Deriving Bisimulation Congruences for Conditional Reactive Systems*

Mathias Hülsbusch and Barbara König

Abteilung für Informatik und Angewandte Kognitionswissenschaft,
Universität Duisburg-Essen, Germany

Abstract. We consider conditional reactive systems, a general abstract framework for rewriting, in which reactive systems à la Leifer and Milner are enriched with (nested) application conditions. We study the problem of deriving labelled transitions and bisimulation congruences from a reduction semantics. That is, we synthesize interactions with the environment in order to obtain a compositional semantics. Compared to earlier work we not only address the problem of deriving information about the (minimal) context needed to obtain a full left-hand side and thus be able to perform a reduction, but also generate conditions on the remaining context.

1 Introduction

Given the reduction semantics of a process calculus, it is often hard to define a labelled transition system in such a way that the resulting bisimilarity is a congruence, in order to obtain a compositional semantics. That is, we want to ensure that subsystems can be replaced by behaviourally equivalent subsystems, without changing the overall behaviour. In [15,14] Leifer and Milner showed how to generate such labels in the general framework of reactive systems (an abstract setting for rewriting), using the notion of idem pushout squares (IPOs). This idea has been further developed by other authors [8,20,7]. The underlying idea is simply to label transitions with the minimal context required by a process/term to perform a transition. That is, using the standard example, a CCS process $a.P$ can do a move $a.P \xrightarrow{\_ \mid \overline{a}.Q} P \mid Q$, with the meaning that the context $\_ \mid \overline{a}.Q$ is provided by the environment.

Since the existence and derivation of IPOs is a non-trivial task (due to tricky automorphism problems), Sassone and Sobociński studied the definition of IPOs in a bicategorical setting [20] (so-called groupoidal idem pushouts). Inspired by this line of work we have adapted the approach to label derivation for graph transformation systems [3,4] (borrowed contexts), for which no encompassing theory of labelled transitions and bisimulation was available until this point.

The overall approach works fine in a setting where left-hand sides are simply replaced by right-hand sides, but no extra requirements on the surrounding context are made. Such requirements or application conditions not only require the existence of a left-hand side, but ask for the presence or (more importantly) absence of certain

* Supported by the DFG Project Behaviour-GT.

L. Birkedal (Ed.): FOSSACS 2012, LNCS 7213, pp. 361–375, 2012.
© Springer-Verlag Berlin Heidelberg 2012

components or items in the remaining system. The paper [15] introduced reactive systems together with the notion of reactive contexts, which allow to impose some limited conditions on the context, but reactive contexts are not dependent on the rule and furthermore they have to satisfy the fairly strict requirement that the decomposition of a reactive context always yields two reactive contexts.

In graph transformation there is a well-developed theory of nested application conditions [19,9]. While such conditions do not seem to be strictly necessary for programming formalisms (i.e. the setting where process calculi are often used), they are ubiquitous in specification formalisms, such as languages describing UML model transformations (see for instance [5]). On the other hand, a theory of bisimulation is extremely helpful to show behaviour preservation in model transformation, i.e., to prove that a source model is transformed into a behaviourally equivalent target model. A successful proof strategy is to show that every left-hand side of a transformation rule is bisimilar to the corresponding right-hand side and then rely on the fact that bisimilarity is a congruence. However, the presence of application conditions, especially negative application conditions which destroy monotonicity, are a severe problem, in fact the major problem we had to deal with in a case study where we compared proof techniques for showing semantics preservation in model transformation [11].

Hence we believe that it is important to study the theory of bisimulation congruences in the setting of reactive systems equipped with application conditions, so-called conditional reactive systems which we introduced in [2]. In [18] we already studied bisimulation congruences for graph transformation systems with negative application conditions. The present paper generalizes this in several respects: (i) Generalizing negative application conditions we use nested application conditions; (ii) We work in the more general framework of reactive systems instead of graph transformation systems, which are a specific instance; (iii) Instead of fixing a specific way to derive "minimal" context (via IPOs or borrowed context diagrams as in [4,18]) we define the general (and very simple) notion of representative squares, which leads to the same results (at least for saturated equivalences).

We define a notion of bisimilarity, show that it is a congruence and study it by giving an alternative characterization, which is less practical, but more intuitive. Furthermore we compare with other (different) notions of behavioural equivalence. We will here study saturated bisimilarity (as compared to IPO-bisimilarity), where a transition labelled with a minimal context can be answered by a transition with an arbitrary (possibly non-minimal) context (see [1]).

2 Conditional Reactive Systems

2.1 Reactive Systems with Conditions

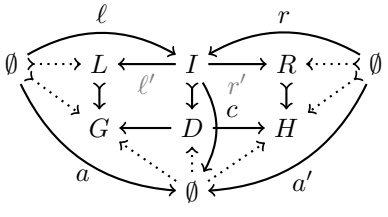
We now define the notion of reactive systems, first introduced in [15,14] for label derivation and the definition of bisimulation congruences.

Definition 1 (Reactive System Rules). *Let $\mathbb{C}$ be a category with a distinguished object 0 (not necessarily initial). A* reactive system rule *is a pair $R = (\ell, r)$ of arrows – called left-hand side and right-hand side respectively – with $\ell, r\colon 0 \to I$ for some object I. Let $\mathcal{R}$ be a set of rules. We say that an arrow $a\colon 0 \to J$* reduces *to $a'\colon 0 \to J$ with the rules in $\mathcal{R}$ (in symbols: $a \rightsquigarrow_{\mathcal{R}} a'$ or simply $a \rightsquigarrow a'$) if there exists a rule $(r, \ell) \in \mathcal{R}$ with $\ell, r\colon 0 \to I$ and an arrow $c\colon I \to J$ such that $a = \ell; c$ and $a' = r; c$.*[1]

In [15] it is additionally required that c is contained in a subcategory of reactive contexts, which, in our setting, will be replaced later by the requirement that c satisfies a given condition.

