Math. Struct. in Comp. Science (2009), *vol.* 19, *pp.* 245–296. © 2009 Cambridge University Press
doi:10.1017/S0960129508007202 First published online 13 January 2009 Printed in the United Kingdom

Correctness of high-level transformation systems relative to nested conditions[†]

ANNEGRET HABEL and KARL-HEINZ PENNEMANN

Computing Science, D-26111 Oldenburg, Germany
Email: {habel;pennemann}@Informatik.Uni-Oldenburg.de

Received 24 August 2007; revised 11 June 2008

In this paper we introduce the notions of nested constraints and application conditions, short nested conditions. For a category associated with a graphical representation such as graphs, conditions are a graphical and intuitive, yet precise, formalism that is well suited to describing structural properties. We show that nested graph conditions are expressively equivalent to first-order graph formulas. A part of the proof includes transformations between two satisfiability notions of conditions, namely $\mathscr{M}$-satisfiability and $\mathscr{A}$-satisfiability. We consider a number of transformations on conditions that can be composed to construct constraint-guaranteeing and constraint-preserving application conditions, weakest preconditions and strongest postconditions. The restriction of rule applications by conditions can be used to correct transformation systems by pruning transitions leading to states violating given constraints. Weakest preconditions and strongest postconditions can be used to verify the correctness of transformation systems with respect to pre- and postconditions.

1. Introduction

In the context of the increasingly large and more complex systems that hardware and software engineers have to construct, visual modelling techniques can be expected to play a key role in the future. More than 30 years ago, graph replacement was proposed as a generalisation of string replacement with the intention of producing an operational notation that captures and inherits the advantage of graphs – the ability to directly visualise relationships between elements. In particular, the operational behaviour of structure changing systems such as 'mobile' systems (in the sense of dynamically changing communication topologies) is suited to modelling by graph transformation rules (Ehrig *et al.* 1991).

However, surprisingly little effort has been made to transfer the idea of a graph-based formalism to logical languages – until now. In this paper, we investigate nested conditions for graphs and high-level structures (Heckel and Wagner 1995; Koch *et al.* 2005; Ehrig *et al.* 2006b). For the category of graphs, nested conditions are a visual and intuitive, yet precise formalism, that is well suited to describing structural properties of system states. We propose to model structure-changing systems by a graph or high-level replacement system and to model the required system properties using nested conditions.

[†] This work is supported by the German Research Foundation (DFG), grants GRK 1076/1 (Graduate School on Trustworthy Software Systems) and HA 2936/2 (Development of Correct Graph Transformation Systems).

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

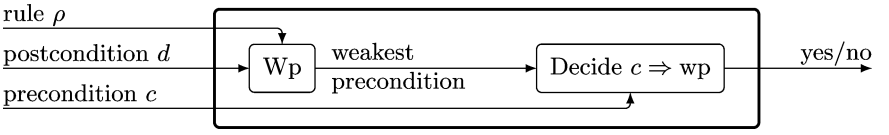


Figure 1. Decider for the correctness problem for transformation rules

However, the use of visual modelling techniques alone does not guarantee the correctness of a design. As standards for trustworthy systems grow, there is an increasing need for methods that support the development of correct transformation systems. We investigate two approaches:

1 the construction of constraint-guaranteeing and weaker constraint-preserving conditions that restrict the applicability of rules to correct situations;
2 the verification of transformation systems with respect to pre- and postconditions.

A well-known method is to construct a weakest precondition relative to the postcondition and to prove that the precondition implies the weakest precondition (Dijkstra 1976; Dijkstra and Scholten 1989):

As proposed in Rensink (2004), we extend the existing notion of constraints and application conditions (conditions for short) to nested conditions, and show that nested graph conditions and first-order graph formulas are expressively equivalent. We describe the construction of constraint-guaranteeing and constraint-preserving conditions as well as the computation of weakest preconditions and strongest postconditions with the help of certain basic transformations on conditions. Additionally, we investigate two notions of condition satisfiability and rule matching, and show that the satisfiability and matching notions, respectively, are equivalent under reasonable assumptions. We are interested in conditions over graph-like structures. To avoid the need to carry out similar investigations for comparable structures, we abstract from specific definitions and conduct our research in the framework of weak adhesive high-level replacement categories, HLR for short. Therefore, our results hold for replacement-capable structures such as Petri-nets, graphs and hypergraphs. To illustrate the constructions presented, examples are given in the category of directed, labelled graphs.

The paper is organised as follows. In Section 2, we recall the notions of graphs and weak adhesive HLR categories as a framework for high-level structures, and recount their basic properties. In Section 3, we introduce nested conditions for weak adhesive HLR categories. In Sections 4 and 5, we investigate $\mathscr{M}$- and $\mathscr{A}$-satisfiability for conditions and $\mathscr{A}$- and $\mathscr{M}$-matching for rules. For weak adhesive HLR categories with epi-$\mathscr{M}$-factorisations, $\mathscr{M}$-initial object and a monomorphism class $\mathscr{M}$ strictly closed under decompositions, we show that $\mathscr{M}$- and $\mathscr{A}$-satisfiability and $\mathscr{A}$- and $\mathscr{M}$-matching are expressively equivalent. In Section 6, we prove the basic transformation results, which are applied in Section 7 to yield constraint-guaranteeing and constraint-preserving application conditions as well as weakest preconditions and strongest postconditions. Finally, in Section 8, we investigate the expressiveness of nested graph conditions and show that nested graph conditions and first-order graph formulas are expressively equivalent. A conclusion including suggestions for further work is given in Section 9.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

This paper follows on from Ehrig *et al.* (2006b), is a revised version of Habel and Pennemann (2005) and Habel and Pennemann (2006) and contains parts of Habel *et al.* (2006) concerning transformation rules. Additionally, we present transformations between graph conditions and semantically equivalent first-order graph formulas, consider the example of a handover-protocol in cellular networks and investigate the construction of strongest postconditions.

2. Graphs and high-level structures

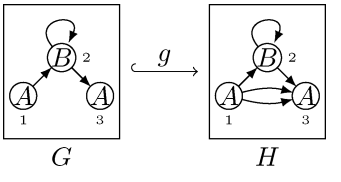
In this paper we consider graphs as a typical structure that is suited to modelling discrete aspects of system states. Our motivation for considering graphs is that they are a general structure and their graphical representation is able to visualise directly the relationships between elements. In this section we recall the basic notions of directed, labelled graphs (Ehrig 1979; Corradini *et al.* 1997) and generalise them to high-level structures (Ehrig *et al.* 1991). The idea behind the consideration of high-level structures is to avoid the need to repeat similar investigations for similar structures such as Petri-nets, graphs and hypergraphs.

Directed, labelled graphs and graph morphisms are defined as follows.

Definition 1 (graphs and graph morphisms). Let $\mathrm{C} = \langle \mathrm{C_V}, \mathrm{C_E} \rangle$ be a fixed, finite, disjoint label alphabet. A *graph* over C is a system $G = (\mathrm{V}_G, \mathrm{E}_G, \mathrm{s}_G, \mathrm{t}_G, \mathrm{l}_G, \mathrm{m}_G)$ consisting of two finite sets V_G and E_G of *nodes* (or *vertices*) and *edges*, *source* and *target functions* $\mathrm{s}_G, \mathrm{t}_G : \mathrm{E}_G \to \mathrm{V}_G$, and two *labelling functions* $\mathrm{l}_G : \mathrm{V}_G \to \mathrm{C_V}$ and $\mathrm{m}_G : \mathrm{E}_G \to \mathrm{C_E}$. A graph with an empty set of nodes is *empty* and denoted by $\varnothing$. A *graph morphism* $g : G \to H$ consists of two functions $g_\mathrm{V} : \mathrm{V}_G \to \mathrm{V}_H$ and $g_\mathrm{E} : \mathrm{E}_G \to \mathrm{E}_H$ that preserve sources, targets and labels, that is, $\mathrm{s}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{s}_G$, $\mathrm{t}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{t}_G$, $\mathrm{l}_H \circ g_\mathrm{V} = \mathrm{l}_G$ and $\mathrm{m}_H \circ g_\mathrm{E} = \mathrm{m}_G$. A morphism g is *injective* (*surjective*) if g_V and g_E are injective (surjective), and an *isomorphism* if it is both injective and surjective. The *composition* $h \circ g$ of g with a morphism $h : H \to M$ consists of the composed functions $h_\mathrm{V} \circ g_\mathrm{V}$ and $h_\mathrm{E} \circ g_\mathrm{E}$.

For simplicity, we sometimes consider a single labelling function $\mathrm{l}_G : \mathrm{V}_G + \mathrm{E}_G \to \mathrm{C_V} \cup \mathrm{C_E}$ with $\mathrm{l}_G(\mathrm{V}_G) \subseteq \mathrm{C_V}$ and $\mathrm{l}_G(\mathrm{E}_G) \subseteq \mathrm{C_E}$, where $+$ denotes the disjoint union. Of course, two labelling functions for nodes and edges, respectively, can be composed into one, and *vice versa.*

Example 1. Consider the graphs G and H and the injective graph morphism $g : G \to H$ given in the figure below. G and H are graphs over the label alphabet $\mathrm{C} = \langle \{A, B\}, \{\square\} \rangle$, where the symbol $\square$ stands for the invisible edge label and is not drawn. Note that the graph G contains a loop, while the graph H contains two parallel edges and a loop.



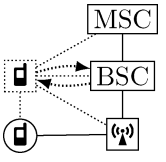
G H

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

In drawings of graphs, nodes are drawn as circles carrying the node label inside; edges are drawn as arrows pointing from the source to the target node and the edge label is placed next to the arrow. Arbitrary graph morphisms are drawn as normal arrows '$\rightarrow$', while injective graph morphisms are drawn as hooked arrows '$\hookrightarrow$'. For graph morphisms, the mapping of nodes is depicted by indices, if necessary.

Example 2 (GSM graph). In the Global System for Mobile Communications (GSM), mobile stations (MS) are connected by wireless links to the cellular network of their operators. Such a network consists of mobile services switching centres (MSC) managing a number of base station controllers (BSC) that control a number of base transceiver stations (BTS). In this running example, we model selected aspects of the handover procedure (ETSI TS 100 527) between two base station controllers of a mobile services switching centre, the so-called inter-BSC-intra-MSC handover. States of the GSM are represented as graphs. The following elements are modelled as labelled nodes: MSC and BSC, base transceiver stations, mobile stations, and mobile station records, that is, mobile stations as seen by the network and its components.

Relations between these elements are modelled by labelled edges: physical connections including wireless ones are modelled by solid edges (—), logical or virtual connections by dotted edges (·····), while certain messages exchanged between mobile stations and BSCs in the stages of the handover procedure are modelled by request ($\overset{req}{\cdots\blacktriangleright}$), command ($\overset{cmd}{\cdots\blacktriangleright}$) and complete edges ($\overset{cpl}{\cdots\blacktriangleright}$), respectively. Undirected edges represent a pair of directed edges in opposite directions.



In the sense of category theory, graphs and graph morphisms form a category.

Fact 1 (Ehrig 79). Graphs and graph morphisms form the category **Graphs**.

The following development is carried out with respect to objects and morphisms with specific properties, so-called weak adhesive HLR categories. The objects can be all kinds of structures that are of interest in computer science and mathematics, for example, Petri-nets, (hyper)graphs, and algebraic specifications, the morphisms can be net, (hyper)graph, and specification morphisms, respectively. Readers interested in the category-theoretic background of these concepts may consult, for example, Adamek *et al.* (1990) and Ehrig *et al.* (2006c).

For a category $\mathscr{C}$, let $\mathscr{A}$ be the class of all morphisms.

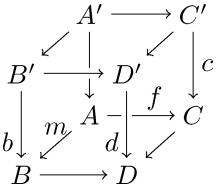
Definition 2 (weak adhesive HLR category). A category $\mathscr{C}$ with a morphism class $\mathscr{M} \subseteq \mathscr{A}$ is a *weak adhesive HLR category*, if the following properties hold:

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

(1) $\mathscr{M}$ is a class of monomorphisms closed under isomorphisms, composition and decomposition. That is, for morphisms $g \circ f\colon f \in \mathscr{M}$, g isomorphism (or *vice versa*) implies $g \circ f \in \mathscr{M}$; $f, g \in \mathscr{M}$ implies $g \circ f \in \mathscr{M}$; and $g \circ f \in \mathscr{M}$, $g \in \mathscr{M}$ implies $f \in \mathscr{M}$.
(2) $\mathscr{C}$ has pushouts and pullbacks along $\mathscr{M}$-morphisms, that is, pushouts and pullbacks, where at least one of the given morphisms is in $\mathscr{M}$, and $\mathscr{M}$-morphisms are closed under pushouts and pullbacks, that is, given a pushout (1), $m \in \mathscr{M}$ implies $n \in \mathscr{M}$ and, given a pullback (1), $n \in \mathscr{M}$ implies $m \in \mathscr{M}$.

$$\begin{array}{ccc} A & \longrightarrow & C \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle n} \\ B & \longrightarrow & D \end{array}$$

(3) Pushouts in $\mathscr{C}$ along $\mathscr{M}$-morphisms are weak VK-squares, that is, for any commutative cube in $\mathscr{C}$ where we have the pushout with $m \in \mathscr{M}$ and ($f \in \mathscr{M}$ or $b, c, d \in \mathscr{M}$) in the bottom and the back faces are pullbacks, we have that the top is a pushout if and only if the front faces are pullbacks.



Fact 2 (Ehrig *et al.* 2006c). The category $\langle \mathbf{Graphs}, \mathit{Inj} \rangle$ of graphs with class *Inj* of all injective graph morphisms is a weak adhesive HLR category. Further examples of weak adhesive HLR categories are the categories of hypergraphs with the class of all injective hypergraph morphisms, place-transition nets with the class of all injective net morphisms, and algebraic specifications with the class of all strict injective specification morphisms.

Weak adhesive HLR categories have a number of nice properties, called HLR properties (Ehrig *et al.* 1991).

Fact 3 (HLR properties of weak adhesive HLR categories). Given a weak adhesive HLR category $\langle \mathscr{C}, \mathscr{M} \rangle$, the following HLR conditions are satisfied:

(1) Pushouts along $\mathscr{M}$-morphisms are pullbacks.
(2) Pushout–pullback decomposition.
If the diagram (1)+(2) is a pushout, (2) a pullback, $w \in \mathscr{M}$ and ($l \in \mathscr{M}$ or $c \in \mathscr{M}$), then (1) and (2) are pushouts and also pullbacks:

$$\begin{array}{ccccc} A & \overset{c}{\longrightarrow} & C & \overset{r}{\longrightarrow} & E \\ {\scriptstyle l}\downarrow & (1) & {\scriptstyle s}\downarrow & (2) & \downarrow{\scriptstyle v} \\ B & \underset{u}{\longrightarrow} & D & \underset{w}{\longrightarrow} & F \end{array}$$

(3) Uniqueness of pushout complements for $\mathscr{M}$-morphisms.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Given morphisms $c: A \to C$ in $\mathscr{M}$ and $s: C \to D$, there is up to isomorphism at most one B with $l: A \to B$ and $u: B \to D$ such that diagram (1) is a pushout.

Proof. See Lack and Sobociński (2004) and Ehrig *et al.* (2006c). □

For some results in this paper, we use the following properties.

Definition 3. Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category. $\langle \mathscr{C}, \mathscr{M} \rangle$ has *pushouts* if for each pair of arbitrary morphisms there is a pushout. $\langle \mathscr{C}, \mathscr{M} \rangle$ has an *epi-$\mathscr{M}$-factorisation* if for every morphism there is an epi-mono-factorisation with monomorphism in $\mathscr{M}$, and this decomposition is unique up to isomorphism. $\langle \mathscr{C}, \mathscr{M} \rangle$ has an *$\mathscr{M}$-initial object* if there is some object I in $\mathscr{C}$ such that for every object G in $\mathscr{C}$ there exists a unique morphism $i_G: I \to G$ and i_G is in $\mathscr{M}$. We refer to i_G as an *initial $\mathscr{M}$-morphism* to G. $\mathscr{M}$ is *strictly* closed under decomposition if $g \circ f \in \mathscr{M}$ implies $f \in \mathscr{M}$.

Fact 4. The weak adhesive HLR category $\langle$**Graphs**, *Inj*$\rangle$ with *Inj* the class of all injective graph morphisms has pushouts, an epi-*Inj*-factorisation and an *Inj*-initial object, and *Inj* is strictly closed under decompositions.

Proof. The category **Graphs** has pushouts, see, for example, Ehrig *et al.* (2006c, Fact A.19). It has epi-mono factorisations (A.15) and the monomorphisms are exactly those morphisms that are injective (A.14 and Fact 2.15). The empty graph is an *Inj*-initial object in $\langle$**Graphs**, *Inj*$\rangle$. The class *Inj* of injective graph morphisms is strictly closed under decompositions because the class of injective mappings also has this property. □

3. Conditions and rules

In the search for a graphical formalism to specify sets of objects as well as morphisms, we investigate nested conditions for graphs and high-level structures. We use the framework of weak adhesive HLR categories and introduce nested conditions and rules for high-level structures like Petri nets, (hyper)graphs and algebraic specifications.

