CONDITIONAL BISIMILARITY FOR REACTIVE SYSTEMS

MATHIAS HÜLSBUSCH [a], BARBARA KÖNIG [a], SEBASTIAN KÜPPER [b], AND LARA STOLTENOW [a]

[a] Universität Duisburg-Essen, Germany
e-mail address: {barbara_koenig,lara.stoltenow}@uni-due.de

[b] FernUniversität in Hagen, Germany
e-mail address: sebastian.kuepper@fernuni-hagen.de

ABSTRACT. Reactive systems à la Leifer and Milner, an abstract categorical framework for rewriting, provide a suitable framework for deriving bisimulation congruences. This is done by synthesizing interactions with the environment in order to obtain a compositional semantics.

We enrich the notion of reactive systems by conditions on two levels: first, as in earlier work, we consider rules enriched with application conditions and second, we investigate the notion of conditional bisimilarity. Conditional bisimilarity allows us to say that two system states are bisimilar provided that the environment satisfies a given condition.

We present several equivalent definitions of conditional bisimilarity, including one that is useful for concrete proofs and that employs an up-to-context technique, and we compare with related behavioural equivalences. We consider examples based on DPO graph rewriting, an instantiation of reactive systems.

1. INTRODUCTION

Behavioural equivalences, such as bisimilarity, relate system states with the same behaviour. Here, we are in particular interested in conditional bisimilarity, which allows us to say that two states a, b are bisimilar provided that the environment satisfies a condition $\mathcal{C}$. Work on such conditional bisimulations appears somewhat scattered in the literature (see for instance [Lar86, HL95, Fit02, BKKS17]). They also play a role in the setting of featured transition systems for modelling software product lines [CCP+12], where the behaviour of many products is specified in a single transition system. In this setting it is possible to state that two states are bisimilar for certain products, but not for others.

We believe that conditional notions of behavioural equivalence are worthy of further study. In practice it may easily happen that two sub-systems are only ever used in restricted environments and it is too much to ask that they behave equivalently under all possible contexts. Furthermore, instead of giving a simple yes/no-answer, bisimulation checks can answer in a more fine-grained way, specifying conditions which ensure bisimilarity.

We state our results in a very general setting: reactive systems à la Leifer and Milner [LM00], a categorical abstract framework for rewriting, which provides a suitable framework for deriving bisimulation congruences. In particular, this framework allows to

synthesize labelled transitions from plain reaction rules, such that the resulting bisimilarity is automatically a congruence. Intuitively, the label is the minimal context that has to be borrowed from the environment in order to trigger a reduction. (Transitions labelled with such a minimal context will be called representative steps in the sequel. They are related to the idem pushout steps of [LM00].) Here, we rely on the notion of saturated bisimilarity introduced in [BKM06] and we consider reactive system rules with application conditions, generalizing [HK12].

Important instances of reactive systems are process calculi with contextualization, bigraphs [JM03] and double-pushout graph rewriting [CMR+97], or in general rewriting in adhesive categories [LS05]. Hence we can use our results to reason about process calculi as well as dynamically evolving graphs and networks for various different types of graphs (directed or undirected graphs, node- or edge-labelled graphs, hypergraphs, etc.).

Our contributions in this paper can be summarized as follows:

- We define the notion of conditional bisimilarity, in fact we provide three equivalent definitions: two notions are derived from saturated bisimilarity, where a context step (or a representative step) can be mimicked by several answering steps. Third, we compare with the notion of conditional environment congruence, which is based on the idea of annotating transitions with passive environments enabling a step.
- Conditional bisimulation relations tend to be very large — often infinite in size. To obtain possibly substantial reductions of proof obligations in a bisimulation proof, we propose an up-to context technique (for up-to techniques and their history see [PS19]). In particular, it can replace an infinite conditional bisimulation relation by a possibly finite bisimulation up-to context, which provides a witness for bisimilarity. We also view our up-to technique in a general lattice-theoretical setting and prove the compatibility property [PS11b], which not only implies soundness of the technique, but also allows it to be composed with other (compatible) up-to techniques.
- We use the notion of representative steps in order to obtain finitely branching transition systems, further reducing proof obligations.
- We compare conditional bisimilarity with related notions of behavioural equivalence.
- To illustrate our concepts, we work out a small case study in the context of double-pushout graph rewriting, where we model message passing over reliable and unreliable channels.

The article is structured as follows: First, in Section 2 we introduce the fundamental ideas for reactive systems without conditions, including all preliminary definitions and techniques developed for reactive systems relevant to our work. In Section 3, we consider the refinement to conditional reactive systems, before we turn towards our main contribution in Section 4, which is conditional bisimulation and its up-to variant in Section 5. In Section 6 we give an alternative characterization of conditional bisimilarity and compare to related notions of behavioural equivalence and we conclude in Section 7.

2. Reactive Systems

We denote the composition of arrows $f\colon A \to B$, $g\colon B \to C$ by $f;g\colon A \to C$. Usually written $g \circ f$, we chose $f;g$ to better match the reading order of the diagram $A \xrightarrow{f} B \xrightarrow{g} C = A \xrightarrow{f;g} C$.

2.1. **Reactive Systems without Conditions.** We now define reactive systems, which were introduced in [LM00] and extended in [HK12] with application conditions for rules. We initially only look at reactive systems without conditions. Conditions and the definition of reactive systems with conditions will be introduced later, in Section 3.

Definition 2.1 (Reactive system rules, reaction)**.** Let $\mathbf{C}$ be a category with a distinguished object 0 (not necessarily initial). A *rule* is a pair (ℓ, r) of arrows $\ell, r\colon 0 \to I$ (called left-hand side and right-hand side). A *reactive system* is a set of rules.

Let $\mathcal{S}$ be a reactive system and $a, a'\colon 0 \to J$ be arrows. We say that *a reduces to a'* ($a \rightsquigarrow a'$) whenever there exists a rule $(\ell, r) \in \mathcal{S}$ with $\ell, r\colon 0 \to I$ and an arrow $c\colon I \to J$ (the *reactive context*) such that $a = \ell;c$ and $a' = r;c$.

Using a notation closer to process calculi, we could write $C[P] \rightsquigarrow C[P']$ whenever there is a reaction rule $P \to P'$ and a context $C[\_]$. Fixing a distinguished object 0 means that we consider only ground reaction rules (as opposed to the open reactive systems investigated in [KSS05]).

An important instance are reactive systems where the arrows are cospans in a base category $\mathbf{D}$ with pushouts [SS05, Sob04]. A *cospan* is a pair of arrows $f_L\colon A \to C$, $f_R\colon B \to C$. A cospan is *input linear* if its left arrow f_L is mono.
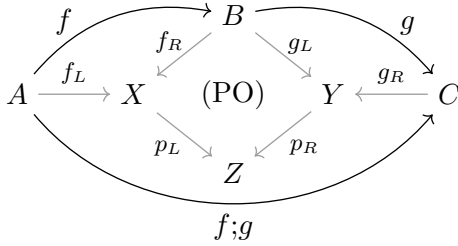

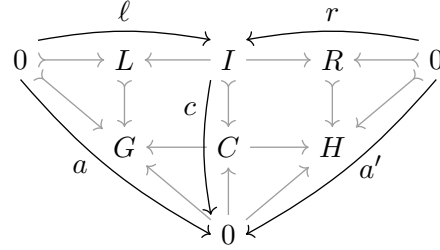Figure 1: Composition of cospans f and g is done via pushouts

Figure 2: Double-pushout graph transformation as reactive system steps

Two cospans $f\colon A \xrightarrow{f_L} X \xleftarrow{f_R} B$, $g\colon B \xrightarrow{g_L} Y \xleftarrow{g_R} C$ are composed by taking the pushout (p_L, p_R) of (f_R, g_L) as shown in Figure 1. The result is the cospan $f;g\colon A \xrightarrow{f_L;p_L} Z \xleftarrow{g_R;p_R} C$, where Z is the pushout object of f_R, g_L. For adhesive categories [LS05] (see Appendix A), the composition of input linear cospans again yields an input linear cospan (by applying [LS05, Lemma 4.2] to the cospan composition diagram). Given an adhesive category $\mathbf{D}$, $ILC(\mathbf{D})$ is the category where the objects are the objects of $\mathbf{D}$, the arrows $f\colon A \to C$ are input linear cospans $f\colon A \to B \leftarrow C$ of $\mathbf{D}$ and composition is performed via pushouts as described above. We see an arrow $f\colon A \to C$ of $ILC(\mathbf{D})$ as an object B of $\mathbf{D}$ equipped with two interfaces A, C and corresponding arrows f_L, f_R to relate the interfaces to B, and composition glues the inner objects of two cospans via their common interface. Input linearity is useful since we rely on adhesive categories where pushouts along monos are well-behaved. In particular, they always exist and form Van Kampen squares (see Appendix A), the latter being a requirement for borrowed context diagrams (Subsection 2.3).

In this article, as a running example we consider the category $\mathbf{Graph_{fin}}$, which has finite graphs (we use directed multigraphs with node and edge labels) as objects and total graph morphisms (functions that map nodes and edges of one graph to another, with the

edge map being consistent to the node map) as arrows. In $\mathbf{Graph_{fin}}$, monos are exactly the injective graph morphisms. We then use reactive systems over $ILC(\mathbf{Graph_{fin}})$ (input-linear cospans of graphs), i.e. we rewrite graphs with interfaces. If the distinguished object 0 is the empty graph (the initial object of $\mathbf{Graph_{fin}}$), such reactive systems coincide [SS05] with the well-known *double pushout (DPO) graph transformation* approach [EPS73, HMP01] when used with injective matches. As shown in Figure 2, a DPO rewrite step $G \Rightarrow H$ can be expressed as a reactive system reaction $a \rightsquigarrow a'$ where the pushouts of the DPO step are obtained from cospan compositions $\ell;c$ and $r;c$.

2.2. Deriving Bisimulation Congruences.

The reduction relation $\rightsquigarrow$ generates an unlabelled transition system, where the states are the reactive agents (in our example, graphs). Note that bisimilarity on this transition system only checks whether any reaction is possible: for two bisimilar agents, it is not required that the same rule is used in their reactions, or even that the reaction is applied at the same position.

A disadvantage of bisimilarity on $\rightsquigarrow$ is that it usually is not a congruence: it is easy to construct an example where neither a nor b can perform a step since no complete left-hand side is present (hence a would be bisimilar to b). However, by adding a suitable context f, $a;f$ could contain a full left-hand side and can reduce, whereas $b;f$ can not.

Therefore, to check whether two components can be exchanged, they have to be combined with every possible context and bisimilarity has to be shown for each.

In order to obtain a congruence, we can resort to defining bisimulation on labelled transitions, using as labels the additional contexts that allow an agent to react [LM00, HK12].

Definition 2.2 (Context step (without conditions) [HK12]). Let $\mathcal{S}$ be a reactive system and $a: 0 \to J$, $f: J \to K$, $a': 0 \to K$ be arrows. We write $a \xrightarrow{f}_C a'$ whenever $a;f \rightsquigarrow a'$ (i.e. there exists a rule $(\ell, r) \in \mathcal{S}$ and an arrow c such that $a;f = \ell;c$, $a' = r;c$). Such steps are called *context steps*.

$$\begin{array}{ccc} 0 & \xrightarrow{\ell} I \xleftarrow{r} & 0 \\ a\downarrow & f \;\; c\downarrow \;\; \swarrow a' \\ J & \xrightarrow{} K \end{array}$$

Intuitively we have to find a context f for the arrow a (which we want to rewrite) such that we obtain the left-hand side ℓ plus some additional context c. The name *context step* stems from the fact that a might not be able to do a reaction on its own, but requires an additional context f. This can be seen in the following example:
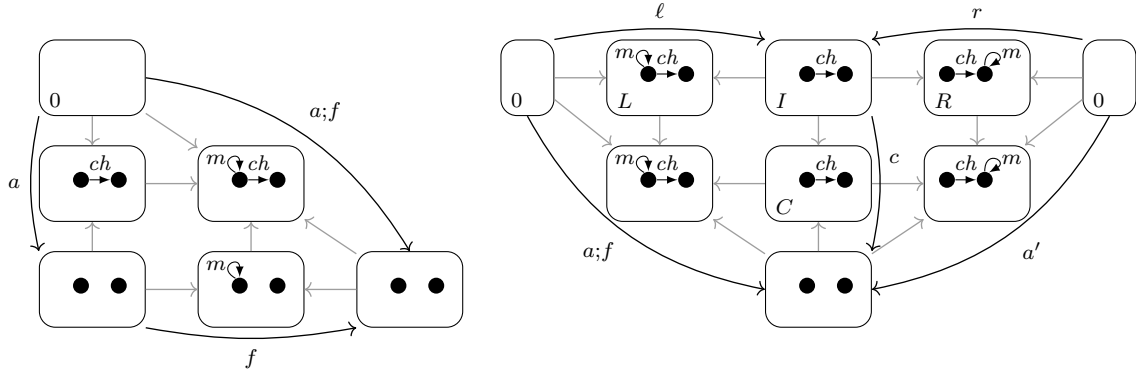
Example 2.3 (Context step (without conditions)). Consider the following reactive system over $ILC(\mathbf{Graph_{fin}})$, i.e. all arrows (such as $\ell, r, \dots$) are input-linear cospans of graphs that represent graphs with interfaces. We model a network of nodes that pass messages (represented by m-loops) over communication channels (represented by ch-edges). The transmission of a message from the left node to the right node can be represented with the following rule:
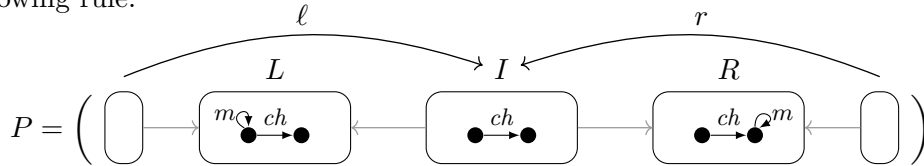
All graph morphisms are induced by edge labels and position of nodes, i.e. the left node is always mapped to the left node.

We can observe that a channel by itself ($a = \emptyset \to \bullet \xrightarrow{ch} \bullet \leftarrow \bullet\;\bullet$) cannot do a reaction, since there is no message to be transferred. However, if a message on the left node

is borrowed ($f = \bullet\,\bullet \to \boxed{m\,\bullet\,\bullet\,\bullet} \leftarrow \bullet\,\bullet$), we obtain $a;f$ (Figure 3a), to which the example rule can be applied (Figure 3b). As a result, we obtain the context step $a \xrightarrow{f}_C a'$ or $(\emptyset \to \bullet\,\underline{ch}\,\bullet \leftarrow \bullet\,\bullet) \xrightarrow{(\bullet\,\bullet \to \boxed{m\,\bullet\,\bullet\,\bullet} \leftarrow \bullet\,\bullet)}_C (\emptyset \to \bullet\,\underline{ch}\,\bullet\,m \leftarrow \bullet\,\bullet)$.   ⌟

(a) A channel a together with borrowed context f, resulting in $a;f$.

(b) $a;f$ can now do a reaction to $a' = r;c$.

Figure 3: Visualization of the context step described in Example 2.3.

A bisimulation relation over $\to_C$ is called *saturated bisimulation*, as it checks all contexts. Consequently, saturated bisimilarity $\sim_C$ ($\sim_{SAT}$ in [HK12]) is a congruence [BKM06, HK12], i.e., it is closed under contextualization. In other words $a \sim_C b$ implies $a;c \sim_C b;c$ for all contexts c.

2.3. **Representative Squares.** Checking bisimilarity of context steps is impractical because the transition system is generally infinitely branching: usually, f can be chosen from an infinite set of possible contexts, which all have to be checked. Most of these contexts are larger than necessary, that is, they contain elements that do not actively participate in the reduction. (In Example 2.3, contexts can be arbitrarily large, as long as they have an m-loop on the left node.) An improvement would be to check only the minimal contexts from which all other context steps can be derived.

When checking which contexts are required to make a rule applicable, in the reaction diagram (Definition 2.2) the arrows a, ℓ are given and we need to check for possible values of f (which generate matching c, a'). To derive a set of contexts f which is as small as possible — preferably finite — [BCHK11, HK12] introduced the notion of representative squares, which describe a way to represent all possible squares that close a pair a, ℓ by a smaller set of squares (the so-called representative squares). We can then limit bisimilarity checking to just the steps using representative squares, which, if this smaller set is indeed finite, leads to a finitely branching transition system.

Definition 2.4 (Representative squares [BCHK11])**.** A class κ of commuting squares in a category **C** is *representative* if κ satisfies the following condition: for each commuting square $(\alpha_1, \alpha_2, \delta_1, \delta_2)$ in **C** there exists a commuting square $(\alpha_1, \alpha_2, \beta_1, \beta_2)$ in κ and an arrow γ, such that $\delta_1 = \beta_1;\gamma$, $\delta_2 = \beta_2;\gamma$. This situation is depicted in Figure 4.

$$
\begin{array}{ccc}
A \xrightarrow{\alpha_1} B & & A \xrightarrow{\alpha_1} B \\
\alpha_2 \downarrow \quad \downarrow \delta_1 & \rightarrow & \alpha_2 \downarrow \quad \beta_1 \searrow \quad \downarrow \delta_1 \\
C \xrightarrow{\delta_2} D' & & C \xrightarrow{\beta_2} D \xrightarrow{\gamma} \\
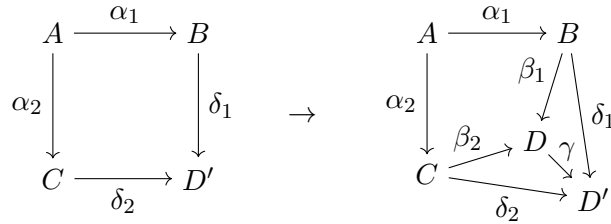& & C \xrightarrow{\delta_2} D'
\end{array}
$$

Figure 4: Every commuting square of the category (left) can be reduced to a representative square in κ and an arrow γ which extends the representative square to the original square (right).

For two arrows $\alpha_1 \colon A \to B$, $\alpha_2 \colon A \to C$, we define $\kappa(\alpha_1, \alpha_2)$ as the set of pairs of arrows (β_1, β_2) which, together with α_1, α_2, form representative squares in κ.

The original paper on reactive systems [LM00] used the (more restrictive) notion of idem pushouts instead of representative squares. Unfortunately, the universal property of idem pushouts leads to complications, in particular for cospan categories, where one has to resort to the theory of bicategories in order to be able to express this requirement. For the purposes of this paper, we stick to the simpler notion of representative squares, in order to keep our results independent of the concrete class of squares chosen.

The question arises which constructions yield suitable classes of representative squares, ideally with finite $\kappa(\alpha_1, \alpha_2)$, in order to represent all possible contexts δ_1, δ_2 with a finite set of representative contexts β_1, β_2. Pushouts can be used when they exist [HK12], however, they do not exist for $ILC(\mathbf{Graph_{fin}})$.

For cospan categories over adhesive categories, borrowed context diagrams — initially introduced as an extension of DPO rewriting [EK04] — can be used as representative squares. Before we can introduce such diagrams, we first need the notion of jointly epi.

Definition 2.5 (Jointly epi). A pair of arrows $f \colon B \to D$, $g \colon C \to D$ is *jointly epi* (*JE*) if for each pair of arrows $d_1, d_2 \colon D \to E$ the following holds: if $f; d_1 = f; d_2$ and $g; d_1 = g; d_2$, then $d_1 = d_2$.

In $\mathbf{Graph_{fin}}$ jointly epi equals jointly surjective, meaning that each node or edge of D is required to have a preimage under f or g or both (it contains only images of B or C).

This criterion is similar to, but weaker than a pushout: For jointly epi graph morphisms $d_1 \colon B \to D$, $d_2 \colon C \to D$, there are no restrictions on which elements of B, C can be merged in D. However, in a pushout constructed from morphisms $a_1 \colon A \to B$, $a_2 \colon A \to C$, elements in D can (and must) only be merged if they have a common preimage in A. (Hence every pushout generates a pair of jointly epi arrows, but not vice versa.)

Definition 2.6 (Borrowed context diagram [HK12]). A commuting diagram in the category $ILC(\mathbf{C})$, where $\mathbf{C}$ is adhesive, is a *borrowed context diagram* whenever it has the form of the diagram shown in Figure 5a, and the four squares in the base category $\mathbf{C}$ are pushout (PO), pullback (PB) or jointly epi (JE) as indicated. In particular $L \rightarrowtail G^+$, $G \rightarrowtail G^+$ must be jointly epi.

Figure 5b shows a more concrete version of Figure 5a, where graphs and their overlaps are depicted by Venn diagrams (assuming that all morphisms are injective). Because of the two pushout squares, this diagram can be interpreted as composition of cospans
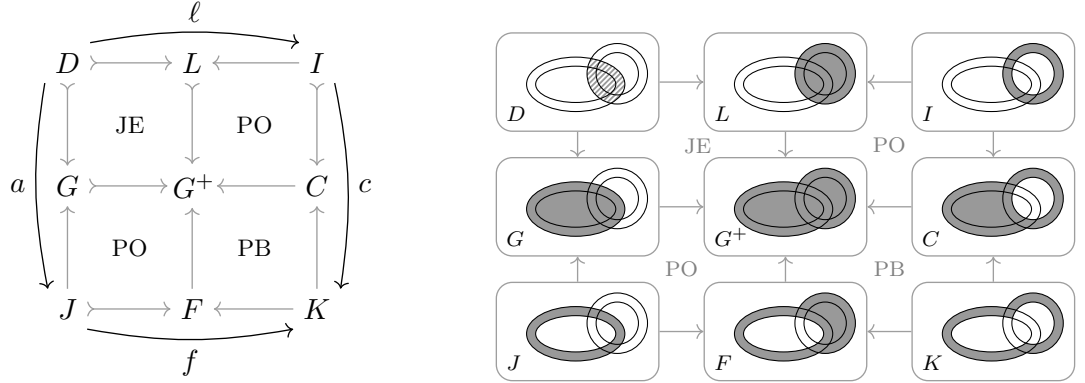
(a) Structure of a borrowed context diagram. The inner, lighter arrows are morphisms of the base category $\mathbf{C}$, while the outer arrows are morphisms of $ILC(\mathbf{C})$.