An important class of reactive systems can be defined over a base category $\mathbb{D}$ which has all pushouts along monos and in which pushouts preserve monos. Then we define as $\mathbf{C} = ILC(\mathbb{D})$ the category which has as objects the objects of $\mathbb{D}$ and as arrows cospans of the form $A \overset{f}{\rightarrowtail} B \overset{g}{\leftarrow} C$ (called *input-linear cospans*, because the left arrow f is a mono), where the middle object is taken up to isomorphism. Composition of cospans is performed via pushouts.

In several of the examples we will use as base category the category $\mathbb{D} = \mathbf{Graph}_{\mathbf{fin}}$ which has finite graphs (with node and edge labels) as objects and graph morphisms as arrows. Reactive systems over $ILC(\mathbf{Graph}_{\mathbf{fin}})$ coincide exactly with DPO graph transformation systems with injective matches (see [21]). Consider the figure below: in DPO rewriting a rule is given by a span $L \overset{\ell'}{\leftarrow} I \overset{r'}{\rightarrow} R$ of arrows in $\mathbf{Graph}_{\mathbf{fin}}$ and a graph G can be transformed into a graph H if we can find a graph D and morphisms such that the inner two squares (drawn with gray arrows) are pushouts. (Intuitively every item of the left-hand side L not contained in I is removed from G by a pushout complement and the right-hand side R is glued to the resulting graph D.) By completing the diagram with empty graphs (corresponding to the distinguished object 0) and the dotted arrows, we obtain two commuting triangles in the cospan category (black arrows), which correspond to the conditions for reactive systems ($a = \ell; c$, $a' = r; c$). The objects that are rewritten in $\mathbf{Graph}_{\mathbf{fin}}$ are the inner objects of the cospans a and a'.



In the rest of this section we will summarize definitions and results from [2]. We will first define conditions, similar to the presentation in [19,9], as tree-like structures, where nodes are annotated with quantifiers and objects and edges are annotated with arrows.

[1] For arrows $f\colon A \to B$ and $g\colon B \to C$ we use the notation $f; g$ for their composition, that is $f; g\colon A \to C$.

Definition 2 (Conditions). *Let $\mathbb{C}$ be a category. A* condition *(in $\mathbb{C}$) is a triple $\mathcal{A} = (A, \mathcal{Q}, S)$ where*

- *A is an object of $\mathbb{C}$ (called the root object of $\mathcal{A}$ or $\mathrm{RO}(\mathcal{A})$),*
- *$\mathcal{Q}$ is a quantifier (either $\forall$ or $\exists$) and*
- *S is a finite set of pairs $(\mathcal{A}', f)$ such that $\mathcal{A}'$ is a condition and $f\colon A \to \mathrm{RO}(\mathcal{A}')$ is a $\mathbb{C}$-arrow.*

The pair $(\mathcal{A}', f)$ will be denoted by $A \xrightarrow{f} \mathcal{A}'$. A condition $\mathcal{A}$ can be viewed as a tree, with $\mathrm{RO}(\mathcal{A})$ as the root and edges labelled with arrows.

Definition 3. *For all conditions $\mathcal{A}$, all objects C and all arrows $c\colon \mathrm{RO}(\mathcal{A}) \to C$ we define a satisfaction relation as follows:*

$c \models (A, \forall, S)$ *iff for every $(A \xrightarrow{f} \mathcal{A}') \in S$ and every arrow $\alpha\colon \mathrm{RO}(\mathcal{A}') \to C$ such that $f; \alpha = c$ we have $\alpha \models \mathcal{A}'$*

$c \models (A, \exists, S)$ *iff for some $(A \xrightarrow{f} \mathcal{A}') \in S$ there is an arrow $\alpha\colon \mathrm{RO}(\mathcal{A}') \to C$ with $f; \alpha = c$ and $\alpha \models \mathcal{A}'$*

In [2] we have shown that if we instantiate $\mathbb{C}$ with $\mathbf{Graph}_{\mathbf{fin}}$ we are equal in expressiveness to first-order logic on graphs. If we instantiate with $ILC(\mathbf{Graph}_{\mathbf{fin}})$ instead we are more expressive, since we are able to express the existence of isolated nodes directly in the logics (even in the presence of infinitely many edge labels).

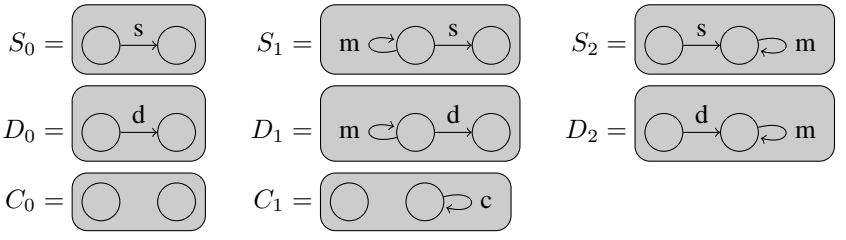
Given two conditions $\mathcal{A}, \mathcal{B}$ with root object C we will in the following write $\mathcal{A} \models \mathcal{B}$ if for every arrow c (with source object C) $c \models \mathcal{A}$ implies $c \models \mathcal{B}$. Furthermore we write $\mathcal{A} \equiv \mathcal{B}$ whenever $\mathcal{A} \models \mathcal{B}$ and $\mathcal{B} \models \mathcal{A}$. Note that for many categories, for instance for $\mathbf{Graph}_{\mathbf{fin}}$ and $ILC(\mathbf{Graph}_{\mathbf{fin}})$ implication and equivalence will be undecidable, a consequence of the undecidability of implication and equivalence in first-order logic.

Now, given a reactive system over $\mathbb{C}$, it is natural to associate conditions with the target object of left-hand and right-hand sides and to interpret them on the contexts.

