Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 206 (2008) 41–58

www.elsevier.com/locate/entcs

Constrained Simulations, Nested Simulation Semantics and Counting Bisimulations

David de Frutos Escrig[1,3]

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
Madrid, Spain

Carlos Gregorio Rodríguez [2,4]

Departamento de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
Madrid, Spain

Abstract

Nested simulations define an interesting hierarchy of semantic preorders and equivalences in which every semantics refines the previous one and it is refined by the following. This nested nature provides a fruitful framework for the study of the formal meaning and the properties of concurrent processes. In this paper we present the notion of *constrained simulation* that, although rather simple, allows us to find general results for a wide family of semantics. In particular, we provide an axiomatization for both the preorder and the equivalence induced by any constrained simulation. Nested simulations are constrained simulations and therefore our results can be instantiated directly to them. Besides, constrained simulations suggest the definition of a new family of semantics, *generalised nested simulation semantics*, constructed over the base of any order relation, instead of plain simulation. Finally, we conclude the study of the (generalised) nested semantics defining a generalisation of bisimulation relations, *counting bisimulation*, that allows us to define a characterisation of nested semantics in terms of a bisimulation-like game.

Keywords: Process Preorders and Equivalences, Bisimulations, Nested Simulations, Constrained Simulations, Axiomatization of Semantics.

1 Introduction

The behaviour of concurrent processes is usually described by means of labelled transition systems (lts) and a number of different semantics have been defined in

[1] Partially supported by the MCyT project DESAFIOS TIN2006-15660-C02-02 and the project PROMESAS-CAM S-0505/TIC/0407.

[2] Partially supported by the MCyT project WEST TIN2006-15578-C02.

[3] Email: defrutos@sip.ucm.es

[4] Email:cgr@sip.ucm.es

doi:10.1016/j.entcs.2008.03.074

the literature in order to assign the formal meaning of processes. Every semantics determines which aspects of the behaviour of processes are of importance and which are not. Mainly because of the generality and diversity of the applications of process algebras, there is no prevailing semantics notion but rather there are a number of different proposals that have arisen from diverse approaches, contexts and applications.

We consider that this variety of process semantics is a good sign of the applicability of process algebras, just proving the healthiness of the formalism. However, the diversity of semantics makes difficult to choose the one that suits a given application better and therefore, the comparative study of semantics is an essential field of study.

In [10] many semantics were presented in a uniform way and compared with each other according to their distinction power. The result is the well known linear time-branching time spectrum. Furthermore, for most of the semantics also a complete axiomatization for a basic language of processes was provided.

One of the semantics in the linear time-branching time spectrum spectrum is the nested simulation semantics, that was introduced in [11]. That paper was devoted to the study of a format of rules in Plotkin style to define structured operational semantics, that was called tyft/tyxt. All the operators defined within this format preserve the strong bisimulation equivalence and thus bisimulation is a congruence with respect to them. The 2-nested simulation semantics was proved to be a characterisation of the complete trace congruence induced by the operators in pure tyft/tyxt format. Besides, the nested simulations define an interesting hierarchy of nested semantic preorders and equivalences. Any of the preorders $\subseteq_{\rightarrow_i}$ induces an equivalence, $=_i$, and the preorder $\subseteq_{\rightarrow_{i+1}}$ is defined only over processes that are $=_i$-equivalent. That is, we could decide to work in a given semantics level (say i) which would define the set of classes of equivalent processes. Then, in order to compare the processes within these classes we could use the next semantic level, $\subseteq_{\rightarrow_{i+1}}$, as an order relation that induces a new equivalence relation $=_{i+1}$. Therefore, by means of nested semantics we get a whole family of finer and finer semantics defined in a common framework.

In order to capture the common properties of all the nested simulation semantics we introduce the notion of *constrained simulations*, which are just plain simulations to whom we impose the additional constraint of being related by a given relation C. Although this notion is rather general it has been a surprise for us not to find it anywere at the available literature. We have proved an interesting collection of results valid for any constrained simulation semantics which open the door for a nice algebraic theory. In particular, we have obtained a simple axiomatization for both the preorder and the equivalence induced by any constraint.

Nested simulations turn out to be constrained simulations and therefore all the results mentioned above apply to them. Moreover, the study of constrained simulations suggests a natural generalisation of the classical definition of nested simulations. As we will see, we have defined a new family of semantics, the *generalised*

nested simulation semantics, which have similar properties and can be studied all together. This family extends the possibilities of choosing the right semantics for a given model or application while keeping in the framework of the constrained simulations.

Calling back Van Glabbeek's linear time-branching time spectrum, one of the most important of all the semantics there is the bisimulation semantics. Bisimulation is a mathematically elegant concept which is the basis of coalgebraic theories. Bisimulation is the strongest of the semantics in the spectrum, thus whenever two processes are bisimilar they are also related under any of the other semantics. But for the same reason, it can be considered too strong in many occasions. One of the main virtues of bisimulation is that there are efficient algorithms [18,15] to decide bisimulation equivalence and several tools that can effectively check process bisimilarity [5]. In order to understand bisimulation semantics, its game theory presentation is rather useful, see for instance [19]. The idea is to present the pairs of transitions of the compared processes as the plays of two players. The first player (the attacker) has to try to find a transition that cannot be replied by the other (the defender). Then we have that the defender has an strategy to win the game if and only if the given processes are indeed bisimilar.

Part of our recent research has focused on the search of a general framework based on bisimulation in which to define weaker equivalences preserving, at the same time, the coinductive flavour of bisimulation. This brings us new possibilities both from the theoretical and the practical point of view: On the one side, we can use coinductive methods that have proved to be mathematically powerful and elegant; on the other side, the possibility of using the tools to decide bisimilarity for other semantics (see for instance, [4]). In [6,8] we defined global bisimulations in which the players could modify somehow the transition system of the processes, by means of what we called global transitions. In a more algebraic setting, by using bisimulations up-to, in [7] we presented a systematic way to characterise every semantics in the linear time-branching time spectrum coarser than the ready simulation as a bisimulation-like game.

In this paper, devoted to the study of the nested semantics, we conclude by providing a framework, the *counting bisimulation*, that can be used to get a coinductive, bisimulation like characterisation, for the family of generalised nested simulation semantics.

