A comparison of bisimulation-based semantic equivalences for noninterleaving behaviour over CCS processes

Vashti Galpin

Department of Computer Science, University of the Witwatersrand,
vashti@cs.wits.ac.za
http://www.cs.wits.ac.za/~vashti

Abstract

A number of extensions to the process algebra CCS (Calculus of Communicating Systems) have been proposed to deal with noninterleaving behaviour such as location and causality. The aim of the paper is to use existing and new comparison results to provide a hierarchy of these semantic equivalences over pure finite CCS terms. It is not possible to include some extensions in this hierarchy and the reasons for the exclusion are given.
Keywords: *comparison of semantic equivalences, process algebra, bisimulation, location, causality*
Computing Review Categories: *D.3.1, F.1.2, F.3.2, F.4.3*

1 Introduction

Process algebras provide mathematical models of concurrent behaviour and are used for the specification and verification of concurrent systems. A well-known process algebra is Milner's Calculus for Communicating Systems (CCS) [36, 37]. Many extensions to and modifications of CCS have been proposed to deal with different aspects of concurrent behaviour, such as location and causality. These types of behaviour are sometimes referred as noninterleaving, in contrast to the interleaving semantics of standard CCS. This paper considers the relationships between the noninterleaving semantic equivalences over finite pure CCS processes.

Process algebras consist of three components: a syntax to describe processes and their actions, operational semantics to describe the behaviour of the processes in terms of actions (in CCS, the behaviour is expressed as a labelled transition system) and semantic equivalences which equate processes that are understood to have the same behaviour (in CCS, bisimulation relates processes with the same behaviour with respect to the actions that can be performed).

This paper considers how the semantic equivalences (specifically bisimulations) of these extensions relate to each other with respect to CCS processes. As new extensions have been proposed, these have been compared with a subset of previous approaches, but the comparison is not complete. The aim of this paper is to present a complete comparison of some of these extensions by gathering together existing results and providing new results.

The approach taken here is *ad hoc* in the sense that each result is individually obtained using techniques specific to the equivalences being compared. Processes are compared over finite, pure CCS processes, namely over the processes consisting only of the operators defined for CCS, and only for those processes exhibiting finite behaviour. Since the comparison is based on results proved by a variety of techniques, a wide range of equivalences can be compared in this manner. However, there are some limitations, as will be discussed when considering equivalences that cannot be compared over these CCS terms.

Interleaving semantic equivalences defined on labelled transition systems have been extensively investigated by van Glabbeek [46, 47], with respect to linear and branching time, and abstraction from internal actions. He has developed a complete lattice of 11 different semantic notions which do not abstract from internal actions for finitely branching processes – the finest is bisimulation and the coarsest trace equivalence [46]. He has also investigated semantics which abstract from internal actions, and presents a hierarchy of 155 different equivalences [47]. The research presented here looks at noninterleaving equivalences and instead of using arbitrary transition systems, considers those transition systems generated for pure finite CCS terms by the process algebras.

Other research has approached the issue of comparison in less *ad hoc* manner, by presenting formal frameworks in which to do the comparison. A brief discussion of this work is presented here as related research. Each of these approaches has limitations as to what can be compared. Some research has attempted an overview of the different semantics and models. Category theoretical approaches are taken in [30, 35, 38, 44] and other approaches appear in [7, 17, 20, 24, 45]. Many of these approaches do not investigate semantic equivalences, but focus on comparing the behavioral semantics of processes [7, 30, 35, 38, 44]. The approaches most relevant are now discussed briefly.

Gorrieri and Laneve [25] compare equivalences over split action transition systems. They use transition system/transition preserving homomorphisms between transition systems to show that transition systems are bisimilar.

Some of the process algebras proposed in the liter-

$$\frac{}{\alpha.P \xrightarrow{\alpha} P} \qquad \frac{P \xrightarrow{\alpha} P'}{P+Q \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{Q+P \xrightarrow{\alpha} P'} \qquad \frac{P \xrightarrow{\alpha} P'}{P\backslash L \xrightarrow{\alpha} P'\backslash L} \quad \alpha, \overline{\alpha} \notin L$$

$$\frac{P \xrightarrow{\alpha} P'}{P \mid Q \xrightarrow{\alpha} P' \mid Q} \qquad \frac{P \xrightarrow{\alpha} P'}{Q \mid P \xrightarrow{\alpha} Q \mid P'} \qquad \frac{P \xrightarrow{a} P' \qquad Q \xrightarrow{\overline{a}} Q'}{P \mid Q \xrightarrow{\tau} P' \mid Q'} \qquad \frac{P \xrightarrow{\alpha} P'}{P[f] \xrightarrow{f(\alpha)} P'[f]}$$

Table 1: Rules for CCS

ature allow for the comparison of different equivalences over CCS processes. The observation trees of Degano *et al* [18] permit different types of observations to be made and hence it is possible to compare the different equivalences generated by these observations. The observation trees are also used in a parametric approach to compare equivalences using mixed orderings and partial orderings, and localities and causalities [40] and also used in extended transition systems to define and compare a number of different equivalences [29]. Kiehn defines a new transition system based on local and global causes [32, 33]. By changing the function used in the definition of bisimulation, different semantic equivalences can be compared based on local and global causes as well as ST-equivalence.

