Checking Bisimilarity for Attributed Graph Transformation

Fernando Orejas[1,⋆], Artur Boronat[1,2,⋆⋆], Ulrike Golas[3], and Nikos Mylonakis[1]

[1] Universitat Politècnica de Catalunya, Spain
{orejas,nicos}@lsi.upc.edu
[2] University of Leicester, UK
aboronat@mcs.le.ac.uk
[3] Konrad-Zuse-Zentrum für Informationstechnik Berlin, Germany
golas@zib.de

Abstract. Borrowed context graph transformation is a technique developed by Ehrig and Koenig to define bisimilarity congruences from reduction semantics defined by graph transformation. This means that, for instance, this technique can be used for defining bisimilarity congruences for process calculi whose operational semantics can be defined by graph transformation. Moreover, given a set of graph transformation rules, the technique can be used for checking bisimilarity of two given graphs. Unfortunately, we can not use this ideas to check if attributed graphs are bisimilar, i.e. graphs whose nodes or edges are labelled with values from some given data algebra and where graph transformation involves computation on that algebra. The problem is that, in the case of attributed graphs, borrowed context transformation may be infinitely branching. In this paper, based on borrowed context transformation of what we call symbolic graphs, we present a sound and relatively complete inference system for checking bisimilarity of attributed graphs. In particular, this means that, if using our inference system we are able to prove that two graphs are bisimilar then they are indeed bisimilar. Conversely, two graphs are not bisimilar if and only if we can find a proof saying so, provided that we are able to prove some formulas over the given data algebra. Moreover, since the proof system is complex to use, we also present a tableau method based on the inference system that is also sound and relatively complete.

Keywords: Attributed graph transformation, symbolic graph transformation, borrowed contexts, bisimilarity.

1 Introduction

Bisimilarity [18] is a core concept in Computer Science and, thus, it has been studied in very different contexts, especially in the framework of process calculi. However, the case where processes include data has received relatively little attention. We think that there are two main reasons for this. On the one hand, abstracting from data allows us to

⋆ This work has been partially supported by the CICYT project (ref. TIN2007-66523) and by the AGAUR grant to the research group ALBCOM (ref. 00516).
⋆⋆ Supported by a Study Leave from University of Leicester.

F. Pfenning (Ed.): FOSSACS 2013, LNCS 7794, pp. 113–128, 2013.

concentrate better on the study of communication and interaction. On the other hand, in general, bisimilarity is already undecidable. Hence, adding values and computation will not only add another source of undecidability, but also of incompleteness, if the data domain is rich enough.

Borrowed context (BC) graph transformation [6] is a technique developed by Ehrig and Koenig to define bisimilarity congruences from reduction semantics defined by graph transformation. This means that, for instance, this technique can be used for defining bisimilarity congruences for process calculi whose operational semantics can be defined by graph transformation (as e.g. CCS [1], the π-calculus [7], or the ambient calculus [2]). As usual in the area of graph transformation [4], the results in [6] apply to all kinds of graphs that form a category that is M-adhesive [13,5], i.e. most classes of graphical structures. In [6] they also show how this technique can be used for checking bisimilarity of two given graphs. Unfortunately, even if attributed graphs (i.e. graphs whose nodes or edges are labelled with values from some given data algebra and where graph transformation involves computation on that algebra) are an M-adhesive category, their techniques can not be used for checking bisimilarity of this kind of graphs, because BC transformation may be infinitely branching.

In this paper, using BC transformation, but applied to a class of *symbolic graphs*, we present an inference system for checking bisimilarity of attributed graphs. The key issue is that, using symbolic graphs, we can decouple the proof of properties about the graph structure of the given graphs from the proof of properties of data and computations, in a similar way that constraint logic programming [12] decouples computation or constraint solving from deduction. The paper builds on [15], where we showed that bisimilarity of attributed graphs is in a way equivalent to a relation, which we call s-bisimilarity, of symbolic graphs. However, in [15] it was unclear how we could use those results to check bisimilarity, since the notion of s-bisimilarity is somewhat involved.

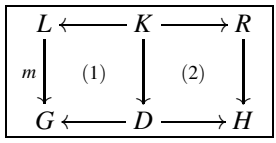
Our inference system is shown to be sound and refutationally complete. This means that, if using our inference system we are able to prove that two graphs are bisimilar, then they are indeed bisimilar. Conversely, two graphs are not bisimilar if and only if we can find a proof saying so, provided that we are able to prove some formulas over the given data algebra. In this sense, it could be better said that our inference system is relatively complete. In addition, since it may be not obvious how to use this inference system, we also present a related tableau method that is also sound and complete.

The paper is organized as follows. In Sections 2 and 3, we introduce borrowed context transformation and attributed and symbolic graphs. In Section 4, we recall the main results from [15]. Sections 5 and 6 are devoted to present the inference system and the tableau method. Finally, in Section 7 we review related work and draw some conclusions.

2 Graph Transformation with Borrowed Contexts

Graph transformation is a powerful approach to describe local computations on systems whose states can be described by graphs. In our context, transformations are specified by rules $p : L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R$, which are spans of graph inclusions (or, in general, of some kind of monomorphisms).

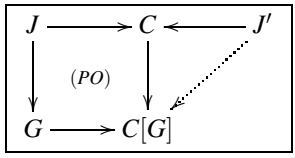
A rule p can be applied to a graph G if there is a *match* monomorphism $m : L \to G$ such that pushout (1) on the right exists. The result is the transformation $G \Longrightarrow_{p,m} H$ (or just $G \Longrightarrow H$ if p and m are implicit), where H is defined by the diagram on the right and (2) is also a pushout.



Intuitively, the *pushout complement* D is obtained by deleting from G the images through m of all the elements (nodes and edges) in L which are not in K, and H is obtained by adding to D all the elements in R that are not in K.

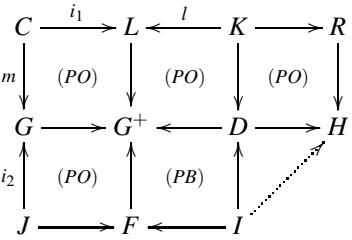
Graph transformation with borrowed contexts [6] is a technique that allows us to study the behavior of systems described by graph transformation. In particular, it allows us to analyze how a graph can evolve when embedded in different contexts for a given set of transformation rules.

The first idea behind this technique is that we have to specify explicitly what is the *open* (or visible) part of the given graph G, i.e. what part of G can be extended by a context. This is called the *interface* of the graph and it may be any arbitrary subgraph of G. This means that a graph with interface is an inclusion or, in general, a monomorphism $J \to G$. Then, a context should be a graph with two interfaces $J \to C \leftarrow J'$, so that, when we embed $J \to G$ in the context $J \to C \leftarrow J'$, the result is a graph $J' \to C[G]$, where $C[G]$ is obtained gluing G and C by a pushout, as shown on the diagram on the right.