(b) Borrowed context diagrams represented as Venn diagrams. The outer circles represent graphs L, G, and the area between the inner and outer circles represents their interfaces I, J.

Figure 5: Borrowed context diagrams

$a;f = \ell;c = D \to G^+ \leftarrow K$ with extra conditions on the top left and the bottom right square. The top left square fixes an overlap G^+ of L and G, while D is contained in the intersection of L and G (shown as a hatched area). Being jointly epi ensures that it really is an overlap and does not contain unrelated elements. The top right pushout corresponds to the left pushout of a DPO rewriting diagram. It contains a total match of L in G^+. Then, the bottom left pushout gives us the minimal borrowed context F such that applying the rule becomes possible. The bottom right pullback ensures that the interface K is as large as possible.

We will discuss an example of a borrowed context diagram below (Example 2.9). For additional examples, we refer to [EK04].

For cospan categories over adhesive categories, borrowed context diagrams form a representative class of squares [BCHK11]. Furthermore, for some categories (such as **Graph$_{\mathbf{fin}}$**), there are — up to isomorphism — only finitely many jointly epi squares for a given span of monos and hence only finitely many borrowed context diagrams given a, ℓ (since pushout complements along monos in adhesive categories are unique up to isomorphism).

This motivates the following finiteness assumption that we will refer to in this paper: given a, ℓ, we require that $\kappa(a, \ell)$ is finite. (FIN)

2.4. **Representative Steps.** It is possible to define a reaction relation based on representative squares. By requiring that the left square is representative, we ensure that the contexts $\hat{f}$ are not larger than necessary:

Definition 2.7 (Representative step (without conditions) [HK12])**.** Let $a: 0 \to J$, $\hat{f}: J \to K$, $a': 0 \to K$ be arrows. We write $a \xrightarrow{\hat{f}}_R a'$ if a context step $a \xrightarrow{\hat{f}}_C a'$ is possible (i.e. $a;\hat{f} \rightsquigarrow a'$, i.e. for some rule (ℓ, r) and some arrow $\hat{c}$ we have $a;\hat{f} = \ell;\hat{c}$ and $r;\hat{c} = a'$) and additionally $\kappa(a, \ell) \ni (\hat{f}, \hat{c})$ (i.e. the arrows $(a, \ell, \hat{f}, \hat{c})$ form a representative square). Such steps are called *representative steps.*

Remark 2.8. Definitions 2.2 and 2.4 imply that every context step $a \xrightarrow{f}_C a'$ (left diagram) can be reduced to a representative step $a \xrightarrow{\hat{f}}_R r;\hat{c}$ (right diagram), a fact used in the proofs.

$$
\begin{array}{ccc}
0 \xrightarrow{\ell} I \xleftarrow{r} 0 & & 0 \xrightarrow{\ell} I \xleftarrow{r} 0 \\
\downarrow a \quad \downarrow c \; {}^{a'} & \rightarrow & \downarrow a \; {}^{\hat{f}} \; {}^{\hat{c}} \; K' {}_{\hat{g}} \; {}^{c} \; {}^{a'} \\
J \xrightarrow{f} K & & J \xrightarrow{f} K
\end{array}
$$

For this, we construct the representative square $(a, \ell, \hat{f}, \hat{c}) \in \kappa$ (which, according to Definition 2.4, always exists) from the square (a, ℓ, f, c) describing the context step. We obtain arrows $\hat{f}, \hat{c}$ and an arrow $\hat{g}$ which completes $\hat{f}, \hat{c}$ to f, c (i.e. $\hat{f};\hat{g} = f, \; \hat{c};\hat{g} = c$). ⌟

Example 2.9 (Representative steps)**.** Reconsider the reactive system described in Example 2.3, i.e., a message m can be transferred along a channel ch. One possible context step allows a channel $\boxed{\bullet \xrightarrow{ch} \bullet}$ to borrow a message $\boxed{m \curvearrowright \bullet \;\; \bullet}$ (depicted in Figure 6a) and do a transfer:
$(\emptyset \to \boxed{\bullet \xrightarrow{ch} \bullet} \leftarrow \boxed{\bullet \; \bullet}) \xrightarrow{(\boxed{\bullet \; \bullet} \to \boxed{m \curvearrowright \bullet \; \bullet} \leftarrow \boxed{\bullet \; \bullet})}_C (\emptyset \to \boxed{\bullet \xrightarrow{ch} \bullet \curvearrowleft m} \leftarrow \boxed{\bullet \; \bullet})$.

Another possible context step is to borrow an additional message on the right node, i.e.
$(\emptyset \to \boxed{\bullet \xrightarrow{ch} \bullet} \leftarrow \boxed{\bullet \; \bullet}) \xrightarrow{(\boxed{\bullet \; \bullet} \to \boxed{m \curvearrowright \bullet \; \bullet \curvearrowright m} \leftarrow \boxed{\bullet \; \bullet})}_C (\emptyset \to \boxed{\bullet \xrightarrow{ch} \bullet \curvearrowleft^m_m} \leftarrow \boxed{\bullet \; \bullet})$ (depicted in Figure 6b). Clearly, this is a valid context step, but the right message is not required by the rule, and we do not want to consider such steps in our analysis (by adding yet more messages, we obtain infinitely many context steps).

However, the second context step is not a representative step (assuming that representative squares correspond to borrowed context diagrams). We try to construct a borrowed context diagram: First we fill in the graphs given by a, f and ℓ, then we construct the bottom left pushout, we obtain $G^+ = \boxed{m \curvearrowright \bullet \xrightarrow{ch} \bullet \curvearrowleft m}$ as depicted in Figure 6b. Then however the top left square is not jointly epi, since neither $L = \boxed{m \curvearrowright \bullet \xrightarrow{ch} \bullet}$ (from ℓ) nor $G = \boxed{\bullet \xrightarrow{ch} \bullet}$ (from a) provide a preimage for the right m-loop $\boxed{\bullet \; \bullet \curvearrowleft m}$.
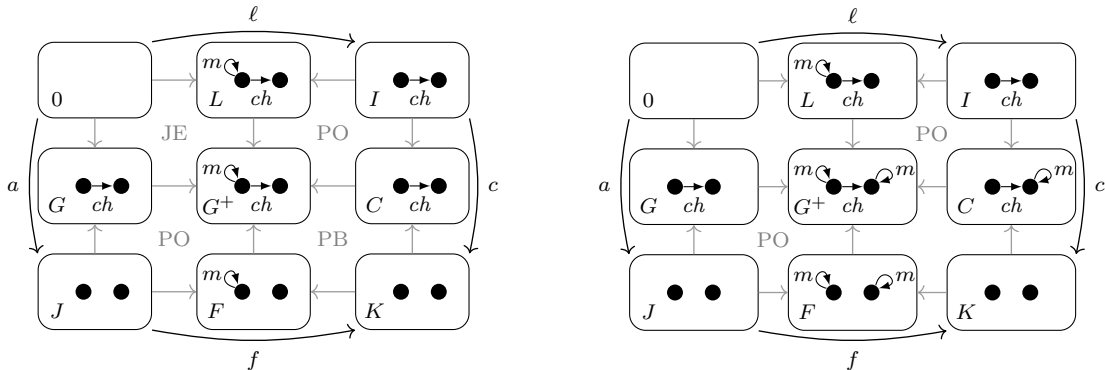
(a) Borrowed context diagram for a channel borrowing a message on the left node

(b) Commuting diagram for a channel additionally borrowing a message on the right, which is not needed for the reaction

Figure 6: Diagrams for the two steps described in Example 2.9.

On the other hand, the first context step is representative, since there $G^+ = \boxed{m\,\text{⟳}\text{•}\,ch\,\text{▸•}}$ does not contain the problematic right m-loop and it is possible to complete the borrowed context diagram as shown in Figure 6a. (To obtain the result of the context step, the right-hand side a' is constructed just as for context steps (see Example 2.3), which is not depicted here.) ⌟

In a *semi-saturated bisimulation*, $\to_R$-steps are answered by $\to_C$-steps (for every $(a,b) \in R$ and step $a \xrightarrow{f}_R a'$ there is $b \xrightarrow{f}_C b'$ such that $(a',b') \in R$ and vice versa). The resulting bisimilarity $\sim_R$ is identical [HK12] to saturated bisimilarity (i.e. $\sim_R = \sim_C$) and therefore also a congruence. Whenever (FIN) holds, $\sim_R$ is amenable to mechanization, since we have to consider only finitely many $\to_R$-steps ($\to_R$ is finitely branching).

Remark 2.10. Note that answering $\to_R$-steps with $\to_R$-steps gives a different, finer notion of behavioural equivalence than answering $\to_R$-steps with $\to_C$-steps. As an example, consider the reactive system with two rules: $\boxed{b\,{}^a\text{⟳}\text{•}} \Rightarrow \boxed{b\,{}^a\text{⟳}\text{•}}$ and $\boxed{\text{•↝}c} \Rightarrow \boxed{\text{•↝}c}$, where the single node is in the rule interface. Both rules replace the graphs with themselves, hence, any rewriting step does not change the graph at all.

By exhaustive enumeration of all representative steps, it is easy to see that $\boxed{{}^a\text{⟳•↝}c}$, $\boxed{\text{•↝}c}$ are semi-saturated bisimilar. It is important to keep in mind that the answering step does not need to use the same rule. For example, a step $\boxed{{}^a\text{⟳•↝}c} \xrightarrow{\boxed{b\,\text{⟳•}}}_R \boxed{b\,{}^a\text{⟳•↝}c}$ (using the first rule) can be answered by a step $\boxed{\text{•↝}c} \xrightarrow{\boxed{b\,\text{⟳•}}}_C \boxed{b\,\text{⟳•↝}c}$ (using the second rule). Because the resulting graphs both contain a c-loop, they are also bisimilar, since any subsequent steps (using either rule) can always be answered by applying the second rule.

However, the step $\boxed{\text{•↝}c} \xrightarrow{\boxed{b\,\text{⟳•}}}_C \boxed{b\,\text{⟳•↝}c}$ is not a representative step, because it borrows more than what is necessary to apply the second rule. There is also no other representative step that originates from $\boxed{\text{•↝}c}$ and borrows exactly $\boxed{b\,\text{⟳•}}$. Hence, under a notion of bisimulation where $\to_R$-steps are answered by $\to_R$-steps, $\boxed{{}^a\text{⟳•↝}c}$, $\boxed{\text{•↝}c}$ are not bisimilar. ⌟

3. Conditions for Reactive Systems

The reactive systems defined so far cannot represent rules where a certain component is required to be absent: whenever a reaction $a \rightsquigarrow a'$ is possible, a reaction $a;c \rightsquigarrow a';c$ (with additional context c) is also possible, with no method to prevent this. Restricting rule applications can be useful, e.g. to model access to a shared resource, which may only be accessed if no other entity is currently using it.

For graph transformation systems, application conditions with a first-order logic flavour have been studied extensively (e.g. in [HHT96, HP09]) and generalized to reactive systems in [BCHK11]. If we interpret such conditions in $ILC(\mathbf{Graph_{fin}})$, we obtain a logic that subsumes first-order logic (for more details on expressiveness see [BCHK11]).

In this section, we summarize the definitions from [BCHK11] and define shifting of conditions as partial evaluation. We then summarize the changes that are necessary to extend reactive systems with conditions. We illustrate the concepts of this chapter with various examples. An example for conditional reactive systems will be discussed later (Example 4.7). For an additional example, we refer to [BCHK11].

3.1. Conditions and Satisfaction.

Definition 3.1 (Condition [BCHK11]). Let $\mathbf{C}$ be a category and A be an object of $\mathbf{C}$. The set of conditions over A is defined inductively as:

- $\text{true}_A := (A, \forall, \emptyset)$ and $\text{false}_A := (A, \exists, \emptyset)$ are conditions over A (base case)
- $\mathcal{A} = (A, \mathcal{Q}, S)$ is a condition over A, where
  - $A = \text{Ro}(\mathcal{A})$ is the *root object* of $\mathcal{A}$,
  - $\mathcal{Q} \in \{\forall, \exists\}$ is a quantifier and
  - S is a finite set of pairs $(h, \mathcal{A}')$, where $h\colon A \to A'$ is an arrow and $\mathcal{A}'$ is a condition over A'.

Note that conditions can be represented as finite trees.

Definition 3.2 (Satisfaction [BCHK11]). Let $\mathcal{A}$ be a condition over A. For an arrow $a\colon A \to B$ and a condition $\mathcal{A}$ we define the satisfaction relation $a \models \mathcal{A}$ as follows:

- $a \models (A, \forall, S)$ iff for every pair $(h, \mathcal{A}') \in S$ and every arrow $g\colon \text{Ro}(\mathcal{A}') \to B$ we have: if $a = h;g$, then $g \models \mathcal{A}'$.
- $a \models (A, \exists, S)$ iff there exists a pair $(h, \mathcal{A}') \in S$ and an arrow $g\colon \text{Ro}(\mathcal{A}') \to B$ such that $a = h;g$ and $g \models \mathcal{A}'$.

From the above it follows that true_A is satisfied by every arrow with domain A, and false_A is satisfied by no arrow.

We write $\mathcal{A} \models \mathcal{B}$ ($\mathcal{A}$ implies $\mathcal{B}$) if for every arrow c with $\text{dom}(c) = \text{Ro}(\mathcal{A}) = \text{Ro}(\mathcal{B})$ we have: if $c \models \mathcal{A}$, then $c \models \mathcal{B}$. Two conditions are equivalent ($\mathcal{A} \equiv \mathcal{B}$) if $\mathcal{A} \models \mathcal{B}$ and $\mathcal{B} \models \mathcal{A}$.

With satisfaction defined in this way, universal conditions implicitly implement conjunction, i.e., $(A, \forall, \{(f_1, \mathcal{A}_1), (f_2, \mathcal{A}_2), \dots\})$ can be understood as $\forall(f_1, \mathcal{A}_1) \wedge \forall(f_2, \mathcal{A}_2) \wedge \dots$ (with a meaning analogous to the conditions of [HHT96, HP09]); similarly, existential conditions implement disjunction ($\exists(f_1, \mathcal{A}_1) \vee \exists(f_2, \mathcal{A}_2) \wedge \dots$).

Based on this, we can define negation of conditions, and conjunctions and disjunctions of arbitrarily quantified conditions such as $\exists(\dots) \wedge \exists(\dots)$ (note the flipped logical connective):

Proposition 3.3 (Boolean operations [BCHK11]). *Consider the following Boolean operations:*

- $\neg(A, \forall, S) := (A, \exists, \{(h, \neg\mathcal{A}') \mid (h, \mathcal{A}') \in S\})$, $\neg(A, \exists, S) := (A, \forall, \{(h, \neg\mathcal{A}') \mid (h, \mathcal{A}') \in S\})$
- $\mathcal{A} \vee \mathcal{B} := (A, \exists, \{(\text{id}_A, \mathcal{A}), (\text{id}_A, \mathcal{B})\})$ *for two conditions* $\mathcal{A}, \mathcal{B}$ *over* A
- $\mathcal{A} \wedge \mathcal{B} := (A, \forall, \{(\text{id}_A, \mathcal{A}), (\text{id}_A, \mathcal{B})\})$ *for two conditions* $\mathcal{A}, \mathcal{B}$ *over* A

These operations satisfy the standard laws of propositional logic, i.e. $a \models \neg\mathcal{A}$ if and only if $a \not\models \mathcal{A}$; $a \models \mathcal{A} \vee \mathcal{B}$ if and only if $a \models \mathcal{A} \vee a \models \mathcal{B}$, analogously for $\mathcal{A} \wedge \mathcal{B}$. Conjunction and disjunction can be extended in the obvious way to arbitrary, rather than binary, conjunctions and disjunctions.

Example 3.4 (Examples of conditions).

- $\mathcal{A}_{node}$ recognizes graphs that contain at least one node:

$$\mathcal{A}_{node} = \left(\emptyset, \exists, \left\{(\emptyset \to \bullet \leftarrow \bullet, \text{ true}_\bullet)\right\}\right)$$

The condition is checked as follows: any arrow that satisfies the condition must be decomposable into two arrows, the first of which is given in the condition and contributes the required node, and the second optionally provides additional elements. Since the output interface is not empty, the second arrow is free to connect edges to the required node, i.e. the condition matches both isolated and non-isolated nodes.

- The condition $\mathcal{A}_{iso}$ recognizes graphs that contain an isolated node:

$$\mathcal{A}_{iso} = \Big(\emptyset, \exists, \Big\{ (\emptyset \to \bullet \leftarrow \emptyset, \; \text{true}_\emptyset) \Big\} \Big)$$

As the outer interface of $h = \emptyset \to \bullet \leftarrow \emptyset$ is empty, $h;g$ has to contain an isolated node (g can only connect an edge to the node provided by h if it is contained in the interface).

- $\mathcal{A}_{ab}$ recognizes the graphs where for all occurrences of an a-edge, there also exists a b-edge in the opposite direction:

$$\mathcal{A}_{ab} = \Big(\emptyset, \forall, \Big\{ (\emptyset \;\to\; \bullet\!\xrightarrow{a}\!\bullet \;\leftarrow\; \bullet\,\bullet, \; \mathcal{A}') \Big\} \Big)$$

$$\mathcal{A}' = \Big(\bullet\,\bullet, \exists, \Big\{ (\bullet\,\bullet \;\to\; \bullet\!\xleftarrow{b}\!\bullet \;\leftarrow\; \bullet\,\bullet, \text{true}_{\bullet\bullet}) \Big\} \Big)$$

Note that in the examples above, the root object of the condition is empty, since we only consider isolated conditions. When using conditions in a transformation rule, we would use the interface of the rule instead. This ensures that the condition is evaluated at the same position where the rule is applied, and not in any other position.   ⌟

3.2. Shifting as Partial Evaluation of Conditions.

When evaluating conditions, it is sometimes known that a given context is guaranteed to be present. In this case, a condition can be rewritten, using representative squares, under the assumption that this context is provided by the environment. This operation is known as shift [HP09]:

Definition 3.5 (Shift of a condition [BCHK11])**.** Given a fixed class of representative squares κ, the *shift of a condition* $\mathcal{A} = (A, \mathcal{Q}, S)$ *along an arrow* $c\colon A \to B$ is inductively defined as follows:

$$\mathcal{A}_{\downarrow c} := \Big(B, \mathcal{Q}, \Big\{ (\beta, \mathcal{A}'_{\downarrow \alpha}) \,\Big|\, (h, \mathcal{A}') \in S, \; (\alpha, \beta) \in \kappa(h, c) \Big\} \Big)$$

The shift operation can be understood as a partial evaluation of $\mathcal{A}$ under the assumption that c is already present. It satisfies $c;d \models \mathcal{A} \iff d \models \mathcal{A}_{\downarrow c}$.

$$Z \dashrightarrow^{a} A \xrightarrow{c} B \xrightarrow{d} X$$

$$\mathcal{A} \qquad \mathcal{A}_{\downarrow c}$$

The typical case, which we will encounter throughout the rest of this paper, is that a condition on the context of some arrow a is given, this arrow is then placed into some environment c (which might not fully satisfy the condition, but possibly parts of it), and we are interested in a condition that an additional context d has to satisfy. (For instance, if $\mathcal{A}$ requires the existence of two elements and c already provides one of them, then d only needs to add the other one, which is reflected in $\mathcal{A}_{\downarrow c}$.)

The representation of the shifted condition may differ depending on the class of representative squares chosen. However, no matter which class is chosen, the resulting conditions are equivalent to each other. Furthermore, if we assume that (FIN) holds, shifting a finite condition will again result in a finite condition.

Representative squares as well as shift play a major role in the diagrammatic proofs. The shift operation satisfies a few equivalences that we will use in the proofs of our theorems:
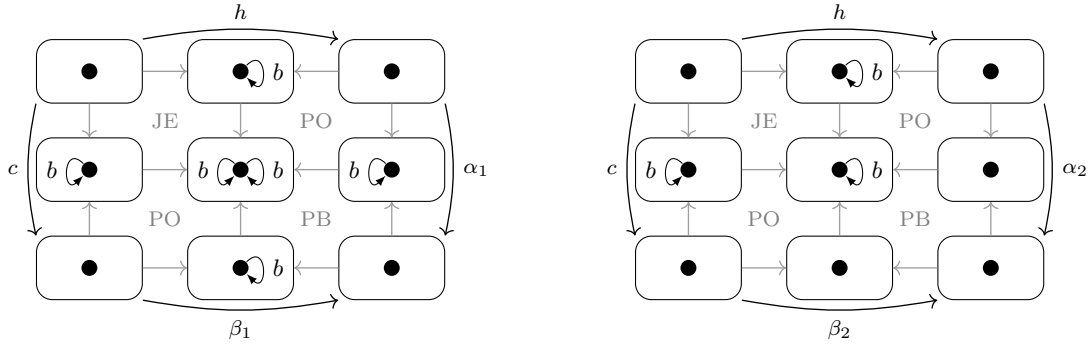
(a) b-loops of h and c are kept separate in the center graph.

(b) b-loops of h and c are mapped to a single loop.

Figure 7: Borrowed context diagrams for h, c as given in Example 3.7.