Results for the extended *tyft/tyxt* format give conditions under which the sum of two transition system specifications in the extended *tyft/tyxt* format can give coarser or finer bisimulations compared to those of the original systems [23, 24]. The AdS format allows for comparison by using different algebras to interpret the labels of the transition system specification [21].

In the next section, CCS is introduced and an indication is given of how extensions can be defined. In Section 3 the comparison is given in terms of a hierarchy in the spirit of similar work by van Glabbeek [46, 47], and reasons are given for the fact that some extensions cannot be included. Earlier versions of this work has appeared as [22, 23]

2 CCS and its extensions

In this section, a brief overview of Milner's Calculus for Communicating Systems (CCS) [36, 37] is given, as well as a brief discussion of the extensions to CCS that will be considered in this paper.

2.1 An overview of CCS

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 $\overline{\overline{a}} = a$. $\mathcal{A} \cup \overline{\mathcal{A}}$ is the set of actions ranged over by a. τ is a distinguished action not in $\mathcal{A} \cup \overline{\mathcal{A}}$ called the silent, internal or perfect action. Let $Act = \mathcal{A} \cup \overline{\mathcal{A}} \cup \{\tau\}$, ranged over by α. A relabelling function $f : Act \to Act$ is a function such that $f(\overline{a}) = \overline{f(a)}$ and $f(\tau) = \tau$. Processes $\mathcal{P}$ are generated by the grammar

$$P ::= 0 \mid \alpha.P \mid P + P \mid P|P \mid P\backslash L \mid P[f]$$

with $\alpha \in Act$, $L \subset \mathcal{A}$ and f a relabelling function.

The operational semantics of CCS are defined in terms of a labelled transition system, $(\mathcal{P}, Act, \{\xrightarrow{\alpha} \mid \alpha \in Act\})$, where the relation $\xrightarrow{\alpha}$ is defined to be the smallest relation satisfying the rules in Table 1. $\alpha.P$ represents a process that can perform an α action and then behave like P, $P + P$ represents the nondeterministic choice between two processes, $P \mid P$ represents two processes in parallel, $P\backslash L$ restricts the action a process can perform, and $P[f]$ renames the actions of a process. 0 is the process which can perform no actions.

Example 1 Consider $(a.b.0 + c.0 \mid \overline{a}.0)\backslash\{a\}$ which has the following transition built out of the rules

$$\frac{\dfrac{\dfrac{a.b.0 \xrightarrow{a} b.0}{a.b.0 + c.0 \xrightarrow{a} b.0} \qquad \overline{a}.0 \xrightarrow{\overline{a}} 0}{(a.b.0 + c.0) \mid \overline{a}.0 \xrightarrow{\tau} b.0 \mid 0}}{((a.b.0 + c.0) \mid \overline{a}.0)\backslash\{a\} \xrightarrow{\tau} (b.0 \mid 0)\backslash\{a\}}.$$

It also has the transition

$$\frac{\dfrac{\dfrac{c.0 \xrightarrow{c} 0}{a.b.0 + c.0 \xrightarrow{c} 0}}{(a.b.0 + c.0) \mid \overline{a}.0 \xrightarrow{c} 0 \mid \overline{a}.0}}{((a.b.0 + c.0) \mid \overline{a}.0)\backslash\{a\} \xrightarrow{c} (0 \mid \overline{a}.0)\backslash\{a\}}.$$

The semantic equivalence of interest in this paper is (weak) bisimulation. This equivalence abstracts from internal actions in the sense that they do not have to be matched when considering bisimilar processes. First, let $t = \alpha_1 \ldots \alpha_n \in Act^*$ then $\stackrel{t}{\Rightarrow} = (\xrightarrow{\tau})^* \xrightarrow{\alpha_1} (\xrightarrow{\tau})^* \ldots (\xrightarrow{\tau})^* \xrightarrow{\alpha_n} (\xrightarrow{\tau})^*$. Note that $\stackrel{\varepsilon}{\Rightarrow} = (\xrightarrow{\tau})^*$ where ε is the empty string of actions. The following definition is chosen for reasons of elegance, but other formulations of this equivalence exist [37].

Definition 1 A *(weak) bisimulation* is a binary relation $\mathcal{R} \subseteq \mathcal{P} \times \mathcal{P}$ such that $(P, Q) \in \mathcal{R}$ only if for all $s \in Act^*$

1. $P \stackrel{s}{\Rightarrow} P'$ implies $\exists Q'$, $Q \stackrel{s}{\Rightarrow} Q'$ and $(P', Q') \in \mathcal{R}$
2. $Q \stackrel{s}{\Rightarrow} Q'$ implies $\exists P'$, $P \stackrel{s}{\Rightarrow} P'$ and $(P', Q') \in \mathcal{R}$.

