True concurrency equivalence semantics: an overview

Vashti Galpin

Department of Computer Science, University of Edinburgh
vcg@dcs.ed.ac.uk

This paper will present an overview of true concurrency semantic equivalences for CCS which have been presented in the literature in recent years. Selected equivalences will be described and examples will be given of their use. Finally, these equivalences will be compared with respect to CCS and a partial hierarchy will be given. I will also describe the research that I propose to pursue towards my PhD. This research is in the area of theory of concurrency and deals specifically with semantic equivalences defined on labelled transition systems, with the main focus being on equivalences for true concurrency. I wish to compare the numerous equivalences presented in the literature and to determine when and how these equivalences can be used.

1 Introduction

Over the last 15 years, there has been much research in the area of concurrency theory, that is research that attempts to define theoretical foundations for concurrent computation. This is generally considered to be a more complex task than dealing with sequential computation for two reasons: first, there is the possibility of interaction between different parts of a computation, and second, concurrent computations often form reactive systems [PR88]. A reactive system is one that interacts with its environment on an ongoing basis, and hence is difficult to characterise by a set of inputs and a set of outputs.

Algebraic calculi of processes provide an approach to modelling communication and concurrency by describing the behaviour of concurrent communicating systems in terms of an operational semantics. Milner's CCS (Calculus of Communicating Systems) [Mil89] is an example of this type of calculus. CCS provides for the definition of processes using a small number of operators and allows for the description and specification of concurrent systems. It consists of a syntax and an operational semantics defined using Structured Operational Semantics [Plo88]. These together give rise to a labelled transition system, where each process is represented by a state in the labelled transition system and the transitions between states represent actions that processes can perform. However, labelled transition systems are not sufficiently abstract, and therefore equivalences are defined over labelled transition systems to equate elements of the domains with similar behaviours. Each equivalence class can then be viewed as representing a particular behaviour, or alternatively as the meaning of a process that is a member of that equivalence class. Such equivalences on process domains are called semantic equivalences. Well-known interleaving equivalences defined over labelled transition systems are strong equivalence which is based on the concept of a relation over processes being a strong bisimulation and observation equivalence based on the concept of (weak) bisimulation [Mil89].

In interleaving semantics, concurrency is equated with nondeterministic sequentiality. This is demonstrated in CCS by the expansion law [Mil89] and an example of its application is as follows:

$$a.nil \mid b.nil \approx a.b.nil + b.a.nil \quad (a \neq \overline{b}).$$

In true concurrency semantics[1], this equation does not hold. Other differences are that interleaving semantics are held to be mathematically more tractable, and that true concurrency semantics are better for dealing with certain properties such as fairness. It is an important area of research to determine what properties hold for the different equivalences. There appear to be two main directions in which true concurrency semantics have been pursued. The first relates to causality, and involves distinguishing actions that are dependent

[1] By the phrase 'true concurrency', I mean any equivalence where this equation does not hold; the word 'non-interleaving' is also used in the literature

on other actions from those that are independent; the second relates to locality, and involves distinguishing actions that occur in different locations of a distributed system.

Sometimes, in the literature there is lack of clarity about the equivalence semantics. For example, Petri nets are sometimes described as having true concurrency semantics when they can have either type of semantics defined; the structure of Petri nets does not inhibit the definition of true concurrency equivalence semantics. The term 'true concurrency semantics' is also used to refer to the operational semantics of a particular model as opposed to equivalence semantics that are used to equate processes. In this research, I intend to focus solely on equivalence semantics. Labelled transition systems in their simplest form [Mil89, vG90b, vG93] appear only to allow the definition of interleaving equivalences. However, many authors have defined extensions of CCS on modified labelled transition systems that allow for definition of true concurrency equivalences and these will be the focus of this paper.

An important question is how equivalences can be applied. They are often used to show that an implementation meets its specification. In CCS, this is done by describing both specification and implementation as CCS processes and then showing that they are observationally equivalent. It is important to find out properties of the true concurrency equivalences, so we understand how they can be used in this type of application.

This paper will describe some of the research that I intend to do for my PhD and it is based on my recently submitted proposal. In this paper, I aim to give some background to this research, as well as describing the research I plan to carry out. In the second section, I will describe different approaches to true concurrency equivalence semantics that have been presented in the literature. Then in the third section, I will discuss in detail the approach taken in terms of CCS and extensions to CCS. This section will include examples of how the equivalences can be used. Finally, I will discuss the research I propose to do.

2 Background

There appear to be two basic approaches to defining true concurrency equivalence semantics for process algebras.

- The first is to map the syntax of a process algebra onto some construct that allows for definition of true concurrency semantics such as event structures or Petri nets. Examples of this approach are [DM87, DDNM88a, DDNM88b, DDNM90, GJ92, Gol90, Tau90, vGV87, Win82]. However, in many of these papers, semantic equivalences are not defined—the authors are concerned with true concurrency at an operational level only. I do not intend to discuss such approaches further in this document.
- The second approach is to modify the underlying transition system so that it contains the information that allows for the distinction between concurrency and nondeterministic sequentiality. This is the focus of this paper and hence this approach will be discussed in the following section.

Some research has attempted an overview of the different semantics and models. Category theoretical approaches are taken in [Mes90, MY89, SNW93] and other approaches appear in [AD89, Arn93, BC90, DN87, vG90b, vG93]. Many of these approaches do not investigate semantic equivalences [BC90, Mes90, MY89, SNW93]. The work of van Glabbeek [vG90b, vG93] investigates interleaving equivalences in terms of linear time and branching time. In [DN87], an early investigation into equivalences on labelled transition systems is given. In [AD89, Arn93], bisimulations are defined as homomorphisms on labelled transition systems. Different notions of equivalence have been investigated on event structures [vG90a]. Labelled transition systems and bisimulation have been studied in both a domain theoretic setting [Abr91] and a category theoretic setting [CFM90].

Most of the articles that give overviews of different true concurrency semantics focus on operational semantics and not equivalence semantics. The research I propose to do, is a first attempt to find a general framework in which to describe and compare all true concurrency semantic equivalences that have been defined for various versions of labelled transition systems.

3 CCS and extensions of CCS

As mentioned in the introduction, process algebras are defined in terms of a syntax and an operational semantics, which form a labelled transitions system. Labelled transition systems have the following definition:

Definition 1 (Labelled transition system, pure labelled transition system)
A labelled transition system *is defined as*

$$(\mathcal{S}, \mathcal{A}, \{\xrightarrow{a} \subseteq \mathcal{S} \times \mathcal{S} \mid a \in \mathcal{A}\})$$

where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of transition labels called actions, and the relations $\xrightarrow{a}$ describe which transitions occur between states.