Theorem 3.6 (Shift laws [BCHK11, Proposition 17])**.**

$$(\mathcal{A} \vee \mathcal{B})_{\downarrow c} \equiv \mathcal{A}_{\downarrow c} \vee \mathcal{B}_{\downarrow c} \qquad\qquad \mathcal{A}_{\downarrow \mathrm{id}} \equiv \mathcal{A} \qquad\qquad \mathcal{A} \models \mathcal{B} \implies \mathcal{A}_{\downarrow c} \models \mathcal{B}_{\downarrow c}$$

$$(\mathcal{A} \wedge \mathcal{B})_{\downarrow c} \equiv \mathcal{A}_{\downarrow c} \wedge \mathcal{B}_{\downarrow c} \qquad\qquad \mathrm{true}_{\downarrow c} \equiv \mathrm{true} \qquad\qquad \mathcal{A}_{\downarrow c_1 ; c_2} \equiv (\mathcal{A}_{\downarrow c_1})_{\downarrow c_2}$$

$$\neg(\mathcal{A}_{\downarrow c}) \equiv (\neg \mathcal{A})_{\downarrow c} \qquad\qquad \mathrm{false}_{\downarrow c} \equiv \mathrm{false}$$

Example 3.7 (Simplifying conditions by shifting)**.** Let the following condition $\mathcal{A}_{nb}$ be given, which requires that the interface node does not have a b-loop attached:

$$\mathcal{A}_{nb} = \left(\bullet, \forall, \left\{ (\bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet, \; \mathrm{false}) \right\} \right)$$

Furthermore let the cospan $c = \bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet$. We now compute the result of the shift $\mathcal{A}_{nb\downarrow c}$, i.e., the condition $\mathcal{A}_{nb}$ under the assumption that c is already given. We expect the resulting condition to be equivalent to false, since the presence of the b-loop in c already violates $\mathcal{A}_{nb}$. We will show that this is indeed the case. By Definition 3.5 we have:

$$\mathcal{A}_{nb\downarrow c} = \left(\bullet, \forall, \left\{ (\beta, \mathcal{A}'_{\downarrow \alpha}) \mid (h, \mathcal{A}') \in S, \; (\alpha, \beta) \in \kappa(h, c) \right\} \right)$$
$$= \left(\bullet, \forall, \left\{ (\beta, \mathcal{A}'_{\downarrow \alpha}) \mid \mathcal{A}' = \mathrm{false}, \; h = \bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet, \; (\alpha, \beta) \in \kappa(h, c) \right\} \right)$$

We can obtain possible α, β by enumerating the borrowed context diagrams where h, c are already given. As seen in Figure 7, there are two possible choices for the jointly epi square in the top left: the b-loops of h and c can be mapped to two different loops in the center graph (Figure 7a) or they can be mapped to a single loop (Figure 7b). The remaining pushout and pullback squares are then uniquely determined. We therefore obtain:

$$= \left(\bullet, \forall, \left\{ \left(\bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet, (\mathrm{false})_{\downarrow \alpha_1} \right), \left(\bullet \to \bullet \leftarrow \bullet, (\mathrm{false})_{\downarrow \alpha_2} \right) \right. \right.$$
$$\left. \left. \mid \alpha_1 = \bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet, \; \alpha_2 = \bullet \to \bullet \leftarrow \bullet \right\} \right)$$

Shifting false along any arrow α_i again results in false:

$$= \left(\bullet, \forall, \left\{ (\bullet \to \bullet\!\!\curvearrowleft b \; \leftarrow \bullet, \mathrm{false}), (\bullet \to \bullet \leftarrow \bullet, \mathrm{false}) \right\} \right)$$

Furthermore, since $\bullet \to \bullet \leftarrow \bullet$ is an identity cospan, the condition is equivalent to $\ldots \wedge \mathrm{false} \equiv \mathrm{false}$, which is the expected result. ⌟

3.3. **Conditional Reactive Systems.** We now extend reactive systems with application conditions:

Definition 3.8 (Conditional reactive system [BCHK11])**.** A *rule with condition* is a triple $(\ell, r, \mathcal{R})$ where $\ell, r \colon 0 \to I$ are arrows and $\mathcal{R}$ is a condition with root object I. A *conditional reactive system* is a set of rules with conditions.

As the root object I of the condition is the codomain of the rule arrow, it is also the domain of the reactive context, which has to satisfy the rule condition in order to be able to apply the rule:

Definition 3.9 (Reaction)**.** Let a, a' be arrows of a conditional reactive system with rules $\mathcal{S}$. We say that *a reduces to a'* ($a \rightsquigarrow a'$) whenever there exists a rule $(\ell, r, \mathcal{R}) \in \mathcal{S}$ with $\ell, r \colon 0 \to I$ and a reactive context $c \colon I \to J$ such that $a = \ell;c$, $a' = r;c$ and additionally $c \models \mathcal{R}$.

In order to define a bisimulation for conditional reactive systems that is also a congruence, it is necessary to enrich labels with conditions derived from the application conditions. Since we can not assume that the full context is present, the application condition might refer to currently unknown parts of the context and this has to be suitably integrated into the label.

Definition 3.10 (Context/representative step with conditions [HK12])**.** Let $\mathcal{S}$ be a conditional reactive system, let $a \colon 0 \to J$, $f \colon J \to K$, $a' \colon 0 \to K$ be arrows and $\mathcal{A}$ be a condition over K. We write $a \xrightarrow{f, \mathcal{A}}_C a'$ whenever there exists a rule $(\ell, r, \mathcal{R}) \in \mathcal{S}$ and an arrow c such that $a;f = \ell;c$, $a' = r;c$ (i.e. the reaction is possible without conditions) and furthermore $\mathcal{A} \models \mathcal{R}_{\downarrow c}$ (a condition on an additional context d as explained below). Such steps are called *context steps*.

We write $a \xrightarrow{f, \mathcal{A}}_R a'$ whenever $a \xrightarrow{f, \mathcal{A}}_C a'$, $\kappa(a, \ell) \ni (f, c)$ and $\mathcal{A} = \mathcal{R}_{\downarrow c}$. Such steps are called *representative steps*.

Conditions are represented graphically in the form of "arrowhead shapes" depicted next to the root object. Intuitively $a \xrightarrow{f, \mathcal{A}}_C a'$ means that a can make a step to a' when borrowing f, if the yet unknown context d beyond f satisfies condition $\mathcal{A}$ (since this context d does not directly participate in the reduction, we call it *passive context*).

The intuition behind this requirement is that $\mathcal{A}$ should allow only the contexts that are allowed by the rule condition $\mathcal{R}$ (thereby checking that the rule can actually be applied). Since $\mathcal{A}$ is a condition over an additional context d that is beyond the reaction context c, and c might partially satisfy $\mathcal{R}$, we shift $\mathcal{R}$ over c to obtain a condition that only requires the parts of $\mathcal{R}$ that are still missing. For context steps, $\mathcal{A}$ may also be stronger, hence $\models$.

In the case of a representative step, we require that a context step is possible, the borrowed context is minimal, and the condition on the passive context is not stronger than necessary.

In the proofs, we will make extensive use of the following construction to obtain a representative step for a given context step:

Remark 3.11. Definitions 2.4 and 3.10 imply, analogously to Remark 2.8, that every context step $a \xrightarrow{f, \mathcal{A}}_C a'$ (left diagram) can be reduced to a representative step $a \xrightarrow{\hat{f}, \mathcal{R}_{\downarrow \hat{c}}}_R r;\hat{c}$ (right diagram), with $\mathcal{R}$ being the condition of the rule $(\ell, r, \mathcal{R})$ that enables the given context step.
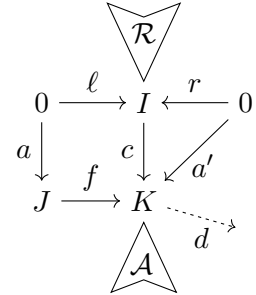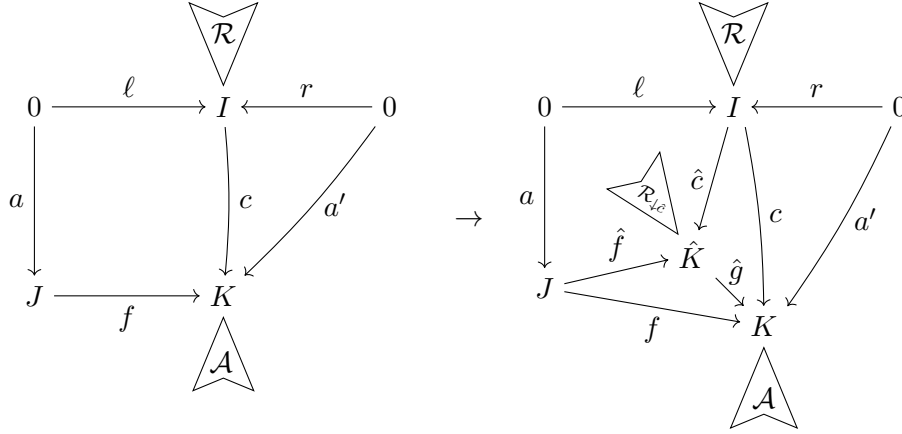
We will also make use of the following context-step-rewriting lemma:

Lemma 3.12. *A context step* $a;d \xrightarrow{f,\mathcal{A}}_C a'$ *is equivalent to a step* $a \xrightarrow{d;f,\,\mathcal{A}}_C a'$. *In particular,* $a;d \xrightarrow{f,\mathcal{A}}_C a'$ *if and only if* $a \xrightarrow{d;f,\,\mathcal{A}}_C a'$.

Proof. According to Definition 3.10, for a step $a;d \xrightarrow{f,\mathcal{A}}_C a'$ there exists a rule $(\ell, r, \mathcal{R})$ such that $(a;d);f = \ell;c$, $a' = r;c$, $\mathcal{A} \models \mathcal{R}_{\downarrow c}$ for some arrow c. Since composition is associative, we rewrite this to $a;(d;f) = \ell;c$, which immediately results in the definition of $a \xrightarrow{d;f,\,\mathcal{A}}_C a'$. $\quad\square$

We now extend (semi-)saturated bisimilarity to rules with conditions:

Definition 3.13 ((Semi-)Saturated bisimilarity [HK12])**.** Let $\mathcal{S}$ be a conditional reactive system. A *saturated bisimulation* is a relation R, relating arrows $a, b\colon 0 \to J$, such that: for all $(a,b) \in R$ and for every context step $a \xrightarrow{f,\mathcal{A}}_C a'$ there exist answering moves $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$, $i \in I$, such that $(a', b'_i) \in R$ and $\mathcal{A} \models \bigvee_{i \in I} \mathcal{B}_i$, where I is a finite index set; and, vice versa, for every context step $b \xrightarrow{f,\mathcal{B}}_C b'$ there exist answering moves $a \xrightarrow{f,\,\mathcal{A}_j}_C a'_j$, $j \in J$, such that $(a'_j, b') \in R$ and $\mathcal{B} \models \bigvee_{j \in J} \mathcal{A}_j$.

Two arrows a, b are called *saturated bisimilar* ($(a,b) \in \sim_C$) whenever there exists a saturated bisimulation R with $(a,b) \in R$. Similarly, for *semi-saturated bisimilarity* we require that $\to_R$-steps of a can be answered by $\to_C$-steps of b, and vice versa for $\to_R$-steps of b. Saturated and semi-saturated bisimilarity agree and both are congruences [HK12].

4. Conditional Bisimilarity

We will now introduce our new results on conditional bisimilarity: as stated earlier, our motivation is to extend the notion of saturated bisimilarity, which is often too strict, since it requires that two system states behave identically in all possible contexts. However, sometimes it is enough to ensure behavioural equivalence only in specific environments.

Hence we now replace standard bisimilarity, which is a binary relation, by a ternary relation — called conditional relation — with tuples of the form $(a, b, \mathcal{C})$. Then, a conditional bisimulation is a conditional relation, where a tuple $(a, b, \mathcal{C})$ can be read as: a, b are bisimilar in all contexts satisfying $\mathcal{C}$.

4.1. Definition, Properties and Examples.

Definition 4.1 (Conditional relation). A *conditional relation* is a set of triples $(a, b, \mathcal{C})$, where $a, b \colon 0 \to J$ are arrows with identical target and $\mathcal{C}$ is a condition over J.

Note that for a triple $(a, b, \mathcal{C})$, the root object of the condition $\mathcal{C}$ is not the source of a (as is the case for satisfaction), but the target $\mathrm{codom}(a)$. This is because we do not state a condition on the arrows a, b themselves, but on the context in which they are embedded ($a;f$ resp. $b;f$ for some context f), so the condition is over $\mathrm{dom}(f) = \mathrm{codom}(a) = \mathrm{codom}(b)$.

Definition 4.2 (Closure under contextualization, $u(R)$, conditional congruence). If R is a conditional relation, then:
- R is *reflexive* if $(a, a, \mathcal{C}) \in R$ for all $a, \mathcal{C}$ with $\mathrm{codom}(a) = \mathrm{Ro}(\mathcal{C})$
- R is *symmetric* if $(a, b, \mathcal{C}) \in R$ implies $(b, a, \mathcal{C}) \in R$
- R is *transitive* if $(a, b, \mathcal{C}) \in R$ and $(b, c, \mathcal{C}) \in R$ implies $(a, c, \mathcal{C}) \in R$
- R is *closed under contextualization* if $(a, b, \mathcal{C}) \in R$ implies $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in R$
- R is a *conditional congruence* if it is an equivalence (reflexive, symmetric, transitive) and closed under contextualization

For a conditional relation R, $u(R)$ is its closure under contextualization, that is, $u(R) := \{(a;d, b;d, \mathcal{C}_{\downarrow d}) \mid (a, b, \mathcal{C}) \in R, \ a, b \colon 0 \to J, \ d \colon J \to K\}$.

Closure under contextualization means that whenever a, b are related under a context satisfying $\mathcal{C}$, then they are still related when we contextualize under d, where however the condition has to be shifted since we commit to the fact that the context is of the form $d;c$ for some additional context c.

We will now introduce one of the central definitions of this paper. Here, given a conditional reactive system we will describe when two arrows a, b are bisimilar in all contexts that satisfy a condition $\mathcal{C}$.

Definition 4.3 (Conditional bisimulation). We fix a conditional reactive system. A conditional bisimulation R is a conditional relation such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each context step $a \xrightarrow{f, \mathcal{A}}_C a'$, there are answering steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$, $i \in I$ (where I is possibly infinite), and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$; and vice versa[1]. Two arrows are *conditionally bisimilar under $\mathcal{C}$* ($(a, b, \mathcal{C}) \in {\,\raisebox{0.3ex}{$\scriptstyle\circ$}\kern-0.55em\sim}_C$) whenever a conditional bisimulation R with $(a, b, \mathcal{C}) \in R$ exists.[2]

The situation for one answer step is depicted in Figure 8. Since the definition is rather complex, we will discuss its various aspects in the following remarks.

Remark 4.4 (Logical implication). In Definition 4.3, the implication $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$ is to be understood as follows: For every step, we have a borrowed context f and an additional passive context d (as explained below Definition 3.10). The condition $\mathcal{C}$ from the triple refers to the full context of a (i.e. both the borrowed context f and the passive context d, hence $f;d \models \mathcal{C}$ or equivalently $d \models \mathcal{C}_{\downarrow f}$), while $\mathcal{A}$, coming from the context step, only refers to the passive context d (hence $d \models \mathcal{A}$).

Every environment d that is valid for the context step of a (i.e. which satisfies $\mathcal{A} \wedge \mathcal{C}_{\downarrow f}$) must also be valid for some answering step of b, i.e. satisfies at least one $\mathcal{B}_i$. Depending on

---

[1] For each triple $(a, b, \mathcal{C}) \in R$ and each context step $b \xrightarrow{f, \mathcal{B}}_C b'$, there are answering steps $a \xrightarrow{f, \mathcal{A}_j}_C a'_j$ and conditions $\mathcal{C}'_j$ such that $(a'_j, b', \mathcal{C}'_j) \in R$ and $\mathcal{B} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{j \in J} (\mathcal{C}'_j \wedge \mathcal{A}_j)$

[2] Note that since conditional bisimulations are closed under union, ${\,\raisebox{0.3ex}{$\scriptstyle\circ$}\kern-0.55em\sim}_C$ is itself a conditional bisimulation.
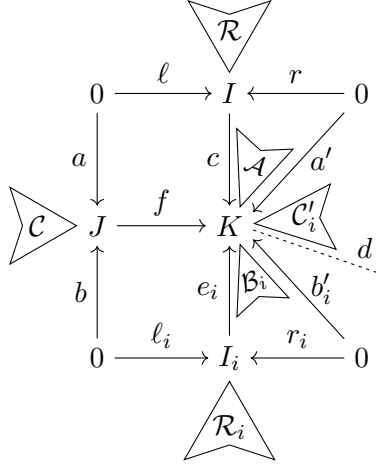
Figure 8: A context step $a \xrightarrow{f,\mathcal{A}}_C a'$ (top half) and a single answer step $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$ (bottom half) in conditional bisimulation

the context, different answering steps may be chosen, and the resulting pair a', b'_i might only be conditionally bisimilar for some contexts, which is indicated by the condition $\mathcal{C}'_i$. ⌟

Remark 4.5 (Necessity of multiple answering steps). As for saturated bisimilarity, we need to allow several answering moves for a single step of a: the answering step taken by b might depend on the context, using different rules for contexts satisfying different conditions $\mathcal{B}_i$. We just have to ensure that all answering step conditions taken together (disjunction on the right-hand side) fully cover the conditions under which the step of a is feasible (left-hand side). As an example for this, consider the following example (originally presented in [HK12, remark after Definition 15]). Assume three rules:

1. $R_A \ = \big(\emptyset \to \boxed{\bullet\!\xrightarrow{a}\!\bullet}\leftarrow \bullet\ \bullet,\ \emptyset \to \bullet\ \bullet \leftarrow \bullet\ \bullet, \mathrm{true}_{\bullet\ \bullet}\big)$ (unconditionally delete an a-edge)

2. $R_{B1} = \big(\emptyset \to \boxed{\bullet\!\xrightarrow{b}\!\bullet}\leftarrow \bullet\ \bullet,\ \emptyset \to \bullet\ \bullet \leftarrow \bullet\ \bullet,\ \mathcal{A}_q\big)$ (delete a b-edge if the context satisfies some condition $\mathcal{A}_q$)

3. $R_{B2} = \big(\emptyset \to \boxed{\bullet\!\xrightarrow{b}\!\bullet}\leftarrow \bullet\ \bullet,\ \emptyset \to \bullet\ \bullet \leftarrow \bullet\ \bullet,\ \neg\mathcal{A}_q\big)$ (delete a b-edge in contexts not satisfying $\mathcal{A}_q$)

(Hence rules 2 and 3 together allow a b-edge to be deleted in any context, since every context satisfies either $\mathcal{A}_q$ or $\neg\mathcal{A}_q$. The condition $\mathcal{A}_q$ can be chosen arbitrarily, as long as it is not equivalent to true or false.) Then, an a-edge is conditionally bisimilar to a b-edge under true (all contexts): a step that deletes the a-edge can be answered by deleting the b-edge, but depending on the context that the step happens in, a different rule has to be chosen: two answering steps with $\mathcal{B}_1 = \mathcal{A}_q$, $\mathcal{B}_2 = \neg\mathcal{A}_q$ are required, and together they cover true. (The other direction — deleting the b-edge using either rule 2 or 3 being answered by deleting the a-edge using rule 1 — does not require multiple answering steps in this example.) ⌟

Remark 4.6 (Infinitely many answering steps). The definition explicitly permits an infinite index set I for the answering steps (this is in contrast to saturated bisimilarity, cf. Definition 3.13, which required finite I). If we do not consider conditional bisimilarity and the finiteness assumption (FIN) holds, it does not make a difference whether we consider finite

or infinite index sets, since there are only finitely many possible answering steps [HK12]. However, in the presence of conditions, it might make a difference.

Since the logic does not support infinite disjunctions, $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} \mathcal{C}'_i \wedge \mathcal{B}_i$ means that for every d with $d \models \mathcal{A} \wedge \mathcal{C}_{\downarrow f}$, there exists $i \in I$ such that $d \models \mathcal{C}'_i \wedge \mathcal{B}_i$.

In many practical applications, it may be useful to restrict to a variant of the definition that permits only finitely many answering steps. Our theorems are valid for either variant (finite or infinite), except for the proof of Theorem 6.4 which in its current version requires I to be infinite.   ⌟

Example 4.7 (Message passing over unreliable channels). We now work in the category of input-linear cospans of graphs, i.e., $ILC(\mathbf{Graph_{fin}})$.

We extend our previous example (cf. Example 2.3) of networked nodes, introducing different types of channels. A channel can be reliable or unreliable, indicated by an *rel*-edge or *unr*-edge respectively. Sending a message over a reliable channel always succeeds (rule P_R), while an unreliable channel only transmits a message if there is no noise (indicated by a parallel *noise*-edge) in the environment that disturbs the transmission (rule P_U). The reactive system has the following rules with application conditions, where condition $\mathcal{A}_{U\sharp n}$ states that the unreliable channel must not have an *noise*-edge in parallel:

$$\mathcal{A}_{U\sharp n} = \left(U_0, \forall, \left\{ \left(\boxed{\,U_0 \;\; \bullet \xrightarrow{unr} \bullet\,} \rightarrow \boxed{\,U_N \;\; \bullet \overset{noise}{\underset{unr}{\rightrightarrows}} \bullet\,} \leftarrow \boxed{\,U_0 \;\; \bullet \xrightarrow{unr} \bullet\,} , \text{false}_{U_0} \right) \right\} \right)$$

Hence the application condition $\mathcal{A}_{U\sharp n}$ says that the context must *not* be decomposable into $U_0 \to U_N \leftarrow U_0$ and some other cospan, which is only the case if the *unr*-edge in the interface of P_U has no parallel *noise*-edge. In other words: there is no noise.

