not represented in the action $\{a, b\}$. Consider next another interpretation $\mathcal{I}_2$ which gives another behaviour to M:

$\mathcal{I}_2$:

$$\underset{M}{\bullet} \xrightarrow{\{1\}} \bullet \xrightarrow{\{2\}} \bullet$$

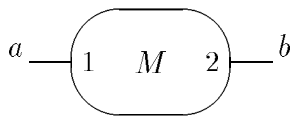
This transition graph means that M first synchronises on port number 1 and then on port number 2. The behaviour of $\mathcal{N}_1$ in $\mathcal{I}_2$ becomes:

$$\bullet \xrightarrow{\{a\}} \bullet \xrightarrow{\emptyset} \bullet \xrightarrow{\{b\}} \bullet$$

Intuitively, the leftmost node of $\mathcal{N}_1$ performs its first action on port 1 which is the external port a. Thereafter the two interlinked ports engage in an internal synchronisation built from the second action of the left node and the first action of the right node; this event does not involve any external port and is therefore labelled $\emptyset$. Finally, the right node performs its second action, involving the external port b.

As another and even simpler example consider the network $\mathcal{N}_2$ consisting of only one node:

$\mathcal{N}_2$:



In $\mathcal{I}_1$ this network has exactly the same behaviour as $\mathcal{N}_1$. In $\mathcal{I}_2$ the behaviour of $\mathcal{N}_2$ is:

$$\underset{M}{\bullet} \xrightarrow{\{a\}} \bullet \xrightarrow{\{b\}} \bullet$$

It may be argued that this behaviour is, from an external point of view, *equivalent* with the behaviour of $\mathcal{N}_1$. The only difference is the absence of the internal action (labelled $\emptyset$). Indeed, these two behaviours are identified by most of the behavioural equivalences (such as observation equivalence, failure equivalence, testing equivalence etc.) proposed to date.

Of course, the fact that two networks have equivalent behaviours in particular operational interpretations does not mean that they are equivalent in every interpretation. Consider again $\mathcal{N}_1$ and $\mathcal{N}_2$ in yet another interpretation $\mathcal{I}_3$:

$$\mathcal{I}_3 : \quad \underset{M}{\bullet} \xrightarrow{\{1\}} \bullet \xrightarrow{\{1\}} \bullet \xrightarrow{\{2\}} \bullet$$

$\mathcal{I}_3$ first requires two actions on port number 1 in M. The behaviour of $\mathcal{N}_2$ in $\mathcal{I}_3$ is

$$\bullet \xrightarrow{\{a\}} \bullet \xrightarrow{\{a\}} \bullet \xrightarrow{\{b\}} \bullet$$

but the behaviour of $\mathcal{N}_1$ in the same interpretation is

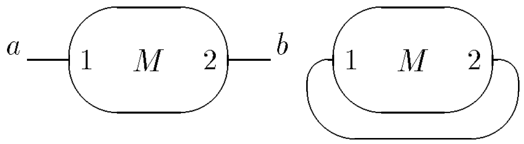
$$\bullet \xrightarrow{\{a\}} \bullet \xrightarrow{\{a\}} \bullet \xrightarrow{\emptyset} \bullet$$

This last behaviour lacks an external action $\{b\}$ so it cannot be equivalent with the behaviour of $\mathcal{N}_2$ for any reasonable definition of equivalence. In essence, after the internal action the left node of $\mathcal{N}_1$ has finished while the right node awaits a second event on port number 1. No further action is therefore possible.

In summary, the two networks $\mathcal{N}_1$ and $\mathcal{N}_2$ are different structurally, and also behaviourally in the sense that there exists a distinguishing interpretation (namely $\mathcal{I}_3$) which gives nonequivalent behaviours to the networks.

Clearly, two networks which are structurally isomorphic have the same behaviour regardless of the interpretation, but the converse does not hold. As an example, consider the network $\mathcal{N}_3$ which is like $\mathcal{N}_2$ with one additional "disconnected" node (no ports are external, and all links lead back to the node):

$\mathcal{N}_3$:



This network is certainly not isomorphic with $\mathcal{N}_2$. However, in *all* operational interpretations and for any reasonable definition of behavioural equivalence, $\mathcal{N}_2$ is equivalent with $\mathcal{N}_3$.

In this paper we will identify a class of "reasonable" behavioural equivalences. This class includes for example observation equivalence, branching bisimulation equivalence, and many forms of testing and failure equivalences. Although these equivalences are different it turns out that applied to networks (we consider two networks equivalent if and only if their behaviours are equivalent in all interpretations) they all coincide. Our main result is that for networks without isolated parts (an isolated part is a subset of the nodes without external ports and without links leading out of the subset, for example the singleton set containing the rightmost node of $\mathcal{N}_3$) the behavioural equivalences coincide with network isomorphism. In other words, two such networks with different topological structures can be made behaviourally unequal by choosing proper behaviours of the nodes.

We will derive our results within the framework of many-sorted algebra, thereby admitting an alternative characterisation of the equivalences through complete axiom systems. Thus we define a small algebraic language where networks can be built from atomic networks of type $M(a_1, \ldots, a_n)$ (containing exactly one node labelled M and external ports labelled $a_1, \ldots, a_n$) and two types of operations: a binary *parallel* operation $|$, where two networks are composed in parallel without any internetwork links, and a unary *linking* operation $\langle a \frown b \rangle$, where a link is formed between two ports named a and b in a network. For example, the networks $\mathcal{N}_1 - \mathcal{N}_3$ correspond to the following terms:

$$
\begin{aligned}
\mathcal{N}_1 &= (M(a,c) \,|\, M(d,b))\langle c \frown d \rangle \\
\mathcal{N}_2 &= M(a,b) \\
\mathcal{N}_3 &= (M(a,b) \,|\, M(c,d))\langle c \frown d \rangle
\end{aligned}
$$

This language will be interpreted both structurally as networks and behaviourally (via operational interpretations) as rooted transition graphs. We will completely axiomatise the structural equivalence which corresponds to isomorphism in the structural interpretation. We will also completely axiomatise behavioural equivalence, where two terms are regarded as behaviourally equivalent if their behaviours are equivalent in all operational interpretations of the modules. The difference between structural and behavioural equivalence is captured in a simple axiom schema corresponding to the removal of an isolated part.

Structural equivalence on a different class of networks has been investigated by Milner [8] (our modules correspond to the *generators* in Milner's article), and networks and operations on networks are prominent in work on data flow [5, 11, 6, 14, 4]. However, neither of these articles treat behavioural equivalences in the sense of this paper. Conversely, behavioural semantics and equivalences are prominent in algebras of processes such as CCS [10], SCCS [9], CSP [3], MEIJE [1], ACP [2], CIRCAL [7]. There, however, emphasis is given to languages richer than ours; in particular they contain primitives such as nondeterminism and sequential composition which can not readily be given structural interpretations. So, although both structural and behavioural equivalences have been investigated before, they have not been directly related.

In Section 2 we will describe our class of networks and the operations on networks. We present an axiomatisation which is sound and complete with respect to this model. In Section 3 we describe our class of behaviours, and the interpretation of the operations on behaviours. The concept of behavioural equivalence is formally defined, and we present our main result that this coincides with network isomorphism for networks without isolated parts. We finally show how the axiomatisation can be strengthened to become complete for behavioural equivalences. Section 4 contains some comments on an alternative formulation and ideas for further work. A companion paper [12] investigates the expressive power of the language with respect to the behavioural semantics and gives several examples and comparisons with other process algebras.

An earlier presentation of that work and of the work in the present paper can be found in a technical report [13].

2 Networks and Structural Equivalence

2.1 Networks

We here formally define the class of networks. We begin by assuming an infinite set of *port names* Λ ranged over by $a, b, \ldots$ and an infinite set of *modules* $\mathbf{M}$ ranged over by $M, M', \ldots$, where each module M has a nonnegative *arity* $\#M$ which determines the number of ports at instances of the module. Most concepts in this paper are dependent on $\mathbf{M}$ and Λ. For the sake of brevity such dependencies will be tacitly in force whenever appropriate, so we will simply say "network", "algebra" etc. instead of "($\mathbf{M}$, Λ)-network", "($\mathbf{M}$, Λ)-algebra" etc.

Definition 1 A *network* is a tuple $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L} \rangle$ where

$\mathcal{V}$, the set of *nodes*, is a nonempty finite set.

$\mathcal{M}$, the *module labelling*, is a function which assigns to each node in $\mathcal{V}$ a module in $\mathbf{M}$.

$\mathcal{E}$ is a finite set of *links*. Each link is a set of exactly two *ports*, where a port is a pair (v, n) such that $v \in \mathcal{V}$ and $1 \leq n \leq \#\mathcal{M}(v)$. A port can occur in at most one link in $\mathcal{E}$, and a port which occurs in a link is called *internal* to the network.

$\mathcal{L}$, the *port labelling*, is a partial injective function from ports to Λ; a port (v, n) is in the domain of $\mathcal{L}$ precisely if the port does not occur in any link in $\mathcal{E}$. Such a port is called *external* to the network. □

Thus, a port is either internal or external to a given network. Note that two nodes may be connected through several links, and that a link may connect two (distinct) ports at the same node. However, each port can be attached to at most one link. A node may have several external ports, and each external port has a unique (for the whole network) port name. Notice also that a module is not a network by itself: a module M can be regarded as having "formal port names" $1, \ldots, \#M$ which can be "instantiated" in networks to names in Λ. These formal port names serve only to distinguish between different internal ports, so no generality is lost by using natural numbers here. In other process algebras there is usually no difference in kind between internal and external port names, and a relabelling operation is used to instantiate networks. Our formulation is in some respects technically simpler; for example we will not need a relabelling operation (although it can easily be added as demonstrated in Section 4).

Definition 2 The *sort* of a network $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L} \rangle$ is the range of $\mathcal{L}$. We write $L(\mathcal{N})$ for the sort of a network $\mathcal{N}$. □

Thus, a sort is a finite subset of Λ; the sort of a network is exactly the set of its external port names. As an example, the network in Figure 1 has sort $\{a, b, c\}$ and can be defined as the tuple $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L} \rangle$ where

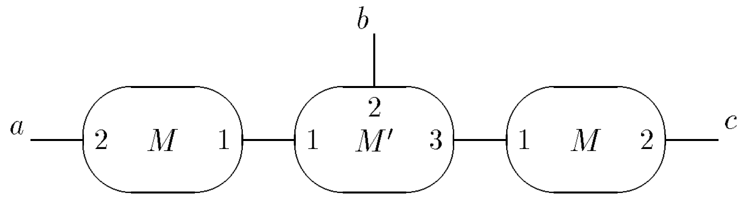


Figure 1: An example network.

$\mathcal{V} = \{v_1, v_2, v_3\}$,

$\mathcal{M}(v_1) = \mathcal{M}(v_3) = M;\;\; \mathcal{M}(v_2) = M'$,

$\mathcal{E} = \{\{(v_1, 1), (v_2, 1)\},\; \{(v_2, 3), (v_3, 1)\}\}$,

$\mathcal{L}(v_1, 2) = a,\;\; \mathcal{L}(v_2, 2) = b,\;\; \mathcal{L}(v_3, 2) = c.$

Two networks are called *isomorphic* if there is a bijection between the sets of nodes of the networks which preserves module labelling, links, and port labelling:

Definition 3 The networks $\mathcal{N} = \langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L}\rangle$ and $\mathcal{N}' = \langle \mathcal{V}', \mathcal{M}', \mathcal{E}', \mathcal{L}'\rangle$ are *isomorphic* if there is a bijection $h : \mathcal{V} \to \mathcal{V}'$ such that

1. $\mathcal{M}(v) = \mathcal{M}'(h(v))$
2. $\mathcal{L}(v, n)$ is defined iff $\mathcal{L}'(h(v), n)$ is defined, and if so $\mathcal{L}(v, n) = \mathcal{L}'(h(v), n)$
3. $\{(v_1, n_1), (v_2, n_2)\} \in \mathcal{E}$ iff $\{(h(v_1), n_1), (h(v_2), n_2)\} \in \mathcal{E}'$

□

In the following we will not distinguish between isomorphic networks; formally, we are interested only in the equivalence classes of networks under isomorphism, and we will write $\mathcal{N} = \mathcal{N}'$ to mean that $\mathcal{N}$ and $\mathcal{N}'$ are isomorphic.