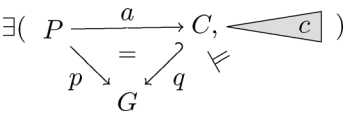
Assumption 1. We assume that $\langle \mathscr{C}, \mathscr{M} \rangle$ is a weak adhesive HLR category with $\mathscr{M}$-initial object I.

Syntactically, nested conditions may be seen as a tree of morphisms equipped with certain logical symbols such as quantifiers and connectives.

Definition 4 (conditions). A (*nested*) *condition* over an object P is of the form true or $\exists(a, c)$, where $a: P \to C$ is a morphism and c is a condition over C. Moreover, Boolean formulas over conditions over P yield conditions over P, that is, $\neg c$ and $\wedge_{j \in J} c_j$ are (Boolean) conditions over P, where J is an index set and c, $(c_j)_{j \in J}$ are conditions over P. Additionally, $\exists a$ abbreviates $\exists(a, \text{true})$, $\forall(a, c)$ abbreviates $\neg\exists(a, \neg c)$, false abbreviates $\neg$true, $\vee_{j \in J} c_j$ abbreviates $\neg \wedge_{j \in J} \neg c_j$ and $c \Rightarrow d$ abbreviates $\neg c \vee d$.

Every object and morphism *satisfies* true. A morphism p *satisfies* a condition $\exists(a, c)$ if there exists a morphism q in $\mathscr{M}$ such that $q \circ a = p$ and q satisfies c.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press



An object G *satisfies* a condition $\exists(a,c)$, if the condition is over the initial object I and the initial morphism $i_G: I \to G$ satisfies the condition. The satisfaction of conditions by objects and morphisms is extended onto Boolean conditions in the usual way. We write $G \models c$ and $p \models c$ to denote the fact that the object G or morphism p, respectively, satisfies c. Two conditions c and c' are *equivalent*, denoted $c \equiv c'$, if, for all morphisms p, $p \models c$ if and only if $p \models c'$.

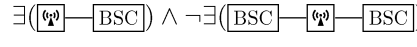
In the context of objects, conditions are also called *constraints* (Heckel and Wagner 1995) and in the context of rules, conditions are also called *application conditions.*

Example 3. For graph morphisms with domain $\bigcirc$, the graph condition $c = \exists(\bigcirc \to \bigcirc\!\!\rightarrow\!\!\bigcirc) \vee \exists(\bigcirc \to \bigcirc\!\!\leftarrow\!\!\bigcirc)$ has the meaning 'The image of the node has a proper outgoing or incoming edge'. For graphs, one may consider the universal closure $\forall(\varnothing \to \bigcirc, c)$ with the meaning 'All nodes have an outgoing or incoming proper edge'.

Notation. For a morphism $a: P \to C$ in a condition, if the domain P can be unambiguously inferred, we will just show the codomain C, that is, if it is known over which object a condition is imposed. For instance, as constraints are always over the initial object, the constraint $\forall(\varnothing \to \bigcirc, \exists(\bigcirc \to \bigcirc\!\!\rightarrow\!\!\bigcirc))$ meaning 'Every node has an outgoing edge to a distinct node' can be represented by $\forall(\bigcirc, \exists(\bigcirc\!\!\rightarrow\!\!\bigcirc))$.

Example 4 (GSM conditions). Consider the GSM graph model in Example 2. Structural requirements on the GSM states can be described by graph constraints, that is, graph conditions over the empty graph.

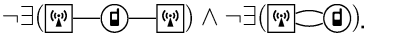
(c_1) Every transceiver station has exactly one controller:



(c_2) No mobile station is associated to two or more mobile station records and *vice versa*:

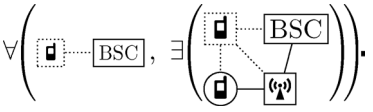
(c_3) Every mobile station record is associated with a base transceiver station:

(c_4) There is no mobile station connected to two or more base transceiver stations and no mobile station with two channels at a transceiver station:

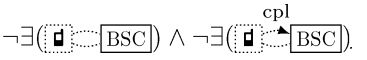


https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

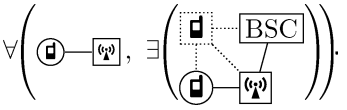
(c_5) Every logical connection between a base transceiver station and a mobile station record corresponds to a physical connection between the network and the mobile station:



(c_6) Between a base transceiver station and mobile station record, there cannot be two logical connections, or a logical connection and a handover complete message edge:



(c_7) Every wireless connection between a mobile and a transceiver station is reflected by the network:



Let $\mathit{consistency} = \wedge_{j=1}^{7} c_j$ denote the conjunction of the GSM conditions. A GSM graph is said to be *consistent* if it satisfies the graph condition *consistency*.

Remark. Nested conditions, as proposed in Rensink (2004), subsume the previous notions of basic constraints and application conditions. For every condition in the sense of Koch *et al.* (2005) and Ehrig *et al.* (2006b), there is an equivalent condition according to Definition 4. The notion of conditions presented here corresponds to the notions in Habel and Pennemann (2005) with one difference: only constraints over the initial object are considered. This is no restriction, as every constraint over P in the sense of Habel and Pennemann (2005) can be transformed into an equivalent constraint over the initial object I, for example, $\exists(a, c)$ may be transformed into $\forall(i_P, \exists(a, c))$. Using this modification, we obtain a common notion of equivalence for objects and morphisms.

In the following, we show that the concepts of constraints and application conditions coincide in terms of equivalence. We use this fact in Section 4 in defining transformations for conditions that may be used for constraints as well as application conditions. Note that the add-on [in $\mathscr{M}$] is optional.

Fact 5 (equivalence). For conditions c, c' over I, for all morphisms p [in $\mathscr{M}$], $p \models c \Leftrightarrow p \models c'$ if and only if for all objects G, $G \models c \Leftrightarrow G \models c'$.

Proof.

— **Only if:**
Assume that for all morphisms p [in $\mathscr{M}$] we have $p \models c \Leftrightarrow p \models c'$. In particular, this means that for objects G we have $i_G \models c \Leftrightarrow i_G \models c'$ where i_G is the initial $\mathscr{M}$-morphisms to G. By Definition 4, we conclude that for every object G we have $G \models c \Leftrightarrow G \models c'$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

— **If:**
Conversely, if for every object G we have $G \models c \Leftrightarrow G \models c'$, then $i_G \models c \Leftrightarrow i_G \models c'$ where i_G is the initial $\mathcal{M}$-morphism to G. Let $p: I \to G$ be any morphism. By the $\mathcal{M}$-initiality of I, we know that $p = i_G$ [in $\mathcal{M}$], so we conclude for all morphisms p [in $\mathcal{M}$] that $p \models c \Leftrightarrow p \models c'$. □

Conditions are allowed to consist of morphisms not in $\mathcal{M}$. In the context of rules, such conditions are useful as application conditions to specify the identification/non-identification of elements. However, for constraints, the use of morphisms not in $\mathcal{M}$ does not increase the expressiveness. We show that every condition over I can be transformed into $\mathcal{M}$-normal form.

Definition 5 ($\mathcal{M}$-normal form). A condition is in *$\mathcal{M}$-normal form* if for all subconditions of the form $\exists(a, c)$, the morphism a is in $\mathcal{M}$.

Fact 6 ($\mathcal{M}$-normal form). For every condition c over I, there is an equivalent condition $\mathcal{M}\mathrm{NF}(c)$ in $\mathcal{M}$-normal form.

Construction. Define $\mathcal{M}\mathrm{NF}(\mathrm{true}) = \mathrm{true}$ and

$$\mathcal{M}\mathrm{NF}(\exists(a, c')) = \begin{cases} \mathrm{false} & \text{if } a \notin \mathcal{M} \\ \exists(a,\ \mathcal{M}\mathrm{NF}(c')) & \text{otherwise.} \end{cases}$$

For Boolean conditions the transformation is extended in the usual way, that is,

$$\mathcal{M}\mathrm{NF}(\neg c') = \neg \mathcal{M}\mathrm{NF}(c')$$

and

$$\mathcal{M}\mathrm{NF}(\wedge_{j \in J}\ c_j) = \wedge_{j \in J}\ \mathcal{M}\mathrm{NF}(c_j).$$

Proof. Obviously, for every condition c, $\mathcal{M}\mathrm{NF}(c)$ is in $\mathcal{M}$-normal form. By structural induction, we show for every $p \in \mathcal{M}$ that $p \models c$ if and only if $p \models \mathcal{M}\mathrm{NF}(c)$.

— **Basis:**
For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathcal{M}\mathrm{NF}(\mathrm{true}) = \mathcal{M}\mathrm{NF}(c)$.

— **Hypothesis:**
Assume for every $p \in \mathcal{M}$ that $p \models c'$ if and only if $p \models \mathcal{M}\mathrm{NF}(c')$, and that $p \models c_j$ if and only if $p \models \mathcal{M}\mathrm{NF}(c_j)$ for every $j \in J$.

— **Step:**
Let $c = \exists(a, c')$. If $a \in \mathcal{M}$, we have for every $p \in \mathcal{M}$ that

$$p \models \mathcal{M}\mathrm{NF}(c) = \mathcal{M}\mathrm{NF}(\exists(a, c')) = \exists(a,\ \mathcal{M}\mathrm{NF}(c'))$$

if and only if $p \models \exists(a, c') = c$ by the definition of $\mathcal{M}\mathrm{NF}$ and the induction hypothesis. Otherwise, $a \notin \mathcal{M}$, and $\mathcal{M}\mathrm{NF}(\exists(a, c')) = \mathrm{false}$. We now show for every $p \in \mathcal{M}$ that $p \models \exists(a, c')$ if and only if $p \models \mathrm{false}$:

– **Only if:**
Assume morphism p in $\mathcal{M}$ satisfies $\exists(a, c')$. Then there is a morphism q in $\mathcal{M}$ with $p = q \circ a$. However, p, q in $\mathcal{M}$ and $\mathcal{M}$ closed under decomposition implies $a \in \mathcal{M}$, which gives a contradiction.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

– **If:**
No morphism satisfies false, so for every morphism p, we have $p \models \text{false}$ implies $p \models \exists(a, c')$.

For Boolean formulas over conditions, the statement follows directly from the definitions and the induction hypothesis. For $c = \neg c'$, we have for every $p \in \mathcal{M}$ that

$$p \models \mathcal{M}\text{NF}(c) = \mathcal{M}\text{NF}(\neg c') = \neg \mathcal{M}\text{NF}(c')$$

if and only if $p \models \neg c' = c$. For $c = \wedge_{j \in J}\, c_j$, we have for every $p \in \mathcal{M}$ that

$$p \models \mathcal{M}\text{NF}(c) = \mathcal{M}\text{NF}(\wedge_{j \in J}\, c_j) = \wedge_{j \in J}\, \mathcal{M}\text{NF}(c_j)$$

if and only if $p \models \wedge_{j \in J}\, c_j = c$. □

Remark ($\mathcal{M}$-satisfiability). The satisfaction of a condition is established by the presence and absence of certain morphisms from the objects within the condition to the tested object. The presented satisfiability notion restricts these morphisms to the class $\mathcal{M}$: no identification of nodes and edges is allowed. Hence, explicit counting such as the existence/non-existence of n nodes or n edges is easily expressible. We speak of *$\mathcal{M}$-satisfiability* and *$\mathcal{M}$-satisfiable conditions.*

Besides $\mathcal{M}$-satisfiability, we may also consider *$\mathcal{A}$-satisfiability*/*$\mathcal{A}$-satisfiable conditions*, where $\mathcal{A}$ denotes the class of all morphisms. The definition of $\mathcal{A}$-satisfiability is obtained from the definition of $\mathcal{M}$-satisfiability by replacing all occurrences of $\mathcal{M}$ by $\mathcal{A}$, or by deleting all occurrences of 'in $\mathcal{M}$', respectively.

Definition 6 ($\mathcal{A}$-satisfiability). Every object and morphism *$\mathcal{A}$-satisfies* true. An object G *$\mathcal{A}$-satisfies* a condition $\exists(a, c)$ if the condition is over the initial object I and the initial morphism $i_G : I \to G$ $\mathcal{A}$-satisfies the condition. A morphism p *$\mathcal{A}$-satisfies* a condition $\exists(a, c)$ if there exists a morphism q such that $q \circ a = p$ and q $\mathcal{A}$-satisfies c. The $\mathcal{A}$-satisfaction of conditions by objects and morphisms is extended onto Boolean conditions in the usual way. We write $G \models_{\mathcal{A}} c$ or $p \models_{\mathcal{A}} c$ to denote the fact that the object G or the morphism p, respectively, $\mathcal{A}$-satisfies c. Two conditions c and c' are *$\mathcal{A}$-equivalent*, denoted $c \equiv_{\mathcal{A}} c'$, if for all morphisms p we have $p \models_{\mathcal{A}} c$ if and only if $p \models_{\mathcal{A}} c'$.

Remark ($\mathcal{A}$-satisfiability). $\mathcal{A}$-satisfiability allows nodes or edges of the conditions to be identified and is closely related to the satisfiability of first-order formulas, as indicated in Section 8 for the case of directed, labelled graphs. We will see in Section 4 that, under reasonable assumptions, $\mathcal{A}$- and $\mathcal{M}$-satisfiability are expressively equivalent. Unless explicitly stated, theorems concern $\mathcal{M}$-satisfiable conditions.

Example 5 ($\mathcal{A}$-satisfiability). The meaning of the graph condition $\exists(\bigcirc_1 \to \bigcirc_1 \to \bigcirc)$ for graph morphisms with domain $\bigcirc_1$ with respect to $\mathcal{A}$-satisfiability is 'There exists an outgoing edge (proper edge or loop)'.

In the following, we consider rules with application conditions (Ehrig *et al.* 2006b; Habel and Pennemann 2005). Examples and pointers to the literature can be found in Ehrig (1979) and Corradini *et al.* (1997).

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

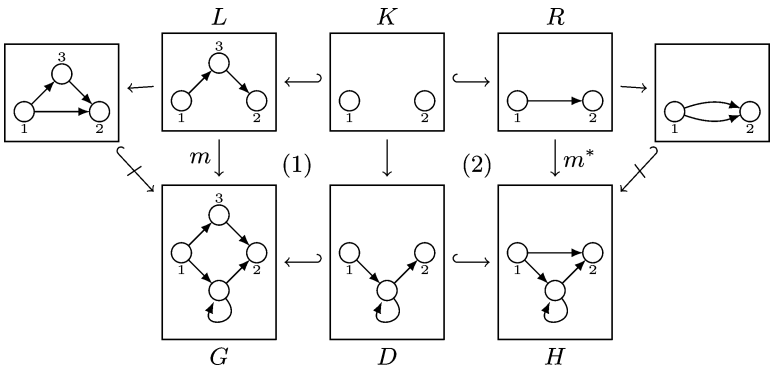
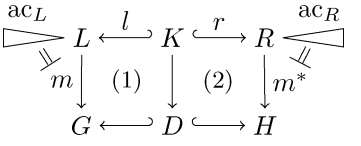


Figure 2. Application of a graph replacement rule

Definition 7 (rules). A *plain rule* $\rho = \langle L \hookleftarrow K \hookrightarrow R \rangle$ consists of two morphisms in $\mathscr{M}$ with a common domain K. L is called the left-hand side, R is called the right-hand side and K is called the interface. An *application condition* $\mathrm{ac} = \langle \mathrm{ac}_L, \mathrm{ac}_R \rangle$ for ρ consists of two conditions over L and R, respectively. A *rule* $\hat{\rho} = \langle \rho, \mathrm{ac} \rangle$ consists of a plain rule ρ and an application condition ac for ρ.



Given a plain rule ρ and a morphism $K \to D$, a *direct derivation* consists of two pushouts (1) and (2). We write $G \Rightarrow_{\rho,m,m^*} H$, or $G \Rightarrow_{\rho} H$ for short, and say that m is the *match* and m^* is the *comatch* of ρ in H. Given a rule $\hat{\rho} = \langle \rho, \mathrm{ac} \rangle$ and a morphism $K \to D$, there is a *direct derivation* $G \Rightarrow_{\hat{\rho},m,m^*} H$ if $G \Rightarrow_{\rho,m,m^*} H$, $m \models \mathrm{ac}_L$ and $m^* \models \mathrm{ac}_R$.

Example 6. Consider the graph replacement rule with inclusions given in Figure 2. Then there is an injective graph morphism from the left-hand side L of the rule to the graph G, which is fixed by the numbers beside the nodes, that satisfies the so-called gluing condition (Ehrig 1979). The removal of $m(L-K)$ yields the graph D and the addition $R-K$ yields the graph H. The match m satisfies the left application condition $\neg\exists(\bullet \to \bullet \to \bullet \rightrightarrows \bullet)$; the comatch m^* satisfies the right application condition $\neg\exists(\bullet \to \bullet \to \bullet \rightleftarrows \bullet)$. Both application conditions guarantee that multiple edges cannot arise.