P and Q are *(weakly) bisimilar*, $P \approx Q$, if there exists a (weak) bisimulation $\mathcal{R}$ such that $(P, Q) \in \mathcal{R}$.

Using this definition, $a.0 \mid b.0 \approx a.b.0 + b.a.0$. This identification of a process with concurrency with a process that has no concurrency but sequential nondeterminism is

the reason that bisimulation is called interleaving. The sequential process $a.b.0 + b.a.0$ represents all possible interleavings of the actions in the parallel process $a.0 \mid b.0$. By contrast with this, many of the extensions to CCS, particularly those discussed in this paper, do not identify these two processes and hence are known as noninterleaving or are sometimes described as having true concurrency. Also $(a.b.0 \mid \bar{b}.c.0)\backslash\{b\} \approx a.c.0$ and $\tau.a.0 \approx a.0 \approx a.\tau.0$.

2.2 Extensions to CCS

This section will give an indication of how CCS can be modified to deal with different aspects of concurrent behaviour, but due to space constraints it is not possible to give full details of each process algebra. The reader is referred to the original papers. Each semantic equivalence for a CCS extension in the hierarchy will be briefly discussed. Most of the extensions involve the introduction of new operators and the modification of the labelled transition system, as well as the modification of definition of bisimulation to deal with the new labels of the transition system.

Location bisimulation ($\approx_l$) [9]

As an example, CCS with locations [9] will be considered in more detail than the others. In this case, additional syntax is added, that of location prefix. Let $\mathcal{L}$ be a set of locations distinct from the actions, then for $l \in \mathcal{L}$, $l :: P$ is a process. This describes the idea that the actions of P are located at l. The rules of the operational semantics are also modified. The rule for prefix is replaced with

$$\frac{}{\alpha.P \xrightarrow[l]{\alpha} l :: P} \quad l \in \mathcal{L}$$

to associate a location with each process. A new rule is introduced for processes prefixed with locations

$$\frac{P \xrightarrow[u]{\alpha} P'}{l :: P \xrightarrow[lu]{\alpha} l :: P'}$$

where u is a string of locations. Note that transitions of the labelled transition system now carry both action and location information; the transitions in all other rules need to be modified to carry both action and location information. The only transitions without locations are τ transitions and these are obtained from the original rules for CCS with the addition of a rule for the location prefix operator. Also the transition system now has many transitions from a process $\alpha.P$, one for each possible location.

The notion of bisimulation also needs to be modified to take account of the fact that there are both actions and locations, and it is changed to require that a transition with an action and string of locations is matched by a transition with the same action and string of locations – this is called location bisimulation.

This extension to CCS takes into account location information; essentially, actions are associated with a string of locations that represent which actions have occurred previously at that location. This distinguishes actions that occur in a process exhibiting sequential nondeterminism to those that occur in a process with concurrency, and hence $a.b.0 + b.a.0 \not\approx_l a.0 \mid b.0$

Example 2 The process $a.b.0 \mid c.0$ has the transition

$$\frac{a.b.0 \xrightarrow[l_1]{a} l_1 :: b.0}{a.b.0 \mid c.0 \xrightarrow[l_1]{a} l_1 :: b.0 \mid c.0}$$

and the resulting process has the transitions

$$\frac{\dfrac{b.0 \xrightarrow[l_2]{b} l_2 :: 0}{l_1 :: b.0 \xrightarrow[l_1 l_2]{b} l_1 :: l_2 :: 0}}{l_1 :: b.0 \mid c.0 \xrightarrow[l_1 l_2]{b} l_1 :: l_2 :: 0 \mid c.0}$$

and

$$\frac{c.0 \xrightarrow[l_3]{c} l_3 :: 0}{l_1 :: b.0 \mid l_3 :: 0 \xrightarrow[l_3]{c} l_1 :: b.0 \mid l_3 :: 0}$$

Loose location bisimulation ($\approx_{ll}$) [8]

Loose location bisimulation has a similar definition to that of location bisimulation, but the underlying operational semantics has a different rule for action prefix where any string of locations (including the empty string), and not just a single location can be associated with an action.

Static location bisimulation ($\approx_l^s$) [11]

This is based on transitions composed of actions and locations as are the previous two notions. Locations are only introduced by the location prefix operator (hence the transition system only has one transition from a process $\alpha.P$ which has the empty string as its location). This requires a more complex notion of bisimulation based on families of bisimulations to match up locations between two processes.

Equivalence with location observations ($\approx_{loc}$) [39]

This is a parametric approach (in the sense that it is parameterised on different types of observations) based on observation trees where location-oriented observations are made after which a standard form of bisimulation appropriate to observation trees is applied.

Distributed bisimulation ($\approx_d$) [10, 12]

Transitions have the form $P \xrightarrow{a} \langle P', P'' \rangle$ where P' is the local residual and P'' is the global residual, in the sense that P' represents what the local component changes to after the transition and P'' represents what the whole process changes to after the transition. Bisimulation requires that the local residuals are bisimilar and the global residuals are bisimilar.