The rest of the paper is structured as follows. In Section 2, the preliminaries, notation, and basic definitions are presented. In Section 3, we introduce our constrained simulations that allow us to prove some general results, as the existence of a conditional axiomatisation for any member of the family of semantics. These results are particularised in Section 4 for the classic nested simulation semantics. Constrained simulations also suggest a generalisation of nested simulations semantics, that is studied in Section 5. Section 6 is devoted to the definition of a bisimulation-like game for the (generalised) nested simulations semantics, the counting bisimulation game. We sum up our conclusions in Section 7, where we also discuss some lines for future work on the subject.

2 Preliminaries

Although some of the results in this paper can be extended to infinite processes, we will mainly concentrate on finite processes represented by trees, which just correspond to those in the class BCCSP.

Definition 2.1 [10] Given a set of actions Act, the set of BCCSP processes is defined by the following BNF-expression:

$$p ::= \mathbf{0} \mid ap \mid p + q$$

where $a \in Act$. $\mathbf{0}$ represents the process that can perform no action. For every action in Act, there is a prefix operator that defines the sequential execution of actions. Finally, $+$ is called the choice operator and denotes the election between two processes.

The operational semantics for BCCSP terms is defined by a labelled transition system (lts) and is given by the rules in Figure 1. Each BCCSP processes has as semantics an acyclic lts that can be unfolded to get a finite tree. Then, the depth of a process is the depth of the corresponding tree.

It is true that if we simply consider the transitions, then the obtained lts is a directed acyclic graph, but where each process appears only once, so that it is possible to have several arcs (transitions) reaching the same node (process). But since the graph is acyclic, by unfolding it we get an equivalent tree where each node appears as many times as ways we have to reach it by a proved computation, where we distinguish two transitions that have been generated in a different way using the SOS rules. It is easy to prove that this tree is also equivalent to the syntactic tree of p, once we consider that $+$ operator is commutative and associative.

Along the paper some usual notations for lts are used. We write $p \xrightarrow{a}$ if there exists a process q such that $p \xrightarrow{a} q$ and, on the opposite, we write $p \not\xrightarrow{a}$ if there exists no process q such that $p \xrightarrow{a} q$. For a string of actions $\sigma = a_1 a_2 \cdots a_n$, $a_i \in Act$, $p \xrightarrow{\sigma} q$ means that there exist processes $q_1 \ldots q_{n-1}$, such that $p \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots q_{n-1} \xrightarrow{a_n} q$. The function I calculates the set of initial actions of a process, $I(p) = \{a \mid a \in Act \text{ and } p \xrightarrow{a}\}$.

$$ap \xrightarrow{a} p \qquad \frac{p \xrightarrow{a} p'}{p + q \xrightarrow{a} p'} \qquad \frac{q \xrightarrow{a} q'}{p + q \xrightarrow{a} q'}$$

Fig. 1. Operational Semantics for BCCSP Terms

As usual, trailing occurrences of the constant $\mathbf{0}$ are omitted, we write a instead of $a\mathbf{0}$. By using $\sum$ as a shorthand for multiple choice (which is commutative and associative) we can define any process as $\sum_i \sum_j a_i p_{ij}$. A process aq' is a summand of the process q if and only if $q \xrightarrow{a} q'$. Given $a \in Act$ we define $p|_a$ as the (sub)process we get by adding all the a-summands of p. That is, if $p = \sum_i \sum_j a_i p_{ij}$, then $p|_{a_i} = \sum_j a_i p_{ij}$.

An important part of the classic behavioural semantics that were thoroughly classified in [10] are based on the simple concept of simulation. Besides, simulation is also in the core of the definition of nested semantics.

Definition 2.2 A binary relation S over processes is a *simulation*, if pSq implies that:

- For every a, if $p \xrightarrow{a} p'$ there exists q', $q \xrightarrow{a} q'$ and $p'Sq'$.

We say that process p is simulated by process q, or that q simulates p, written $p \sqsubseteq_S q$, whenever there exists a simulation, S, such that pSq.

Bisimilarity is a capital notion in process theory and it is the cornerstone for one of the targets of our work: providing coinductive characterisations of process semantics.

Definition 2.3 [17] A binary relation $\mathcal{R}$ is a (strong) *bisimulation* if for all p, q processes such that $p \mathcal{R} q$, and for all $a \in Act$, the following properties are satisfied:

- Whenever $p \xrightarrow{a} p'$ there exists some q' such that $q \xrightarrow{a} q'$ and $p' \mathcal{R} q'$.
- Whenever $q \xrightarrow{a} q'$ there exists some p' such that $p \xrightarrow{a} p'$ and $p' \mathcal{R} q'$.

Two processes p and q are *bisimilar*, written $p =_B q$, if there exists a bisimulation containing the pair $\langle p, q \rangle$.

As it is well known, bisimularity admits multiple characterisations. In the rest of the paper we will make an extensive use of its axiomatisation, that is given in Figure 2.

$$
\begin{array}{ll}
(B_1) & x + y = y + x \\
(B_2) & (x + y) + z = x + (y + z) \\
(B_3) & x + x = x \\
(B_4) & x + \mathbf{0} = x
\end{array}
$$

Fig. 2. Axiomatisation for the (Strong) Bisimulation Equivalence

We are interested in semantic preorders for processes and some of the essential properties are gathered under the name of behaviour preorder.

Definition 2.4 A preorder relation $\sqsubseteq$ over processes is a *behaviour preorder* when it is weaker than the bisimulation equivalence and it is a precongruence with respect to the prefix and choice operators, i.e. if $p \sqsubseteq q$ then $ap \sqsubseteq aq$; and if $p \sqsubseteq q$ then $p + r \sqsubseteq q + r$.

This definition is quite natural and all the preorders in the linear time-branching time spectrum are indeed behaviour preorders. For the sake of simplicity, we often use the symbol $\sqsupseteq$ to represent the preorder relation $\sqsubseteq^{-1}$.

In [7] we introduced the concept of bisimulation up-to a given relation, in order to get coinductive characterisations of equivalence relations.

Definition 2.5 Let $\sqsubseteq$ be a behaviour preorder, we say that a binary relation S over processes is a *bisimulation up-to* $\sqsubseteq$, if pSq implies that:

- For every a, $p \xrightarrow{a} p'_a$, there exist q' and q'_a, $q \sqsupseteq q' \xrightarrow{a} q'_a$ and $p'_a S q'_a$;
- For every a, $q \xrightarrow{a} q'_a$, there exist p' and p'_a, $p \sqsupseteq p' \xrightarrow{a} p'_a$ and $p'_a S q'_a$.