2.2 An Algebra of Networks

We will use the following operations on networks:

Definition 4 If M is a module of arity n and $a_1, \ldots, a_n$ are distinct port names, then the *atomic network* $M(a_1, \ldots, a_n)$ is the network

$$\langle \{v\},\;\; [v \mapsto M],\;\; \emptyset,\;\; [(v, 1) \mapsto a_1,\; \ldots\; , (v, n) \mapsto a_n]\rangle \qquad \square$$

Thus, $M(a_1, \ldots, a_n)$ is a network consisting of a single node labelled M where all ports are external and labelled $a_1, \ldots, a_n$. Note that $L(M(a_1, \ldots, a_n)) = \{a_1, \ldots, a_n\}$.

Operation	Condition on formation	Sort
$M(a_1,\ldots,a_n)$	$n = \#M,\ \ i \neq j \Rightarrow a_i \neq a_j$	$\{a_1,\ldots,a_n\}$
$\mathcal{N}_1 \vert \mathcal{N}_2$	$L(\mathcal{N}_1) \cap L(\mathcal{N}_2) = \emptyset$	$L(\mathcal{N}_1) \cup L(\mathcal{N}_2)$
$\mathcal{N}\langle a{\frown}b\rangle$	$a \neq b,\ \ a,b \in L(\mathcal{N})$	$L(\mathcal{N}) - \{a,b\}$

Table 1: Summary of the operations on networks.

Definition 5 The unary *linking* operations $\langle a{\frown}b\rangle$ for $a,b \in \Lambda, a \neq b$ are defined as follows: if $\mathcal{N} = \langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L}\rangle$ is a network where $\mathcal{L}(v_1,n_1) = a$ and $\mathcal{L}(v_2,n_2) = b$, then

$$\mathcal{N}\langle a{\frown}b\rangle = \langle \mathcal{V},\ \ \mathcal{M},\ \ \mathcal{E} \cup \{\{(v_1,n_1),(v_2,n_2)\}\},\ \ \mathcal{L} - \{(v_1,n_1),(v_2,n_2)\}\rangle$$

where $\mathcal{L} - \{(v_1,n_1),(v_2,n_2)\}$ means the function gained by removing (v_1,n_1) and (v_2,n_2) from the domain of $\mathcal{L}$. □

Thus, the operation $\langle a{\frown}b\rangle$ adds a new link between the ports labelled a and b; these ports then become internal. It is trivial to verify that if $\mathcal{N}$ is a network with $a,b \in L(\mathcal{N})$ then $\mathcal{N}\langle a{\frown}b\rangle$ is indeed also a network, and that $L(\mathcal{N}\langle a{\frown}b\rangle) = L(\mathcal{N}) - \{a,b\}$.

Definition 6 The binary *parallel* operation $|$ on networks is defined as follows: if $\mathcal{N}_1 = \langle \mathcal{V}_1, \mathcal{M}_1, \mathcal{E}_1, \mathcal{L}_1\rangle$ and $\mathcal{N}_2 = \langle \mathcal{V}_2, \mathcal{M}_2, \mathcal{E}_2, \mathcal{L}_2\rangle$ are networks where $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ and the ranges of $\mathcal{L}_1$ and $\mathcal{L}_2$ are disjoint, then

$$\mathcal{N}_1|\mathcal{N}_2 = \langle \mathcal{V}_1 \cup \mathcal{V}_2,\ \mathcal{M}_1 \cup \mathcal{M}_2,\ \mathcal{E}_1 \cup \mathcal{E}_2,\ \mathcal{L}_1 \cup \mathcal{L}_2\rangle$$

where the union $f \cup g$ of two functions with disjoint domains is defined as usual by $(f \cup g)(x) = y$ if either $f(x) = y$ or $g(x) = y$. □

Thus, the parallel operation essentially builds the union of two networks, and does not create any internetwork links. Since we do not distinguish between isomorphic networks the condition $\mathcal{V}_1 \cap \mathcal{V}_2 = \emptyset$ is harmless, but note that the two networks cannot have any common external port names, i.e. $L(\mathcal{N}_1) \cap L(\mathcal{N}_2)$ must be $\emptyset$ for $\mathcal{N}_1|\mathcal{N}_2$ to be defined, and if so then $L(\mathcal{N}_1|\mathcal{N}_2) = L(\mathcal{N}_1) \cup L(\mathcal{N}_2)$.

The operations are summarised in Table 1. As an example, the network in Figure 1 can be described as

$$((M(d,a) \mid M'(e,b,f)) \mid M(g,c))\langle d{\frown}e\rangle\langle f{\frown}g\rangle$$

Another expression defining the same network is

C1	$A\|B$	$=$	$B\|A$	
C2	$(A\|B)\|C$	$=$	$A\|(B\|C)$	
L1	$A\langle a^\frown b\rangle$	$=$	$A\langle b^\frown a\rangle$	
L2	$A\langle a^\frown b\rangle\langle c^\frown d\rangle$	$=$	$A\langle c^\frown d\rangle\langle a^\frown b\rangle$	
D	$(A\|B)\langle a^\frown b\rangle$	$=$	$A\langle a^\frown b\rangle\|B$	$(a, b \notin L(B))$
S	$A\langle a^\frown b\rangle$	$=$	$A[c/a]\langle c^\frown b\rangle$	(c does not occur in A)

Table 2: Axioms $\mathcal{A}_s$. Here $A[c/a]$ represents syntactic substitution of c for a in A.

$((M(d,a) \mid M'(e,b,f))\langle d^\frown e\rangle \mid M(g,c))\langle f^\frown g\rangle$

We can now formally define a many-sorted algebra to reflect these operations (where sorts as before are finite subsets of Λ). The signature contains the following symbols: for each sort L and distinct names $a, b \in L$ there is a function symbol $\langle a^\frown b\rangle_L : L \to L - \{a, b\}$; for each pair of disjoint sorts L_1 and L_2 there is a function symbol $|_{L_1,L_2} : L_1 \times L_2 \to L_1 \cup L_2$; and for each module M and distinct names $a_1, \ldots, a_{\#M}$ there is a constant symbol $M(a_1, \ldots, a_{\#M})$ of sort $\{a_1, \ldots, a_{\#M}\}$. We use $A, B, \ldots$ to range over terms of this signature. For convenience we will drop the subscripts of $|$ and $\langle a^\frown b\rangle$ (the subscripts turn out to be unimportant or can be inferred from context); hence $\langle a^\frown b\rangle$ and $|$ are used both as operations on networks and as part of the formal syntax. The *network algebra* is obtained by interpreting the function and constant symbols into the carriers of sorted networks according to definitions 4–6; we write $\mathcal{N}[\![A]\!]$ for the interpretation of A in this algebra. It is straightforward to show that any network is denoted by a term.

2.3 Structural Equivalence

Say that two terms A and B are *structurally equivalent* if they denote the same network, i.e. $\mathcal{N}[\![A]\!] = \mathcal{N}[\![B]\!]$. The main result of this section is that structural equivalence is characterised by the axioms $\mathcal{A}_s$ in Table 2. Note that each line in this table represents an axiom schema, which generates axioms as the metavariables A, B, and C are instantiated to terms (for **D** and **S**, these terms must satisfy the side conditions).

The schemas **C1** and **C2** state that $|$ is commutative and associative. The schemas **L1** and **L2** state that the order of a and b is unimportant in $\langle a^\frown b\rangle$, and that linking is commutative. **D** is a distributive law: linking distributes over parallel, as long as the linked ports are in the same operand. **S** says that the names of linked ports can be arbitrarily chosen as long as distinct ports get distinct names. Here we require that c does not occur in A, i.e. that it is a fresh port name, and $A[c/a]$ is gained by replacing each occurrence of a in A with c.

Definition 7 Two terms A and B are *provably equivalent* from an axiom system $\mathcal{A}$, written $\mathcal{A} \vdash A = B$, if $A = B$ can be derived from the axioms $\mathcal{A}$ and equational reasoning (i.e. that $=$ is an equivalence and that a term can substitute an equivalent term). $\square$

The main result in this section is:

Theorem 1 $\mathcal{N}[\![A]\!] = \mathcal{N}[\![B]\!]$ iff $\mathcal{A}_s \vdash A = B$ $\square$

Proof: For the direction $\Leftarrow$ (soundness), it suffices to check that the operations on networks preserve isomorphism and that for any instance $C = D$ of an axiom schema in $\mathcal{A}_s$ it holds that $\mathcal{N}[\![C]\!] = \mathcal{N}[\![D]\!]$. This is a straightforward consequence of the definitions in Section 2.2.

For the direction $\Rightarrow$ (completeness), first prove that each term A is provably equivalent with a *normal form*: a term is a normal form if it is a sequence of linkings applied to a sequence of parallel compositions of atomic networks, i.e. in the form

$$(M_1(a_{1,1},\ldots,a_{1,\#M_1})|\cdots|M_n(a_{n,1},\ldots,a_{n,\#M_n}))\langle b_1{}^\frown c_1\rangle\cdots\langle b_k{}^\frown c_k\rangle$$

for some $n \geq 1$ and $k \geq 0$. It is clear that with **D**, possibly in conjunction with **S** to achieve unique names in linking operations, any linking operation can be propagated out of a parallel composition. By doing this repeatedly, a term can be rewritten to a term with all linking operations outside all parallel compositions. With **C2**, the resulting term can be rewritten to a term in normal form.

Now each node in the denotation of a term corresponds to a constant symbol in the term, and each link corresponds to a linking symbol. Thus two structurally equivalent normal forms may differ only in the relative order of the symbols and in the port names in the linkings. With **C1** and **C2** the atomic networks can be arbitrarily reordered; with **L1** and **L2** the symbols in the linkings can be reordered; finally with **S** linked names can be substituted by arbitrary unique names. Hence structurally equivalent normal forms are provably equivalent from $\mathcal{A}_s$. $\square$

One way to formulate our result is that the network algebra is initial in the class of algebras which satisfy $\mathcal{A}_s$. We will not pursue this topic here. A similar result is obtained by Milner [8]. Milner defines a different class of nets together with a set of operations, and proves a set of *laws of flow* to be sound and complete for these operations. These laws are slightly more complicated than ours, since in Milner's article a port may be connected to several links, and a link may not connect two ports at the same node.

3 Behaviours and Behavioural Equivalences

3.1 Behaviours

In the previous section a term was interpreted as a network, and equivalence between terms corresponded to isomorphism between networks. In this section we will instead define the *behaviour* of terms (and hence, implicitly, the behaviours of networks denoted by terms), and consider equivalences which identify terms with the "same" behaviour. We will do this in a way that has become standard for process algebras and related formalisms: through a definition of a family of so called transition relations $\xrightarrow{\alpha}$ over states.

