# Borrowed Contexts for Attributed Graphs

Fernando Orejas [⋆1], Artur Boronat [⋆⋆12], and Nikos Mylonakis[1]

[1] Universitat Politècnica de Catalunya, Spain
[2] University of Leicester, UK

**Abstract.** Borrowed context graph transformation is a simple and powerful technique developed by Ehrig and König that allow us to derive labeled transitions and bisimulation congruences for graph transformation systems or, in general, for process calculi that can be defined in terms of graph transformation systems. Moreover, the same authors have also shown how to use this technique for the verification of bisimilarity. In principle, the main results about borrowed context transformation do not apply only to plain graphs, but they are generic in the sense that they apply to all categories that satisfy certain properties related to the notion of adhesivity. In particular, this is the case of attributed graphs. However, as we show in the paper, the techniques used for checking bisimilarity are not equally generic and, in particular they fail, if we want to apply them to attributed graphs. To solve this problem, in this paper, we define a special notion of symbolic graph bisimulation and show how it can be used to check bisimilarity of attributed graphs.

**Key words:** Attributed graph transformation, symbolic graph transformation, borrowed contexts, bisimilarity.

## 1 Introduction

Bisimilarity is possibly the most adequate behavioural equivalence relation. In [5] Ehrig and König they show how a notion of bisimilarity could be defined for graph transformation systems. In particular, they introduced borrowed context graph transformation as a simple and powerful technique that allow us to derive labelled transitions and bisimulation congruences for graph transformation systems or, in general, for process calculi that can be defined in terms of graph transformation systems (e.g. [1]). These results are quite general since they apply to any kind of transformation system on a category that satisfies some properties related to adhesivity [6, 3], as the category of attributed graphs. Moreover, in [11], Rangel, König and Ehrig showed how to use these techniques for the verification of bisimilarity. Unfortunately, the approach to verification introduced informally in [5] and studied in detail in [11] does not work when dealing with attributed graphs. This is especially unfortunate, because one of the motivations for [11] is

⋆ 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

being able to show that model transformations (and, in particular, refactorings) preserve behavioural equivalence, and in these cases we work with attributed graphs. The problem is related with the fact that, when applying an attributed graph transformation rule, we must match all the variables in the left-hand side of the rule to some given values. This implies, in the case of borrowed context transformation, that the partial application of a single rule may give rise to an infinite number of different borrowed context transformation, where all of them have different labels. The reason is that, if the borrowed context includes some variable, each substitution of that variable by any data value defines a different borrowed context transformation.

The nature of the problem immediately suggests using borrowed context symbolic graph transformation to check bisimilarity of attributed graphs, since in symbolic graph transformation variables do not need to be immediately substituted by values [7, 9]. Actually, since symbolic graphs form an adhesive HLR category, all the results in [5] apply to this class of graphs. Unfortunately, as a counter-example in Sect. 4 shows, bisimilarity for symbolic graphs, as defined in [5], does not coincide with bisimilarity for attributed graphs. Hence, in this work we define a new notion of symbolic bisimilarity, which is also a congruence, and we show that it coincides with attributed graph bisimilarity. Moreover, using a variation of the case study presented in [5], we show how this new notion can be used for checking bisimilarity for attributed graphs. In particular, the example not only shows that the problem with the substitution of the variables is solved, but it also shows how symbolic graphs allow us to decouple a bisimilarity proof in two parts. The first one, that has to do with graph structure, is based on borrowed context transformation, while the second one, which is related to data in the graphs, has to do with reasoning in the given data algebra.

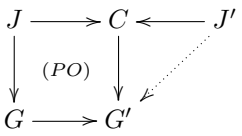
The paper is organized as follows. In Sect. 2 we introduce the main results and constructions presented in [5] and we introduce our case study, describing the problems when attributed graph transformation is considered. In the following section we introduce symbolic graph transformation. Sect. 4 is the core of the paper, where we present our main constructions and results. In Sect 5, we extend some techniques used in [5] for the verification of bisimilarity and we apply them to our case study. Finally, in Sect. 6 we present some related work and draw some conclusions. An Appendix includes the proofs of our results.

## 2 Graph transformation with borrowed contexts

Given a set of transformation rules, graph transformation with borrowed contexts is a technique that allows us to describe and analyze how a graph could evolve when embedded in different contexts. This technique is based on several ideas. The first one 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 part is called the *interface* of the graph and, in general, it may be any arbitrary subgraph of $G$. A consequence of this is that a context should be a graph with two interfaces. The reason is that, when we connect a context

to $G$, by matching the interface of the graph with a corresponding interface of the context, the result is also a graph $G'$ with an interface, so that it can also be embedded into a new context. More precisely, the resulting graph is obtained gluing together, by means of a pushout, $G$ and the context.

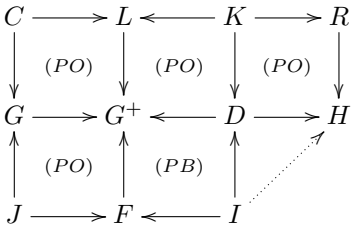
**Definition 1.** *A* graph with interface $J$ *is an injective morphism (usually an inclusion)* $J \to G$*, and a context is a cospan of injective morphisms* $J \to C \leftarrow J'$*. The result of embedding* $J \to G$ *into the context* $J \to C \leftarrow J'$ *is the graph* $J' \to G'$*, defined by the pushout diagram:*




*the resulting graph* $G'$ *will also be denoted as* $C[G]$*.*

The second idea underlying this technique is to allow for a *partial match* between the left-hand side of a rule $L$ and the graph $G$. Then, the associated transformation would start adding to $G$ the missing part of $L$ and, afterwards, applying a standard graph transformation. That is, we add to $G$ a minimal context, so that the given rule can be applied. 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. The third idea is to consider that the interface of the resulting graph is the old interface plus the borrowed context, minus the parts deleted by the rule. Finally, the last idea is to label the borrowed context transformations with the context used in the transformation step.

**Definition 2.** *Given a graph with interface* $J \to G$ *and a graph transformation rule* $p : L \leftarrow K \to R$*, we say that* $J \to G$ *reduces to* $I \to H$ *with label* $J \to F \leftarrow I$*, denoted* $(J \to G) \xrightarrow{J \to F \leftarrow I} (I \to H)$ *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:*




The intuition here is that $C$ is the subgraph of $L$ that completely matches $G$; $J \to F \leftarrow I$ is the context borrowed to extend $G$; and $G^+$ is the graph $G$ enriched with the borrowed context. In particular, $F$, 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, in Fig. 3 we can see an example of

attributed borrowed context transformation that is explained in Example 1. But not all borrowed context transformations are useful for characterizing the behaviour of a graph when embedded in any arbitrary context. This is the case of transformation where the partial match is included in the part of the interface that remains invariant after the transformation, since the same transformation could be applied to any graph with the same interface. These transformations are called *independent*.