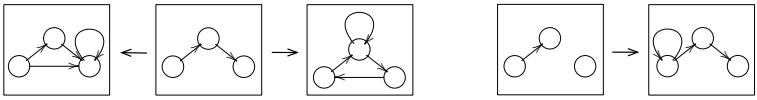
Then, we can model the behavior of a graph G by extending it with minimal contexts allowing the application of the given rules. This means that, to apply a rule $p : L \leftarrow K \to R$, we look for a *partial match* of L in G and add to G the missing part of L, so that we can apply a standard transformation via p. As this context is the part of L that has not been matched with G, we say that G *borrows this context* from the rule. We consider these transformations as transitions labelled by the context borrowed. However, some BC transformations are not useful for studying the behavior of a graph. This is the case of *independent* transformations, where the partial match is included in the part of the interface that remains invariant after the transformation [6].

Definition 1. *Given a graph with interface $J \to G$ and a graph transformation rule $p : L \leftarrow K \to R$, we say that there is a transition from $J \to G$ to $I \to H$ with label $J \to F \leftarrow I$, denoted $(J \to G) \xrightarrow{J \to F \leftarrow I}_{p,m} (I \to H)$ (or just $(J \to G) \xrightarrow{J \to F \leftarrow I} (I \to H)$, if the partial match m and the rule p can remain implicit) if there are graphs C, G^+, D and additional morphisms such that all the squares in the diagram below are pushouts (PO) or pullbacks (PB) and all the morphisms are injective:*

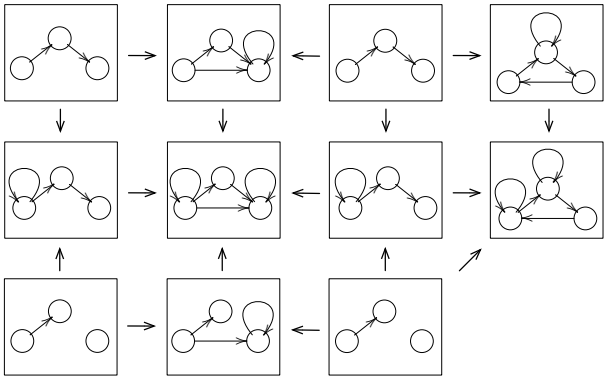


A BC transformation is independent *if there are morphisms $j_1 : C \to K$ and $j_2 : C \to J$ such that $i_1 = l \circ j_1$ and $m = i_2 \circ j_2$.*

The intuition is that C is the subgraph of L that completely matches G; $J \to F \leftarrow I$ is the context borrowed to extend G; G^+ is the graph G enriched with the borrowed context, and H is the result of the transformation. More precisely, F, defined as the pushout complement (if it exists) of the left lower square, extends J with all the elements in G^+ which are not in G. For instance, given the rule below on the left, and the graph with interface $J \to G$ below on the right



the diagram below depicts a BC transformation of $J \to G$ using that rule.



Bisimilarity is the largest symmetric relation between states that is compatible with their observational behaviour. This means that if two states s_1 and s_2 are bisimilar then for every transition from s_1 labelled with ℓ there should be a transition from s_2 with the same label such that the resulting states should again be bisimilar. In our case, states are graphs with interface and transitions are borrowed context transformations.

Definition 2. *Given a set $\mathcal{T}$ of transformation rules, bisimilarity, denoted by $\sim$, is the largest symmetric relation on graphs with interface satisfying that if $(J \to G_1) \sim (J \to G_2)$, for every label $\ell = J \to F \leftarrow I$ and every transition $(J \to G_1) \xrightarrow{\ell} (I \to H_1)$ there exists a transition $(J \to G_2) \xrightarrow{\ell} (I \to H_2)$ such that $(I \to H_1) \sim (I \to H_2)$.*

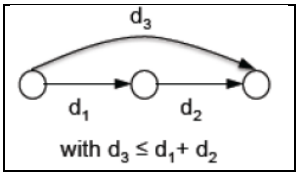
Ehrig and König [6] proved that bisimilarity is a congruence, providing a relatively simple technique for deriving bisimulation congruences out of a (graph transformation) reduction semantics. They also proved some properties that are useful for checking bisimilarity, for instance, that it is possible to use *up to context* techniques [20] or that the condition to show bisimilarity can be restricted to dependent transformations.

3 Attributed Graphs and Symbolic Graphs

There are different approaches in the literature to work with attributed graphs. We consider two of them: attributed graphs as studied in [4] and symbolic graphs [16]. They

are both defined as a special kind of labeled graphs called E-graphs (e.g., see [4]). An attributed graph G, in the sense of [4], consists of two parts: an algebra $\mathcal{A}$, and an E-graph EG, where the labels of EG are the values of $\mathcal{A}$. Similarly, an attributed graph morphism $h : \langle EG, \mathcal{A} \rangle \to \langle EG', \mathcal{A}' \rangle$ consists of two parts: an algebra homomorphism h_{alg} and an E-graph morphism h_{gr}, such that they are compatible, meaning that, for every value v in $\mathcal{A}$, $h_{alg}(v) = h_{gr}(v)$. Attributed graphs and morphisms form the category **AttGraphs**, which is M-adhesive [4].

Attributed graph transformation rules are usually defined as spans $p : L \leftarrow K \to R$, where L, K and R are attributed graphs over a term algebra $T_\Sigma(X)$. A match morphism $m : L \to G$, where G is an attributed graph over a Σ-algebra $\mathcal{A}$ must bind each term t in $T_\Sigma(X)$ (and, in particular, each variable in X) to some element in $\mathcal{A}$. The fact that m_{alg} must be a homomorphism ensures that $m(t)$ must be the result of the evaluation of t, after replacing every variable x in t by $m(x)$.



We also work with symbolic graphs because we use them as a tool for checking bisimilarity of attributed graphs. Intuitively, a symbolic graph may be seen as a graph that specifies a class of attributed graphs sharing the same data algebra. In particular, a symbolic graph SG over the algebra $\mathcal{A}$ is an E-graph G, whose labels are variables from a given set X, together with a *condition* (i.e., a first-order formula) Φ over these variables and over the elements in $\mathcal{A}$. For instance, the graph on the right specifies a class of attributed graphs, including distances in the edges, that satisfy the well-known triangle inequality. The intuition is that each substitution $\sigma : X \to \mathcal{A}$, such that $\mathcal{A} \models \sigma(\Phi)$, defines an attributed graph in the semantics of SG, obtained replacing each variable x in G by the corresponding data value $\sigma(x)$. Formally, the semantics of SG is defined:

$$Sem(SG) = \{\langle \sigma(G), \mathcal{A} \rangle \mid \mathcal{A} \models \sigma(\Phi)\}$$

To enhance readability, we refer to the attributed graphs in the semantics of SG just as $\sigma(SG)$, leaving the algebra $\mathcal{A}$ implicit. Moreover, for (technical) simplicity, we assume that in our symbolic graphs no variable is bound to two different elements of the graph. It should be clear that this is not a limitation since it is enough to replace each repeated occurrence of a variable x by a fresh variable y, and to include the equality $x = y$ in the associated formula.

Every attributed graph may be seen as a symbolic graph by just replacing all its values by variables, and by including, for each value v in the graph, an equation $x_v = v$, in the corresponding condition Φ, where x_v is the variable that has replaced the value v. We call this kind of symbolic graphs *grounded symbolic graphs*. In particular, $GSG(G)$ denotes the grounded symbolic graph defined by G.