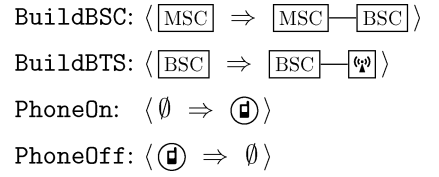
Remark ($\mathscr{A}$- and $\mathscr{M}$-matching). Matching, which is also called *$\mathscr{A}$-matching*, is the process of finding arbitrary matching morphisms. Besides $\mathscr{A}$-matching, one may also consider matches in $\mathscr{M}$ only, called *$\mathscr{M}$-matching* and denoted $\Rightarrow_{\mathscr{M}}$. $\mathscr{M}$-matching restricts the applicability of rules: no identification of nodes and edges is allowed. Moreover, $\mathscr{M}$-matching allows explicit counting such as the deletion of n nodes or edges. We will show in Section 5 that, under reasonable assumptions, $\mathscr{A}$ and $\mathscr{M}$-matching are expressively equivalent.

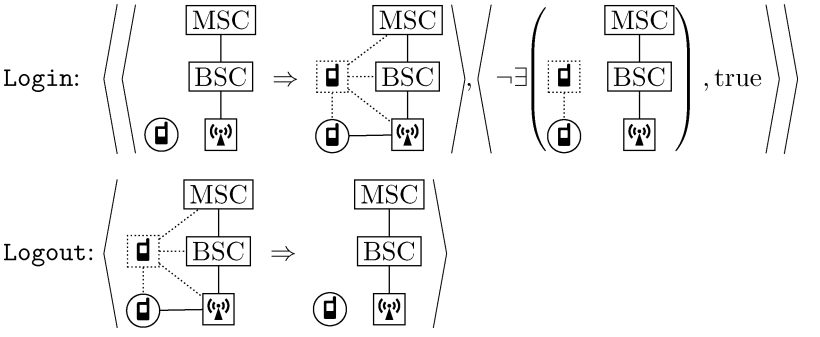
https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

A set of transformation rules constitutes a transformation system.

Definition 8 (transformation system). A *transformation system* is a set of rules $\mathscr{R}$. A *derivation* is a sequence of direct derivations of rules in $\mathscr{R}$.

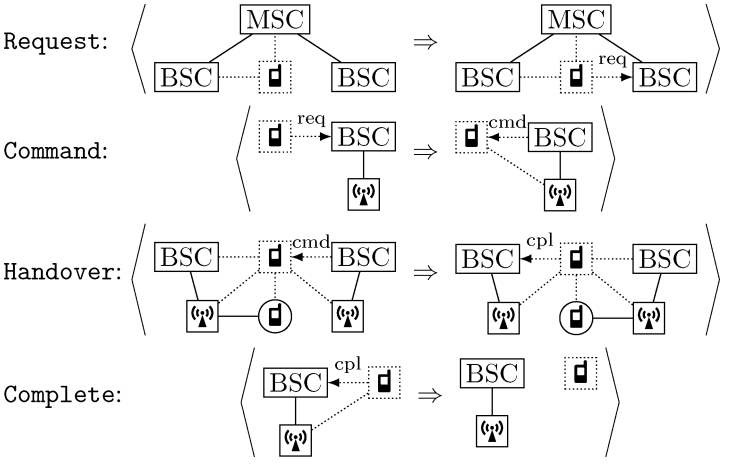
Example 7 (GSM transformation system). Consider the GSM model in Example 2. For simplicity, we select $\mathscr{M}$-matching and denote a rule $\langle L \hookleftarrow K \hookrightarrow R \rangle$ by $\langle L \Rightarrow R \rangle$ for short, where K consists of all elements common to L and R. The dynamics of the GSM network is modelled by six graph transformation rules: `BuildBSC` and `BuildBTS` model the extension of infrastructure; `PhoneOn` and its inverse `PhoneOff` model the operational status of a mobile station; and `Login` and its inverse `Logout` model the creation and deletion of a mobile station record.





The operational behaviour of the handover protocol is modelled by the graph transformation rules `Request`, `Command`, `Handover` and `Complete`, representing a division of the handover operation into four phases. In the first phase, the necessity of the handover is detected by the base station controller based on the information last received from the transceiver station. A handover `Request` message is sent to the MSC, which selects a new BSC and forwards the message to it. In the second phase, the new BSC transfers the handover `Command` through the old BSC to the MS, assigning it to a radio channel of one of its base transceiver stations. In the third phase, the MS disconnects the old radio channel and establishes the radio channel with the new BSC. If successful, the connection to the old BSC is reestablished *via* the new BSC through its transceiver station. In the final phase, the MS sends the handover `Complete` message to the old BSC, which releases the old radio channels on its transceiver station.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press



The question arises as to whether the GSM transformation system is or is not correct with respect to the pre- and postcondition *consistency*, that is, whether the application of the GSM rules on a consistent GSM graph always results in a consistent graph.

4. Satisfiability of conditions

In this section we investigate the different satisfiability notions for conditions and show that $\mathscr{A}$-satisfiability and $\mathscr{M}$-satisfiability are expressively equivalent. First, there is a transformation from $\mathscr{A}$- to $\mathscr{M}$-satisfiability. When pushouts for arbitrary pairs of morphisms exist, this transformation of conditions shows some similarity to the transformation of constraints into application conditions in Ehrig *et al.* (2006b) and the original paper of Heckel and Wagner (1995). When the existence of pushouts cannot be guaranteed, one can resort to a modified transformation, which requires an $\mathscr{M}$-initial object and makes use of the existence of pushouts along $\mathscr{M}$-morphisms.

Theorem 1 (from $\mathscr{A}$- to $\mathscr{M}$-satisfiability). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object and epi-$\mathscr{M}$-factorisation. There is a transformation Msat such that, for every condition c and for every morphism p, we have $p \models \mathrm{Msat}(c) \Leftrightarrow p \models_{\mathscr{A}} c$, and for every condition c over I and for object G, we have $G \models \mathrm{Msat}(c) \Leftrightarrow G \models_{\mathscr{A}} c$.

$\mathscr{A}$-satisfiable conditions allow elements to be identified. The idea of the following construction is to consider a disjunction of $\mathscr{M}$-satisfiable conditions, one for each possible level of identification. The construction of $\mathrm{Msat}(c)$ with a single parameter of type condition is defined by the auxiliary transformation $\mathrm{Msat}(e, c)$ where the first parameter is of type epimorphism and the second parameter is of type condition.

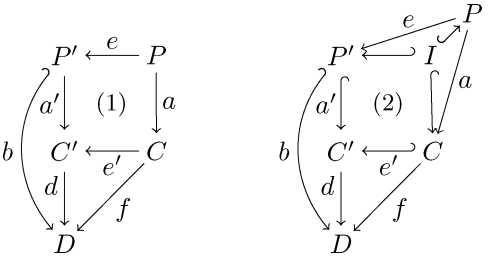
Construction. For a condition c over P, define $\mathrm{Msat}(c) = \vee_{e \in \mathscr{E}'} \exists(e,\ \mathrm{Msat}(e, c))$ where the set $\mathscr{E}'$ ranges over all epimorphisms with domain P. For every epimorphism e, the transformation $\mathrm{Msat}(e, c)$ is defined inductively by $\mathrm{Msat}(e, \mathrm{true}) = \mathrm{true}$ and

$$\mathrm{Msat}(e, \exists(a, c')) = \vee_{d \in \mathscr{E}} \exists(b,\ \mathrm{Msat}(f, c')),$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

where, if $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts, (1) is the pushout of the morphisms a and e leading to morphisms $a' : P' \to C'$ and $e' : C \to C'$ and the set $\mathscr{E}$ ranges over all epimorphisms d with domain C' such that $b = d \circ a'$ is in $\mathscr{M}$ and $f = d \circ e'$.

Alternatively, construct the pushout (2) of the initial $\mathscr{M}$-morphisms $i_C : I \to C$ and $i_{P'} : I \to P'$ leading to morphisms $a' : P' \to C'$ and $e' : C \to C'$ and the set $\mathscr{E}$ ranges over all epimorphisms d with domain C' such that $b = d \circ a'$ is in $\mathscr{M}$, $f = d \circ e'$ and $b \circ e = f \circ a$.



For Boolean formulas over conditions, the transformations Msat(_) and Msat(_, _) are extended in the usual way.

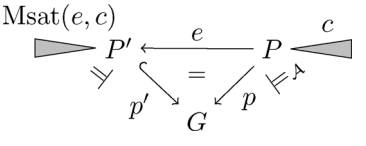
Example 8. The meaning of condition $c = \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$ for graph morphisms with respect to $\mathscr{A}$-satisfiability is 'There exists an edge from the image of node 1 to the image of node 2'. Msat(c) is constructed as follows:

$$\begin{aligned}
\text{Msat}(c) &= \vee_{e \in \mathscr{E}} \exists(e,\ \text{Msat}(e, c)) \\
&= \exists(\text{id},\ \text{Msat}(\text{id}, c)) \vee \exists(f,\ \text{Msat}(f, c)) \\
&= \exists(\text{id},\ \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)) \vee \exists(f,\ \exists(\bigcirc_1 \to \bigcirc_1 \circlearrowright)) \\
&\equiv \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2) \vee \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_{1=2} \circlearrowright)
\end{aligned}$$

where $f : \bigcirc_1 \ \bigcirc_2 \to \bigcirc_{1=2}$. The meaning of Msat($c$) for $\mathscr{M}$-satisfiability is 'There is a proper edge from the image of node 1 to a distinct image of node 2, or both images are identical and there is a loop'.

Before we give the proof of Theorem 1, we prove a property for the auxil1iary transformation Msat(e, c).

Lemma 1 (Msat(e, c)**).** Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object and epi-$\mathscr{M}$-factorisation. For every condition c over P, every epimorphism $e : P \to P'$ and every morphism $p' : P' \to G$ in $\mathscr{M}$, we have $p' \models \text{Msat}(e, c) \Leftrightarrow p' \circ e \models_{\mathscr{A}} c$.



Proof. The proof is by structural induction.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

— **Basis:**
For $c = \text{true}$, we have $c = \text{true} = \text{Msat}(e, \text{true}) = \text{Msat}(e, c)$.
— **Hypothesis:**
Assume the statement holds for condition c'.
— **Step:**
For $c = \exists(a, c')$, we distinguish two cases:

– **Case 1:** $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts of arbitrary morphisms.

- **Only if:**
Let $p' \models \text{Msat}(e, \exists(a, c')) = \vee_{d \in \mathscr{E}} \exists(b, \text{Msat}(f, c'))$. There is some epimorphism $d: C' \to D$ with $b = d \circ a'$ in $\mathscr{M}$ and $f = d \circ e'$ such that $p' \models \exists(b, \text{Msat}(f, c'))$. By the definition of $\mathscr{M}$-satisfiability, there is some $q': D \to G$ in $\mathscr{M}$ such that $q' \circ b = p'$ and $q' \models \text{Msat}(f, c')$. Now define $q = q' \circ f$. Taking this together with $q' \circ b = p'$, $b = d \circ a'$, $a' \circ e = e' \circ a$, $f = d \circ e'$, we observe $q \circ a = p$ ($p \models \exists a$). By the induction hypothesis, $q \models_{\mathscr{A}} c'$, so $p \models_{\mathscr{A}} \exists(a, c')$.

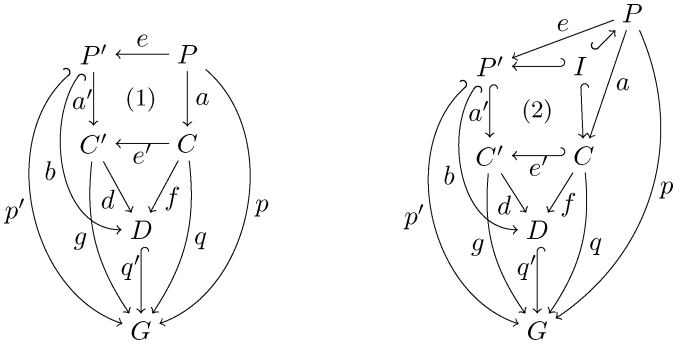
- **If:**
Let $p \models_{\mathscr{A}} \exists(a, c')$. By the definition of $\mathscr{A}$-satisfiability, there is some $q: C \to G$ such that $q \circ a = p$ and $q \models_{\mathscr{A}} c'$. Let (1) be the pushout of $a: P \to C$ and $e: P \to P'$. By the universal property of pushouts, there is some $g: C' \to G$ with $g \circ a' = p'$ and $g \circ e' = q$. Let $g = q' \circ d$ be an epi-$\mathscr{M}$-factorisation of g with epimorphism d and monomorphism q' in $\mathscr{M}$, $b = d \circ a'$, and $f = d \circ e'$. Since p' and q' are in $\mathscr{M}$, we have $q' \circ b = p'$, and $\mathscr{M}$ is closed under decomposition, the morphism b is in $\mathscr{M}$ and d is in $\mathscr{E}$. As $p' = g \circ a'$, $g = q' \circ d$ and $b = d \circ a'$, we observe $p' = q' \circ b$ ($p' \models \exists b$). By the induction hypothesis, $q' \models \text{Msat}(f, c')$, so $p' \models \vee_{d \in \mathscr{E}} \exists(b, \text{Msat}(f, c')) = \text{Msat}(e, \exists(a, c'))$.

– **Case 2:** $\langle \mathscr{C}, \mathscr{M} \rangle$ has only pushouts along $\mathscr{M}$-morphisms.

- **Only if:**
The proof is the same as above using the morphism $f: C \to D$ and the induction hypothesis.

- **If:**
Let $p \models_{\mathscr{A}} \exists(a, c')$. By the definition of $\mathscr{A}$-satisfiability, there is some $q: C \to G$ such that $q \circ a = p$ and $q \models_{\mathscr{A}} c'$. Since $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts along $\mathscr{M}$-morphisms, we can construct the pushout (2) of the $\mathscr{M}$-morphisms $i_C: I \to C$ and $i_{P'}: I \to P'$ leading to morphisms $a': P' \to C'$ and $e': C \to C'$. By the universal property of pushouts, there is some $g: C' \to G$ with $g \circ a' = p'$ and $g \circ e' = q$. Let $g = q' \circ d$ be an epi-$\mathscr{M}$-factorisation of g with epimorphism d and monomorphism q' in $\mathscr{M}$, $b = d \circ a'$, and $f = d \circ e'$. Since p' and q' are in $\mathscr{M}$, $q' \circ b = p'$ and $\mathscr{M}$ is closed under decomposition, the morphism b is in $\mathscr{M}$ and d is in $\mathscr{E}$. It turns out that $q' \circ b \circ e = q' \circ f \circ a$ and, by the monomorphism property of q', we have $b \circ e = f \circ a$. As b is in $\mathscr{M}$ and $f \circ a = b \circ e$, the tuple $\langle D, b, f \rangle$ belongs to the construction. As $p' = g \circ a'$, $g = q' \circ d$ and $b = d \circ a'$, we observe that $p' = q' \circ b$ ($p' \models \vee_{d \in \mathscr{E}} \exists b$). By the induction hypothesis, $q' \models \text{Msat}(f, c')$, so $p' \models \vee_{d \in \mathscr{E}} \exists(b, \text{Msat}(f, c')) = \text{Msat}(e, \exists(a, c'))$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press



For Boolean formulas over conditions, the statement follows from the definitions and the induction hypothesis. Consequently, the statement holds for all conditions. □

Theorem 1 follows directly from Lemma 1.

Proof of Theorem 1. Let $p: P \to G$ be a morphism and $p = p' \circ e$ be an epi-$\mathcal{M}$-factorisation of p with epimorphism e and monomorphism p' in $\mathcal{M}$. By Lemma 1, the definitions of $\mathcal{M}$-satisfiability, $\models$ and $\mathrm{Msat}(c)$ and the uniqueness of epi-$\mathcal{M}$-factorisations up to isomorphism, we obtain the statement for Msat:

$$\begin{aligned} p \models_{\mathscr{A}} c &\Leftrightarrow p' \models \mathrm{Msat}(e, c) \\ &\Leftrightarrow p \models \exists(e,\ \mathrm{Msat}(e, c)) \\ &\Leftrightarrow p \models \vee_{e \in \mathscr{E}'} \exists(e,\ \mathrm{Msat}(e, c)) \\ &= \mathrm{Msat}(c). \end{aligned}$$

Fact 5 lifts the result to objects and conditions over I. □

Remark. The case differentiation '$\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts' in the first construction of Theorem 1 is not required if one assumes that the conditions are in $\mathscr{M}$-normal form. However, in the context of $\mathscr{A}$-satisfiability, this is a significant restriction, for example, conditions like 'the morphism is in $\mathscr{M}$" in the proof of Theorem 2 can only be expressed using morphisms not in $\mathscr{M}$.

Under certain assumptions, there is also a transformation from $\mathscr{M}$- to $\mathscr{A}$-satisfiability.