In labelled transition systems, a bisimulation is a symmetric relation between states that is compatible with their observational behaviour. This means that if two states $s_1$ and $s_2$ are related then for every transition from $s_1$ labelled with $l$ there should be a transition from $s_2$ with the same label and the resulting states should again be related. Then, bisimilarity is the largest bisimulation relation.

**Definition 3.** *Given a set $\mathcal{T}$ of transformation rules, a relation $\mathcal{R}$ on graphs with interface is a bisimulation if it is symmetric and moreover if $(J \to G_1)\mathcal{R}(J \to G_2)$, for every transformation $(J \to G_1) \xrightarrow{J \to F \leftarrow I} (I \to H_1)$ there exists a transformation $(J \to G_2) \xrightarrow{J \to F \leftarrow I} (I \to H_2)$ such that $(I \to H_1)\mathcal{R}(I \to H_2)$.*

*Bisimilarity, denoted $\sim$ is the largest bisimulation relation (or, equivalently, the union of all bisimulation relations).*

In [5] it is proved that the condition to prove that a relation is a bisimulation can be restricted to dependent transformations. Moreover, the main result in that paper is that bisimilarity is a congruence.

*Example 1.* The running example that we use in this paper is an adaptation of the example used in [5], but now including some attributes. The rules in Fig. 1 describe communication in a network. For simplicity we have omitted the interface part of the rules, which is assumed to be the common part between their left and right-hand sides. Round nodes represent locations in the network and edges between these nodes represent communication links. There are two kinds of links, simplex communication links, represented by thin arrows, and duplex communication links, represented by thick arrows. The rules specify that messages can be sent in one direction via simplex links and in two directions via duplex links. The difference with respect to the example in [5], is that we use some attributes to describe a simple form of encrypted communication. We assume that the data algebra includes two sorts of values, *messages* and *keys*, and two operations, $e$ and $d$, for encrypting and decrypting messages, respectively. Both operations have two parameters, a key and a message, and return a message. In this sense, the rules also include square and elliptic nodes that include some data values (i.e. they are attributed). Square nodes represent messages and include a value of sort *message*, and elliptic nodes represent keys and include a value of sort *key*. Message nodes may be connected to a round node meaning that the message is at that location. Elliptic nodes may also be connected to round nodes indicating that this key is shared by the connected locations.

The three rules describe how (encrypted) messages are sent through the network. In particular, the first rule describes how a message is sent from the

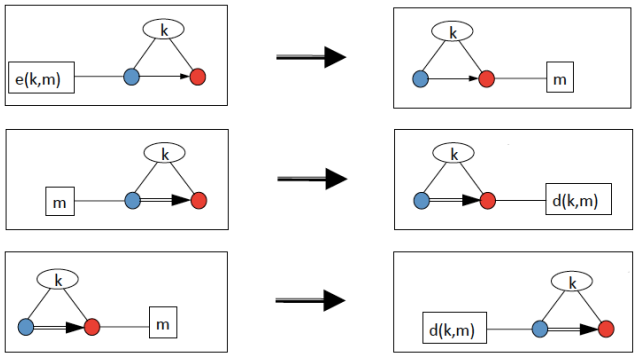



**Fig. 1.** Transformation rules

source to the target node of a simplex communication link, while the second and third rules show how a message can be sent in both directions through a duplex communication link. Moreover, the three rules describe in a different (and rather artificial) way that the messages are assumed to be encrypted before they are sent, and they are decrypted when they are received. In the first rule, the message to be sent is explicitly assumed to be encrypted, since it is the result of the term $e(k, m)$, where $k$ is the key shared by the two locations. After the transformation, the decrypted message $m$ is associated to the target node. In the other two rules, it is not explicitly stated if the message $m$ attached to the sending node is encrypted or not. However, the message received is explicitly decrypted, since the message received is the result of $d(k, m)$.


**Fig. 2.** Bisimilar graphs

The example used in [5] is similar but graphs have no attributes: there are no keys or encryption/decryption of messages, and messages are not assumed to be values of any kind, but just nodes. Then, using the techniques described in the paper, it is shown that the two graphs $J \to G$ and $J \to G'$, depicted in Fig. 2, are bisimilar. In particular, they analyze what are all the borrowed context transformations that can be applied to the two graphs and check that for every transformation applied to one of them, there is another transformation that can be applied to the other one, with the same label, such that the resulting graphs $I \to H$ and $I \to H'$ are extensions, with the same context, of $J \to G$ and $J \to G$, respectively. And this means that $J \to G$ and $J \to G'$ are bisimilar.

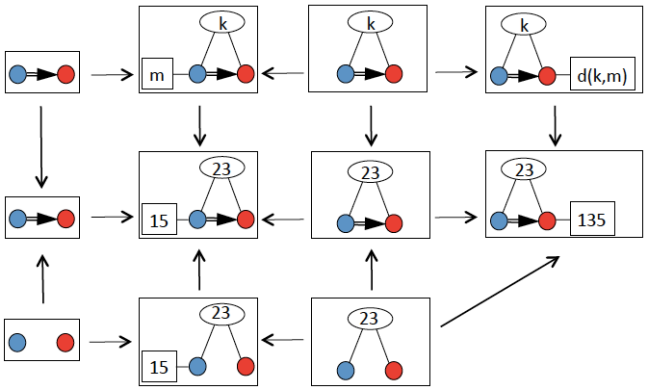



**Fig. 3.** Attributed borrowed context transformation

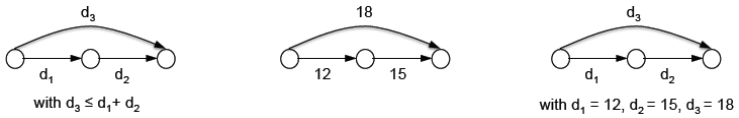
However, in our example, that kind of proof is not possible. The main problem is that, when applying an attributed graph transformation rule to an attributed graph, all the variables in the rule must be matched to some given data values and, accordingly, the terms included must be evaluated. This applies also to borrowed context transformation, but variables in the context would not be matched but must be substituted by arbitrary values. For instance, in Fig. 3 we show a possible borrowed context transformation by applying the second rule in Fig. 1 to the graph $J \to G'$ on the left in Fig. 2, assuming that messages and keys are integers, that the variables $m$ and $k$ are substituted by 23 and 15, respectively, and that the value of $d(23, 15) = 135$. and other substitutions of $m$ and $k$ would lead to a different borrowed context transformation. This causes that, to prove the bisimilarity of the above two graphs we would need to prove the bisimilarity of an infinite number of graphs. A second problem is that the data (the attributes) in the rules do not exactly coincide. Actually, as we will see in Sect. 5, the two graphs are bisimilar if the encrypting and decrypting operations are the inverse of each other. Then it may be unclear how to prove this fact in the context of attributed graph transformation.