Generally, we write $s \xrightarrow{a} s'$ for $(s, s') \in \xrightarrow{a}$ where $s, s' \in \mathcal{S}$. Note that in this definition, no structure is specified for the states or for the actions. A specific labelled transition which has unstructured states and transitions will be referred to as a pure *labelled transition system.*

As discussed above, I will focus on the process algebra CCS [Mil89]. This can be described in the following manner: Let $\mathcal{A}$ be an infinite set of names $a, b, c, \ldots$, and $\overline{\mathcal{A}}$ be the set of co-names $\overline{a}, \overline{b}, \overline{c}, \ldots$, with $\overline{\overline{a}} = a$. $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}}$ is the set of labels, ℓ and $\overline{\ell}$ range over $\mathcal{L}$, and K, L range over $\mathcal{L}$. τ is a distinguished action such that $\tau \notin \mathcal{L}$, called the invisible, silent or perfect action. Let $Act = \mathcal{L} \cup \{\tau\}$, with α, β ranging over Act. A relabelling function $f : \mathcal{L} \to \mathcal{L}$ is a function such that $f(\overline{\ell}) = \overline{f(\ell)}$. It is extended to Act by defining $f(\tau) = \tau$.

(Finite) CCS has the following syntax:

$$P ::= nil \mid \alpha.P \mid P + P \mid P|P \mid P\backslash L \mid P[f]$$

We use $\mathcal{P}$ to define the set of processes generated by this syntax. To describe each operator briefly, nil represents the process that can nothing; $a.P$ represents the prefixing of a process by an action; $P + P$ represents nondeterministic choice between processes; $P|P$ represents parallel composition of processes; $P\backslash L$ represents the restriction of certain actions in P, that is P cannot perform the actions in the set L; and $P[f]$ represents the relabelling of the actions in P by the function f.

The operational semantics of CCS are defined in terms of a labelled transition system, $(\mathcal{P}, Act, \{\xrightarrow{\alpha} \subseteq \mathcal{P} \times \mathcal{P} \mid \alpha \in Act\})$, where the relations $\xrightarrow{\alpha}$ are defined to be the smallest relations satisfying the rules in Figure 1. These rules can be informally described as follows: (ST1) states that the action a prefixed process can perform is the action with which it is prefixed. (ST2) states that if P can perform an action, then that same action can be performed by the nondeterministic choice of P and Q and the result of this will be the result of P performing the action—Q is discarded. (ST3) states that if P can perform an action, then P in parallel with Q can perform the same action and the result will be the result of performing P in parallel with Q. (ST4) states that if P can perform an action and Q can perform a complementary action when P and Q are in parallel, then communication can occur between P and Q which is represented by the invisible action τ. (ST5) states that if P can perform an action then P relabelled can perform the action relabelled, and finally (ST6) states that if P can perform an action then P restricted by a set can perform the same action, as long as the action does not appear in the set.

Example Consider the following processes:

- $a.nil + \overline{a}.nil$ can perform the action a or the action $\overline{a}$.
- $a.nil \mid \overline{a}.nil$ can perform the actions a then $\overline{a}$, or the actions $\overline{a}$ then a, or the perfect action τ.
- $(a.nil \mid \overline{a}.nil)\backslash a$ can only perform the perfect action τ.

A well-known semantic equivalence defined for labelled transition systems, and hence for CCS, is strong equivalence. It has the following definition:

(ST1)	$\alpha.P \xrightarrow{\alpha} P$	$\alpha \in Act$	
(ST2)	$P \xrightarrow{\alpha} P'$	implies	$P + Q \xrightarrow{\alpha} P'$ $Q + P \xrightarrow{\alpha} P'$
(ST3)	$P \xrightarrow{\alpha} P'$	implies	$P \mid Q \xrightarrow{\alpha} P' \mid Q$ $Q \mid P \xrightarrow{\alpha} Q \mid P'$
(ST4)	$P \xrightarrow{\alpha} P', Q \xrightarrow{\overline{\alpha}} Q'$	implies	$P \mid Q \xrightarrow{\tau} P' \mid Q'$
(ST5)	$P \xrightarrow{\alpha} P'$	implies	$P[f] \xrightarrow{f(\alpha)} P'[f]$
(ST6)	$P \xrightarrow{\alpha} P'$	implies	$P\backslash L \xrightarrow{\alpha} P'\backslash L \quad \alpha, \overline{\alpha} \notin L$

Figure 1: Operational semantics for CCS

Definition 2 (Strong bisimulation, strong equivalence)
A strong bisimulation *is a binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ such that $(P, Q) \in \mathcal{R}$ if for all $\alpha \in Act$*

1. *whenever $P \xrightarrow{\alpha} P'$, then there exists $Q' \in \mathcal{P}$ such that $Q \xrightarrow{\alpha} Q'$ and $(P', Q') \in \mathcal{R}$*
2. *whenever $Q \xrightarrow{\alpha} Q'$, then there exists $P' \in \mathcal{P}$ such that $P \xrightarrow{\alpha} P'$ and $(P', Q') \in \mathcal{R}$.*

The relation strong equivalence $\sim$ *is defined as $\sim = \bigcup\{\mathcal{R} \mid \mathcal{R}$ is a strong bisimulation $\}$ and two processes, P and Q are said to be* strongly bisimilar, *$P \sim Q$, if there exists a strong bisimulation $\mathcal{R}$ such that $(P, Q) \in R$.*

It is known from a general result that the relation $\sim$ is the largest strong bisimulation and an equivalence relation, hence the name strong equivalence.

Example Consider the processes $a.nil \mid b.nil$ and $a.b.nil + b.a.nil$ where $a \neq \overline{b}$. Let S be the relation

$$\{(a.nil \mid b.nil, a.b.nil + b.a.nil), (a.nil \mid nil, a.nil), (b.nil \mid nil, b.nil), (nil \mid nil, nil)\}.$$

Then S is a strong bisimulation and hence $a.nil \mid b.nil \sim a.b.nil + b.a.nil$.

Strong equivalence is a fairly fine equivalence, and perhaps equates more processes than one would like. One important aspect of an equivalence is the ability to abstract away from internal/invisible actions, namely to ignore them in certain contexts. This will enable us to use such an equivalence to check specifications and implementations for similar behaviour as described in the introduction. Observational equivalence is an equivalence that abstracts from internal actions which in CCS are represented by τ actions. Hence, as can be seen from the rule (ST4), communication between processes results in an invisible action. This is defined in a slightly different labelled transition system.