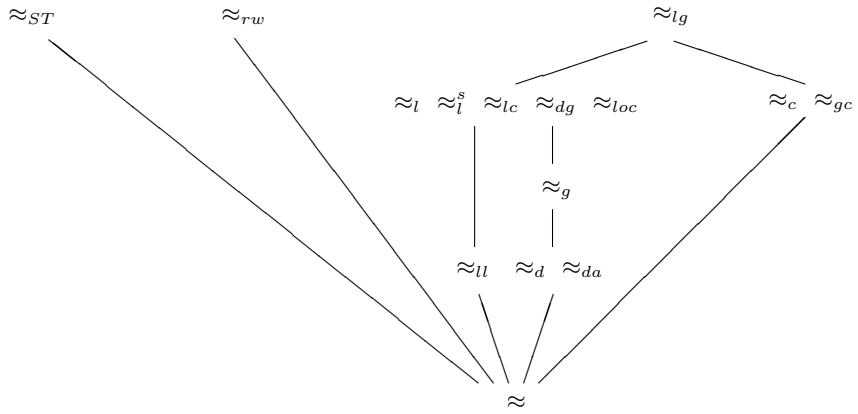

Figure 1: A hierarchy of equivalences for finite CCS processes

K-grapes distributed bisimulation ($\approx_{da}$) [14]
Generalised distributed bisimulation ($\approx_g$) [14]
Distributed grapes equivalence/maximal distribution equivalence ($\approx_{dg}$) [13, 14]

All these approaches rely on decomposing CCS processes into sequential components called grapes. The bisimulations are defined with respect to sets of grapes which represent processes.

Weak causal bisimulation ($\approx_c$) [15, 16]

The transition systems keeps track of which previous actions the current action is causally dependent on; either as a result of sequential behaviour or communication between processes. Transitions have the form $P \xrightarrow{\langle a,K\rangle} P'$ where K records the previous actions a is dependent on. Bisimulation requires that actions have the same dependencies.

Local/global cause bisimulation ($\approx_{lg}$) [32]
Local cause bisimulation ($\approx_{lc}$) [32]
Global cause bisimulation ($\approx_{gc}$) [32]

Each transition is labelled with an action, a set of local causes, a set of global causes and the current cause. Local causes are due to actions that occurred in the sequential component from which the current action came, and global causes are understood to come from an action in any component. The bisimulation equivalences are parameterised by a function which extracts the appropriate information from the causes on a transition.

Read-write bisimulation ($\approx_{rw}$) [43]

The proved transition system (a transition system where each transition is labelled with the proof that was used to obtain the transition) is translated into a labelled transition system labelled with actions and cause sets. This transformation determines causality with respect to sequential behaviour, or asymmetric communication. Causality is obtained from asymmetric communication by considering the flow of information – an output action $\overline{a}$ is viewed as causing the corresponding input action a. Bisimulation matches on actions and sets of causes.

ST-bisimulation ($\approx_{ST}$) [27]

In the transition system, an action a is understood to be split into two parts – the start of the action $s(a)$ and the finish of the action $f(a)$. Bisimulation requires that start actions match with start actions and finish actions with finish actions.

3 Comparison over pure CCS processes

In this section, the results for the comparison of extensions to CCS are presented. These results will yield a hierarchy of equivalences with respect to pure CCS processes. Additionally, examples of process algebras which cannot be included in this hierarchy will be given and discussed.

As mentioned above, a number of extensions to CCS have been proposed which permit noninterleaving semantics. It is possible to compare the equivalences on finite CCS processes, although most of the process algebras used for the definition of these equivalences have additional operators. This comparison is possible because pure CCS processes are still valid processes in these process algebras and they also display the specific concurrent behaviour for which the process algebra is designed. Many authors when presenting a new process algebra based on CCS, compare their process algebra with a few others using pure CCS terms. However, many gaps remain and it is important to fill in these gaps and assemble these results together. Because many of the existing comparisons are over pure CCS terms, it makes sense to continue the comparison in this fashion.

3.1 The hierarchy

In the spirit of van Glabbeek, a hierarchy of equivalences that are finer than Milner's observation equivalence is developed, by assembling results from the literature and providing additional counter-examples. The equivalences described in Section 2.2 appear in the hierarchy and Figure 1 displays the relationship between these equivalences.