The label α of a transition $q\xrightarrow{\alpha}q'$ is called the *action* of the transition. An action will be a set of port names, and we use $\alpha, \beta, \ldots$ to range over actions. The intended meaning of the transition $q\xrightarrow{\alpha}q'$ is that the state q can evolve to the state q' by participating in synchronisations on all ports in α. There may be several different transitions from a state with the same action;

this amounts to nondeterminism. Note that the empty set $\emptyset$ is also an action, this corresponds to an internal action (cf. τ in CCS). [1]

We first define the general concepts of transition graph and behaviour:

Definition 8 A *transition graph* is a tuple $\langle Q,\ L,\ \longrightarrow\rangle$ where

Q is a nonempty set, the set of *states*,

L is a set, the *sort* of the transition graph,

$\longrightarrow \subseteq Q \times \mathcal{P}(L) \times Q$ is called the family of *transition relations* ($\mathcal{P}(L)$ denotes the powerset of L). $\square$

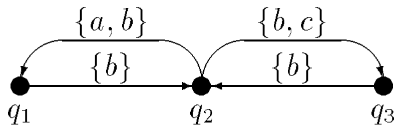
The set of states in a transition graph is not necessarily finite. We write $q \xrightarrow{\alpha} q'$ for $(q, \alpha, q') \in \longrightarrow$.

Definition 9 A *behaviour* is a pair (T, q^0) where T is a transition graph and q^0 is a state in T, called the *initial* state of the behaviour. The *sort* $L(T, q^0)$ of the behaviour is the sort of T. $\square$

In the following we will use $P, P' \ldots$ to range over behaviours. If $P = (T, q)$ and $P' = (T, q')$ we will write $P \xrightarrow{\alpha} P'$ to mean that $q \xrightarrow{\alpha} q'$ is a transition in T. As an example of a transition graph, consider

$$T = \langle \{q_1, q_2, q_3\},\quad \{a, b, c\},\quad \{q_1 \xrightarrow{\{b\}} q_2,\ q_3 \xrightarrow{\{b\}} q_2,\ q_2 \xrightarrow{\{a,b\}} q_1,\ q_2 \xrightarrow{\{b,c\}} q_3\}\rangle$$

The behaviour (T, q_2) can initially do either $\{a, b\}$ or $\{b, c\}$ and then continue as (T, q_1) or (T, q_3). Both these can initially do $\{b\}$ and continue as (T, q_2). The transition graph T can pictorially be drawn as follows:



Definition 10 Two behaviours (T, q) and (T', q') are *isomorphic* if they have the same sort, and there is a bijection h from the set of states in T to the set of states in T' such that $h(q) = q'$ and which preserves transitions (i.e. $q_1 \xrightarrow{\alpha} q_2$ is a transition in T iff $h(q_1) \xrightarrow{\alpha} h(q_2)$ is a transition in T'). $\square$

In the following we will not distinguish between isomorphic behaviours; formally, we are interested only in the equivalence classes of behaviours under isomorphism, and we will write $P = P'$ to mean that the behaviours P and P' are isomorphic.

[1]The structuring of actions as sets of names is used in CIRCAL but is otherwise not standard in process algebra; in CCS and CSP an action can contain at most one port name, and in e.g. SCCS actions form multisets rather than sets.

3.2 Algebras of Behaviours

We will here interpret terms into behaviours. The behaviour of a term will be determined by an *operational interpretation* which assigns behaviours to the modules. Thus a module of arity n can be regarded as a variable, or placeholder, and an operational interpretation assigns to it a concrete behaviour which will be of sort $\{1, \ldots, n\}$. The behaviour of $M(a_1, \ldots, a_n)$ will be obtained by replacing i by a_i in the behaviour of M, and parallel and linking will be interpreted as operations on behaviours.

Definition 11 An *operational interpretation* $\mathcal{I}$ on $\mathbf{M}$ is a function which assigns to each n-ary module M in $\mathbf{M}$ a behaviour of sort $\{1, \ldots, n\}$. $\square$

Definition 12 Let $P = (\langle Q,\ \{1, \ldots, n\},\ \longrightarrow\rangle,\ q^0)$. The behaviour $P(a_1, \ldots, a_n)$ is defined to be

$$(\langle Q,\ \{a_1, \ldots, a_n\},\ \longrightarrow'\rangle,\ q^0)$$

where $\longrightarrow'$ is obtained from $\longrightarrow$ by replacing all occurrences of all i ($1 \leq i \leq n$) by a_i. $\square$

Definition 13 Let $P_i = (\langle Q_i,\ L_i,\ \longrightarrow_i\rangle,\ q_i^0)$ for $i = 1, 2$ be two behaviours with disjoint sorts. The *parallel composition* $P_1|P_2$ is the behaviour

$$(\langle Q_1 \times Q_2,\ L_1 \cup L_2,\ \longrightarrow\rangle,\ (q_1^0, q_2^0))$$

where $\longrightarrow$ is defined as the least relation satisfying the following rules:

$$\frac{q_1 \xrightarrow{\alpha}_1 q_1'}{(q_1, q_2) \xrightarrow{\alpha} (q_1', q_2)} \qquad \frac{q_2 \xrightarrow{\alpha}_2 q_2'}{(q_1, q_2) \xrightarrow{\alpha} (q_1, q_2')} \qquad \frac{q_1 \xrightarrow{\alpha}_1 q_1' \quad q_2 \xrightarrow{\beta}_2 q_2'}{(q_1, q_2) \xrightarrow{\alpha \cup \beta} (q_1', q_2')}$$

$\square$

The first and second rules say that $P_1|P_2$ can do whatever P_1 or P_2 can do in isolation. The third rule says that if both P_1 and P_2 can do something, then $P_1|P_2$ can do the union of the actions. The parallel operation expresses a form of "independent" parallelism — P_1 and P_2 execute asynchronously side by side without affecting each other.

Definition 14 Let $P_1 = (\langle Q_1,\ L_1,\ \longrightarrow_1\rangle,\ q_1^0)$, and let $a, b \in L_1$ and $a \neq b$. The *linking* $P_1\langle a \frown b\rangle$ is the behaviour

$$(\langle Q_1,\ L_1 - \{a, b\},\ \longrightarrow\rangle,\ q_1^0)$$

where $\longrightarrow$ is defined by the following rules:

$$\frac{q \xrightarrow{\alpha}_1 q' \quad a, b \notin \alpha}{q \xrightarrow{\alpha} q'} \qquad \frac{q \xrightarrow{\alpha}_1 q' \quad a, b \in \alpha}{q \xrightarrow{\alpha - \{a,b\}} q'}$$

$\square$

For the linking operation, the intuition is that if P can do an action involving neither a nor b, then $P\langle a\frown b\rangle$ can do the same action (the link has no effect at all). If P can do an action involving both a and b, then $P\langle a\frown b\rangle$ can do the same action, now with a and b removed (intuitively this action involves a synchronisation over the link). As a consequence, actions of P involving exactly one of a and b are disallowed in $P\langle a\frown b\rangle$ (such actions would correspond to synchronisations where only one endpoint of the link is involved). While the parallel operation is "independent" parallelism, the linking operation is used to explicitly express dependencies: actions involving different endpoints of the link must be synchronised.

It will be technically convenient to regard modules as states; we can then represent an operational interpretation $\mathcal{I}$ as a transition graph

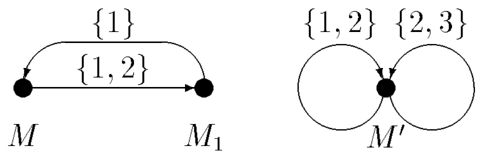
$$T_{\mathcal{I}} \;=\; \langle \mathbf{M},\, \omega,\, \longrightarrow\rangle$$

A transition $M\xrightarrow{K}M'$ in $T_{\mathcal{I}}$ means that according to $\mathcal{I}$, the behaviour of M can evolve to the state, or module, M' by participating in synchronisations on the ports in K. In this case M' must have the same arity as M: a module can not change its number of ports when it executes. Also, K will then be a subset of $\{1,\ldots,\#M\}$. The behaviour of M in $\mathcal{I}$ is simply obtained from $T_{\mathcal{I}}$ by choosing M itself as the initial state, in other words $\mathcal{I}(M) = (T_{\mathcal{I}}, M)$. Where no confusion is possible we will not distinguish between an operational interpretation $\mathcal{I}$ and its representation $T_{\mathcal{I}}$.

As an example, let an operational interpretation contain the following four transitions:

$$M\xrightarrow{\{1,2\}}M_1,\;\; M_1\xrightarrow{\{1\}}M,\;\; M'\xrightarrow{\{1,2\}}M',\;\; M'\xrightarrow{\{3,2\}}M'$$

This operational interpretation may be drawn as follows:



The reader may want to check the behaviour of

$$((M(d,a)\,|\,M'(e,b,f))\,|\,M(g,c))\langle d\frown e\rangle\langle f\frown g\rangle$$

(cf. Figure 1) under this operational interpretation; the reachable states of the behaviour correspond to the example in Section 3.1. In our companion paper [12] we give several other examples.

Given a fixed operational interpretation $\mathcal{I}$ we define a *behaviour algebra*, using the signature of Section 2, where the carrier of sort L is the class of behaviours of sort L and where the function symbols are interpreted according to definitions 12–14 above. We let $\mathcal{B}_{\mathcal{I}}[\![\bullet]\!]$ stand for the interpretation of terms into this algebra in the obvious way; i.e.

$$\begin{array}{lcl}
\mathcal{B}_{\mathcal{I}}[\![M(a_1,\ldots,a_n)]\!] & = & (\mathcal{I}(M))(a_1,\ldots,a_n)\\
\mathcal{B}_{\mathcal{I}}[\![A|B]\!] & = & \mathcal{B}_{\mathcal{I}}[\![A]\!]\,|\,\mathcal{B}_{\mathcal{I}}[\![B]\!]\\
\mathcal{B}_{\mathcal{I}}[\![A\langle a\frown b\rangle]\!] & = & (\mathcal{B}_{\mathcal{I}}[\![A]\!])\langle a\frown b\rangle
\end{array}$$

Note that the symbols $|$ and $\langle a\frown b\rangle$ are now used both in the formal syntax and as operations on networks and on behaviours. The role of these symbols will always be clear from context.

We define $\mathcal{B}[\![A]\!]$ as $\lambda\mathcal{I}.\mathcal{B}_{\mathcal{I}}[\![A]\!]$ and call it the *schematic behaviour* of A. So, for a term A, its schematic behaviour maps an operational interpretation of modules to the behaviour of A in that interpretation. The reader may want to regard $\mathcal{B}[\![\bullet]\!]$ as a map from terms into an algebra of schematic behaviours, or alternatively regard $\mathbf{M}$ as a set of variables.

We can now prove the claim that $\mathcal{B}[\![\bullet]\!]$ determines the schematic behaviour of a network uniquely:

Theorem 2 If $\mathcal{N}[\![A]\!] = \mathcal{N}[\![B]\!]$, then $\mathcal{B}[\![A]\!] = \mathcal{B}[\![B]\!]$. $\Box$

Proof: The idea is to use the axiomatisation $\mathcal{A}_s$. First fix an arbitrary operational interpretation $\mathcal{I}$. Assume $\mathcal{N}[\![A]\!] = \mathcal{N}[\![B]\!]$. Then $\mathcal{A}_s \vdash A = B$ by Theorem 1. We must prove that there is a bijection h relating the states in $\mathcal{B}_{\mathcal{I}}[\![A]\!]$ and $\mathcal{B}_{\mathcal{I}}[\![B]\!]$ and which preserves transitions, i.e. that $q \xrightarrow{\alpha} q'$ is a transition in $\mathcal{B}_{\mathcal{I}}[\![A]\!]$ iff $h(q) \xrightarrow{\alpha} h(q')$ is a transition in $\mathcal{B}_{\mathcal{I}}[\![B]\!]$. This is proven by induction on the length of the inference of $A = B$ in $\mathcal{A}_s$. The proof is a bit tedious but all cases are straightforward. We give here only two sample cases.