Definition 3 (Weak bisimulation, observational equivalence)
Define the following new transitions: $\Longrightarrow$ for $(\xrightarrow{\tau})^n, n \geq 0$, $\stackrel{a}{\Longrightarrow}$ for $\Longrightarrow\xrightarrow{a}\Longrightarrow$ and $\stackrel{m}{\Longrightarrow}$ where $m = a_1.a_2.\ldots.a_k, k \geq 0$ for $\stackrel{a_1}{\Longrightarrow}\stackrel{a_2}{\Longrightarrow}\cdots\stackrel{a_k}{\Longrightarrow}$.

Consider the labelled transition system

$$(\mathcal{P}, \mathcal{L}^*, \{\stackrel{m}{\Longrightarrow} \subseteq \mathcal{P} \times \mathcal{P} \mid m \in \mathcal{L}^*\}).$$

A (weak) bisimulation *is a binary relation* $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ *such that* $(P,Q) \in \mathcal{R}$ *if for all* $m \in \mathcal{L}^*$

1. *whenever* $P \stackrel{m}{\Longrightarrow} P'$, *then there exists* $Q' \in \mathcal{P}$ *such that* $Q \stackrel{m}{\Longrightarrow} Q'$ *and* $(P',Q') \in \mathcal{R}$
2. *whenever* $Q \stackrel{m}{\Longrightarrow} Q'$, *then there exists* $P' \in \mathcal{P}$ *such that* $P \stackrel{m}{\Longrightarrow} P'$ *and* $(P',Q') \in \mathcal{R}$.

The relation observational equivalence $\approx$ *is defined as* $\approx = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a weak bisimulation* $\}$. *Two processes* P *and* Q *are said to be* observationally equivalent, $P \approx Q$, *if there exists a (weak) bisimulation* $\mathcal{R}$ *such that* $(P,Q) \in \mathcal{R}$.

The relation $\approx$ is known to be the largest (weak) bisimulation and an equivalence relation.

Example The following is an example of how observational equivalence can be used to compare an implementation (in CCS) with its specification (in CCS). Consider the processes

$$(a.c.nil \mid \overline{c}.b.nil)\backslash\{c\} \quad \text{and} \quad a.b.nil.$$

The relation

$$\mathcal{R} = \{((a.c.nil \mid \overline{c}.b.nil)\backslash\{c\}, a.b.nil), ((c.nil \mid \overline{c}.b.nil)\backslash\{c\}, b.nil), ((nil \mid b.nil)\backslash\{c\}, b.nil), ((nil \mid nil)\backslash\{c\}, nil)\}$$

is a bisimulation and hence the two processes are observationally equivalent.

For reasons of space, I will only illustrate a few steps of the proof. Consider the first pair in the relation.

- $(a.c.nil \mid \overline{c}.b.nil)\backslash\{c\} \stackrel{a}{\Longrightarrow} (c.nil \mid \overline{c}.b.nil)\backslash\{c\}$ and the second element in the pair can match this as follows: $a.b.nil \stackrel{a}{\Longrightarrow} b.nil$ and clearly $((c.nil \mid \overline{c}.b.nil)\backslash\{c\}, b.nil) \in \mathcal{R}$.
- $(a.c.nil \mid \overline{c}.b.nil)\backslash\{c\} \stackrel{a}{\Longrightarrow} (nil \mid b.nil)\backslash\{c\}$ and the second element in the pair can match this as follows: $a.b.nil \stackrel{a}{\Longrightarrow} b.nil$ and clearly $((nil \mid b.nil)\backslash\{c\}, b.nil) \in \mathcal{R}$.
- $(a.c.nil \mid \overline{c}.b.nil)\backslash\{c\} \stackrel{ab}{\Longrightarrow} (nil \mid nil)\backslash\{c\}$ and the second element in the pair can match this as follows: $a.b.nil \stackrel{ab}{\Longrightarrow} nil$ and clearly $((nil \mid nil)\backslash\{c\}, nil) \in \mathcal{R}$.

To complete the proof, it is necessary to perform a similar analysis for $a.b.nil$, and then for each of the remaining pairs in the relation.

The two equivalences described above are the two standard interleaving equivalences for CCS. Other weaker interleaving equivalences have been defined for labelled transition systems, such as trace equivalence and failures equivalence, and these can also be applied to CCS processes. For an overview of these equivalences, see [vG90b].

I will now move onto equivalences for true concurrency, namely those equivalences which do not equate $a.nil \mid b.nil$ and $a.b.nil + b.a.nil$. All of the equivalences that I will describe here are defined on modified labelled transition systems, that is, on transition systems that have structured states or structured actions.

3.1 An equivalence based on location

In [BC91, BCHK91a], location information is added to CCS processes, so that the transitions have the form $P \stackrel{a}{\underset{u}{\longrightarrow}} P'$ where $u \in Loc^*$ and Loc is a set of atomic locations. The syntax of the processes remains the same except for the addition of a new type of prefixing, location prefixing $u :: P$, where $u \in Loc^*$. I will refer to this new set of processes as $\mathcal{P}_{Loc}$.

It is necessary to define operational semantics for these new processes and they are presented in Figure 2. The relations $\stackrel{a}{\underset{u}{\longrightarrow}}$ are defined to be the smallest relations satisfying the rules given in Figure 2. Note that the operational rules for CCS without locations are used to obtain τ transitions. The new rule (LT1) states that whenever a action occurs, it is assigned a location which is prefixed to the resulting process. (LT2) states

(LT1)	$a.P \xrightarrow[u]{a} u :: P$	$a \in \mathcal{L}$,	$u \in Loc^*$
(LT2)	$P \xrightarrow[u]{a} P'$	implies	$v :: P \xrightarrow[vu]{a} v :: P'$
(LT3)	$P \xrightarrow[u]{a} P'$	implies	$P + Q \xrightarrow[u]{a} P'$ $Q + P \xrightarrow[u]{a} P'$
(LT4)	$P \xrightarrow[u]{a} P'$	implies	$P \mid Q \xrightarrow[u]{a} P' \mid Q$ $Q \mid P \xrightarrow[u]{a} Q \mid P'$
(LT5)	$P \xrightarrow[u]{a} P'$	implies	$P[f] \xrightarrow[u]{f(a)} P'[f]$
(LT6)	$P \xrightarrow[u]{a} P'$	implies	$P \backslash L \xrightarrow[u]{a} P' \backslash L \quad \alpha, \overline{\alpha} \notin L$

Figure 2: Operational semantics for CCS with locations

that if P can perform an action at a location, then P prefixed by a location can perform an action at the location consisting of the original location plus the prefixed location. The remaining rules in Figure 2 are similar to those for CCS without locations. Note that there is no communication rule.