We compare the behaviour of a reliable channel ($r := \emptyset \to \bullet \xrightarrow{rel} \bullet \leftarrow \bullet \; \bullet$) to that of an unreliable channel ($u := \emptyset \to \bullet \xrightarrow{unr} \bullet \leftarrow \bullet \; \bullet$). It is easy to see that they are not saturated bisimilar: r can do a step by borrowing a message on the left ($f := \bullet \; \bullet \to \bullet \xrightarrow{m} \bullet \; \bullet \leftarrow \bullet \; \bullet$) without further restrictions (i.e. using an environment condition $\mathcal{A} = \text{true}$). But u is unable to answer this step, because the corresponding rule is only applicable if no *noise*-edge is present.

However, r and u are conditionally bisimilar under the assumption that no *noise*-edge is present between the two nodes ($\mathcal{C} = \mathcal{A}_{\overline{n}}$, where $\mathcal{A}_{\overline{n}} := \left(\bullet \; \bullet, \forall, \left\{ (\bullet \; \bullet \to \bullet \xrightarrow{noise} \bullet \leftarrow \bullet \; \bullet, \right.\right.$ $\left.\left. \text{false}_{\bullet \; \bullet}) \right\} \right)$), i.e. there exists a conditional bisimulation that contains $(r, u, \mathcal{A}_{\overline{n}})$. A direct proof is hard, since the proof involves checking infinitely many context steps, since messages accumulate on the right-hand side. However, in Example 4.18 we will use an argument based on representative steps to construct a proof.   ⌟

Remark 4.8 (Condition strengthening)**.** It holds that $(a, b, \mathcal{C}') \in \backsimeq_C$, $\mathcal{C} \models \mathcal{C}'$ implies $(a, b, \mathcal{C}) \in \backsimeq_C$. (This is due to the fact that $\mathcal{C} \models \mathcal{C}'$ implies $\mathcal{C}_{\downarrow f} \models \mathcal{C}'_{\downarrow f}$ which, in Definition 4.3, implies $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \mathcal{A} \wedge \mathcal{C}'_{\downarrow f}$ for any condition $\mathcal{A}$ and arrow f.)   ⌐

Our motivation for introducing the notion of conditional bisimilarity was to check whether two systems are behaviourally equivalent when they are put into a context that satisfies some condition $\mathcal{C}$. It is not immediately obvious that our definition can be used for this purpose, since all context steps are checked, not just the ones that actually satisfy $\mathcal{C}$.

Hence we now show that our definition is sound, i.e. if two systems are conditionally bisimilar, then they show identical behaviour under all contexts that satisfy $\mathcal{C}$.

Theorem 4.9. *Let R be a conditional bisimulation. Then $R' = \{(a;d, b;d) \mid (a, b, \mathcal{C}) \in R \wedge d \models \mathcal{C}\}$ is a bisimulation for the reaction relation $\rightsquigarrow$.*

Proof. To prove that R' is a bisimulation, we need to show that if $a;d\ R'\ b;d$ and $a;d \rightsquigarrow a'$, then there exists b' such that $b;d \rightsquigarrow b'$ and $a'\ R'\ b'$; and vice versa. Equivalently, if $a;d$ can do a step, then $b;d$ can answer this step (and vice versa) and the result is again contained in the bisimulation R'. We show only one direction ($a;d$ answered by $b;d$), the other one can be done analogously.

Now let some $(a;d, b;d) \in R'$ be given, for which there must exist a triple $(a, b, \mathcal{C}) \in R$. Consider a step $a;d \rightsquigarrow a'$. This step is due to some rule $(\ell, r, \mathcal{R})$, shown graphically in Figure 9a.
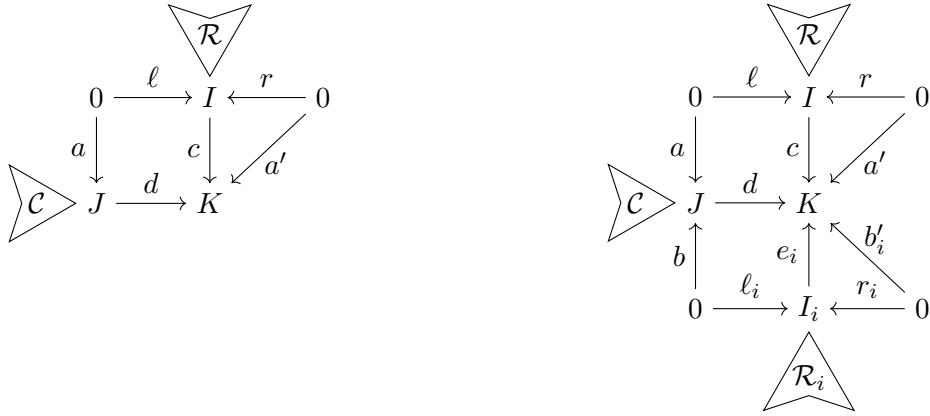
(a) A reaction $a;d \rightsquigarrow a'$ for some context d that satisfies $\mathcal{C}$.

(b) The same step, together with answering steps provided by the conditional bisimulation.

Figure 9: The steps considered in the proof of Theorem 4.9

We have $c \models \mathcal{R}$ (otherwise the rule would not be applicable and therefore the step $a;d \rightsquigarrow a'$ would not be possible) and $d \models \mathcal{C}$ (follows from the given $(a;d, b;d) \in R'$ by construction of R'). To make them usable for the answering steps, we transform $\mathcal{R}, \mathcal{C}$ to be conditions over K. Trivially $d = d; \mathrm{id}_K$, so using Definition 3.5 we rewrite $d; \mathrm{id}_K \models \mathcal{C}$ to $\mathrm{id}_K \models \mathcal{C}_{\downarrow d}$. Analogously, we rewrite $c; \mathrm{id}_K \models \mathcal{R}$ to $\mathrm{id}_K \models \mathcal{R}_{\downarrow c}$.

We set $\mathcal{A} := \mathcal{R}_{\downarrow c}$ and interpret this diagram as $a \xrightarrow{d, \mathcal{A}}_C a'$. Since R is a conditional bisimulation and $(a, b, \mathcal{C}) \in R$, b can answer the step of a, using one of possibly several rules $(\ell_i, r_i, \mathcal{R}_i)$ depending on the given context d. Setting $\mathcal{B}_i := \mathcal{R}_{i \downarrow e_i}$, we get steps $b \xrightarrow{d, \mathcal{B}_i}_C b'_i$ and we extend the diagram as shown in Figure 9b.

Generally, not every answering step that is possible for our given triple $(a, b, \mathcal{C})$ is a suitable answering step for the given context d. But since R is a conditional bisimulation, we know that $\mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow d} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$. Previously we derived $\mathrm{id}_K \models \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow d}$. Therefore, id_K also satisfies $\bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$, that is, id_K satisfies $\mathcal{C}'_i \wedge \mathcal{B}_i = \mathcal{C}'_i \wedge \mathcal{R}_{i \downarrow e_i}$ for some i. From now on, we only consider answering steps for which this is indeed the case.

Using Definition 3.5 we rewrite $\mathrm{id}_K \models \mathcal{R}_{i \downarrow e_i}$ to $e_i; \mathrm{id}_K = e_i \models \mathcal{R}_i$, which means that the rule $(\ell_i, r_i, \mathcal{R}_i)$ can actually be applied, that is, $b; d \rightsquigarrow b'_i = r_i; e_i$. So b has a suitable answering step.

To show that R' is a bisimulation, we only have to show that $(a', b'_i) \in R'$. As R is a conditional bisimulation, for the given answering step we know that $(a', b'_i, \mathcal{C}'_i) \in R$. Previously we had $\mathrm{id}_K \models \mathcal{C}'_i$, therefore, the requested pair $(a'; \mathrm{id}_K, b'_i; \mathrm{id}_K) = (a', b'_i)$ is added during the construction of R'. $\qquad \square$

Remark 4.10. Note that the converse of Theorem 4.9 (if R' is a bisimulation, then R is a conditional bisimulation) does not hold. Consider the following counterexample:

$$R_1 = \left(\bullet \to \boxed{a\,\underset{\bullet}{\leftrightarrow}\,x} \leftarrow \boxed{\quad \bullet\!\leftrightarrow x} \, , \ \bullet \to \boxed{e\,\underset{\bullet}{\leftrightarrow}\,x} \leftarrow \boxed{\quad \bullet\!\leftrightarrow x} \, , \ \mathrm{true}_X \right)$$

$$R_2 = \left(\bullet \to \boxed{b\,\leftrightarrow\bullet} \leftarrow \boxed{\quad \bullet} \, , \ \bullet \to \boxed{e\,\leftrightarrow} \leftarrow \boxed{\quad \bullet} \, , \ \mathcal{A}_{\exists X} \right)$$

$$\mathcal{A}_{\exists X} = \left(\bullet, \exists, \left\{ (\bullet \to \boxed{\bullet\!\leftrightarrow x} \leftarrow \boxed{\bullet\!\leftrightarrow x}, \ \mathrm{true}_X) \right\} \right)$$

Here $X = \boxed{\bullet\!\leftrightarrow x}$. In this case, an a-loop can be replaced with an e-loop if an x-loop is present, ensured by requiring (and retaining) it in the rule R_1. A b-loop can also be replaced with an e-loop, also if an x-loop is present, this time ensured by an application condition.

Now consider the conditional relation $R = \{(\boxed{a\leftrightarrow\bullet}, \boxed{b\leftrightarrow\bullet}, \mathrm{true})\} \cup \{(G, G, \mathrm{true}) \mid G \text{ is a graph}\}^{[3]}$ (all graphs are seen as cospans with empty interfaces) and the accompanying relation $R' = \{(a;d, b;d) \mid (a, b, \mathcal{C}) \in R \wedge d \models \mathcal{C}\}$.

Clearly, the graphs $\boxed{a\leftrightarrow\bullet}$ and $\boxed{b\leftrightarrow\bullet}$ are bisimilar under all contexts, and therefore R' is a bisimulation: either the context contains an x-loop, then they both reduce to a graph that contains $\boxed{e\leftrightarrow\bullet\leftrightarrow x}$ and possibly further context (both steps reach the same graph), or the context does not contain an x-loop, in which case neither rule is applicable.

However, R is not a conditional bisimulation, the violating triple being $(\boxed{a\leftrightarrow\bullet}, \boxed{b\leftrightarrow\bullet}, \mathrm{true})$: the step $\boxed{b\leftrightarrow\bullet} \xrightarrow{\mathrm{id},\ \mathcal{A}_{\exists X}}_C \boxed{e\leftrightarrow\bullet}$ cannot be answered by $\boxed{a\leftrightarrow\bullet}$, since in R_1, the x-loop is directly participating in the reaction, but $\mathcal{A}_{\exists X}$ only guarantees its existence in a passive environment (i.e. it is *not* participating in the reaction). ($\boxed{a\leftrightarrow\bullet}$ could only do a step by borrowing $\boxed{\bullet\!\leftrightarrow x}$, but this does not constitute a valid answering step for the step of $\boxed{b\leftrightarrow\bullet}$ where id (i.e. no additional elements) has been borrowed.) $\qquad \lrcorner$

Next, we will show that conditional bisimilarity $\mathrel{\overset{\circ}{\sim}}_C$ is a conditional congruence. This is an important plausibility check, since reactive systems have been introduced with the express purpose to define and reason about bisimulation congruences.

Lemma 4.11. *Conditional bisimilarity $\mathrel{\overset{\circ}{\sim}}_C$ is a conditional congruence.*

Proof. We show that $\mathrel{\overset{\circ}{\sim}}_C$ is:

reflexive: We prove that $R = \{(a, a, \mathcal{C}) \mid \mathrm{codom}(a) = \mathrm{Ro}(\mathcal{C})\}$ is a conditional bisimulation.

---

[3] The reflexive triples (G, G, true) are needed because $\boxed{a\leftrightarrow\bullet}$, $\boxed{b\leftrightarrow\bullet}$ can both be transformed to $\boxed{e\leftrightarrow\bullet}$ in the presence of $\boxed{\bullet\!\leftrightarrow x}$ and we require that the resulting (identical) graphs are related.

Any context step $a \xrightarrow{f,\mathcal{A}}_C a'$ can be trivially answered by the exact same step, setting $\mathcal{B}_1 := \mathcal{A}$, $\mathcal{C}_1' := \mathcal{C}_{\downarrow f}$, $b_1' := a'$, where $I = \{1\}$, and we have $(a', b_1', \mathcal{C}_1') = (a', a', \mathcal{C}_{\downarrow f}) \in R$.

symmetric: Let R be a conditional bisimulation. It is easily seen that R^{-1}, due to the symmetric nature of the definition, is also a conditional bisimulation. Then, $R \subseteq \overset{\circ}{\sim}_C$ implies $R^{-1} \subseteq \overset{\circ}{\sim}_C$, which proves symmetry of $\overset{\circ}{\sim}_C$.

transitive: Let R_1, R_2 be conditional bisimulations that are closed under condition strengthening, i.e. $(a, b, \mathcal{C}') \in R_i$ and $\mathcal{C} \models \mathcal{C}'$ implies $(a, b, \mathcal{C}) \in R_i$. We show that $R_1 R_2 := \{(a, c, \mathcal{D}) \mid \text{there exists } b \text{ such that } (a, b, \mathcal{D}) \in R_1 \text{ and } (b, c, \mathcal{D}) \in R_2\}$ is a conditional bisimulation.

Then, since $\overset{\circ}{\sim}_C$ is a conditional bisimulation closed under condition strengthening (Remark 4.8), $(a, b, \mathcal{D}) \in \overset{\circ}{\sim}_C, (b, c, \mathcal{D}) \in \overset{\circ}{\sim}_C$ implies $(a, c, \mathcal{D}) \in \overset{\circ}{\sim}_C \overset{\circ}{\sim}_C$ and, since we show that $\overset{\circ}{\sim}_C \overset{\circ}{\sim}_C$ is a conditional bisimulation, a, c are conditionally bisimilar under $\mathcal{D}$, which proves transitivity of $\overset{\circ}{\sim}_C$.

Consider a triple $(a, c, \mathcal{D}) \in R_1 R_2$, which by construction of $R_1 R_2$ results from some $(a, b, \mathcal{D}) \in R_1$, $(b, c, \mathcal{D}) \in R_2$. Also consider a step $a \xrightarrow{f,\mathcal{A}}_C a'$. Then, $R_1 R_2$ fulfills the requirements of a conditional bisimulation:

(1) *Answering steps* $c \xrightarrow{f,\mathcal{C}_{i,j}}_C c_{i,j}'$: Since $(a, b, \mathcal{D}) \in R_1$ and R_1 is a conditional bisimulation, we know that there exist answering steps $b \xrightarrow{f,\mathcal{B}_i}_C b_i'$ such that for all $i \in I$, $(a', b_i', \mathcal{D}_i') \in R_1$ and $\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{D}_i' \wedge \mathcal{B}_i)$.

Additionally, since $(b, c, \mathcal{D}) \in R_2$, for each $b \xrightarrow{f,\mathcal{B}_i}_C b_i'$ there exist answering steps $c \xrightarrow{f,\mathcal{C}_{i,j}}_C c_{i,j}'$ such that for all $j \in J_i$ we have $(b_i', c_{i,j}', \mathcal{D}_{i,j}') \in R_2$ and $\mathcal{B}_i \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{j \in J_i} (\mathcal{D}_{i,j}' \wedge \mathcal{C}_{i,j})$.

We now collect all answering steps $c \xrightarrow{f,\mathcal{C}_{i,j}}_C c_{i,j}'$ and use them as answering steps for the original step $a \xrightarrow{f,\mathcal{A}}_C a'$.

(2) $(a', c_{i,j}', \mathcal{D}_{i,j}') \in R_1 R_2$: Since $(a', b_i', \mathcal{D}_i') \in R_1$, and R_1 is closed under condition strengthening, and $\mathcal{D}_{i,j}' \wedge \mathcal{D}_i' \models \mathcal{D}_i'$, we also have $(a', b_i', \mathcal{D}_{i,j}' \wedge \mathcal{D}_i') \in R_1$ for all $j \in J_i$. Similarly, we obtain $(b_i', c_{i,j}', \mathcal{D}_{i,j}' \wedge \mathcal{D}_i') \in R_2$. By construction of $R_1 R_2$ we then also have $(a', c_{i,j}', \mathcal{D}_{i,j}' \wedge \mathcal{D}_i') \in R_1 R_2$.

(3) $\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{D}_i' \wedge \mathcal{C}_i')$: We now rewrite

$$\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{D}_i' \wedge \mathcal{B}_i) \qquad \text{(given from } (a, b, \mathcal{D}) \in R_1)$$

$$\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{D}_i' \wedge \mathcal{B}_i) \wedge \mathcal{D}_{\downarrow f} \quad \text{(since } \mathcal{D}_{\downarrow f} \text{ is given on the left already)}$$

$$\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{D}_i' \wedge \mathcal{B}_i \wedge \mathcal{D}_{\downarrow f}) \quad \text{(distributivity)}$$

From $(b, c, \mathcal{D}) \in R_2$ we know $\mathcal{B}_i \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{j \in J_i} (\mathcal{D}_{i,j}' \wedge \mathcal{C}_{i,j})$, therefore this also implies:

$$\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I} \left(\mathcal{D}_i' \wedge \bigvee_{j \in J_i} (\mathcal{D}_{i,j}' \wedge \mathcal{C}_{i,j})\right)$$

$$\mathcal{A} \wedge \mathcal{D}_{\downarrow f} \models \bigvee_{i \in I, j \in J_i} (\mathcal{D}_i' \wedge \mathcal{D}_{i,j}' \wedge \mathcal{C}_{i,j})$$

Observe that this implication is of the required form for the previously derived tuples $(a', c_{i,j}', \mathcal{D}_{i,j}' \wedge \mathcal{D}_i') \in R_1 R_2$.

In case I is infinite, the same idea can be applied, but we need to slightly change the notation to prevent the creation of infinite disjunctions:

$$c \models (\mathcal{A} \wedge \mathcal{D}_{\downarrow f}) \implies \exists i \in I : c \models (\mathcal{D}'_i \wedge \mathcal{B}_i) \qquad \text{(since } (a, b, \mathcal{D}) \in R_1)$$

$$c \models (\mathcal{A} \wedge \mathcal{D}_{\downarrow f}) \implies \exists i \in I : c \models (\mathcal{D}'_i \wedge \mathcal{B}_i) \wedge c \models \mathcal{D}_{\downarrow f}(\mathcal{D}_{\downarrow f} \text{ on left})$$

$$c \models (\mathcal{A} \wedge \mathcal{D}_{\downarrow f}) \implies \exists i \in I : c \models (\mathcal{D}'_i \wedge \mathcal{B}_i \wedge \mathcal{D}_{\downarrow f}) \qquad \text{(distributivity)}$$

From $(b, c, \mathcal{D}) \in R_2$ we know $c \models (\mathcal{B}_i \wedge \mathcal{D}_{\downarrow f}) \implies \exists j \in J_i : (\mathcal{D}'_{i,j} \wedge \mathcal{C}_{i,j})$, therefore this also implies:

$$c \models (\mathcal{A} \wedge \mathcal{D}_{\downarrow f}) \implies \exists i \in I : c \models \mathcal{D}'_i \wedge \exists j \in J_i : c \models (\mathcal{D}'_{i,j} \wedge \mathcal{C}_{i,j})$$

$$c \models (\mathcal{A} \wedge \mathcal{D}_{\downarrow f}) \implies \exists i \in I, j \in J_i : c \models (\mathcal{D}'_i \wedge \mathcal{D}'_{i,j} \wedge \mathcal{C}_{i,j})$$

Symmetrically, steps $c \xrightarrow{f, \mathcal{C}}_C c'$ can be answered by a. Therefore, $R_1 R_2$ is a conditional bisimulation.

closed under contextualization: We show that $u(R) = \{(a;d, b;d, \mathcal{C}_{\downarrow d}) \mid (a, b, \mathcal{C}) \in R\}$ is a conditional bisimulation, assuming R is a conditional bisimulation.

Consider a triple $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in u(R)$ and a step $a;d \xrightarrow{f, \mathcal{A}}_C a'$. This step is due to some rule $(\ell, r, \mathcal{R})$. We have to show that there exist answering steps $b;d \xrightarrow{f, \mathcal{B}_i}_C b'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in u(R)$ and $\mathcal{A} \wedge (\mathcal{C}_{\downarrow d})_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$.

According to Lemma 3.12, the given step can be rewritten to $a \xrightarrow{d;f, \mathcal{A}}_C a'$.

By construction of $u(R)$, for the given triple $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in u(R)$ there must exist a triple $(a, b, \mathcal{C}) \in R$. As R is a conditional bisimulation, the step $a \xrightarrow{d;f, \mathcal{A}}_C a'$ has answering steps $b \xrightarrow{d;f, \mathcal{B}_i}_C b'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow d;f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$.

However, we are not interested in answering steps for the context $d;f$, but rather for the original step $a;d \xrightarrow{f, \mathcal{A}}_C a'$.

(1) *Answering steps* $b;d \xrightarrow{f, \mathcal{B}_i}_C b'_i$: According to Lemma 3.12, the answering steps $b \xrightarrow{d;f, \mathcal{B}_i}_C b'_i$ can be rewritten to $b;d \xrightarrow{f, \mathcal{B}_i}_C b'_i$.

(2) $(a', b'_i, \mathcal{C}'_i) \in u(R)$: Since $(a', b'_i, \mathcal{C}'_i) \in R$, we have $(a';\text{id}, b'_i;\text{id}, \mathcal{C}'_{i \downarrow \text{id}}) = (a', b'_i, \mathcal{C}'_i) \in u(R)$.

(3) $\mathcal{A} \wedge (\mathcal{C}_{\downarrow d})_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$: The implication to be shown is identical to the one that we obtained above, except for $(\mathcal{C}_{\downarrow d})_{\downarrow f}$, which, however, is equivalent to $\mathcal{C}_{\downarrow d;f}$.