For the first case, assume that the last step in the inference used an instance of **C1**. Then $A = A_1|A_2$ and $B = A_2'|A_1'$, where $A_1 = A_1'$ and $A_2 = A_2'$ are proven by shorter inferences. Let the behaviours of A_1, A_2, A_1', and A_2' be P_1, P_2, P_1', and P_2' respectively. By induction there are bijections h_1 and h_2 which preserve transitions between the states of P_1 and P_1' and between the states of P_2 and P_2' respectively. The required bijection h is then the function which takes any state (q_1, q_2) to $h_2(q_2)|h_1(q_1)$. The fact that this h preserves transitions follows from Definition 13 and the fact that h_1 and h_2 preserve transitions.

For the second case assume that the last step in the inference used equational reasoning, namely $A = A_1\langle a\frown b\rangle$, $B = B_1\langle a\frown b\rangle$ and $A_1 = B_1$ is proven by a shorter inference. Let the behaviours of A_1 and B_1 be P_1 and P_1' respectively. By induction there is a bijection h_1 between the states of P_1 and P_1'. The required bijection h is then the same as h_1. $\Box$

In view of this result we can unambiguously talk about *the* behaviour $\mathcal{B}_{\mathcal{I}}(\mathcal{N})$ of a network $\mathcal{N}$; this behaviour is defined as $\mathcal{B}_{\mathcal{I}}[\![A]\!]$ for some term A such that $\mathcal{N} = \mathcal{N}[\![A]\!]$. Similarly the schematic behaviour $\mathcal{B}(\mathcal{N})$ is defined as $\lambda\mathcal{I}.\mathcal{B}_{\mathcal{I}}(\mathcal{N})$. When $\mathcal{I}$ is understood from context we will write $\mathcal{N} \xrightarrow{\alpha} \mathcal{N}'$ for $\mathcal{B}_{\mathcal{I}}(\mathcal{N}) \xrightarrow{\alpha} \mathcal{B}_{\mathcal{I}}(\mathcal{N}')$.

It may be informative to compare the behaviour algebras with other process algebras. First, we use instantiation of modules (rather than relabelling) and linking (rather than restriction and hiding). See our paper [12] for an investigation of these differences and of the expressive power of the behaviour algebras. Second, we define the semantics through an interpretation of terms into behaviours rather than giving a family of transition relations directly on terms. There is however only a superficial difference between these approaches; we could instead have given a family of transition relations as in Table 3 (given here only as an illustration). For any fixed $\mathcal{I}$ the relations obtained in this way satisfies $A \xrightarrow{\alpha} B$ iff $\mathcal{B}_{\mathcal{I}}[\![A]\!] \xrightarrow{\alpha} \mathcal{B}_{\mathcal{I}}[\![B]\!]$. Third, our behaviour algebras are parameterised on an operational interpretation. The reason for this is that we want to derive an axiomatisation which is valid for *all* such interpretations. In a sense our modules act as variables, and the axiomatisation will be sound for an arbitrary assignment of behaviours to these variables.

$$\frac{M \xrightarrow{K} M' \in T_{\mathcal{I}} \qquad \alpha = \{a_i : i \in K\}}{M(a_1, \ldots, a_n) \xrightarrow{\alpha} M'(a_1, \ldots, a_n)}$$

$$\frac{A \xrightarrow{\alpha} A'}{A|B \xrightarrow{\alpha} A'|B} \qquad \frac{B \xrightarrow{\alpha} B'}{A|B \xrightarrow{\alpha} A|B'} \qquad \frac{A \xrightarrow{\alpha} A' \quad B \xrightarrow{\beta} B'}{A|B \xrightarrow{\alpha \cup \beta} A'|B'}$$

$$\frac{A \xrightarrow{\alpha} A' \qquad a, b \notin \alpha}{A\langle a \frown b \rangle \xrightarrow{\alpha} A'\langle a \frown b \rangle} \qquad \frac{A \xrightarrow{\alpha} A' \quad a, b \in \alpha}{A\langle a \frown b \rangle \xrightarrow{\alpha - \{a,b\}} A'\langle a \frown b \rangle}$$

Table 3: Plotkin-style transitional semantics corresponding to the behaviour algebra.

3.3 Behavioural Equivalences

Intuitively, two behaviours can be regarded as "equivalent" if they are indistinguishable through experiments (sequences of transitions) where empty actions are insignificant. Many such equivalences have been proposed in the literature (branching bisimulation, observation equivalence, failure equivalence, testing equivalence, etc.). Our results will hold for most such equivalences which are congruences, ignore divergence (infinite internal computation), and respect termination properties such as deadlocks.

We use σ to range over sequences of non-empty actions, and write $\langle\rangle$ for the empty sequence.

Definition 15

1. $P \stackrel{\alpha}{\Longrightarrow} P'$ if $\begin{cases} \exists n \geq 0 : \ P \underbrace{\xrightarrow{\emptyset} \cdots \xrightarrow{\emptyset}}_{n \text{ times}} P' & \text{if } \alpha = \emptyset \\ P \stackrel{\emptyset}{\Longrightarrow} \xrightarrow{\alpha} \stackrel{\emptyset}{\Longrightarrow} P' & \text{if } \alpha \neq \emptyset \end{cases}$

2. $P \stackrel{\langle\rangle}{\Longrightarrow} P'$ if $P \stackrel{\emptyset}{\Longrightarrow} P'$, and $P \stackrel{\alpha_1 \ldots \alpha_n}{\Longrightarrow} A'$ if $P \stackrel{\alpha_1}{\Longrightarrow} \cdots \stackrel{\alpha_n}{\Longrightarrow} P'$.

3. P is *terminated* if not $P \stackrel{\alpha}{\Longrightarrow} P'$ for any P' and $\alpha \neq \emptyset$.

4. $P \stackrel{\sigma}{\Longrightarrow} \bullet$ (σ is a *completed trace* of P) if $P \stackrel{\sigma}{\Longrightarrow} P'$ for some terminated P'.

5. An equivalence $\simeq$ on behaviours is a *behavioural equivalence* if it satisfies the following three conditions:

   (a) It is a congruence in the algebra of behaviours, i.e. $P \simeq P'$ implies

      i. $L(P) = L(P')$,

ii. $P|Q \simeq P'|Q$ and $Q|P \simeq Q|P'$ for all Q with $L(Q) \cap L(P) = \emptyset$,

iii. $P\langle a^\frown b\rangle \simeq P'\langle a^\frown b\rangle$ for all $a, b \in L(P)$;

(b) $P|Q \simeq P$ if Q is terminated;

(c) if $P \simeq Q$ and $P \stackrel{\sigma}{\Longrightarrow} \bullet$ then $Q \stackrel{\sigma}{\Longrightarrow} \bullet$. □

Note that if a behaviour is terminated then the only actions it can ever do is the empty action. Such a behaviour may still *diverge* by engaging in an infinite sequence of empty actions. We will however only consider equivalences which take no account of divergence (thereby disqualifying some forms of testing and failure equivalence). Requirement 5 (b) means that an interleaving of empty actions is insignificant; this requirement is satisfied by most equivalences which claim to "ignore" internal actions. Requirement 5 (c) means that $\simeq$ must be sensitive to completed traces (consisting of non-empty actions).

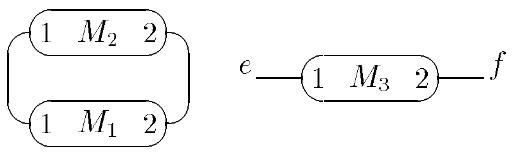
We write $\mathcal{B}[\![A]\!] \simeq \mathcal{B}[\![B]\!]$ (A and B are *behaviourally $\simeq$-equivalent*) to mean that $\mathcal{B}_\mathcal{I}[\![A]\!] \simeq \mathcal{B}_\mathcal{I}[\![B]\!]$ for all operational interpretations $\mathcal{I}$.

3.4 Comparing Structural and Behavioural Equivalences

Our first main result in this paper is that structural and behavioural equivalences coincide for a certain class of networks. From Theorem 2 we immediately get that isomorphic networks have equivalent behaviours but unfortunately the converse is not true in general. The reason is that networks may have isolated parts: sets of nodes with no possibility to ever engage in a non-empty action. Consider as an example

$$A \quad = \quad (M_1(a,b)|M_2(c,d)|M_3(e,f))\langle a^\frown c\rangle\langle b^\frown d\rangle$$

The network $\mathcal{N}[\![A]\!]$ is:



Now by using the law **D**, the term A can be rewritten to $A'|M_3(e,f)$ where

$$A' \quad = \quad (M_1(a,b)|M_2(c,d))\langle a^\frown c\rangle\langle b^\frown d\rangle$$

Here $L(A') = \emptyset$, this implies that $\mathcal{B}_\mathcal{I}[\![A']\!]$ is terminated regardless of $\mathcal{I}$, so (by requirement 5 (b) in Definition 15) $\mathcal{B}_\mathcal{I}[\![A]\!] \simeq \mathcal{B}_\mathcal{I}[\![M_3(e,f)]\!]$ holds for all $\mathcal{I}$. But it is not the case that $\mathcal{N}[\![A]\!] = \mathcal{N}[\![M_3(e,f)]\!]$; the network $\mathcal{N}[\![A]\!]$ has two nodes more than $\mathcal{N}[\![M_3(e,f)]\!]$. In this example $\mathcal{N}[\![A]\!]$ had an isolated part consisting of the nodes labelled M_1 and M_2.

Definition 16 An *isolated part* of a network $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L}\rangle$ is a nonempty subset $\mathcal{V}'$ of $\mathcal{V}$ such that all ports on all nodes in $\mathcal{V}'$ are linked (through links in $\mathcal{E}$) with other ports in $\mathcal{V}'$. □

Hence, an isolated part is a set of nodes with no external ports and no links leading out of the set. A key result is that the converse of Theorem 2 holds for networks without isolated parts:

Theorem 3 If $\mathcal{N}_1$ and $\mathcal{N}_2$ have no isolated parts, and $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ holds for some behavioural equivalence $\simeq$, then $\mathcal{N}_1 = \mathcal{N}_2$. □

The rather long proof of this theorem is contained in Section 3.6. The proof idea is to first define a set of structural properties of networks called chains and meeting chains; intuitively a chain is a way in which a network can be traversed by following links between nodes, and two chains meet if they terminate at the same node. We proceed to show that two networks with the same (meeting) chains must be isomorphic. The remainder of the proof shows that two behaviourally equivalent networks must have the same (meeting) chains. The main idea is here to show that there exists a "distinguishing interpretation" for each chain: in this operational interpretation a network has a certain completed trace exactly if it has the chain. Since behavioural equivalence must respect completed traces (requirement 5 (c) in Definition 15), two networks which differ in that only one of them has a certain chain cannot be behaviourally equivalent. Thus, if $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ then $\mathcal{N}_1$ and $\mathcal{N}_2$ must have the same (meeting) chains, whence they are isomorphic.

The major difficulty of the proof is to establish the distinguishing interpretations. This is comparatively easy if each module labels at most one node, since then a unique behaviour can be assigned to each node. In the general case, however, a module may occur more than once.

In summary we now have, for all behavioural equivalences $\simeq$:

Corollary 4 If $\mathcal{N}_1$ and $\mathcal{N}_2$ have no isolated parts, then $\mathcal{N}_1 = \mathcal{N}_2$ iff $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ □

3.5 Characterising Behavioural Equivalences

We will finally give an axiom schema which extends $\mathcal{A}_s$ to a complete system for behavioural equivalence. The idea is to let the new schema correspond to the removal of an isolated part from a network:

$$\boxed{\mathbf{I} \quad A|B \;=\; A \qquad (L(B) = \emptyset)}$$

The schema **I** says that a parallel factor with empty sort can be removed.