The idea behind locations is that each event occurs at a specific location and this location is related to the location of any event that occurred before in a sequential subprocess. So the locations of the actions performed by $a.b.c.nil$ will be related, whereas the locations of the actions performed by $a.nil \mid b.nil$ may be unrelated.

The equivalence, loose location bisimulation (called location bisimulation in [BC91, BCHK91a]) is defined as follows:

Definition 4 (Loose location bisimulation)
Define the following new transition: $\stackrel{a}{\underset{u}{\Longrightarrow}}$ *for* $\Longrightarrow \xrightarrow[u]{a} \Longrightarrow$. *Consider the modified labelled transition system defined as*

$$(\mathcal{P}_{Loc}, \mathcal{L}, Loc, \{\stackrel{a}{\underset{u}{\Longrightarrow}} \subseteq \mathcal{P}_{Loc} \times \mathcal{P}_{Loc} \mid a \in \mathcal{L}, u \in Loc^*\} \cup \{\stackrel{\tau}{\Longrightarrow} \subseteq \mathcal{P}_{Loc} \times \mathcal{P}_{Loc}\}).$$

Then a loose location bisimulation *is a binary relation* $\mathcal{R} \subseteq \mathcal{P}_{Loc} \times \mathcal{P}_{Loc}$ *such that* $(P, Q) \in \mathcal{R}$ *iff*

1. *whenever* $P \Longrightarrow P'$ *then there exists* $Q' \in \mathcal{P}_{Loc}$ *such that* $Q \Longrightarrow Q'$ *and* $(P', Q') \in \mathcal{R}$
2. *whenever* $Q \Longrightarrow Q'$ *then there exists* $P' \in \mathcal{P}_{Loc}$ *such that* $P \Longrightarrow P'$ *and* $(P', Q') \in \mathcal{R}$
3. *whenever* $P \stackrel{a}{\underset{u}{\Longrightarrow}} P'$ *then there exists* $Q' \in \mathcal{P}_{Loc}$ *such that* $Q \stackrel{a}{\underset{u}{\Longrightarrow}} Q'$ *and* $(P', Q') \in \mathcal{R}$.
4. *whenever* $Q \stackrel{a}{\underset{u}{\Longrightarrow}} Q'$ *then there exists* $P' \in \mathcal{P}_{Loc}$ *such that* $P \stackrel{a}{\underset{u}{\Longrightarrow}} P'$ *and* $(P', Q') \in \mathcal{R}$.

As usual, we use the largest loose location bisimulation relation, $\approx_l' = \bigcup\{\mathcal{R} \mid \mathcal{R}$ *is a loose location bisimulation*$\}$.

Example The two processes of the previous example are not loose location bisimulation equivalent. The first process has the following possible transitions where $l, m \in Loc$ and $l \neq m$:

$$(a.c.nil \mid \overline{c}.b.nil) \backslash \{c\} \stackrel{a}{\underset{l}{\Longrightarrow}} (l :: c.nil \mid \overline{c}.b.nil) \backslash \{c\} \stackrel{\tau}{\Longrightarrow} (l :: nil \mid b.nil) \backslash \{c\} \stackrel{b}{\underset{m}{\Longrightarrow}} (l :: nil \mid m :: nil)$$

Figure 3: Examples of causal trees

The second process must be able to match these transitions, so the first transition must be

$$a.b.nil \stackrel{a}{\underset{l}{\Longrightarrow}} l :: b.nil$$

however the only transition that can be performed after that is

$$l :: b.nil \stackrel{b}{\underset{lu}{\Longrightarrow}} l :: u :: nil$$

for some $u \in Loc^*$ and it is not possible for lu to equal m. Hence it can be seen that this equivalence distinguishes the fact that in the first process, the two actions can occur at separate locations, whereas in the second process, the locations are related.

I have now presented a true concurrency semantic equivalence in some detail, and this demonstration has shown how it differs from interleaving semantic equivalences. For reasons of space, I will not describe any further equivalences in detail, although I will discuss a few briefly, namely weak causal bisimulation and distributed bisimulation. I will also briefly refer to other equivalences that have been suggested in the literature. In the presentation of the equivalences, I will not discuss the extensions to CCS that are required, but will rather present the equivalences in terms of the general modified labelled transition systems on which they are defined. My aim in this section is to give insight into the way labelled transition systems are modified to allow for the definition of true concurrency semantic equivalences.

Causal equivalence

In [DD89, DD90], information about the causal structure of processes is used to distinguish nondeterministic sequentiality from concurrency. In [DD90], causal trees are defined and I will recast them here in terms of a labelled transition system.

Definition 5 (Weak causal bisimulation)
Let μ range over $(\mathcal{L} \times \mathcal{P}(N)) \cup \{\tau\}$. So each transition is either a τ action, or an transition $\langle a, B\rangle$ consisting of an action and a set of causes. The transition $\stackrel{\mu}{\Longrightarrow}$ is defined as $\Longrightarrow\stackrel{\mu}{\rightarrow}\Longrightarrow$, and let $\stackrel{m}{\Longrightarrow}$ for $m = \mu_1.\mu_2.\ldots.\mu_k, k \geq 0$ be $\stackrel{\mu_1}{\Longrightarrow}\stackrel{\mu_2}{\Longrightarrow}\cdots\stackrel{\mu_k}{\Longrightarrow}$.

A weak causal bisimulation *is defined to be a binary relation $\mathcal{R}$, such that $(P, Q) \in \mathcal{R}$ iff*

1. *whenever $P \stackrel{\mu}{\Longrightarrow} P'$ then there exists Q' such that $Q \stackrel{\mu}{\Longrightarrow} Q'$ and $(P', Q') \in \mathcal{R}$.*
2. *whenever $Q \stackrel{\mu}{\Longrightarrow} Q'$ then there exists P' such that $P \stackrel{\mu}{\Longrightarrow} P'$ and $(P', Q') \in \mathcal{R}$.*

Again, we take the largest weak causal bisimulation and denote it $\approx_c$.

The idea behind the transition labels $\langle a, B\rangle$ is that a is the action that occurs and B represents a set of causes that point back to the transitions that caused the current transition—in this manner it is possible to capture the causality in the computation. In Figure 3, two different causal trees are given—the leftmost tree represents $a.nil \mid b.nil$ and the rightmost $a.b.nil + b.a.nil$. In the first tree, the second level transitions (those on the lower branches) are independent of the previous transitions since they have empty cause sets. In the second tree, the transitions that form the lower branches are dependent on the previous transitions—this is indicated by the fact that the cause set is $\{1\}$.