Answering steps for $b;d \xrightarrow{f, \mathcal{B}}_C b'$ can be derived analogously. $\qquad \square$

4.2. Alternative Characterization using Fixpoint Theory.
Behavioural equivalences can be characterized as fixpoints of certain functions on complete lattices [PS11b]. Before we provide definitions that characterize conditional bisimulation relations as fixpoints, we provide a quick summary of fixpoint theory. We do not rely on this characterization in the proofs in this section. However, we will use the theory and the alternative definitions for the proofs of up-to techniques in Section 5.

A *complete lattice* is a partially ordered set $(L, \sqsubseteq)$ where each subset $Y \subseteq L$ has an infimum, denoted by $\bigsqcap Y$ and a supremum, denoted by $\bigsqcup Y$. In this paper, the type of lattices that we consider contain relations ordered by inclusion, i.e. the elements of the lattice are relations and thus the functions we consider map relations to relations.

A function $f\colon L \to L$ is *monotone* if for all $l_1, l_2 \in L$, $l_1 \sqsubseteq l_2$ implies $f(l_1) \sqsubseteq f(l_2)$, *idempotent* if $f \circ f = f$, and *extensive* if $l \sqsubseteq f(l)$ for all $l \in L$. When f is monotone, extensive and idempotent it is called an *(upper) closure*. In this case, $f(L) = \{f(l) \mid l \in L\}$ is a complete lattice.

Given some behavioural equivalence, we define a monotone function $f\colon L \to L$ in such a way that its greatest fixpoint νf equals the behavioural equivalence. Behavioural equivalence can then be checked by establishing whether some given element of the lattice $l \in L$ (for instance a relation consisting of a single pair) is under the fixpoint, i.e., if $l \sqsubseteq \nu f$. By Tarski's Theorem [Tar55], $\nu f = \bigsqcup\{x \mid x \sqsubseteq f(x)\}$, i.e., the greatest fixpoint is the supremum of all post-fixpoints. Hence for showing that $l \sqsubseteq \nu f$, it is sufficient to prove that l is under some post-fixpoint l', i.e., $l \sqsubseteq l' \sqsubseteq f(l')$.

Using these preliminaries, we can now give an alternative characterization of conditional bisimulation using fixpoint theory:

Remark 4.12 (Conditional bisimulation function f_C). Consider the complete lattice Condrel, which is the set of all conditional relations ordered by set inclusion. Then, conditional bisimulations can also be seen as post-fixpoints of f_C (i.e. R is a conditional bisimulation if and only if $R \subseteq f_C(R)$), and conditional bisimilarity as the greatest fixpoint of f_C (i.e. $\backsim_C = \nu f_C$), where the monotone function $f_C\colon \text{Condrel} \to \text{Condrel}$ is the conditional bisimulation function defined by

$$f_C(R) := \big\{(a, b, \mathcal{C}) \mid \text{for each } a \xrightarrow{f, \mathcal{A}}_C a' \text{ there exist } b \xrightarrow{f, \mathcal{B}_i}_C b'_i \text{ and conditions } \mathcal{C}'_i$$
$$\text{such that } (a', b'_i, \mathcal{C}'_i) \in R \text{ and } \mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i),$$
$$\text{and vice versa (for each } b \xrightarrow{f, \mathcal{B}}_C b' \dots)\big\}$$

The correctness of this characterization (i.e. that "f_C is the right function") can be seen by expanding the definition of f_C on the right-hand side of $(a, b, \mathcal{C}) \in R \implies (a, b, \mathcal{C}) \in f_C(R)$, which results in exactly the definition of a conditional bisimulation relation. ⌟

4.3. **Representative Conditional Bisimulations.** Checking whether two arrows are conditionally bisimilar, or whether a given relation is a conditional bisimulation, can be hard in practice, since we have to check all possible context steps, of which there are typically infinitely many.

For saturated bisimilarity, we used representative steps instead of context steps (cf. Subsections 2.3 and 2.4) to reduce the number of contexts to be checked. In this section, we extend our definition of conditional bisimulation to use representative steps and prove that the resulting bisimilarity is identical to the one previously defined.

Definition 4.13 (Representative conditional bisimulation). We fix a conditional reactive system. A *representative conditional bisimulation R* is a conditional relation such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each representative step $a \xrightarrow{f, \mathcal{A}}_R a'$, there are answering context steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$; and vice versa. Two arrows are *representative conditionally bisimilar under $\mathcal{C}$* $((a, b, \mathcal{C}) \in \backsim_R)$ whenever a representative conditional bisimulation R with $(a, b, \mathcal{C}) \in R$ exists.

Remark 4.14. Analogously to Remark 4.12, we can define representative conditional bisimulations as post-fixpoints of f_R, and representative conditional bisimilarity as the greatest fixpoint of f_R, where f_R is defined on Condrel as follows:

$$f_R(R) := \big\{(a, b, \mathcal{C}) \mid \text{for each } a \xrightarrow{f,\mathcal{A}}_R a' \text{ there exist } b \xrightarrow{f,\mathcal{B}_i}_C b_i' \text{ and conditions } \mathcal{C}_i'$$
$$\text{such that } (a', b_i', \mathcal{C}_i') \in R \text{ and } \mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I}(\mathcal{C}_i' \wedge \mathcal{B}_i),$$
$$\text{and vice versa (for each } b \xrightarrow{f,\mathcal{B}}_R b' \dots)\big\}$$

It is easy to see that $f_C \subseteq f_R$: Their definitions differ only in the type of steps which are checked. A triple that satisfies the requirements for all context steps (is in $f_C(R)$) naturally satisfies them for all representative steps (is in $f_R(R)$), since every representative step is also a context step. ⌟

To show that the two conditional bisimilarities using context and representative steps are equivalent (Theorem 4.17) and for the proofs of Theorems 5.16, 6.4 and 6.6, we need the following two lemmas:

Lemma 4.15 [HK12, Lemma 16]. *Given a context step* $a \xrightarrow{f,\mathcal{A}}_C a'$, *the borrowed context* f *can be extended by an additional context* c', *that is:* $a \xrightarrow{f,\mathcal{A}}_C a'$ *implies* $a \xrightarrow{f;c',\ \mathcal{A}_{\downarrow c'}}_C a';c'$

Lemma 4.16. *Representative conditional bisimilarity is closed under contextualization, that is,* $(a, b, \mathcal{C}) \in \mathbin{\overset{\circ}{\sim}}_R$ *implies* $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in \mathbin{\overset{\circ}{\sim}}_R$.

Proof. We show that $u(R) = \{(a;d, b;d, \mathcal{C}_{\downarrow d}) \mid (a, b, \mathcal{C}) \in R\}$ is a representative conditional bisimulation, assuming R is a representative conditional bisimulation.

Consider a triple $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in u(R)$ and a step $a;d \xrightarrow{f,\mathcal{A}}_R a'$. This step is due to some rule $(\ell, r, \mathcal{R})$. We have to show that there exist answering steps $b;d \xrightarrow{f,\mathcal{B}_i}_C b_i'$ such that $(a', b_i', \mathcal{C}_i') \in u(R)$ and $\mathcal{A} \wedge (\mathcal{C}_{\downarrow d})_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}_i' \wedge \mathcal{B}_i)$.

The representative step is of course also a context step $(a;d \xrightarrow{f,\mathcal{A}}_C a')$, which, according to Lemma 3.12, can be rewritten to $a \xrightarrow{d;f,\ \mathcal{A}}_C a'$. As a result, the step is not necessarily a representative one anymore. However we can find a matching representative step (see also Remark 3.11):
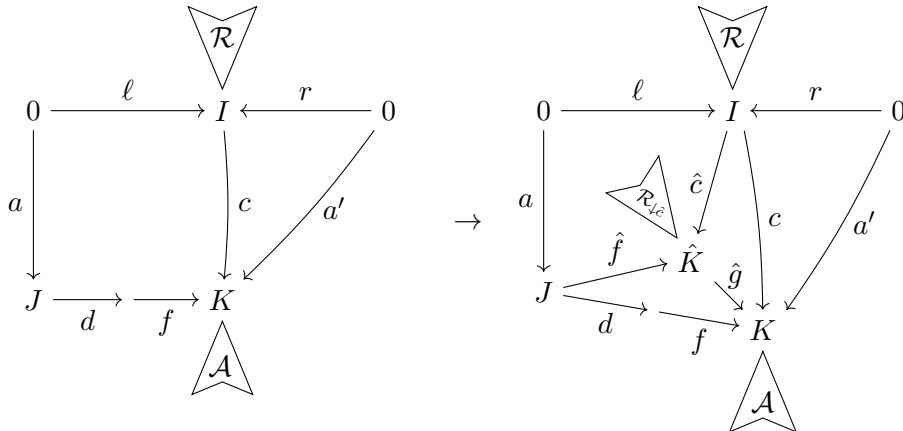
Note that $\hat{f};\hat{g} = d;f$, $\hat{c};\hat{g} = c$. Variables with a hat (e.g. $\hat{c}$) refer to the representative step, but otherwise play the same role than their unhatted counterparts (e.g. c), which refer to the original step. The result is a representative step $a \xrightarrow{\hat{f},\, \mathcal{R}_{\downarrow\hat{c}}}_R r;\hat{c}$.

By construction of $u(R)$, for the given triple $(a;d, b;d, \mathcal{C}_{\downarrow d}) \in u(R)$ there must exist a triple $(a, b, \mathcal{C}) \in R$. As R is a representative conditional bisimulation, the step $a \xrightarrow{\hat{f},\, \mathcal{R}_{\downarrow\hat{c}}}_R r;\hat{c}$ has answering steps $b \xrightarrow{\hat{f},\, \hat{\mathcal{B}}_i}_C \hat{b}'_i$ such that $(r;\hat{c}, \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ and $\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \models \bigvee_{i\in I} \left(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i \right)$.

However we are not interested in answering steps for the representative context $\hat{f}$, but rather for the original step $a;d \xrightarrow{f,\mathcal{A}}_R a'$, that is, we need answering steps of $b;d$ using context f. So we need (1) answering steps $b;d \xrightarrow{f,\mathcal{B}_i}_C b'_i$ (2) such that $(a', b'_i, \mathcal{C}'_i) \in u(R)$ and (3) $\mathcal{A} \wedge (\mathcal{C}_{\downarrow d})_{\downarrow f} \models \bigvee_{i\in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$.

(1) *Answering steps* $b;d \xrightarrow{f,\mathcal{B}_i}_C b'_i$: Lemma 4.15 and $b \xrightarrow{\hat{f},\, \hat{\mathcal{B}}_i}_C \hat{b}'_i$ imply that steps $b \xrightarrow{\hat{f};\hat{g},\, \hat{\mathcal{B}}_{i\downarrow\hat{g}}}_C \hat{b}'_i;\hat{g}$ are possible. Since $d;f = \hat{f};\hat{g}$, these are equivalent to $b \xrightarrow{d;f,\, \hat{\mathcal{B}}_{i\downarrow\hat{g}}}_C \hat{b}'_i;\hat{g}$. We set $b'_i := \hat{b}'_i;\hat{g}$, $\mathcal{B}_i := \hat{\mathcal{B}}_{i\downarrow\hat{g}}$ and get steps $b \xrightarrow{d;f,\, \mathcal{B}_i}_C b'_i$, which, according to Lemma 3.12, are equivalent to steps $b;d \xrightarrow{f,\mathcal{B}_i}_C b'_i$.

(2) $(a', b'_i, \mathcal{C}'_i) \in u(R)$: Set $\mathcal{C}'_i := \hat{\mathcal{C}}'_{i\downarrow\hat{g}}$. By construction of $u(R)$, $(r;\hat{c}, \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ implies $(r;\hat{c};\hat{g}, \hat{b}'_i;\hat{g}, \hat{\mathcal{C}}'_{i\downarrow\hat{g}}) = (a', b'_i, \mathcal{C}'_i) \in u(R)$.

(3) $\mathcal{A} \wedge (\mathcal{C}_{\downarrow d})_{\downarrow f} \models \bigvee_{i\in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$: Above, we already showed $\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \models \bigvee_{i\in I} \left(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i \right)$. By shifting both sides with $\hat{g}$ and applying the rules of Theorem 3.6, we get:

$$\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \models \bigvee_{i\in I} \left(\hat{\mathcal{C}}_i \wedge \hat{\mathcal{B}}_i \right)$$
$$\Rightarrow \left(\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \right)_{\downarrow\hat{g}} \models \left(\bigvee_{i\in I} \left(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i \right) \right)_{\downarrow\hat{g}}$$
$$\Leftrightarrow \mathcal{R}_{\downarrow\hat{c};\hat{g}} \wedge \mathcal{C}_{\downarrow\hat{f};\hat{g}} \models \bigvee_{i\in I} \left(\hat{\mathcal{C}}'_{i\downarrow\hat{g}} \wedge \hat{\mathcal{B}}_{i\downarrow\hat{g}} \right)$$

By substituting $\hat{c};\hat{g} = c$, $\hat{f};\hat{g} = d;f$, $\mathcal{C}'_i = \hat{\mathcal{C}}'_{i\downarrow\hat{g}}$, $\mathcal{B}_i = \hat{\mathcal{B}}_{i\downarrow\hat{g}}$, we obtain $\mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow d;f} \models \bigvee_{i\in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$. Since $\mathcal{A} \models \mathcal{R}_{\downarrow c}$, we have $\mathcal{A} \wedge \mathcal{C}_{\downarrow d;f} \models \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow d;f} \models \bigvee_{i\in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$, which was to be shown.

Analogously, answering steps for $b;d \xrightarrow{f,\mathcal{B}}_R b'$ can be constructed. $\qquad\square$

The following theorem is based on a proof strategy similar to Lemma 4.16.

Theorem 4.17. *Conditional bisimilarity and representative conditional bisimilarity coincide, that is, $\overset{\circ}{\backsim}_C = \overset{\circ}{\backsim}_R$.*

Proof.

- ($\overset{\circ}{\backsim}_C \subseteq \overset{\circ}{\backsim}_R$):

  Consider a triple $(a, b, \mathcal{C}) \in \overset{\circ}{\backsim}_C$. By Definition 4.3, for each step $a \xrightarrow{f,\mathcal{A}}_C a'$ there exist answering steps of b with the requirements listed there; symmetrically, each step $b \xrightarrow{f,\mathcal{B}}_C b'$ can be answered by a. To show that $(a, b, \mathcal{C}) \in \overset{\circ}{\backsim}_R$, the same statement has to be shown for each representative step $a \xrightarrow{f,\mathcal{A}}_R a'$ (and $b \xrightarrow{f,\mathcal{B}}_R b'$). Since it already holds for all

context steps, which are a superset of representative steps (every $\to_R$ step is also a $\to_C$ step), Definition 4.13 is trivially satisfied.

- $(\mathring{\sim}_C \supseteq \mathring{\sim}_R)$:
  By definition of $\mathring{\sim}_C$, if a conditional relation R is a conditional bisimulation, then $R \subseteq \mathring{\sim}_C$. We show that $R = \mathring{\sim}_R$ is a conditional bisimulation, i.e. that it satisfies the requirements of Definition 4.3.

  Consider a triple $(a, b, \mathcal{C}) \in R$ and a context step $a \xrightarrow{f, \mathcal{A}}_C a'$. This step is due to some rule $(\ell, r, \mathcal{R})$. According to Remark 3.11, this context step can be reduced to a representative step $a \xrightarrow{\hat{f}, \mathcal{R}_{\downarrow\hat{c}}}_R r;\hat{c}$, and there exists $\hat{g}$ such that $\hat{f};\hat{g} = f$, $\hat{c};\hat{g} = c$. Again, all variables with a hat (e.g. $\hat{c}$) refer to the representative step, but otherwise take the same role as their unhatted counterparts.

  Since R is the representative conditional bisimilarity, R is a representative conditional bisimulation. Together with $(a, b, \mathcal{C}) \in R$, this means that for the aforementioned step there exist answering steps $b \xrightarrow{\hat{f}, \hat{\mathcal{B}}_i}_C \hat{b}'_i$ and conditions $\hat{\mathcal{C}}'_i$, such that $(r;\hat{c}, \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ and $\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \models \bigvee_{i \in I} (\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i)$.

  However we are not interested in answering steps for the representative step, but rather for the original step $a \xrightarrow{f, \mathcal{A}}_C a'$, that is, we need answering steps of b using context f. So we need (1) answering steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$ (2) such that $(a', b'_i, \mathcal{C}'_i) \in R$ and (3) $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$. Analogously to the proof of Lemma 4.16, we have:

(1) *Answering steps:* Having steps $b \xrightarrow{\hat{f}, \hat{\mathcal{B}}_i}_C \hat{b}'_i$ implies that steps $b \xrightarrow{\hat{f};\hat{g}, \hat{\mathcal{B}}_{i\downarrow\hat{g}}}_C \hat{b}'_i;\hat{g}$ are possible (Lemma 4.15). Rewritten as $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$ (where $\mathcal{B}_i := \hat{\mathcal{B}}_{i\downarrow\hat{g}}$, $b'_i := \hat{b}'_i;\hat{g}$), we get the desired answering steps for the original step.

(2) $(a', b'_i, \mathcal{C}'_i) \in R$: Since R is the representative conditional bisimilarity, by Lemma 4.16 we know that R is closed under contextualization. Therefore, $(r;\hat{c}, \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ implies $(r;\hat{c};\hat{g}, \hat{b}'_i;\hat{g}, \hat{\mathcal{C}}'_{i\downarrow\hat{g}}) = (a', b'_i, \mathcal{C}'_i) \in R$, i.e. the original target a' is conditionally bisimilar to the targets $\hat{b}'_i;\hat{g}$ of the answering steps.

(3) $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}'_i \wedge \mathcal{B}_i)$: Using the rules of Theorem 3.6 we get:

$$\mathcal{R}_{\downarrow\hat{c}} \wedge \mathcal{C}_{\downarrow\hat{f}} \models \bigvee \left(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i \right)$$
$$\Rightarrow (\mathcal{R}_{\downarrow\hat{c}})_{\downarrow\hat{g}} \wedge (\mathcal{C}_{\downarrow\hat{f}})_{\downarrow\hat{g}} \models \bigvee \left(\hat{\mathcal{C}}'_{i\downarrow\hat{g}} \wedge \hat{\mathcal{B}}_{i\downarrow\hat{g}} \right)$$
$$\Leftrightarrow \mathcal{R}_{\downarrow\hat{c};\hat{g}} \wedge \mathcal{C}_{\downarrow\hat{f};\hat{g}} \models \bigvee \left(\hat{\mathcal{C}}'_{i\downarrow\hat{g}} \wedge \hat{\mathcal{B}}_{i\downarrow\hat{g}} \right)$$
$$\Leftrightarrow \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow f} \models \bigvee (\mathcal{C}'_i \wedge \mathcal{B}_i)$$

  Since $\mathcal{A} \models \mathcal{R}_{\downarrow c}$, we have $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow f} \models \bigvee (\mathcal{C}'_i \wedge \mathcal{B}_i)$, which is the required condition for the triple $(a, b, \mathcal{C})$.

Analogously, we can construct answering steps for $b \xrightarrow{f, \mathcal{B}}_C b'$. We have therefore shown that $R = \mathring{\sim}_R$ is a conditional bisimulation and therefore $\mathring{\sim}_R \subseteq \mathring{\sim}_C$. □

We now discuss the notion of representative condition bisimulation in two examples.

Example 4.18 (Message passing over unreliable channels, continued)**.** Consider the reactive system of Example 4.7. There exists a representative conditional bisimulation R such that

$(\emptyset \to \boxed{\bullet \xrightarrow{rel} \bullet} \leftarrow \bullet\ \bullet,\ \emptyset \to \boxed{\bullet \xrightarrow{unr} \bullet} \leftarrow \bullet\ \bullet,\ \mathcal{A}_{\overline{n}}) \in R$ (where $\mathcal{A}_{\overline{n}}$ requires that no *noise*-edge exists).

We consider the representative steps that are possible from either $\boxed{\bullet \xrightarrow{rel} \bullet}$ or $\boxed{\bullet \xrightarrow{unr} \bullet}$ and only explain the most interesting cases (cf. Figure 10a).