A	$(c.a \mid b.(\overline{c} + a))\backslash\{c\}$	$(b.c \mid \overline{c}.a)\backslash\{c\}$
B	$(e.(c.\Sigma_a + d.\Pi_a + \overline{c}.\Sigma_b + \overline{d}.\Pi_b) \mid (\overline{d}.\Sigma_b + \overline{c}.\Pi_b + c.\Sigma_a + d.\Pi_a))\backslash\{c,d\}$	$(e.(c.\Sigma_a + d.\Pi_a + \overline{c}.\Sigma_b + \overline{d}.\Pi_b) \mid (\overline{c}.\Sigma_b + \overline{d}.\Pi_b + d.\Sigma_a + c.\Pi_a))\backslash\{c,d\}$
C	$(a.e.c \mid b.\overline{e}.d)\backslash\{e\}$	$(a.e.d \mid b.\overline{e}.c)\backslash\{e\}$
D	$(a.b \mid c.d)$	$(a.(e.b + b) \mid c.(\overline{e}.d + d))\backslash\{e\}$
E	$(a.c + b.d \mid \overline{c}.b + \overline{d}.a)$	$(a.c + b.d \mid \overline{c}.b + \overline{d}.a) + a.b$
F	$(a.c + b.d \mid \overline{c}.b + \overline{d}.a)$	$(a.c + b.d \mid \overline{c}.b + \overline{d}.a) + (a \mid b)$
G	$(a.\overline{b} \mid b.c)\backslash\{b\}$	$a.c$
H	$(a.e.c \mid b.\overline{e}.d)\backslash\{e\}$	$(a.\overline{e}.c \mid b.e.d)\backslash\{e\}$
I	$(a.\overline{b}.d \mid b.c)\backslash\{b\} + (c.b \mid \overline{b}.a.d)\backslash\{b\}$	$(a.d \mid c)$
J	$(a \mid b) + (a.c \mid \overline{c}.b)\backslash\{c\}$	$(a \mid b)$
K	$(a.b \mid \overline{b}.c) + a.c$	$(a.b \mid \overline{b}.c)$
L	$a.(b + \tau.c) + a.c$	$a.(b + \tau.c)$
M	$(e.j.(m.a.p \mid \overline{n}.c) \mid (\overline{j}.f.s.(m.a.b \mid \overline{p}.c) \mid ((m.a.n \mid \overline{b}.c) \mid \overline{s}.\overline{m})))\backslash\{b,j,m,n,p,s\}$	$(e.j.(m.a.p \mid \overline{n}.c) \mid (\overline{j}.f.s.(m.a.b \mid \overline{b}.c) \mid ((m.a.n \mid \overline{p}.c) \mid \overline{s}.\overline{m})))\backslash\{b,j,m,n,p,s\}$

where $\Sigma_a = a_1 a_2 + a_2 a_1$ and $\Pi_a = a_1 \mid a_2$.

Table 2: CCS processes used for comparison

A path from an equivalence to one lower in the diagram means that the higher equivalence is contained in the lower one. This means that the lower equivalence equates the same processes that the higher one does, but also equates other processes as well, namely it is coarser[1]. Note that observation equivalence is the coarsest of all these equivalences as would be expected. The equivalences that are not connected by a downwards path are incomparable; that is, it is possible to find two pairs of processes such that the first pair is equated by the first equivalence but not the second, and such that the second pair is equated by the second equivalence, but not the first. It is also known that on finite restriction and renaming free CCS, $\approx_d$, $\approx_l$ and $\approx_{ll}$ coincide [9], as do $\approx_d$, $\approx_{da}$, $\approx_g$ and $\approx_{dg}$ [14]; and that on finite restriction and renaming free CCS without communication, $\approx_l$, $\approx_c$ and $\approx_{ST}$ have the same axiomatisation [31]. Aceto and Murphy show that their timed bisimulation coincides with these equivalences on this subset [4].

Figure 1 shows that ST-bisimulation and read-write bisimulation are not comparable and do not relate to any of the other equivalences. Local/global cause bisimulation is finer than any of the location bisimulations and causal bisimulation. It is interesting that there appear to be a number of possible approaches to obtaining a location-based bisimulation, and that there are many different ways to obtain the location bisimulation of Boudol *et al*. It is to be expected that unlike the work of van Glabbeek where there is one finest equivalence, here there are likely to be many, as there is little utility in combining the features of the incomparable equivalences to construct a finer one.

The details of the comparison are given in Tables 2 and 3. These tables are constructed out of existing results in the literature, new counter-examples and rules of logical implication. The two tables can be interpreted as follows

- Table 2 contains pairs of processes that are used to show that certain equivalences are incomparable. Note that the 0 operator has been omitted in these processes.

- Table 3 describes the relationship between each pair of equivalences. For each pairwise comparison, there is a block in the table. In the centre of each block, there is a symbol giving the relationship – $\#$ indicates when the two equivalences in question are incomparable. As an example, the block at the fourth column and third row of Table 3 is to be read as $\approx_g \subset \approx_{da}$. If the result of the comparison comes from the literature, there will be a citation at the bottom of the block. Finally, if the two equivalences are incomparable, there may be two letters at the top of the block, each of which refer to a pair of processes in Table 2. The first pair is equated by the equivalence that appears in the column, and not equated by the equivalence that appears in the row; and the second pair is not equated by the equivalence that appears in the column, and equated by the equivalence that appears in the row. These pairs of letters do not appear for all incomparable equivalences – since there

[1] An equivalence that makes more equations than another equivalence is said to be *coarser*, and an equivalence that makes fewer equations than another equivalence is said to be *finer*. This usage is consistent with the fact that a finer equivalence results in more equivalence classes than one that is coarser.

are a number of identical equivalences in the table, letter pairs are only given to one (usually the first one) in a group of equivalences that are the same.

