New Bisimulation Semantics for Distributed Systems*

David de Frutos-Escrig, Fernando Rosa-Velardo, and Carlos Gregorio-Rodríguez

Dpto. de Sistemas Informáticos y Computación
Universidad Complutense de Madrid
{defrutos,fernandorosa,cgr}@sip.ucm.es

Abstract. Bisimulation semantics are a very pleasant way to define the semantics of systems, mainly because the simplicity of their definitions and their nice coalgebraic properties. However, they also have some disadvantages: they are based on a sequential operational semantics defined by means of an ordinary transition system, and in order to be bisimilar two systems have to be "too similar". In this work we will present several natural proposals to define weaker bisimulation semantics that we think properly capture the desired behaviour of distributed systems. The main virtue of all these semantics is that they are real bisimulation semantics, thus inheriting most of the good properties of bisimulation semantics. This is so because they can be defined as particular instances of Jacobs and Hughes' categorical definition of simulation, which they have already proved to satisfy all those properties.

1 Introduction

Bisimulation is a usual way to define the semantics of systems. It is defined starting from an operational semantics that defines the (low level) behaviour of the system as a labelled transition system (lts) whose states correspond to the possible internal states of the systems, while the transitions represent the change of state, observable by means of labels. Bisimulations have many pleasant theoretical and practical properties that justify its use to define the semantics of systems. At the theoretical level, bisimulations are the adequate way to define the behaviour of a system defined by a coalgebra $s : X \to \mathcal{P}(A \times X)$. They capture the idea that in order to be equivalent, two states must have two sets of labelled successors that have to be related in both directions: $\forall s \xrightarrow{a} s' \;\; \exists t \xrightarrow{a} t'$ with $(s', t') \in R$ and $\forall t \xrightarrow{a} t' \;\; \exists s \xrightarrow{a} s'$ with $(s', t') \in R$.

This only slightly generalizes the isomorphism of transition systems, mainly by taking into account the idempotent law. This means that the correspondence relating the a-successors of two related states do not need to be bijective. For instance, the relation $R = \{(x, y), (x_1, y_1), (x_2, y_1), (x_3, y_2), (x_3, y_3)\}$ is the smallest bisimulation relating the two states x and y of the two systems in Fig. 1.

* Work partially supported by the Spanish projects DESAFIOS TIN2006-15660-C02-02, WEST TIN2006-15578-C02-01 and PROMESAS-CAM S-0505/TIC/0407.

Fig. 1. Two bisimilar systems

Besides the simple and easy to manipulate way in which they are defined, bisimulations and the equivalence relation they induce, bisimilarity, satisfy many pleasant properties that have been thoroughly studied since they were introduced by Park [20]. For instance, we can prove that whenever the operational semantics of a language is defined by a SOS-system [21] of several quite large syntactical classes, such as the De Simone class [6], then bisimulation equivalence is a congruence with respect to all the syntactical constructors of the language.

At the practical level, bisimilarity is an interesting way to define the equivalence of two systems, since it can be checked by efficient algorithms [8]. When, instead, we prefer to use symbolic proofs to prove the equivalence between two systems described by two syntactical terms of a language, we can construct the corresponding bisimulation relating them by using quite powerful techniques such as bisimulation up-to [18].

The most important disadvantage of using bisimulation semantics is that bisimulation equivalence is a too coarse relation: all the extensional semantics that have been proposed to define the semantics of systems by adding some information to the quite simple trace semantics, such as the failure semantics or the readiness semantics, have less discriminatory power than the bisimulation equivalence, as we can see in the famous Van Glabbeek's spectrum [27].

Bisimulation is also too powerful with respect to the testing framework. This is also seen in [27]: copy and "parallel" testing are needed in order to characterize bisimulation equivalence as a testing equivalence. Besides, in [3] Bloom et al. have proved that ready simulation equivalence, that is also weaker than bisimilarity, is the strongest equivalence relation that is preserved by any operator defined by means of GSOS rules. We can sum up this discussion by saying that bisimulation equivalence is too fine because it forces the two compared transition systems to be "too similar". Our aim in this paper will be to present other bisimulation-like semantics that generalize the definition of plain bisimulations, by allowing us to get other equivalences between systems that we will naturally justify when comparing distributed systems.

Simulations are one of the first natural ways to relax the definition of bisimulation. In the one hand, because its definition is obtained by retaining just one half of the two symmetric parts of the definition of bisimulation. In this way, we obtain an order relation, similarity, that also has a coalgebraic definition. However, mutual simulation, that is again an equivalence relation, is not as powerful as bisimulation equivalence. We can try to enforce the simulation semantics by adding some additional constraints, getting for instance the ready simulations and the ready simulation equivalence. However, there is not any non-trivial order relation whose kernel is bisimilarity. Even so, simulations are a reasonable and useful way to compare two given systems, and also a powerful tool to define interesting equivalence relations, as ready similarity.

Another way to generalize the concept of bisimulation is by means of its categorical definition, by allowing any functor F in the definition of the coalgebras $a : X \to F(X)$ and $b : Y \to F(Y)$ to be related. Besides the seminal work on the subject [1], you can look at the wonderful monography [16] to find a thorough study of the subject. Even if it would be interesting to know all the technical details, in this paper we mainly pretend to motivate the use of several bisimulation-like equivalence relations, which can in fact be supported by all that abstract machinery. Therefore, we are both saying that those semantics can be formally defined, and have all the pleasant properties of bisimulation semantics; and we are proving that those general abstract studies have indeed a practical use, since these new interesting semantics can be obtained as particular instances of the bisimulation semantics they allow to define.

For instance, we will present "commutative bisimulation", that checks "from time to time", by means of some introduced "checkpoints" that the compared systems have executed the same actions, but possibly in a different order; and "action sets bisimulation", where we also introduce a simple definition of "distributed transition system". We also discuss "approximated bisimulation", where the compared systems need not to execute exactly the same actions but some "similar" ones; this includes the notion of amortized bisimulation, where the costs of the executed actions need to be only similar. All these bisimulation-like equivalences are weaker than strong bisimulation, so that they diminish the proof obligations imposed by the ordinary definition of bisimulation.