Distributed equivalence

In [Cas88, CH89, Hen88], the structure of states in the labelled transition system is modified to have the following form: $P \stackrel{a}{\rightarrow} \langle P', P'' \rangle$. The first component is a local residual of the transition and the second component is the global residual.

Definition 6 (Distributed bisimulation)
Consider the following transition system: given a set of states $\mathcal{S}$, define $\mathcal{S}'$ to be $\mathcal{S} \cup \{\langle S_1, S_2 \rangle \mid S_1, S_2 \in \mathcal{S}\}$. Then the labelled transition system is

$$(\mathcal{S}', A, \{\stackrel{a}{\rightarrow} \subseteq \mathcal{S}' \times \mathcal{S}' \mid a \in \mathcal{S}'\}).$$

Then a distributed bisimulation, is defined to be a binary relation $\mathcal{R} \subseteq \mathcal{S}' \times \mathcal{S}'$ such that $(S,T) \in \mathcal{R}$ iff

1. *whenever $S \stackrel{a}{\rightarrow} \langle S', S'' \rangle$ then there exists $T', T'' \in \mathcal{S}'$ such that $T \stackrel{a}{\rightarrow} \langle T', T'' \rangle$ and $(S', T'), (S'', T'') \in \mathcal{R}$.*
2. *whenever $T \stackrel{a}{\rightarrow} \langle T', T'' \rangle$ then there exists $S', S'' \in \mathcal{S}'$ such that $S \stackrel{a}{\rightarrow} \langle S', S'' \rangle$ and $(S', T'), (S'', T'') \in \mathcal{R}$.*

As usual, we take the largest distributed bisimulation, and denote it $\sim_d$.

Other approaches

Other work on location-based equivalences is found in [BCHK92, San94, Ace94, Cas93, Kri91, Fan92]. In [Kie93a], a labelled transition system that allows for the capturing of information about local causality and global causality is defined. Specific instantiations of the general bisimulation defined give rise to three equivalences, one of which coincides with location equivalence [BCHK92] on CCS processes and one of which coincides with causal bisimulation [DD89] on CCS processes. The third equivalence combines both local and global causality.

Equivalences involving duration or time In [Hen88], it is assumed that actions have non-zero duration. A modified labelled transition system is defined where the label on the transition can indicate the start or finish of an action. Bisimulation equivalences have been defined for this transition system [Hen88, Hen91, GL91] based on the notion of ST-bisimulation which was originally defined on Petri nets [vGV87]. Labelled transition systems that have been modified to account for the passage of time are presented in [AM93, BB91].

Equivalences based on the structure of actions In the literature, there are a number of labelled transition systems defined where the set of actions has a particular structure. Examples include Synchronous CCS [Mil89] where actions are drawn from a commutative group, proved transitions systems where the actions contain information about the proofs (transition rules) used [BC88a, BC88b, Cas88] and structured transition systems which have algebraic structure on both states and transitions [FM90, CFM90, FGM91].

In [DDNM93], observation trees are defined, and parameterised bisimulation is defined with respect to an observation function on the observation trees. Observation trees form unlabelled transition systems with structured states that store details of the computation up until that state. In [MY92], observation trees are used to present a parametric approach to localities. A slightly different approach is taken in [DP92] where the transition system is labelled with proofs to form proved trees after which an observation function is used to extract the relevant information from these proofs. In [PY94], an equivalence based on read–write causality is defined using this approach.

Other papers of interest deal with modified labelled transition systems but do not involve equivalences. Examples are [LRT88] where distributed transition systems are defined with transitions labelled with sets of actions and the notion of a concurrent step is presented, but no equivalences are defined. In [Jef91], a partially ordered time domain is used and operational and denotational semantics are given. Although partially ordered time is used, it is not used to distinguish between concurrency and nondeterministic sequentiality.

4 Outline of proposed research

In this section, I will detail the research I intend to do. My research will focus on two topics:

- Comparison of true concurrency equivalence semantics defined on labelled transition systems
- Applicability of equivalence semantics.

This research will encompass the investigation and use of true concurrency equivalence semantics defined on labelled transition systems, and I believe that a coherent framework for the definition and application of these equivalence semantics will contribute to the knowledge and understanding of this area of research.

4.1 Comparison of true concurrency equivalence semantics

As mentioned above, many equivalences have been defined to deal with true concurrency issues. Two main types of equivalences are those based on causality and those based on location. I intend to develop a hierarchy/partial order (or if possible a lattice) to describe the relationship between different causal and location equivalences. The comparison of true concurrency equivalence will be conducted in three ways: first in terms of CCS processes (that is in terms of a specific labelled transitions system), second in terms of specific properties that can be used to distinguish between equivalences and finally, in terms of modified labelled transition systems.

I intend to focus on modified labelled transitions systems and equivalences that have been derived specifically to deal with true concurrency issues; however, it must be noted that labelled transition systems and equivalences that have been defined for other purposes such as timing or action refinement may also provide for the distinction of concurrency from nondeterministic sequentiality, and other properties important to true concurrency.

A hierarchy of process algebra equivalence semantics

A first approach to finding a hierarchy is based on CCS processes. This is not ideal because each process algebra developed for true concurrency semantics tends to have its own special features relating to the particular aspect of true concurrency that the researcher is trying to capture, and also this approach abstracts away from labelled transition systems. However, it is still possible to compare the equivalence on CCS processes, although most of the process algebras used for the definition of these equivalences have additional operators. This is because the pure CCS processes are still valid terms in these process algebras. The comparison will be done using the following notation:

$\approx$	observational equivalence [Mil89]
$\approx_d$	distributed bisimulation [Cas88]
$\approx_l$	(dynamic) location bisimulation [BCHK92, BCHK91b]
$\approx_l^l$	loose location bisimulation [BC91, BCHK91a]
$\approx_l^s$	static location bisimulation [Cas93]
$\approx_c$	weak causal bisimulation [DD89]
$\approx_{lc}$	local cause bisimulation [Kie93a]
$\approx_{gc}$	global cause bisimulation [Kie93a]
$\approx_{lg}$	local/global cause bisimulation [Kie93a]
$\approx_{rw}$	read/write bisimulation [PY94]

The equivalences $\approx$, $\approx_d$, $\approx_l$ and $\approx_c$ are defined in Section 3 and some others are described there briefly. The relationship between these equivalences (for finite CCS processes) is summarised in Figure 4. These results are drawn from [BC91, BCHK91a, BCHK91b, BCHK92, Cas88, Cas93, Kie93a, PY94] and work of my own.