## 3 Symbolic graphs and symbolic graph transformation

A symbolic graph is a graph that specifies a class of attributed graphs. More precisely, a symbolic graph $SG = \langle G, \Phi \rangle$ over a given data algebra $\mathcal{A}$ is a kind of labelled graph $G$ (technically, an E-graph [3]), whose nodes and edges may be decorated with labels from a given set of variables $X$, together with a set of formulas $\Phi$ over these variables and over the values in $\mathcal{A}$. The intuition is that each substitution $\sigma : X \to \mathcal{A}$ of the variables in $X$ by values of $\mathcal{A}$ such that $\mathcal{A} \models \sigma(\Phi)$, defines an attributed graph $\sigma(G)$ in the semantics of $G$, obtained replacing each variable $x$ in $G$ by the corresponding data value $\sigma(x)$. That is, the semantics of $SG$ is defined:

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

For instance, the graph below on the left specifies a class of attributed graphs, including distances in the edges, that satisfy the well-known triangle inequality, and the graph in the center would belong to its semantics




It may be noticed that every attributed graph may be seen as a symbolic graph by just replacing all its values by variables, and by including an equation $x_v = v$, into the corresponding set of formulas, for each value $v$, where $x_v$ is the variable that has replaced the value $v$. We call these kind of symbolic graphs *grounded symbolic graphs*. In particular, $GSG(G)$ denotes the grounded symbolic graph defined by $G$. For instance, the graph above on the right, can be seen as the symbolic representation of the attributed graph in the center.

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 set of formulas obtained when replacing in $\Phi_1$ 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 $\mathbf{SymbGraph}_{\mathcal{A}}$.

For (technical) simplicity, we assume that in our graphs no variable is bound to two different elements of the graph. We call these graphs in *normal form*. It should be clear that this is not a limitation since every symbolic graph $SG$ is equivalent to a symbolic graph $SG'$ in normal form, in the sense that $Sem(SG) = Sem(SG')$. 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 $\Phi_{G'}$.

In [7], we showed that $\mathbf{SymbGraph}_{\mathcal{A}}$ is an adhesive HLR category [6, 4] taking as M-morphisms all injective graph morphisms where the formulas constraining the source and target graphs are equivalent. In particular, the proposition below shows how pushouts of symbolic graphs are defined:

**Proposition 1.** *[7] Diagram* (1) *below is a pushout if and only if diagram* (2) *is also a pushout and* $\mathcal{A} \models \Phi_3 \Leftrightarrow (g_1(\Phi_1) \cup g_2(\Phi_2))$.

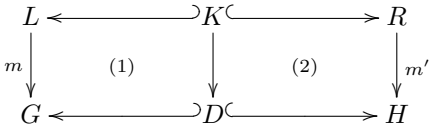
$$\begin{array}{ccc} \langle G_0, \Phi_0\rangle & \xrightarrow{h_1} & \langle G_1, \Phi_1\rangle \\ {\scriptstyle h_2}\downarrow & (1) & \downarrow{\scriptstyle g_1} \\ \langle G_2, \Phi_2\rangle & \xrightarrow[g_2]{} & \langle G_3, \Phi_3\rangle \end{array} \qquad \begin{array}{ccc} G_0 & \xrightarrow{h_1} & G_1 \\ {\scriptstyle h_2}\downarrow & (2) & \downarrow{\scriptstyle g_1} \\ G_2 & \xrightarrow[g_2]{} & G_3 \end{array}$$

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 the sets of variables $X_L, X_K$ and $X_R$, respectively, and $\Phi$ is a set of formulas over $X_R$ and over the values in $\mathcal{A}$. We consider that a rule is a span of symbolic graph inclusions $\langle L, \emptyset\rangle \hookleftarrow \langle K, \emptyset\rangle \hookrightarrow$

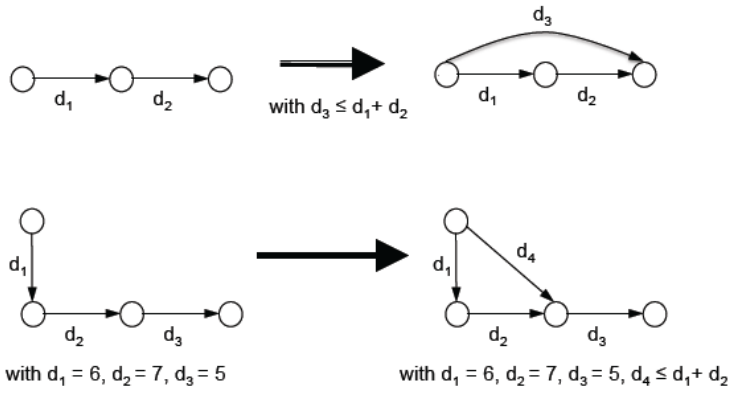
$\langle R, \Phi \rangle$, Intuitively, $\Phi$ relates the attributes in the left and right-hand side of the rule. This means that we implicitly assume that $X_L = X_K \subseteq X_R$. In [9] we allow for more general rules.

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 [9]).

**Definition 4.** *Given a transformation rule $r = \langle L \hookleftarrow K \hookrightarrow R, \Phi \rangle$ over a data algebra $\mathcal{A}$ and a morphism $m : L \to G$, $\langle G, \Phi' \rangle \Longrightarrow_{r,m} \langle H, \Phi' \cup m'(\Phi) \rangle$ if and only if the diagram below is a double pushout and $\Phi' \cup m'(\Phi)$ is satisfiable in the given data algebra.*




For instance in the upper part of the figure below, we can see a symbolic graph transformation rule stating that if a given graph has two consecutive edges, $e_1$ and $e_2$, with some given distances $d_1$ and $d_2$, respectively, then we can add a new edge from the source of $e_1$ to the target of $e_2$ whose distance must be smaller or equal than $d_1 + d_2$. Moreover, in the bottom part of the figure we may see the result of applying that rule to the graph on the left.




We may notice that, in general, $\Phi' \cup m'(\Phi)$ may be unsatisfiable. This would mean that the resulting graph $\langle H, \Phi' \cup m'(\Phi) \rangle$ would have an empty semantics, i.e. it would be inconsistent. This is avoided by requiring explicitly that $\Phi' \cup m'(\Phi)$ must be satisfiable. It is not difficult to show that the above construction defines a double pushout in the category of symbolic graphs [9].