Although we will recall that categorical definition, and we will show how can be indeed used to define some of the semantics we propose, in this paper we will mainly focus on the presentation of these new semantics, leaving the details of their categorical definition to other more appropriate forum.

It is important to point out that although there were several proposals for bisimulations for distributed systems in the past, they were in the opposite direction to our approach, since they tried to capture the differences between systems induced by facts such as the location where the actions were executed, and therefore produce semantic equivalences finer than ordinary bisimilarity; instead, as said before, we are looking for coarser equivalences, which therefore are more easily accomplished.

The rest of the paper is structured as follows. Section 2 defines the new bisimulation-like semantics that we propose. Section 3 is a brief survey of abstract results on categorical bisimulations that can be applied to justify the coalgebraic character of all the new bisimulation notions that we have introduced. As an illustration of how this can be done we present the details for one of the semantics. Section 4 discusses some related work, and finally Sect. 5 briefly presents our conclusions and directions for future work.

2 Bisimulations for distributed systems

We have looked for several directions in which we could relax the definition of plain bisimulations getting nice weaker semantics which could be still rigorously

presented as coalgebraic semantics, thus preserving their good properties. Next we present those simplest proposals that, at the same time, seem to be more promising in practice.

2.1 Commutative bisimulations

There are several scenarios in which we are not interested in the order in which the actions are executed, but in the set of actions that is finally executed. If we only have finite sequential systems to compare, then we could define the trace semantics as a starting point, by applying the seq-to-multiset operator that transforms the sequence of executed actions into the corresponding multiset of actions. However, if we are considering reactive systems that possibly run forever, we need to consider adequate bisimulation-like versions of that intended semantics.

As a first proposal in this direction, we present *checkpoint commutative bisimulations*, that are defined by incorporating into the transition systems that define the operational semantics of our distributed systems a boolean attribute *checkpoint* that signals the times where we have to check for the equality of the multiset of actions that the systems have executed from their previous checkpoints.

We can describe the desired bisimulation equivalence using plain, but accurate words, as follows: in order to check if two states of two systems are equivalent, we will play the ordinary bisimulation game, but now we are not forced to replicate the execution of any action a by executing the same action in the other process; instead, we remember the multiset of actions executed through the paired computations until we arrive to a checkpoint. Then, the other process has to arrive to another checkpoint and the two remembered multisets of actions should be the same.

To formalize this new class of bisimulations we need to introduce those sets of remembered actions. This is done by defining our bisimulations not just as relations on states, but as relations on pairs $(s, m) \in S \times \mathcal{MS}(A)$, where s is a state and m a multiset of actions. This takes us to the following formal definitions.

Definition 1. $(S, A, \rightarrow, chk)$ *is an* lts with checkpoints *if* $(S, A, \rightarrow)$ *is an ordinary lts and* $chk : S \rightarrow \{0, 1\}$ *is the characteristic function of a set of so called checkpoints of the system.*

Definition 2. *A* commutative checkpoint bisimulation *relating states of an lts with checkpoints* $(S, A, \rightarrow, chk)$ *is a relation* $R \subseteq (S \times \mathcal{MS}(A)) \times (S \times \mathcal{MS}(A))$ *that satisfies:*

- $(s_1, m_1)R(s_2, m_2) \wedge (chk(s_1) \vee chk(s_2)) \Rightarrow chk(s_1) \wedge chk(s_2) \wedge m_1 = m_2$,
- $(s_1, m_1)R(s_2, m_2) \wedge s_1 \xrightarrow{a} s_1' \Rightarrow \exists s_2 \xrightarrow{b} s_2' \wedge (s_1', m_1 + \{a\})R(s_2', m_2 + \{b\})$,
- $(s_1, m_1)R(s_2, m_2) \wedge s_2 \xrightarrow{b} s_2' \Rightarrow \exists s_1 \xrightarrow{a} s_1' \wedge (s_1', m_1 + \{a\})R(s_2', m_2 + \{b\})$,

where $+$ *represents the union of multisets.*

Fig. 2. Checkpoint bisimilar states

As usual, we say that (s_1, m_1) and (s_2, m_2) are *checkpoint bisimilar*, and we write $(s_1, m_1) \sim_{chk} (s_2, m_2)$, if and only if there exists a commutative checkpoint bisimulation R such that $(s_1, m_1)R(s_2, m_2)$. We simply say that s_1 and s_2 are checkpoint bisimilar, and we also write $s_1 \sim_{chk} s_2$, if and only if $(s_1, \emptyset) \sim_{chk} (s_2, \emptyset)$.

First notice that in order to simplify the definition above, we are remembering the complete multiset of executed actions from the very beginning, and not only from the last checkpoint. If we prefer to faithfully capture that more local memory constraint, it is easy to check that changing the second condition in Def. 2 by the following one

$$(s_1, m_2)R(s_2, m_2) \wedge s_1 \xrightarrow{a} s'_2 \Rightarrow \exists s_2 \xrightarrow{b} s'_2 \wedge \begin{cases} chk(s_1) \Rightarrow (s'_1, \{a\})R(s'_2, \{b\}) \\ \neg\, chk(s_1) \Rightarrow (s'_1, m_1 + \{a\})R(s'_2, m_2 + \{b\}) \end{cases}$$

and similarly for the third condition, we obtain an equivalent definition.

As a first and trivial example, let us consider the lts with checkpoints in Fig. 2. In it, we denote by c's the states which are checkpoints. Then, trivially the states c_1 and c_2 are checkpoint bisimilar. Indeed, the relation $R = \{\langle (c_0, \emptyset), (c_1, \emptyset)\rangle, \langle (s_1, \{a\}), (s_2, \{b\})\rangle, \langle (c'_1, \{a, b\}), (c'_2, \{a, b\})\rangle\}$ is a checkpoint bisimulation.

As it has been done many other times in the past, once we have a bisimulation-like definition of an equivalence relation, we could prove one by one all the properties of such a relation. However, what we advocate here is the use of the general results that have been recently developed in a general framework, so that those properties are obtained just for free, as particular cases of those general results. We will recall in Sect. 3 some of those general results and the way in which they can be used to prove that all the bisimulation-like semantics proposed in this paper have, indeed, a pure coalgebraic flavour.