- The graph $\boxed{\bullet \xrightarrow{rel} \bullet}$ can do a step using rule P_R by borrowing a message on the left node, that is, $f = \bullet\ \bullet \to \boxed{m \curvearrowright \bullet\ \bullet} \leftarrow \bullet\ \bullet$, reacting to $\boxed{\bullet \xrightarrow{rel} \bullet \curvearrowright m}$. No further restrictions on the environment are necessary, so $\mathcal{A} = \text{true}$. The graph $\boxed{\bullet \xrightarrow{unr} \bullet}$ can answer this step using P_U and reacts to $\boxed{\bullet \xrightarrow{unr} \bullet \curvearrowright m}$, but only if no noise is present (environment satisfies $\mathcal{B}_i = \mathcal{A}_{\overline{n}}$). We evaluate the implication $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \equiv \text{true} \wedge \mathcal{A}_{\overline{n}\downarrow f} \equiv \mathcal{A}_{\overline{n}} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{A}_{\overline{n}}) \equiv \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$, setting $\mathcal{C}'_i = \mathcal{A}_{\overline{n}}$. (Note that $\mathcal{A}_{\overline{n}\downarrow f} \equiv \mathcal{A}_{\overline{n}}$ since $\mathcal{A}_{\overline{n}}$ forbids the existence of an *noise*-edge between the two interface nodes and f is unrelated, providing an m-loop on the left-hand node.) We now require $(\emptyset \to \boxed{\bullet \xrightarrow{rel} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \emptyset \to \boxed{\bullet \xrightarrow{unr} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \mathcal{A}_{\overline{n}}) \in R$.
- Symmetrically, $\boxed{\bullet \xrightarrow{unr} \bullet}$ can do a step using P_U by borrowing a message on the left node, reacting to $\boxed{\bullet \xrightarrow{unr} \bullet \curvearrowright m}$ in an environment without noise ($\mathcal{A} = \mathcal{A}_{\overline{n}}$). $\boxed{\bullet \xrightarrow{rel} \bullet}$ can answer this step under any condition $\mathcal{B}_i$. Then, the implication is satisfied if we set $\mathcal{C}'_i = \mathcal{A}_{\overline{n}}$, so we require again $(\emptyset \to \boxed{\bullet \xrightarrow{rel} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \emptyset \to \boxed{\bullet \xrightarrow{unr} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \mathcal{A}_{\overline{n}}) \in R$.
- There are additional representative steps that differ in how much of the left-hand side is borrowed, but can be proven analogously to the two previously discussed steps.
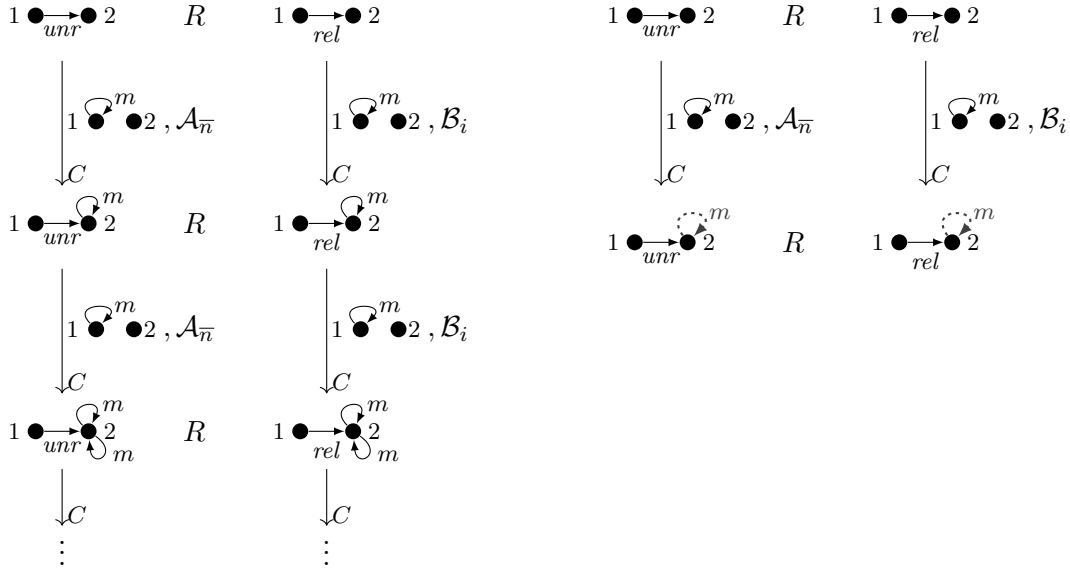
(a) Showing conditional bisimilarity may require an infinite number of steps

(b) Removing unrelated context (dotted edge) before relating the pair would allow stopping after a single step

Figure 10: Showing conditional bisimilarity of two types of channels using the rules of Example 4.7

This means we have to add the pair $(\emptyset \to \boxed{\bullet \xrightarrow{rel} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \emptyset \to \boxed{\bullet \xrightarrow{unr} \bullet \curvearrowright m} \leftarrow \bullet\ \bullet,\ \mathcal{A}_{\overline{n}})$ to R and to continue adding pairs until we obtain a bisimulation: with every step, a new triple with an additional m-loop on the right node is added to the relation, therefore, the smallest conditional bisimulation has infinite size. This is visualized in Figure 10a.

However, except for the additional m-loop on the right node, which does not affect rule application, this pair is identical to the initial one and we can hence use a similar argument. In Section 5 we show how to make this formal, using up-to technique. In summary, we conclude that $\bullet\!\xrightarrow{rel}\!\bullet$ is conditionally bisimilar to $\bullet\!\xrightarrow{unr}\!\bullet$ under the condition $\mathcal{A}_{\overline{n}}$.   ⌟

Example 4.19 (Unreliable channel vs. no channel)**.** For Examples 4.7 and 4.18, it can also be shown that under the condition $\neg\mathcal{A}_{\overline{n}}$, the unreliable channel $\emptyset \to \bullet\!\xrightarrow{unr}\!\bullet \leftarrow \bullet\;\bullet$ is conditionally bisimilar to not having a channel between the two nodes ($\emptyset \to \bullet\;\bullet \leftarrow \bullet\;\bullet$).

In this case, $\bullet\!\xrightarrow{unr}\!\bullet$ can still do a reaction under $\mathcal{A}_{\overline{n}}$. Then, $\bullet\;\bullet$ can answer with an empty set of steps. The implication $\mathcal{A}_{\overline{n}}\wedge\mathcal{C}_{\downarrow f} \models \bigvee_{i\in I}\left(\mathcal{C}'_i \wedge \mathcal{B}_i\right)$ is then simplified to $\mathcal{A}_{\overline{n}}\wedge\neg\mathcal{A}_{\overline{n}} \models$ false, which is easily seen to be valid.   ⌟

5. Up-to Techniques for Proving Conditional Bisimilarity

Our optimizations so far involved replacing context steps by representative steps, which ensure finite branching and thus greatly reduce the proof obligations for a single step. However, it can still happen very easily that the smallest possible bisimulation is of infinite size, in which case automated proving of conditional bisimilarity becomes impossible. For instance, in Example 4.18, the least conditional bisimulation relating the two cospans u, r (representing (un)reliable channels) under $\mathcal{A}_{\overline{n}}$ contains infinitely many triples $(u;m^n, r;m^n, \mathcal{A}_{\overline{n}})$ for any number n of messages on the right node ($m = \bullet\;\bullet \to \bullet\;\bullet\!\!\curvearrowright^{m} \leftarrow \bullet\;\bullet$).

On the other hand, conditional bisimilarity is closed under contextualization, hence if u, r are related, we can conclude that $u;m$ and $r;m$ must be related as well. Intuitively the relation $R = \{(u, r, \mathcal{A}_{\overline{n}})\}$ is a sufficient witness, since after one step we reach the triple $(u;m, r;m, \mathcal{A}_{\overline{n}})$, from which we can "peel off" a common context m to obtain a triple already contained in R (visualized in Figure 10b).

This is an instance of an *up-to technique*, which can be used to obtain smaller witness relations by identifying and removing redundant elements from a bisimulation relation. Instead of requiring the redundant triple $(u;m, r;m, \mathcal{A}_{\overline{n}})$ to be contained in the relation, it is sufficient to say that *up to* the passive context m, the triple is represented by $(u, r, \mathcal{A}_{\overline{n}})$, which is already contained in the relation. In particular, this specific up-to technique is known as *up-to context* [PS11b], a well-known proof technique for process calculi.[4]

Note that in general, a bisimulation up-to context is not a bisimulation relation. However, it can be converted into a bisimulation by closing it under all contexts.

In this section, we show how to adapt this concept to conditional bisimilarity and in particular discuss how to deal with the conditions in a conditional bisimulation up-to context.

5.1. **Up-To Techniques and Fixpoint Theory.** As in Section 4, we will provide defini-tions and proofs that are based on fixpoint theory. Hence, we first introduce the remaining preliminary concepts for implementing up-to techniques using fixpoint theory, again mostly following [PS11b].

In Subsection 4.2 we already explained that to show that some element l of the lattice is contained in behavioural equivalence ($l \sqsubseteq \nu f$), it is sufficient to prove that l is under some post-fixpoint l' ($l \sqsubseteq l' \sqsubseteq f(l')$). The idea of using up-to techniques is now to define

---

[4]Stated in the language of process algebra, a symmetric relation R is a bisimulation up-to context, if whenever $(P,Q) \in R$ and $P \xrightarrow{a} P'$, then $Q \xrightarrow{a} Q'$, where $P' = C[P'']$, $Q' = C[Q'']$ and $(P'',Q'') \in R$.

a monotone function u (the up-to function) and check if $l \sqsubseteq \nu(f \circ u)$ by showing that l is a post-fixpoint of $f \circ u$. Typically, the characteristics of u should make it easier to prove $l \sqsubseteq f(u(l))$ than proving $l \sqsubseteq f(l)$. This is clearly the case when u is extensive, since extensiveness of u and monotonicity of f implies $f(l) \sqsubseteq f(u(l))$ and thus obtaining $l \sqsubseteq f(u(l))$ is easier than obtaining $l \sqsubseteq f(l)$.

Naturally, for the up-to technique to be useful, it also has to be shown that $\nu(f \circ u) \sqsubseteq \nu f$:

Definition 5.1 (sound up-to function). Let L be a complete lattice and let $f \colon L \to L$ be a monotone function. A *sound up-to function* for f is any monotone function $u \colon L \to L$ such that $\nu(f \circ u) \sqsubseteq \nu f$.

Instead of soundness, we will use the stronger notion of compatibility:

Definition 5.2 (f-compatibility of u). Let L be a complete lattice and let $f \colon L \to L$ be a monotone function. A monotone function $u \colon L \to L$ is f-*compatible* if $u \circ f \sqsubseteq f \circ u$.

Compatibility has several advantages over soundness: First, compatibility implies soundness, second, f-compatible up-to techniques can be combined with each other to obtain more powerful proof techniques, and third, it implies that the up-to function preserves the greatest fixpoint (a kind of congruence result):

Lemma 5.3 (Compatibility implies soundness). *Let $f \colon L \to L$ be a monotone function and let $u \colon L \to L$ be an f-compatible closure. Then $\nu f = \nu(f \circ u)$.*

Proof. $\nu(f \circ u) \sqsubseteq \nu f$ by [PS11a, Theorem 6.3.9]. $\nu f \sqsubseteq \nu(f \circ u)$ because u is a closure and therefore extensive. $\qquad\square$

Note that the first inclusion is sufficient to show soundness, but we will later also use the second one to simplify some of the proofs.

This lemma gives rise to the following proof rule: to show that l is in the behavioural equivalence ($l \sqsubseteq \nu f$), it is sufficient to show that l is a post-fixpoint of $f \circ u$ (i.e., $l \sqsubseteq f(u(l))$), which implies $l \sqsubseteq \nu(f \circ u) = \nu f$.

Lemma 5.4 (Compositionality of f-compatible functions [Pou07, Proposition 1.6]). *If u_1, u_2 are f-compatible, then $u_1 \circ u_2$ is also f-compatible.*

Proposition 5.5. *Let $f \colon L \to L$ be a monotone function and let $u \colon L \to L$ be an f-compatible closure. Then $u(\nu f) = \nu f$.*

Proof. $\nu f \sqsubseteq u(\nu f)$ holds because u is a closure and therefore extensive.
Since u is f-compatible, [Pou07, Remark 1.5] guarantees $u(\nu f) \sqsubseteq \nu f$. $\qquad\square$

We will also need the following (straightforward) result.

Lemma 5.6 (Characterization of post-fixpoints of $f \circ u$). *Let $f \colon L \to L$ be a monotone function, let $u \colon L \to L$ be an f-compatible closure, and let $l \in L$. Then, $l \sqsubseteq f(u(l)) \iff u(l) \sqsubseteq f(u(l))$.*

Proof.

- ($l \sqsubseteq f(u(l)) \Rightarrow u(l) \sqsubseteq f(u(l))$)**:**
  Since u is monotone, $l \sqsubseteq f(u(l))$ implies $u(l) \sqsubseteq u(f(u(l)))$. By f-compatibility of u ($u \circ f \sqsubseteq f \circ u$), then also $u(l) \sqsubseteq f(u(u(l)))$, which by idempotence of u is equivalent to $u(l) \sqsubseteq f(u(l))$.

- $\big(l \sqsubseteq f(u(l)) \Leftarrow u(l) \sqsubseteq f(u(l))\big)$**:**
  u is extensive, therefore $l \sqsubseteq u(l)$. Combined, we obtain $l \sqsubseteq u(l) \sqsubseteq f(u(l))$ and hence our desired result. $\qquad\square$

5.2. **Conditional Bisimilarity Up-To Context.** We start our investigation of conditional bisimilarity up-to context with the idea of a relation that can be extended to a conditional bisimulation. To show, using such a conditional bisimulation up-to context R, that a pair of arrows is conditionally bisimilar, it is not in general necessary to find this pair in R, but one can instead extend a pair in R to the pair under review. As this extension might provide parts of the context that the original condition referred to, it is necessary to shift the associated condition over the extension.

Definition 5.7 (Conditional bisimulation up-to context (CBUC)). A conditional relation R is a *conditional bisimulation up-to context* if the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each context step $a \xrightarrow{f,\mathcal{A}}_C a'$, there are answering steps $b \xrightarrow{f,\mathcal{B}_i}_C b'_i$, $i \in I$, and conditions $\mathcal{C}''_i$ such that for each $i \in I$ there exists $(a''_i, b''_i, \mathcal{C}''_i) \in R$ with $a' = a''_i;j_i$, $b'_i = b''_i;j_i$ for some arrow j_i and additionally $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} \big(\mathcal{C}''_{i\downarrow j_i} \wedge \mathcal{B}_i\big)$; vice versa for steps $b \xrightarrow{f,\mathcal{B}}_C b'$.

The situation for one answer step is depicted in Figure 11. The weakest possible $\mathcal{A}, \mathcal{B}_i$ can be derived from the rule conditions as $\mathcal{A} = \mathcal{R}_{\downarrow c}$, $\mathcal{B}_i = \mathcal{R}_{i\downarrow e_i}$.
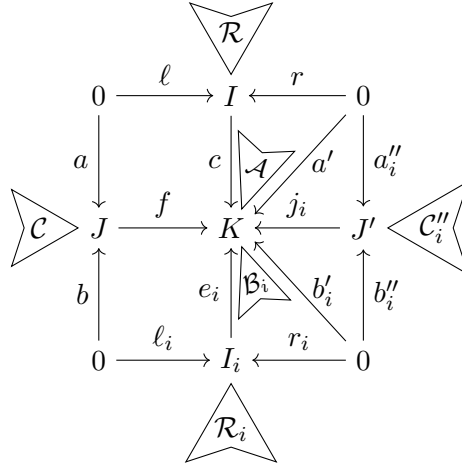

Figure 11: A single answer step in conditional bisimulation up-to context

Compared to a regular conditional bisimulation, which directly relates the results of the answering steps $(a', b'_i, \mathcal{C}'_i)$, in a CBUC it is sufficient to relate some pair $(a''_i, b''_i, \mathcal{C}''_i)$, where a''_i, b''_i are obtained from a', b'_i by removing an identical context j_i. (The conditional bisimilarity of the actual successors a', b'_i can then be derived by contextualizing the relation, i.e., we use Theorem 5.10 and refer to a triple in $u(R)$ that is contextualized under j_i.)

Remark 5.8. A CBUC can also be defined based on the closure under contextualization u (see Definition 4.2, note that u is easily seen to be a closure[5]): A conditional relation R is a

---

[5]Monotonicity: trivial. Extensiveness: $(a, b, \mathcal{C}) \in R$ implies $(a; \mathrm{id}, b; \mathrm{id}, \mathcal{C}_{\downarrow \mathrm{id}}) = (a, b, \mathcal{C}) \in u(R)$. Idempotence: $u \subseteq u \circ u$ by extensiveness and monotonicity, $u \circ u \subseteq u$ because subsequent contextualization first under d_1 then d_2 can also be done in a single step of $d_1; d_2$.

CBUC if and only if $R \subseteq f_C(u(R))$ (i.e. R is a post-fixpoint of $f_C \circ u$). This can be seen by expanding the definitions of f_C and u on the right-hand side of $(a, b, \mathcal{C}) \in R \implies (a, b, \mathcal{C}) \in f_C(u(R))$, which results in exactly the definition of a CBUC. ⌟

We now show that this up-to technique is useful or *sound* (Definition 5.1), that is, all elements recognized as bisimilar by the up-to technique are actually bisimilar [San98, PS11b]. In fact, we prove the stronger result that the technique is f_C-compatible, which (as outlined in Subsection 4.2) not only implies soundness, but also makes it possible to combine our technique with other f_C-compatible up-to techniques.

Theorem 5.9 (u is f_C-compatible)**.** *Let R be a conditional relation. Then it holds that $u(f_C(R)) \subseteq f_C(R)$. This implies that $u(f_C(R)) \subseteq f_C(u(R))$, i.e., u is f_C-compatible.*

Proof. We show that $(a, b, \mathcal{C}) \in u(f_C(R))$ implies $(a, b, \mathcal{C}) \in f_C(R)$.

Within this proof, we use the following notation: Given arrows x, d, $x_{\backslash d}$ indicates an arrow such that $x = x_{\backslash d};d$. Given a condition $\mathcal{C}$, $\mathcal{C}_{\uparrow d}$ is a condition such that $\mathcal{C} = (\mathcal{C}_{\uparrow d})_{\downarrow d}$.

- Since $(a, b, \mathcal{C}) \in u(f_C(R))$, by definition of u this means that

  there exist $d, a_{\backslash d}, b_{\backslash d}, \mathcal{C}_{\uparrow d}$ such that $a = a_{\backslash d};d$, $b = b_{\backslash d};d$, $\mathcal{C} = (\mathcal{C}_{\uparrow d})_{\downarrow d}$, and $(a_{\backslash d}, b_{\backslash d}, \mathcal{C}_{\uparrow d}) \in f_C(R)$

- This means that

  for all steps $a_{\backslash d} \xrightarrow{f, \mathcal{A}}_C a'$ there exist answering steps $b_{\backslash d} \xrightarrow{f, \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge (\mathcal{C}_{\uparrow d})_{\downarrow f} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$; vice versa for steps of $b_{\backslash d}$

- Some of the borrowed contexts f for which this statement holds are of the shape $f = d;f'$, therefore, we also know that

  for all steps $a_{\backslash d} \xrightarrow{d;f', \mathcal{A}}_C a'$ there exist answering steps $b_{\backslash d} \xrightarrow{d;f', \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge (\mathcal{C}_{\uparrow d})_{\downarrow d;f'} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$; vice versa for steps of $b_{\backslash d}$

- Applying Lemma 3.12 and Theorem 3.6 to this statement, we obtain:

  for all steps $a_{\backslash d};d \xrightarrow{f', \mathcal{A}}_C a'$ there exist answering steps $b_{\backslash d};d \xrightarrow{f', \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge ((\mathcal{C}_{\uparrow d})_{\downarrow d})_{\downarrow f'} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$; vice versa for steps of $b_{\backslash d};d$

- Applying the equalities for the variables annotated with $_{\backslash d}$ and $_{\uparrow d}$, we get:

  for all steps $a \xrightarrow{f', \mathcal{A}}_C a'$ there exist answering steps $b \xrightarrow{f', \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ such that $(a', b'_i, \mathcal{C}'_i) \in R$ and $\mathcal{A} \wedge (\mathcal{C})_{\downarrow f'} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$; vice versa for steps of b

  This is exactly the definition of $(a, b, \mathcal{C}) \in f_C(R)$. Therefore, $u(f_C(R)) \subseteq f_C(R)$.

  Since $R \subseteq u(R)$ due to extensiveness of u, we can infer $f_C(R) \subseteq f_C(u(R))$ since f_C is monotone. Combined, this gives us f_C-compatibility of u. □

Note the stronger result $(u(f_C(R)) \subseteq f_C(R)$ instead of just $u(f_C(R)) \subseteq f_C(u(R)))$ can intuitively be explained as follows: since f_C quantifies over all context steps and the size of the borrowed context f is not bounded, this means the successor triples are already closed under contextualization.

From compatibility, we obtain as a corollary that this up-to technique is useful or *sound*, that is, all elements recognized as bisimilar by the up-to technique are actually bisimilar (see Lemma 5.3 and [San98, PS11b]).

Theorem 5.10 (Characterization of CBUC). *A conditional relation R satisfies Definition 5.7 (i.e. it is a CBUC) if and only if its closure under contextualization $u(R)$ is a conditional bisimulation.*

Proof. R satisfying Definition 5.7 is, by Remark 5.8, equivalent to R being a post-fixpoint of $f_C \circ u$, i.e., $R \subseteq f_C(u(R))$. Also, R satisfying the definition from Theorem 5.10, i.e., $u(R)$ being a conditional bisimulation, is, by Remark 4.12, equivalent to $u(R) \subseteq f_C(u(R))$.

Since u is f_C-compatible and u is a closure, we can instantiate Lemma 5.6 to obtain the desired result. Hence, every relation that our initial definition recognizes as a CBUC indeed represents a conditional bisimulation (when closed under contextualization), and all relations that should intuitively be a CBUC are recognized by Definition 5.7 as such. □

Remark 5.11. From Theorem 5.10 we easily obtain as a corollary that every CBUC R is contained in $\backsim_C$ ($R \subseteq \backsim_C$), i.e. all elements contained in some CBUC are indeed conditionally bisimilar. This follows from the fact that $R \subseteq u(R)$ (set $d = \mathrm{id}_J$) and $u(R) \subseteq \backsim_C$ (since by Theorem 5.10 $u(R)$ is a conditional bisimulation). ⌟

Note that while Theorem 5.10 gives a more accessible definition of CBUCs than Definition 5.7, the latter definition is more amenable to mechanization, since R might be finite, whereas $u(R)$ is infinite.

5.3. **Conditional Bisimilarity Up-To Context with Representative Steps.** CBUCs allow us to represent certain infinite bisimulation relations in a finite way. For instance, we can use a finite CBUC in Example 4.18. However, automated checking if two agents are conditionally bisimilar — which can be done by incrementally extending a conditional bisimulation relation — is still hard, even using up-to context, since up-to context can only reduce the size of the relation itself. However, for just a single triple, there are infinitely many context steps to be checked.

For conditional bisimulations, we introduced an alternative definition using representative steps (Definition 4.13) and showed that it yields an equivalent notion of conditional bisimilarity (Theorem 4.17). We will show that the same approach can be used for CBUCs.