Two processes are *bisimilar up-to* $\sqsubseteq$, written $p \approx_{\sqsubseteq} q$, if there exists a bisimulation up-to $\sqsubseteq$, S, such that pSq.

In [7] we have proved that every semantics in the linear time-branching time spectrum under the ready simulation can be characterised in terms of the adequate bisimulations up-to. These characterisations are based on the preorder defining the corresponding equivalence and the axioms defining such a preorder can be used as rules that generate the global transitions that weaken the bisimulation game as required.

We also proved in [7] that other semantics can be characterised with bisimulations up-to, as it is the case for the n-nested simulation semantics. However, given that there exists no finite unconditional axiomatization for the n-nested simulation semantics, we cannot take advantage of their axioms. We overcome this problem in Section 6, where we get a coinductive characterisation of the nested simulation semantics based on counting bisimulations and bisimulations up-to.

3 Constrained Simulations

C-constrained simulations are just plain simulations to which we impose that their pairs should also be related by the constraint C.

Definition 3.1 Given a relation C over BCCSP processes, a relation S_C is a *C-constrained simulation*, if pS_Cq implies:

- For every a, if $p \xrightarrow{a} p'$ there exists q', $q \xrightarrow{a} q'$ and $p' S_C q'$, and
- pCq.

We say that process p is C-simulated by process q, or that q C-simulates p, written $p \sqsubseteq^C_{\rightarrow} q$, whenever there exists a C-constrained simulation S_C, such that pS_Cq.

Since we want to characterise behaviour preorders by using C-simulations it is reasonable to impose on these simulations the condition of being themselves behaviour preorders, that is guaranteed whenever the constraints are also behaviour preorders. Given that the operators in our basic algebra BCCSP are those generating finite trees, this condition is quite natural and the results we will prove based on it are indeed rather general.

Example 3.2 Let us briefly present several examples of constrained simulations, all of them corresponding to relations being behaviour preorders.

- Ordinary simulation is a constrained simulation taking as C the universal relation, xCy for every x and y.
- Ready simulation is just the I-constrained simulation, where $pIq \Leftrightarrow I(p) = I(q)$.

- Ready simulation is perhaps the most important C-constrained simulation but we can also achieve a greater discriminatory power. Let us consider, for instance, $\sqsubseteq_S$ to be the simulation preorder and $C = \sqsubseteq_S^{-1}$; then 2-nested simulations [11] are just the corresponding class of C-constrained simulations.

For finite processes[5] C-similarity can be inductively defined by applying the following characterisation:

Proposition 3.3 *Let p, q be BCCSP terms, then $p \sqsubseteq_{\rightarrow}^{C} q$ iff*

- *for all $p \xrightarrow{a} p'$ there exists q' such that $q \xrightarrow{a} q'$ with $p' \sqsubseteq_{\rightarrow}^{C} q'$, and*
- *pCq.*

Proof. The left to right implication is trivial by definition. For the other direction we take as S the relation defined by the right hand side of the statement. Then S is a simulation since we have just proved that $p \sqsubseteq_{\rightarrow}^{C} q$ implies pSq. Moreover, by definition, S is C-constrained. □

C-constrained similarity, $\sqsubseteq_{\rightarrow}^{C}$, can be conditionally axiomatized in a simple way. For any constraint C we just need to consider the axiom

$$(P_C) \qquad xCy \Rightarrow x \sqsubseteq x + y$$

We define the axiomatization $\mathcal{P}_C$ as the set of axioms obtained by adding the axiom P_C to the set of axioms that characterises bisimulation equivalence (Figure 2), $\mathcal{P}_C = \{B_1, B_2, B_3, B_4, P_C\}$. As usual, we write $\mathcal{P}_C \vdash p \sqsubseteq q$ when the relation $p \sqsubseteq q$ is provable from $\mathcal{P}_C$ using the rules of inequational logic. We next prove that $\mathcal{P}_C$ is sound and complete with respect to $\sqsubseteq_{\rightarrow}^{C}$.

Theorem 3.4 *For every constraint C being a behaviour preorder, we have that*

$$\mathcal{P}_C \vdash p \sqsubseteq q \Leftrightarrow p \sqsubseteq_{\rightarrow}^{C} q$$

Proof. Soundness. Bisimilarity axioms are sound for both the relation C and for the C-constrained simulation preorder. Therefore, we only need to prove that axiom (P_C) is also sound. Process $p + q$ can obviously simulate p, and since we have pCq and C is a congruence with respect to choice, we also have $pC(q+p)$ and we conclude that $p \sqsubseteq_{\rightarrow}^{C} p + q$.

Completeness. By induction on the depth of processes. If $p = \mathbf{0}$ then $\mathbf{0}Cq$ and, applying $(P_C), (B_1)$ and (B_4), $\mathcal{P}_C \vdash \mathbf{0} \sqsubseteq q$. Let us consider now the general case $p = \sum a_i p_i$. On the one hand, if $p \sqsubseteq_{\rightarrow}^{C} q$ then pCq and we can use (P_C) to prove that $\mathcal{P}_C \vdash p \sqsubseteq p + q$. On the other hand, whenever $p \xrightarrow{a_i} p_i$ then $q \xrightarrow{a_i} q_{j_i}$ with $p_i \sqsubseteq_{\rightarrow}^{C} q_{j_i}$; by induction hypothesis we have $\mathcal{P}_C \vdash p_i \sqsubseteq q_{j_i}$, therefore we have

[5] In fact the proposition itself is valid for arbitrary processes, but in the case of infinite processes it cannot be turned into an inductive definition.

$\mathcal{P}_C \vdash q + \sum a_i p_i \sqsubseteq q + \sum a_i q_{j_i}$, or equivalently $\mathcal{P}_C \vdash q + p \sqsubseteq q$. Combining both results we get $\mathcal{P}_C \vdash p \sqsubseteq q + p \sqsubseteq q$. □

We next study the axiomatization of the equivalence relation associated to the C-constrained simulation, $\rightleftarrows^C = \sqsubseteq_{\rightarrow}^C \cap \sqsupseteq_{\leftarrow}^C$. We propose the following axiom for each constraint C:

$$(E_C) \qquad xCy \Rightarrow a(x+y) = a(x+y) + ay$$

Adding the axiom E_C to the set of axioms that characterises bisimulation equivalence (Figure 2), we get the set $\mathcal{E}_C = \{B_1, B_2, B_3, B_4, E_C\}$. We write $\mathcal{E}_C \vdash p = q$ when the equation $p = q$ is provable from $\mathcal{E}_C$.