A morphism $h : \langle G_1, \Phi_1 \rangle \to \langle G_2, \Phi_2 \rangle$ is a graph morphism $h : G_1 \to G_2$ such that $\mathcal{A} \models \Phi_2 \Rightarrow h(\Phi_1)$, where $h(\Phi_1)$ is the formula obtained when replacing in Φ_1 every variable x_1 in the set of labels of G_1 by $h(x_1)$. Symbolic graphs and morphisms over a given data algebra $\mathcal{A}$ form the category $\mathbf{SymbGraph}_{\mathcal{A}}$, which is M-adhesive [16].

In this paper, a *symbolic graph transformation rule* is a pair $\langle L \hookleftarrow K \hookrightarrow R, \Phi \rangle$, where L, K and R are graphs over a set of variables X and Φ is a condition over X and over

the elements in $\mathcal{A}$. We consider that a rule is a span of symbolic graph inclusions $\langle L, \mathbf{true}\rangle \hookleftarrow \langle K, \mathbf{true}\rangle \hookrightarrow \langle R, \Phi\rangle$. Intuitively, Φ defines applicability conditions and relates the attributes in the left and right-hand side of the rule. As usual, we can define the application of a graph transformation rule $\langle L \hookleftarrow K \hookrightarrow R, \Phi\rangle$ by a double pushout in the category of symbolic graphs [17].

Definition 3. *Given a transformation rule $p = \langle L \hookleftarrow K \hookrightarrow R, \Phi\rangle$ over a data algebra $\mathcal{A}$ and a morphism $m : L \to G$, $\langle G, \Phi'\rangle \Longrightarrow_{r,m} \langle H, \Phi' \wedge m'(\Phi)\rangle$ if (1) and (2) are pushouts and $\Phi' \wedge m'(\Phi)$ is satisfiable in $\mathcal{A}$.*

$$\begin{array}{ccccc} L & \hookleftarrow & K & \hookrightarrow & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow & (2) & \downarrow{\scriptstyle m'} \\ G & \hookleftarrow & D & \hookrightarrow & H \end{array}$$

If $\Phi' \wedge m'(\Phi)$ are unsatisfiable, the resulting graph $\langle H, \Phi' \wedge m'(\Phi)\rangle$ has an empty semantics. This is avoided by requiring $\Phi' \wedge m'(\Phi)$ satisfiable. The above construction defines a double pushout in $\mathbf{SymbGraph}_{\mathcal{A}}$ [17].

A symbolic graph transformation rule can be seen as a specification of a class of attributed graph transformation rules. More precisely, we may consider that the rule $p = \langle L \hookleftarrow K \hookrightarrow R, \Phi\rangle$ denotes the class of all rules $\sigma(L) \hookleftarrow \sigma(K) \hookrightarrow \sigma(R)$, where σ is a substitution such that $\mathcal{A} \models \sigma(\Phi)$, i.e.:

$$Sem(p) = \{\sigma(L) \hookleftarrow \sigma(K) \hookrightarrow \sigma(R) \mid \mathcal{A} \models \sigma(\Phi)\}$$

It is not difficult to see [16] that given a rule p and a symbolic graph SG, $SG \Longrightarrow_p SG'$ if for every graph $G \in Sem(SG)$, $G \Longrightarrow_{p'} G'$, with $G' \in Sem(SG')$ and $p' \in Sem(p)$. Vice versa for every $G' \in Sem(SG')$, there is a graph $G \in Sem(SG)$ and a rule $p' \in Sem(p)$ such that $G \Longrightarrow_{p'} G'$.

4 Bisimilarity of Attributed Graphs and S-bisimilarity

Checking bisimilarity of attributed graphs, using directly the notions presented in Section 2, faces a main problem: given an attributed graph with interface $J \to G$ and a finite set of transformation rules, there may exist an infinite number of different transitions $(J \to G) \xrightarrow{\ell} (I \to H)$. For instance, in the example in Section 6, the borrowed context application of any of the given rules to any of the given graphs would require the assignment of a value to the variable x. Hence we would have an infinite number of possible matches, each of them corresponding to each different value.

We may think that we may avoid this infinite branching by using symbolic graph transformation, where we are not forced to substitute every variable in the interface. So that for deciding if two attributed graphs are bisimilar we could check if their associated grounded graphs are bisimilar in the category of symbolic graphs. Unfortunately, in [15] we proved that two attributed graphs may be bisimilar as attributed graphs, while their associated grounded symbolic graphs are not bisimilar as symbolic graphs.

However, in [15] we also proved that the following notion of S-bisimilarity over symbolic graphs could be used for proving bisimilarity of attributed graphs.

Definition 4. *S-bisimilarity, $\sim_S$, is the largest symmetric relation on symbolic graphs with interface satisfying that if $(J \to SG_1) \sim_S (J \to SG_2)$ then for every dependent transition $(J \to SG_1) \xrightarrow{\ell} (I \to SG_1')$, with $SG_1' = \langle G_1', \Phi_1' \rangle$ there exists a family of conditions $\{\Psi_i\}_{i \in \mathcal{I}}$ and a family of transitions $\{(J \to SG_2) \xrightarrow{\ell} (I \to SH_i)\}_{i \in \mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i \rangle$ such that:*

1. *For every substitution σ_1' such that $\mathcal{A} \models \sigma_1'(\Phi_1')$, there is an index i and a substitution σ_i such that $\mathcal{A} \models \sigma_i(\Psi_i \wedge \Pi_i)$ and $\sigma_1'|_I = \sigma_i|_I$, where $\sigma|_I$ denotes the restriction of σ to the variables in I.*
2. *For every i, $(I \to \langle G_1', \Phi_1' \wedge \Psi_i \rangle) \sim_S (I \to \langle H_i, \Pi_i \wedge \Psi_i \rangle)$.*

Moreover, given a label ℓ, we write $(J \to SG_1) \sim_S^{\ell} (J \to SG_2)$ if for every dependent transition $(J \to SG_1) \xrightarrow{\ell} (I \to SG_1')$ there exists a family of conditions $\{\Psi_i\}_{i \in \mathcal{I}}$ and a family of transitions $\{(J \to SG_2) \xrightarrow{\ell} (I \to SH_i)\}_{i \in \mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i \rangle$ such that conditions 1 *and* 2 *above hold.*

The definition of S-bisimilarity is easy to understand if we think that every symbolic transition $tr = (J \to SG_1) \xrightarrow{\ell} (I \to SG_1')$ denotes a family of attributed transitions. In particular, every substitution σ_1' of the variables in SG_1' such that $\mathcal{A} \models \sigma(\Phi_1')$ denotes an attributed transition $\sigma_1'(tr) = \sigma_1'(J \to G_1) \xrightarrow{\sigma_1'(\ell)} \sigma_1'(I \to SG_1')$. Then, each condition Ψ_i should characterize which attributed transitions denoted by tr are simulated by an attributed transition denoted by $tr_i' = (J \to SG_2) \xrightarrow{\ell} (I \to SH_i)$. In this context, conditions 1 and 2 just state that each $\sigma_1'(tr)$ must be simulated by some attributed transition denoted by tr_i', for some i. Then, as said above, we have:

Theorem 1. *[15] Given transformation rules $\mathcal{T}$, $(J \to G_1) \sim (J \to G_2)$ with respect to $Sem(\mathcal{T})$ if and only if $GSG(J \to G_1) \sim_S GSG(J \to G_2)$ with respect to $\mathcal{T}$.*

In [15] we also proved that S-bisimilarity is a congruence and that up-to-context techniques can also be applied in this setting.