Definition 4 (Rules with application conditions). *Let $\mathbb{C}$ be a category with a distinguished object 0. A* rule with application condition *is a triple $(\ell, r, \mathcal{B})$ where $\ell, r\colon 0 \to I$ and $\mathcal{B}$ is a condition with root object I. We say that the rule is applicable to $a\colon 0 \to J$ whenever $a = \ell; c$ for some $c\colon I \to J$ such that $c \models \mathcal{B}$. The result of the rule application is $r; c$. Again we denote by $\rightsquigarrow_{\mathcal{R}}$ (or simply $\rightsquigarrow$) the rewriting relation induced by a set $\mathcal{R}$ of rules with application conditions.*

Example 5. As a running example, we will introduce a simple message transportation protocol in a network of nodes, using duplex as well as simplex connections. In both cases, messages can only be transferred from node a to node b, whenever there is no message on b, waiting to be processed. If b however has a buffer (or extra capacity), it is possible to forward the message to b, even if there are already messages at b. To model this, we choose the following graph representation: Nodes in the network are (unlabelled) nodes in the graph; a simplex connection is a directed s-labelled edge; a duplex connection is a a directed edge, labelled d; a message is an m-loop at that node.

If the node has a buffer, we attach a c-loop. As category we choose $ILC(\mathbf{Graph_{fin}})$. To describe the rules, we first give the following graphs:



Using these graphs, we can now give the set of rules:

$$PassS = (\emptyset \to S_1 \leftarrow S_0, \emptyset \to S_2 \leftarrow S_0, Cond_{S_0})$$
$$PassD_1 = (\emptyset \to D_1 \leftarrow D_0, \emptyset \to D_2 \leftarrow D_0, Cond_{D_0})$$
$$PassD_2 = \left(\emptyset \to D_2 \leftarrow D_0, \emptyset \to D_1 \leftarrow D_0, Cond'_{D_0}\right)$$

where conditions are defined as follows, where $true_A = (A, \forall, \emptyset)$:

$$Cond_{S_0} = (S_0, \forall, \{((C_0, \exists, \{(true_{C_0}, C_0 \to C_1 \leftarrow C_0)\}), S_0 \to S_2 \leftarrow C_0)\})$$
$$Cond_{D_0} = (D_0, \forall, \{((C_0, \exists, \{(true_{C_0}, C_0 \to C_1 \leftarrow C_0)\}), D_0 \to D_2 \leftarrow C_0)\})$$
$$Cond'_{D_0} = (D_0, \forall, \{((C_0, \exists, \{(true_{C_0}, C_0 \to C_1 \leftarrow C_0)\}), D_0 \to D_1 \leftarrow \alpha - C_0)\})$$

All graph morphisms (apart from α) are induced by the edge labels and by the positions of the nodes in the images above. The morphism α instead swaps the two nodes, in order to ensure that the buffer is at the left node of the duplex edge (see rule $PassD_2$). The conditions can be read as follows: In *all* cases, where there is a message at the target node, there must *exist* a c-loop at that node to make the rule applicable.

2.2 Representative Squares

We will now define the notion of representative squares (first introduced in [2]), which describe representative ways to close a span of arrows. Such squares are intimately related to idem pushouts [15] or borrowed context diagrams [4].

Definition 6 (Representative class of squares). *A class κ of commuting squares in a category $\mathbb{C}$ is called* representative *if κ satisfies the following property: for every commuting square of $\mathbb{C}$ (such as the one consisting of f_0, f_1, g_0, g_1 on the left) there exists a square in κ (consisting of f_0, f_1, h_0, h_1) and an arrow $h\colon D \to E$ which makes the diagram commute (on the right).*

For two arrows $f_0 \colon A \to B$, $f_1 \colon A \to C$ we denote by $\kappa(f_0, f_1)$ the set of pairs (h_0, h_1) of arrows $h_0 \colon B \to D$ and $h_1 \colon C \to D$ such that f_0, f_1, h_0, f_1 form a representative square in κ.

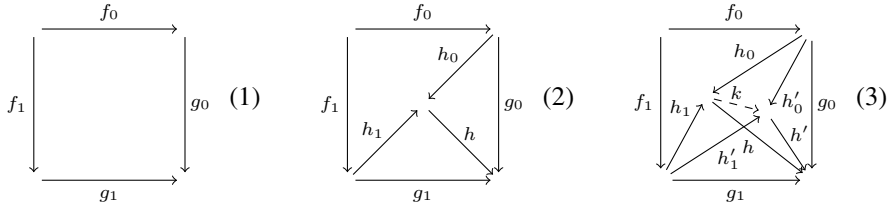
In the following, we fix a representative class κ of squares and we shall call every square in κ *representative*. Also note that the class of all squares of $\mathbb{C}$ is representative. However, we will in the following require that for each pair a, b the set $\kappa(a, b)$ is finite, which means that the constructions described in Section 2.3 are effective since the finiteness of the transformed conditions is preserved. In the following we will discuss possible choices for the class of representative squares.

Pushouts: If we have a category where all pushouts exist, they are the most natural candidate for representative squares and we can define that every class $\kappa(a, b)$ contains only one pair of arrows: a representative pushout. Unfortunately, pushouts do not exist in many categories of interest.

Jointly epi squares: Consider the subcategory of an adhesive category [12] which consists exactly of the mono arrows. Then, the class of all squares a, b, c, d where c, d are jointly epi, is representative.

Idem pushouts: In a category where every commuting square contains an idem pushout (IPO), the IPOs are a representative class of squares. IPOs were introduced in [15] for the purpose of automatically deriving labels, specifying interactions with the environment in reactive systems, and bisimulation congruences.

IPOs are defined as follows: consider a commuting square as shown in 1 below such that $f_0; g_0 = f_1; g_1$. A *relative pushout (RPO)* for this commuting square is a triple h_0, h_1, h satisfying the following two properties: (i) *commutativity:* $f_0; h_0 = f_1; h_1$ and $h_i; h = g_i$ for $i = 0, 1$ (see (2)); (ii) *universality:* for any h'_0, h'_1, h' satisfying $f_0; h'_0 = f_1; h'_1$ and $h'_i; h' = g_i$ there exists a unique mediating arrow k such that $k; h' = h$ and $h_i; k = h'_i$ for $i = 0, 1$ (see (3)).



A commuting square as in Diagram (1) is an *idem pushout (IPO)* if the triple g_0, g_1, id is a relative pushout for the same Diagram (1).

Note that in order to obtain the congruence results of [15] more properties of IPOs than the one of Definition 6 are required.