We next prove that $\mathcal{E}_C$ is sound and complete with respect to $\rightleftarrows^C$. However, in this case, in order to prove these results, the constraint relation has to be symmetric, that is, it has to be a behaviour equivalence.

Theorem 3.5 *For every constraint C being a behaviour equivalence, we have that*

$$\mathcal{E}_C \vdash p = q \Leftrightarrow p \rightleftarrows^C q$$

Proof. Soundness. Let us just prove that (E_C) is sound. Whenever pCq we also have $\mathcal{E}_C \vdash a(p+q) = a(p+q) + aq$. We have to prove that $a(p+q) \sqsubseteq_{\rightarrow}^C a(p+q) + aq$ and $a(p+q) + aq \sqsubseteq_{\rightarrow}^C a(p+q)$. Both sides can trivially simulate each other and using that C is a behaviour preorder, from pCq we derive $a(p+q)\ C\ a(p+q) + aq$ and we immediately conclude that the first one is a C-simulation. In the second, case to have a C-simulation we need to prove both $a(p+q) + aq\ C\ a(p+q)$ and $qC(p+q)$. As before, from pCq we derive $a(p+q)\ C\ a(p+q) + aq$ and, since C is symmetric, we conclude $a(p+q) + aq\ C\ a(p+q)$. For the later, since C is symmetric we have qCp and then $qC(p+q)$. Thus both simulations are indeed C-constrained.

Completeness. The proof of the completeness of the axiomatization of the simulation equivalence in [10] (Section 17.2) can be transferred without any changes just checking the additional proof obligations imposed by the condition in the axiom (E_C). □

It is interesting to note that the cases in which C is not symmetric are not completely excluded from the result above.

Definition 3.6 We say that two constraints C_1 and C_2 are *cs-equivalent*, what we denote by $C_1 \sim C_2$, iff they define the same C-constrained similarity relation, that is $\sqsubseteq_{\rightarrow}^{C_1} = \sqsubseteq_{\rightarrow}^{C_2}$.

Next proposition is just a snapshot of a nice algebraic theory that can be developed around constrained simulations and cs-equivalence.

Proposition 3.7 *For any behaviour preorders C, C_1 and C_2 we have:*

(i) $C_1 \sim C_2 \Rightarrow (C_1 \cap C_2) \sim C_1$.

(ii) $C \sim \sqsubseteq_{\rightarrow}^{C}$ *and* $\sqsubseteq_{\rightarrow}^{C}$ *is the smallest* C*-simulation that is cs-equivalent to* C*.*

(iii) *If* $C_1 \sim C_2$ *and* $C_1 \supseteq C \supseteq C_2$*, then* $C \sim C_1$*.*

(iv) *For the simulation preorder* $\sqsubseteq_S$ *we have that* $C \sim (C \cap \sqsubseteq_S)$*.*

Example 3.8 Next we present some illustrative examples of cs-equivalent constraints.

- Let us consider the classical simulation preorder, $\sqsubseteq_S$. As we saw in Example 3.2, $\sqsubseteq_S = \sqsubseteq_{\rightarrow}^{U}$, where U is the universal relation, xUy for every x and y. On the other hand, if we use $\sqsubseteq_S$ as constraint, it is immediate to see that $\sqsubseteq_S = \sqsubseteq_{\rightarrow}^{\sqsubseteq_S}$ and therefore $U \sim \sqsubseteq_S$, but while U is symmetric, $\sqsubseteq_S$ is not.
- Taking the constraint $I_{\supseteq}$ given by $pI_{\supseteq}q \Leftrightarrow I(p) \supseteq I(q)$, we have that ready simulation was originally defined in [3] as $\sqsubseteq_{\rightarrow}^{I_{\supseteq}}$, but it is well known that it also coincides with $\sqsubseteq_{\rightarrow}^{I}$, see for instance [10]. Again, I is symmetric, $I_{\supseteq}$ is not, and $I \sim I_{\supseteq}$.
- In a similar way, we can define the 2-nested simulation as a simulation constrained using either $\sqsubseteq_S^{-1}$ or the equivalence relation $\sqsubseteq_S^{-1} \cap \sqsubseteq_S$.

From the examples above, one could guess that any constraint might be cs-equivalent to some other symmetric constraint. This is indeed the case for any "interesting" constraint we have found, but in general it is not true, as the following counterexample shows.

Example 3.9 If we consider the behaviour preorder $\sqsubseteq$ defined by the axioms of bisimulation equivalence (Figure 2) together with the axiom $x \sqsubseteq x + aa$, where a represents any arbitrary action in Act, we can check that there is no symmetric constraint cs-equivalent to $\sqsubseteq$.

4 Nested Simulation Semantics

Nested semantics were originally defined by Groote and Vaandrager in [11], for arbitrary labelled transition systems. But since we are considering BCCSP terms we will use instead the following definition:

Definition 4.1 [2] For $n \geq 0$, we define the relation $\subseteq_{\rightarrow n}$ inductively over BCCSP terms thus:

- $p \subseteq_{\rightarrow 0} q$ for all p, q,
- $p \subseteq_{\rightarrow n+1} q$ iff pRq for some simulation R with R^{-1} included in $\subseteq_{\rightarrow n}$

Let us note that $\subseteq_{\rightarrow 1}$ is in fact the simulation preorder, that we usually denote by $\sqsubseteq_S$. Besides, from the results in [11] we can give an alternative definition of nested simulation semantics.