The finer equivalences appear at the top of the lattice, so an arc from one equivalence to another appearing lower in the figure means that the higher equivalence is contained in the lower equivalence. Saying that

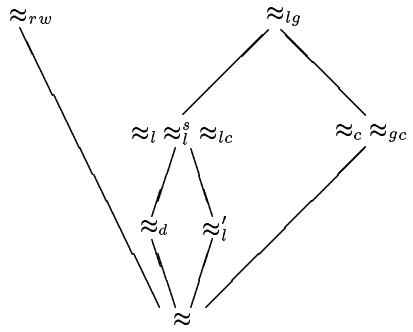

Figure 4: A hierarchy of equivalences for finite CCS processes

$\approx_{rw} \subset \approx$, for example, means that if $P \approx_{rw} Q$ for some P and Q, then $P \approx Q$. Equivalences that are not related by a path of downward arcs are incomparable, except that the relationship between $\approx_{rw}$ and $\approx_d$ has not yet been determined although it is known that $\approx_d \not\subset \approx_{rw}$. There are other equivalences in the literature such as ST-bisimulation $\approx_{ST}$ [Hen91] that need to be added to the hierarchy. It is also known that on finite restriction and renaming free CCS, $\approx_d$, $\approx_l$ and $\approx_l'$ coincide [BCHK91b], and that on finite restriction and renaming free CCS without communication, $\approx_l$, $\approx_c$ and $\approx_{ST}$ have the same axiomatisation [Kie93b].

Using properties to compare true concurrency equivalence semantics

A further approach to distinguishing the properties of equivalences is to identify certain processes that one may wish to equate or distinguish. This would allow one not only to say when one equivalence is finer than another, but also to give real examples of what type of processes the equivalences can distinguish. Examples of properties are as follows:

Definition 7
An equivalence $\asymp$ is said to distinguish

location	*iff*	$(a.c.b \mid d.\overline{c}.e) \setminus c \not\asymp (a.c.e \mid d.\overline{c}.b) \setminus c$
read-write causality	*iff*	$(a.c.b \mid d.\overline{c}.e) \setminus c \not\asymp (a.\overline{c}.b \mid d.c.e) \setminus c$
concurrency	*iff*	$a \mid b \not\asymp a.b + b.a$

The following statements hold:

- $\approx_l$, $\approx_l'$ and $\approx_l^s$ all distinguish location, but not read-write causality.
- $\approx_{rw}$ distinguishes read-write causality, but not location.
- $\approx_c$ doesn't distinguish location or read-write causality.
- All equivalences in Figure 4 except $\approx$ distinguish concurrency.

This is obviously preliminary. Part of this research will be to identify where properties can be used to meaningfully distinguish between different true concurrency semantic equivalences.

Modified labelled transition systems

In the previous section, true concurrency equivalences were compared with respect to CCS processes; this means that they were compared in terms of specific labelled transition systems—a different modified labelled transition system for each extension of CCS used. It is possible to do this because each CCS term is mapped to a specific state in each modified transition system.

To explain this in more detail with a specific example, consider loose location bisimulation [BC91, BCHK91a]. A new operator, location prefixing ($l :: P$) is added to the syntax of CCS. The operational semantics are the same except the prefixing rule is modified to $a.P \xrightarrow[l]{a} l :: P$ and a new rule is introduced: $P \xrightarrow[u]{a} P'$ then $v :: P \xrightarrow[vu]{a} v :: P'$. Hence it is possible to have terms without the use of the location prefixing operator. However, note that it is not possible for any transitions to have a target that is a pure CCS term (this is not strictly true, because τ-actions do not introduce the location prefixing operator). This holds for many other of the variants of CCS that have been introduced to deal with true concurrency. Thus, in the labelled transition systems, the states that represent pure CCS processes only have outward (non-τ) transitions. This differs from the case of pure labelled transition systems where such states do not occur. This makes the comparison using CCS processes unsatisfactory as it is not possible to compare the equivalences on other states within the labelled transition system. For this reason, it would be preferable to be able to compare these equivalences on a single transition system. Another reason to prefer a more unified approach is because equivalences that are identical on CCS processes may actually be related differently when examined in terms of a more general setting. For example, one equivalence may be finer than the other in the more general case, whereas viewed in terms of CCS, the equivalences identify the same processes.

As mentioned above, a labelled transition system is defined as a set of states, a set of actions and a collection of transition relations, one for each action. In a pure labelled transition system, actions do not have structure, although there may be some distinguished actions (for example τ), and the states do not have structure either (although a process is identified with a state, the process descriptor is not used in the definition of equivalences). However, for many of the causal and distributed equivalences, a modified labelled transition system is used where states and actions do have structure, and the equivalences are defined with respect to this structure.

For example, in location-based equivalences, the transition is decorated with location information, for example [BC91]. An example of structured states is the labelled transition system used to define distributed bisimulation [Cas88], where a successor state consists of two components—a local residual and a global residual.

Now that I have identified these different approaches, the question is whether it is possible to define a base modified labelled transition system on which all these different equivalences can be compared. Some of the approaches in the literature allow for parameterisation which in turn allows for comparison. Examples are the global/local cause transition system of [Kie93a] where bisimulation is parameterised by a function on local and global clauses, and parameterised location bisimulation where bisimulations are parameterised by a relation over locations [BCHK92]. In [DDNM93], bisimulation is parameterised by an observation function, and in [DP92], transitions are modified by an observation function before a standard definition of bisimulation is used. Most of these approaches capture some parts of the hierarchy given in Figure 4, although none of them capture it all. The following are some suggestions for a more general approach that I intend to investigate:

- take the union of all modified labelled transition systems—this can be done in a straightforward manner. However, this does not appear to advance things much further than the approach that uses CCS processes since equivalences will still be defined on different parts of the transition system.
- define a general enough transition system so that any other transition systems can be mapped into it in such a manner that equivalences are preserved. This approach would be a further generalisation of the approach taken in [BCHK92, Kie93a].
- Another possible approach is a parameterised transition system which can encapsulate the information in a given modified labelled transition system. Equivalences can then be defined on this in a uniform fashion; however, this does not necessarily facilitate the comparison of equivalences, although it may give a standard way for discussing them. This would be a similar approach to [DP92, DDNM93].

These ideas are preliminary and require much more investigation. To determine what would be a satisfactory approach, it is necessary to check whether the existing equivalences can be expressed in this new framework

and whether comparison is possible. It may also be possible to develop new equivalences of interest too, since it is a very general approach. Another aim of my research is to obtain an understanding of the amount of information or knowledge a particular labelled transition system contains. This may make it possible to define the finest equivalence on a specific transition system. A question that can be asked here is whether there are finer interesting equivalences—obviously syntactic equality of CCS processes is a finer equivalence, but is hardly interesting—and whether there is a general notion of bisimulation that can be applied in any labelled transition system.