Borrowed context squares: The category $ILC(\mathbb{D})$, where $\mathbb{D}$ is adhesive, is of special interest for the construction of IPOs, since it integrates nicely with double-pushout (graph) rewriting as shown above. Unfortunately, this category does not have IPOs, due to automorphism problems. One solution is to switch to a bicategorical setting [21,20], another – that we are following here – is to give up the characterizations of the squares via a universal property and concentrate on the relevant properties.

The borrowed context diagrams introduced in [4] (with the extensions introduced in [21]) can be seen as GIPOs (groupoidal pushouts) in a bicategory and they are also representative squares in our sense. A commuting diagram in the cospan category is a borrowed context diagram if and only if it has the form of the diagram on the right, where the left upper square is jointly epi (j.e.), the right lower square is a pullback and the two remaining squares are pushouts: The idea behind these borrowed context diagrams is exactly to close a span of cospans in all minimal representative ways. If we assume that the right leg of all cospans is the identity, then we are in the sub-case treated above, where we consider jointly epi squares.

$$\begin{array}{ccccc} \emptyset & \rightarrowtail & L & \longrightarrow & I \\ \downarrow & \text{j.e} & \downarrow & \text{PO} & \downarrow \\ G & \rightarrowtail & G^{+} & \longleftarrow & C \\ \uparrow & \text{PO} & \uparrow & \text{PB} & \uparrow \\ J & \rightarrowtail & F & \longleftarrow & K \end{array}$$

2.3 Operations on Conditions

We continue to recapitulate concepts and results from [2]: first we define boolean operations on conditions.

Definition 7 (Boolean Operations on Conditions)

Constants: *For an object A define $false_A := (A, \exists, \emptyset)$, $true_A := (A, \forall, \emptyset)$.*
Negation: *For a condition of the form $(A, \mathcal{Q}, S)$ with $\mathcal{Q} \in \{\forall, \exists\}$ we define:*

$$\neg(A, \forall, S) := (A, \exists, \{A \xrightarrow{f} \neg\mathcal{A}' \mid (A \xrightarrow{f} \mathcal{A}') \in S\})$$
$$\neg(A, \exists, S) := (A, \forall, \{A \xrightarrow{f} \neg\mathcal{A}' \mid (A \xrightarrow{f} \mathcal{A}') \in S\})$$

Conjunction und Disjunction: *For two conditions $\mathcal{A}, \mathcal{B}$ with root object C we define:*

$$\mathcal{A} \wedge \mathcal{B} := (C, \forall, \{C \xrightarrow{id_C} \mathcal{A}, C \xrightarrow{id_C} \mathcal{B}\}) \quad \mathcal{A} \vee \mathcal{B} := (C, \exists, \{C \xrightarrow{id_C} \mathcal{A}, C \xrightarrow{id_C} \mathcal{B}\})$$

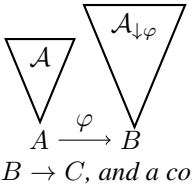
The boolean operations satisfy the usual laws of propositional logic.

One central operation is the shift of a condition along an arrow (see also [17]). Intuitively a shift corresponds to a partial evaluation, where we assume that the arrows on which the condition is to be evaluated are of the form $\varphi; c$ for a fixed φ.

Definition 8 (Shift of a Condition). *Given a fixed set κ of representative squares, the shift $\mathcal{A}_{\downarrow\varphi}$ of a condition $\mathcal{A} = (A, \mathcal{Q}, S)$ along an arrow $\varphi\colon A \to B$ is inductively defined as follows:*

$$\mathcal{A}_{\downarrow\varphi} = (B, \mathcal{Q}, \{(B \xrightarrow{g} \mathcal{A}'_{\downarrow\psi}) \mid (A \xrightarrow{f} \mathcal{A}') \in S,\ (\psi, g) \in \kappa(f, \varphi)\})$$

That is we perform a kind of partial evaluation on the condition. In the following we will visualize conditions by triangles and denote a shift as depicted on the right. We will now give some properties of the shift operator.



Proposition 9 (Shift [2]). *Given two arrows $\varphi\colon A \to B$ and $c\colon B \to C$, and a condition $\mathcal{A}$ with root object A, the following holds:*

$$\varphi; c \models \mathcal{A} \iff c \models \mathcal{A}_{\downarrow\varphi}$$

As a consequence shift satisfies the following laws:

$$\mathcal{A}_{\downarrow\varphi;\psi} \equiv (\mathcal{A}_{\downarrow\varphi})_{\downarrow\psi} \qquad (\mathcal{A} \wedge \mathcal{B})_{\downarrow\varphi} \equiv \mathcal{A}_{\downarrow\varphi} \wedge \mathcal{B}_{\downarrow\varphi} \qquad (\mathcal{A} \vee \mathcal{B})_{\downarrow\varphi} \equiv \mathcal{A}_{\downarrow\varphi} \vee \mathcal{B}_{\downarrow\varphi}$$

2.4 Finiteness Assumptions

In the rest of the paper we will make the following two finiteness assumptions, which hold for many categories, for instance for our example category $ILC(\mathbf{Graph_{fin}})$:

(i) for every pair of arrows a, b $\kappa(a, b)$ is a finite set;
(ii) for two arrows a, c with the same domain, there are only finitely many arrows b such that $a; b = c$.

Note that for (left-linear) cospans over graphs Condition (i) can be enforced by taking borrowed context diagrams as representative squares. Furthermore Condition (ii) holds for cospans (over finite graphs), since the middle object is taken up to isomorphism.

3 Label Derivation and Saturated Bisimilarity

Our aim is now to define behavioural congruences on reactive systems. As observed by many authors before (e.g., in [15]), defining bisimulation relations on the reduction semantics introduced in Section 2 is insufficient for a compositional semantics, since in general the resulting equivalence will not be congruence. It is easy to construct an example involving two arrows a, b which both can not perform a step, whereas a can do a step when put into a suitable context c, but b can not.

Hence, it is necessary to incorporate potential interactions with the environment. We will study this first for reactive systems without conditions, i.e., we will summarize results from [15,1] with the new contribution that the IPO squares used in these papers are replaced by the conceptually simpler representative squares.