Definition 4.2 For $n \geq 1$, we define the relation $\subseteq\!\!\rightarrow_n$ inductively over BCCSP terms as follows:

- $p \subseteq\!\!\rightarrow_1 q$ iff $p \sqsubseteq_S q$,
- $p \subseteq\!\!\rightarrow_{n+1} q$ iff pRq for some simulation R included in $\rightleftarrows_n$

As we have already commented on Example 3.2, simulations are constrained simulations under the universal relation U, given by xUy for every x and y, that is, $\subseteq\!\!\rightarrow_1 = \sqsubseteq_{\rightarrow}^{U}$. Then 2-nested simulations are also constrained simulations, so that we have $\subseteq\!\!\rightarrow_2 = \sqsubseteq_{\rightarrow}^{(\subseteq\!\!\rightarrow_1)^{-1}} = \sqsubseteq_{\rightarrow}^{\rightleftarrows_1}$, and in general, for the $n+1$ nested simulation preorder we have $\subseteq\!\!\rightarrow_{n+1} = \sqsubseteq_{\rightarrow}^{(\subseteq\!\!\rightarrow_n)^{-1}} = \sqsubseteq_{\rightarrow}^{\rightleftarrows_n}$. Therefore, all the results in Section 3 are applicable for these semantics. In particular, we have a conditional axiomatisation for the n-nested simulation semantics given by the axiom

$$(A_{ns}) \qquad y \sqsubseteq_n x \Rightarrow x \sqsubseteq_{n+1} x + y$$

together with the axioms defining bisimulation equivalence (see Figure 2). As a matter of fact, this is the same axiomatisation already proved to be complete for the case of 2-nested simulation preorder in [1]. There, they also enunciate the generalisation for the case of n-nested simulation, although the proof was omitted. Although this is a subtle point, the interested reader can check that our proof (see Theorem 3.5) although being more general is in fact simpler, since we do not need to use any particular property of the nested simulations ordering, but just the fact that it is a constrained simulation.

If we consider n as a parameter in axiom (A_{ns}), we see that we need a single axiom to characterise all the nested simulation preorders in a common framework. To complete the axiomatisation we can add the following unconditional axiom characterising the simulation preorder,

$$(A_s) \qquad x \sqsubseteq_1 x + y$$

but if we prefer a more symmetrical treatment of these relations we can also take $U = \sqsubseteq_0$ as the foundation of our construction, and then we only need

$$(A_u) \qquad x \sqsubseteq_0 y$$

The same is true for the n-nested simulation equivalences, that can be axiomatised by means of a single axiom

$$(E_{ns}) \qquad x =_n y \Rightarrow a(x + y) =_{n+1} a(x + y) + ax$$

based on the inconditional axiom $(E_U) \quad x =_0 y$.

As we have seen, our results on constrained simulations are quite general and can be applied to many simulation preorders (see Example 3.2) from plain simulation to any of the n-nested simulations. However, for simulation, complete simulation,

and ready simulation preorders there exist non-conditional axiomatisations (see, for instance, [10]). Instead for the n-nested simulations it is impossible for any $n \geq 2$ to find such a non-conditional axiomatisation [2]. We consider that it could be interesting to investigate in which cases such a non-conditional axiomatisation exists and when they could be systematically generated from the conditional ones obtained as instantiations of our general axiomatization for constrained simulations.

5 Generalised Nested Simulation Semantics

After the results presented in Section 3, showing the regularity of the constrained simulations, Definition 4.2 of nested simulations can be naturally generalised. The new definition we propose, while keeping the nesting of simulations, is parameterised in its first floor by any behaviour preorder R.

Definition 5.1 Given a behaviour preorder R, for $n \geq 1$, we inductively define the n-nested R-simulation relation, denoted by $\subseteq\!\!\rightarrow_n^R$, as follows:

- $p \subseteq\!\!\rightarrow_1^R q \Leftrightarrow pRq$
- $p \subseteq\!\!\rightarrow_{n+1}^R q \Leftrightarrow pSq$ for some simulation S included in $\rightleftarrows_n^R$

We denote by $\rightleftarrows_n^R$ the equivalence relation induced by the preorder $\subseteq\!\!\rightarrow_n^R$, that is, $\rightleftarrows_n^R = \subseteq\!\!\rightarrow_n^R \cap (\subseteq\!\!\rightarrow_n^R)^{-1}$.

The next proposition proves the preservation of the properties of nesting semantics for the generalised n-nested R-simulation semantics. In fact, this proposition also justifies that our Definition 5.1 is indeed a generalisation of Definition 4.2. In general we would not get always these pleasant properties generalising Definition 4.1, although it would be enough to impose to R the condition of being an equivalence relation in order to satisfy the properties stated in Proposition 5.2, starting from a generalisation of Definition 4.1.

Proposition 5.2 *For any behaviour preorder R and for all $n \geq 1$ we have:*

- $\subseteq\!\!\rightarrow_{n+1}^R \subseteq \subseteq\!\!\rightarrow_n^R$
- $\subseteq\!\!\rightarrow_{n+1}^R \subseteq (\subseteq\!\!\rightarrow_n^R)^{-1}$
- $\rightleftarrows_{n+1}^R \subseteq \subseteq\!\!\rightarrow_{n+1}^R \subseteq \rightleftarrows_n^R$
- *for all $n \geq 2$ we have* $\subseteq\!\!\rightarrow_n^R = \subseteq\!\!\rightarrow_n^{(R \cap R^{-1})}$

These results allow us to define a translation from n-nested R-simulations into

constrained simulations that will be useful:

Nested		Constrained	Nested		Constrained
$\subseteq^R_{\rightarrow 1}$	$\Leftrightarrow$	R	$\rightleftarrows^R_1$	$\Leftrightarrow$	$R \cap R^{-1}$
$\subseteq^R_{\rightarrow 2}$	$\Leftrightarrow$	$\sqsubseteq^{\rightleftarrows^R_1}_{\rightarrow}$	$\rightleftarrows^R_2$	$\Leftrightarrow$	$\rightleftarrows^{\rightleftarrows^R_1}$
	$\vdots$			$\vdots$	
$\subseteq^R_{\rightarrow n+1}$	$\Leftrightarrow$	$\sqsubseteq^{\rightleftarrows^R_n}_{\rightarrow}$	$\rightleftarrows^R_{n+1}$	$\Leftrightarrow$	$\rightleftarrows^{\rightleftarrows^R_n}$

Clearly, any n-nested R-simulation relation is a constrained simulation. Taking as R the simulation relation we get the classical nested simulation semantics, that we denote $\subseteq^S_{\rightarrow n}$. But we can define many other semantics. For instance, if we take as R the trace preorder, then we get a completely new hierarchy of nested semantics, $\subseteq^T_{\rightarrow n}$, the n-nested trace simulation semantics.

As an illustrative example, let us briefly study the relationships between the two families of nested semantics $\subseteq^S_{\rightarrow n}$ and $\subseteq^T_{\rightarrow n}$. First we give a very simple proposition that holds for any pair of constrained simulations.