Lemma 5 If $\mathbf{I} \vdash C = D$ then $\mathcal{B}[\![C]\!] \simeq \mathcal{B}[\![D]\!]$ for all behavioural equivalences $\simeq$. □

Proof: The proof is immediate from requirement 5 (b) in Definition 15 since $L(B) = \emptyset$ implies that $\mathcal{B}_{\mathcal{I}}[\![B]\!]$ is terminated regardless of $\mathcal{I}$. □

In contrast, **I** is not sound for structural equivalence. The fact that **I** can be used to remove isolated parts is stated as follows:

Lemma 6 For any term A with $L(A) \neq \emptyset$ there exists a term A^+ such that $\mathcal{N}[\![A^+]\!]$ has no isolated part and such that $\mathcal{A}_s + \mathbf{I} \vdash A = A^+$. □

Proof: Let A be a term with a nonempty sort. As in the proof of Theorem 1, A is provably equivalent with a normal form A^* of type

$$(M_1(a_{1,1},\ldots,a_{1,\#M_1})|\cdots|M_n(a_{n,1},\ldots,a_{n,\#M_n}))\langle b_1\frown c_1\rangle\cdots\langle b_k\frown c_k\rangle$$

where each atomic network corresponds to a node in $\mathcal{N}[\![A]\!]$. Now assume that a subset of these nodes form an isolated part. Since $L(A) = L(A^*) \neq \emptyset$, some nodes must lie outside the isolated part. Through **C1** and **C2**, A^* can be rewritten to the form

$$(C|D)\langle b_1\frown c_1\rangle\cdots\langle b_k\frown c_k\rangle$$

where D is a parallel composition of all the atomic networks which correspond to nodes in the isolated part, and C is a parallel composition of the other atomic networks. Now for each linking operation $\langle b_i\frown c_i\rangle$, either both b_i and c_i are in $L(D)$, or both are in $L(C)$ (otherwise there would be a link out of the isolated part). So by using **D** and **C1**, all linking operations can be propagated past the parallel composition, to obtain a term of the form

$$C\langle b'_1\frown c'_1\rangle\cdots\langle b'_{k'}\frown c'_{k'}\rangle \mid D\langle b''_1\frown c''_1\rangle\cdots\langle b''_{k''}\frown c''_{k''}\rangle$$

Now $L(D\langle b''_1\frown c''_1\rangle\cdots\langle b''_{k''}\frown c''_{k''}\rangle) = \emptyset$ (otherwise the isolated part would have a node with an external port), so through **I**, the term can be rewritten to $C\langle b'_1\frown c'_1\rangle\cdots\langle b'_{k'}\frown c'_{k'}\rangle$. Clearly, this term denotes a network without the isolated part. In this way, any isolated part can be removed; in particular the largest isolated part can be removed (the isolated parts of a network are closed under union), this leaves a network with no isolated parts. $\square$

With these results it is not difficult to prove that $\mathcal{A}_s + \mathbf{I}$ is complete for behavioural equivalences:

Theorem 7 Let $\simeq$ be any behavioural equivalence. Then $\mathcal{B}[\![A]\!] \simeq \mathcal{B}[\![B]\!]$ iff $\mathcal{A}_s+\mathbf{I} \vdash A = B$. $\square$

Proof: The direction $\Leftarrow$ (soundness) is immediate from Theorems 1 and 2 and Lemma 5. For the direction $\Rightarrow$ (completeness), assume $\mathcal{B}[\![A]\!] \simeq \mathcal{B}[\![B]\!]$. Then $L(A) = L(B)$. If $L(A) = L(B) = \emptyset$, then $\mathbf{C1}+\mathbf{I} \vdash A = A|B = B|A = B$. If $L(A) = L(B) \neq \emptyset$, then by Lemma 6 there are terms A^+ and B^+ such that $\mathcal{A}_s + \mathbf{I} \vdash A = A^+$ and $\mathcal{A}_s + \mathbf{I} \vdash B = B^+$, and both A^+ and B^+ denote networks without isolated parts. By soundness, $\mathcal{B}[\![A^+]\!] \simeq \mathcal{B}[\![B^+]\!]$, so by Theorems 3 and 1 it follows that $\mathcal{A}_s \vdash A^+ = B^+$. In conclusion, $\mathcal{A}_s + \mathbf{I} \vdash A = A^+ = B^+ = B$. $\square$

So, the axiom schema **I** captures exactly the difference between structural and behavioural equivalence. An immediate corollary is that all behavioural equivalences on terms coincide. Remember that this does not mean that the equivalences on behaviours coincide; two terms are behaviourally equivalent only if they are identified in *all* operational interpretations on modules (i.e., if the modules are regarded as variables). For particular operational interpretations, and particular equivalences, stronger axiom systems may be sound. Theorem 7 says that for any behavioural equivalence $\mathcal{A}_s + \mathbf{I}$ is the strongest system which is sound for *all* such interpretations.

3.6 Proof of Theorem 3

3.6.1 Paths and Chains

In this section we always write $\mathcal{N}$ for the network $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L}\rangle$; similarly $\mathcal{N}' = \langle \mathcal{V}', \mathcal{M}', \mathcal{E}', \mathcal{L}'\rangle$ and $\mathcal{N}_i = \langle \mathcal{V}_i, \mathcal{M}_i, \mathcal{E}_i, \mathcal{L}_i\rangle$. We will use u, v etc. to range over nodes and x, y, z etc. to range over integers representing internal port numbers.

Definition 17 A *path* of length n (for $n \geq 1$) in the network $\mathcal{N}$ is a sequence

$$(a, x_1, v_1, y_1, x_2, v_2, y_2, \ldots, x_n, v_n)$$

where

$v_i \in \mathcal{V}$ for all i, $1 \leq i \leq n$;

$1 \leq x_i, y_i \leq \#\mathcal{M}(v_i)$ for all i, $1 \leq i \leq n$;

$a = \mathcal{L}(v_1, x_1)$;

$\{(v_i, y_i), (v_{i+1}, x_{i+1})\} \in \mathcal{E}$ for all i, $1 \leq i \leq n-1$. □

A path represents a way to traverse a network starting at an external port name a corresponding to the port (v_1, x_1), and continuing with links $\{(v_i, y_i)(v_{i+1}, x_{i+1})\}$. We will use the suggestive notation

$$\xleftrightarrow{a \frown x_1} v_1 \xleftrightarrow{y_1 \frown x_2} v_2 \xleftrightarrow{y_2 \frown x_3} \cdots \xleftrightarrow{y_{n-1} \frown x_n} v_n$$

for such a path. The ports (v_i, x_i) will be referred to as *upwards* ports and the ports (v_i, y_i) as *downwards* ports in the path.

Definition 18 A path is *linear* if all nodes in it are distinct. A path is *loop-ended* if it has length at least 2 and the ultimate and penultimate nodes coincide and all other nodes are distinct. A loop-ended path

$$\xleftrightarrow{a \frown x_1} v_1 \xleftrightarrow{y_1 \frown x_2} \cdots \xleftrightarrow{y_{n-2} \frown x_{n-1}} v \xleftrightarrow{y_{n-1} \frown x_n} v$$

will sometimes be written

$$\xleftrightarrow{a \frown x_1} v_1 \xleftrightarrow{y_1 \frown x_2} \cdots \xleftrightarrow{y_{n-2} \frown x_{n-1}} v \xleftrightarrow{y_{n-1} \frown x_n}$$

to highlight the fact that it is loop-ended. In the following we will only consider linear and loop-ended paths, so "path" will mean linear or loop-ended path from now on. □

Definition 19

1. A *linear chain* of length n (for $n \geq 1$) is a sequence $(a, x_1, M_1, y_1, x_2, M_2, y_2, \ldots, x_n, M_n)$ where $a \in \Lambda$ and $M_1, \ldots, M_n \in \mathbf{M}$ and $1 \leq x_i, y_i \leq \#M_i$ for all i, $1 \leq i \leq n$.

2. A *loop-ended chain* of length n (for $n \geq 2$) is a sequence $(a, x_1, M_1, y_1, \ldots, x_{n-1}, M_{n-1}, y_{n-1}, x_n)$ where $a \in \Lambda$ and $M_1, \ldots, M_{n-1} \in \mathbf{M}$ and $1 \leq x_i, y_i \leq \#M_i$ for all i, $1 \leq i \leq n-1$, and also $1 \leq x_n \leq \#M_{n-1}$.

3. A *chain* is a linear chain or a loop-ended chain. The chain *corresponding to* a particular path in $\mathcal{N}$ is the chain obtained by replacing each node v in the path by $\mathcal{M}(v)$. We say that $\mathcal{N}$ *possesses* (or *has*) a linear/loop-ended chain CH, written $CH \in \mathcal{N}$, if CH corresponds to some linear/loop-ended path in $\mathcal{N}$. □

We use the same notation for chains as for paths, so a linear chain will be written

$$\stackrel{a^\frown x_1}{\longleftrightarrow} M_1 \stackrel{y_1{}^\frown x_2}{\longleftrightarrow} M_2 \stackrel{y_2{}^\frown x_3}{\longleftrightarrow} \cdots \stackrel{y_{n-1}{}^\frown x_n}{\longleftrightarrow} M_n$$

and similarly a loop-ended chain will be written

$$\stackrel{a^\frown x_1}{\longleftrightarrow} M_1 \stackrel{y_1{}^\frown x_2}{\longleftrightarrow} \cdots \stackrel{y_{n-2}{}^\frown x_{n-1}}{\longleftrightarrow} M_{n-1} \stackrel{y_{n-1}{}^\frown x_n}{\longleftrightarrow}$$

Proposition 8 If $CH \in \mathcal{N}$ then there is exactly one path corresponding to CH in $\mathcal{N}$. □

Proof: We need to prove that the path corresponding to CH is unique in $\mathcal{N}$; the proof is by induction on the length of CH. The base case (CH has length 1) follows from the fact that the port labelling of a network is injective. The inductive step follows from the fact that a port in $\mathcal{N}$ can occur in at most one link. □

So, if $CH \in \mathcal{N}$ we can unambiguously talk about *the* path of CH in $\mathcal{N}$.

Definition 20 A path *ends* at its ultimate node. Similarly, a chain CH *ends* at v in $\mathcal{N}$ if $CH \in \mathcal{N}$ and its path in $\mathcal{N}$ ends at v. Two chains *meet at v in $\mathcal{N}$* if they end at the same node v in $\mathcal{N}$. □

Note that a chain meets itself in $\mathcal{N}$ iff it is in $\mathcal{N}$.

Definition 21 Two paths are in *conflict* if an upwards port in one of the paths is a downwards port in the other path. Two paths are *conflict free* if they are not in conflict. Similarly two chains in $\mathcal{N}$ are in conflict/conflict free in $\mathcal{N}$ if their paths in $\mathcal{N}$ are in conflict/conflict free. □

Note that a port cannot occur more than once in a path, so it cannot be both upwards and downwards in the same path. Hence a path is always conflict free with itself. Also, note that an upwards port of a path in a network is always linked to a downwards port in the path and vice versa (with the exception of the first upwards port). Thus two paths are conflict free if and only if no downwards port in one path is linked to a downwards port in the other path.

3.6.2 Chains and Structural Equivalence

Lemma 9 Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two networks without isolated parts satisfying

1. A loop-ended chain is in $\mathcal{N}_1$ iff it is in $\mathcal{N}_2$.
2. For all pairs of linear chains, the chains are conflict free in $\mathcal{N}_1$ iff they are conflict free in $\mathcal{N}_2$, and if so then they meet in $\mathcal{N}_1$ iff they meet in $\mathcal{N}_2$.

Then $\mathcal{N}_1$ is isomorphic to $\mathcal{N}_2$. □

Proof: Without loss of generality assume that $\mathcal{N}_1$ has at least as many nodes as $\mathcal{N}_2$, and let the nodes of $\mathcal{N}_1$ be enumerated by $v_1, v_2, \ldots$. Let the *depth* $d(v)$ of a node v in $\mathcal{N}_1$ be the length of the shortest path in $\mathcal{N}_1$ containing v. Since $\mathcal{N}_1$ has no isolated parts this defines the depth of all nodes in $\mathcal{N}_1$.