5 An Inference System for Proving Bisimilarity

The results in [15], and in particular Theorem 1, provide a convenient characterization of the bisimilarity relation for attributed graphs that avoids the infinite branching problem associated to the direct application of the results in [6]. However, it is not obvious how this characterization can be actually used for checking bisimilarity. In particular, the main problem is to find the conditions Ψ_i that are needed, according to Def. 4, to play the bisimulation game. Below, we present seven inference rules that describe implicitly how we can compute these conditions.

The judgements that we use in our rules are constrained sequents of the form $\Gamma \vdash (J \to SG_1) \sim_S (J \to SG_2)[\Psi^+, \Psi^-]$ or $\Gamma \vdash (J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\Psi^+, \Psi^-]$, where:

- The antecedent Γ is the context, i.e. a set of facts $(I \to SG) \sim_S (I \to SG')$ that we assume to hold. Contexts are used for *up-to* inference steps.

- The only common variables of SG_1 and SG_2 are the variables in J.
- The succedent $(J \to SG_1)\mathcal{R}(J \to SG_2)[\Psi^+, \Psi^-]$, where $\mathcal{R}$ is either $\sim_S$ or $\sim_S^\ell$ and where Ψ^+ and Ψ^- are formulas including the variables in SG_1 and SG_2, is a statement whose intended meaning is:
  - Ψ^+ is a formula where all its variables not in SG_1 or in SG_2 are (implicitly) quantified universally, such that if it holds then $(J \to SG_1 \wedge \Psi^+)\mathcal{R}(J \to SG_2 \wedge \Psi^+)$ must hold.
  - If Ψ^- is satisfiable then $(J \to SG_1)\mathcal{R}(J \to SG_2)$ does not hold.

  where, if $SG = \langle G, \Phi \rangle$, $SG \wedge \Psi$ denotes the symbolic graph $\langle G, \Phi \wedge \Psi \rangle$.

As a consequence, if we want to check if two attributed graphs, $J \to G$ and $J \to G'$ are bisimilar, and if Φ and Φ' are the conditions of $GSG(G)$ and $GSG(G')$, respectively, we will try to infer judgements of the form $\emptyset \vdash GSG(J \to G) \sim_S GSG(J \to G')[\Psi^+, \Psi^-]$, where $\emptyset$ is the empty context. If Φ and Φ' imply Ψ^+ then we would conclude that $J \to G$ and $J \to G'$ are bisimilar. The reason is that if Φ and Φ' imply Ψ^+, then $GSG(J \to G) = (GSG(J) \to GSG(G) \wedge \Psi^+) \sim_S (GSG(J) \to GSG(G') \wedge \Psi^+) = GSG(J \to G')$ and, by Thm. 1, $(J \to G) \sim (J \to G')$. However, if Ψ^- is satisfiable, also by Thm. 1, we would conclude that $J \to G$ and $J \to G'$ are not bisimilar.

The first rule is just a consequence of how the relation $\sim_S^\ell$ is defined. In particular the rule says that if for each label ℓ, $(J \to SG_1) \sim_S^\ell (J \to SG_2)$ under the condition Ψ_ℓ^+, then $(J \to SG_1) \sim_S (J \to SG_2)$ under the conjunction of all the Ψ_ℓ^+. Conversely, if for each label ℓ, $(J \to SG_1) \nsim_S^\ell (J \to SG_2)$ under the condition Ψ_ℓ^-, then $(J \to SG_1) \nsim_S (J \to SG_2)$ under the disjunction of all the Ψ_ℓ^-.

1. Labels

$$\frac{\Gamma \vdash (J \to SG_1) \sim_S^{\ell_1} (J \to SG_2)[\Psi_{\ell_1}^+, \Psi_{\ell_1}^-] \quad \cdots \quad \Gamma \vdash (J \to SG_1) \sim_S^{\ell_n} (J \to SG_2)[\Psi_{\ell_n}^+, \Psi_{\ell_n}^-]}{\Gamma \vdash (J \to SG_1) \sim_S (J \to SG_2)[\bigwedge_{i=1}^{n} \Psi_{\ell_i}^+, \bigvee_{i=1}^{n} \Psi_{\ell_i}^-]}$$

If $\{\ell_1, \ldots, \ell_n\}$ is the set of all labels ℓ such that there is a dependent transformation $(J \to SG_1) \xrightarrow{\ell} (I \to SG_1')$ or $(J \to SG_2) \xrightarrow{\ell} (I \to SG_2')$.

If two graphs are equal then they are obviously bisimilar. However, if their underlying E-graphs are equal, but their conditions are different, the rule below tells us that the two graphs are bisimilar under the conjunction of their associated conditions.

2. Equality

$$\Gamma \vdash (J \to \langle G, \Phi \rangle) \sim_S (J \to \langle G, \Phi' \rangle)[\Phi \wedge \Phi', \mathbf{false}]$$

A trivial rule that is needed for technical reasons in the completeness proof:

3. Trivial

$$\Gamma \vdash (I \to SG) \sim_S^\ell (I \to SG')[\mathbf{false}, \mathbf{false}]$$

The fourth rule is also quite simple. Let us assume that $Cond(SG,\ell)$ is the condition that covers all possible transitions of SG with label ℓ, i.e.

$$Cond(SG,\ell) = \bigvee_{p,m} \Phi_{p,m},$$

such that $(J \to SG) \xrightarrow{\ell}_{p,m} (I \to \langle G', \Phi_{p,m}\rangle)$. Then, if $\neg Cond(SG,\ell)$ holds, no transition of SG with label ℓ is possible. Therefore, if $\neg Cond(SG_1,\ell) \wedge \neg Cond(SG_2,\ell)$ holds no transition with label ℓ is possible of neither SG_1 nor SG_2. Thus, under that condition they are ℓ-bisimilar. Conversely, when $(Cond(SG_1,\ell) \setminus Cond(SG_2,\ell)) \vee (Cond(SG_2,\ell) \setminus Cond(SG_1,\ell))$ holds, either there is a transition with label ℓ from SG_1, but not from SG_2, or vice versa, meaning that they are not ℓ-bisimilar.

4. Complement

$$\Gamma \vdash (J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\Psi^+, \Psi^-]$$

where

$$\begin{aligned}\Psi^+ =& \neg Cond(SG_1,\ell) \wedge \neg Cond(SG_2,\ell)\\ \Psi^- =& (Cond(SG_1,\ell) \setminus Cond(SG_2,\ell)) \vee\\ & (Cond(SG_2,\ell) \setminus Cond(SG_1,\ell))\end{aligned}$$

The next rule states that if $(J \to SG_1)$ and $(J \to SG_2)$ are bisimilar when Ψ_1^+ holds and, also, when Ψ_2^+ holds, then they are bisimilar when either of them hold. Conversely, if $(J \to SG_1)$ and $(J \to SG_2)$ are not bisimilar when Ψ_1^- is satisfiable and also when Ψ_2^- is satisfiable, then if any of them are satisfiable the two graphs are not bisimilar.