We will start by defining *context transitions*, which describe that a component a can do a step (and evolve to b) whenever the environment provides a context f. *Representative transitions* are those context transitions that are generated by representative squares.

Definition 10 (Context transitions, representative transitions). *Let $\mathcal{R}$ be a set of reactive system rules. Let $a\colon 0 \to I$, $f\colon I \to J$, $a'\colon 0 \to J$. We write $a \xrightarrow{f}_C a'$ whenever $a; f \rightsquigarrow a'$, i.e. if there exists a rule $(\ell, r) \in \mathcal{R}$ such that $a' = r; c$ and $a; f = \ell; c$ for some arrow c. Such transitions are called* context transitions. *Furthermore we write $a \xrightarrow{f}_R r; c$ when in addition $(f, c) \in \kappa(a, \ell)$. We call such transitions* representative transitions.

From the definitions it follows immediately that $a \xrightarrow{f}_R a'$ implies $a \xrightarrow{f}_C a'$. Here we are not treating the case of simulating $\xrightarrow{f}_R$-transitions by $\xrightarrow{f}_R$-transitions, which leads to a notion of equivalence analogous to IPO-bisimilarity. We take the position that it should not be observable whether a context is minimal (resp. representative) or not, leading to saturated bisimilarity defined below. Furthermore IPO-bisimilarity admits, to our knowledge, no interesting alternative characterization (as in Section 5), different from saturated bisimilarity. We state here, for completeness, that bisimilarity on representative transition does not necessarily give rise to a congruence: it would be necessary to impose some extra conditions on representative squares (such as the composition and decomposition properties of IPOs established in [15]).

Instead we concentrate on saturated semantics, requiring that every $\xrightarrow{f}_C$-step is matched by a $\xrightarrow{f}_C$-step. Under normal circumstances, this notion is impractical, since the resulting labelled transition system is infinitely branching. However, it can be easily shown that saturated semantics coincides with semi-saturated bisimilarity where every $\xrightarrow{f}_R$-step is matched by a $\xrightarrow{f}_C$-step. Since in many application areas of interest we can guarantee that $\kappa(a,b)$ is finite for every pair of arrows a, b, the transition relation $\xrightarrow{f}_R$ is finitely branching and hence amenable to mechanization.

Definition 11 (Saturated/semi-saturated bisimilarity). *Let $\mathcal{R}$ be a set of reactive system rules. Let R be a symmetric relation, which relates pairs of arrows a, b with source object 0 and identical target object. We say that R is a* saturated bisimulation *if whenever $a\,R\,b$ and $a \xrightarrow{f}_C a'$, then there exists b' such that $b \xrightarrow{f}_C b'$ and $a'\,R\,b'$. Two arrows a, b are called* saturated bisimilar *($a \sim_{SAT} b$) whenever there exists a saturated bisimulation R with $a\,R\,b$.*

A relation R is a semi-saturated bisimulation *if whener $a\,R\,b$ and $a \xrightarrow{f}_R a'$, then there exists b' such that $b \xrightarrow{f}_C b'$ and $a'\,R\,b'$. Two arrows a, b are called* semi-saturated bisimilar *($a \sim b$) whenever there exists a semi-saturated bisimulation R with $a\,R\,b$.*

Theorem 12 ([1])

1. *Saturated and semi-saturated bisimilarity coincide, i.e., for two arrows a, b we have $a \sim_{SAT} 0b \iff a \sim b$.*
2. *Furthermore saturated bisimilarity is a congruence, i.e., whenever we have $a, b\colon 0 \to I$ with $a \sim_{SAT} b$ and $c\colon I \to J$, then $a;c \sim_{SAT} b;c$.*
3. *Finally, saturated bisimularity is the coarsest bisimulation on $\rightsquigarrow$ that is also a congruence.*

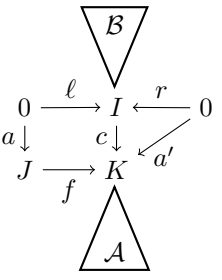
The theorem is due to [1], however observe that we here employ representative squares, different from the IPOs used in [1]. The main aim of this paper is to extend the theorem to deal with conditions and to establish an analogous (but more complex) result in the setting of conditional reactive systems, which generalizes Theorem 12. For this we will first define a notion of labelled transitions and of (semi-)saturated bisimilarity for conditional reactive systems.

4 Bisimilarity for Conditional Reactive Systems

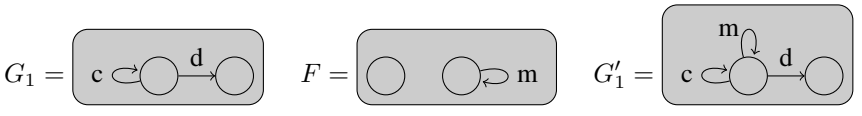
We will first define notions of context and representative transitions in the presence of (application) conditions.

Definition 13 (Context transitions, representative transitions (with application conditions)). *Let $\mathcal{R}$ be a set of reactive system rules with conditions. Let $a\colon 0 \to J$, $f\colon J \to K$, $a'\colon 0 \to K$ and $\mathcal{A}$ be a condition with root object K. We write $a \xrightarrow{f,\mathcal{A}}_C a'$ whenever there exists a rule $(\ell, r, \mathcal{B}) \in \mathcal{R}$ such that $a; f = \ell; c$, $a' = r; c$ and $\mathcal{A} \models \mathcal{B}_{\downarrow c}$ for some arrow c. Furthermore we write $a \xrightarrow{f,\mathcal{A}}_R a'$ when in addition $(f, c) \in \kappa(a, \ell)$ and $\mathcal{A} = \mathcal{B}_{\downarrow c}$.*

The figure to the right visualizes this situation. Note that in the case of a context transition the square simply commutes, whereas in case of a representative transition it must be a representative square. Intuitively $a \xrightarrow{f,\mathcal{A}}_C a'$ means that a can be transformed to a' whenever the environment provides or "borrows" f and furthermore $a; f$ is put into a *passive* context (i.e. a context not participating in the reduction) satisfying $\mathcal{A}$. On the other hand $a \xrightarrow{f,\mathcal{A}}_R a'$ means that the context f is not arbitrary, but one of the representative contexts required for the move. In addition $\mathcal{A}$ is the weakest possible requirement on the context. Clearly $a \xrightarrow{f,\mathcal{A}}_R a'$ implies $a \xrightarrow{f,\mathcal{A}}_C a'$.