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 $r = \langle L \hookleftarrow K \hookrightarrow R, \Phi \rangle$ denotes the class of all rules $\sigma(L) \hookleftarrow \sigma(K) \hookrightarrow \sigma(R)$, where $\sigma$ is a substitution such that $\mathcal{A} \models \sigma(\Phi)$, i.e.:

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

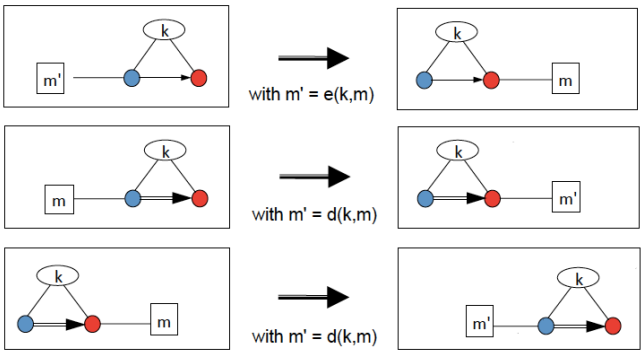



**Fig. 4.** Symbolic transformation rules

It is not difficult to see that given a rule $r$ and a symbolic graph $SG$, $SG \Longrightarrow_r SG'$ if and only if for every graph $G \in Sem(SG)$, $G \Longrightarrow_r G'$, with $G' \in Sem(SG')$ and moreover, for every graph $G' \in Sem(SG')$, $G \Longrightarrow_r G'$, for some $G \in Sem(SG)$ [7].

We will require that transformation rules must be *strict* meaning that the result of a rule application to a grounded model must also be grounded[3]:

**Definition 5.** *A transformation rule* $\langle L \hookleftarrow K \hookrightarrow R, \Phi \rangle$ *is* strict *if for every transformation* $\langle G, \Phi' \rangle \Longrightarrow_{p,m} \langle H, \Phi' \cup m'(\Phi) \rangle$, *whenever* $\langle G, \Phi' \rangle$ *is grounded* $\langle H, \Phi' \cup m'(\Phi) \rangle$ *is also grounded.*

For instance, the rule in the figure above is not strict. However, assuming that all rules must be strict is not really a restriction since every non-strict rule may be made strict including in $X_L$ all the variables in $X_R$. This forces to match every variable in $X_R$ to some given value causing that the result of the transformation $\langle H, \Phi' \cup m'(\Phi) \rangle$ must be grounded. Moreover, it is easy to see that the semantics of a non-strict rule (as defined above) and its associated strict one coincide.

As shown in [7], symbolic graph transformation is more powerful than attributed graph transformation. In particular, any attributed graph transformation rule $r$ can be represented by a symbolic graph transformation rule $SR(r)$ such that an attributed graph $G$ can be transformed by $r$ into a graph $H$ if and only if the grounded graph associated to $G$, $SG$, can be transformed by $SR(r)$ into the grounded graph associated to $H$. However, the converse is not true.

For example, in Fig 4 we show the symbolic rules associated to the attributed rules depicted in Fig 1.

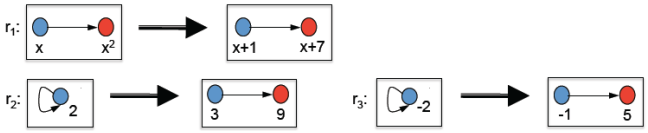
[3] This does not mean that the result of a borrowed context transformation of a grounded graph using a strict rule must also be grounded. In particular, the rules in Example 1 are strict but, as shown in Example 3, the results of their borrowed context application to two grounded graphs are not grounded.

## 4 Symbolic bisimilarity

In this section we study a notion of bisimilarity for symbolic graphs to be used for proving the bisimilarity of attributed graphs. First we show that a definition of symbolic bisimulation just applying the concepts introduced in Section 2 to $\mathbf{SymbGraph}_{\mathcal{A}}$ is not adequate for checking the bisimilarity of attributed graphs. Then, we define a notion of symbolic bisimulation and show that it coincides with attributed bisimulation, when restricted to grounded graphs. Finally, we show that this symbolic bisimilarity is also a congruence on $\mathbf{SymbGraph}_{\mathcal{A}}$.

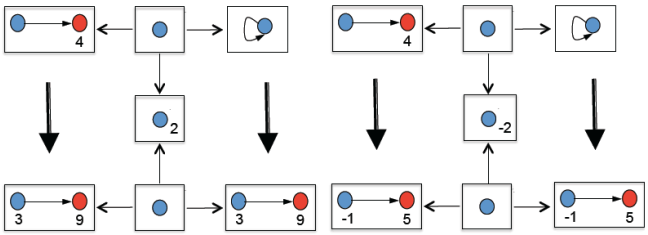
Symbolic graphs form an M-adhesive category [7]. Hence, all the definitions and results in [5] concerning borrowed context transformation and bisimilarity apply to this category. For instance, we have notions of symbolic graph with interface, of context, and of borrowed context transformation exactly as in Section 2, but within the category of symbolic graphs. To be precise, we consider that graph interfaces are not arbitrary symbolic graphs, but graphs with an empty set of conditions, since we consider that the interface must only specify the open part of a graph. Then, we may think that a direct application of these results may be a solution for the problem of checking bisimilarity for attributed graphs, since we may directly work with terms and variables, without having to compute all its possible substitutions, as described in Example 1. This means 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, as we can see in the counter-example below, two attributed graphs may be bisimilar as attributed graphs, while their associated grounded symbolic graphs are not bisimilar as symbolic graphs.

*Example 2.* Let us consider the attributed graph transformation system, consisting of three rules, depicted below.

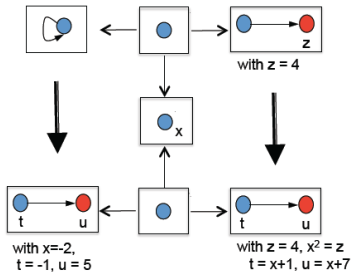



The first rule includes a variable $x$ used in different expressions including $x^2$. The remaining two rules are simple attributed rules with integer values. Now, let us consider the two attributed graphs, including as interface just the source node of the edges in the graphs, $I \to G_1$ and $I \to G_2$ that are depicted below.


We can see that these graphs are bisimilar. The only dependent borrowed context transformations that we can apply on both graphs are depicted below and, in both cases, the resulting graphs are equal.




Notice that, in the figure, we do not depict the data values that are not bound to any node or edge. However, if we consider the symbolic versions of the above rules we can show that $GSG(I \to G_1)$ and $GSG(I \to G_2)$ are not bisimilar. In particular, below we can find the symbolic borrowed context transformation of the two grounded graphs using the symbolic versions of $r_3$ and $r_1$, and we may see that no direct transformation can be applied to the resulting graph on the left, $GSG(I \to G_4)$. However, we may also see that the conditions associated to the resulting graph on the right are equivalent to $(z = 4 \wedge x = 2 \wedge t = 3 \wedge u = 9) \vee (z = 4 \wedge x = -2 \wedge t = -1 \wedge u = 5)$, which means that we can transform this graph using the symbolic version of $r_1$ (matching $x$ to 3).