Proposition 5.3 *Let R and S be behaviour preorders, whenever $R \subseteq S$ we have also $\sqsubseteq^R_{\rightarrow} \subseteq \sqsubseteq^S_{\rightarrow}$.*

Then we can state the following result that relates the members of the two families of semantics introduced above.

Proposition 5.4 *For all $n \geq 1$ we have*

(i) $\subseteq^S_{\rightarrow n} \subseteq \subseteq^T_{\rightarrow n}$,

(ii) $\subseteq^T_{\rightarrow n+1} \subseteq \subseteq^S_{\rightarrow n}$.

Proof. We prove only the first statement, since the proof of the second is rather similar. We proceed by induction on n. For $n = 1$ we have to prove that $S \subseteq T$, what is well known. In the general case, by induction hypothesis, we have $\subseteq^S_{\rightarrow n} \subseteq \subseteq^T_{\rightarrow n}$, and therefore $\rightleftarrows^S_n \subseteq \rightleftarrows^T_n$; to prove that $\subseteq^S_{\rightarrow n+1} \subseteq \subseteq^T_{\rightarrow n+1}$ we use the translation into constrained simulations: $\subseteq^S_{\rightarrow n+1} \Leftrightarrow \sqsubseteq^{\rightleftarrows^S_n}_{\rightarrow}$ and $\subseteq^T_{\rightarrow n+1} \Leftrightarrow \sqsubseteq^{\rightleftarrows^T_n}_{\rightarrow}$, and applying Proposition 5.3 we conclude the proof. □

Once again we have shown how by working on the general framework of constrained simulations we get a simple and general proof. As an immediate corollary of propositions 5.4 and 5.2 we get the nice diagram in Figure 3.

In the same way as we have defined the brand new family of nested trace simulation semantics we could lift any of the preorders in the linear time-branching time spectrum to the corresponding family of nested semantics. A trivial generalisation of the first statement of Proposition 5.4 tells us that the inclusion relation defined

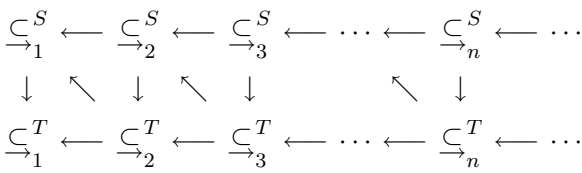

Fig. 3. Inclusion relations of two different families of generalised nested simulations semantics

in the linear time-branching time spectrum is naturally preserved at any level of nesting. Other relations between levels, as the one stated in the second statement of Proposition 5.4, deserve further study.

6 Counting Bisimulation

In [8,7] we have used global bisimulations and bisimulations up-to to characterise all the semantics in Van Glabbeek's spectrum in a coinductive way. This characterisation also provides a bisimulation-like game defining each of the semantics in the spectrum. But, when they have not a finite axiomatisation we have not a simple way to generate the global transitions that are needed to weaken the (original) bisimulation game to capture the desired semantics.

In this section we define a bisimulation-like game, the counting bisimulation game, that characterises the nested simulation semantics. Let us briefly explain how we obtained this new game. Since each nested simulation is a constrained simulation, in order to check it, we can use the simulation game where the attacker has always to play in the p-side of the board (when proving $p \subseteq_{n+1} q$). But the imposed constraint $q \subseteq_n p$ should also be checked continuing with the game but, since we are comparing q with p, turning the board. In this way we get a nearly symmetric condition, so that nested simulations become nearly bisimulations. But of course the symmetry is not total: the index has been decremented, so that it must be taken into account in order to differentiate the level of the nesting. In particular, when n becomes one, the bisimulation part of the game has terminated and the game reduces to checking the basic relation in the nested semantics. In particular, for classic nested simulation semantics the base game is plain simulation.

As we will justify below, instead of checking this relation we can check mutual similarity, what can be made playing a bisimulation up-to game [7]. This way we will be playing a bisimulation-like game all the time: first the counting bisimulation and, at the end, the bisimulation up-to simulation.

Let us precisely define the counting bisimulation for the nested simulation semantics. Let $\sqsubseteq_1$, $\sqsupseteq_1$ and $=_1$ denote the simulation preorder, their inverse and the mutual simulation equivalence, respectively. As stated in [2], the nested simulation semantics can be characterised as follows:

Proposition 6.1 *Let p, q be BCCSP terms, and $n \geq 0$*

$$p \subseteq_{n+1} q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \xrightarrow{a} q' \text{ with } p' \subseteq_{n+1} q', \text{ and} \\ q \subseteq_n p. \end{cases}$$

where $\subseteq_0$ is the total relation that relates any pair of processes.

When $n \geq 1$ we propose the following unfold:

$$p \subseteq^{cb}_{n+1} q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \xrightarrow{a} q' \text{ with } p' \subseteq^{cb}_{n+1} q', \text{ and} \\ \text{for all } q \xrightarrow{a} q' \text{ there is } p \xrightarrow{a} p' \text{ with } q' \subseteq^{cb}_{n} p'. \end{cases}$$

We have introduced the superscript *cb* (counting bisimulation) because we cannot directly simply write $\subseteq_{n+1}$, since the complete unfold of $q \subseteq_n p$ would have added also the condition $p \subseteq_{n-1} q$, which we have removed. However, in the following we will prove that in fact both definitions are equivalent.

Using the characterisation given by Definition 4.2 for the case of the 2-nested simulation semantics, we have

$$p \subseteq_2 q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \xrightarrow{a} q' \text{ with } p' \subseteq_2 q', \text{ and} \\ q =_1 p. \end{cases}$$

Using our bisimulation up-to characterisation of $=_1$ (see [7])

$$p =_1 q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \sqsupseteq_1 q' \xrightarrow{a} q'_a \text{ with } p' =_1 q'_a, \text{ and} \\ \text{for all } q \xrightarrow{a} q' \text{ there is } p \sqsupseteq_1 p' \xrightarrow{a} p'_a \text{ with } q' =_1 p'_a. \end{cases}$$

We unfold the characterisation of $\subseteq_2$ to obtain:

$$p \subseteq^{cb}_2 q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \xrightarrow{a} q' \text{ with } p' \subseteq^{cb}_2 q', \text{ and} \\ \text{for all } q \xrightarrow{a} q' \text{ there is } p \sqsupseteq_1 p' \xrightarrow{a} p'_a \text{ with } q' =_1 p'_a. \end{cases}$$

Putting together the counting bisimulation relations $\subseteq^{cb}_k$ and the bisimulation up-to characterisation for the mutual simulation equivalence, that in the following we will also write as $=^{cb}_1$, we get the following counting bisimulation game:

Definition 6.2 Given $n > 1$ and p, q two BCCSP process, the *counting bisimulation game* CBisi(p, q, n) is that governed by the following rules:

Initialisation Assign to counter c the value n.