Example 14. To illustrate one label derivation, in this case for a representative transition (using borrowed context diagrams as the class of representative squares), we need the following graphs:



We consider rule $PassD_2$ (see Example 5) and the arrow $a = \emptyset \to G_1 \leftarrow C_0$ and take a borrowed context diagram where the two graphs G_1, D_2 overlap on the d-edge (there are other possible representative squares). In this case we obtain as label f (which describes the context to be borrowed) a cospan with F as the center graph (i.e., we borrow a message on the right node). Furthermore we derive the condition $\mathcal{A} = (Cond'_{D_0})_{\downarrow c} \equiv true_{C_0}$. This reflects the fact, that a message at the right node can always be transferred to the left node, since this node has unbounded capacity. In addition, the resulting cospan a' has graph G_1' as center graph.

Definition 15 (Saturated/semi-saturated bisimilarity (with application conditions)). *Let $\mathcal{R}$ be a set of reactive system rules with conditions. Let R be a symmetric relation, which relates pairs of arrows a, b with source object 0 and identical target object. We say that R is a* saturated bisimulation *if whenever $a\,R\,b$ and $a \xrightarrow{f,\mathcal{A}}_C a'$, then there exist arrows $b'_1, \ldots, b'_n$ and conditions $\mathcal{A}_1, \ldots, \mathcal{A}_n$ such that $b \xrightarrow{f,\mathcal{A}_i}_C b'_i$ with $a'\,R\,b'_i$ for all $i \in \{1, \ldots, n\}$ and $\mathcal{A} \models \bigvee_{i=1}^n \mathcal{A}_i$. Two arrows a, b are called* saturated bisimilar *($a \sim_{SAT} b$) whenever there exists a saturated bisimulation R with $a\,R\,b$.*

A relation R is a semi-saturated bisimulation *if in the definition above $a \xrightarrow{f,\mathcal{A}}_C a'$ is replaced by $a \xrightarrow{f,\mathcal{A}}_R a'$. Two arrows a, b are called* semi-saturated bisimilar *($a \sim b$) whenever there exists a saturated bisimulation R with $a\,R\,b$.*

We will motivate why several answering moves are allowed: whenever the first partner makes a move, the second partner may have the chance to simulate this move, but the used rule may depend on the context. For instance, assume that we want to show that a single A-edge is bisimilar to a single B-edge under the presence of the following rules: the A-edge can be (unconditionally) deleted, but there are two different deletion rules

for the B-edge, one that requires the presence of C edge and another that forbids the presence of a C-edge.

Observe that a move of a with condition $\mathcal{A} \equiv \mathit{false}$ can always be mimicked by b by doing nothing, that is, an empty set of answering moves is allowed.

Note also, that in the definition of semi-saturated bisimulation above, in the answering moves of the form $b \xrightarrow{f,\mathcal{A}_i}_C b'_i$ we can always assume that $\mathcal{A}_i$ is the weakest possible condition, i.e., $\mathcal{A}_i$ is obtained by shifting the rule condition over the context and is hence of the form $\mathcal{A} = \mathcal{B}_{\downarrow c}$ for some $\mathcal{B}, c$.

Furthermore it is straightforward to show that the relations $\sim, \sim_{SAT}$ themselves are also (semi-)saturated bisimulations.

Lemma 16. *Let $a\colon 0 \to I$, $f\colon I \to J$, $a'\colon 0 \to J$ and $\mathcal{A}$ be a condition with root object J. In addition let d, c', f' be arrows with $d; f' = f; c'$. Then $a \xrightarrow{f,\mathcal{A}}_C a'$ implies $a; d \xrightarrow{f',\mathcal{A}_{\downarrow c'}}_C a'; c'$. As a special case (if $d = id$ and $f' = f; c'$) we obtain that $a \xrightarrow{f,\mathcal{A}}_C b$ implies $a \xrightarrow{f;c',\mathcal{A}_{\downarrow c'}}_C a'; c'$.*

We will first show that semi-saturated bisimilarity is a congruence. It is easier to show that saturated bisimilarity is a congruence (and the two coincide anyway), but we need this result for the proof of Theorem 18.

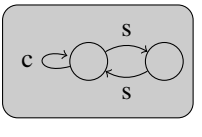
Theorem 17 (Congruence). *Semi-saturated bisimilarity is a congruence, i.e., whenever we have $a, b\colon 0 \to I$ with $a \sim b$ and $c\colon I \to J$, then $a; c \sim b; c$.*

Note that we would not obtain a congruence result if we omitted conditions from the labels: again, it is easy to think of an example with two arrows a, b where a contains a left-hand side, but can not perform the step, since the condition requires the presence of another item. On the other hand, b can not perform a reduction at all. Then both would be bisimilar if we do not take conditions into account, but by putting them into a context containing the item required by a we would obtain a pair of non-bisimilar arrows. As a next step we show that saturated and semi-saturated bisimilarity coincide.

Theorem 18 (Saturated vs. semi-saturated bisimilarity). *Saturated and semi-saturated bisimilarity coincide, i.e., for two arrows a, b we have:*

$$a \sim_{SAT} b \iff a \sim b.$$

Example 19. Using the techniques presented in this paper, we can now show that the cospans $a = \emptyset \to G_1 \leftarrow C_0$ (see Example 14) and $b = \emptyset \to G_2 \leftarrow C_0$ (for G_2 shown below) are (saturated) bisimular.

$G_2 =$ 

It is also possible to prove that a duplex connection can be mimicked by two simplex connections and vice versa, if the c-loops are placed alike in both cases.

5 An Alternative Characterization

In order to characterize the equivalence that we have obtained, we will give an alternative definition as the coarsest bisimilarity for a certain transition system that is also a congruence.