Example 3 For example, consider $\approx_l$ and $\approx_{ST}$. They are not comparable. $\approx_l$ does not equate $(a.\bar{b}.0 \mid b.c.0)\setminus\{b\}$ and $a.c$ (the pair of processes labelled G) since after two transitions the first process can become $(l_1 :: 0 \mid c.0)\setminus\{b\}$. This process has the transition

$$(l_1 :: 0 \mid c.0)\setminus\{b\} \xrightarrow[l_2]{c} (l_1 :: 0 \mid l_2 :: 0)\setminus\{b\}$$

The second process cannot match this transition since from $l_1 :: c.0$ the only transitions have a string of two locations.

$$l_1 :: c.0 \xrightarrow[l_1 l_2]{c} l_1 :: l_2 :: 0$$

However, $\approx_{ST}$ does equate these processes as both can only perform the sequence of actions $s(a)$, $f(a)$, $s(c)$, $f(c)$ (abstracting from τ actions). The pair of processes labelled J are equated by $\approx_l$ since no location information is conveyed between the actions on different sides of the parallel operator, hence the actions resulting from $(a.c.0 \mid \bar{c}.b.0)\setminus\{c\}$ can be matched by those of $(a.0 \mid b.0)$. These processes are not equated by $\approx_{ST}$ since in $(a.c.0 \mid \bar{c}.b.0)\setminus\{c\}$ it is not possible to do a $s(b)$ action before an $f(a)$, but in $(a.0 \mid b.0)$ it is possible.

3.2 Other semantic equivalences

There are a number of equivalences based on extensions to CCS that are not suitable to add to this hierarchy using this type of comparison.

- For some extensions to CCS, pure CCS processes do not exhibit noninterleaving behaviour or only a subset of possible noninterleaving behaviour. Examples are distributed CCS [34], multiprocessor CCS [34] and the process algebra with a dynamic parallel operator developed by Fanchon [19].

- The comparison here has been done for semantic equivalences which abstract from internal actions. Some extensions to CCS only have a strong equivalence which does not abstract from these actions. Examples are distributed CCS [34], multiprocessor CCS [34] and the split-action bisimulations of Gorrieri and Laneve [25]. In the latter, τ is split and it is not clear how a weak semantic equivalence can be defined.

- Some process algebras have operators and hence processes that differ from those of CCS, therefore comparison is not possible. For example, the ST-equivalences of Aceto and Hennessy are based on a process algebra consisting of actions (as opposed to prefixed actions) and a termination predicate [2, 3]. In Murphy's algebra of located processes, a subset of CCS is used where processes represent parallel combinations of sequential process [42]. However, there are some comparison results proved for these process algebras.
  - In the work of Aceto, a subset of CCS processes is used in a process algebra with static localities, where the static operators of CCS, namely parallel, renaming and relabelling, can only appear at the highest level of a term [1]. These processes are called *nets of automata*. Aceto has shown that on the set of nets, his equivalence coincides with location bisimulation [9]. Moreover, this work has been generalised, and the generalisation $\approx_l^s$ [11] is included in the hierarchy.
  - In the work of Mukund and Nielsen, CCS with guarded sums is given operational semantics based on asynchronous transition systems. Hence the set of CCS processes is only a subset of the pure CCS processes [41]. The equivalence defined coincides with the equivalence of Aceto [1] on nets of automata and the authors conjecture that their equivalence coincides with location bisimulation [9] on CCS with guarded sums.

- Also note that for many other process algebras that are extensions of CCS, the equivalence over pure CCS processes is the same as weak bisimulation over CCS processes. This is particularly true for those extensions based on time, priorities and probabilities. So this method of comparison does not work, since it is not powerful enough to distinguish meaningful differences.

- Additionally, note that some equivalences for the process algebras are not based on bisimulation such as applying testing equivalence to split actions [28], and hence are not included here.

4 Conclusion

This paper has taken 14 (noninterleaving) semantic equivalences from the literature and done a complete comparison of them using existing results and new counter-examples. This has resulted in a hierarchy of these equivalences. All of these equivalences are finer than weak bisimulation, and many are incomparable.

The method used of comparing over pure, finite CCS processes has utility for these semantic equivalences since the process algebras under consideration demonstrate all their possible noninterleaving behaviour over these terms. Other semantic equivalences have been omitted for reasons of not being defined on all appropriate CCS processes, or because not all noninterleaving behaviour is displayed by these processes.

In conclusion, this paper has demonstrated the relationship between these equivalences in a manner which has not been done before, and shown the utility of the approach of doing the comparison over pure CCS terms.