4.2 Applicability of equivalence semantics

I intend to investigate issues of applicability. By this I mean, how one would decide which equivalence to use in a particular set of circumstances. This will relate back to the work involving distinguishing properties, as it is important to understand how equivalences are different and how these differences can be used.

Most researchers give informal arguments for why their equivalences are useful. For example, location equivalence is justified on the grounds that it distinguishes local deadlock [BCHK92]. In [Cas93], it is noted that location preorder is useful because an implementation is often not equivalent to its specification since the specification is likely to be more sequential. Other true concurrency equivalences are justified on the grounds that they have more veracity and that true concurrency semantics express the degree of concurrency in a computation, although little work appears to have been done in terms of quantifying this degree of concurrency. Research done on this topic in the area of process algebras includes a suggestion in [Mol89] on the application of decomposition for CCS processes, and [Gal93] where a concurrency measure is applied to CCS processes. Another important issue is that of abstraction from internal actions—are equivalences that do not abstract useful? It would appear that this property is important because most specifications and implementations differ by internal actions.

Issues of decidability and efficiency are also important—is the equivalence decidable and if yes, what is the complexity of the algorithm for checking that two processes are equivalent? A recent paper [KH94] shows that some of the true concurrency equivalences are decidable.

It is important to have a broad understanding of different equivalences and the circumstances in which they may be useful (or not useful). In this part of the research, it may be important to discover which equivalences are being used for the modelling and verification of concurrent systems within academia and industry. I will address this issue by surveying the relevant literature.

5 Conclusion

I have presented an overview of true concurrency equivalences for CCS and given examples of how these equivalences can be used. I have also outlined the research I intend to pursue over the next two years. This research involves the comparisons of different true concurrency equivalences. It is important that the differences between these equivalences are understood, so that one knows how they can be used and when it is appropriate to use them. I believe that this will contribute to our understanding of concurrency.

References

[Abr91] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, **92**, pp. 161–217, 1991.

[Ace94] L. Aceto. A static view of localities. *Formal Aspects of Computing*, **6**(2), pp. 201–222, 1994.

[AD89] A. Arnold and A. Dicky. An algebraic characterisation of transition system equivalences. *Information and Computation*, **82**, pp. 198–229, 1989.

[ADCRDR89] G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, (eds). *ICALP 88*, Lecture Notes in Computer Science **372**. Springer-Verlag, 1989.

[AM91] S. Abrasky and T.S.E. Maibaum, (eds). *TAPSOFT '91*, Lecture Notes in Computer Science **494**. Springer-Verlag, 1991.

[AM93] L. Aceto and D. Murphy. On the ill–timed but well–caused. In Best [Bes93], pp. 97–111.

[Arn90] A. Arnold, (ed). *CAAP 90*, Lecture Notes in Computer Science **431**. Springer-Verlag, 1990.

[Arn93] A. Arnold. verification and comparison of transition systems. In Gaudel and Jouannaud [GJ93], pp. 121–135.

[BB91] J.C.M. Baeten and J.A. Bergstra. Real space process algebra. In Baeten and Groote [BG91], pp. 96–110.

[BC88a] G. Boudol and I. Castellani. A non-interleaving semantics for CCS based on proved transitions. *Fundamenta Informatica*, **XI**, pp. 433–452, 1988.

[BC88b] G. Boudol and I. Castellani. Permutations of transitions: an event structure semantics for CCS and SCCS. In de Bakker et al. [dBdRR88], pp. 411–427.

[BC90] G. Boudol and I. Castellani. Three equivalent semantics for CCS. In Guessarian [Gue90], pp. 96–141.

[BC91] G. Boudol and I. Castellani. Observing localities. In Tarlecki [Tar91], pp. 93–102.

[BCHK91a] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. Technical Report 4/91, Computer Science, University of Sussex, 1991.

[BCHK91b] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. Technical Report 13/91, Computer Science, University of Sussex, 1991.

[BCHK92] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. In Cleaveland [Cle92], pp. 108–122.

[Bes93] E. Best, (ed). *CONCUR '93*, Lecture Notes in Computer Science **715**. Springer-Verlag, 1993.

[BG91] J.C.M. Baeten and J.F. Groote, (eds). *CONCUR '91*, Lecture Notes in Computer Science **527**. Springer-Verlag, 1991.

[BK90] J.C.M. Baeten and J.W. Klop, (eds). *CONCUR '90*, Lecture Notes in Computer Science **458**. Springer-Verlag, 1990.

[BS93] A.M. Borzyszkowski and S. Sokolowski, (eds). *MFCS '93*, Lecture Notes in Computer Science **711**. Springer-Verlag, 1993.

[Cas88] I. Castellani. Bisimulations for concurrency. Technical Report CST-51-88, PhD Thesis, Department of Computer Science, University of Edinburgh, 1988.

[Cas93] I. Castellani. Observing distribution in processes. In Borzyszkowski and Sokolowski [BS93], pp. 321–331.

[CFM90] A. Corradini, G.L. Ferrari, and U. Montanari. Transition systems with algebraic structure as models of computation. In Guessarian [Gue90], pp. 185–222.

[CH89] I. Castellani and M. Hennessy. Distributed bisimulations. *Journal of the ACM*, **36**(4), pp. 887–911, October 1989.

[Cle92] W.R. Cleaveland, (ed). *CONCUR '92*, Lecture Notes in Computer Science **630**. Springer-Verlag, 1992.

[dBdRR88] J.W. de Bakker, W.-P. de Roever, and G. Rozenberg, (eds). *Linear time, branching time and partial orders in logics and models for concurrency.* Lecture Notes in Computer Science **354**. Springer-Verlag, 1988.

[dBNT87] J.W. de Bakker, A.J. Nijman, and P.C. Treleaven, (eds). *PARLE '87*, Lecture Notes in Computer Science **259**. Springer-Verlag, 1987.

[DD89] P. Darondeau and P. Degano. Causal trees. In Ausiello et al. [ADCRDR89], pp. 234–248.

[DD90] P. Darondeau and P. Degano. Causal trees, interleaving + causality. In Guessarian [Gue90], pp. 239–255.

[DDNM88a] P. Degano, R. De Nicola, and U. Montanari. A distributed operational semantics for CCS based on condition/event systems. *Acta Informatica*, **26**, pp. 59–91, 1988.

[DDNM88b] P. Degano, R. De Nicola, and U. Montanari. Partial ordering descriptions and observations of nondeterministic concurrent processes. In de Bakker et al. [dBdRR88], pp. 438–466.