Definition 20 (Environment transition). *Let $\mathcal{R}$ be a set of reactive system rules with conditions. Let $a\colon 0 \to I$, $a'\colon 0 \to I$, $d\colon I \to J$. We write $a \overset{d}{\rightsquigarrow} a'$ whenever there exists a rule $(\ell, r, \mathcal{B}) \in \mathcal{R}$ and an arrow c such that $a = \ell; c$, $a' = r; c$ and $c; d \models \mathcal{B}$.*

Note that $\overset{d}{\rightsquigarrow}$-transitions are infinitely branching. Intuitively a $\overset{d}{\rightsquigarrow}$-transition is possible if a rule can be applied under a *passive context* d that does not participate in the reduction. $a \overset{d}{\rightsquigarrow} a'$ implies $a; d \rightsquigarrow a'; d$, but the reverse does not necessarily hold, since the right-hand reduction may consume (and recreate) parts of d.

Definition 21 (Environment bisimilarity). *Let $\mathcal{R}$ be a set of reactive system rules with conditions. Let R be a symmetric relation, which relates pairs of arrows a, b with source object 0 and identical target object. We say that R is an* environment bisimulation *if whenever $a \overset{d}{\rightsquigarrow} a'$, then $b \overset{d}{\rightsquigarrow} b'$ and $a' \,R\, b'$ for some arrow b'. We denote by $\sim^e$ (*environment congruence*) the coarsest relation that is a congruence and an environment bisimulation.*

In order to show that saturated bisimilarity and environment congruence coincide, we first need the following lemma.

Lemma 22. *Let $a\colon 0 \to I, f\colon I \to J, a'\colon 0 \to J, d\colon J \to K$. Let $\mathcal{A}$ be a condition with root object J such that $a \overset{f,\mathcal{A}}{\Rightarrow}_C a'$ and $d \models \mathcal{A}$. Then we have $a; f \overset{d}{\rightsquigarrow} a'$.*

In the other direction if $a; f \overset{d}{\rightsquigarrow} a'$, then there exists a condition $\mathcal{A}$ with $a \overset{f,\mathcal{A}}{\Rightarrow}_C a'$ and $d \models \mathcal{A}$. More precisely there exists a rule $(\ell, r, \mathcal{B})$ and an arrow c with $a; f = \ell; c$, $a' = r; c$ and $\mathcal{A} = \mathcal{B}_{\downarrow c}$.

We will now show that the natural, but impractical, notion of environment congruence is equivalent to (semi-)saturated bisimilarity, which is more amenable to mechanization.

Theorem 23 (Saturated bisimilarity vs. environment congruence). *Saturated bisimilarity (and hence also semi-saturated bisimilarity) and environment congruence coincide, i.e., for two arrows a, b we have $a \sim_{SAT} b \iff a \sim^e b$.*

Note that the notion of environment congruence and hence also of (semi-)saturated bisimilarity is entirely independent on the notion of representative squares. This allows to choose any suitable class of representative squares in implementations or proofs.

Our new results generalize the results of Section 3: if we assume that all application conditions for rules are true, the condition $\mathcal{A}$ in a transition will also always be true and hence (semi-)saturated bisimilarity with conditions specializes to (semi-)saturated bisimilarity without conditions. Furthermore, whenever an arrow a can do a $\rightsquigarrow$-step without conditions, it can do an environment transition $\overset{d}{\rightsquigarrow}$ for *every* (composable) arrow d. Hence environment bisimilarity is simply the coarsest relation which is a congruence and a bisimilarity.

6 Comparison

There is another equivalence to which we could naturally compare: the coarsest congruence that is a bisimulation wrt. $\overset{id}{\rightsquigarrow}$-steps (equivalently $\rightsquigarrow$-steps), i.e., the bisimulation where we consider only reduction steps. Clearly, (semi-)saturated bisimilarity ($\sim$) is finer than this equivalence, here we show that the inclusion is strict by means of two examples. In both cases we work in the category $ILC(\mathbf{Graph_{fin}})$ and assume node-labelled graphs.

Example 24. Consider two rules: one replaces a node labelled B by a node labelled D, but only if an A-labelled node is present (rule R_1). The other (rule R_2) replaces two nodes labelled A and C by two nodes labelled A and D (i.e., here node A is deleted and recreated).

Now we consider as cospans $b = \emptyset \to G_B \leftarrow \emptyset$ the graph with two empty interfaces consisting simply of a B-labelled node, similarly we define $c = \emptyset \to G_C \leftarrow \emptyset$. Clearly b, c are not saturated bisimilar: b can perform a transition $b \overset{id,\mathcal{A}}{\to}_R$ where $\mathcal{A}$ requires the presence of an A-node, whereas c can not answer with context id. In other words b can reduce under an empty environment, while c can not. Both graphs differ wrt. their consumption of items provided by the environment.

On the other hand if we close the pair (b, c) wrt. all possible contexts (and take the union with the identity relation) we obtain a congruence that is also a bisimulation. It is a congruence by definition and it is a bisimilarity: if either the B-node or the C-node is replaced by a D-node, then an A-node is present in the environment, which means that the step can be mimicked by the respective partner.

The two equivalences also differ concerning the notion of an environment which may change (independently of the system rules).

Example 25. Now keep rule R_1 above, but replace R_2 by rule R_3 where a C-labelled node is replaced by an F-labelled node if an A-node is present. Furthermore imagine another rule (rule R_4), which replaces a node labelled D by a node labelled E, but only if *no* A-labelled node is present. Now use R_1, R_3 and R_4 as the only rules.

Again, consider the two cospans b, c above. They are not saturated bisimilar, since b can do two environment steps (where the first step demands the presence of an A-node and the second its absence), whereas c can do only one. This means that we take into account that some outside entity might change the environment without using the predefined reduction rules.

However, if we close the pair (b, c) wrt. all possible contexts and furthermore close the pair of graphs consisting of a single D-node and a single F-node under all contexts containing A, we obtain a bisimulation that is also a congruence. It is a bisimulation since neither b nor c can do a step (if no A-node is present), or – if they can do a step – we are sure that there exists an A in the environment. However, in this case we end up in the second part of the bisimulation, where no further moves are possible.