2.2 Amortized commutative bisimulation

One could argue that the use of checkpoints is not very natural, although we could give some examples where they can be introduced in a quite simple way. For instance, we could consider the comparison between two search engines that collect information in the web in two different ways. In this case, the checkpoints correspond to the points in which they have completed a search: it is at that time that we have to compare the results of the search.

However, we could prefer a more "continuous" equivalence where the comparison is done after each step of the bisimulation game, although allowing multiple

steps in order to allow the interleaving of other actions whenever we need to replicate the execution of a given action. In order to make easier the presentation of this semantics, we prefer to start in this case by the formal definitions.

Definition 3. *Given a transition system* $(S, A, \rightarrow)$*, we define the* step transition system *induced by it as* $(S, A^*, \Rightarrow)$*, where* $s \stackrel{\alpha}{\Rightarrow} s'$ *with* $\alpha = a_1 \ldots a_n$ *if and only if*

$$s = s_0 \stackrel{a_1}{\rightarrow} s_1 \ldots s_i \stackrel{a_{i+1}}{\rightarrow} s_{i+1} \ldots s_{n-1} \stackrel{a_n}{\rightarrow} s_n = s'$$

Definition 4. *An* amortized commutative bisimulation *relating states of an lts* $(S, A, \rightarrow)$ *is a relation* $R \subseteq (S \times \mathcal{MS}(A)) \times (S \times \mathcal{MS}(A))$ *that satisfies*

- $(s_1, m_1)R(s_2, m_2) \wedge s_1 \stackrel{a}{\rightarrow} s_1' \Rightarrow \exists s_2 \stackrel{\alpha}{\Rightarrow} s_2' \quad m_1 + \{a\} \subseteq m_2 + \{\alpha\}$ *and* $(s_1', \emptyset)R(s_2', m)$ *with* $m + m_1 + \{a\} = m_2 + \{\alpha\}$,
- $(s_1, m_1)R(s_2, m_2) \wedge s_2 \stackrel{a}{\rightarrow} s_2' \Rightarrow \exists s_1 \stackrel{\alpha}{\Rightarrow} s_1' \quad m_2 + \{a\} \subseteq m_1 + \{\alpha\}$ *and* $(s_1', m)R(s_2', \emptyset)$ *with* $m + m_2 + \{a\} = m_1 + \{\alpha\}$,

where by abuse of notation we take $\{\alpha\} = \{a_1, \ldots, a_n\}$ *if* $\alpha = a_1 \ldots a_n$.

In this case we could start by considering only the pairs $\langle (s_0, m_0), (s_1, m_1) \rangle$ with $m_0 = \emptyset \vee m_1 = \emptyset$. Then we could see the corresponding set $m_i \neq \emptyset$ as the stock accumulated by s_i when comparing it with s_{1-i}.

We could also consider a restricted variant where the size of this stock is somehow bounded. For instance, given a size bound B we could impose to the sets $m_i = \emptyset$, $m_{1-i} \neq \emptyset$ that $|m_{1-i}| \leq B$, in order to define the corresponding bisimilarity $\sim^B_{acb}$. The idea is that we cannot execute too many other actions in advance when simulating the execution of an action a.

If we disregard checkpoints in Fig.2 then states c_1 and c_2 are amortized bisimilar, since the following relation is an amortized bisimulation.

$$R = \{\langle (c_1, \emptyset), (c_2, \emptyset) \rangle, \langle (s_1, \emptyset), (c_2', \{b\}) \rangle, \langle (c_1', \{a\}), (s_2, \emptyset) \rangle, \langle (c_1', \emptyset), (c_2', \emptyset) \rangle\}$$

2.3 Idempotent bisimulations

If we assume that the execution of actions should be not only commutative, but also idempotent, so that after executing once an action a the repeated execution of that action is of no use but has no negative consequence either, then we are in a scenario where we should use the powerset constructor $\mathcal{P}$ instead of using multisets. Then we can define an exact ic-bisimulation as follows:

Definition 5. *An* exact ic-bisimulation *relating states of* $(S, A, \rightarrow)$ *is a relation* $R \subseteq S \times S \times \mathcal{P}(A)$ *that satisfies*

- $(s_1, s_2, P) \in R$, $s_1 \stackrel{a}{\rightarrow} s_1' \Rightarrow \exists s_2 \stackrel{\alpha}{\Rightarrow} s_2'$ *such that* $P \cup \{a\} = P \cup \{\alpha\}$ *and* $(s_1', s_2', P \cup \{a\}) \in R$
- $(s_1, s_2, P) \in R$, $s_2 \stackrel{a}{\rightarrow} s_2' \Rightarrow \exists s_1 \stackrel{\alpha}{\Rightarrow} s_1'$ *such that* $P \cup \{a\} = P \cup \{\alpha\}$ *and* $(s_1', s_2', P \cup \{a\}) \in R$.

We define as usual the corresponding bisimilarity notion $\sim_{eic}$.

Note that in this case we do not need two sets of remembered actions because the related states have to correspond to the common set of executed actions P. Instead, we need a perpetuous memory, since we consider that the repeated execution of an action, from the very beginning, does not have any consequence, so that it can be replicated by executing any sequence of actions in P^*. We could also imagine that once an action has been executed, and therefore included in the set of executed actions P, from then on the repeated execution of actions in P behaves as if they had become internal actions, so that we could also say that our ic-bisimulations is a kind of dynamic weak bisimulation.

Besides, we could define the corresponding amortized ic-bisimulations and bounded versions of these new bisimilarity notions, where we can also limit the length of the replicating sequences α. This would be related with efficiency issues, in which we want to impose the condition that the number of actions executed by comparable computations of two bisimilar processes will be somehow similar.

Obviously, we can also define checkpoint idempotent bisimulations, although in this case we should also allow replicating steps $\stackrel{\alpha}{\Rightarrow}$ in the right-hand side of the defining conditions, since due to the idempotence of actions we could need to repeat the execution of some actions in order to reach the adequate bisimilar state, so that the lengths of two equivalent computations could be different.