5. Disjunction

$$\frac{\Gamma \vdash (J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\Psi_1^+, \Psi_1^-] \quad \Gamma \vdash (J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\Psi_2^+, \Psi_2^-]}{\Gamma \vdash (J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\Psi_1^+ \vee \Psi_2^+, \Psi_1^- \vee \Psi_2^-]}$$

The following rule is a bit more involved. It essentially follows from the definition of $\sim_S^{\ell}$. If $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$, the disjunction of all the conditions associated with the transformations $(J \to SG') \xrightarrow{\ell}_{(p',m')} (I \to SH'_{(p',m')})$ that are bisimilar to $(I \to SH_{(p,m)})$ should cover $\Phi_{(p,m)}$. But, in general, we cannot ensure this. We can only ensure that, under the condition $\Psi^+_{(p,m)} = \bigvee_{(p',m')} \Psi^+_{(p,m),(p',m')}$, the attributed transitions denoted by $\xrightarrow{\ell}_{(p,m)}$ are simulated by transitions denoted by $\xrightarrow{\ell}_{(p',m')}$. This means that under the condition $\Phi_{(p,m)} \wedge \Psi^+_{(p,m)}$ the transition $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$ is simulated by transitions from $(J \to SG')$. On the other hand, it may happen that on the condition $\Phi_{(p,m)} \setminus \Psi^+_{(p,m)}$ the transition $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$ is not simulated by any transition from $(J \to SG')$. Hence, if $\Phi_{(p,m)} \setminus \Psi^+_{(p,m)}$ holds, we cannot ensure that $(J \to SG) \sim_S^{\ell} (J \to SG')$. Since this is true for each (p,m), all ℓ- transitions from $(J \to SG)$ are simulated by ℓ'- transitions from $(J \to SG')$ when any of the conditions $\Phi_{(p,m)} \wedge \Psi^+_{(p,m)}$ holds, unless any of the conditions $\Phi_{(p,m)} \setminus \Psi^+_{(p,m)}$ holds, and vice versa

for the ℓ- transitions from $(J \to SG')$. Altogether, this means that we can ensure that $(J \to SG) \sim_S^{\ell} (J \to SG')$ on the condition Ψ^+ as defined in the rule.

Conversely, if $\Psi^-_{(p,m),(p',m')}$ is satisfied then $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$ is not simulated by $(J \to SG') \xrightarrow{\ell}_{(p',m')} (I \to SH'_{(p',m')})$. So, if the conjunction of conditions $\Psi^-_{(p,m)} = \bigwedge_{(p',m')} \Psi^-_{(p,m),(p',m')}$ is satisfied then $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$ is not simulated by any ℓ-transition from $(J \to SG')$. But this means that if any of the conditions $\Psi^-_{(p,m)}$ is satisfied then no transition from $(J \to SG)$ can be simulated, and something similar happens with respect to $(J \to SG')$. In short, this means that we can ensure that if Ψ^-, as defined in the rule, is satisfied then $(J \to SG) \nsim_S^{\ell} (J \to SG')$.

Finally, the rule also states that, when proving $(I \to SH_{(p,m)}) \sim_S (I \to SH'_{(p',m')})$ we may assume that $(J \to SG) \sim_S (J \to SG')$ already holds, so that we can use up-to-context techniques that have been shown valid for S-bisimilarity [15].

6. Bisimulation

$$\frac{\Gamma \cup \{(J \to SG) \sim_S (J \to SG')\} \vdash \bigwedge_{(p,m),(p',m')} (I \to SH_{(p,m)}) \sim_S (I \to SH'_{(p',m')})[\Psi^+_{(p,m),(p',m')}, \Psi^-_{(p,m),(p',m')}]}{\Gamma \vdash (J \to SG) \sim_S^{\ell} (J \to SG')[\Psi^+, \Psi^-]}$$

For all rules p, p' and partial matches m, m' such that $(J \to SG) \xrightarrow{\ell}_{(p,m)} (I \to SH_{(p,m)})$ and $(J \to SG') \xrightarrow{\ell}_{(p',m')} (I \to SH'_{(p',m')})$, and where, if $SH_{(p,m)} = \langle H_{(p,m)}, \Phi_{(p,m)} \rangle$ and $SH'_{(p',m')} = \langle H'_{(p',m')}, \Phi'_{(p',m')} \rangle$, then Ψ^+, Ψ^- are defined:

$$\Psi^+ = \Big(\bigvee (\Phi_{(p,m)} \wedge \Psi^+_{(p,m)}) \setminus \bigvee (\Phi_{(p,m)} \setminus \Psi^+_{(p,m)}) \Big) \wedge \Big(\bigvee (\Phi_{(p',m')} \wedge \Psi^+_{(p',m')}) \setminus \bigvee (\Phi_{(p',m')} \setminus \Psi^+_{(p',m')}) \Big)$$

$$\Psi^- = \big(\bigvee (\Psi^-_{(p,m)} \wedge \Phi_{(p,m)}) \vee \bigvee (\Psi^-_{(p',m')} \wedge \Phi_{(p',m')}) \big)$$

and where

$$\Psi^+_{(p,m)} = \bigvee_{(p',m')} \Psi^+_{(p,m),(p',m')} \qquad \Psi^+_{(p',m')} = \bigvee_{(p,m)} \Psi^+_{(p,m),(p',m')}$$

$$\Psi^-_{(p,m)} = \bigwedge_{(p',m')} \Psi^-_{(p,m),(p',m')} \qquad \Psi^-_{(p',m')} = \bigwedge_{(p,m)} \Psi^-_{(p,m),(p',m')}$$

The last rule is based on the result from [15] that shows that the up to context technique is sound for proving S-bisimilarity. This means that, when trying to prove $(J \to SG) \sim_S (J \to SG')$, we may assume that for all contexts $J \to F \leftarrow I$: $(I \to F[SG_1]) \sim_S (I \to F[SG_2])$. That is that if $(J \to SG) \sim_S (J \to SG')$ is part of the context, then we could infer $(I \to F[SG_1]) \sim_S (I \to F[SG_2])[\mathbf{true}, \mathbf{false}]$. But this can be generalized to the case where the judgement to infer does not exactly include $F[SG_1]$ and $F[SG_2]$, but $(F[SG_1] \wedge \Phi)$ and $(F[SG_2] \wedge \Phi')$ as the rule shows:

7. Up-to-context

$$\Gamma \cup \{(J \to SG) \sim_S (J \to SG')\} \vdash (I \to SH) \sim_S (I \to SH')[\neg\Phi_1 \wedge \neg\Phi'_1, \mathbf{false}]$$

where, $SH = \langle H, \Phi \vee \Phi_1 \rangle$ and $SH' = \langle H', \Phi' \vee \Phi'_1 \rangle$, and $\langle H, \Phi \rangle$ and $\langle H', \Phi' \rangle$ are the result of embedding SG and SG', respectively, in a context $J \to F \leftarrow I$.