Using the depth and enumeration of nodes we define a total order $<$ between the ports in $\mathcal{N}_1$ as follows: $(v_i, z) < (v_j, z')$ if either of (1) $d(v_i) < d(v_j)$ or (2) $d(v_i) = d(v_j)$ and $i < j$ or (3) $i = j$ and $z < z'$. Say that a chain $CH \in \mathcal{N}_1$ is *admissible* if whenever its path contains $(\ldots v, z, z', v' \ldots)$ then $(v, z) < (v', z')$. This means that a downwards port must be smaller (w.r.t. $<$) than the upwards port to which it is linked.

Let AS be the set of all admissible linear chains in $\mathcal{N}_1$. Observe that any two chains in AS are conflict free in $\mathcal{N}_1$, since in each link the order on the ports determine which of the ports in the link can be an upwards port and which can be a downwards port. Hence by the premises the chains in AS are also conflict free in $\mathcal{N}_2$.

Define the binary relation $R \subseteq \mathcal{V}_1 \times \mathcal{V}_2$ by vRu if some linear chain in AS ends at v in $\mathcal{N}_1$ and ends at u in $\mathcal{N}_2$. We will now prove that R is a bijection.

First, to prove that R is a function, assume two linear chains CH, CH' in AS which both end at v in $\mathcal{N}_1$, and in u and u' respectively in $\mathcal{N}_2$. Since CH, CH' are in AS they are conflict free in $\mathcal{N}_1$, and thus they meet (at v) in $\mathcal{N}_1$, and by the premises of the lemma they then meet in $\mathcal{N}_2$, whence $u = u'$. This proves R a function. Similarly, R is injective by a symmetric argument.

We next argue that the domain of R contains all nodes in $\mathcal{N}_1$. Evidently it suffices to show that each node in $\mathcal{N}_1$ is the end of an admissible chain. Let v be a node and consider the shortest path containing v. Since $\mathcal{N}_1$ has no isolated parts such a path always exists, and by definition of depth it traverses nodes of depth $1, \ldots, d(v)$, whence the chain corresponding to that path is admissible.

Finally, the range of R contains all nodes in $\mathcal{N}_2$ since R is an injection and $\mathcal{N}_1$ has at least as many nodes as $\mathcal{N}_2$. This concludes the argument that R is a bijection.

For clarity we now define $h : \mathcal{V}_1 \to \mathcal{V}_2$ by $h(v) = u$ iff vRu. We have proved that R and hence h is a bijection, and will conclude the proof of the lemma by showing that h satisfies the requirements of isomorphism between $\mathcal{N}_1$ and $\mathcal{N}_2$.

First we must prove $\mathcal{M}_1(v) = \mathcal{M}_2(h(v))$. This is trivial since any chain in AS which terminates at v in $\mathcal{N}_1$ has the ultimate module $\mathcal{M}_1(v)$, and since it also terminates at $h(v)$ in $\mathcal{N}_2$ this ultimate module is also $\mathcal{M}_2(h(v))$.

Next we must prove $\mathcal{L}_1(v, z) = \mathcal{L}_2(h(v), z)$. Let $a = \mathcal{L}_1(v, z)$ and $M = \mathcal{M}_1(v)$. Then the chain (a, z, M) is in $\mathcal{N}_1$. Clearly this chain is admissible whence it is also in $\mathcal{N}_2$ where it (by definition) ends at $h(v)$. This means that that $\mathcal{N}_2$ has the path $(a, z, h(v))$ which is to say that $\mathcal{L}_2(h(v), z) = a$.

Finally we must prove that $\{(v, z), (v', z')\} \in \mathcal{E}_1$ iff $\{((h(v), z), (h(v'), z')\} \in \mathcal{E}_2$. Consider first the case where $v \neq v'$. To prove the forward implication assume $\{(v, z), (v', z')\} \in \mathcal{E}_1$. Assume without loss of generality that $(v, z) < (v', z')$. Consider an admissible linear chain CH terminating at v in $\mathcal{N}_1$ and thus by definition at $h(v)$ in $\mathcal{N}_2$. Let $M' = \mathcal{M}_1(v')$. Define the chain CH' by appending $\xleftrightarrow{z \frown z'} M'$ to CH; thus CH' is admissible and terminates at v' in $\mathcal{N}_1$ and (by definition) at $h(v')$ in $\mathcal{N}_2$. So the path corresponding to CH' in $\mathcal{N}_2$ is $(\ldots, h(v), z, z', h(v'))$; this establishes that $\{((h(v), z), (h(v'), z')\} \in \mathcal{E}_2$.

For the backward implication extend the total order on the ports to ports in $\mathcal{N}_2$ by $(h(v), z) < (h(v'), z')$ iff $(v, z) < (v', z')$; this extension is well defined since h is a bijection preserving the module labelling, and hence the number of ports at the nodes. The proof of the backward implication then is symmetric with the proof of the forward implication.

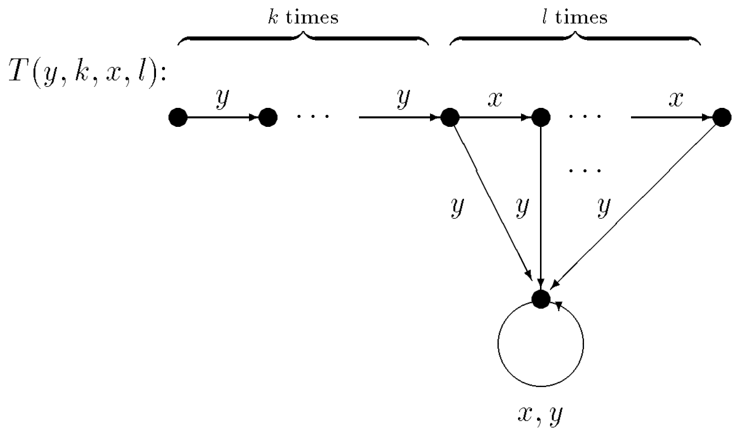
Lastly we consider the case where $v = v'$. The argument is similar to the case where $v \neq v'$; the only difference is that CH' will be a loop-ended chain, so we will here need the premise that $\mathcal{N}_1$ and $\mathcal{N}_2$ have the same loop-ended chains. This concludes the proof of Lemma 9. $\Box$

It should be noted that there exist nonisomorphic networks which have the same loop-ended and linear chains. Hence the premise that chains meet in $\mathcal{N}_1$ iff they meet in $\mathcal{N}_2$ cannot be relaxed.

3.6.3 Chains and Behavioural Equivalence

Lemma 10 Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two networks such that $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ for some behavioural equivalence $\simeq$. Then $\mathcal{N}_1$ and $\mathcal{N}_2$ have the same linear chains. $\Box$

Proof: The main idea of the proof is to construct a distinguishing interpretation $\mathcal{I}$ for any chain CH; this means that in $\mathcal{I}$ a network will have a certain completed trace exactly if the network has the chain CH. It follows that two behaviourally equivalent networks must have the same linear chains. This distinguishing interpretation is nontrivial, so we begin with some preliminary definitions of transition graphs which will be used in the construction. First consider the following class of transition graphs $T(y, k, x, l)$:



All labels of transitions are singleton sets; we omit the set brackets in this diagram and from now on. The bottom state has two transitions leading back to the same state: one is labelled x and the other is labelled y. For convenience we name the leftmost state $T^{left}(y, k, x, l)$. The behaviour from this state is as follows: it begins by doing k transitions labelled y, after which it can do l transitions labelled x. If it does an additional y transition (after the k first) then it ends up in the bottom state where arbitrarily many x and y actions are possible.

Consider next the following simpler transition graphs $U(x, l)$:

$$U(x,l): \quad \overbrace{\bullet \xrightarrow{x} \bullet \cdots \xrightarrow{x} \bullet}^{l \text{ times}}$$

Call the leftmost state $U^{left}(x,l)$. The behaviour from this state is simply to do l transitions labelled x leading to a terminated behaviour.

Finally let $\mathbf{0}$ stand for the rightmost state of $U(x,l)$: this is a terminated state, i.e. no transition leads from $\mathbf{0}$.

We shall here use $T^{left}(y,k,x,l)$ and $U^{left}(x,l)$ by, briefly stated, "assigning" instances of these behaviours to nodes in the path. The intuition is that a node will await a certain number of synchronisations on its downwards port and then engage in a number of synchronisations on its upwards port. The ultimate node, which lacks a downwards port, directly synchronises on its upwards port. The numbers of synchronisations (k and l) is important here: a node is always ready to synchronise on its downwards port, and any excessive such synchronisations will enable excessive synchronisations on its upwards port. The significance of this will become clear later.

A technical complication is that we can only assign behaviours to modules and not to nodes. A module can occur at several nodes in the chain, so it must have the possibility to enact each appropriate instance of T and U.

Let the chain CH be

$$\stackrel{a\frown x_1}{\longleftrightarrow} M_1 \stackrel{y_1\frown x_2}{\longleftrightarrow} M_2 \stackrel{y_2\frown x_3}{\longleftrightarrow} \cdots \stackrel{y_{n-1}\frown x_n}{\longleftrightarrow} M_n$$

Choose n distinct natural numbers $m_1,\ldots,m_n$. Define the interpretation $\mathcal{I}$ by assigning the following transitions to the modules in the corresponding transition graph $T_{\mathcal{I}}$:

$$\begin{array}{lll} M_i & \xrightarrow{\emptyset} & T^{left}(y_i, m_{i+1}, x_i, m_i) \text{ for } i = 1,\ldots,n-1; \\ M_n & \xrightarrow{\emptyset} & U^{left}(x_n, m_n); \\ M & \xrightarrow{\emptyset} & \mathbf{0} \text{ for all modules } M. \end{array}$$

Note that this may assign several transitions to modules: a module has one $\xrightarrow{\emptyset}$ transition for each place where it occurs in the chain (these transitions lead to different states), and in addition each module has a $\xrightarrow{\emptyset}$ transition to a terminated state.

Let $\sigma = a^{m_1}$ (i.e. a sequence of m_1 a-actions). We now argue that for all networks $\mathcal{N}$ such that $CH \in \mathcal{N}$ it holds that $\mathcal{N} \stackrel{\sigma}{\Longrightarrow} \bullet$. Let $v_1,\ldots,v_n$ be the nodes in the path of CH in $\mathcal{N}$. Consider the sequence of transitions which involve first m_n synchronisations on the link $\{(v_n,x_n),(v_{n-1},y_{n-1})\}$, thereafter m_{n-1} synchronisations on the link $\{(v_{n-1},x_{n-1}),(v_{n-2},y_{n-2})\}$, and similarly continuing with m_i synchronisations on the link $\{(v_i,x_i),(v_{i-1},y_{i-1})\}$, and finally m_1 actions at port (v_1,x_1). It is clear from the path of CH that all these links exist, and from the interpretation $\mathcal{I}$ that all nodes in that path are assigned modules which enable this transition sequence (for this it is important that $v_1,\ldots v_n$ are distinct, i.e. that the chain is linear). Continue this transition sequence by letting all nodes which are not in the path of CH do the transition $\xrightarrow{\emptyset}$ to $\mathbf{0}$. The result of this transition sequence is a network $\mathcal{N}'$ where all nodes are associated with modules which are rightmost states in instances of T or U. Hence $\mathcal{N}'$ has a terminated behaviour in $\mathcal{I}$: nodes in the rightmost states of the transition diagrams can only continue by engaging in synchronisations on downwards ports in the chain, and no such synchronisation is possible since no downwards port is linked to another downwards port. Furthermore the only transitions with nonempty labels in the transition sequence are m_1 transitions on port (v_1, x_1). The path of CH implies that $\mathcal{L}(v_1, x_1) = a$. This proves that $\mathcal{N} \stackrel{\sigma}{\Longrightarrow} \mathcal{N}'$ where $\mathcal{N}'$ is terminated, i.e. $\mathcal{N} \stackrel{\sigma}{\Longrightarrow} \bullet$.