2.4 Amortized quantitative bisimulation

There have recently been two approaches to amortized bisimulation [15, 30], where the authors had to develop by hand the corresponding theories, in order to proof the good properties of the new bisimulation notions they introduce. These amortized notions, besides the replication of the execution of an action, impose that the total costs of the actions executed by two comparable computations are somehow similar. Next we present our simple proposal for a symmetric notion of amortized bisimulation.

Definition 6. *A* weighted lts *is a tuple* $(S, A, \rightarrow, w)$ *where* $(S, A, \rightarrow)$ *is an lts and* $w : \{s \stackrel{a}{\rightarrow} s' \in \rightarrow\} \rightarrow \mathcal{P}(\mathbb{R}^+)$.

The function w represents the cost of the execution of a transition. It returns a set of possible costs, because once we have represented the set of transitions as a set, and not as a multiset, this is the way we can represent the possibility of having several ways, with different costs, to execute the same transition.

From now on, we write just $s \stackrel{a}{\underset{c}{\rightarrow}} s'$ whenever $c \in w(s \stackrel{a}{\rightarrow} s')$.

Definition 7. *An amortized bisimulation relating states of* $(S, A, \rightarrow, w)$ *for the absolute bound* $B \in \mathbb{R}^+$ *is a relation* $R \subseteq S \times S \times [-B, B]$ *that satisfies*

- $(s_1, s_2, d) \in R \wedge s_1 \stackrel{a}{\underset{c_1}{\rightarrow}} s_1' \Rightarrow \exists s_2 \stackrel{a}{\underset{c_2}{\rightarrow}} s_2'$ *and* $(s_1', s_2', d - c_1 + c_2) \in R$,
- $(s_1, s_2, d) \in R \wedge s_2 \stackrel{a}{\underset{c_1}{\rightarrow}} s_2' \Rightarrow \exists s_1 \stackrel{a}{\underset{c_2}{\rightarrow}} s_1'$ *and* $(s_1', s_2', d - c_1 + c_2) \in R$.

We write $\sim_{ab}^{B}$ for the amortized bisimilarity relation. As for any other relation expressing an inexact or approximated equivalence, these amortized bisimilarity relations are not equivalence relations, because we can have $P_1 \sim_{ab}^{B} P_2 \sim_{ab}^{B} P_3$ but not $P_1 \sim_{ab}^{B} P_3$. Instead, they behave as a distance measure, so that we have $P_1 \sim_{ab}^{B_1} P_2 \sim_{ab}^{B_2} P_3 \Rightarrow P_1 \sim_{ab}^{B_1+B_2} P_3$. Oppositely to what was done in [15], we have defined a symmetric relation that can be read as "similarly fast on the large", and not an order relation "amortized faster". We could get an equivalence relation related to the amortized costs by taking $\sim_{ab} = \bigcup \sim_{ab}^{B}$. Obviously, this would be the full relation if we just considered finite processes, but it becomes interesting for infinite behaviours where this coalgebraic notion accurately reflects the notion of "equal amortized cost".

We can also define an exact distance relation between processes by taking $d_{ab}(P,Q) = min\{B \mid P \sim_{ab}^{B} Q\}$, which has all the properties imposed to a topological distance relation.

Instead of a pure absolute amortized character that imposes the common bound B, that does not take into account the length of computations, we could also define a relativized amortized bisimilarity as follows

Definition 8. *A* relativized amortized bisimulation *relating states of* $(S, A, \rightarrow, w)$ *for the margin* $B \in \mathbb{R}^+$ *is a relation* $R \subseteq (S, S, \mathbb{R}, \mathbb{N})$ *that satisfies:*

- $(s_1, s_2, r, n) \in R \Rightarrow |r| \leq B \cdot n$,
- $(s_1, s_2, r, n) \in R \wedge s_1 \xrightarrow[c_1]{a} s_1' \Rightarrow \exists s_2 \xrightarrow[c_2]{a} s_2' \quad (s_1', s_2', r - c_1 + c_2, n+1) \in R$,
- $(s_1, s_2, r, n) \in R \wedge s_2 \xrightarrow[c_2]{a} s_2' \Rightarrow \exists s_1 \xrightarrow[c_1]{a} s_1' \quad (s_1', s_2', r - c_1 + c_2, n+1) \in R$.

We write $\sim_{ra}^{B}$ *for the relativized amortized bisimilarity relation.*

It is clear that this relativized notion is closer to the simple approximated cost bisimilarity that just imposed the simulation of the execution of an action with a given cost by executing the same action with a similar cost.

2.5 Bisimulations with non-atomic actions

In order to prepare the field for other more interesting examples, here we discuss the case in which the transitions are labelled not with a single action but with a multiset of actions. Then we can replicate the executions of $\xrightarrow{C}$ with $C \subseteq A$ by executing $\overset{\overline{C}}{\Rightarrow}$ with $\overline{C} = C_1 \cdot \ldots \cdot C_k$ and $C = \bigcup_{i=1}^{k} C_i$, to get a plain non-atomic actions bisimulation, whose induced bisimilarity relation we denote by $\sim_{naa}$. It is immediate to define the corresponding non-atomic actions versions of our checkpoint, idempotent or amortized quantitative bisimulations.

2.6 Distributed bisimulations

Let us now consider the case in which we have distributed systems composed by agents that execute their actions in parallel. A first simple proposal corresponds to the case in which any agent is just a state of a common ordinary lts.