We can prove that the above rules are sound and complete. More precisely:

Theorem 2 (Soundness of the inference rules). *Given attributed graphs $J \to G_1$ and $J \to G_2$, then:*

- *If we can infer $\emptyset \vdash (J \to GSG(G_1)) \sim_S (J \to GSG(G_2))[\Psi^+, \Psi^-]$, and $\Phi_{GSG(G_1)} \wedge \Phi_{GSG(G_2)}$ implies Ψ^+ in $\mathcal{A}$ then $J \to G_1 \sim J \to G_2$.*
- *If we can infer $\emptyset \vdash (J \to GSG(G_1)) \sim_S (J \to GSG(G_2))[\Psi^+, \Psi^-]$ and Ψ^- is satisfiable in $\mathcal{A}$ then $J \to G_1 \nsim J \to G_2$.*

The proof essentially follows the intuitions of the rules that are given above.

Theorem 3 (Completeness of the inference rules). *Given attributed graphs $J \to G_1$ and $J \to G_2$, if $(J \to G_1) \nsim (J \to G_2)$ then, using the above rules, we can infer $\emptyset \vdash (J \to GSG(G_1)) \sim_S (J \to GSG(G_2))[\Psi^+, \Psi^-]$, where $\emptyset$ is the empty context and Ψ^- is a satisfiable condition.*

The proof is done by induction, using the standard definition of stratified bisimilarity [10]. This is sound, since for each $J \to G$ and each ℓ there is a finite number of transitions $(J \to SG) \xrightarrow{\ell} (I \to SH)$.

6 A Tableau Method for Checking Bisimilarity

In the previous section we have presented a set of rules for proving or disproving bisimilarity of attributed graphs. The problem with these rules is that it may not be obvious how to use them to check whether two given graphs $J \to G_1$ and $J \to G_2$ are bisimilar. In this section, we describe a method with this purpose, based on the construction of a kind of constrained tableau [8], i.e. a tableau whose nodes include constraints, following the inference rules from the previous section.

More precisely, our tableaux are trees whose nodes are labelled by formulas $(J \to SG_1) \sim_S (J \to SG_2)$ or $(J \to SG_1) \sim_S^{\ell} (J \to SG_2)$ and by constraints Ψ^+ and Ψ^-, as our judgements in the proof rules. To construct a tableau for $J \to G_1$ and $J \to G_2$, to check if they are bisimilar, we start creating the root, labelling it with $GSG((J \to G_1)) \sim_S GSG((J \to G_2))[\mathbf{false}, \mathbf{false}]$. Then, we start with an iteration where, at each step, we choose a node in the tableau and we apply to it either an expansion step enlarging the tree (just when the node is a leaf), or a constraint computation step changing the constraints of the node. We stop when the tableau is *closed*, i.e. either when $\Phi_{GSG(G_1)}$ and $\Phi_{GSG(G_2)}$ imply Ψ^+ or when Ψ^- is satisfiable in $\mathcal{A}$, where Ψ^+ and Ψ^- are the constraints in the root. In the former case we would conclude that $J \to G_1$ and $J \to G_2$ are bisimilar, and in the latter case we would conclude that they are not.

As said above, the steps for the construction of the tableau can be either *expansion* steps or *constraint computation* steps. There are two kinds of expansion steps:

1. Label Expansion If a leaf n is labelled with the formula $(J \to SG_1) \sim_S (J \to SG_2)$, we create a child of n and we label it with $(J \to SG_1) \sim_S^{\ell} (J \to SG_2)[\mathbf{false}, \mathbf{false}]$, for each ℓ such that there is a dependent transition labelled with ℓ from $(J \to SG_1)$ or from $(J \to SG_2)$.

2. Bisimulation Expansion If a leaf n is labelled with the formula $(J \to SG_1) \sim_S^{\ell} (J \to SG_2)$, for each pair of transitions $(J \to SG_1) \xrightarrow{\ell} (I \to SG_1')$ and $(J \to SG_2) \xrightarrow{\ell} (I \to SG_2')$, we create a child of n and we label it with $(I \to SG_1') \sim_S (I \to SG_2')[\mathbf{false}, \mathbf{false}]$.

There are five kinds of constraint computation steps:

3. Labels Computation If a node n is labelled with $(J \to SG_1) \sim_S (J \to SG_2)[\Pi^+, \Pi^-]$, we can compute new constraints $\Psi^+ = \Pi^+ \vee \bigwedge_{i=1}^n \Psi_i^+$ and $\Psi^- = \Pi^- \vee \bigvee_{i=1}^n \Psi_i^-$, where $\Psi_1^+, \Psi_1^- \ldots, \Psi_n^+, \Psi_n^-$ are the constraints of the descendants of that node.

4. Complement Computation If a node n is labelled with $(J \to SG_1) \sim_S (J \to SG_2)[\Psi_1^+, \Psi_1^-]$ then we can compute new constraints Ψ^+ and Ψ^- for n as follows:

$\Psi^+ = \Psi_1^+ \vee (\neg Cond(SG_1, \ell) \wedge \neg Cond(SG_2, \ell))$

$\Psi^- = \Psi_1^- \vee (Cond(SG_1, \ell) \setminus Cond(SG_2, \ell)) \vee (Cond(SG_2, \ell) \setminus Cond(SG_1, \ell))$

5. Equality Computation If a node n is labelled with $(J \to SG_1) \sim_S (J \to SG_2)[\Psi_1^+, \Psi_1^-]$, then we can compute a new constraint $\Psi^+ = \Psi_1^+ \vee (\Phi_1 \wedge \Phi_1')$ for n, leaving the negative constraint Ψ_1^- unchanged.

6. Bisimulation Computation If a node n is labelled with $(J \to SG_1) \sim_S (J \to SG_2)[\Psi_1^+, \Psi_1^-]$ then we can compute new constraints Ψ^+ and Ψ^- for n as follows:

$$\Psi^+ = \Psi_1^+ \vee \Big(\bigvee (\Phi_{(p,m)} \wedge \Psi_{(p,m)}^+) \setminus \bigvee (\Phi_{(p,m)} \setminus \Psi_{(p,m)}^+) \Big) \wedge \Big(\bigvee (\Phi_{(p',m')} \wedge \Psi_{(p',m')}^+) \setminus \bigvee (\Phi_{(p',m')} \setminus \Psi_{(p',m')}^+) \Big)$$

$$\Psi^- = \Psi_1^- \vee (\bigvee (\Psi_{(p,m)}^- \wedge \Phi_{(p,m)}) \vee \bigvee (\Psi_{(p',m')}^- \wedge \Phi_{(p',m')})$$

where the conditions $\Phi_{(p,m)}$, $\Psi_{(p,m)}^+$, $\Psi_{(p,m)}^-$, $\Phi_{(p',m')}$, $\Psi_{(p',m')}^+$, and $\Psi_{(p',m')}^-$ are as in the Bisimulation inference rule.