Theorem 2 (from $\mathscr{M}$- to $\mathscr{A}$-satisfiability). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ be strictly closed under decomposition. There is a transformation Asat on conditions such that for every condition c and for every morphism p we have $p \models_{\mathscr{A}} \mathrm{Asat}(c) \Leftrightarrow p \models c$, and for every condition c over I and for every object G we have $G \models_{\mathscr{A}} \mathrm{Asat}(c) \Leftrightarrow G \models c$.

$\mathscr{A}$-satisfiable conditions allow elements to be identified. The idea of the construction is to prevent identification by expressing the property 'the morphism is in $\mathscr{M}$' through subconditions.

Construction. For a morphism $a: P \to C$ and a condition c over C, we have that $\mathrm{Asat}(\mathrm{true}) = \mathrm{true}$ and $\mathrm{Asat}(\exists(a, c')) = \exists(a,\ \mathrm{inM}_C \wedge \mathrm{Asat}(c'))$ where $\mathrm{inM}_C = \wedge_{e \in \mathscr{E}} \neg \exists e$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

is a condition over C, the conjunction ranges over all epimorphisms $e: C \to C'$ not in $\mathcal{M}$. For Boolean formulas over conditions, the transformation is extended in the usual way.

Example 9. The condition $c = \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$ meaning for graph morphisms with respect to $\mathcal{M}$-satisfiability 'There exists a proper edge from the image of 1 to the distinct image of 2' is transformed into the condition $\mathrm{Asat}(c) = \exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2, \ \neg\exists(\bigcirc_1 \to \bigcirc_2 \to \bigcirc_{1=2}))$ meaning with respect to $\mathcal{A}$-satisfiability 'There exists an edge from the image of 1 to the image of 2, and the images are distinct'.

Proof. First we prove that for every morphism $q: C \to G$ we have $q \models_{\mathcal{A}} \mathrm{inM}_C$ if and only if q is in $\mathcal{M}$. We prove this by contradiction.

— **Only if:**
Assume $q \models_{\mathcal{A}} \mathrm{inM}_C$, but q not in $\mathcal{M}$. Consider an epi-$\mathcal{M}$-factorisation $q = q' \circ e$ of q with epimorphism e and monomorphism q' in $\mathcal{M}$. Then e is not in $\mathcal{M}$, $q \models_{\mathcal{A}} \exists e$ and $q \not\models_{\mathcal{A}} \mathrm{inM}_C$. Otherwise, by closure of $\mathcal{M}$ under compositions, e and q' in $\mathcal{M}$ would imply q in $\mathcal{M}$, which gives a contradiction.

— **If:**
Assume $q \not\models_{\mathcal{A}} \mathrm{inM}_C$ and q in $\mathcal{M}$. Then $q \models_{\mathcal{A}} \exists e$ for some epimorphism $e: C \to C'$ not in $\mathcal{M}$. Then there is some $q': C' \to G$ such that $q' \circ e = q$. Then q is not in $\mathcal{M}$. Otherwise, by the strict closure of $\mathcal{M}$ under decomposition, q in $\mathcal{M}$ would imply e in $\mathcal{M}$, which gives a contradiction.

$$\exists\Big(\ C \xrightarrow{\ e\ } C' \ \Big) \qquad \begin{array}{ccc} & = & \\ q \searrow & & \swarrow q' \\ & G & \end{array}$$

We show the statement for Asat by structural induction:

— **Basis:**
For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathrm{Asat}(\mathrm{true}) = \mathrm{Asat}(c)$.

— **Hypothesis:**
Assume the statement holds for condition c'.

— **Step:**
For $c = \exists(a, c')$, we have the following.

– **If:**
Let $p \models \exists(a, c')$. Then there is a morphism $q: C \to G$ in $\mathcal{M}$ such that $q \circ a = p$ and $q \models c'$. By the induction hypothesis and the fact that the application condition inM_C is equivalent to 'morphism is in $\mathcal{M}$', $q \models_{\mathcal{A}} \mathrm{inM}_C$ and $q \models_{\mathcal{A}} \mathrm{Asat}(c')$. Consequently, $p \models_{\mathcal{A}} \exists(a, \mathrm{inM}_C \wedge \mathrm{Asat}(c')) = \mathrm{Asat}(\exists(a, c'))$.

– Only if:
Let $p \models_{\mathcal{A}} \mathrm{Asat}(\exists(a, c')) = \exists(a, \mathrm{inM}_C \wedge \mathrm{Asat}(c'))$. Then there is some $q: C \to G$ such that $q \circ a = p$, $q \models_{\mathcal{A}} \mathrm{inM}_C$ and $q \models_{\mathcal{A}} \mathrm{Asat}(c')$. The property of inM_C yields $q \in \mathcal{M}$, and by the induction hypothesis, $q \models c'$. Taking these together, we get $p \models \exists(a, c')$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

For Boolean formulas over conditions, the statement follows from the definitions and the induction hypothesis. This completes the inductive proof.

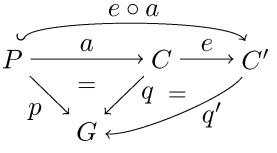
Fact 5 lifts the result to objects and conditions over I. □

Fact 7. The construction above inserts the requirement 'in $\mathscr{M}$' behind every morphism of the condition. However, it is enough to say that newly introduced elements are distinct. An optimised transformation Asat can be defined as follows:

$$\begin{aligned} \mathrm{Asat}(\mathrm{true}) &= \mathrm{true} & \qquad \mathrm{Asat}'(\mathrm{true}) &= \mathrm{true} \\ \mathrm{Asat}(\exists(a,c')) &= \exists(a,\ \mathrm{inM}_C \wedge \mathrm{Asat}'(c')) & \qquad \mathrm{Asat}'(\exists(a,c')) &= \exists(a,\ \mathrm{inM}_a \wedge \mathrm{Asat}'(c')) \end{aligned}$$

where $\mathrm{inM}_a = \wedge_{e \in \mathscr{E}'} \neg \exists e$ is a condition over C and $\mathscr{E}'$ is constructed as follows: let $\mathscr{E}$ be the set of all epimorphisms $e: C \to C'$ not in $\mathscr{M}$ such that $e \circ a$ in $\mathscr{M}$. The set $\mathscr{E}'$ consists of all epimorphisms $e \in \mathscr{E}$ such that there is no (non-isomorphic) decomposition of e into epimorphisms $e'' \circ e' = e$ such that $e' \in \mathscr{E}$ and e'' not in $\mathscr{M}$. For all morphisms p in $\mathscr{M}$ and q with $p = q \circ a$, we get $q \models_{\mathscr{A}} \mathrm{inM}_a$ if and only if q in $\mathscr{M}$. Note that for the initial $\mathscr{M}$-morphism $i_C: I \to C$, we have $\mathrm{inM}_{i_C} = \mathrm{inM}_C$.

Proof. We show for all morphisms p in $\mathscr{M}$ and q with $p = q \circ a$ we have $q \models_{\mathscr{A}} \mathrm{inM}_a$ if and only if q in $\mathscr{M}$. To show a contradiction, let p be in $\mathscr{M}$ and $p = q \circ a$.



— **Only if:**
Assume $q \models_{\mathscr{A}} \mathrm{inM}_a$, but q not in $\mathscr{M}$. As in the proof of Theorem 2, consider an epi-$\mathscr{M}$-factorisation $q = q' \circ e$ of q with epimorphism e and monomorphism q' in $\mathscr{M}$. Then e is not in $\mathscr{M}$. Otherwise, by closure of $\mathscr{M}$ under compositions, e and q' in $\mathscr{M}$ would imply q in $\mathscr{M}$. Moreover, $e \circ a$ in $\mathscr{M}$. Otherwise, by q' in $\mathscr{M}$ and the closure of $\mathscr{M}$ under decompositions, $e \circ a$ not in $\mathscr{M}$ would imply p not in $\mathscr{M}$. Now e is an epimorphism not in $\mathscr{M}$ and $e \circ a$ in $\mathscr{M}$, so $e \in \mathscr{E}$. If there is now a decomposition $e'' \circ e'$ of e with $e' \in \mathscr{E}$ and e'' not in $\mathscr{M}$, let $e = e'$ and repeat this argument. If there is no such decomposition, we have $e \in \mathscr{E}'$, and we conclude $q \models_{\mathscr{A}} \exists e$ and $q \not\models_{\mathscr{A}} \mathrm{inM}_a$, which gives a contradiction.

— **If:**
Assume $q \not\models_{\mathscr{A}} \mathrm{inM}_a$ and q in $\mathscr{M}$. As in the proof of Theorem 2, $q \not\models_{\mathscr{A}} \mathrm{inM}_a$ implies q not in $\mathscr{M}$, which gives a contradiction. □

By Theorems 1 and 2, we obtain the following corollary.

Corollary 1 (equivalence). For weak adhesive HLR categories with $\mathscr{M}$-initial object, epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ strictly closed under decomposition, $\mathscr{A}$-satisfiability and $\mathscr{M}$-satisfiability are expressively equivalent:

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

$$\mathscr{A}\text{-satisfiability} \underset{\text{Asat}}{\overset{\text{Msat}}{\rightleftarrows}} \mathscr{M}\text{-satisfiability}$$

The equivalence result is valid for nested constraints and application conditions; it is not valid for plain constraints in the sense of Ehrig *et al.* (2006b), as the transformations presented increase the depth of nesting. As shown later in Section 8, the transformations Msat and Asat are an important step in the conversion of graph conditions into first-order graph formulas and *vice versa.*

5. Matching of rules

In this section, we investigate the different matching notions for rules and show that $\mathscr{A}$- and $\mathscr{M}$-matching are expressively equivalent. First, there is a transformation from $\mathscr{A}$- to $\mathscr{M}$-matching. We establish a Simulation Theorem saying that any direct derivation with arbitrary matching can be simulated by a direct derivation with $\mathscr{M}$-matching. This extends the Simulation Theorem in Habel *et al.* (2001) to weak adhesive HLR categories with epi-$\mathscr{M}$-factorisations, in which a single rule is simulated by set of so-called quotient rules.

More precisely, we introduce quotient constructions that transform plain rules and conditions with respect to epimorphisms, for example, epimorphism e as depicted below:

$$\begin{array}{ccccc} \text{ac} & & L & \hookleftarrow & K & \hookrightarrow & R \\ & & \downarrow e & (1) & \downarrow & (2) & \downarrow e^* \\ \text{ac}' & & L' & \hookleftarrow & K' & \hookrightarrow & R' \end{array}$$

To support both satisfiability notions of conditions independently of the matching notion, we consider two quotient constructions for conditions, N and B.

If the application condition ac is $\mathscr{M}$-satisfiable, $\text{ac}' = \text{N}(e, \text{ac})$, while $\text{ac}' = \text{B}(e, \text{ac})$ in the case of $\mathscr{A}$-satisfiability.

Lemma 2 (quotients of conditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with epi-$\mathscr{M}$-factorisation. There is a transformation N such that for all conditions c over P and all morphisms p with domain P and epi-$\mathscr{M}$-factorisation $p = p' \circ e$, we have

$$p' \models \text{N}(e, c) \iff p \models c.$$

There is a transformation B such that for all conditions c over P, all morphisms p with domain P and all factorisations $p = p' \circ e$ (e is not required to be an epimorphism), we have

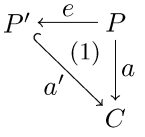
$$p' \models_{\mathscr{A}} \text{B}(e, c) \iff p \models_{\mathscr{A}} c.$$

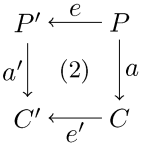
https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

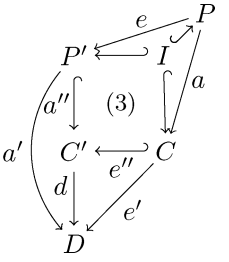
Construction. $N(e, \text{true}) = \text{true}$ and $N(e, \exists(a, c')) = \exists(a', c')$ if e can be extended to an epi-$\mathscr{M}$-factorisation of a, that is, if there exists a morphism a' in $\mathscr{M}$ such that $a = a' \circ e$; otherwise $N(e, \exists(a, c')) = \text{false}$.

If $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts of arbitrary morphisms, then we let $B(e, \text{true}) = \text{true}$ and $B(e, \exists(a, c')) = \exists(a', B(e', c'))$ where (2) is the pushout of the morphisms a and e leading to the morphisms a' and e'.

Alternatively, let $B(e, \text{true}) = \text{true}$ and $B(e, \exists(a, c')) = \vee_{d \in \mathscr{E}} \exists(a', B(e', c'))$ and construct the pushout (3) of the initial $\mathscr{M}$-morphisms $i_{P'}$ and i_C yielding the morphisms a'' and e'' to object C', let the set $\mathscr{E}$ range over all epimorphisms $d: C' \to D$ such that $d \circ e'' \circ a = d \circ a'' \circ e$ and define $a' = d \circ a''$ and $e' = d \circ e''$.







For Boolean formulas over conditions, the transformations are extended in the usual way.

When the existence of pushouts of arbitrary morphisms cannot be guaranteed, transformation B preserves the logical structure and therefore the size of the condition. When the existence of pushouts cannot be guaranteed, transformation B is similar to the alternative construction of transformation Msat of Theorem 1, except that a' is arbitrary and not required to be in $\mathscr{M}$, corresponding just to some epi-factorisation and not necessarily an epi-$\mathscr{M}$-factorisation.

Example 10. The condition $\neg c = \neg\exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$ meaning for graph morphisms with respect to $\mathscr{A}$ [$\mathscr{M}$]-satisfiability 'There does not exist a [proper] edge from the image of 1 to the image of 2' is transformed by the surjective graph morphism $e: \bigcirc_1 \ \bigcirc_2 \to \bigcirc_{1=2}$ into the condition $B(e, \neg c) = \neg B(e, c)) = \neg\exists(\bigcirc_{1=2} \to \bigcirc_{1=2})$ [$N(e, \neg c) = \neg N(e, c) = \neg\text{false} = \text{true}$] meaning 'There does not exist a loop at the common image of node 1 and 2' ['Always satisfied'].

Example 11. The condition $\neg c = \neg\exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_{1=2})$ meaning for graph morphisms with respect to $\mathscr{A}$- and $\mathscr{M}$-satisfiability 'There does not exist a loop at the common image of 1 and 2' is transformed over e into the condition $B(e, \neg c) = \neg\exists(\bigcirc \to \bigcirc) = N(e, \neg c)$ meaning 'There does not exist a loop at the image of the node'.

Proof. The proof is by structural induction.

— **Case 1:**
Transformation N.

– **Basis:**
For $c = \text{true}$, we have $c = \text{true} = N(e, \text{true}) = N(e, c)$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

- **Hypothesis:**
  Assume the statement holds for condition c'.
- **Step:**
  For $c = \exists(a, c')$, we have the following:
  - **Only if:**
    Let $p' \models \mathrm{N}(e, \exists(a, c'))$. As no morphism satisfies false, $\mathrm{N}(e, \exists(a, c')) = \exists(a', c')$ for some morphism a' in $\mathscr{M}$ with epi-$\mathscr{M}$ factorisation $a = a' \circ e$. Then there is some $q : C \to G$ in $\mathscr{M}$ such that $q \circ a' = p'$ and $q \models c'$. Moreover, $q \circ a = p$. Consequently, $p \models \exists(a, c')$.
  - **If:**
    Let $p \models \exists(a, c')$. Then there is some $q : C \to G$ in $\mathscr{M}$ such that $q \circ a = p$ and $q \models c'$. Let $a = a'' \circ e'$ be an epi-$\mathscr{M}$-factorisation of a with epimorphism $e' : P \to P''$ and monomorphism $a'' : P'' \to C$ in $\mathscr{M}$. Then $p'' = q \circ a''$ is in $\mathscr{M}$. Since $a = a'' \circ e'$ is an epi-$\mathscr{M}$-factorisation of a, $p = q \circ a = q \circ a'' \circ e' = p'' \circ e'$ is an epi-$\mathscr{M}$-factorisation of p with epimorphism e' and monomorphism p'' in $\mathscr{M}$. As $p = p' \circ e'$ is also an epi-$\mathscr{M}$-factorisation, the uniqueness of epi-$\mathscr{M}$-factorisations yields $e = e'$, $p' = p''$ and $a' = a''$ (up to isomorphism). Therefore, $p' \models \exists(a', c')$.

— **Case 2:**
Transformation B, $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts of arbitrary morphisms.