Definition 9. *A* plain distributed bisimulation *relating multisets of states of* $(S, A, \rightarrow)$ *is a relation* $R \subseteq \mathcal{MS}(S) \times \mathcal{MS}(S)$*, that satisfies:*

- $(M_1, M_2) \in R$, $\{s_1^1, \ldots, s_k^1\} = N_1 \subseteq M_1 \wedge \forall i \in \{1, \ldots, k\} \;\; s_i^1 \stackrel{a_i}{\rightarrow} s_i'^1 \Rightarrow \exists N_2 = \{s_1^2, \ldots, s_k^2\} \subseteq M_2, \forall i \in \{1, \ldots, k\} \;\; s_i^2 \stackrel{a_i}{\rightarrow} s_i'^2 \wedge (M_1', M_2') \in R$, *where* $M_j' = M_j - N_j + \{s_1'^j, \ldots, s_k'^j\}, \; \forall j \in \{1, 2\}$,
- $(M_1, M_2) \in R$, $\{s_1^2, \ldots, s_k^2\} = N_2 \subseteq M_2 \wedge \forall i \in \{1, \ldots, k\} \;\; s_i^2 \stackrel{a_i}{\rightarrow} s_i'^2 \Rightarrow \exists N_1 = \{s_1^1, \ldots, s_k^1\} \subseteq M_1, \forall i \in \{1, \ldots, k\} \;\; s_i^1 \stackrel{a_i}{\rightarrow} s_i'^1 \wedge (M_1', M_2') \in R$, *where* $M_j' = M_j - N_j + \{s_1'^j, \ldots, s_k'^j\}, \; \forall j \in \{1, 2\}$.

We say that two systems given by two multisets of actions M_1 *and* M_2 *are distributely bisimilar, and we write* $M_1 \sim_d M_2$*, if there exists a distributed bisimulation that contains the pair* (M_1, M_2)*.*

Under this simple definition, it is clear that in order to be distributely bisimilar, two systems must have the same set of non-completed agents, where we say that s is a completed agent if there is no transition $s \stackrel{a}{\rightarrow} s'$. Instead, the defined equivalence already has an interesting parallel character, so that it does not coincide with the plain bisimulation equivalence that would be obtained by considering the corresponding interleaving semantics.

There are many ways in which we can get more realistic distributed bisimulation notions by extending or modifying the definition above, either by modifying the conditions imposed to the bisimulations, or by defining an adequate notion of distributed transition system.

The first proposal in the first direction is just the combination of the definitions of both distributed and non-atomic actions bisimulation, thus making possible to replicate the simultaneous execution of $s_i^1 \stackrel{C_i^1}{\rightarrow} s_i'^1$ with $N_1 = \{s_1^1, \ldots, s_k^1\} \subseteq M_1$, by means of $N_2 = \{s_1^2, \ldots, s_l^2\} \subseteq M_2$ with $s_j^2 \stackrel{C_j^2}{\rightarrow} s_j'^2$ and $\bigcup_{i=1}^{k} C_i^1 = \bigcup_{j=1}^{l} C_j^2$.

We could also remove the partial synchronous character of this definition by allowing the sequential firing of transitions in the replicating system, thus getting $s_j^2 \stackrel{\overline{C_j^2}}{\Rightarrow} s_j'^2$ with $\bigcup_{i=1}^{k} C_i^1 = \bigcup_{j=1}^{l} \overline{C_j^2}$, where by abuse of notation we are identifying the sequences of multisets $\overline{C_j^2}$ with the multiset composed of its elements.

Obviously, starting from these asynchronous, non-atomic actions, distributed semantics, we could easily define the corresponding checkpoint idempotent or amortized quantitative bisimulation.

In the opposite direction, we could define specific notions of distributed lts's by incorporating special transitions for the creation of agents, or mechanisms to synchronize the firing of transitions when needed. We do not need a special mechanism for the removal of agents since that can be easily represented by means of completed states of the system. Just to give a concrete proposal, which is at the same time flexible and simple, we present the following:

Definition 10. *A* distributed transition system *is a tuple* $(S, A, \mapsto)$ *where* S *is a set of states,* A *is a set of actions (possibly somehow structured) and* $\mapsto$ *is*

a distributed transition relation, which means $\mapsto \subseteq S \times A \times \mathcal{P}(S)$*. A concrete distributed system based on* $(S, A, \mapsto)$ *is just a multiset* $M \in \mathcal{MS}(S)$*. We call each state in* M *an agent of the system.*

In order to impose the adequate synchronization conditions we introduce the following firing rule for distributed transitions:

Definition 11. *We define a synchronized distributed system as a pair* $\langle (S, A, \mapsto), \mathcal{Z} \rangle$*, where* $(S, A, \mapsto)$ *is a plain distributed transition system and* $\mathcal{Z} \subseteq \mathcal{MS}(A)$ *defines the allowed steps of the computations of the system: given a concrete system for it* $M \in \mathcal{MS}(S)$*, we say that* $M \overset{Z}{\Rrightarrow} M'$ *is a computation step of the system if* $Z = \{a_1, \ldots, a_n\} \in \mathcal{Z}$ *and there exists* $N = \{s_1, \ldots, s_k\} \subseteq M$ *with* $s_i \overset{a_i}{\mapsto} S'_i$ *for all* $i \in \{1, \ldots, k\}$ *and* $M' = M - N + \sum_{i=1}^{k} S'_i$*.*

This is indeed quite a general synchronization framework that allows the consideration of autonomous actions that can be executed by a single agent without having to synchronize ($\{a\} \in \mathcal{Z}$), pairs of synchronizing actions in the CCS style ($\{a, \overline{a}\} \in \mathcal{Z}$), and general synchronizing steps ($Z \in \mathcal{Z}$) as they were introduced in E-LOTOS [13]. The framework even considers broadcasting scenarios: if a represents the communication of an action, and $\overline{a}_1, \ldots, \overline{a}_k$ represent the reception of that information by all the "participants" of the system, so that at least one agent of each participant receives the information, then we can represent this scenario by having $(\{a\} + \sum_{i=1}^{k} k_i \cdot \overline{a}_i) \in \mathcal{Z}$ if and only if for all $i \in \{1, \ldots, k\}$ $k_i \geq 1$. We have used an instance of this synchronization model in our ubiquitous nets [10], where we have both autonomous transitions and synchronization transitions that represent the offering and request of services to providers.