The problem in the above counter-example is that, when considering attributed graph transformation, each instance of the rule $r_1$ (i.e. when $x = 2$ or when $x = -2$) is simulated by $r_2$ and $r_3$, respectively, and vice versa. However, when considering symbolic transformation, we need to say that $r_1$ is simulated by $r_2$ and $r_3$ *together*, and vice versa. This is not possible if we define symbolic bisimulation as in [5]. Instead, we define a new notion that solves this problem:

**Definition 6.** *A relation $\mathcal{R}$ on symbolic graphs with interface is a symbolic bisimulation with respect to a set of transformation rules, if it is symmetric and moreover if $(J \to SG_1)\mathcal{R}(J \to SG_2)$, for every transformation $(J \to SG_1) \xrightarrow{J \to F \leftarrow I} (I \to SG_1')$, with $SG_1' = \langle G_1', \Phi_1' \rangle$ there exist a family of conditions $\{\Psi_i\}_{i \in \mathcal{I}}$ and a family of transformations $\{(J \to SG_2) \xrightarrow{J \to F \leftarrow I} (I \to SH_i)\}_{i \in \mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i \rangle$ such that:*

- *For every substitution $\sigma_1'$ such that $\mathcal{A} \models \sigma_1'(\Phi_1')$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$ and $\sigma_1' \restriction_F = \sigma_i \restriction_F$, where $\sigma \restriction_F$ denotes the restriction of $\sigma$ to the variables in $F$.*

– *For every i,* $(I \to \langle G'_1, \Phi'_1 \cup \Psi_i \rangle)\ \mathcal{R}\ (I \to \langle H_i, \Pi_i \cup \Psi_i \rangle)$.

*Symbolic bisimilarity, denoted* $\sim_S$*, is the largest symbolic bisimulation relation.*

For instance, in the above counter-example, to show $GSG(I \to G_1) \sim_S GSG(I \to G_2)$, the symbolic transformation of $GSG(I \to G_2)$ via $r_3$ depicted above on the left, would be simulated by the transformation of $GSG(I \to G_1)$ via rule $r_1$, together with the condition $x = -2$. Similarly, the symbolic transformation of $GSG(I \to G_1)$ via $r_1$ depicted above on the right, would be simulated by the transformations of $GSG(I \to G_2)$ via rule $r_2$, together with the condition $x = 2$ and of $GSG(I \to G_2)$ via rule $r_3$, together with the condition $x = -2$.

Using symbolic bisimilarity we can prove the bisimilarity of attributed graphs:

**Theorem 1.** *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* $\mathcal{T}$*.*

To prove this theorem we use two lemmas:

**Lemma 1.** *Let* $\mathcal{R}$ *be the following relation defined on symbolic graphs.* $(J \to SG_1)\mathcal{R}(J \to SG_2)$ *if:*

– *For every attributed graph* $\sigma_1(J \to SG_1) \in Sem(J \to SG_1)$ *there is an attributed graph* $\sigma_2(J \to SG_2) \in Sem(J \to SG_2)$ *such that* $\sigma_1(J \to SG_1) \sim \sigma_2(J \to SG_2)$*.*
– *For every attributed graph* $\sigma_2(J \to SG_2) \in Sem(J \to SG_2)$ *there is an attributed graph* $\sigma_1(J \to SG_1) \in Sem(J \to SG_1)$ *such that* $\sigma_1(J \to SG_1) \sim \sigma_2(J \to SG_2)$*.*

*Then,* $\mathcal{R}$ *is a bisimulation.*

**Lemma 2.** *The relation on attributed graphs* $(J \to G_1)\mathcal{R}(J \to G_2)$ *if* $GSG(J \to G_1) \sim_S GSG(J \to G_2)$ *is a bisimulation.*

To prove this theorem we have that, if $(J \to G_1) \sim (J \to G_2)$ then $GSG(J \to G_1)$ and $GSG(J \to G_2)$ satisfy the conditions of Lemma 1. So they must be bisimilar. Conversely, if $GSG(J \to G_1) \sim_S GSG(J \to G_2)$, lemma 2 directly implies that $(J \to G_1)$ and $(J \to G_2)$ are bisimilar.

The last theorem shows that symbolic bisimilarity is a congruence:

**Theorem 2.** *If* $(J \to SG_1) \sim_S (J \to SG_2)$ *then, for every context* $J \to C \leftarrow I$*,* $(I \to C[SG_1]) \sim_S (I \to C[SG_2])$*.*

The proof uses a property shown in [5] that if a graph $J \to G$ is embedded in $J' \to G'$ with a given context and if $J' \to F' \leftarrow I'$ is a possible label for transforming $J' \to G'$ then there is a context $I \to C' \leftarrow I'$ and a label $J \to F \leftarrow I$, such that any transformation of $J' \to G'$ with label $J' \to F' \leftarrow I'$ can be obtained by first transforming $J \to G$ with a label $J \to F \leftarrow I$ and, then, embedding the result in the context $I \to C \leftarrow I'$.

## 5 Checking bisimilarity

In this section we show the basic ideas of how we can use the previous results to show that two attributed graphs $(J \to G_1)$ and $(J \to G_2)$ are bisimilar. In principle, we would need to consider all the possible borrowed context transformations with the same label of their associated grounded graphs and show that we can group them in pairs, so that under suitable conditions $\Psi_i$, the resulting graphs are bisimilar. The obvious problem is that this clearly leads to a non-terminating process. To avoid this non-termination (if possible) Sangiorgi [12] defined the notion of bisimulation up to context that is adapted to the case of graph transformation in [5]. In our case this notion would be defined as follows:

**Definition 7.** *A relation $\mathcal{R}$ is a symbolic bisimulation up to context if whenever $(J \to SG_1)\mathcal{R}(J \to SG_2)$, then for every transformation $(J \to SG_1) \xrightarrow{J \to F \leftarrow I} (I \to SG'_1)$, with $SG'_1 = \langle G'_1, \Phi'_1 \rangle$ there exist a family of conditions $\{\Psi_i\}_{i \in \mathcal{I}}$ and a family of transformations $\{(J \to SG_2) \xrightarrow{J \to F \leftarrow I} (I \to SH_i)\}_{i \in \mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i \rangle$ such that:*

- *For every substitution $\sigma'_1$ such that $\mathcal{A} \models \sigma'_1(\Phi'_1)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$ and $\sigma'_1 \restriction_F = \sigma_i \restriction_F$.*
- *For every $i$, $(I \to \langle G'_1, \Phi'_1 \cup \Psi_i \rangle)$ and $(I \to \langle H_i, \Pi_i \cup \Psi_i \rangle)$ are the result of embedding $(J \to SG_1)$ and $(J \to SG_2)$ in the same context.*

**Proposition 2.** *[5] If $\mathcal{R}$ is a symbolic bisimulation up to context then $\mathcal{R} \subseteq \sim_S$.*

This means that if, when trying to check if $(J \to SG_1)$ and $(J \to SG_2)$, we have to prove that $(I \to SG'_1)$ and $(I \to SG'_2)$ are also bisimilar and if the latter graphs can be obtained by embedding the former graphs into the same context, then we can consider that $(I \to SG'_1)$ and $(I \to SG'_2)$ are bisimilar. So, if this happens for all the pairs of graphs that we have to prove bisimilar, then we can conclude that $(J \to SG_1)$ and $(J \to SG_2)$ are indeed bisimilar.