Definition 5.12 (CBUC with representative steps). A *CBUC with representative steps* is a conditional relation R such that the following holds: for each triple $(a, b, \mathcal{C}) \in R$ and each representative step $a \xrightarrow{f, \mathcal{A}}_R a'$, there are answering steps $b \xrightarrow{f, \mathcal{B}_i}_C b_i'$ and conditions $\mathcal{C}_i''$ such that for each answering step there exists $(a_i'', b_i'', \mathcal{C}_i'') \in R$ with $a' = a_i''; j_i$, $b_i' = b_i''; j_i$ for some arrow j_i per answering step, and additionally $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}_{i \downarrow j_i}'' \wedge \mathcal{B}_i)$; vice versa for steps $b \xrightarrow{f, \mathcal{B}}_R b'$.

Remark 5.13. A CBUC with representative steps can also be defined based on the closure under contextualization u: A conditional relation R satisfies Definition 5.12 if and only if $R \subseteq f_R(u(R))$ (i.e. R is a post-fixpoint of $f_R \circ u$). Analogously to Remark 5.8, this can be seen by expanding definitions. ⌟

To show that CBUCs defined using context and representative steps are essentially equivalent, we first relate the underlying functions by showing $f_R \subseteq f_C \circ u$.[6] Afterwards, we use that result to show that the two up-to techniques are equivalent.

---

[6]Intuitively, f_R guarantees only that representative steps are answered by context steps and that their successors are related again, but f_C requires this for non-representative steps as well, so generally $f_R \not\subseteq f_C$. Therefore we contextualize using u to let f_C access the non-representative successors as well.

Lemma 5.14. $f_R(R) \subseteq f_C(u(R))$.

Proof. Let $(a, b, \mathcal{C}) \in f_R(R)$ be given, which by its definition means that:

for all representative steps $a \xrightarrow{\hat{f}, \mathcal{A}}_R \hat{a}'$ (1) there are answering steps $b \xrightarrow{\hat{f}, \hat{\mathcal{B}}_i}_C \hat{b}'_i$
and conditions $\hat{\mathcal{C}}'_i$ (2) such that $(\hat{a}', \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ and (3) $\mathcal{A} \wedge \mathcal{C}_{\downarrow \hat{f}} \models \bigvee_{i \in I}(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i)$;
vice versa for representative steps of b

We show that this implies $(a, b, \mathcal{C}) \in f_C(u(R))$. Consider a context step $a \xrightarrow{f, \mathcal{A}}_C a'$. This step is not necessarily a representative step. According to Remark 3.11, this context step can be reduced to a representative step $a \xrightarrow{\hat{f}, \mathcal{R}_{\downarrow \hat{c}}}_R \hat{a}'$, where $\mathcal{R}$ is the condition of the rule used for the step, c is the reactive context of the context step, and there exists $\hat{g}$ such that $\hat{f};\hat{g} = f$, $\hat{c};\hat{g} = c$, $\hat{a}';\hat{g} = a'$, with $\hat{f}, \hat{c}, \hat{a}'$ referring to the representative step.

Since $(a, b, \mathcal{C}) \in f_R(R)$, we know that answering steps for our representative step exist. From that we can conclude the following:

(1) $b \xrightarrow{\hat{f}, \hat{\mathcal{B}}_i}_C \hat{b}'_i$ implies, according to Lemma 4.15, that a step $b \xrightarrow{\hat{f};\hat{g}, \ \hat{\mathcal{B}}_{i \downarrow \hat{g}}}_C \hat{b}'_i;\hat{g}$, equivalently, $b \xrightarrow{f, \ \hat{\mathcal{B}}_{i \downarrow \hat{g}}}_C \hat{b}'_i;\hat{g}$, is possible. We select $\mathcal{B}_i := \hat{\mathcal{B}}_{i \downarrow \hat{g}}$, $b'_i := \hat{b}'_i;\hat{g}$, $\mathcal{C}'_i := \hat{\mathcal{C}}'_{i \downarrow \hat{g}}$ for the answering steps of b using borrowed context f.

(2) $(\hat{a}', \hat{b}'_i, \hat{\mathcal{C}}'_i) \in R$ implies $(\hat{a}';\hat{g}, \ \hat{b}'_i;\hat{g}, \ \hat{\mathcal{C}}'_{i \downarrow \hat{g}}) = (a', b'_i, \mathcal{C}'_i) \in u(R)$.

(3) Using the rules of Theorem 3.6 we get:

$$\mathcal{R}_{\downarrow \hat{c}} \wedge \mathcal{C}_{\downarrow \hat{f}} \models \bigvee \left(\hat{\mathcal{C}}'_i \wedge \hat{\mathcal{B}}_i \right)$$

$$\Rightarrow (\mathcal{R}_{\downarrow \hat{c}})_{\downarrow \hat{g}} \wedge \left(\mathcal{C}_{\downarrow \hat{f}} \right)_{\downarrow \hat{g}} \models \bigvee \left(\hat{\mathcal{C}}'_{i \downarrow \hat{g}} \wedge \hat{\mathcal{B}}_{i \downarrow \hat{g}} \right)$$

$$\Leftrightarrow \mathcal{R}_{\downarrow \hat{c};\hat{g}} \wedge \mathcal{C}_{\downarrow \hat{f};\hat{g}} \models \bigvee \left(\hat{\mathcal{C}}'_{i \downarrow \hat{g}} \wedge \hat{\mathcal{B}}_{i \downarrow \hat{g}} \right)$$

$$\Leftrightarrow \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow f} \models \bigvee \left(\mathcal{C}'_i \wedge \mathcal{B}_i \right)$$

For context steps, $\mathcal{A} \models \mathcal{R}_{\downarrow c}$ holds, so we have $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \mathcal{R}_{\downarrow c} \wedge \mathcal{C}_{\downarrow f} \models \bigvee (\mathcal{C}'_i \wedge \mathcal{B}_i)$.

Analogously, we can construct answering steps for $b \xrightarrow{f, \mathcal{B}}_C b'$. To summarize, for the given $(a, b, \mathcal{C}) \in f_R(R)$ we have concluded that

for all context steps $a \xrightarrow{f, \mathcal{A}}_C a'$ (1) there exist answering steps $b \xrightarrow{f, \mathcal{B}_i}_C b'_i$ and conditions $\mathcal{C}'_i$ (2) such that $(a', b'_i, \mathcal{C}'_i) \in u(R)$ and (3) $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I}(\mathcal{C}'_i \wedge \mathcal{B}_i)$;
vice versa for context steps of b

which is exactly the definition of $(a, b, \mathcal{C}) \in f_C(u(R))$. $\square$

Corollary 5.15. *It holds that $f_C \circ u = f_R \circ u$.*

Proof.

• $(f_C \circ u \subseteq f_R \circ u)$:

$$f_C \subseteq f_R \implies f_C \circ u \subseteq f_R \circ u$$

- $(f_R \circ u \subseteq f_C \circ u)$:

$$\text{(Lemma 5.14)} \qquad f_R \quad \subseteq f_C \circ u$$
$$\Longrightarrow f_R \circ u \subseteq f_C \circ u \circ u$$
$$\text{(u idempotent)} \iff f_R \circ u \subseteq f_C \circ u \qquad \square$$

Theorem 5.16. *A conditional relation is a CBUC (Definition 5.7) if and only if it is a CBUC with representative steps (Definition 5.12).*

Proof. R satisfying Definitions 5.7 and 5.12 means that R is a post-fixpoint of $f_C \circ u$ or $f_R \circ u$, respectively. By Corollary 5.15, $f_C \circ u = f_R \circ u$, therefore post-fixpoints of one are also post-fixpoints of the other. $\square$

Observe that this is a stronger result than for normal conditional bisimilarity: for that, we know that the bisimilarities are the same ($\nu f_C = \nu f_R$) but the bisimulation functions are not ($f_C \neq f_R$). On the other hand, using up-to techniques, the difference between the bisimulation functions themselves disappears (i.e. $f_C \circ u = f_R \circ u$ instead of just $\nu(f_C \circ u) = \nu(f_R \circ u)$). This results from Lemma 5.14 and can be explained intuitively as follows: the function f_R requires that every representative step can be answered by a context step and the resulting pair is in R, while f_C requires such an answer for all context steps. This means that $f_C(R) \subseteq f_R(R)$ as explained earlier. The pairs potentially missing in $f_C(R)$ resulted from larger-than-necessary contexts and hence f_R did not require them. Using u, however, the relation is contextualized beforehand, using all (even non-representative) contexts, and hence makes these triples "available" to f_C.

Note that even though the difference between the two variants disappears, using representative steps with CBUCs still is advantageous because it typically results in a finitely branching transition system. This can be seen in the continuation of our example:

Example 5.17. Consider again Examples 4.7 and 4.18. We have previously seen that it is possible to repeatedly borrow a message on the left-hand node and transfer it to the right-hand node, which leads to more and more received messages accumulating at the right-hand node. We now show that the two types of channels are conditionally bisimilar by showing that $R = \left\{ (\emptyset \to \bullet\!\xrightarrow{rel}\!\bullet \leftarrow \bullet\ \bullet,\ \emptyset \to \bullet\!\xrightarrow{unr}\!\bullet \leftarrow \bullet\ \bullet,\ \mathcal{A}_{\overline{n}}) \right\}$ is a CBUC, i.e. it satisfies Definition 5.12. We consider the same steps as in Example 4.18:

- The graph $\bullet\!\xrightarrow{rel}\!\bullet$ can do a step using rule P_R by borrowing a message on the left node, with environment condition $\mathcal{A} = \text{true}$, and reduces to $a' = \emptyset \to \bullet\!\xrightarrow{rel}\!\bullet\!\curvearrowright\! m \leftarrow \bullet\ \bullet$. Then, $\bullet\!\xrightarrow{unr}\!\bullet$ can answer this step using P_U under $\mathcal{B}_i = \mathcal{A}_{\overline{n}}$ (no noise) and reacts to $b_i' = \emptyset \to \bullet\!\xrightarrow{unr}\!\bullet\!\curvearrowright\! m \leftarrow \bullet\ \bullet$.

  Now set $j_i = \bullet\ \bullet \to \bullet\ \bullet\!\curvearrowright\! m \leftarrow \bullet\ \bullet$, i.e. we consider the m-loop on the right node as irrelevant context. Then, using $a_i'' = \emptyset \to \bullet\!\xrightarrow{rel}\!\bullet \leftarrow \bullet\ \bullet$, $b_i'' = \emptyset \to \bullet\!\xrightarrow{unr}\!\bullet \leftarrow \bullet\ \bullet$, $\mathcal{C}_i'' = \mathcal{A}_{\overline{n}}$ we have $a' = a_i'';j_i$, $b_i' = b_i'';j_i$, and we find that the triple without the irrelevant context j_i, that is $(a_i'', b_i'', \mathcal{C}_i'')$ (which happens to be the same as our initial triple), is contained in R. As before, the implication $\mathcal{A} \wedge \mathcal{C}_{\downarrow f} \models \bigvee_{i \in I} (\mathcal{C}_i'' \wedge \mathcal{B}_i)$ holds.
- Symmetrically, $\bullet\!\xrightarrow{unr}\!\bullet$ borrows a message on the left node and reacts to $\bullet\!\xrightarrow{unr}\!\bullet\!\curvearrowright\! m$ under $\mathcal{A} = \mathcal{A}_{\overline{n}}$. Analogously to the previous case and to Example 4.18, $\bullet\!\xrightarrow{rel}\!\bullet$ answers this step, using $\mathcal{C}_i'' = \mathcal{A}_{\overline{n}}$ and $j_i = \bullet\ \bullet \to \bullet\ \bullet\!\curvearrowright\! m \leftarrow \bullet\ \bullet$.
- Again, the remaining representative steps can be proven in an analogous way.

Figure 10 shows a comparison of the necessary steps with and without using up-to techniques. Note that instead of working with an infinite bisimulation, we now have a singleton. ⌟

Finally, we show compatibility of f_R and summarize the theorems of this section.

Corollary 5.18 (u is f_R-compatible)**.** *Let R be a conditional relation. Then it holds that* $u(f_R(R)) \subseteq f_R(u(R))$, *i.e., u is f_R-compatible.*

Proof. By transitivity from previous results:

$$
\begin{array}{rl}
\text{(Lemma 5.14)} & f_R \subseteq \quad f_C \circ u \\
\implies & u \circ f_R \subseteq u \circ f_C \circ u \\
(u \circ f_C \subseteq f_C \text{ by Theorem 5.9}) & \subseteq \quad f_C \quad \circ u \\
\text{(by Corollary 5.15)} & = f_R \circ u \qquad \square
\end{array}
$$

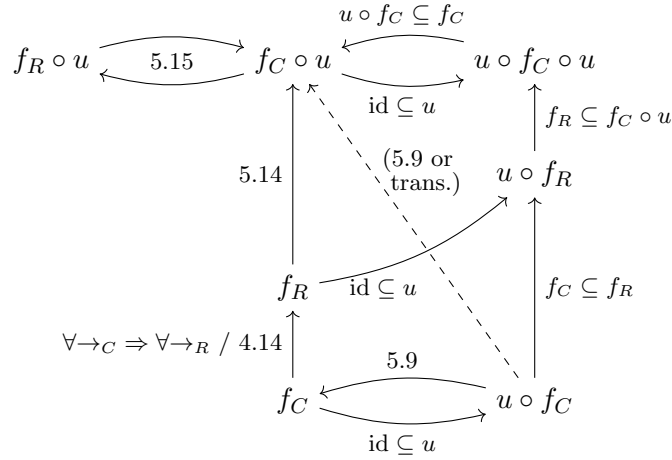Figure 12 summarizes the known inclusions and equalities that were proven throughout this section.

Figure 12: Known relations ($X \to Y$ indicates $X \subseteq Y$)

Note that using the results of this section, it is possible to provide alternative proofs of various theorems of Section 4, in particular:

- Remark 4.8 ($\backsim_C$ is closed under condition strengthening): We can define the function $str(R) := \{(a, b, \mathcal{C}) \mid (a, b, \mathcal{C}') \in R,\ \mathcal{C} \models \mathcal{C}'\}$ and show that str is f_C-compatible by showing $(a, b, \mathcal{C}) \in str(f_C(R)) \implies (a, b, \mathcal{C}) \in f_C(str(R))$. This can be done by expanding the definitions of str, f_C in $(a, b, \mathcal{C}) \in str(f_C(R))$ and rewriting it to match the definition of $(a, b, \mathcal{C}) \in f_C(str(R))$. Then, we can apply Proposition 5.5.
- Lemma 4.11 part 4 ($\backsim_C$ is closed under contextualization): Restated using u, f_C (note that $\backsim_C = \nu f_C$), we have to show that $u(\nu f_C) \subseteq \nu f_C$. This follows immediately from f_C-compatibility and Proposition 5.5.
- Lemma 4.16 ($\backsim_R$ is closed under contextualization): As for $\backsim_C$, but use f_R instead of f_C.
- Theorem 4.17 ($\backsim_C = \backsim_R$): We show that $\nu f_C = \nu f_R$. For this we use the fact that $f_R \circ u = f_C \circ u$ (Corollary 5.15) and therefore also $\nu(f_R \circ u) = \nu(f_C \circ u)$. u is f_C-compatible (Theorem 5.9), so by Lemma 5.3, $\nu(f_C \circ u) = \nu f_C$. Similarly, u is f_R-compatible (Corollary 5.18) and therefore $\nu(f_R \circ u) = \nu f_R$. Combining these results, we have $\backsim_C = \nu f_C = \nu(f_C \circ u) = \nu(f_R \circ u) = \nu f_R = \backsim_R$.

6. Comparison and An Alternative Characterization

6.1. **An Equivalent Characterization Based on Environment Steps.** We will now give an alternative characterization of conditional bisimilarity, in order to justify Definitions 4.3 and 4.13. This alternative definition is more elegant since it characterizes $\backsimeq_C$ as the largest conditional congruence that is a conditional environment bisimulation. On the other hand, this definition is (like conditional bisimilarity, as described in Example 4.18) not directly suitable for mechanization, since the underlying transition system is not finitely branching.

In [HK12], environment steps, which capture the idea that a reaction is possible under some *passive* context d, have been defined to obtain a more natural characterization of saturated bisimilarity. Unlike the borrowed context f, the passive context d does not participate in the reaction itself, but we refer to it to ensure that the application condition of the rule holds.

Definition 6.1 (Environment step [HK12])**.** Let $\mathcal{S}$ be a conditional reactive system and let $a: 0 \to K$, $a': 0 \to K$, $d: K \to J$ be arrows. We write $a \stackrel{d}{\rightsquigarrow} a'$ whenever there exists a rule $(\ell, r, \mathcal{R}) \in \mathcal{S}$ and an arrow c such that $a = \ell;c$, $a' = r;c$ and $c;d \models \mathcal{R}$.

Environment steps and context steps are related: they can be transformed into each other. Furthermore saturated bisimilarity is the coarsest bisimulation relation over environment steps that is also a congruence [HK12]. We now give a characterization of conditional bisimilarity based on environment steps:

Definition 6.2 (Conditional environment congruence)**.** A conditional relation R is a *conditional environment bisimulation* if whenever $(a, b, \mathcal{C}) \in R$ and $a \stackrel{d}{\rightsquigarrow} a'$ for some $d \models \mathcal{C}$, then $b \stackrel{d}{\rightsquigarrow} b'$ and $(a', b', \mathcal{C}') \in R$ for some condition $\mathcal{C}'$ such that $d \models \mathcal{C}'$; vice versa for $b \stackrel{d}{\rightsquigarrow} b'$. We denote by $\backsimeq_E$ the largest conditional environment bisimulation that is also a conditional congruence and call it *conditional environment congruence*.

For the proof of Theorem 6.4, we need the following lemma:

Lemma 6.3 [HK12, Lemma 22]**.** *Given a context step $a \xrightarrow{f, \mathcal{A}}_C a'$ and a passive context d such that $d \models \mathcal{A}$, we have an environment step $a;f \stackrel{d}{\rightsquigarrow} a'$. Conversely, given an environment step $a;f \stackrel{d}{\rightsquigarrow} a'$, there exists a condition $\mathcal{A}$ such that $d \models \mathcal{A}$ and we have a context step $a \xrightarrow{f, \mathcal{A}}_C a'$.*

Theorem 6.4. *Conditional bisimilarity and conditional environment congruence coincide, that is, $\backsimeq_C = \backsimeq_E$.*

Proof. In both parts we show only how steps of a can be answered by b, the other direction can be shown analogously.

- **($\backsimeq_C \subseteq \backsimeq_E$):**
  We show that $\backsimeq_C$ is a conditional environment bisimulation. Together with the fact that $\backsimeq_C$ is a conditional congruence (Lemma 4.11), we obtain the result that $\backsimeq_C$ is contained in conditional environment congruence.

  Let $(a, b, \mathcal{C}) \in \backsimeq_C$ and $a \stackrel{d}{\rightsquigarrow} a'$ for some $d \models \mathcal{C}$. We rewrite $a \stackrel{d}{\rightsquigarrow} a'$ as $a;\mathrm{id} \stackrel{d}{\rightsquigarrow} a'$ and, using Lemma 6.3, obtain a context step $a \xrightarrow{\mathrm{id}, \mathcal{A}}_C a'$ for some condition $\mathcal{A}$ such that $d \models \mathcal{A}$.

Since $(a, b, \mathcal{C}) \in \mathord{\approx}_C$, there exist answering steps $b \xrightarrow{\mathrm{id}, \mathcal{B}_i}_C b_i'$ such that $(a', b_i', \mathcal{C}_i') \in \mathord{\approx}_C$ for some $\mathcal{C}_i'$ and $\mathcal{A} \wedge \mathcal{C}_{\downarrow\mathrm{id}} \models \bigvee_{i \in I} (\mathcal{B}_i \wedge \mathcal{C}_i')$. Since d satisfies both $\mathcal{A}$ and $\mathcal{C} \equiv \mathcal{C}_{\downarrow\mathrm{id}}$, there exists an index i such that $d \models \mathcal{B}_i \wedge \mathcal{C}_i'$ and $(a', b_i', \mathcal{C}_i') \in \mathord{\approx}_C$. This directly gives us the answering step required by conditional environment bisimilarity: Since $d \models \mathcal{B}_i$, using Lemma 6.3 we rewrite the corresponding context step $b \xrightarrow{\mathrm{id}, \mathcal{B}_i}_C b_i'$ to an environment step $b; \mathrm{id} = b \xrightarrow{d}_{\rightsquigarrow} b_i'$. Setting $b' := b_i'$, $\mathcal{C}' := \mathcal{C}_i'$, we obtain the required triple $(a', b', \mathcal{C}') \in \mathord{\approx}_C$.

- **($\mathord{\approx}_E \subseteq \mathord{\approx}_C$):**

  We show that $\mathord{\approx}_E$ is a conditional bisimulation. Let $(a, b, \mathcal{C}) \in \mathord{\approx}_E$ and $a \xrightarrow{f, \mathcal{A}}_C a'$.

  Let d be some context. If $d \not\models \mathcal{A} \wedge \mathcal{C}_{\downarrow f}$, we can easily satisfy Definition 4.3 by letting b answer with an empty set of answering steps. We therefore assume that $d \models \mathcal{A} \wedge \mathcal{C}_{\downarrow f}$.

  Since $d \models \mathcal{A}$, using Lemma 6.3 we rewrite $a \xrightarrow{f, \mathcal{A}}_C a'$ to $a; f \xrightarrow{d}_{\rightsquigarrow} a'$. As $\mathord{\approx}_E$ is, by definition, a conditional congruence, $(a, b, \mathcal{C}) \in \mathord{\approx}_E$ implies $(a; f, b; f, \mathcal{C}_{\downarrow f}) \in \mathord{\approx}_E$.

  Since $(a; f, b; f, \mathcal{C}_{\downarrow f}) \in \mathord{\approx}_E$, $a; f \xrightarrow{d}_{\rightsquigarrow} a'$ and $d \models \mathcal{C}_{\downarrow f}$, there exists an answering step $b; f \xrightarrow{d}_{\rightsquigarrow} b_d'$ and for some condition $\mathcal{C}_d'$ such that $d \models \mathcal{C}_d'$ we have $(a', b_d', \mathcal{C}_d') \in \mathord{\approx}_E$.