7. Up-to-Context Computation If a node n is labelled with $(J \to SG_1) \sim_S (J \to SG_2)[\Psi_1^+, \Psi_1^-]$, if there is an ancestor of n labelled with the formula $(I \to SG_2) \sim_S (I \to SG_2')$, and if there is a context $I \to F \leftarrow J$, where $F[SG_2] = \langle G_1, \Pi_1 \rangle$ and $F[SG_2'] = \langle G_1', \Pi_1' \rangle$ then we can compute a new constraint $\Psi^+ = \Psi_1^+ \vee (\neg(\Phi_1 \setminus \Pi_1) \wedge \neg(\Phi_1' \setminus \Pi_1'))$ for n, leaving unchanged the negative constraint Ψ_1^-.

Then, we have:

Theorem 4 (Soundness). *If we can construct a closed tableau for graphs $J \to G_1$ and $J \to G_2$ whose root is labelled by the constraints Ψ^+ and Ψ^-, then:*

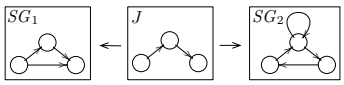
- *If $\Phi_{GSG(G_1)} \wedge \Phi_{GSG(G_2)}$ implies Ψ^+ in $\mathcal{A}$ then $J \to G_1 \sim J \to G_2$.*
- *If Ψ^- is satisfiable in $\mathcal{A}$ then $J \to G_1 \nsim J \to G_2$.*

The proof is a direct consequence of the soundness of the inference rules presented in the previous section.

Theorem 5 (Completeness). *If $(J \to G_1) \nsim (J \to G_2)$, we can construct a closed tableau for $J \to G_1$ and $J \to G_2$ whose negative constraint at the root Ψ^- is satisfiable in $\mathcal{A}$.*

The proof is very similar to the completeness proof of the inference rules.

Let us now see an example of the construction of a tableau. Suppose that we want to check if the graphs $(J \to SG_1)$ and $(J \to SG_2)$ on the right are bisimilar with respect to the rules depicted below (for simplicity, the rules are presented including only the left and right-hand sides, leaving the intermediate part implicit). Part of the tableau that we would use for this proof is shown in Fig. 1. The interfaces of the graphs are not depicted because, in the transformations considered, J (with the obvious inclusions) would be the interface of all the graphs in the tableau.



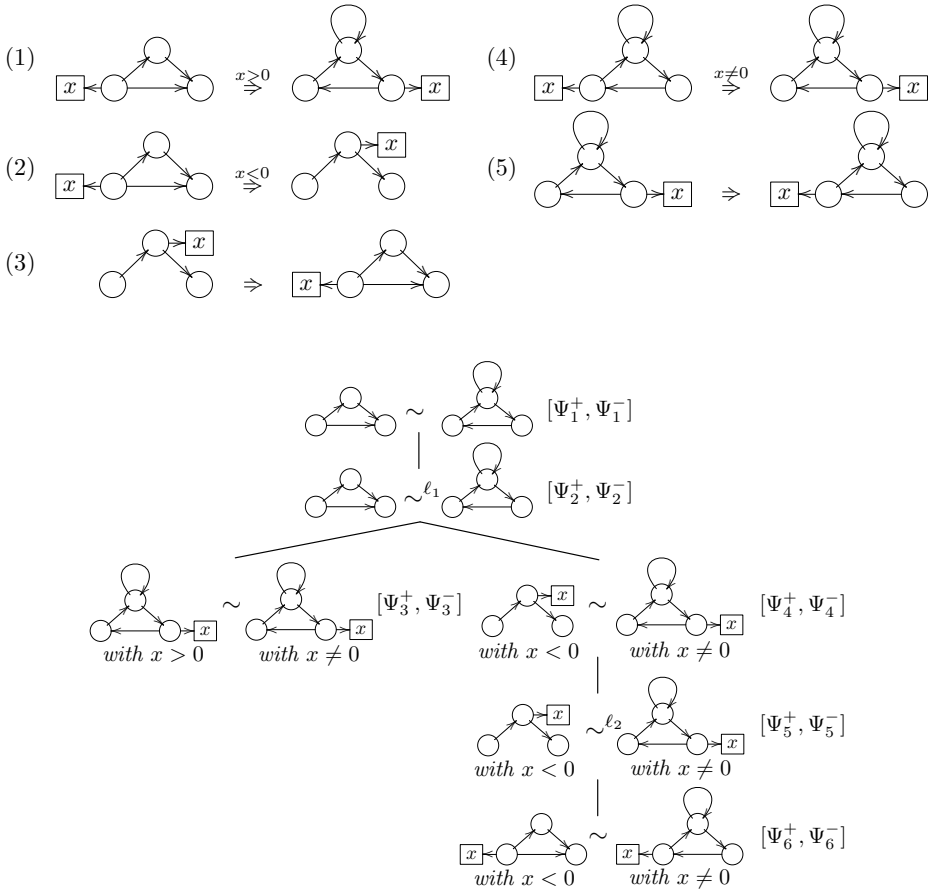
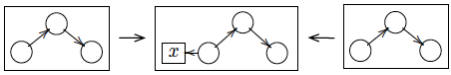


Fig. 1. (Part of a) Tableau

The construction of the tableau starts with the creation of the root and the application of a label expansion step. Due to lack of space, we suppose that we can only transform SG_1 using rules (1) and (2) and SG_2 using rule (4), and using a borrowed context consisting of the square node, together with the attribute x and an edge to the leftmost round node. Actually, there are other transformations with other borrowed contexts that we will not consider. This means that this step would create just one node, corresponding to that borrowed context. We call ℓ_1 this context (and label), which is depicted below.



Then, we proceed with bisimulation expansion corresponding to the BC transformations mentioned above. This step creates two nodes. We can see that the graphs in the node on the left are equal (except for the condition), so we can apply an equality computation step, yielding $[\Psi_3^+, \Psi_3^-] = [x > 0, \mathbf{false}]$. Now, we apply label expansion followed by bisimulation expansion to the node on the right. Again, we consider that the only possible BC transformations of these graphs correspond to the application of rules (3) and (5) without adding any context (i.e. the label would be $J \to J \leftarrow J$). Now, we can apply up to context computation to the bottom right node of the tableau, with respect to the node on the root and the context ℓ_1, yielding $[\Psi_6^+, \Psi_6^-] = [x < 0 \wedge x \neq 0, \mathbf{false}]$. Then, going bottom up, using twice labels and bisimulation computation, we can compute the constraints $[\Psi_5^+, \Psi_5^-] = [x < 0 \wedge x \neq 0, \mathbf{false}]$, $[\Psi_4^+, \Psi_4^-] = [x < 0 \wedge x \neq 0, \mathbf{false}]$, $[\Psi_2^+, \Psi_2^-] = [x > 0 \vee (x < 0 \wedge x \neq 0), \mathbf{false}]$, and $[\Psi_1^+, \Psi_1^-] = [x > 0 \vee (x < 0 \wedge x \neq 0), \mathbf{false}]$. Finally, since we supposed that there are no other BC-transformations of the root, applying complement computation to it, we have $[\Psi_1^+, \Psi_1^-] = [x > 0 \vee (x < 0 \wedge x \neq 0) \vee x = 0, \mathbf{false}]$. To end, since $(x > 0 \vee (x < 0 \wedge x \neq 0) \vee x = 0) \equiv \mathbf{true}$, we would conclude that the two graphs are bisimilar.