Attack and Defence

- Moves of the attacker:
  - Either he executes some transition $p \xrightarrow{a} p'$,
  - Or he *turns* the board and executes $q \xrightarrow{a} q'$, but in this case if the counter c has a value greater than one, then c is decremented by one.
- Moves of the defender:
  - He always move in the opposite side where the attacker has made his last move. To simplify, we note that process by r. Then, if the counter c has a value greater than one, the defender has to do an ordinary move $r \xrightarrow{a} r'$; on the contrary, if the counter c is just one, then the defender can perform an up-to move, $r \sqsupseteq_1 r' \xrightarrow{a} r'_a$.

Winner

- If the attacker cannot move ($p = \mathbf{0} = q$) then the game is over and the defender wins.
- If the defender cannot reply to the last move of the attacker ($r \not\xrightarrow{a}$), then the attacker has won the game.

In plain words, any play of a counting bisimulation game has two parts: the first one, while the counter is greater than one, the game is governed by the bisimulation rules; the second one, when the counter becomes one, the rules are those of the corresponding bisimulation up-to game that characterises the base equivalence $=_1$.

Note that we have not defined the counting bisimulation game for $n = 1$. There are two reasons for that: first, we do not need it, because counting bisimulation games are introduced in order to characterise the nested semantics; degenerated 1-nested semantics is just the plain simulation semantics that needs no coinductive characterisation, since it is directly coinductive by itself. Besides, if we would start our game with $n = 1$ we would obtain a symmetric definition, that could never correspond to the asymmetric simulation preorder.

Theorem 6.3 *For any $n \geq 2$, the defender player has a winning strategy to win any play of the game CBisi(p, q, n) iff $p \subseteq_{\rightarrow_n} q$.*

Proof. It is immediate to check that the defender has a winning strategy for the game CBisi(p, q, n) iff $p \subseteq^{cb}_{\rightarrow_n} q$. Then, let us prove by induction on $k \geq 2$ that $p \subseteq_{\rightarrow_n} q$ iff $p \subseteq^{cb}_{\rightarrow_n} q$.

For $k = 2$, since

$$p =_1 q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \sqsupseteq_1 q' \xrightarrow{a} q'_a \text{ with } p' =_1 q'_a, \text{ and} \\ \text{for all } q \xrightarrow{a} q' \text{ there is } p \sqsupseteq_1 p' \xrightarrow{a} p'_a \text{ with } q' =_1 p'_a. \end{cases}$$

it is clear that $p \subseteq_{\rightarrow_2} q \Rightarrow p \subseteq^{cb}_{\rightarrow_2} q$. For the opposite we just need to prove that $p \subseteq^{cb}_{\rightarrow_2} q$ implies $p =_1 q$, what is immediate since $\subseteq^{cb}_{\rightarrow_2}$ is a simulation and $p' =_1 q' \Rightarrow p' \sqsubseteq_S q'$.

For $k+1>2$, if we complete the unfolding we did to generate $\subseteq_{\rightarrow k+1}$ we obtain

$$p \subseteq_{\rightarrow k+1} q \Leftrightarrow \begin{cases} \text{for all } p \xrightarrow{a} p' \text{ there is } q \xrightarrow{a} q' \text{ with } p' \subseteq_{\rightarrow k+1} q', \text{ and} \\ \text{for all } q \xrightarrow{a} q' \text{ there is } p \xrightarrow{a} p \text{ with } q' \subseteq_{\rightarrow k} p', \text{ and} \\ p \subseteq_{\rightarrow k-1} q. \end{cases}$$

Then we prove the equivalence between both relations by induction on depth(p). Since depth(p') $<$ depth(p) we can assume $p' \subseteq_{\rightarrow k+1} q' \Leftrightarrow p' \subseteq^{cb}_{\rightarrow k+1} q'$ and, by induction on k, $p' \subseteq_{\rightarrow k} q' \Leftrightarrow p' \subseteq^{cb}_{\rightarrow k} q'$. So that, $p \subseteq_{\rightarrow k+1} q \Rightarrow p \subseteq^{cb}_{\rightarrow k+1} q$. For the opposite we need to check that $p \subseteq^{cb}_{\rightarrow k+1} q \Rightarrow p \subseteq^{cb}_{\rightarrow k-1} q$, which is an immediate consequence of the fact $p \subseteq^{cb}_{\rightarrow k+1} q \Rightarrow p \subseteq^{cb}_{\rightarrow k} q$, which is obvious from the definition of $\subseteq^{cb}_{\rightarrow k+1}$. □

We have two reasons for the current presentation of our counting bisimulation game for the nested simulation relations: First, because it can be immediately generalised to characterise any nested simulation based on any behaviour equivalence characterisable via bisimulations up-to; we just need to make use of the corresponding bisimulation up-to in the last part of the game. Second, because we looked for a presentation of the game as close to the original bisimulation game as possible. In particular, the attacker can always turn the board, as in the bisimulation game, although once $n-1$ turns have been made the defender can use up-to movements when playing.

However, in the particular case of the classical nested simulation relations another nice approach is possible that emanates from the smooth and compact Definition 4.1: we can define *plain counting bisimulations* as those governed by the rules of plain bisimulation, but removing the possibility of turning the board when n becomes 1, or equivalently, stopping the game, taking the defender as winner, if n becomes 0. Note that even if we are now playing with completely symmetric rules, the initial configuration makes a difference and therefore the game still characterises a preorder and not an equivalence.

Theorem 6.4 *The plain counting bisimulation game defined above also characterises the nested simulation $p \subseteq_{\rightarrow n} q$, in this case for any $n \geq 0$.*

Let us note that in this case the result is also valid for $n=1$ since CBisi$(p,q,1)$ is just the simulation game, while for $n=0$ we always obtain $p \subseteq_{\rightarrow 0} q$ as stated in Definition 4.1

7 Conclusions and Future Work

In this paper we have presented several results that shed some light into the nature of nested simulation semantics. In particular, we have characterised them by means

of counting bisimulations that are simply defined by incorporating a counter which limits the number of times that the attacker can turn the board during a play of the bisimulation game.