  By Lemma 6.3, $b; f \xrightarrow{d}_{\rightsquigarrow} b_d'$ implies $b \xrightarrow{f, \mathcal{B}_d}_C b_d'$ for some $\mathcal{B}_d$ such that $d \models \mathcal{B}_d$.

  Thus, whenever $d \models \mathcal{A} \wedge \mathcal{C}_{\downarrow f}$, there exists an answering step $b \xrightarrow{f, \mathcal{B}_d}_C b_d'$ such that $d \models \mathcal{B}_d \wedge \mathcal{C}_d'$ and $(a', b_d', \mathcal{C}_d') \in \mathord{\approx}_E$. which concludes the proof of $\mathord{\approx}_E$ being a conditional bisimulation. $\qquad\square$

Note that in the second part of the proof, we use the fact that b can reply with an infinite set of answering steps, since the infinitely many answering steps $b; f \xrightarrow{d}_{\rightsquigarrow} b_d'$ might give rise to infinitely many different $\mathcal{B}_d$ and accompanying $\mathcal{C}_d'$.

It is an open question if the proof is also possible with finitely many answering steps. In [HK12, Theorem 23], a similar comparison of saturated bisimilarity and environment congruence was done, although for binary relations which did not include conditions in the relation itself. In that proof, the finiteness assumption (Fin) was used to obtain a finite set of answering steps, which is however not possible in the presence of conditions.

6.2. **Comparison to Other Equivalences.** We conclude this section by considering the binary relation $\mathord{\approx}_T := \{(a, b) \mid (a, b, \mathrm{true}) \in \mathord{\approx}_C\}$, derived from conditional bisimilarity, which is ternary. Intuitively it contains pairs (a, b), where a, b are system states that behave equivalently in every possible context. We investigate how $\mathord{\approx}_T$ compares to other behavioural equivalences that also check for identical behaviour in all contexts. First, we consider saturated bisimilarity ($\sim_C$), which has been characterized in [HK12] as the coarsest relation which is a congruence as well as a bisimilarity:

Theorem 6.5. *Saturated bisimilarity implies* true-*conditional bisimilarity ($\sim_C \subseteq \mathord{\approx}_T$). However,* true-*conditional bisimilarity does* not *imply saturated bisimilarity ($\mathord{\approx}_T \not\subseteq \sim_C$).*

Proof.

- **($\sim_C \subseteq \mathord{\approx}_T$):**

  Let R be a saturated bisimulation relation. Then we define $R' = R \times \{\mathrm{true}\}$ and show that R' is a conditional bisimulation relation. For that purpose, let some $(a, b, \mathrm{true}) \in R'$ be given, i.e. $(a, b) \in R$. Now assume a transition $a \xrightarrow{f, \mathcal{A}}_C a'$, then by the fact that $(a, b) \in R$

we know that there exist some answering steps $b \xrightarrow{f,\mathcal{B}_i}_C b_i'$, $i \in I$, such that $(a', b_i') \in R$ for all $i \in I$. By definition of R' it follows that, for all $i \in I$, $(a', b_i', \text{true}) \in R'$. So it remains to show that $\mathcal{A} \wedge \text{true}_{\downarrow f} \models \bigvee_{i \in I} (\text{true} \wedge \mathcal{B}_i)$. We can simplify this to $\mathcal{A} \models \bigvee_{i \in I} \mathcal{B}_i$, which holds because R is a saturated bisimulation.

Steps $b \xrightarrow{f,\mathcal{B}}_C b'$ can be answered analogously.

- $(\overset{\circ}{\sim}_T \nsubseteq \sim_C)$:
  Consider the following reactive system $\{R_A, R_{B1}, T_{B1}, R_{B2}, T_{B2}\}$:

$$R_A = \left(\emptyset \to \boxed{a\,\text{⊙}} \leftarrow \emptyset,\ \emptyset \to \boxed{e_a\,\text{⊙}} \leftarrow \emptyset, \text{true}_\emptyset \right)$$

$$R_{B1} = \left(\emptyset \to \boxed{b\,\text{⊙}} \leftarrow \emptyset,\ \emptyset \to \boxed{b_1\,\text{⊙}} \leftarrow \emptyset, \mathcal{A}_C \right) \quad T_{B1} = \left(\emptyset \to \boxed{b_1\,\text{⊙}} \leftarrow \emptyset,\ \emptyset \to \boxed{e_1\,\text{⊙}} \leftarrow \emptyset, \neg\mathcal{A}_C \right)$$

$$R_{B2} = \left(\emptyset \to \boxed{b\,\text{⊙}} \leftarrow \emptyset,\ \emptyset \to \boxed{b_2\,\text{⊙}} \leftarrow \emptyset, \neg\mathcal{A}_C \right) \quad T_{B2} = \left(\emptyset \to \boxed{b_2\,\text{⊙}} \leftarrow \emptyset,\ \emptyset \to \boxed{e_2\,\text{⊙}} \leftarrow \emptyset, \mathcal{A}_C \right)$$

where $\mathcal{A}_C = \left(\emptyset, \forall, \left\{(\emptyset \to \boxed{\text{⊙}\,c} \leftarrow \boxed{\text{⊙}\,c},\ \text{true}_C)\right\}\right)$. An a-loop can be replaced with a graph which allows no further steps. A b-loop can, in case the environment contains a c-loop $(\mathcal{A}_C)$, transition to a b_1-loop, from which another transition is possible if *no* c-loop is present (as this contradicts the condition of the first step, this transition can never actually be executed). Similarly, if *no* c-loop is present, a transition to a b_2-loop is possible and subsequently another transition is possible *if* there is a c-loop.

It is easy to see that no matter which context $\boxed{a\,\text{⊙}}$ and $\boxed{b\,\text{⊙}}$ are placed into, both admit at most one transition. Therefore, $(\boxed{a\,\text{⊙}}, \boxed{b\,\text{⊙}}) \in \overset{\circ}{\sim}_T$, as witnessed by the conditional bisimulation relation $R = \{(\boxed{a\,\text{⊙}}, \boxed{b\,\text{⊙}}, \text{true}),\ (\boxed{e_a\,\text{⊙}}, \boxed{b_1\,\text{⊙}}, \mathcal{A}_C),\ (\boxed{e_a\,\text{⊙}}, \boxed{b_2\,\text{⊙}}, \neg\mathcal{A}_C)\}$. For saturated bisimilarity however, the initial step of $\boxed{a\,\text{⊙}}$ to $\boxed{e_a\,\text{⊙}}$ can be answered by $\boxed{b\,\text{⊙}}$ with two steps as for conditional bisimilarity, and it would be required that $(\boxed{e_a\,\text{⊙}}, \boxed{b_1\,\text{⊙}}), (\boxed{e_a\,\text{⊙}}, \boxed{b_2\,\text{⊙}}) \in \sim_C$. But then, $\boxed{b_1\,\text{⊙}}$ can do a step (under $\neg\mathcal{A}_C$ as indicated by rule T_{B1}) which $\boxed{e_a\,\text{⊙}}$ cannot answer. $\qquad\square$

For saturated bisimilarity, if a step of a is answered by b with multiple steps, all b_i' reached in this way must be saturated bisimilar to a' (that is, show the same behaviour even if the environment is later changed to one which did not allow the given b_i' to be reached). In fact, it was an explicit goal in the design of saturated bisimilarity to account for external modification of the environment.

On the other hand, for conditional bisimilarity, each b_i' is only required to be conditionally bisimilar to a' under the condition which allowed this particular answering step — that is, after a step, the environment is fixed (or, depending on the system, can only assume a subset of all possible environments, cf. Definition 6.2 and Theorem 6.4).

Next, we compare $\overset{\circ}{\sim}_T$ to id-congruence, the coarsest congruence contained in bisimilarity over the reaction relation $\rightsquigarrow$. It simply relates two agents whenever they are bisimilar in all contexts, i.e. $\sim_{\text{id}} := \{(a, b) \mid \text{for all contexts } d, \ a;d, \ b;d \text{ are bisimilar wrt. } \rightsquigarrow\}$.

Theorem 6.6. *It holds that* true-*conditional bisimilarity implies* id-*congruence* $(\overset{\circ}{\sim}_T \subseteq \sim_{\text{id}})$. *However,* id-*congruence does* not *imply* true-*conditional bisimilarity* $(\sim_{\text{id}} \nsubseteq \overset{\circ}{\sim}_T)$.

Proof.

- $(\overset{\circ}{\sim}_T \subseteq \sim_{\text{id}})$:
  Given $(a, b) \in \overset{\circ}{\sim}_T$, equivalently $(a, b, \text{true}) \in \overset{\circ}{\sim}_C$, by Theorem 4.9 we know that for all contexts d such that $d \models \text{true}$ (i.e. all contexts), $(a;d, b;d)$ is contained in a bisimulation relation over $\rightsquigarrow$. This, however, is the exact requirement for $(a, b) \in \sim_{\text{id}}$.

- $(\sim_{\mathrm{id}} \not\subseteq \overset{\circ}{\sim}_T)$:
  Consider the example presented in Remark 4.10: The graphs A and B are bisimilar under all contexts (i.e., $(a, b) \in \sim_{\mathrm{id}}$), however, they are not conditionally bisimilar under true (i.e., $(a, b) \notin \overset{\circ}{\sim}_T$). $\qquad\square$

Intuitively, true-conditional bisimilarity allows to observe whether some item is consumed and recreated (by including it in both sides of a rule) or whether it is simply required (using an existential rule condition, cf. Theorem 6.6). On the other hand, id-congruence does not recognize this and simply checks whether reactions are possible in the same set of contexts.

Hence we have $\sim_C \subsetneq \overset{\circ}{\sim}_T \subsetneq \sim_{\mathrm{id}}$, which implies that checking for identical behaviour in all contexts using conditional bisimilarity gives rise to a new kind of behavioural equivalence, which does not allow arbitrary changes to the environment (as $\sim_C$ does), yet allows distinguishing borrowed and passive context (which $\sim_{\mathrm{id}}$ does not).

7. Conclusion, Related and Future Work

The conditions that we studied in this paper are also known under the name of nested conditions or graph conditions and were introduced in [Ren04], where their equivalence to first-order logic was shown. They were studied more extensively in [HP09, Pen09] and generalized to reactive systems in [BCHK11]. In fact, the related notion of Q-trees was introduced earlier in [FS90].

As stated earlier, there are some scattered approaches to notions of behavioural equivalence that can be compared to conditional bisimilarity. The concept of behaviour depending on a context is also present in Larsen's PhD thesis [Lar86]. There, the idea is to embed an LTS into an environment, which is modelled as an action transducer, an LTS that consumes transitions of the system under investigation — similar to CCS synchronization. Larsen then defines environment-parameterized bisimulation by considering only those transitions that are consumed in a certain environment. In [HL95], Hennessy and Lin describe symbolic bisimulations in the setting of value-passing processes, where Boolean expressions restrict the interpretations for which one shows bisimilarity. Instead in [BBB02], Baldan, Bracciali and Bruni propose bisimilarity on open systems, specified by terms with a hole or place-holder. Instead of imposing conditions on the environment, they restrict the components that are filling the holes.

In [Fit02], Fitting studies a matrix view of unlabelled transition systems, annotated by Boolean conditions. In [BKKS17] we have shown that such systems can alternatively be viewed as conditional transition systems, where activation of transitions depends on conditions of the environment and one can state the bisimilarity of two states provided that the environment meets certain requirements. This view is closely tied to featured transition systems, which have been studied extensively in the software engineering literature. The idea here is to specify system behaviour dependent on the features that are present in the product (see for instance [CCP$^+$12] for simulations on featured transition systems).

Our contribution in this paper is to consider conditional bisimilarity based on contextualization in a rule-based setting. That is, system behaviour is specified by generic rewriting rules, system states can be composed with a context specifying the environment and we impose restrictions on those contexts. By viewing both system states and contexts as arrows of a category, we can work in the framework of reactive systems à la Leifer and Milner and define a general theory of conditional bisimilarity. While in [HK12] conditions were only used

to restrict applicability of the rules and bisimilarity was checked for all contexts, we here additionally use conditions to establish behavioural equivalence only in specific contexts.

As future work we want to take a closer look at the logic that we used to specify conditions. Conditional bisimilarity is defined in a way that is largely independent of the kind of logic, provided that the logic supports Boolean operators and shift. It is unclear and worth exploring whether the logic considered by us is expressive enough to characterize all contexts that ensure bisimilarity of two given arrows. This also affects the question whether or not infinitely many answering steps are required in Theorem 6.4.

Furthermore, it is an open question whether there is an alternative characterization of the id-congruence of Theorem 6.6 that is amenable to mechanization.

We have already implemented label derivation and bisimulation checking in the borrowed context approach, see for instance [Nol12], and successfully applied it to a system with message-passing rules similar to the ones given in Example 4.18, however without conditions in either the rules or the bisimulation relation. Our aim is to also obtain an efficient implementation for the scenario described in this paper. Note that our conditions subsume first-order logic [BCHK11] and hence in order to come to terms with the undecidability of implication we have to resort to simpler conditions or use approximative methods.

Another natural question is whether our results can be stated in a coalgebraic setting, since coalgebra provides a generic framework for behavioural equivalences. We have already studied a much simplified coalgebraic version of conditional systems (without considering contextualization) in [ABH+12], using coalgebras living in Kleisli categories. Reactive systems can also be viewed as coalgebras (see [Bon08]). However, a combination of these features has not yet been considered as far as we know.

Another direction for future research are further optimizations in terms of the up-to context technique. Note, that even bisimulations up-to context can still be infinite in size, which is somehow unavoidable due to undecidability issues, so further optimizations should be investigated. Furthermore we plan to integrate this method with other kinds of up-to techniques such as up-to bisimilarity, which should be easy due to the integration into the lattice-theoretical framework.

References

[ABH+12] Jiří Adámek, Filippo Bonchi, Mathias Hülsbusch, Barbara König, Stefan Milius, and Alexandra Silva. A coalgebraic perspective on minimization and determinization. In *Proc. of FOSSACS '12*, pages 58–73. Springer, 2012. LNCS/ARCoSS 7213.

[BBB02]   Paolo Baldan, Andrea Bracciali, and Roberto Bruni. Bisimulation by unification. In *Proc. of AMAST '02*, pages 254–270. Springer, 2002. LNCS 2422.

[BCHK11] H.J. Sander Bruggink, Raphaël Cauderlier, Mathias Hülsbusch, and Barbara König. Conditional reactive systems. In *Proc. of FSTTCS '11*, volume 13 of *LIPIcs*. Schloss Dagstuhl – Leibniz Center for Informatics, 2011.

[BKKS17] Harsh Beohar, Barbara König, Sebastian Küpper, and Alexandra Silva. Conditional transition systems with upgrades. In *Proc. of TASE '17 (Theoretical Aspects of Software Engineering)*. IEEE Xplore, 2017.

[BKM06]   Filippo Bonchi, Barbara König, and Ugo Montanari. Saturated semantics for reactive systems. In *Proc. of LICS '06*, pages 69–80. IEEE, 2006.

[Bon08] Filippo Bonchi. *Abstract Semantics by Observable Contexts*. PhD thesis, Università degli Studi di Pisa, Dipartimento di Informatica, May 2008.

[BW99] Michael Barr and Charles Wells. *Category Theory for Computing Science*. Les Publications CMR, 1999.

[CCP⁺12] Maxime Cordy, Andreas Classen, Gilles Perrouin, Pierre-Yves Schobbens, Patrick Heymans, and Axel Legay. Simulation-based abstractions for software product-line model checking. In *Proc. of ICSE '12*, pages 672–682. IEEE, 2012.

[CMR⁺97] Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation—part I: Basic concepts and double pushout approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*, chapter 3. World Scientific, 1997.

[EK04] Hartmut Ehrig and Barbara König. Deriving bisimulation congruences in the DPO approach to graph rewriting. In *Proc. of FOSSACS '04*, pages 151–166. Springer, 2004. LNCS 2987.

[EPS73] Hartmut Ehrig, Michael Pfender, and Hans Jürgen Schneider. Graph-grammars: An algebraic approach. In *14th Annual Symposium on Switching and Automata Theory (SWAT 1973)*, pages 167–180, Oct 1973.

[Fit02] Melvin Fitting. Bisimulations and boolean vectors. In *Advances in Modal Logic*, volume 4, pages 1–29. World Scientific Publishing, 2002.

[FS90] Peter J. Freyd and Andre Scedrov. *Categories, Allegories*. North-Holland, 1990.

[HHT96] Annegret Habel, Reiko Heckel, and Gabriele Taentzer. Graph grammars with negative application conditions. *Fundamenta Informaticae*, 26(3,4):287–313, December 1996.

[HK12] Mathias Hülsbusch and Barbara König. Deriving bisimulation congruences for conditional reactive systems. In *Proc. of FOSSACS '12*, pages 361–375. Springer, 2012. LNCS/ARCoSS 7213.

[HL95] Matthew Hennessy and Huimin Lin. Symbolic bisimulations. *Theoretical Computer Science*, 138(2):353–389, 1995.

[HMP01] Annegret Habel, Jürgen Müller, and Detlef Plump. Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science*, 11(5):637–688, October 2001.

[HP09] Annegret Habel and Karl-Heinz Pennemann. Correctness of high-level transformation systems relative to nested conditions. *Mathematical Structures in Computer Science*, 19(2):245–296, 2009.

[JM03] Ole Høgh Jensen and Robin Milner. Bigraphs and transitions. In *Proc. of POPL 2003*, pages 38–49. ACM, 2003.

[KSS05] Bartek Klin, Vladimiro Sassone, and Paweł Sobociński. Labels from reductions: towards a general theory. In *Proc. of CALCO '05*, pages 30–50. Springer, 2005. LNCS 3629.

[Lar86] Kim Guldstrand Larsen. *Context-Dependent Bisimulation between Processes*. PhD thesis, University of Edinburgh, 1986.

[LM00] James J. Leifer and Robin Milner. Deriving bisimulation congruences for reactive systems. In *CONCUR 2000 — Concurrency Theory: 11th International Conference University Park, PA, USA, August 22–25, 2000 Proceedings*, pages 243–258. Springer Berlin Heidelberg, 2000.

[LS05] Stephen Lack and Paweł Sobociński. Adhesive and quasiadhesive categories. *RAIRO – Theoretical Informatics and Applications*, 39(3):511–545, 2005.

[Nol12] Dennis Nolte. Automatischer Nachweis von Bisimulationsäquivalenzen bei Graphtransformationssystemen. Master's thesis, Universität Duisburg-Essen, November 2012.

[Pen09] Karl-Heinz Pennemann. *Development of Correct Graph Transformation Systems*. PhD thesis, Universität Oldenburg, May 2009.

[Pou07] Damien Pous. Complete lattices and up-to techniques. In *Proc. of APLAS '07*, pages 351–366. Springer, 2007. LNCS 4807.

[PS11a] Damien Pous and Davide Sangiorgi. Enhancements of the bisimulation proof method. In *Advanced Topics in Bisimulation and Coinduction*, Cambridge Tracts in Theoretical Computer Science, pages 233–289. Cambridge University Press, 2011.

[PS11b] Damien Pous and Davide Sangiorgi. Enhancements of the coinductive proof method. In Davide Sangiorgi and Jan Rutten, editors, *Advanced Topics in Bisimulation and Coinduction*. Cambridge University Press, 2011.

[PS19] Damien Pous and Davide Sangiorgi. Bisimulation and coinduction enhancements: A historical perspective. *Formal Aspects of Computing*, 31(6):733–749, 2019.

[Ren04]   Arend Rensink. Representing first-order logic using graphs. In *Proc. of ICGT '04*, pages 319–335. Springer, 2004. LNCS 3256.

[San98]   Davide Sangiorgi. On the bisimulation proof method. *Mathematical Structures in Computer Science*, 8(5):447–479, 1998.

[Sob04]   Paweł Sobociński. *Deriving process congruences from reaction rules*. PhD thesis, University of Aarhus, 2004.

[SS05]   Vladimiro Sassone and Paweł Sobociński. Reactive systems over cospans. In *Proc. of LICS '05*, pages 311–320. IEEE, 2005.

[Tar55]   Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

APPENDIX A. SUPPLEMENTARY MATERIAL ON ADHESIVE CATEGORIES

We recall here the definition of adhesive categories [LS05]. We do not provide any introduction to basic categorical constructions such as products, pullbacks and pushouts, instead referring the reader to Sections 5 and 9 of [BW99].

Definition A.1 (adhesive categories)**.** A category is called *adhesive* if
- it has pushouts along monos;
- it has pullbacks;
- pushouts along monos are *Van Kampen* (VK) squares.

Referring to Figure 13, a VK square is a pushout such as (*i*), such that for each commuting cube as in (*ii*) having (*i*) as bottom face and the back faces of which are pullbacks, the front faces are pullbacks if and only if the top face is a pushout.
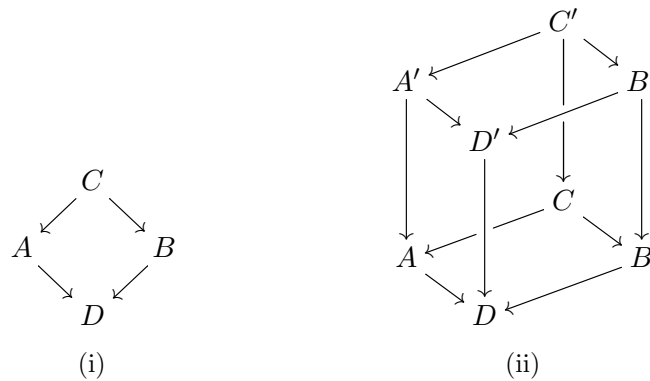

Figure 13: A pushout square (*i*), and a commutative cube (*ii*).

The motivation for using adhesive categories is that they are a suitable categorical framework for reasoning one rewriting of abstract objects, in the spirit of graph rewriting.