- **Basis:**
  For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathrm{B}(e, \mathrm{true}) = \mathrm{B}(e, c)$.
- **Hypothesis:**
  Assume the statement holds for condition c'.
- **Step:**
  For $c = \exists(a, c')$, we have the following:
  - **Only if:**
    Let $p' \models_{\mathscr{A}} \mathrm{B}(e, \exists(a, c')) = \exists(a', \mathrm{B}(e', c'))$. Then there is some $q' : C' \to G$ such that $q' \circ a' = p'$ and $q' \models_{\mathscr{A}} \mathrm{B}(e', c')$. By the induction hypothesis, $q' \circ e' \models_{\mathscr{A}} c'$. Define $q = q' \circ e'$. Then $q \circ a = q' \circ e' \circ a = p' \circ e = p$. Consequently, $p \models_{\mathscr{A}} \exists(a, c')$.
  - **If:**
    Let $p \models_{\mathscr{A}} \exists(a, c')$. Then there is some $q : C \to G$ such that $q \circ a = p$ and $q \models_{\mathscr{A}} c'$. Construct pushout (2). By the universal property of pushouts, there is a morphism $q' : C' \to G$ such that $q' \circ a' = p'$ and $q' \circ e' = q$. By the induction hypothesis, $q' \models_{\mathscr{A}} \mathrm{B}(e', c')$. Consequently, $p' \models_{\mathscr{A}} \exists(a', \mathrm{B}(e', c')) = \mathrm{B}(e, \exists(a, c'))$.

— **Case 3**.
Transformation B, $\langle \mathscr{C}, \mathscr{M} \rangle$ has an $\mathscr{M}$-initial object. The proof is similar to that of Msat in Theorem 1.

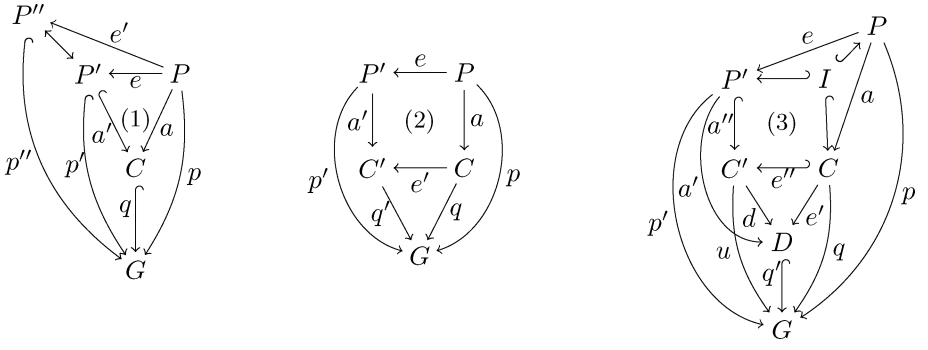
- **Basis:**
  For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathrm{B}(e, \mathrm{true}) = \mathrm{B}(e, c)$.
- **Hypothesis:**
  Assume the statement holds for condition c'.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

– **Step:**
For $c = \exists(a, c')$, we have the following:

- **Only if:**
As in Case 2, by case differentiation for some $d: C' \to D$, using the corresponding morphism $e': C \to D$ and the induction hypothesis.

- **If:**
Let $p \models_{\mathscr{A}} \exists(a, c')$. By the definition of $\mathscr{A}$-satisfiability, there is some $q: C \to G$ such that $q \circ a = p$ and $q \models_{\mathscr{A}} c'$. Since $\langle \mathscr{C}, \mathscr{M} \rangle$ has pushouts along $\mathscr{M}$-morphisms, we can construct the pushout (3) of the $\mathscr{M}$-morphisms $i_C: I \to C$ and $i_{P'}: I \to P'$ leading to morphisms $a'': P' \to C'$ and $e'': C \to C'$. By the universal property of pushouts, there is some $u: C' \to G$ with $u \circ a'' = p'$ and $u \circ e'' = q$. Let $u = q' \circ d$ be an epi-$\mathscr{M}$-factorisation of u with epimorphism d and monomorphism q' in $\mathscr{M}$, and define $a' = d \circ a''$ and $e' = d \circ e''$.
It turns out that $q' \circ a' \circ e = q' \circ e' \circ a$ and, by the monomorphism property of q', $a' \circ e = e' \circ a$. Therefore, d belongs to the set $\mathscr{E}$. As $p' = u \circ a''$, $u = q' \circ d$ and $a' = d \circ a''$, we have $p' = q' \circ a'$ and $p' \models \exists a'$. By the induction hypothesis, $q' \models_{\mathscr{A}} \mathrm{B}(e', c')$. Consequently, $p' \models_{\mathscr{A}} \vee_{d \in \mathscr{E}} \exists(a', \mathrm{B}(e', c')) = \mathrm{B}(e, \exists(a, c'))$.



For Boolean formulas over conditions, the statement follows from the definitions and the induction hypothesis. Consequently, the statement holds for all conditions. □

Remark. In the case of the alternative construction of transformation B, that is, with $\mathrm{B}(e, \mathrm{true}) = \mathrm{true}$ and $\mathrm{B}(e, \exists(a, c')) = \vee_{d \in \mathscr{E}} \exists(a', \mathrm{B}(e', c'))$, the set $\mathscr{E}$ may be restricted to a set $\mathscr{E}'$ with $d \in \mathscr{E}'$, if and only if $d \in \mathscr{E}$ and there exists no $d' \in \mathscr{E}$ and no epimorphism f not in $\mathscr{M}$, such that $d = d' \circ f$ (d is more general than d'). In the case of pushouts and $\mathscr{M}$-initial objects, this optimised transformation eventually yields the only morphism d with the pushout object of (2) as its codomain.

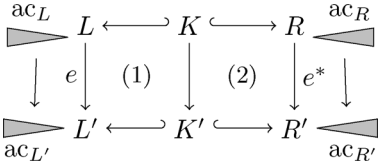
We are now able to present a transformation from $\mathscr{A}$- to $\mathscr{M}$-matching. For simplicity, we assume a fixed satisfiability notion. For a rule ρ, let $[\![\rho]\!]_{\mathscr{A}}$ and $[\![\rho]\!]_{\mathscr{M}}$ denote the sets of pairs $\langle G, H \rangle$ with direct derivation $G \Rightarrow_{\rho, m, m^*} H$ through a match m in $\mathscr{A}$ and $\mathscr{M}$, respectively. The definition is extended to sets of rules $\mathscr{R}$, that is, $[\![\mathscr{R}]\!]_{\mathscr{A}} = \bigcup_{\rho \in \mathscr{R}} [\![\rho]\!]_{\mathscr{A}}$ and $[\![\mathscr{R}]\!]_{\mathscr{M}} = \bigcup_{\rho \in \mathscr{R}} [\![\rho]\!]_{\mathscr{M}}$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Theorem 3 (from $\mathscr{A}$- to $\mathscr{M}$-matching). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with epi-$\mathscr{M}$-factorisation, and, in the case of $\mathscr{A}$-satisfiability, pushouts or an $\mathscr{M}$-initial object. There is a transformation Q such that for every rule ρ, we have

$$[\![\rho]\!]_{\mathscr{A}} = [\![\mathrm{Q}(\rho)]\!]_{\mathscr{M}}.$$

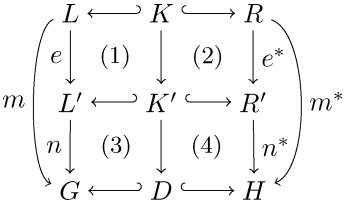
Construction. For a rule $\rho = \langle q, \mathrm{ac} \rangle$ with $\mathrm{ac} = \langle \mathrm{ac}_L, \mathrm{ac}_R \rangle$, let $\mathrm{Q}(\rho)$ be the set of rules $\rho' = \langle q', \mathrm{ac}' \rangle$ where q' is a *quotient rule* of q with respect to an epimorphism e, that is, there are two pushouts (1) and (2) with match e and comatch e^* as below, and $\mathrm{ac}' = \langle \mathrm{N}(e, \mathrm{ac}_L), \mathrm{N}(e^*, \mathrm{ac}_R) \rangle$ for $\mathscr{M}$-satisfiability and $\langle \mathrm{B}(e, \mathrm{ac}_L), \mathrm{B}(e^*, \mathrm{ac}_R) \rangle$ otherwise.



Example 12. Consider the rule $\langle \bigcirc_1\ \bigcirc_2 \hookleftarrow \bigcirc_1\ \bigcirc_2 \hookrightarrow \bigcirc_1 \to \bigcirc_2 \rangle$ with left application condition $\mathrm{ac} = \neg\exists(\bigcirc_1\ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$ meaning with respect to $\mathscr{A}$-matching and $\mathscr{A}$-satisfiability 'Add an edge, provided there does not exist one'. Then the rule $\langle \bigcirc \hookleftarrow \bigcirc \hookrightarrow \bigcirc\!\!\circlearrowright \rangle$ with left application condition $\mathrm{B}(\bigcirc\ \bigcirc \to \bigcirc, \mathrm{ac}) = \neg\exists(\bigcirc \to \bigcirc\!\!\circlearrowright)$ (compare with Example 10) is a quotient rule meaning with respect to $\mathscr{M}$-matching 'Add a loop at the node, provided there does not exist one'.

Proof.

— **If:**
Let $G \Rightarrow_{\rho', n, n^*} H$ be a direct derivation through $\rho' = \langle q', \mathrm{ac}' \rangle$ with ac' as in the construction and $n \in \mathscr{M}$. Then the diagrams (1), (2), (3) and (4) in the figure below are pushouts and, by the Composition Lemma of pushouts, the composed diagrams (1)+(3) and (2)+(4) are pushouts as well. Hence, there is a direct derivation $G \Rightarrow_{q, m, m^*} H$ with $m \in \mathscr{A}$. By assumption, $n \models \mathrm{ac}'_L$ and $n^* \models \mathrm{ac}'_R$ ($\mathscr{M}$-satisfiability) or $n \models_{\mathscr{A}} \mathrm{ac}'_L$ and $n^* \models_{\mathscr{A}} \mathrm{ac}'_R$ ($\mathscr{A}$-satisfiability). By Lemma 2, $m \models \mathrm{ac}_L$ and $m^* \models \mathrm{ac}_R$. Thus, $G \Rightarrow_{\rho, m, m^*} H$ with $m \in \mathscr{A}$.



— **Only if:**
Let $G \Rightarrow_{\rho, m, m^*} H$ be a direct derivation through $\rho = \langle q, \mathrm{ac} \rangle$ with $m \in \mathscr{A}$, and let $m = n \circ e$ be an epi-$\mathscr{M}$ factorisation of m with epimorphism $e: L \to L'$ and monomorphism $n: L' \to G$ in $\mathscr{M}$. Then there is a decomposition of the original diagrams into diagrams (1) and (3), (2) and (4) as follows. Construct the object K' as a pullback object of

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

$L' \to G \leftarrow D$ and denote the diagram by (3). By the universal property of pullbacks, there is a unique morphism $e' : K \to K'$ such that $K \to K' \to D = K \to D$ and diagram (1) commutes. By the pushout–pullback decomposition, (1) and (3) are pushouts. Now construct the object R' as the pushout object of $K' \leftarrow K \to R$ and denote the diagram by (2). By the universal property of pushouts, there is a unique morphism $n^* : R' \to H$ such that $R \to R' \to H = R \to H$ and diagram (4) commutes. By the Decomposition Lemma for pushouts, diagram (4) is a pushout. Since epimorphisms and $\mathscr{M}$-morphisms are closed under pullbacks and pushouts, the vertical morphisms e, e' and e^* are epimorphisms and n, n' and n^* are in $\mathscr{M}$. Let ac' be as in the construction. Then $\rho' = \langle q', \mathrm{ac}' \rangle$ is a rule in $\mathrm{Q}(p)$ and, by Lemma 2, $G \Rightarrow_{\rho', n, n^*} H$ is a direct derivation with $n \in \mathscr{M}$. □

If $\mathscr{M}$ is strictly closed under decomposition, there is a transformation from $\mathscr{M}$- to $\mathscr{A}$-matching.

Theorem 4 (from $\mathscr{M}$- to $\mathscr{A}$-matching). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category and $\mathscr{M}$ be strictly closed under decomposition. There is a transformation R such that, for every rule ρ,

$$[\![\rho]\!]_{\mathscr{M}} = [\![\mathrm{R}(\rho)]\!]_{\mathscr{A}}.$$

The idea of the transformation is to use the application condition inM, which is satisfied if and only if the match is in $\mathscr{M}$.

Construction. For a rule $\rho = \langle q, \mathrm{ac} \rangle$, let $\mathrm{R}(\rho) = \langle q, \langle \mathrm{inM}_L \wedge \mathrm{ac}_L, \mathrm{ac}_R \rangle \rangle$.

Example 13. The rule $q = \langle \bigcirc_1 \ \bigcirc_2 \leftarrow \bigcirc_1 \ \bigcirc_2 \hookrightarrow \bigcirc_1 \to \bigcirc_2 \rangle$ with left application condition $\mathrm{ac}_L = \neg\exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$, meaning with respect to $\mathscr{M}$-matching 'Add an edge between distinct nodes, provided there does not exist a connecting edge', is transformed into the rule q with left application condition $\mathrm{inM}_L \wedge \mathrm{ac}_L = \neg\exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_{1=2}) \wedge \neg\exists(\bigcirc_1 \ \bigcirc_2 \to \bigcirc_1 \to \bigcirc_2)$ meaning with respect to $\mathscr{A}$-matching 'Add an edge between the nodes, provided the nodes are distinct and there does not exist a connecting edge'.

Proof. Since $\mathscr{M}$ is strictly closed under decomposition, the property 'the match is in $\mathscr{M}$' can be expressed by the application condition inM (see the first statement in the proof of Theorem 2). Then, for every rule ρ, we have $G \Rightarrow_{\mathscr{M},\rho} H$ if and only if $G \Rightarrow_q H$ for some $q \in \mathrm{R}(\rho)$, that is, $[\![\rho]\!]_{\mathscr{M}} = [\![\mathrm{R}(\rho)]\!]_{\mathscr{A}}$. □

By Theorems 3 and 4, we obtain the following corollary.

Corollary 2 (equivalence). For weak adhesive HLR categories with pushouts or an $\mathscr{M}$-initial object, epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ strictly closed under decomposition, $\mathscr{A}$-matching and $\mathscr{M}$-matching are expressively equivalent:

$$\mathscr{A}\text{-matching} \underset{\mathrm{R}}{\overset{\mathrm{Q}}{\rightleftarrows}} \mathscr{M}\text{-matching}$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

6. Transformations of conditions

In this section we present transformations from constraints to application conditions, from right to left application conditions, and from application conditions to constraints. Moreover, we consider the construction of explicit gluing conditions. These basic transformations can be used in the development of correct transformation systems, as we will show in Section 7.

First, we show that a constraint can be transformed into an equivalent right application condition in the sense that the existence of a comatch is conjunctively incorporated into the constraint. This transformation is used in Section 7, for example, in the construction of weakest preconditions. The construction is a generalised version of the corresponding construction for basic constraints in Ehrig *et al.* (2006b), which was first described in Heckel and Wagner (1995).

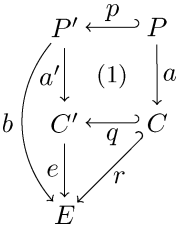
Theorem 5 (transformation of constraints into application conditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object I and epi-$\mathscr{M}$-factorisation. There is a transformation A such that, for all conditions c over I, all rules ρ with right-hand side R, and all morphisms $m^*: R \to H$, we have

$$m^* \models \mathrm{A}(\rho, c) \iff H \models c.$$

The idea of the transformation A is to consider a disjunction of all possible overlappings of R and the objects of the condition.

Construction. For conditions c over I and rules ρ, define $\mathrm{A}(\rho, c) = \mathrm{A}_1(i_R, c)$, where i_R is the initial $\mathscr{M}$-morphism to R. For morphisms $p: P \to P'$ in $\mathscr{M}$, conditions over P and switch $k \in \{1, 2\}$,

$$\begin{aligned} \mathrm{A}_k(p, \text{true}) &= \text{true}, \\ \mathrm{A}_k(p, \exists(a, c')) &= \vee_{e \in \mathscr{E}} \exists(b, \mathrm{A}_2(r, c')). \end{aligned}$$



For a morphism $p: P \to P'$ in $\mathscr{M}$, construct the pushout (1) of p and a leading to morphisms $a': P' \to C'$ and $q: C \to C'$. The disjunction $\vee_{e \in \mathscr{E}}$ ranges over all epimorphisms $e: C' \to E$ such that both $b = e \circ a'$ and $r = e \circ q$ are in $\mathscr{M}$ if $k = 2$, and just $r = e \circ q$ in $\mathscr{M}$ if $k = 1$.

For Boolean formulas over conditions, the transformation $\mathrm{A}_k(p, c)$ is extended in the usual way.