Moreover, as in [5], we can restrict ourselves to checking only *dependent* transformations. This is important since, if the number of transformation rules is finite, and the given graph, $J \to G$, is also finite, then there is a finite number of possible borrowed context transformations that can be applied to $J \to G$. Let us now show how these ideas would be applied to the example in Section 2.

*Example 3.* We want to check if the graphs $J \to G$ and $J \to G'$ depicted in Fig. 2 are bisimilar. In this case, since $G$ and $G'$ include no explicit attributes, their associated grounded graphs would look similar. Now, there are only two dependent borrowed context transformations that can be applied to each of the two graphs. In Fig. 5 we depict the transformations that can be applied to $J \to G$ and $J \to G'$, using the first and the second rule in Fig. 4, respectively, when we add to the two graphs a context consisting of a common key $k$ and a message $m$ attached to the leftmost node. In particular, on the left and the right of the figure we depict the transformations of $G$ and $G'$ and, in the middle, we depict the label

of the transformation. Fig. 6 is similar, and describes the transformations over $J \to G$ and $J \to G'$, using the first and the third rules, respectively.

To prove $J \to G \sim_S J \to G'$, according to the definition of symbolic bisimulation, we have to show, on the one hand, that the conditions associated to the transformations, $\{m = e(k, m')\}$ and $\{m' = d(k, m)\}$ are equivalent; and, on the other hand, that the resulting graphs are bisimilar, $(I \to SH_1) \sim_S (I \to SH_1')$, with $SH_1 = \langle H1, \{m = e(k, m')\}\rangle$ and $SH_1' = \langle H1', \{m' = d(k, m)\}\rangle$, and $(I \to SH_2) \sim_S (I \to SH_2')$, with $SH_2 = \langle H2, \{m = e(k, m')\}\rangle$ and $SH_2' = \langle H2', \{m' = d(k, m)\}\rangle$. But if the conditions are equivalent, $SH_i$ and $SH_i'$ $(i = 1, 2)$ are just the original graphs extended by the same context. Therefore, the bisimilarity of the graphs $J \to G$ and $J \to G'$ depends on the equivalence of the above conditions. More precisely, if the given data algebra $\mathcal{A}$ satisfies:

$$d(k, e(k, m)) = m$$
$$e(k, d(k, m)) = m$$

i.e. if encryption and decryption are the inverse of each other.

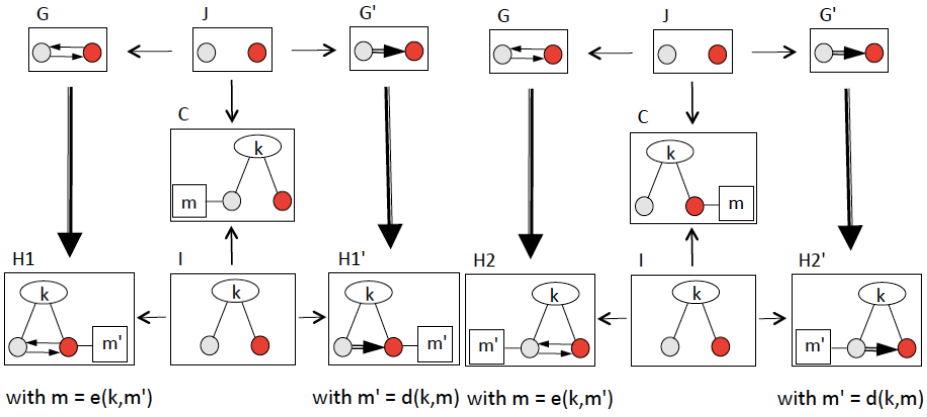



**Fig. 5.** Transformation 1




**Fig. 6.** Transformation 2

## 6 Conclusion and related work

Bisimilarity was introduced by Park in [10] and, since then, it has been studied by many authors in relation to many different formalisms. In [5], Ehrig and König not only introduced a notion of bisimilarity for graph transformation systems, but they provided a simple and general technique to derive labelled transitions and bisimulation congruences from unlabelled ones. This paper was followed by

[11] where they showed how these techniques could be used for the verification of bisimilarity. Unfortunately, as we have seen in this paper, these techniques do not work in the case of attributed graphs. Borrowed context transformations have been used for the definition of other behavioral equivalence relations [2]. On the other hand, symbolic graphs were introduced in [8] in order to define constraints on attributed graphs. Then symbolic graph transformation was studied in detail in [7, 9].

In this paper we have presented a new notion of bisimulation for symbolic graph transformation that has been shown to be useful for checking the bisimilarity of attributed graphs. The key issue is that in symbolic graph transformation we do not need to replace all the variables by values. Moreover, the neat separation in symbolic graphs between the graph structure and the algebra of data also helps for this purpose. Currently we are working in devising a specific proof method to implement these ideas.

## 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., Montanari, U.: Saturated ltss for adhesive rewriting systems. In: Graph Transformations - 5th International Conference, ICGT 2010. Lect. Notes in Comp. Sc., vol. 6372, pp. 123–138. Springer (2010)
3. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs of Theoretical Comp. Sc., Springer (2006)
4. Ehrig, H., Padberg, J., Prange, U., Habel, A.: Adhesive high-level replacement systems: A new categorical framework for graph transformation. Fundamenta Informaticae 74(1), 1–29 (2006)
5. Ehrig, H., König, B.: Deriving bisimulation congruences in the dpo approach to graph rewriting with borrowed contexts. Mathematical Structures in Computer Science 16(6), 1133–1163 (2006)
6. Lack, S., Sobocinski, P.: Adhesive and quasiadhesive categories. Theoretical Informatics and Applications 39, 511–545 (2005)
7. Orejas, F., Lambers, L.: Symbolic attributed graphs for attributed graph transformation. In: Int. Coll. on Graph and Model Transformation On the occasion of the 65th birthday of Hartmut Ehrig (2010)
8. Orejas, F.: Symbolic graphs for attributed graph constraints. J. Symb. Comput. 46(3), 294–315 (2011)
9. Orejas, F., Lambers, L.: Lazy graph transformation. Fundamenta Informaticae To appear (2011)
10. Park, D.: Concurrency and automata on infinite sequences. In: Theoretical Computer Science, 5th GI-Conference. Lecture Notes in Computer Science, vol. 104, pp. 167–183. Springer (1981)
11. Rangel, G., König, B., Ehrig, H.: Bisimulation verification for the dpo approach with borrowed contexts. ECEASST 6 (2007)
12. Sangiorgi, D.: On the proof method for bisimulation. In: Mathematical Foundations of Computer Science 1995, 20th International Symposium, MFCS'95. Lecture Notes in Computer Science, vol. 969, pp. 479–488. Springer (1995)