We next argue the converse: if $\mathcal{N} \stackrel{\sigma}{\Longrightarrow} \bullet$ then $CH \in \mathcal{N}$. So assume $\mathcal{N} \stackrel{\sigma}{\Longrightarrow} \bullet$. This means that some node, call it u_1, in $\mathcal{N}$ has a port, call it (u_1, z), such that $\mathcal{L}(u_1, z) = a$. Moreover, the module $\mathcal{M}(u_1)$ must be able to reach a state where it can do precisely m_1 actions on its port number z and then *not* continue with actions on z. Inspection of the interpretation $\mathcal{I}$ yields that such a state can only be in $T(y_1, m_2, x_1, m_1)$ (or, if the length of CH is 1, in $U(x_1, m_1)$) with the one possible value x_1 for z. Hence $\mathcal{M}(u_1)$ must be a module which can reach $T^{left}(y_1, m_2, x_1, m_1)$ (or $U^{left}(x_1, m_1)$); by definition of $\mathcal{I}$ this module must be M_1. This establishes the path $\stackrel{a\frown x_1}{\longleftrightarrow} u_1$ with the corresponding chain $\stackrel{a\frown x_1}{\longleftrightarrow} M_1$ in $\mathcal{N}$, so if the length of CH is 1 then we are done.

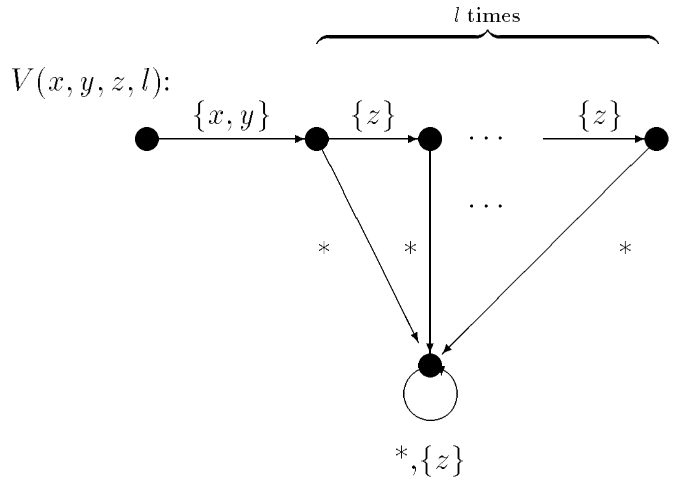
If the length of CH is more than 1 then M_1, in order to perform its m_1 actions on x_1, first requires m_2 actions on port number y_1 (this by definition of $T(y_1, m_2, x_1, m_1)$). Moreover, after these actions no additional y_1 action should ever be allowed, otherwise M_1 will be able to continue with x_1-actions, contradicting the fact that $\mathcal{N}$ can reach a terminated state after σ. It follows that (u_1, y_1) cannot be an external port (or the actions on that port would have to precede σ); say that it is linked to another port (u_2, z). Then $\mathcal{M}(u_2)$ must be able to reach a state where it can do precisely m_2 actions on its port number z and then *not* continue with actions on z. Inspection of the interpretation $\mathcal{I}$ yields that such a state can only be in $T(y_2, m_3, x_2, m_2)$ (or, if the length of CH is 2, in $U(x_2, m_2)$) with the one possible value x_2 for z. Hence $\mathcal{M}(u_2)$ must be a module which can reach this state; by definition of $\mathcal{I}$ this module must be M_2. This establishes the path $\stackrel{a\frown x_1}{\longleftrightarrow} u_1 \stackrel{y_1\frown x_2}{\longleftrightarrow} u_2$ with the corresponding chain $\stackrel{a\frown x_1}{\longleftrightarrow} M_1 \stackrel{y_1\frown x_2}{\longleftrightarrow} M_2$ in $\mathcal{N}$, so if the length of CH is 2 then we are done.

If the length of CH is more than 2 then the argument continues in a similar way until all of CH has been established in $\mathcal{N}$. This concludes the proof of Lemma 10. □

Note that in this proof the numbers $m_1, \ldots, m_n$ can be freely chosen. We call these numbers the *characteristic numbers* for the interpretation $\mathcal{I}$. For the purposes of the proof above it would suffice to fix the characteristic numbers by e.g $m_i = i$, but we will later (in the proof of Lemma 12 below) use the more general formulation of the proof of Lemma 10.

Lemma 11 Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two networks such that $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ for some behavioural equivalence $\simeq$. Then $\mathcal{N}_1$ and $\mathcal{N}_2$ have the same loop-ended chains. □

Proof: The proof is similar to the proof of Lemma 10; we here only indicate the differences. Consider the following transition graph $V(x, y, z, l)$:



We here write out the set brackets since the first transition from the leftmost state $V^{left}(x, y, z, l)$ is the set $\{x, y\}$. All other transitions in the graph are labelled with singleton sets; transitions labelled "$*$" actually correspond to two transitions, one labelled $\{x\}$ and one labelled $\{y\}$. The behaviour from $V^{left}(x, y, z, l)$ begins with one transition $\{x, y\}$ and continues with l transitions labelled $\{z\}$. Following the first transition it can always do $\{x\}$ and $\{y\}$ transitions, thereby ending up in the bottom state where further $\{x\}$, $\{y\}$ and $\{z\}$ transitions are possible.

Assume that the loop-ended chain CH is

$$\overset{a^\frown x_1}{\longleftrightarrow} M_1 \overset{y_1^\frown x_2}{\longleftrightarrow} \dots \overset{y_{n-2}^\frown x_{n-1}}{\longleftrightarrow} M_{n-1} \overset{y_{n-1}^\frown x_n}{\longleftrightarrow}$$

The interpretation $\mathcal{I}$ is now defined as follows:

$$\begin{array}{lll} M_i & \overset{\emptyset}{\longrightarrow} & T^{left}(y_i, m_{i+1}, x_i, m_i) \text{ for } i = 1, \dots, n-2; \\ M_{n-1} & \overset{\emptyset}{\longrightarrow} & V^{left}(x_n, y_{n-1}, x_{n-1}, m_{n-1}); \\ M & \overset{\emptyset}{\longrightarrow} & \mathbf{0} \text{ for all modules } M. \end{array}$$

So assume that $CH \in \mathcal{N}_1$; we will prove that $CH \in \mathcal{N}_2$. As in Lemma 10 let $\sigma = a^{m_1}$. The argument that $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$ is similar to the corresponding argument in that lemma. Let $v_1, \dots, v_n$ be the nodes in the path of CH in $\mathcal{N}_1$ (note that here $v_{n-1} = v_n$ since CH is loop-ended). Consider the sequence of transitions which involve first one synchronisation on the link $\{(v_n, x_n), (v_{n-1}, y_{n-1})\}$ and thereafter proceeds as in the proof of Lemma 10; this sequence demonstrates that $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$.

By the premises and $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$ it follows that $\mathcal{N}_2 \overset{\sigma}{\Longrightarrow} \bullet$. The proof that $CH \in \mathcal{N}_2$ follows the corresponding argument in Lemma 10 until the (linear) chain

$$\overset{a^\frown x_1}{\longleftrightarrow} M_1 \overset{y_1^\frown x_2}{\longleftrightarrow} \dots \overset{y_{n-1}^\frown x_{n-1}}{\longleftrightarrow} M_{n-1}$$

has been established in $\mathcal{N}_2$ with the nodes $u_1, \ldots, u_{n-1}$. There remains to establish the final link $\{(u_{n-1}, y_{n-1}), (u_{n-1}, x_n)\}$ in $\mathcal{N}_2$. We already know that $\mathcal{M}(u_{n-1}) = M_{n-1}$ and that the transition sequence requires this module to perform its m_{n-1} transitions labelled $\{x_{n-1}\}$, hence the transition labelled $\{x_n, y_{n-1}\}$ must have been performed first. So (u_{n-1}, x_n) cannot be an external port (if so, then an action on such an external port would have preceded σ). Assume this port is linked to a port (u, z). Then u must be one of $u_1, \ldots, u_{n-1}$ (otherwise this link establishes a linear chain in $\mathcal{N}_2$ not present in $\mathcal{N}_1$ contradicting Lemma 10). But u must do one synchronisation on its link with (u_{n-1}, x_n), so u cannot be anyone of $u_1, \ldots, u_{n-2}$ (these nodes must enact the behaviours of $T^{left}(y_1, m_2, x_1, m_1), \ldots, T^{left}(y_{n-2}, m_{n-1}, x_{n-2}, m_{n-2})$ respectively). It follows $u = u_{n-1}$, and the only possible value for z is y_{n-1}. This establishes the link $\{(u_{n-1}, y_{n-1}), (u_{n-1}, x_n)\}$ in $\mathcal{N}_2$ and concludes the proof of Lemma 11. □

Lemma 12 Let $\mathcal{N}_1$ and $\mathcal{N}_2$ be two networks such that $\mathcal{B}(\mathcal{N}_1) \simeq \mathcal{B}(\mathcal{N}_2)$ for some behavioural equivalence $\simeq$. Then any two linear chains are conflict free in $\mathcal{N}_1$ iff they are conflict free in $\mathcal{N}_2$, and if so then they meet in $\mathcal{N}_1$ iff they meet in $\mathcal{N}_2$. □

Proof: Let CH_1 and CH_2 be two linear chains, of length n_1 and n_2 respectively, which are conflict free in $\mathcal{N}_1$. We will prove that they must also be conflict free in $\mathcal{N}_2$, and that if they meet in $\mathcal{N}_1$ they must meet in $\mathcal{N}_2$. The lemma then follows by symmetry.

This proof is an extension of the proof of Lemma 10, so we adopt all definitions from that proof. Consider first the chain CH_1. As in the proof of Lemma 10 we obtain a distinguishing interpretation $\mathcal{I}_1$ for this chain. This interpretation is defined in terms of transition graphs of type T and U, and we can choose the characteristic numbers $m_1, \ldots, m_{n_1}$ freely. For the definition of $\mathcal{I}_1$ we fix $m_i = i$ for all i. Similarly for the chain CH_2 we obtain a distinguishing interpretation $\mathcal{I}_2$ but for this interpretation we choose the characteristic numbers $m_i = i(n_1+1)$. The important point will be that any number between 1 and $(n_2+1)(n_1+1)$ can be uniquely decomposed as a sum of either 0 or one characteristic number from $\mathcal{I}_1$ and either 0 or one characteristic number from $\mathcal{I}_2$.

Now for each state p in $\mathcal{I}_1$ and each state q in $\mathcal{I}_2$ construct a new state $p||q$. We will build an interpretation $\mathcal{J}$ using these new states; the transitions in $T_{\mathcal{J}}$ between the states are given by the following inference rules:

$$\frac{p \xrightarrow{\alpha} p'}{p||q \xrightarrow{\alpha} p'||q} \qquad \frac{q \xrightarrow{\alpha} q'}{p||q \xrightarrow{\alpha} p||q'}$$

Thus $p||q$ behaves as p and q in parallel under interleaving; a transition sequence from $p||q$ is an interleaving between a transition sequence from p and a transition sequence from q. Finally introduce the modules in the networks as states in $T_{\mathcal{J}}$ and give them the following transitions in:

$$M \xrightarrow{\emptyset} p||q \quad \text{if } M \xrightarrow{\emptyset} p \text{ in } \mathcal{I}_1 \text{ and } M \xrightarrow{\emptyset} q \text{ in } \mathcal{I}_2$$

This means that a module has one transition in $\mathcal{J}$ for each combination of transitions in $\mathcal{I}_1$ and $\mathcal{I}_2$. Note that each module receives at least the transition $M \xrightarrow{\emptyset} \mathbf{0}||\mathbf{0}$ with this definition and that the behaviour of $\mathbf{0}||\mathbf{0}$ is isomorphic with the behaviour of $\mathbf{0}$.