Example 14. Given the rule $\rho = \langle \bigcirc\ \bigcirc \hookleftarrow \bigcirc\ \bigcirc \hookrightarrow \bigcirc\!\!\to\!\!\bigcirc \rangle$, the graph constraint $c = \forall(\varnothing \to \bigcirc,\ \exists(\bigcirc \to \circlearrowright))$ meaning 'Every node must have a loop' is transformed into

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

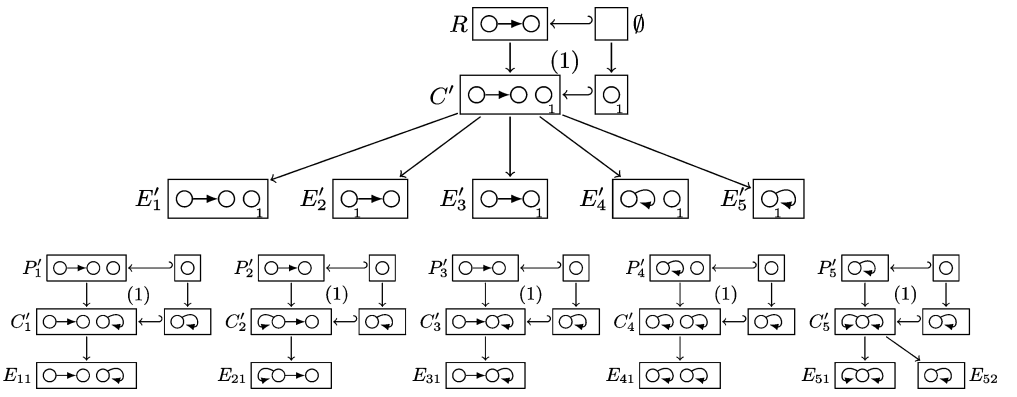


Figure 3. Transformation of a graph constraint into an application condition

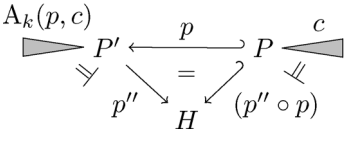
the right application condition $\mathrm{A}(\rho, c) = \neg\vee_{j=1}^{5} \exists(\bigcirc\!\!\rightarrow\!\!\bigcirc \to P'_j,\ \neg\vee_{\ell} \exists(P'_j \to E_{j\ell})) = \wedge_{j=1}^{5} \forall(\bigcirc\!\!\rightarrow\!\!\bigcirc \to P'_j,\ \vee_{\ell} \exists(P'_j \to E_{j\ell}))$ for the graphs $P'_j, E_{j\ell}$ as in Figure 3, meaning 'Every node (outside and inside the occurrence of the right-hand side) must have a loop'.

Remark. For the initial $\mathscr{M}$-morphisms i_R, i_P, the construction of $\mathrm{A}_1(i_R, \exists i_P)$ and the first invocation of $\mathrm{A}_1(i_R, \exists(i_P, c))$ corresponds to the construction of all possible gluings $\langle E, b, r\rangle$ of the right-hand side R of the rule and the object P of the condition such that the morphism r is in $\mathscr{M}$, as in Ehrig *et al.* (2006b).

Before we prove Theorem 5, we will state a general property of the transformation $\mathrm{A}_k(p, c)$. The following lemma states that a condition c over P can be shifted over a morphism p in $\mathscr{M}$ with $\mathrm{Dom}(p) = P$ in the sense of a conjunctive combination of c and $\exists p$, as used in Pennemann (2008b) for instance.

Lemma 3 (transformation $\mathrm{A}_k(p, c)$). For all conditions c over P, all morphisms $p: P \to P'$ in $\mathscr{M}$ and $p'': P' \to H$ and either ($k = 1$ and p $\mathscr{M}$-initial) or (k=2 and p'' in $\mathscr{M}$), we have

$$p'' \models \mathrm{A}_k(p, c) \iff p'' \circ p \models c.$$



Proof. The proof is by structural induction.

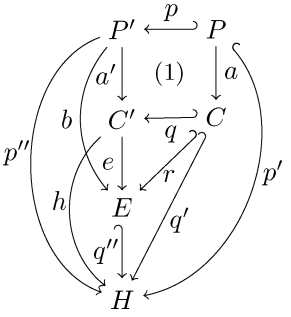
— **Basis:**
For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathrm{A}_k(p, \mathrm{true}) = \mathrm{A}_k(p, c)$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

— **Hypothesis:**
Assume the statement holds for condition c'.

— **Step:**
Let $c = \exists(a, c')$.

– **Only if:**
Assume $p'' \models \mathrm{A}_k(p, \exists(a, c')) = \vee_{e \in \mathscr{E}} \exists(b, \mathrm{A}_2(r, c'))$. There is an $e \in \mathscr{E}$ such that $p'' \models \exists(b, \mathrm{A}_2(r, c'))$. By the definition of $\models$, there exists a morphism $q'' : E \to H$ in $\mathscr{M}$ with $p'' = q'' \circ b$. Define $q' = q'' \circ r$. As q'', r in $\mathscr{M}$ and $\mathscr{M}$ is closed under composition, q' in $\mathscr{M}$. Let $p' = p'' \circ p$. Either $p = i_R$ and $p' = i_H$ in $\mathscr{M}$ $(k = 1)$ or p, p'' in $\mathscr{M}$, $\mathscr{M}$ closed under composition and p' in $\mathscr{M}$ $(k = 2)$. By construction, $a' \circ p = q \circ a$ ((1) is pushout), $r = e \circ q$ and $b = e \circ a'$. Taking these together, $p'' \circ p = p' = q' \circ a$ $(p' = p'' \circ p \models \exists a)$. Using the induction hypothesis, $q'' \models \mathrm{A}_2(p, \exists(a, c'))$ implies $q' = q'' \circ r \models c'$, so we have $p' = p'' \circ p \models \exists(a, c')$.



– **If:**
Assume $p'' \circ p \models \exists(a, c')$. Let $p' = p'' \circ p$. Either $p = i_R$ and $p' = i_H$ in $\mathscr{M}$ $(k = 1)$ or p'', p in $\mathscr{M}$, $\mathscr{M}$ closed under composition and p' in $\mathscr{M}$ $(k = 2)$. By the definition of $\models$, there exists a morphism $q' : C \to H$ in $\mathscr{M}$ with $p' = q' \circ a$. Following the construction, we yield the pushout (1) with object C' together with the morphisms a' and q. As $q' \circ a = p' = p'' \circ p$, the pushout guarantees the existence of a unique morphism $h : C' \to H$ with $p'' = h \circ a'$ and $q' = h \circ q$. Consider $q'' \circ e = h$, which is an epi-$\mathscr{M}$-factorisation of h with epimorphism e and $\mathscr{M}$-morphism q''. Let $r = e \circ q$. As $q'' \circ r = q'$, q', q'' in $\mathscr{M}$ and $\mathscr{M}$ closed under decomposition, r in $\mathscr{M}$. Define $b = e \circ a'$. For $k = 1$, we now have the result. For $k = 2$, the additional assumption p'' in $\mathscr{M}$ and p'', q'' in $\mathscr{M}$ and $\mathscr{M}$ closed under decomposition yields the additional requirement b in $\mathscr{M}$. In every case, $p'' = h \circ a'$, $h = q'' \circ e$ and $b = e \circ a'$ yield $p'' = q'' \circ b$ $(p'' \models \vee_{e \in \mathscr{E}} \exists b = \mathrm{A}_k(p, \exists a))$. Using the induction hypothesis, $q' = q'' \circ r \models c'$ implies $q'' \models \mathrm{A}_2(p, \exists(a, c'))$, so we have $p'' \models \vee_{e \in \mathscr{E}} \exists(b, \mathrm{A}_2(r, c')) = \mathrm{A}_k(p, \exists(a, c'))$.

For Boolean formulas over conditions, the statement follows directly from the definitions and the induction hypothesis. Thus, the statement holds for all conditions over I. □

Proof of Theorem 5. According to Lemma 3, for all conditions c over I, all morphisms $i_R : I \to R$ in $\mathscr{M}$ and $m^* : R \to H$, we have: $m^* \models \mathrm{A}_1(i_R, c)$ if and only if $m^* \circ i_R \models c$ if and only if $i_H \models c$ if and only if $H \models c$. □

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

In the modified cases of $\mathscr{A}$-matching and $\mathscr{A}$-satisfiability or $\mathscr{M}$-matching and $\mathscr{M}$-satisfiability, there are simplified transformations.

Remark ($\mathscr{M}$-matching and $\mathscr{M}$-satisfiability). In the case of $\mathscr{M}$-matching and $\mathscr{M}$-satisfiability, one may directly use transformation $\mathrm{A}_2(i_R, c)$ such that for all conditions c over I and all rules $\rho = \langle L \hookleftarrow K \hookrightarrow R \rangle$ we have that for all morphisms $m^*: R \to H$ in $\mathscr{M}$, $m^* \models \mathrm{A}_2(i_R, c) \Leftrightarrow H \models c$.

According to Lemma 3, for all conditions c over I, all morphisms $i_R: I \to R$ in $\mathscr{M}$ and $m^*: R \to H$ in $\mathscr{M}$, we have that $m^* \models \mathrm{A}_2(i_R, c)$ if and only if $m^* \circ i_R \models c$ if and only if $i_H \models c$ if and only if $H \models c$.

Remark ($\mathscr{A}$-matching and $\mathscr{A}$-satisfiability). In the case of $\mathscr{A}$-matching and $\mathscr{A}$-satisfiability, one may use transformation B of Lemma 2 such that for all conditions c over I, all rules $\rho = \langle L \hookleftarrow K \hookrightarrow R \rangle$ we have that for all morphisms $m^*: R \to H$, $m^* \models_{\mathscr{A}} \mathrm{B}(i_R, c) \Leftrightarrow H \models_{\mathscr{A}} c$.

For all conditions c over I, all initial $\mathscr{M}$-morphisms i_H and all factorisations $i_H = m^* \circ i_R$ of morphism i_H, we have $m^* \models_{\mathscr{A}} \mathrm{B}(i_R, c) \Leftrightarrow i_H \models_{\mathscr{A}} c \Leftrightarrow H \models_{\mathscr{A}} c$.

With the help of transformation $\mathrm{N}(e, c)$ of Lemma 2, we can generalise transformation $\mathrm{A}_k(p, c)$ from morphisms p in $\mathscr{M}$ to arbitrary morphisms.

Corollary 3 (shifting of conditions over morphisms). There is a transformation Shift_k such that for all conditions c over P and all morphisms $p: P \to P'$, $p'': P' \to H$, and either (k=1 and p $\mathscr{M}$-initial) or (k=2 and p'' in $\mathscr{M}$), we have

$$p'' \models \mathrm{Shift}_k(p, c) \iff p'' \circ p \models c.$$

Construction. Let $p = m \circ e$ be an epi-$\mathscr{M}$-factorisation of m into an epimorphism e and morphism m in $\mathscr{M}$. If k=1 and p $\mathscr{M}$-initial, then $e = \mathrm{id}_I$, p in $\mathscr{M}$ and we let $\mathrm{Shift}_1(m, c) = \mathrm{A}_1(p, c)$. If k=2 and p'' in $\mathscr{M}$, then $\mathrm{Shift}_2(p, c) = \mathrm{A}_2(m, \mathrm{N}(e, c))$.

Proof. For k=2 and p'' in $\mathscr{M}$, the proof obligation is a direct consequence of the transformations N and A_k (Lemma 2 and 3), we have $p'' \models \mathrm{Shift}_k(p, c) = \mathrm{A}_2(m, \mathrm{N}(e, c))$ if and only if $(p'' \circ m) \models \mathrm{N}(e, c)$ if and only if $(p'' \circ m) \circ e \models c$ if and only if $(p'' \circ p) \models c$, as $(p'' \circ m)$ is in $\mathscr{M}$ and $(p'' \circ m) \circ e$ is an epi-$\mathscr{M}$-factorisation of $(p'' \circ p)$. □

We now show that there is a transformation from right to left application conditions such that a comatch satisfies an application condition if and only if the match satisfies the transformed application condition. This transformation is used in Section 7, for example, in the construction of weakest preconditions. The construction is a generalised version of the corresponding construction for basic application conditions in Ehrig *et al.* (2006b), which was first described in Heckel and Wagner (1995).

Theorem 6 (transformation of application conditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object. There is a transformation L such that, for every rule ρ, every right application condition c for ρ, and every direct derivation $G \Rightarrow_{\rho, m, m^*} H$, we have

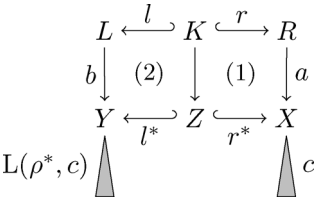
$$m \models \mathrm{L}(\rho, c) \iff m^* \models c.$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

The idea of transformation L is to apply ρ and subsequent 'derived' rules on the objects and morphisms of the condition, if possible, while preserving the (logical) structure of the condition.

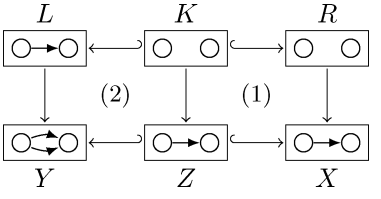
Construction. Transformation $\mathrm{L}(\rho, c)$ is defined inductively as follows:

Let $\mathrm{L}(\rho, \mathrm{true}) = \mathrm{true}$ and $\mathrm{L}(\rho, \exists(a, c')) = \exists(b, \mathrm{L}(\rho^*, c'))$ if $\langle r, a\rangle$ has a pushout complement (1) and $\rho^* = \langle Y \hookleftarrow Z \hookrightarrow X\rangle$ denotes the 'derived' rule by constructing pushout (2). Note, as r is in $\mathscr{M}$, the pushout complement is unique, if it exists. Otherwise, $\langle r, a\rangle$ has no pushout complement and $\mathrm{L}(\rho, \exists(a, c')) = \mathrm{false}$.



For Boolean formulas over application conditions, the transformation L is extended in the usual way.

Example 15. The right application condition $\mathrm{ac} = \neg\exists(\bigcirc\ \ \bigcirc \to \bigcirc\!\!\to\!\!\bigcirc)$ for the rule $\rho = \langle \bigcirc\!\!\to\!\!\bigcirc \hookleftarrow \bigcirc\ \ \bigcirc \hookrightarrow \bigcirc\ \ \bigcirc\rangle$, meaning 'An edge between the nodes in the comatch must not exist', is transformed into the left application condition $\mathrm{L}(\rho, \mathrm{ac}) = \neg\exists(\bigcirc\!\!\to\!\!\bigcirc \to \bigcirc\rightleftarrows\bigcirc)$, meaning that two parallel edges between the nodes in the match must not exist.



Example 16. The right application condition $\mathrm{ac} = \neg\exists(\bigcirc\!\!\to\!\!\bigcirc \to \bigcirc\rightleftarrows\bigcirc)$, meaning 'There do not exist two parallel edges between the nodes in the comatch', is transformed for the rule $\rho = \langle \bigcirc_1 \hookleftarrow \bigcirc_1 \hookrightarrow \bigcirc_1\!\!\to\!\!\bigcirc\rangle$ into the left application condition $\mathrm{L}(\rho, \mathrm{ac}) = \neg\mathrm{false} \equiv \mathrm{true}$, meaning 'The rule can always be applied', because the pair $\langle \bigcirc_1 \to \bigcirc_1\!\!\to\!\!\bigcirc_2,\ \bigcirc_1\!\!\to\!\!\bigcirc_2 \to \bigcirc_1\rightleftarrows\bigcirc_2\rangle$ of morphisms has no pushout complement.

Proof. The proof is by structural induction. Let $G \Rightarrow_{\rho,m,m^*} H$ be any direct derivation.

— **Basis:**
For $c = \mathrm{true}$, we have $c = \mathrm{true} = \mathrm{L}(\rho, \mathrm{true}) = \mathrm{L}(\rho, c)$.

— **Hypothesis:**
Assume the statement holds for condition c'.

— **Step:**
For right application conditions of the form $c = \exists(a, c')$, we distinguish two cases:

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

– **Case 1:** The pair $\langle r, a\rangle$ has a pushout complement. So we have $\mathrm{L}(\rho, \exists(a, c')) = \exists(b, \mathrm{L}(\rho^*, c'))$ and:

(A) Given a morphism $q' : Y \to G$ in $\mathscr{M}$ with $q' \circ b = m$, there is a decomposition of the pushouts of the derivation into pushouts (1′)–(4′) as follows:

- First, construct (4′) as the pullback diagram of q' and d_2. By the universal property of pullbacks, there exists a unique morphism $K \to Z'$ such that the arising diagrams commute. By the pushout–pullback decomposition, (2′) and (4′) are pushouts. As $K \to Z'$ is unique, l is in $\mathscr{M}$, and pushout complements of morphisms in $\mathscr{M}$ are unique up to isomorphisms, pushout (2′) equals pushout (2) from the construction up to isomorphism.
- Now construct pushout (1′) of $K \to Z'$ and r. As $K \to Z'$ is unique, r is in $\mathscr{M}$, and pushout complements of morphisms in $\mathscr{M}$ are unique up to isomorphisms, pushout (1′) equals pushout (1) from the construction up to isomorphism. By the universal property of pushouts, there is a unique morphism $q : X \to H$ with $q \circ a = m^*$. By the Decomposition Lemma of pushouts, the diagram (3′) arising becomes a pushout. Since q' is in $\mathscr{M}$, the morphisms z and q are in $\mathscr{M}$, too.

$$
\begin{array}{ccccc}
L & \xleftarrow{l} & K & \xrightarrow{r} & R \\
{\scriptstyle m}\downarrow{\scriptstyle b} & (2') & \downarrow & (1') & \downarrow{\scriptstyle a}{\scriptstyle m^*} \\
Y & \longleftarrow & Z' & \longrightarrow & X \\
{\scriptstyle q'}\downarrow & (4) & \downarrow{\scriptstyle z} & (3) & \downarrow{\scriptstyle q} \\
G & \xleftarrow[d_2]{} & D & \xrightarrow[d_1]{} & H
\end{array}
$$

(B) Given a morphism $q : X \to H$ in $\mathscr{M}$ with $q \circ a = m^*$, the original pushouts can be decomposed into pushouts (1′)–(4′) similar to the above. First (3′) is constructed as a pullback. Eventually a morphism $q' : Y \to G$ with $q' \circ b = m$ is produced. Since q is in $\mathscr{M}$, z and q' are in $\mathscr{M}$ also.