[DDNM90] P. Degano, R. De Nicola, and U. Montanari. A partial ordering semantics for CCS. *Theoretical Computer Science*, **75**, pp. 223–262, 1990.

[DDNM93] P. Degano, R. De Nicola, and U. Montanari. Universal axioms for bisimulations. *Theoretical Computer Science*, **114**, pp. 63–91, 1993.

[DM87] P. Degano and U. Montanari. A model of distributed systems based on graph rewriting. *Journal of the ACM*, **34**, pp. 411–449, 1987.

[DN87] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, **24**, 1987.

[DP92] P. Degano and C. Priami. Proved trees. In Kuich [Kui92], pp. 629–640.

[ES92] D. Etiemble and J.-C. Syre, (eds). *PARLE '92*, Lecture Notes in Computer Science **605**. Springer-Verlag, 1992.

[Fan92] J. Fanchon. Dynamic concurrent processes. In Etiemble and Syre [ES92], pp. 859–874.

[FGM91] G.L. Ferrari, R. Gorrieri, and U. Montanari. An extended expansion theorem. In Abrasky and Maibaum [AM91], pp. 29–48.

[FM90] G.L. Ferrari and U. Montanari. Toward the unification of models for concurrency. In Arnold [Arn90], pp. 162–176.

[Gal93] V.C. Galpin. Measuring concurrency in CCS. Master's thesis, Department of Computer Science, University of the Witwatersrand, 1993.

[GJ92] E. Goubault and T.P. Jensen. Homology of higher dimensional automata. In Cleaveland [Cle92], pp. 254–268.

[GJ93] M.-C. Gaudel and J.-P. Jouannaud, (eds). *TAPSOFT '93*, Lecture Notes in Computer Science **668**. Springer-Verlag, 1993.

[GL91] R. Gorrieri and C. Laneve. The limit of split_n-bisimulations for ccs agents. In Tarlecki [Tar91], pp. 170–180.

[Gol90] U. Goltz. CCS and Petri nets. In Guessarian [Gue90], pp. 334–357.

[Gue90] I. Guessarian, (ed). *Semantics of Systems of Concurrent Processes.* Lecture Notes in Computer Science **469**. Springer-Verlag, 1990.

[Hen88] M. Hennessy. Axiomatising finite concurrent processes. *SIAM Journal on Computing*, **17**(5), pp. 997–1017, October 1988.

[Hen91] M. Hennessy. A proof system for weak st-bisimulation over a finite process algebra. Technical Report 6/91, Computer Science, University of Sussex, 1991.

[HM94] M. Hagiya and J.C. Mitchell, (eds). *TACS '94*, Lecture Notes in Computer Science. Springer-Verlag, 1994.

[Jef91] A. Jeffrey. Abstract timed observation and process algebra. In Baeten and Groote [BG91], pp. 332–345.

[JP94] B. Jonsson and J. Parrow, (eds). *CONCUR '94*, Lecture Notes in Computer Science **836**. Springer-Verlag, 1994.

[KH94] A. Kiehn and M. Hennessy. On the decidability of non-interleaving process equivalences. In Jonsson and Parrow [JP94], pp. 18–33.

[Kie93a] A. Kiehn. Comparing locality and causality based equivalences. Revision of Technical Report TUM-I9132, TUM, Munich, 1993.

[Kie93b] A. Kiehn. Proof systems for cause based equivalences. In Borzyszkowski and Sokolowski [BS93], pp. 547–556.

[Kri91] P. Krishnan. Distributed CCS. In Baeten and Groote [BG91], pp. 393–407.

[Kui92] W. Kuich, (ed). *ICALP 92*, Lecture Notes in Computer Science **623**. Springer-Verlag, 1992.

[LRT88] K. Lodaya, R. Ramanujam, and P.S. Thiagarajan. A logic from distributed transition systems. In de Bakker et al. [dBdRR88], pp. 508–522.

[Mes90] J. Meseguer. Rewriting as a unified model of concurrency. In Baeten and Klop [BK90], pp. 384–400.

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

[Mol89] F. Moller. Axioms for concurrency. PhD Thesis, Report No. CST-59-89, Department of Computer Science, University of Edinburgh, 1989.

[MY89] U. Montanari and D.N. Yankelevich. An algebraic view of interleaving and distributed operational semantics for CCS. In Pitt et al. [PRD+89], pp. 5–20.

[MY92] U. Montanari and D.N. Yankelevich. A parametric approach to localities. In Kuich [Kui92], pp. 617–628.

[NS82] M. Nielsen and E.M. Schmidt, (eds). *ICALP 82*, Lecture Notes in Computer Science **140**. Springer-Verlag, 1982.

[Plo88] G. Plotkin. A structural approach to operational semantics. Report DAIMI FN-19, Computer Science Department, Aarhus University, 1988.

[PR88] A. Pnueli and R. Rosner. A framework for the synthesis of reactive models. In Vogt [Vog88], pp. 4–17.

[PRD+89] D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, (eds). *Category Theory and Computer Science*. Lecture Notes in Computer Science **389**. Springer-Verlag, 1989.

[PY94] C. Priami and D. Yankelvich. Read-write causality. In *MFCS '94*, 1994. to appear.

[San94] D. Sangiorgi. Locality and true-concurrency in calculi for mobile processes. In Hagiya and Mitchell [HM94], pp. 405–424.

[SNW93] V. Sassone, M. Neilsen, and G. Winskel. A classification of models for concurrency. In Best [Bes93], pp. 308–232.

[Tar91] A. Tarlecki, (ed). *MFCS '91*, Lecture Notes in Computer Science **520**. Springer-Verlag, 1991.

[Tau90] D. Taubner. Representing CCS programs by finite predicate nets. *Acta Informatica*, **27**, pp. 533–565, 1990.

[vG90a] R.J. van Glabbeek. *Comparative concurrency semantics and refinement of actions*. PhD thesis, Free University, Amsterdam, 1990.

[vG90b] R.J. van Glabbeek. The linear time—branching time spectrum. In Baeten and Klop [BK90], pp. 278–297.

[vG93] R.J. van Glabbeek. The linear time—branching time spectrum II (the semantics of sequential systems with silent moves). In Best [Bes93], pp. 66–81.

[vGV87] R. van Glabbeek and F. Vaandrager. Petri net models for algebraic theories of concurrency. In de Bakker et al. [dBNT87], pp. 224–242.

[Vog88] F.H. Vogt, (ed). *Concurrency 88*, Lecture Notes in Computer Science **335**. Springer-Verlag, 1988.

[Win82] G. Winskel. Event structures for CCS and related languages. In Nielsen and Schmidt [NS82], pp. 561–576.