Assume that CH_1 begins with the name a_1 and CH_2 begins with the name a_2. As in the proof of Lemma 10 we get that in $\mathcal{I}_1$ it holds that $\mathcal{N}_1 \overset{a_1}{\Longrightarrow} \mathcal{N}_1'$ where all modules in $\mathcal{N}_1'$ await synchronisations on downwards ports of CH_1. Similarly in $\mathcal{I}_2$ it holds that $\mathcal{N}_1 \overset{{a_2}^{n_1+1}}{\Longrightarrow} \mathcal{N}_1''$ where all modules in $\mathcal{N}_1''$ await synchronisations on downwards ports of CH_2. By construction of $\mathcal{J}$ we then know that in that interpretation $\mathcal{N}_1 \overset{a_1 {a_2}^{n_1+1}}{\Longrightarrow} \mathcal{N}_1'''$ where all modules in $\mathcal{N}_1'''$ are of the form $p||q$, and p awaits synchronisations on downwards ports in CH_1, and q awaits synchronisations on downwards ports in CH_2. Since the chains are conflict free in $\mathcal{N}_1$ no two such ports can be interlinked. Hence $\mathcal{N}_1'''$ is terminated. Put $\sigma = a_1\, {a_2}^{n_1+1}$; we have now proved $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$. Note that this holds even if the paths of CH_1 and CH_2 have nodes in common, and even if $a_1 = a_2$.

By the premises we get that also $\mathcal{N}_2 \overset{\sigma}{\Longrightarrow} \bullet$. By repeating the final argument in the proof of Lemma 10 for each of the two chains we establish paths corresponding to CH_1 and CH_2 in $\mathcal{N}_2$. These paths may share nodes and even ports so here the unique decomposition property for the characteristic numbers is essential: when a node requires precisely m synchronisations (for some m) on its downwards port, then there is at most one module which can supply these m synchronisations. In Lemma 10 this was achieved by requiring the characteristic numbers m_i to be distinct. The present lemma is more complicated since a module may enact one transition graph from $\mathcal{I}_1$ interleaved with one transition graph from $\mathcal{I}_2$. The unique decomposition property, however, guarantees that given m these transition graphs, and thus also the module, are uniquely determined.

This argument shows that the only possible reason for $\mathcal{N}_2 \overset{\sigma}{\Longrightarrow} \bullet$ is that $\mathcal{N}_2 \overset{\sigma}{\Longrightarrow} \mathcal{N}_2'$ where the modules involved with the chains in $\mathcal{N}_2'$ have evolved to the form $p||q$, where p and q represent rightmost states in transition diagrams of type T or U. So for each downwards port in CH_1 there is such a p awaiting synchronisations on that port, similarly for each downwards port in CH_2 there is such a q awaiting synchronisations on that port. If two such ports were interlinked then these synchronisations would be possible, taking p and q to their bottom states. This would mean that further synchronisations are enabled, and these will finally lead to further external actions on a_1 and a_2, contradicting the fact that $\mathcal{N}_2'$ is terminated. Hence no such link between two downwards ports exists. This implies that the chains are conflict free in $\mathcal{N}_2$ and thus proves the first part of the lemma.

For the final part of the lemma we additionally assume that CH_1 and CH_2 meet in $\mathcal{N}_1$; we will show that they then must meet in $\mathcal{N}_2$. Clearly, if the chains meet in a network they must have the same ultimate module. Call this module M. By construction of $\mathcal{J}$ above we know that this module has the transition

$$M \xrightarrow{\emptyset} U(z_1, n_1) || U(z_2, n_2(n_1+1)) \qquad (*)$$

where z_i is the number of the ultimate upwards port in CH_i. This transition makes it possible for M to enact the appropriate instances of U for the two chains in parallel. Now define an interpretation $\mathcal{J}'$ which differs from $\mathcal{J}$ only in that $(*)$ is removed. $\mathcal{J}$ and thus also $\mathcal{J}'$ still have (at least) the two transitions

$$M \xrightarrow{\emptyset} U(z_1, n_1) || \mathbf{0} \qquad \text{and} \qquad M \xrightarrow{\emptyset} \mathbf{0} || U(z_2, n_2(n_1+1))$$

so M can enact either of the instances of U; the main point is that in $\mathcal{J}'$ it cannot enact both in parallel.

Consider the behaviour of $\mathcal{N}_1$ in $\mathcal{J}'$. We argue that $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$ is *not* possible. As was the case for $\mathcal{J}$ the only possible reason for $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$ is an interleaving of the transition sequences for CH_1 in $\mathcal{I}_1$ and CH_2 in $\mathcal{I}_2$. In particular the ultimate node in the paths of the chains would have to enact an interleaving of the two instances of U, but this is impossible.

So by the premises of the lemma we then know that $\mathcal{N}_2 \overset{\sigma}{\Longrightarrow} \bullet$ is not possible. But if the chains do not meet in $\mathcal{N}_2$ then the argument that $\mathcal{N}_1 \overset{\sigma}{\Longrightarrow} \bullet$ in $\mathcal{J}$ carries over to $\mathcal{J}'$ (the transition (*) is never needed); thus the chains must meet in $\mathcal{N}_2$. This concludes the proof of Lemma 12. $\square$

In conclusion, Theorem 3 follows from Lemmas 9–12. Note that the proof uses actions with more than one name only when there are loops, i.e. links connecting two ports at the same node. We therefore expect this proof to carry over to e.g. CCS or CSP, where actions are singleton sets or empty, and loops do not occur. An additional complication in CCS and CSP is that ports may be linked to more than one other port, so it may be that the concepts of path and chain need refinement.

4 Discussion

We have defined an algebraic language for description of networks, and examined structural and behavioural interpretations of the language. We have compared and axiomatised the equivalences induced by these interpretations. Briefly stated, the main result is that nonisomorphic networks without isolated parts can always be made behaviourally unequal by assigning proper behaviours to the nodes.

One interesting open problem concerns the choice of behavioural equivalences: our definition prohibits *trace equivalence*, which identifies behaviours with the same traces (a trace is a sequence of non-empty actions; a behaviour has a trace if it can do this sequence interleaved with arbitrarily many empty actions). The crucial fact is that trace equivalence is not sensitive to *completed* traces. We do not know if Theorem 7 is true if trace equivalence is admitted as a behavioural equivalence; we have neither found a counterexample nor been able to prove a sufficiently strong variant of Theorem 3.

Our investigation can be varied in many ways. First, other classes of networks can be considered; one possibility is to admit multi-links (links connecting more than two ports) or to allow ports to be connected through more than one link. An interesting class of networks in this respect is the "proper nets" investigated by Milner [8] (these are essentially flowgraphs corresponding to CCS expressions); this class admits some multi-links, but not all. Second, other types of operations on networks and behaviours are possible; interesting candidates would be the static operations (parallel compositions, renamings, restrictions, hidings etc.) from formalisms such as CCS, CSP, ACP or MEIJE. Our class of networks and our operations were chosen for simplicity rather than for generality. Nevertheless they admit many interesting examples; see our companion paper [12] for an investigation of the expressiveness of our formalism.

As an example of a variant we here briefly consider introduction of explicit *renaming* operations. For each bijective substitution θ on Λ, add the function symbol $\theta : L \to \theta(L)$ to the signature (here θ is extended pointwise to sets of port names). Let this symbol be interpreted on networks by $\langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L} \rangle \theta = \langle \mathcal{V}, \mathcal{M}, \mathcal{E}, \mathcal{L} \circ \theta \rangle$. Thus, θ renames the external ports of a net-

R1	$(A\|B)\theta$	$=$	$A\theta\|B\theta$	
R2	$A\langle a^\frown b\rangle\theta$	$=$	$A\theta\langle\theta(a)^\frown\theta(b)\rangle$	
R3	$A\theta\theta'$	$=$	$A(\theta\circ\theta')$	
R4	$A\theta$	$=$	A	$(\theta(a)=a$ for all $a\in L(A))$

Table 4: Axiom schemas for explicit renaming operations.

work. Define the behavioural interpretation as follows: The behaviour $(\langle Q_1,\ L_1,\ \longrightarrow_1\rangle,\ q^0)\theta$ is

$$(\langle Q_1,\ \theta(L_1),\ \longrightarrow\rangle,\ q^0)$$

where $\longrightarrow$ is defined by the following rule:

$$\frac{q \xrightarrow{\alpha}_1 q'}{q \xrightarrow{\theta(\alpha)} q'}$$

Now, all results in this paper carry over to the extended algebra, and there is no longer any need for substitution of port names at the meta level (axiom schema **S**). The soundness and completeness results still hold if this schema is replaced by schemas **R1**–**R4** in Table 4.

The setting with renaming operations has the advantage that modules need not be explicitly parametrised on port names in atomic networks; each module can instead be given some arbitrary sort (instead of an arity) and different instances can be expressed with renaming operations.

Acknowledgments

I am grateful to Robin Milner and David Walker for inspiring discussions on this and related subjects, and for critical readings of the report [13], which was written when the author was on leave at the University of Edinburgh, supported by a grant from the British Science and Engineering Research Council. For the later development in the present paper I am grateful to Ed Brinksma, Bengt Jonsson, Gunnar Sjödin, Frits Vaandrager, and the anynomous referees for many helpful comments.

References

[1] D. Austry and G. Boudol. Algèbre de processus et synchronisation. *Theoretical Computer Science*, 30(1):91–131, 1984.

[2] J. Bergstra and J. Klop. Process algebra for synchronous communication. *Information and Computation*, 60:109–137, 1984.

[3] S. Brookes, C.A.R. Hoare, and W. Roscoe. A theory of communicating sequential processes. *J. ACM*, 31(3):560–599, 1984.

[4] B. Jonsson. A model and proof system for asynchronous networks. In *Proceedings of the 4:th ACM Symposium on Principles of Distributed Computing*, pages 49–58, 1985.

[5] G. Kahn. The semantics of a simple language for parallel programming. In *IFIP 74*, pages 471–475, 1974.

[6] R. Keller and P. Panangaden. Semantics of networks containing indeterminate operators. In Brookes, Roscoe, and Winskel, editors, *Seminar on Concurrency 1984, Springer Verlag Lecture Notes of Computer Science vol. 197*, pages 479–496, 1985.

[7] G. Milne. CIRCAL and the representation of communication, concurrency and time. *ACM Transactions on Programming Languages and Systems*, 7(2):270–298, 1985.

[8] R. Milner. Flowgraphs and flow algebras. *J. ACM*, 26(4):794–818, 1979.

[9] R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25:267–310, 1983.

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

[11] D. Park. The 'fairness' problem and nondeterministic computing networks. In de Bakker and van Leeuwen, editors, *Foundations of Computer Science IV, Part 2*, pages 133–161. Amsterdam, 1983. Mathematical Centre Tracts 159.

[12] J. Parrow. The expressive power of parallelism. *Future Generation Computer Systems*, 6:271–285, 1990.

[13] J. Parrow. Synchronisation flow algebra. Technical Report LFCS-87-35, Laboratory for Foundations of Computer Science, Department of Computer Science, University of Edinburgh, 1987.

[14] J. Staples and V. Nguyen. A fixpoint semantics for nondeterministic data flow. *J. ACM*, 32(2):411–444, 1985.