The main tool we have used to develop most of the results in the paper is the notion of constrained simulations. This notion is a generalisation of the ready simulation, which has been a recurrent topic in our recent research [7,8]. We have proved in this paper several results for constrained simulations, as the existence of a conditional axiomatisation, which can be applied in a very general setting.

Since classic nested simulation semantics are constrained simulations the results above apply to them. Besides, the characterisation of classic nested simulations as constrained simulations suggests a general definition of nested simulations: it is possible to define a nested simulation over any relation. In particular, any of the semantics in the Van Glabbeek's spectrum could be nested simulated. We illustrate this generalised definition by constructing the n-nested trace simulation preorders and comparing them with the classic nested simulation preorders. We plan to further investigate the power of constrained simulations. In particular, some of the constrained simulations have a non-conditional finite axiomatisation (simulation, complete simulation, ready simulation), but others, as the 2-nested simulation, have not. This opens the question of whether it is possible to characterise the constrained simulation preorders that can be finitely axiomatised and, even to try to systematically generate the non-conditional axiomatisation in those cases.

Continuing with the study in [8,7], we have next addressed the problem of finding a coinductive characterisation for the nested semantics. For this purpose we have introduced the counting bisimulation game that, used in conjunction with the bisimulation up-to technique previously developed, allows to characterise the (generalised) nested simulation semantics. Counting bisimulation seems to be an interesting generalisation of the notion of bisimulation filling the gap between simulation and bisimulation semantics. In the paper we only provide a proof of the characterisation of nested semantics by means of counting bisimulations for finite BCCSP processes, but we have also a more involved coinductive proof covering arbitrary processes.

We plan to further extend the results in this paper transferring them to the framework of nested trace semantics [13,2]. We hope in this way to achive a coinductive characterisation for every sematics in the linear time-branching time spectrum. This is a much more elaborate work mainly because nested trace semantics are defined in a rather less elegant way than nested simulations, since traces instead of single actions are used in the transitions appearing in their definition. In particular, nested trace semantics do not coincide with the nested trace simulation semantics defined in Section 5.

Finally, let us conclude noticing that our notion of counting bisimulation is a particular case of a general notion of vectorial bisimulation in which we are working. It would also include as other particular cases the recent notions of bisimulation on speed [16] and amortised bisimulations [14], and it is based on the idea of using

coalgebraic techniques to study not just a single relation but a (nearly arbitrary) family of them defined in a mutual recursive (coalgebraic) way. In [9] we have used the general concept of categorical simulation in [12] to start to formalize these ideas, and other interesting bisimulation semantics for distributed systems.

References

[1] Luca Aceto, Wan Fokkink, and Anna Ingólfsdóttir. 2-nested simulation is not finitely equationally axiomatizable. In *STACS 2001*, volume 2010 of *Lecture Notes in Computer Science*, pages 39–50. Springer, 2001.

[2] Luca Aceto, Rob van Glabbeek, Wan Fokkink, and Anna Ingólfsdóttir. Nested semantics over finite tree are equationally hard. *Information and Computation*, 191(2):203–232, 2004.

[3] Bard Bloom, Sorin Istrail, and Albert R. Meyer. Bisimulation can't be traced. *Journal of the ACM*, 42(1):232–268, 1995.

[4] Rance Cleaveland and Matthew Hennessy. Testing equivalence as a bisimulation equivalence. *Formal Aspects of Computing*, 3:1–21, 1992.

[5] Rance Cleaveland, Joachim Parrow, and Bernhard Steffen. The concurrency workbench: a semantics-based tool for the verification of concurrent systems. *ACM Trans. Program. Lang. Syst.*, 15(1):36–72, 1993.

[6] David de Frutos-Escrig and Carlos Gregorio-Rodríguez. Semantics equivalences defined with global bisimulations. Annual meeting of the IFIP Working Group 2.2, Bertinoro, Italy, September 2004.

[7] David de Frutos-Escrig and Carlos Gregorio-Rodríguez. Bisimulations up-to for the linear time-branching time spectrum. In *CONCUR 2005 - Concurrency Theory, 16th International Conference*, volume 3653 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2005.

[8] David de Frutos-Escrig and Carlos Gregorio-Rodríguez. Process equivalences as global bisimulations. *Journal of Universal Computer Science*, 12(11):1521–1550, 2006.

[9] David de Frutos-Escrig, Fernando Rosa-Velardo, and Carlos Gregorio-Rodríguez. New bisimulation semantics for distributed systems. In *FORTE 2007*, volume to appear of *Lecture Notes in Computer Science*. Springer, 2007.

[10] Rob J. van Glabbeek. *Handbook of Process Algebra*, chapter The Linear Time – Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes, pages 3–99. Elsevier, 2001.

[11] Jan Friso Groote and Frits Willem Vaandrager. Structured operational semantics and bisimulations as a congruence. *Information and Computation*, 100(2):202–260, 1992.

[12] Jesse Hughes and Bart Jacobs. Simulations in coalgebra. *Theoretical Computer Science*, 327(1–2):71–108, 2004.

[13] Matthew Hennessy and Robin Milner. Algebraic laws for nondeterminism and concurrency. *J.ACM*, 32:137–161, 1985.

[14] Astrid Kiehn and A. Arun-Kumar. Amortised bisimulations. In *Formal Techniques for Networked and Distributed Systems-FORTE 2005*, volume 3235 of *Lecture Notes in Computer Science*, pages 320–334. Springer, 2005.

[15] Paris C. Kanellakis and Scott A. Smolka. CCS expressions, finite state processes, and three problems of equivalence. *Information and Computation*, 86(1):43–68, 1990.

[16] Gerald Lüttgen and Walter Vogler. Bisimulation on speed: A unified approach. In *FOSSACS 2005*, volume 3341 of *Lecture Notes in Computer Science*, pages 79–94. Springer, 2005.

[17] Robin Milner. *Communication and Concurrency*. Prentice Hall, 1989.

[18] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal of Computing*, 16(6):973–989, 1987.

[19] Colin Stirling. The joys of bisimulation. In *MFCS'98*, volume 1450 of *Lecture Notes in Computer Science*, pages 142–151. Springer-Verlag, 1998.