	$\approx$	$\approx_d$	$\approx_{da}$	$\approx_g$	$\approx_{ll}$	$\approx_l$	$\approx_l^s$	$\approx_{lc}$	$\approx_{dg}$	$\approx_{loc}$	$\approx_c$	$\approx_{gc}$	$\approx_{lg}$	$\approx_{rw}$	$\approx_{ST}$
$\approx$	—	$\subset$ [10]	$\subset$	$\subset$	$\subset$ [8]	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$ [32]	$\subset$	$\subset$ [43]	$\subset$ [27]
$\approx_d$		—	$=$ [14]	$\subset$ [14]	A,B # [8]	$\subset$ [9]	$\subset$	$\subset$	$\subset$	$\subset$	E,F #	#	$\subset$	A,H #	A,F #
$\approx_{da}$			—	$\subset$	A,B #	$\subset$	$\subset$	$\subset$	$\subset$	$\subset$	E,F #	#	$\subset$	A,H #	A,F #
$\approx_g$				—	A,M # [14]	$\subset$	$\subset$	$\subset$	$\subset$ [14]	$\subset$	E,F #	#	$\subset$	A,H #	A,J #
$\approx_{ll}$					—	$\subset$ [9]	$\subset$	$\subset$	$\subset$	$\subset$	C,D # [8]	#	$\subset$	G,H #	G,J #
$\approx_l$						—	$=$ [11]	$=$ [32]	$=$ [13]	$=$ [39]	E,F # [9]	#	$\subset$	G,H # [43]	G,J #
$\approx_l^s$							—	$=$	$=$	$=$	#	#	$\subset$	#	#
$\approx_{lc}$								—	$=$	$=$	#	#	$\subset$ [32]	#	#
$\approx_{dg}$									—	$=$	#	#	$\subset$	#	#
$\approx_{loc}$										—	#	#	$\subset$	#	#
$\approx_c$											—	$=$ [32]	$\subset$	# [43]	#
$\approx_{gc}$												—	$\subset$ [32]	G,H #	D,K #
$\approx_{lg}$													—	I,H #	G,L #
$\approx_{rw}$														—	H,I #
$\approx_{ST}$															—

Table 3: Summary of relationship between equivalences

References

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

[2] L. Aceto and M. Hennessy. Towards action refinement in process algebras. *Information and Computation*, 103:204–269, 1993.

[3] L. Aceto and M. Hennessy. Adding action refinement to a finite process algebra. *Information and Computation*, 115:179–247, 1994.

[4] L. Aceto and D. Murphy. Timing and causality in process algebra. *Acta Informatica*, 33:317–350, 1996.

[5] J.C.M. Baeten and J.W. Klop, editors. *CONCUR '90*, number 458 in Lecture Notes in Computer Science. Springer-Verlag, 1990.

[6] A.M. Borzyszkowski and S. Sokolowski, editors. *MFCS '93*, number 711 in Lecture Notes in Computer Science. Springer-Verlag, 1993.

[7] G. Boudol and I. Castellani. Flow models of distributed computations—three equivalent semantics for CCS. *Information and Computation*, 114:247–314, 1994.

[8] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. Observing localities. *Theoretical Computer Science*, 114:31–61, 1993.

[9] G. Boudol, I. Castellani, M. Hennessy, and A. Kiehn. A theory of processes with localities. *Formal Aspects of Computing*, 6(2):165–200, 1994.

[10] I. Castellani. *Bisimulations for concurrency*. PhD thesis, Department of Computer Science, University of Edinburgh, 1988. LFCS report ECS-LFCS-88-51.

[11] I. Castellani. Observing distribution in processes: static and dynamic localities. *International Journal of Foundations of Computer Science*, 6:353–393, 1995.

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

[13] A. Corradini and R. De Nicola. Distribution and locality of concurrent systems. In S. Abiteboul and E. Shamir, editors, *ICALP '94*, number 820 in Lecture Notes in Computer Science, pages 154–165. Springer-Verlag, 1994.

[14] A. Corradini and R. De Nicola. Locality based semantics for process algebras. *Acta Informatica*, 34:291–324, 1997.

[15] P. Darondeau and P. Degano. Causal trees. In G. Ausiello, M. Dezani-Ciancaglini, and S. Ronchi Della Rocca, editors, *ICALP 88*, number 372 in Lecture Notes in Computer Science, pages 234–248. Springer-Verlag, 1989.

[16] P. Darondeau and P. Degano. Causal trees, interleaving + causality. In Guessarian [26], pages 239–255.

[17] R. De Nicola. Extensional equivalences for transition systems. *Acta Informatica*, 24:211–237, 1987.

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

[19] J. Fanchon. Dynamic concurrent processes. In D. Etiemble and J.-C. Syre, editors, *PARLE '92*, number 605 in Lecture Notes in Computer Science, pages 859–874. Springer-Verlag, 1992.