3 A quick survey on useful abstract bisimulation results

As we said in the introduction, one of the main objectives of this introductory paper is to establish a bridge between the existing theoretical results that could support our Formal Methods and the concrete application of these results. Whenever the need for new formal methods is detected in one field, we always start by developing ad-hoc theories that are as simple as possible, but close enough to the concrete application that has motivated its introduction. Certainly, these first steps are usually only partially satisfactory from both points of view: the theories are not too general, and at the same time they use to be unnecessarily involved and even clumsy; on the other side, they are only adequate to solve simple cases, or cover partial aspects of what we want to cope in our applications.

When a successful, or at least quite promising new theory attracts the attention of both theoreticians and practitioners we hopefully get quite a heap of nice theoretical results and suggestions for interesting applications. But the problem appears when both communities separate each other because the theoretical studies need quite complicate foundations that produce involved theories

that practitioners cannot understand in detail. In many cases this produces a negative attitude which, at the end, even considers those theoretical studies as useless, since they seem unapplicable in practice. On the other side, those nice theories become even more difficult to be understood because nobody looks for interesting and simple examples which, besides illustrating them, constitute a concrete case in which many useful results can be obtained for free, once it is presented as an instance of the general theory first produced.

The formal theory whose great interest we want to illustrate by the long collection of complex notions of bisimulations for distributed systems presented in the previous section, is that of categorical bisimulations [1, 23, 16], that provide a general notion of bisimulation; and categorical simulations [14], that more than a general notion of simulation provide a relaxation of the notion of bisimulation that preserves most of its coalgebraic framework, thus maintaining most of its nice (co)algebraic properties. By lack of space, we cannot give here even their formal definitions in full detail. You could check (and hopefully read in detail) the beautiful studies cited above to look for the details.

We can see a functor $F : Sets \to Sets$ as a constructor of the "set of successors" of the states of a class of systems. Besides, we need a natural translation of the functions relating two sets of states, that preserves composition and identity functions, that is, $\forall\ f : X \to Y,\ g : Y \to Z,\ F(g \circ f) = F(g) \circ F(f)$ and $F(Id_X) = Id_{F(X)}$.

For instance, for the notion of commutative checkpoint bisimulation in Sect. 2.1 we would need a functor $F_{chk}(\overline{X}) = \{0,1\} \times \mathcal{MS}(A) \times \mathcal{P}(A \times \overline{X})$, where roughly the elements of $\overline{X}$ correspond to the tuples in $\{0,1\} \times S \times \mathcal{MS}(A)$, so that they keep memory of the multiset of executed actions since the last checkpoint, and indicate us if that state is a checkpoint or not.

F-coalgebras are just functions $\alpha : X \longrightarrow FX$. Then F-bisimulations can be characterized by means of spans, using the general categorical definition by Aczel and Mendler [1]:

$$\begin{array}{ccccc} X & \xleftarrow{r_1} & R & \xrightarrow{r_1} & Y \\ {\scriptstyle c}\downarrow & & {\scriptstyle e}\downarrow & & \downarrow{\scriptstyle d} \\ FX & \xleftarrow{Fr_1} & FR & \xrightarrow{Fr_2} & FY \end{array}$$

R is a bisimulation iff it is the carrier of some coalgebra e making the above diagram commute, where the r_i are the projections of R into X and Y.

We can also define them by relation lifting: given $R \subseteq X \times Y$, we take

$$Rel(F)(R) = \{(u,v) \in FX \times FY \mid \exists w \in F(R)\ \ u = Fr_1(w) \wedge v = Fr_2(w)\}$$

Then, F-bisimulations are just the support of any $Rel(F)$-coalgebra.

We will also need the general concept of simulation introduced by Hughes and Jacobs [14] using orders on functors. Let $F : Sets \to Sets$ be a functor. An order on F is defined by means of a functorial collection of preorders $\sqsubseteq_X \subseteq FX \times FX$ that must be preserved by renaming: for every $f : X \longrightarrow Y$, if $u \sqsubseteq_X u'$ then $Ff(u) \sqsubseteq_Y Ff(u')$.

Given an order $\sqsubseteq$ on F, a $\sqsubseteq$-simulation for coalgebras $c : X \longrightarrow FX$ and $d : Y \longrightarrow FY$ is a relation $R \subseteq X \times Y$ such that

$$\text{if } (x, y) \in R \text{ then } (c(x), d(y)) \in \mathrm{Rel}(F)_{\sqsubseteq}(R),$$

where $\mathrm{Rel}(F)_{\sqsubseteq}(R)$ is $\sqsubseteq \circ \mathrm{Rel}(F)(R) \circ \sqsubseteq$, which can be expanded to

$$\mathrm{Rel}(F)_{\sqsubseteq}(R) = \{(u, v) \mid \exists w \in F(\mathcal{R}).\ u \sqsubseteq Fr_1(w) \wedge Fr_2(w) \sqsubseteq v\}\,.$$

As we discuss in [12], it could be argued that the class of simulations obtained in this way is perhaps too broad. For example, we would expect simulations to be asymmetric order relations. However, equivalence (functorial) relations, represented by $\equiv$, are a particular class of orders on F, thus generating the corresponding class of $\equiv$-simulations. As it is the case for ordinary bisimulations, $\equiv$-simulations themselves need not be equivalence relations, but the induced notion of $\equiv$-similarity clearly is.

Let us briefly explain what is the idea behind this quite nice relaxation of the notion of F-bisimulation: any F-bisimulation has to satisfy a local coherency condition which roughly says that the successors of two related states $(s_1, s_2) \in R$ can be paired each other getting the same attributes when comparing information not in X, and states also related by R, when we compare elements in X. The introduction of the order $\sqsubseteq$ allows us to change these sets of successors according to it, before comparing them as indicated above. Obviously, the possibility of modifying those sets makes it easier to get the needed correspondence and, therefore, for any order $\sqsubseteq$ on F, the corresponding $\sqsubseteq$-similarity relation is weaker than F-bisimulation. In particular, by means of the adequate orderings, we will be able to relax the condition imposed by bisimulations: any information in the successors of two related states not corresponding to the "reached sets" must be exactly the same.