## APPENDIX

In this appendix we provide the proofs of the results that are omitted in the paper. Moreover, the proposition below is used in some proofs below:

**Proposition 3.** *If* $(J \to \langle G_1, \Phi_1\rangle) \sim_S (J \to \langle G_2, \Phi_2\rangle))$ *then, for any set of conditions* $\Phi$ *over the common variables of* $\Phi_1$ *and* $\Phi_2$, $(J \to \langle G_1, \Phi_1 \cup \Phi\rangle) \sim_S (J \to \langle G_2, \Phi_2 \cup \Phi\rangle)$.

*Proof.* If $(J \to \langle G_1, \Phi_1 \cup \Phi\rangle) \xrightarrow{J \to C \leftarrow I} J \to (\langle G'_1, \Phi'_1\rangle)$ then $(J \to \langle G_1, \Phi_1\rangle) \xrightarrow{J \to C \leftarrow I} J \to (\langle G'_1, \Phi''_1\rangle)$ and, by definition of symbolic graph transformation, $(\Phi''_1 \cup \Phi) \equiv \Phi'_1$. But this means that there exist a family of conditions $\{\Psi_i\}_{i\in\mathcal{I}}$ and a family of transformations $\{(J \to SG_2) \xrightarrow{J \to F \leftarrow I} (I \to SH_i)\}_{i\in\mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i\rangle$ such that:

- For every substitution $\sigma'_1$ such that $\mathcal{A} \models \sigma'_1(\Phi''_1)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$ and $\sigma'_1 \restriction_F = \sigma_i \restriction_F$. But this implies that for every substitution $\sigma'_1$ such that $\mathcal{A} \models \sigma'_1(\Phi''_1 \cup \Phi)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i \cup \Phi)$ and $\sigma'_1 \restriction_F = \sigma_i \restriction_F$.
- For every $i$, $(I \to \langle G'_1, \Phi'_1 \cup \Psi_i\rangle) \mathcal{R} (I \to \langle H_i, \Pi_i \cup \Psi_i\rangle)$, and by coinduction we may assume that for every $i$, $(I \to \langle G'_1, \Phi'_1 \cup \Psi_i \cup \Phi\rangle) \mathcal{R} (I \to \langle H_i, \Pi_i \cup \Psi_i \cup \Phi\rangle)$.

Therefore, $(J \to \langle G_1, \Phi_1 \cup \Phi\rangle) \sim_S (J \to \langle G_2, \Phi_2 \cup \Phi\rangle)$. □

*Proof of Lemma 1* Assume there is a transformation $J \to SG_1 \xrightarrow{J \to C \leftarrow I} I \to \langle G, \Phi\rangle$, we have to show that there is a family of conditions $\{\Psi_i\}_{i\in\mathcal{I}}$ and a family of transformations $J \to SG_2 \xrightarrow{J \to C \leftarrow I} I \to \langle H_i, \Pi_i\rangle$, such that for every $\sigma$ with $\mathcal{A} \models \sigma(\Phi)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$, and $\sigma \restriction_F = \sigma_i \restriction_F$ and $\langle G, \Phi \cup \Psi_i\rangle \mathcal{R} \langle H_i, \Pi_i \cup \Psi_i\rangle$.

For every transformation $J \to SG_2 \xrightarrow{J \to C \leftarrow I} I \to \langle H_{r,m}, \Pi_{r,m}\rangle$ via a rule $r$ with (partial) matching $m$, we define the following family of sets of conditions:

$$\begin{aligned}\Psi_{r,m} = \{\sigma_1 \restriction_C \mid\ & \sigma_1(J \to SG_1) \xrightarrow{\sigma_1(J \to C \leftarrow I)} \sigma_1(I \to G),\\ & \sigma_2(J \to SG_2) \xrightarrow{\sigma_2(J \to C \leftarrow I)} \sigma_2(I \to H_{r,m}),\\ & \sigma_1 \restriction_C = \sigma_2 \restriction_C,\\ & \sigma_1(I \to G) \sim \sigma_2(I \to H_{r,m})\}\end{aligned}$$

Note that in the above definition, we consider that a substitution $(\sigma_1 \restriction_C)$ denotes a set of conditions. Actually, any substitution $\sigma : X \to \mathcal{A}$ may be considered to denote the set of equations $\{x = v \mid x \in X \wedge \sigma(x) = v\}$.

Then, by construction, we have that, for all sets of conditions $\Psi_{r,m}$, $\langle G, \Phi \cup \Psi_{r,m}\rangle \mathcal{R} \langle H_{r,m}, \Pi_{r,m} \cup \Psi_{r,m}\rangle$ and for every $\sigma_1$ with $\mathcal{A} \models \sigma'_1(\Phi)$, there are $r$, $m$ and $\sigma_{r,m}$ such that $\mathcal{A} \models \sigma_i(\Psi_{r,m} \cup \Pi_{r,m})$, and $\sigma_1 \restriction_F = \sigma_{r,m} \restriction_F$.

The converse direction can be shown in a similar way, showing that $\mathcal{R}_s$ is a symbolic bisimulation. □

*Proof of Lemma 2* Let us suppose that $(J \to G_1) \xrightarrow{\sigma(J \to F \leftarrow I)} (I \to G_1')$ via an instance $\sigma(r)$ of a given symbolic rule $r$. This means that $GSG(J \to G_1) \xrightarrow{J' \to F' \leftarrow I'} (I' \to \langle G_1'', \Phi_1'' \rangle)$ via $r$ and $I \to G_1'$ is in the semantics of $I' \to \langle G_1'', \Phi_1'' \rangle$. Therefore, there is a symbolic transformation $GSG(J \to G_2) \xrightarrow{J' \to F' \leftarrow I'} (I' \to \langle G_2', \Phi_2' \rangle)$ such that $I' \to \langle G_1'', \Phi_1'' \rangle \sim_S I' \to \langle G_2', \Phi_2' \rangle$. But according to Proposition **??** this means that $I' \to \langle G_1'', \Phi_1'' \cup \sigma \rangle \sim_S I' \to \langle G_2', \Phi_2' \cup \sigma \rangle$. But since we assume that transformation rules are strict then $I' \to \langle G_1'', \Phi_1'' \cup \sigma \rangle$ and $I' \to \langle G_2', \Phi_2' \cup \sigma \rangle$ must be grounded and $GSG((I \to G_1')) = (I' \to \langle G_1'', \Phi_1'' \cup \sigma \rangle)$. Moreover if $I \to G_2'$ is in the semantics of $\langle G_2', \Phi_2' \cup \sigma \rangle$ then $(J \to G_2) \xrightarrow{\sigma(J \to F \leftarrow I)} (I \to G_2')$ and $(I \to G_1') \mathcal{R} (I \to G_2')$. Implying that $\mathcal{R}$ is a bisimulation. □