[20] T. Fernando. Comparative transition system semantics. In E. Börger, G. Jäger, H. Kleine Büning, S. Martini, and M.M. Ritchter, editors, *CSL '92*, number 702 in Lecture Notes in Computer Science, pages 146–166. Springer-Verlag, 1993.

[21] G.L. Ferrari and U. Montanari. Parameterized structured operational semantics. *Fundamenta Informatica*, 34:1–31, 1998.

[22] V.C. Galpin. True concurrency equivalence semantics: an overview. In *Proceedings of the Ninth Annual Masters and PhD Students on Computer Science Conference*, pages 46–60, Stellenbosch University Seminar Centre, Stellenbosch, South Africa, 14–15 December 1994.

[23] V.C. Galpin. *Equivalence semantics for concurrency: comparison and application*. PhD thesis, Department of Computer Science, University of Edinburgh, 1998. LFCS report ECS-LFCS-98-397. `http://www.dcs.ed.ac.uk/lfcsreps/`.

[24] V.C. Galpin. Comparison of process algebra equivalences using formats. In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *ICALP '99*, number 1644 in Lecture Notes in Computer Science, pages 341–350. Springer-Verlag, 1999.

[25] R. Gorrieri and C. Laneve. The limit of split_n-bisimulations for CCS agents. In A. Tarlecki, editor, *MFCS '91*, number 520 in Lecture Notes in Computer Science, pages 170–180. Springer-Verlag, 1991.

[26] I. Guessarian, editor. *Semantics of Systems of Concurrent Processes*, number 469 in Lecture Notes in Computer Science. Springer-Verlag, 1990.

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

[28] M. Hennessy. Concurrent testing of processes. *Acta Informatica*, 32:509–543, 1995.

[29] P. Inverardi, C. Priami, and D. Yankelevich. Automatizing parametric reasoning on distributed concurrent systems. *Formal Aspects of Computing*, 6:676–695, 1994.

[30] S. Kasangian, A. Labella, and A. Pettorossi. Observers, experiments and agents: a comprehensive approach to parallelism. In Guessarian [26], pages 393–407.

[31] A. Kiehn. Proof systems for cause based equivalences. In Borzyszkowski and Sokolowski [6], pages 547–556.

[32] A. Kiehn. Comparing locality and causality based equivalences. *Acta Informatica*, 31(8):697–718, 1994.

[33] A. Kiehn and M. Hennessy. On the decidability of non-interleaving process equivalences. In B. Jonsson and J. Parrow, editors, *CONCUR '94*, number 836 in Lecture Notes in Computer Science, pages 18–33. Springer-Verlag, 1994.

[34] P. Krishnan. Architectural CCS. *Formal Aspects of Computing*, 162:162–187, 1996.

[35] J. Meseguer. Rewriting as a unified model of concurrency. In Baeten and Klop [5], pages 384–400.

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

[37] R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B, pages 1201–1242. MIT Press, 1994.

[38] U. Montanari and D.N. Yankelevich. An algebraic view of interleaving and distributed operational semantics for CCS. In D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, number 389 in Lecture Notes in Computer Science, pages 5–20. Springer-Verlag, 1989.

[39] U. Montanari and D.N. Yankelevich. A parametric approach to localities. In W. Kuich, editor, *ICALP 92*, number 623 in Lecture Notes in Computer Science, pages 617–628. Springer-Verlag, 1992.

[40] U. Montanari and D.N. Yankelevich. Location equivalence in a parametric setting. *Theoretical Computer Science*, 149:299–332, 1995.

[41] M. Mukund and M. Nielsen. CCC, locations and asynchronous transition systems. In R. Shyamasundar, editor, *FST&TCS 92*, number 652 in Lecture Notes in Computer Science, pages 328–341. Springer-Verlag, 1992.

[42] D. Murphy. Observing located concurrency. In Borzyszkowski and Sokolowski [6], pages 473–484.

[43] C. Priami and D. Yankelvich. Read-write causality. In I. Prívara, B. Rovan, and P. Ružička, editors, *MFCS '94*, number 841 in Lecture Notes in Computer Science, pages 567–576. Springer-Verlag, 1994.

[44] V. Sassone, M. Neilsen, and G. Winskel. Models for concurrency: towards a classification. *Theoretical Computer Science*, 170:297–348, 1996.

[45] E. Shapiro. Embeddings among concurrent programming languages. In W.R. Cleaveland, editor, *CONCUR '92*, number 630 in Lecture Notes in Computer Science, pages 486–504. Springer-Verlag, 1992.

[46] R.J. van Glabbeek. The linear time—branching time spectrum. In Baeten and Klop [5], pages 278–297.

[47] R.J. van Glabbeek. The linear time—branching time spectrum II (the semantics of sequential systems with silent moves). In E. Best, editor, *CONCUR '93*, number 715 in Lecture Notes in Computer Science, pages 66–81. Springer-Verlag, 1993.

Acknowledgements

This research was supported in part by a Patrick and Margaret Flanagan Scholarship. Comments from the referees are gratefully acknowledged.