As a consequence, we cannot define any of our bisimulation notions that need the use of any kind of memory as plain F-bisimulations. Instead, we can capture those notions of memory and the necessary comparisons between them by means of the adequate notion of order on F. Next, we will illustrate all this by means of our first notion of commutative checkpoint bisimulation. For the functor F_{chk} we define the equivalence $\equiv_{chk}$ as follows:

- $(0, M, T) \equiv_{chk} (0, M', T')\ \forall M, M', T, T'$ with

$$T = \{(a_i, s_i) \mid i \in \{1, \ldots, k\}\} \Leftrightarrow T' = \{(a'_i, s_i) \mid i \in \{1, \ldots, k\}\},$$

- $(1, M, T) \equiv_{chk} (1, M, T')\ \forall M, T, T'$ with

$$T = \{(a_i, s_i) \mid i \in \{1, \ldots, k\}\} \Leftrightarrow T' = \{(a'_i, s_i) \mid i \in \{1, \ldots, k\}\}.$$

The idea is that whenever we are in a checkpoint the remembered multiset of executed actions must be the same, so that $\equiv_{chk}$ does not allow to change them. However, if we are not in a checkpoint, we do not need to compare the remembered multisets at all. This is why $\equiv_{chk}$ allows to change any of the

compared values, thus making it equal to the other in order to satisfy the equality imposed "in the middle of the condition" defining $\equiv_{chk}$-simulations. Note also that the actions executed in the transitions need not to be compared, so that we can always change an action a_i by any other a'_i.

In many of the bisimulation notions that we have defined in this paper, we need to consider transition sequences instead of plain transitions, for instance when defining our amortized commutative bisimulations. Certainly, all these equivalence notions could be studied by means of the derived step transition system $\stackrel{\alpha}{\Rightarrow}$, as it is done when characterizing the ordinary weak bisimulation as a strong bisimulation on the expanded system $\Rightarrow$. However, we do not want to explicitly construct such a tremendous system which, in fact, presents any computation of the original system as a single transition of the derived step transition system, thus completely losing the ability of reasoning on the full behaviour of a system in a local way. In other words, by expanding the original transition system and then defining bisimulation relations we are apparently still using a coalgebraic language, but the spirit of coinduction that means getting global properties by local reasonings has completely disappeared in practice.

There are a few recent works on the categorical definition of weak bisimulation and step semantics. In particular, a part of the results and techniques used in [26] can be used to formalize several of the new bisimulation notions introduced in this paper, following the general ideas sketched in [25]. Another more technical approach to the subject is that in [22], which needs a more careful study and more developments in order to find the way of using their ideas easily.

4 (Not so much) related work

Since its official introduction in [18], although we can find some related concepts in several older works devoted to different subjects, as explained in [24], quite a number of generalizations of the bisimulation equivalence have been proposed. However, these generalizations tend to preserve more of the structure of processes, thus obtaining even finer equivalences than bisimulation. For instance, in [5] Castellani et al. define a so called distributed bisimulation that deals with the distributed nature of processes, by distinguishing between concurrent processes and nondeterministic but sequential processes. As a consequence, the processes $a|b$ and $ab + ba$ are not identified by this semantics.

In [4], Boudol et al. follow the same intention, that of defining a notion of bisimulation that distinguishes between concurrency and sequential non-determinism. However, unlike in [5], where the authors focus on the distributed nature of processes, here the authors focus on the atomicity of actions by adding extra structure in the labels of transitions, which become partially ordered sets. Again, the resulting bisimulation semantics is stronger than strong bisimulation. For instance, processes $a|b$, $ab + ba$ and $ab + (a|b) + ba$ are all distinct with respect to that semantics.

Another interesting collection of works, that in this case also introduce a general categorical approach based on so called open maps, is [17, 9, 19], where

Fig. 3. Bisimilar but not FC-bisimilar nets

again a stronger semantics based on event structures that capture the causal relation between actions is studied. History-preserving bisimulation studied by W.Vogler [28] and Maximality preserving bisimulation [7] are other bisimulation semantics for Petri Nets and related models that are based on the so called process semantics for them. This kind of semantics became very popular in the first nineties when action refinement was studied in depth looking for a modular semantics that would be preserved by the implementation of complex actions by means of the corresponding processes (see for instance [29]). In the same direction we can find [2], that presents FC-bisimulation (standing for Fully Concurrent), based on the process semantics of Petri nets, also preserving the level of concurrency. For instance, the two simple nets in Fig. 3 are strong bisimilar, but not FC-bisimilar.

However, when we tried to find previous work on weaker bisimulation semantics we have found nearly nothing, out of, of course, anything related with the classical weak bisimulation. Probably, there is a formal reason why that is the case: any classical bisimulation equivalence imposes the equality of all the compared information, out of the consideration of the compared states themselves, and besides, it has to be defined in a local way. Both conditions produce rather strong equivalences as discussed above.

In order to get weaker equivalences one possibility is to consider adequate bisimulation up-to relations, as we have successfully done in [11], getting coalgebraic characterizations of any semantics in Van Glabbeek's spectrum [27]. The other possibility is to consider categorical simulations, as we have explored in this paper. As a matter of fact, there are some connections between these two approaches, since both relax the proof obligations imposed by the clauses defining bisimulations, by introducing up-to mechanisms. However, an important difference is that orders on functors can only be based in local information in the successors of the compared states, and thus categorial simulations have many pleasant coalgebraic properties. Instead, in [11] we had to renounce to these pure local definitions, since we wanted to characterize all the classical extensional semantics, such as failures or trace semantics, that cannot be captured by local conditions.

5 Conclusions and future work

By means of the new coalgebraic semantics for distributed systems presented in this paper, we have tried to narrow the gap between theoretical developments on categorical bisimulations and the applications of coalgebraic techniques to define and study new interesting semantics for distributed systems. Certainly, this is just an introductory paper that, however, already shows the applicability of some recent general results on categorical simulations and categorical weak

bisimulations. These results allow us to guarantee that our new bisimulation-like semantics are indeed coalgebraically based, so that they have all the good properties of this kind of semantics, without the need to prove them again, because they were established and proved once and forever.