A key step to prove that symbolic bisimilarity is a congruence, is the following property that is proved in [5]

**Proposition 4.** *[5] For any two contexts $J_1 \to C \leftarrow J_1'$ and $J_1' \to C_1' \leftarrow J_2'$ there exist contexts $J_1 \to C_1 \leftarrow J_2$ and $J_2 \to C' \leftarrow J_2'$ such that, for every graph $J_1 \to G_1$, $(J_1' \to C[G_1]) \xrightarrow{J_1' \to C_1' \leftarrow J_2'} (J_2' \to H_1')$ if and only if $(J_1 \to G_1) \xrightarrow{J_1 \to C_1 \leftarrow J_2} (J_2 \to H_1)$ and $J_2' \to H_1' = J_2' \to C'[H_1]$.*

*Proof of Theorem 2* Let us assume that $(I \to C[SG_1]) \xrightarrow{I \to F \leftarrow I'} (I' \to SG_1')$ then, according to Prop. 4, there are contexts $J' \to C' \leftarrow I'$ and $J \to F' \leftarrow J'$, such that $(J \to SG_1) \xrightarrow{J \to F' \leftarrow J'} (J' \to SG)$ and $(I' \to SG_1') = (I' \to C'[SG])$. This means that there exist a family of conditions $\{\Psi_i\}_{i \in \mathcal{I}}$ and a family of transformations $\{(J \to SG_2) \xrightarrow{J \to F \leftarrow I} (I \to SH_i)\}_{i \in \mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i \rangle$ such that:

- For every substitution $\sigma$ such that $\mathcal{A} \models \sigma(\Phi)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$ and $\sigma \restriction_F = \sigma_i \restriction_F$. Then, for every substitution $\sigma'$ such that $\mathcal{A} \models \sigma'(\Phi) \cup \Phi_{C'}$, there is an index $i$ and a substitution $\sigma_i'$ such that $\mathcal{A} \models \sigma_i'(\Psi_i \cup \Pi_i \cup \Phi_{C'})$ and $\sigma' \restriction_{F'} = \sigma_i' \restriction_{F'}$.
- For every $i$, $(I \to \langle G, \Phi \cup \Psi_i \rangle) \sim_S (I \to \langle H_i, \Pi_i \cup \Psi_i \rangle)$.

But, by Prop. 4, for every $i$, we have that $(I \to C[SG_2]) \xrightarrow{I \to F \leftarrow I'} (I' \to C'[SH_i])$. But, by coinduction, we may assume that for every $i$, $(I' \to C'[\langle G, \Phi \cup \Psi_i \rangle]) \sim_S (I' \to C'[\langle H_i, \Pi_i \cup \Psi_i \rangle])$, and this implies that $(I \to C[SG_1]) \sim_S (I \to C[SG_2])$. □

*Proof of Proposition 2* Let $\mathcal{R}'$ be the relation $(J' \to SG_1') \mathcal{R}' (J' \to SG_2')$ if there is a context $J \to C \leftarrow J'$ and graphs $(J \to SG_1)$ and $(J \to SG_2)$, such that $(J' \to SG_1') = (J' \to C[SG_1])$, $(J' \to SG_2') = (J' \to C[SG_2])$ and $(J \to SG_1) \mathcal{R} (J \to SG_2)$. Let us show that $\mathcal{R}'$ is a bisimulation.

If $(J' \to SG_1') \xrightarrow{J' \to F' \leftarrow I'} (I' \to SG')$, with $SG' = \langle G', \Phi' \rangle$, according to Prop. 4, there are contexts $I \to C' \leftarrow I'$ and $J \to F \leftarrow I$, such that $(J \to SG_1) \xrightarrow{J \to F \leftarrow I} (I \to SH)$ and $(I' \to SH') = (I' \to C'[SH])$.

This means that there exist a family of conditions $\{\Psi_i\}_{i\in\mathcal{I}}$ and a family of transformations $\{(J \to SG_2) \xrightarrow{J\to F\leftarrow I} (I \to SH_i)\}_{i\in\mathcal{I}}$, with $SH_i = \langle H_i, \Pi_i\rangle$ such that:

- For every substitution $\sigma$ such that $\mathcal{A} \models \sigma(\Phi)$, there is an index $i$ and a substitution $\sigma_i$ such that $\mathcal{A} \models \sigma_i(\Psi_i \cup \Pi_i)$ and $\sigma \restriction_F = \sigma_i \restriction_F$, and this means that, for every substitution $\sigma'$ such that $\mathcal{A} \models \sigma'(\Phi \cup \Phi_{C'})$, there is an index $i$ and a substitution $\sigma'_i$ such that $\mathcal{A} \models \sigma'_i(\Psi_i \cup \Pi_i \cup \Phi_{C'})$ and $\sigma' \restriction_{F'} = \sigma'_i \restriction_{F'}$.
- For every $i$, $(I \to \langle G, \Phi \cup \Psi_i\rangle)$ and $(I \to \langle H_i, \Pi_i \cup \Psi_i\rangle)$ are the result of embedding $(J \to SG_1)$ and $(J \to SG_2)$ into the same context $J \to K_i \leftarrow I$. But by Prop. 4, we know that $\{(J' \to SG'_2) \xrightarrow{J'\to F'\leftarrow I'} (I' \to C'[SH_i])\}_{i\in\mathcal{I}}$. Hence, $(I' \to \langle G', \Phi' \cup \Psi_i\rangle)$ and $(I' \to C'[\langle H_i, \Pi_i \cup \Psi_i\rangle])$ are the result of embedding $(J \to SG_1)$ and $(J \to SG_2)$ into the composition of the contexts $J \to K_i \leftarrow I$ and $I \to C' \leftarrow I'$. Thus $(I' \to \langle G', \Phi' \cup \Psi_i\rangle)\mathcal{R}'(I' \to C'[\langle H_i, \Pi_i \cup \Psi_i\rangle])$.

Now, from the definitions of bisimulation and bisimulation up to context, and from the facts that $\mathcal{R}$ is a bisimulation up to context and $\mathcal{R}'$ is a bisimulation we have that $\mathcal{R} \subseteq \sim_S$. □