(C) Given a right application condition c' and the derived rule $\rho^* = \langle Y \hookleftarrow Z \hookrightarrow X\rangle$ with morphisms $q' : Y \to G$ and $q : X \to H$, by the induction hypothesis, we have $q \models c'$ if and only if $q' \models \mathrm{L}(\rho^*, c')$.
By the definitions of L and $\models$ and the statements above, we have

$$
\begin{aligned}
m \models \mathrm{L}(\rho, \exists(a, c')) = \exists(b, \mathrm{L}(\rho^*, c')) & \\
&\Leftrightarrow \exists q' : Y \to G \text{ in } \mathscr{M}.\ q' \circ b = m \wedge q' \models \mathrm{L}(\rho^*, c') \\
&\Leftrightarrow \exists q : X \to H \text{ in } \mathscr{M}.\ q \circ a = m^* \wedge q \models c' \\
&\Leftrightarrow m^* \models \exists(a, c').
\end{aligned}
$$

– **Case 2:**
The pair $\langle r, a\rangle$ has no pushout complement and $\mathrm{L}(\rho, \exists(a, c')) = \text{false}$. We have to show that $m \models \text{false} \Leftrightarrow m^* \models \exists(a, c')$. As no morphism satisfies false, it is enough to show $m^* \not\models \exists(a, c')$. Assume $m^* \models \exists(a, c')$. Then there exists some $q : X \to H$ with $q \in \mathscr{M}$ and $q \circ a = m^*$. Thus there is a decomposition of the existing pushout into

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

two pushouts (1) and (3) as in Case 1 above. Hence the pair $\langle r, a\rangle$ has a pushout complement. This gives a contradiction.

For Boolean formulas over right application conditions, the statement follows directly from the definition and the induction hypothesis. Thus, the statement holds for all right application conditions. □

In the case of $\mathscr{A}$-satisfiability, there is a derived transformation from right to left application conditions.

Corollary 4 ($\mathscr{A}$-satisfiability). Let $\langle \mathscr{C}, \mathscr{M}\rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object, epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ strictly closed under decomposition. Then there is a derived transformation $\mathrm{L}_{\mathscr{A}}$ with $\mathrm{L}_{\mathscr{A}}(\rho, \mathrm{ac}) = \mathrm{Asat}(\mathrm{L}(\rho, \mathrm{Msat}(\mathrm{ac})))$ for $\mathscr{A}$-satisfiable conditions ac such that for every derivation $G \Rightarrow_{\rho,m,m^*} H$,

$$m \models_{\mathscr{A}} \mathrm{L}_{\mathscr{A}}(\rho, \mathrm{ac}) \iff m^* \models_{\mathscr{A}} \mathrm{ac}.$$

Proof. The statement follows immediately from Theorems 1, 2, and 6:

$$\begin{aligned} m \models_{\mathscr{A}} \mathrm{L}_{\mathscr{A}}(\rho, \mathrm{ac}) &\Leftrightarrow m \models_{\mathscr{A}} \mathrm{Asat}(\mathrm{L}(\rho, \mathrm{Msat}(\mathrm{ac}))) \\ &\Leftrightarrow m \models \mathrm{L}(\rho, \mathrm{Msat}(\mathrm{ac})) \Leftrightarrow m^* \models \mathrm{Msat}(\mathrm{ac}) \\ &\Leftrightarrow m^* \models_{\mathscr{A}} \mathrm{ac}. \end{aligned}$$

□

For weak adhesive HLR categories with $\mathscr{M}$-initial object, there are two transformations of application conditions to constraints, which correspond to the universal and existential closure of application conditions. In the case of $\mathscr{A}$-matching however, the closures have to be over all morphisms and do not fit in with the notion of $\mathscr{M}$-satisfiability. Therefore the first part of the condition has to be transformed accordingly. Intuitively, these transformations construct weakest preconditions and strongest post conditions of the matching operation and are used in Section 7, for example, in the construction of weakest preconditions and strongest postconditions.

Theorem 7 (transformation of application conditions into constraints). Let $\langle \mathscr{C}, \mathscr{M}\rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object. There are transformations $\mathrm{C}_\forall$ and $\mathrm{C}_\exists$ such that, for every application condition ac over L and for all objects G,

$$G \models \mathrm{C}_\forall(\mathrm{ac}) \Leftrightarrow \forall m\colon L \to G.\ m \models \mathrm{ac}$$

and

$$G \models \mathrm{C}_\exists(\mathrm{ac}) \Leftrightarrow \exists m\colon L \to G.\ m \models \mathrm{ac}.$$

The idea of the transformations is to consider junctions of $\mathscr{M}$-satisfiable conditions, one condition for each possible level of identification.

Construction. For an application condition ac over L, let

$$\begin{aligned} \mathrm{C}_\forall(\mathrm{ac}) &= \wedge_{e \in \mathscr{E}}\, \forall(i_{L'},\ \mathrm{N}(e, \mathrm{ac})) \\ \mathrm{C}_\exists(\mathrm{ac}) &= \vee_{e \in \mathscr{E}}\, \exists(i_{L'},\ \mathrm{N}(e, \mathrm{ac})) \end{aligned}$$

where the set $\mathscr{E}$ ranges over all epimorphisms e starting from L, and, for an epimorphism $e\colon L \to L'$, $i_{L'}$ is the initial $\mathscr{M}$-morphism to L', and the transformation $\mathrm{N}(e, _)$ is defined as

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

in the construction of Lemma 2. For Boolean formulas over application conditions, the transformations $C_\forall$ and $C_\exists$ are extended in the usual way.

Proof. By Lemma 2, for all application conditions ac over L and all morphisms m with domain L with epi-$\mathscr{M}$-factorisation $m = m' \circ e$, $m' \models \mathrm{N}(e, \mathrm{ac}) \Leftrightarrow m \models \mathrm{ac}$. Taking this together with the definitions of $\models$, $\wedge$ and $C_\forall$, we have

$$\begin{aligned}
\forall m: L \to G.\ m \models \mathrm{ac} &\Leftrightarrow \forall e: L \to L' \text{ in } \mathscr{E}.\ \forall m': L' \to G \text{ in } \mathscr{M}.\ m' \models \mathrm{N}(e, \mathrm{ac}) \\
&\Leftrightarrow \forall e: L \to L' \text{ in } \mathscr{E}.\ i_G \models \forall(i_{L'},\ \mathrm{N}(e, \mathrm{ac})) \\
&\Leftrightarrow \forall e: L \to L' \text{ in } \mathscr{E}.\ G \models \forall(i_{L'},\ \mathrm{N}(e, \mathrm{ac})) \\
&\Leftrightarrow G \models \wedge_{e \in \mathscr{E}}\, \forall(i_{L'},\ \mathrm{N}(e, \mathrm{ac})) \\
&\Leftrightarrow G \models C_\forall(\mathrm{ac}).
\end{aligned}$$

The statement for $C_\exists$ then follows from the relationship $C_\exists(\mathrm{ac}) \equiv \neg C_\forall(\neg \mathrm{ac})$. □

Example 17. The application condition $\mathrm{ac} = \exists(\bigcirc_1 \to \bigcirc_1 \to \bigcirc)$ meaning 'For the image of node 1, there exists some proper outgoing edge' is transformed into the constraints

$$\begin{aligned}
C_\forall(\mathrm{ac}) &= \forall(\varnothing \to \bigcirc,\ \mathrm{N}(\mathrm{id}, \mathrm{ac})) = \forall(\varnothing \to \bigcirc,\ \mathrm{ac}) = \forall(\varnothing \to \bigcirc_1,\ \exists(\bigcirc_1 \to \bigcirc_1 \to \bigcirc)) \\
C_\exists(\mathrm{ac}) &= \exists(\varnothing \to \bigcirc,\ \mathrm{N}(\mathrm{id}, \mathrm{ac})) = \exists(\varnothing \to \bigcirc,\ \mathrm{ac}) = \exists(\varnothing \to \bigcirc_1,\ \exists(\bigcirc_1 \to \bigcirc_1 \to \bigcirc))
\end{aligned}$$

meaning 'For every node, there exists some proper outgoing edge' and 'There exists a node with some proper outgoing edge', respectively.

Example 18. The application condition $\mathrm{ac} = \exists(\bigcirc\ \bigcirc \to \bigcirc)$ meaning 'The two nodes have to be identified' is transformed into the constraints

$$\begin{aligned}
C_\forall(\mathrm{ac}) &= \wedge_{e \in \mathscr{E}}\, \forall(e \circ i_L,\ \mathrm{N}(e, \mathrm{ac})) = \forall(\mathrm{id} \circ i_L,\ \mathrm{N}(\mathrm{id}, \mathrm{ac})) \wedge \forall(f \circ i_L,\ \mathrm{N}(f, \mathrm{ac})) \\
&= \forall(\varnothing \to \bigcirc\ \bigcirc,\ \exists(\bigcirc\ \bigcirc \to \bigcirc)) \wedge \forall(\varnothing \to \bigcirc,\ \exists(\bigcirc \to \bigcirc)) \\
&\equiv \forall(\varnothing \to \bigcirc\ \bigcirc,\ \mathrm{false}) \wedge \forall(\varnothing \to \bigcirc,\ \mathrm{true}) \\
&\equiv \neg\exists(\varnothing \to \bigcirc\ \bigcirc) \\
C_\exists(\mathrm{ac}) &= \vee_{e \in \mathscr{E}}\, \exists(e \circ i_L,\ \mathrm{N}(e, \mathrm{ac})) = \exists(\mathrm{id} \circ i_L,\ \mathrm{N}(\mathrm{id}, \mathrm{ac})) \vee \exists(f \circ i_L,\ \mathrm{N}(f, \mathrm{ac})) \\
&= \exists(\varnothing \to \bigcirc\ \bigcirc,\ \exists(\bigcirc\ \bigcirc \to \bigcirc)) \vee \exists(\varnothing \to \bigcirc,\ \exists(\bigcirc \to \bigcirc)) \\
&\equiv \exists(\varnothing \to \bigcirc\ \bigcirc,\ \mathrm{false}) \vee \exists(\varnothing \to \bigcirc,\ \mathrm{true}) \\
&\equiv \exists(\varnothing \to \bigcirc)
\end{aligned}$$

where $i_L: \varnothing \to \bigcirc\ \bigcirc$ and $f: \bigcirc\ \bigcirc \to \bigcirc$, meaning 'There exists at most one node' and 'There exists a node', respectively.

If the matching and satisfiability notions correspond, that is, in the case of $\mathscr{M}$-matching and $\mathscr{M}$-satisfiability and in the case of $\mathscr{A}$-matching and $\mathscr{A}$-satisfiability, $C_\forall$ and $C_\exists$ reduce to the universal and existential closure, respectively.

Remark. For $\mathscr{M}$-matching and $\mathscr{M}$-satisfiability, and in the case of $\mathscr{A}$-matching and $\mathscr{A}$-satisfiability, one may simply define $C_\forall(\mathrm{ac}) = \forall(i_L, \mathrm{ac})$ and $C_\exists(\mathrm{ac}) = \exists(i_L, \mathrm{ac})$ using the universal and existential closure. In the first case, for every application condition ac over L and for all objects G,

$$\begin{aligned}
G \models \forall(i_L, \mathrm{ac}) &\Leftrightarrow \forall m: L \to G \in \mathscr{M}.\ m \models \mathrm{ac} \\
G \models \exists(i_L, \mathrm{ac}) &\Leftrightarrow \exists m: L \to G \in \mathscr{M}.\ m \models \mathrm{ac}.
\end{aligned}$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

By the definition of $\models$, we have $G \models \forall(i_L, \mathrm{ac})$ if and only if $i_G \models \forall(i_L, \mathrm{ac})$ if and only if $\forall m: L \to G \in \mathscr{M}.\ m \models \mathrm{ac}$. The second case is analogous.

The applicability of a rule can be expressed by a left application condition on the matching morphism. We will use this result in the construction of weakest preconditions in Section 7.

Theorem 8 (applicability of a rule). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$ strictly closed under decomposition. There is a transformation Def from rules to application conditions such that, for every rule ρ and every morphism $m: L \to G$,

$$m \models \mathrm{Def}(\rho) \Leftrightarrow \exists H.\ G \Rightarrow_{\rho,m,m^*} H.$$

In the graph case, the idea of the following construction is to express the gluing condition (Ehrig 1979) of a rule by a conjunction of negative application conditions.

Construction. For a rule $\rho = \langle q, \mathrm{ac} \rangle$, let $\mathrm{Def}(\rho) = \mathrm{Appl}(q) \wedge \mathrm{ac}_L \wedge \mathrm{L}(\rho, \mathrm{ac}_R)$, where, for a rule $q = \langle L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R \rangle$, $\mathrm{Appl}(q) = \wedge_{a \in A} \neg\exists a$ and the index set A ranges over all morphisms $a: L \to L'$ such that the pair $\langle l, a \rangle$ has no pushout complement and there is no decomposition $a = a'' \circ a'$ of morphism a such that a'' in $\mathscr{M}$ is not an isomorphism and $\langle l, a' \rangle$ has no pushout complement. The non-existence of such a decomposition ensures that we only consider minimal morphisms $a: L \to L'$. The morphism a'' is required to be non-isomorphic, otherwise there is always such a decomposition and A would be empty. The requirement a'' in $\mathscr{M}$ is necessary in the context of an $\mathscr{M}$-satisfiable condition.

Remark. For the category $\langle \mathbf{Graphs}, Inj \rangle$, a morphism m satisfies the application condition $\mathrm{Appl}(\rho)$ if and only if m satisfies the contact and identification condition (Ehrig 1979). Note that the graph condition $\mathrm{Appl}(\rho)$ is finite, that is, a finite conjunction of conditions (up to isomorphism).

Example 19. For $\texttt{DeleteNode} = \langle \bigcirc \hookleftarrow \varnothing \hookrightarrow \varnothing \rangle$, the application of `DeleteNode` requires the absence of additional edges adjacent to the deleted node:

$$\mathrm{Appl}(\texttt{DeleteNode}) = \neg\exists(\bigcirc_1 \to \bigcirc_1 \rightarrow \bigcirc_2) \wedge \neg\exists(\bigcirc_1 \to \bigcirc_1 \leftarrow \bigcirc_2) \wedge \neg\exists(\bigcirc_1 \to \bigcirc_1 \circlearrowright).$$

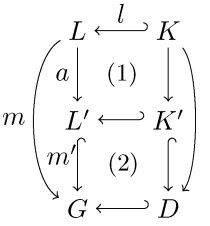
Proof. For plain rules q, we show that, for every morphism $m: L \to G$,

$$m \models \mathrm{Appl}(q) \Leftrightarrow \exists H. G \Rightarrow_{q,m,m^*} H.$$

— **Only if:**
Assume there is no direct derivation $G \Rightarrow_{q,m,m^*} H$, that is, the pair $\langle l, m \rangle$ has no pushout complement. Let $a = m$ and $m' = \mathrm{id}_G$. Now, you have some morphism a with $m = m' \circ a$ for some morphism m' in $\mathscr{M}$ such that $\langle l, a \rangle$ has no pushout complement. If there is a decomposition $a = a'' \circ a'$ of a such that $\langle l, a' \rangle$ has no pushout complement and a'' in $\mathscr{M}$ is not an isomorphism, let $a = a'$, $m' = m' \circ a''$ and m' in $\mathscr{M}$, and repeat the argument. Otherwise there is a no such decomposition and a belongs to the construction. As $m = m' \circ a$ and m' in $\mathscr{M}$, $m \models \exists a$ and $m \not\models \mathrm{Appl}(q)$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

— **If:**
Let $G \Rightarrow_{q,m,m^*} H$, that is, the pair $\langle l, m\rangle$ has a pushout complement. Assume there is $a \in A$ such that $m \models \exists a$, that is, there is a morphism m' in $\mathscr{M}$ such that $m = m' \circ a$. Construct (2) as a pullback of $L' \to G \leftarrow D$. By the universal property of pullbacks, there is a morphism $K \to K'$ such that the resulting diagrams commute. By the pushout–pullback decomposition, the pushout (1)+(2) has a decomposition into two pushouts (1) and (2) and, in particular, $\langle l, a\rangle$ has a pushout complement, which gives a contradiction. Consequently, for every morphism $a \in A$, we have $m \models \neg\exists a$ and $m \models \mathrm{Appl}(q)$.