7 Related Work and Conclusion

As said in the introduction, bisimilarity has been studied in many different contexts, but the case where processes include data has received relatively little attention. An exception is [9], where the authors define a *symbolic bisimilarity* relation for value-passing CCS and present a proof system that is complete for finite symbolic transition systems. Our approach shares with [9] that we both avoid infinite branching in the associated state-transition diagrams by using (free) variables abstracting from concrete values. In their paper, states are process expressions and labels are guarded actions both including variables. In our case, states are symbolic graphs and labels are contexts. The essential difference comes from the fact that they concentrate on value-passing CCS, which means that labels only depend on the given process expression and not on the possible contexts of that process. Then, given a transition $m \xrightarrow{a} n$, the variables in the action a are assumed to be a subset of the variables in m. However, in our case, given a transition $(J \to SG) \xrightarrow{\ell} (I \to SH)$, we have that ℓ may include free variables which are

not in SG, representing values from the context needed for the transition. For example, in their case, if p is a ground process expression, its state-transition graph would not include any free variable either. This is not the case in our paper. In particular, the conditions Ψ_i in the definition of S-bisimilarity are needed because of the existence of these context variables. Hence, the inclusion of the constraints Ψ^+ and Ψ^- in our proof rules and in our tableau method, which are needed to compute the conditions Ψ_i, are also a consequence of these context variables.

In our framework, name-passing processes, like the processes in the π-calculus [14], can be seen as a special case of value-passing processes[1]. In that context, open bisimilarity [21] could correspond to bisimilarity of attributed graphs, as defined directly in terms of BC transformations on that category, and its symbolic version would be somewhat related to S-bisimilarity.

With respect to BC graph transformation, in [19] an algorithm for checking bisimilarity of graphs is presented, but this algorithm would not be applicable to the case of attributed graphs. Moreover, no correctness proof is included. On the other hand, in [11], the authors extend BC-transformation to the case of conditional transformation systems. Even if their results mainly apply to the case of non-attributed graphs, their notion of context transition is, in a way, related to our symbolic transitions and so they are the corresponding notions of bisimilarity. The reason is that we both deal with conditional rules and transitions defined by borrowed context transformations. This causes that, in the bisimulation game, a transition on one graph with condition A needs not to be simulated by a single transition on the other graph, but by a set of transitions with associated conditions A_i, such that these conditions cover A. The main difference, which is a substantial one, is based on the different nature of conditions. In [11], conditions are related to the structure of the graphs that we are transforming. In our case, conditions refer to properties of the attributes. Actually, both papers are orthogonal. On the one hand, dealing with application conditions would be an interesting extension of our paper. On the other hand, if the graphs considered in [11] were attributed graphs, in general, their associated state-transition diagrams would be infinitely branching, which is the problem that we address in our paper.

Finally, our tableau method for proving bisimilarity can be seen as an extension of the method presented in [3]. However, in that paper, processes do not include data, which means that their tableaux are considerably simpler. In particular they are basic unconstrained tableaux.

In this paper, we have presented a proof system and a related tableau method for checking bisimilarity of attributed graphs, using the notion of S-bisimilarity presented in [15], proving their soundness and refutational completeness. We think that the main advantages of our approach are, first, its generality, since it could be used to check bisimilarity of any kind of formalism whose semantics is expressed in terms of graph transformation; and, second, the way in which our approach decouples the proofs on the graph structure from the proofs on the given data algebra.

[1] Obviously, the π-calculus is not a special case of value-passing CCS. However, if we represented the π-calculus in terms of attributed or symbolic graphs (e.g. similarly to [7]), we would consider names as elements of an algebra whose signature includes only the equality predicate.

References

1. Bonchi, F., Gadducci, F., König, B.: Synthesising CCS bisimulation using graph rewriting. Inf. Comput. 207(1), 14–40 (2009)
2. Bonchi, F., Gadducci, F., Monreale, G.V.: Labelled transitions for mobile ambients (as synthesized via a graphical encoding). Electr. Notes Theor. Comput. Sci. 242(1), 73–98 (2009)
3. Christensen, S., Hirshfeld, Y., Moller, F.: Bisimulation Equivalence is Decidable for Basic Parallel Processes. In: Best, E. (ed.) CONCUR 1993. LNCS, vol. 715, pp. 143–157. Springer, Heidelberg (1993)
4. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. In: EATCS Monographs of Theoretical Comp. Sc. Springer (2006)
5. Ehrig, H., Golas, U., Habel, A., Lambers, L., Orejas, F.: M-adhesive transformation systems with nested application conditions. part 1. Math. Struct. in Com. Sc. (2012) (to appear)
6. Ehrig, H., König, B.: Deriving bisimulation congruences in the DPO approach to graph rewriting with borrowed contexts. Math. Struct. in Com. Sc. 16(6), 1133–1163 (2006)
7. Gadducci, F.: Graph rewriting for the pi-calculus. Math. Struct. in Com. Sc. 17(3), 407–437 (2007)
8. Giese, M., Hähnle, R.: Tableaux + constraints. In: TABLEAUX 2003 position paper (2003)
9. Hennessy, M., Lin, H.: Symbolic bisimulations. Theor. Comput. Sci. 138(2), 353–389 (1995)
10. Hennessy, M., Milner, R.: Algebraic laws for nondeterminism and concurrency. J. ACM 32(1), 137–161 (1985)
11. Hülsbusch, M., König, B.: Deriving Bisimulation Congruences for Conditional Reactive Systems. In: Birkedal, L. (ed.) FOSSACS 2012. LNCS, vol. 7213, pp. 361–375. Springer, Heidelberg (2012)
12. Jaffar, J., Maher, M., Marriot, K., Stuckey, P.: The semantics of constraint logic programs. The Journal of Logic Programming 37, 1–46 (1998)
13. Lack, S., Sobocinski, P.: Adhesive and quasiadhesive categories. Theor. Inf. App. 39, 511–545 (2005)
14. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I and II. Inf. Comput. 100(1), 1–77 (1992)
15. Orejas, F., Boronat, A., Mylonakis, N.: Borrowed Contexts for Attributed Graphs. In: Ehrig, H., Engels, G., Kreowski, H.-J., Rozenberg, G. (eds.) ICGT 2012. LNCS, vol. 7562, pp. 126–140. Springer, Heidelberg (2012)
16. Orejas, F., Lambers, L.: Symbolic attributed graphs for attributed graph transformation. ECEASST 30 (2010)
17. Orejas, F., Lambers, L.: Lazy graph transformation. Fund. Inf. 118, 65–96 (2012)
18. Park, D.: Concurrency and Automata on Infinite Sequences. In: Deussen, P. (ed.) GI-TCS 1981. LNCS, vol. 104, pp. 167–183. Springer, Heidelberg (1981)
19. Rangel, G., König, B., Ehrig, H.: Bisimulation verification for the DPO approach with borrowed contexts. ECEASST 6 (2007)
20. Sangiorgi, D.: On the Proof Method for Bisimulation. In: Hájek, P., Wiedermann, J. (eds.) MFCS 1995. LNCS, vol. 969, pp. 479–488. Springer, Heidelberg (1995)
21. Sangiorgi, D.: A theory of bisimulation for the pi-calculus. Acta Inf. 33(1), 69–97 (1996)