There are two directions for further work on the subject: we have to present in detail the reformulations of our new semantics in the categorical framework. We have already done it for most of the semantics presented in the paper, either by directly presenting them as instances of the categorical definition of simulation or by using a step semantics defined by hand, for the cases in which we need to consider sequences of transitions in the definitions. As mentioned above, there is not a general theory for categorical step semantics available yet, and therefore in this case we need either to wait for those general results or to apply the particular cases that have already been solved, which fortunately correspond in particular to the functors defining the kind of transition systems in which we are interested.

Concerning the applications, we hope to motivate the people working in the field to consider the new semantics introduced in this paper, looking for those that could be more useful in practice. Practitioners have always considered bisimulation semantics not so useful because the equivalence it defines is too strong. By relaxing the conditions to become equivalent, but maintaining the good properties of coalgebraic semantics, we could obtain new promising semantics, and then develop for them all the machinery that makes applicable in practice the bisimulation semantics.

References

[1] P.Aczel and N.P.Mendler. A final coalgebra theorem. In David H. Pitt, David E. Rydeheard, Peter Dybjer, Andrew M. Pitts, and Axel Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Lecture Notes in Computer Science*, pages 357–365. Springer-Verlag, 1989.

[2] E.Best, R.Devillers, A.Kiehn, and L.Pomello. *Concurrent bisimulation in Petri nets.* Acta Informatica 28:231-264. Springer, 1991.

[3] B.Bloom, S.Istrail, and A.R.Meyer. *Bisimulation can't be traced.* Journal of ACM 42(1):232-268, 1995.

[4] G.Boudol, and I.Castellani. *Concurrency and atomicity.* Theoretical Computer Science 59:25-84. Elsevier, 1988.

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

[6] R.de Simone. *Higher-Level Synchronising Devices in Meije-SCCS.* Theoretical Computer Science 37:245-267. Elsevier, 1985.

[7] R.R. Devillers. *Maximality Preserving Bisimulation.* Theor. Comput. Sci., vol. 102(1), pages 165-183. Elsevier, 1992.

[8] A.Dovier, C.Piazza, and A.Policriti. *An efficient algorithm for computing bisimulation equivalence.* Theoretical Computer Science 311:221-256. Elsevier, 2004.

[9] M.P. Fiore, G.L.Cattani, and G.Winskel. *Weak Bisimulation and Open Maps.* In Logic in Computer Science, LICS'99, pp. 67-76. IEEE Computer Society, 1999.

[10] D. Frutos-Escrig, O. Marroquín-Alonso and F. Rosa-Velardo. *Ubiquitous Systems and Petri Nets.* Ubiquitous Web Systems and Intelligence, LNCS, vol. 3841. Springer, 2005.

[11] D. Frutos-Escrig, and C. Gregorio-Rodríguez. *Bisimulations Up-to for the Linear Time Branching Time Spectrum.* In 16th Int. Conf. on Concurrency Theory, CONCUR'05, LNCS, vol. 3653, pp. 278-292. Springer, 2005.

[12] D.Frutos-Escrig, M.Palomino, and I.Fábregas. Searching for a canonical notion of simulation. In preparation.

[13] G.F. Lucero, and J.Quemada. *Specifying the ODP Trader: An Introduction to E-LOTOS.* In 10th Int.Conf. on Formal Description Techniques and Protocol Specification, Testing and Verification, FORTE'97. IFIP Conference Proceedings, vol. 107, pages 127-142. Chapman & Hall, 1998.

[14] J.Hughes and B.Jacobs. Simulations in coalgebra. *Theoretical Computer Science*, 327(1-2):71–108, 2004.

[15] A.Kiehn, and S. Arun-Kumar. *Amortized bisimulations.* In Farn Wang, editor, FORTE'05, *LNCS* vol. 3731, pp. 320-334. Springer, 2005.

[16] B.Jacobs. Introduction to coalgebra. Towards mathematics of states and observations. Book in preparation. Available at http://www.cs.ru.nl/B.Jacobs/CLG/JacobsCoalgebraIntro.pdf.

[17] A. Joyal, M. Nielsen, and G. Winskel. *Bisimulation and open maps.* In Logic in Computer Science, LICS'93. IEEE Computer Society, 1993.

[18] R.Milner. A Calculus of Communicating Systems. LNCS, vol. 92. Springer, 1980.

[19] M.Nielsen, and G.Winskel. *Petri Nets and Bisimulation.* Theor. Comput. Sci. 153(1&2):211-244, 1996.

[20] D.Park. *Concurrency and automata on infinite sequences.* In *5th GI-Conference on Theoretical Computer Science*, pages 167–183. Springer, 1981.

[21] G.D. Plotkin. *A structural approach to operational semantics.* TR DAIMI FN-19, Computer Science Dept., Aarhus Univ., 1981.

[22] J. Rothe, and D. Mašulović. *A syntactical approach to weak (bi)-simulation for coalgebras.* In Coalgebraic Methods in Computer Science, CMCS'02. ENTCS vol. 65(1). Elsevier, 2002.

[23] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.

[24] D.Sangiorgi. *Bisimulation and Co-induction: Some Problems.* Electr. Notes Theor. Comput. Sci. 162:311-315. Elsevier, 2006.

[25] A. Sokolova. *On compositions and paths for coalgebras.* Technical report CSR-05-26, TU Eindhoven, 2005.

[26] A. Sokolova, E.P. de Vink, and H. Woracek. *Weak Bisimulation for Action-Type Coalgebras.* In Category Theory and Computer Science, CTCS'04. ENTCS 112:211-228. Elsevier, 2005.

[27] R. van Glabbeek. The linear time - branching time spectrum I; the semantics of concrete, sequential processes. In Handbook of Process Algebra Chapter 1, pp. 3-99. Elsevier, 2001.

[28] W.Vogler. *Deciding History Preserving Bisimilarity.* In 18th Int. Colloquium on Automata, Languages and Programming, ICALP'91. LNCS, vol. 510, pp. 495-505. Springer, 1991.

[29] W.Vogler. *Bisimulation and Action Refinement. Theor. Comput. Sci.*, 114(1):173–200, 1993.

[30] G.Lüttgen and W.Vogler. Bisimulation on speed: A unified approach. *Theor. Comput. Sci.*, 360(1-3):209–227, 2006.