This means that our notion of equivalence is motivated by the following philosophy: we can observe whether an item is consumed or simply required by a rule. And second, we do not trust that the environmental context is only influenced by the reduction rules, but assume that there could be other modifications by an outside entity.

7 Conclusion

We have shown how to generate bisimulation congruences in a rule-based formalism with application conditions. We are aware of only few papers that explicitly add conditions or restrict contexts in defining behavioural equivalences. In [10] a symbolic bisimulation is studied, where bisimulation is a parametrized relation: two processes may not always be bisimilar, but only bisimilar under certain restrictions. A similar parametrization is investigated in [6], however not in the context of process algebra. Furthermore in [13] a form of context-dependent bisimulation for processes is introduced.

We believe that this paper is a first step towards a better understanding of context dependency. Several open problems still remain, for instance: is there a different way to characterize the equivalence of Section 6 (the coarsest congruence which is a bisimilarity wrt. reduction rules)?

One could also study an even coarser relation: the coarsest congruence that is contained in bisimilarity. The problem with this equivalence is that it does not naturally admit a coinductive definition. We believe that it should be possible to adapt ideas from [10] and to parametrize the bisimulation over contexts.

Considering the results of this paper we think that saturated bisimilarity (which we studied here) is more stable than the notion of IPO-bisimilarity, since it is independent on the specific choice of representative squares. On the other hand, the "right" notion of bisimilarity for process calculi is sometimes IPO-bisimilarity and it would be worthwile studying how this notion of bisimilarity integrates into the proposed framework. On the more speculative side, conditions as presented here could be used to define barbs [16] and hence represent a means to fine-tune the equivalence.

Our main application area is graph transformation, but it should be worthwile to study conditional reactive systems in different categories, for instance using a Lawvere theory (i.e., a category with terms as arrows).

Furthermore we plan to continue our work in [11] and to conduct further case studies in the area of the verification of model transformations. In order to obtain a practically usable method, we have to integrate up-to techniques, but we expect that this can be done without problems. Since label derivation is complex and can not easily be done manually, we however require an implemention, which we have already started to developed. In order to come to terms with the undecidability of implication we will either restrict to simpler conditions or use an approximative algorithm as in [17].

Acknowledgements. We would like to thank Vladimiro Sassone for asking an inspiring question after a talk. Furthermore we are grateful to Filippo Bonchi for valuable insights concerning the "right" notion of bisimilarity.

References

1. Bonchi, F., König, B., Montanari, U.: Saturated semantics for reactive systems. In: Proc. of LICS 2006, pp. 69–80. IEEE (2006)
2. Bruggink, H.J.S., Cauderlier, R., König, B., Hülsbusch, M.: Conditional reactive systems. In: Proc. of FSTTCS 2011. LIPICS, vol. 13. Schloss Dagstuhl – Leibniz Center for Informatics (2011)

3. Ehrig, H., König, B.: Deriving Bisimulation Congruences in the DPO Approach to Graph Rewriting. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 151–166. Springer, Heidelberg (2004)
4. 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)
5. Engels, G., Kleppe, A., Rensink, A., Semenyak, M., Soltenborn, C., Wehrheim, H.: From UML Activities to TAAL - Towards Behaviour-Preserving Model Transformations. In: Schieferdecker, I., Hartman, A. (eds.) ECMDA-FA 2008. LNCS, vol. 5095, pp. 94–109. Springer, Heidelberg (2008)
6. Fitting, M.: Bisimulations and boolean vectors. In: Advances in Modal Logic, vol. 4, pp. 1–29. World Scientific Publishing (2002)
7. Di Gianantonio, P., Honsell, F., Lenisa, M.: RPO, second-order contexts, and lambda-calculus. Logical Methods in Computer Science 5(3) (2009)
8. Grohmann, D., Miculan, M.: Reactive Systems Over Directed Bigraphs. In: Caires, L., Vasconcelos, V.T. (eds.) CONCUR 2007. LNCS, vol. 4703, pp. 380–394. Springer, Heidelberg (2007)
9. Habel, A., Pennemann, K.-H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Stuctures in Computer Science 19, 245–296 (2009)
10. Hennessy, M., Lin, H.: Symbolic bisimulations. Theoretical Computer Science 138(2), 353–389 (1995)
11. Hülsbusch, M., König, B., Rensink, A., Semenyak, M., Soltenborn, C., Wehrheim, H.: Showing Full Semantics Preservation in Model Transformation - A Comparison of Techniques. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 183–198. Springer, Heidelberg (2010)
12. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. RAIRO – Theoretical Informatics and Applications 39(3) (2005)
13. Guldstrand Larsen, K.: Context-Dependent Bisimulation between Processes. PhD thesis, University of Edinburgh (1986)
14. Leifer, J.J.: Operational congruences for reactive systems. PhD thesis, University of Cambridge Computer Laboratory (September 2001)
15. Leifer, J.J., Milner, R.: Deriving Bisimulation Congruences for Reactive Systems. In: Palamidessi, C. (ed.) CONCUR 2000. LNCS, vol. 1877, pp. 243–258. Springer, Heidelberg (2000)
16. Milner, R., Sangiorgi, D.: Barbed Bisimulation. In: Kuich, W. (ed.) ICALP 1992. LNCS, vol. 623, pp. 685–695. Springer, Heidelberg (1992)
17. Pennemann, K.-H.: Development of Correct Graph Transformation Systems. PhD thesis, Universität Oldenburg (May 2009)
18. Rangel, G., König, B., Ehrig, H.: Deriving Bisimulation Congruences in the Presence of Negative Application Conditions. In: Amadio, R.M. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 413–427. Springer, Heidelberg (2008)
19. Rensink, A.: Representing First-Order Logic Using Graphs. In: Ehrig, H., Engels, G., Parisi-Presicce, F., Rozenberg, G. (eds.) ICGT 2004. LNCS, vol. 3256, pp. 319–335. Springer, Heidelberg (2004)
20. Sassone, V., Sobociński, P.: Reactive systems over cospans. In: Proc. of LICS 2005, pp. 311–320. IEEE (2005)
21. Sobociński, P.: Deriving process congruences from reaction rules. PhD thesis, Department of Computer Science. University of Aarhus (2004)