By the definition of Def and $\models$, Theorem 8, the statement above and the definition of $\Rightarrow$, we have, for every morphism $m: L \to G$, that $m \models \mathrm{Def}(\rho)$ if and only if $m \models \mathrm{Appl}(q) \wedge m \models \mathrm{ac}_L \wedge m \models \mathrm{L}(\rho, \mathrm{ac}_R)$ if and only if $\exists H.G \Rightarrow_{q,m,m^*} H \wedge m \models \mathrm{ac}_L \wedge m^* \models \mathrm{ac}_R$ if and only if $\exists H.G \Rightarrow_{\rho,m,m^*} H$. This completes the proof. □

7. Correctness of transformation systems

In this section we consider the correctness of transformation systems by integration of constraints into application conditions, and, alternatively, by verification.

Our first result is that, given a specification in form of a constraint, any transformation rule can be 'corrected' in the sense of preventing rule applications leading to objects not satisfying the constraint. More precisely, the transformations of Section 6 can be used to integrate constraints into left application conditions of a rule such that every direct derivation is constraint-guaranteeing or constraint-preserving.

Definition 9 (guarantee and preservation of constraints). Given a constraint c, a rule ρ is *c-guaranteeing* if for every direct derivation $G \Rightarrow_\rho H$, we have $H \models c$. A rule ρ is *c-preserving* if for every direct derivation $G \Rightarrow_\rho H$, we have $G \models c$ implies $H \models c$.

By Theorems 5 and 6, we obtain the following result. For a rule ρ, let $[\![\rho]\!]_{\mathscr{A}}$ denote the sets of pairs $\langle G, H\rangle$ with direct derivation $G \Rightarrow_{\rho,m,m^*} H$.

Corollary 5 (guarantee and preservation of constraints). Let $\langle \mathscr{C}, \mathscr{M}\rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object and epi-$\mathscr{M}$-factorisation. There are transformations Gua and Pres from rules and conditions to rules such that for every transformation rule

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

ρ, and every constraint c,

$$[\![\mathrm{Gua}(\rho,c)]\!]_{\mathscr{A}} \subseteq [\![\rho]\!]_{\mathscr{A}} \text{ and } \mathrm{Gua}(\rho,c) \text{ is } c\text{-guaranteeing,}$$

and

$$[\![\mathrm{Pres}(\rho,c)]\!]_{\mathscr{A}} \subseteq [\![\rho]\!]_{\mathscr{A}} \text{ and } \mathrm{Pres}(\rho,c) \text{ is } c\text{-preserving.}$$

Construction. For a rule $\rho = \langle\langle L \hookleftarrow K \hookrightarrow R\rangle, \langle \mathrm{ac}_L, \mathrm{ac}_R\rangle\rangle$, let

$$\mathrm{Gua}(\rho,c) = \langle q, \langle \mathrm{ac}_L \wedge \mathrm{ac}_{gua}, \mathrm{ac}_R\rangle\rangle$$

with $\mathrm{ac}_{gua} = \mathrm{L}(\rho, \mathrm{A}(\rho,c))$ and let

$$\mathrm{Pres}(\rho,c) = \langle q, \langle \mathrm{ac}_L \wedge \mathrm{ac}_{pres}, \mathrm{ac}_R\rangle\rangle$$

with $\mathrm{ac}_{pres} = (\mathrm{A}(\rho^{-1},c) \Rightarrow \mathrm{ac}_{gua})$, where ρ^{-1} denotes the inverse rule

$$\langle\langle R \hookleftarrow K \hookrightarrow L\rangle, \langle \mathrm{ac}_R, \mathrm{ac}_L\rangle\rangle$$

of ρ.

Proof. Clearly, $[\![\mathrm{Gua}(\rho,c)]\!]_{\mathscr{A}} \subseteq [\![\rho]\!]_{\mathscr{A}}$ and $[\![\mathrm{Pres}(\rho,c)]\!]_{\mathscr{A}} \subseteq [\![\rho]\!]_{\mathscr{A}}$. Using Theorems 5 and 6, we show that $\mathrm{Gua}(\rho,c)$ is c-guaranteeing by showing that ac_{gua} guarantees the satisfiability of c. For every direct derivation $G \Rightarrow_{\mathrm{Gua}(\rho,c)} H$, we have $H \models c$ if and only if

$$m^* \models \mathrm{A}(\rho,c)$$

if and only if

$$m \models \mathrm{L}(\rho, \mathrm{A}(\rho,c))$$

if and only if

$$m \models \mathrm{ac}_{gua}.$$

Finally, we show that $\mathrm{Pres}(\rho,c)$ is c-preserving by showing that ac_{pres} preserves the satisfiability of c. For every direct derivation $G \Rightarrow_{\mathrm{Pres}(\rho,c)} H$, we have

$$(G \models c \text{ implies } H \models c)$$

if and only if

$$(m \models \mathrm{A}(\rho^{-1},c) \text{ implies } m \models \mathrm{ac}_{gua})$$

if and only if

$$m \models (\mathrm{A}(\rho^{-1},c) \Rightarrow \mathrm{ac}_{gua})$$

if and only if

$$m \models \mathrm{ac}_{pres}.$$

□

The transformations Gua and Pres enforce correctness by restricting the applicability of transformation rules. In this way, a transformation system can be transformed into a constraint-guaranteeing or constraint-preserving system. Alternatively, one may use the transformations of Section 6 to verify the correctness of transformation systems with respect to given pre- and postconditions. A standard method is to construct a weakest precondition and to show that the precondition implies the weakest precondition.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

In the following, we show how to construct weakest preconditions and strongest postconditions for transformation rules.

Definition 10 (weakest preconditions). A condition c is a *precondition* of a rule ρ relative to a condition d if for all objects G with $G \models c$, we have $G \Rightarrow_\rho H$ implies $H \models d$ for all H, and $G \Rightarrow_\rho H$ for some H.

A precondition c is a *weakest precondition* of ρ relative to d, denoted by $\mathrm{wp}(\rho, d)$, if any precondition of ρ relative to d implies c. Analogously, a condition c is a *liberal precondition* if for all objects $G \models c$ at least $G \Rightarrow_\rho H$ implies $H \models d$ for all H and a *weakest liberal precondition*, denoted by $\mathrm{wlp}(\rho, d)$, if any liberal precondition of ρ relative to d implies c.

The following characterisation points out a simple proof scheme for weakest liberal preconditions.

Fact 8 (characterisation of wlp**).** A condition c is a weakest liberal precondition of ρ relative to d if, for all objects G, we have $(G \models c)$ if and only if $(G \Rightarrow_\rho H$ implies $H \models d$ for all $H)$.

Proof. Assume that for all objects G that $(G \models c)$ if and only if $(G \Rightarrow_\rho H$ implies $H \models d$ for all $H)$. By Definition 10, c is a liberal precondition of ρ relative to d. It remains to show for any other liberal precondition c' of ρ relative to d that c' implies c. Let G be an arbitrary object and assume $(G \models c')$. According to Definition 10, we have $(G \Rightarrow_\rho H$ implies $H \models d$ for all $H)$. Using the assumption, we have $G \models c$. As c' implies c, c is a weakest liberal precondition. □

The following observation is essential for the construction of weakest preconditions.

Fact 9 (existence of results). $G \models \neg\mathrm{wlp}(\rho, \mathrm{false}) \Leftrightarrow G \Rightarrow_\rho H$ for some H.

Proof. There is an object H such that $G \Rightarrow_\rho H$ if and only if there is an object H such that $G \Rightarrow_\rho H$ and $H \models \mathrm{true}$, if and only if not for all objects H we have not $(G \Rightarrow_\rho H$ and $H \models \mathrm{true})$, if and only if not for all objects H we have not $G \Rightarrow_\rho H$ or $H \not\models \mathrm{true}$, if and only if not for all objects H we have $G \Rightarrow_\rho H$ implies $H \models \mathrm{false}$. □

Theorem 9 (weakest preconditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object, epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ strictly closed under decomposition. There are transformations Wlp and Wp such that for every rule ρ and every condition d, $\mathrm{Wlp}(\rho, d)$ is a weakest liberal precondition and $\mathrm{Wp}(\rho, d)$ is a weakest precondition of ρ relative to d.

Construction. For a rule ρ and condition d, let

$$\mathrm{Wlp}(\rho, d) = \mathrm{C}_\forall(\mathrm{Def}(\rho) \Rightarrow \mathrm{L}(\rho, \mathrm{A}(\rho, d))),$$

and

$$\mathrm{Wp}(\rho, d) = \mathrm{Wlp}(\rho, d) \wedge \neg\mathrm{Wlp}(\rho, \mathrm{false}).$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Proof. We show that $\mathrm{Wlp}(\rho, d) \equiv \mathrm{wlp}(\rho, d)$ for arbitrary rules ρ and conditions d. For all objects G, we have:

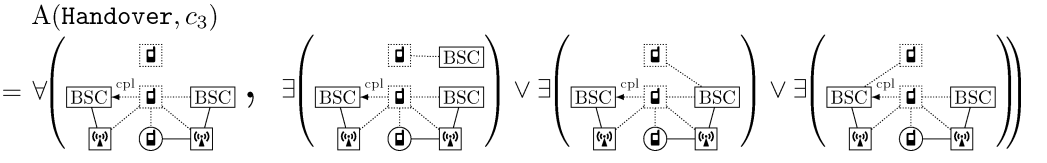
$$\begin{aligned}
G \models \mathrm{Wlp}(\rho, d) &\Leftrightarrow G \models \mathrm{C}_\forall(\mathrm{Def}(\rho) \Rightarrow \mathrm{L}(\rho, \mathrm{A}(\rho, d))) && \text{(Definition Wlp)}\\
&\Leftrightarrow \forall L \xrightarrow{m} G.\ m \models (\mathrm{Def}(\rho) \Rightarrow \mathrm{L}(\rho, \mathrm{A}(\rho, d))) && \text{(Theorem 7)}\\
&\Leftrightarrow \forall L \xrightarrow{m} G.\ m \models \mathrm{Def}(\rho) \Rightarrow m \models \mathrm{L}(\rho, \mathrm{A}(\rho, d)) && \text{(Definition } \models\text{)}\\
&\Leftrightarrow \forall L \xrightarrow{m} G, R \xrightarrow{m^*} H.\ m \models \mathrm{Def}(\rho) \Rightarrow m^* \models \mathrm{A}(\rho, d) && \text{(Theorem 6)}\\
&\Leftrightarrow \forall L \xrightarrow{m} G, R \xrightarrow{m^*} H.\ (G \Rightarrow_{\rho,m,m^*} H) \Rightarrow H \models d && \text{(Theorems 8, 5)}\\
&\Leftrightarrow G \models \mathrm{wlp}(\rho, d). && \text{(Definition wlp, Fact 8)}
\end{aligned}$$

Consequently, $\mathrm{Wlp}(\rho, d)$ is a weakest liberal precondition of ρ relative to d. Then, by Fact 9, Wp is a weakest precondition of ρ relative to d. □

Example 20 (GSM weakest preconditions). Consider the GSM condition *consistency* in Example 4 and the GSM transformation system in Example 7. Weakest preconditions can be used to determine whether the GSM transformation system is or is not correct with respect to the pre- and postcondition *consistency*. We will focus on the rule `Handover` and the subcondition c_3 of *consistency*.

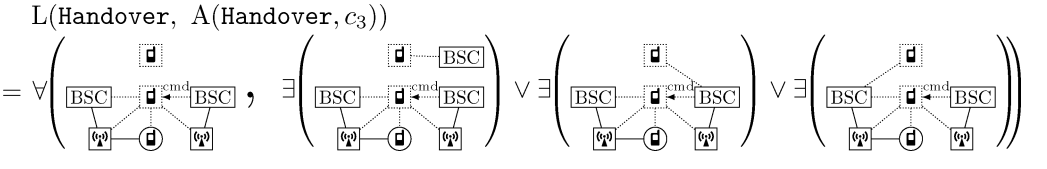
The application of the transformation A to the rule `Handover` and the constraint c_3 yields a right application condition, that is, a condition over the right-hand side of `Handover`:

$\mathrm{A}(\texttt{Handover}, c_3)$



The application of the transformation L to the rule `Handover` and the right application condition $\mathrm{A}(\texttt{Handover}, c_3)$ yields a left application condition, that is, a condition over the left-hand side of `Handover`:

$\mathrm{L}(\texttt{Handover},\ \mathrm{A}(\texttt{Handover}, c_3))$



The application of the transformation Appl to the rule `Handover` yields the left application condition Appl(`Handover`) = true because only injective matches are considered and the rule `Handover` is 'node preserving', thus, for every match, there exists a pushout complement.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Finally, the application of the transformation Wlp to the rule `Handover` and the condition c_3 yields the following constraint over the empty graph:

$$
\begin{aligned}
&\mathrm{Wlp}(\texttt{Handover}, c_3)\\
&= \mathrm{C}_\forall(\mathrm{Def}(\texttt{Handover}) \Rightarrow \mathrm{L}(\texttt{Handover}, \mathrm{A}(\texttt{Handover}, c_3)))\\
&= \mathrm{C}_\forall((\mathrm{Appl}(\texttt{Handover}) \wedge \mathrm{true} \wedge \mathrm{true}) \Rightarrow \mathrm{L}(\texttt{Handover}, \mathrm{A}(\texttt{Handover}, c_3)))\\
&\equiv \mathrm{C}_\forall(\mathrm{true} \Rightarrow \mathrm{L}(\texttt{Handover}, \mathrm{A}(\texttt{Handover}, c_3)))\\
&\equiv \mathrm{C}_\forall(\mathrm{L}(\texttt{Handover}, \mathrm{A}(\texttt{Handover}, c_3)))
\end{aligned}
$$

$$
= \forall\Big(\text{BSC}\ \text{cmd}\ \text{BSC},\ \mathrm{L}(\texttt{Handover}, \mathrm{A}(\texttt{Handover}, c_3)) \Big)
$$

$$
\equiv \forall\Big(\text{BSC}\ \text{cmd}\ \text{BSC},\ \exists\big(\text{BSC}\ \text{cmd}\ \text{BSC} \big) \vee \exists\big(\text{BSC}\ \text{cmd}\ \text{BSC} \big) \vee \exists\big(\text{BSC}\ \text{cmd}\ \text{BSC} \big) \Big)
$$

meaning 'Every mobile station record outside the scope of the rule `Handover` has to be associated with a base transceiver station (outside or inside the occurrence of the left-hand side)'. Obviously, the constraint c_3 implies $\mathrm{Wlp}(\texttt{Handover}, c_3)$.

The rule `Handover` is correct with respect to the pre- and postcondition *consistency*, provided the graph condition *consistency* implies Wlp(`Handover`, *consistency*). The GSM transformation system is correct, provided all rules are correct.

Complementary to weakest liberal preconditions, one may consider strongest postconditions, that is, the strongest condition one can assume to hold in every state after the application of a rule onto an object satisfying a given condition.

Definition 11 (strongest postconditions). A condition c is a *strongest postcondition* of a rule ρ relative to a condition d, denoted by $\mathrm{sp}(\rho, d)$, if for all objects H, H satisfies c if and only if there is some object G such that $G \Rightarrow_\rho H$ and $G \models d$.

As a rule is applicable to G if and only if the inverse rule is applicable to H, we have the following relationship between weakest liberal preconditions and strongest postconditions.

Fact 10 (relationship of sp **and** wlp**).** $\mathrm{sp}(\rho, d) \equiv \neg\mathrm{wlp}(\rho^{-1}, \neg d)$.

Proof. This relationship may be seen as follows:

$$
\begin{aligned}
H \models \neg\mathrm{wlp}(\rho^{-1}, \neg d) &\Leftrightarrow \neg\forall G.\ (H \Rightarrow_{\rho^{-1}} G) \Rightarrow G \models \neg d && \text{(Definition wlp)}\\
&\Leftrightarrow \neg\forall G.\ (G \Rightarrow_{\rho} H) \Rightarrow G \models \neg d && \text{(Definition } \rho^{-1}\text{)}\\
&\Leftrightarrow \exists G.\ (G \Rightarrow_{\rho} H) \wedge G \models d && \text{(Definition } \forall, \Rightarrow, \vee\text{)}\\
&\Leftrightarrow H \models \mathrm{sp}(\rho, d) && \text{(Definition sp)}
\end{aligned}
$$

Consequently, the required relationship between sp and wlp holds. □

As an immediate consequence of Fact 10, we can express the construction of Sp in terms of basic transformations.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Corollary 6 (strongest postconditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{M}$-initial object, epi-$\mathscr{M}$-factorisation and $\mathscr{M}$ strictly closed under decomposition. There is a transformation Sp such that for every rule ρ and every condition d, $\mathrm{Sp}(\rho, d)$ is a strongest postcondition of ρ relative to d.

Construction. For arbitrary rules ρ and conditions d, we have

$$\mathrm{Sp}(\rho, d) = \mathrm{C}_\exists(\mathrm{Def}(\rho^{-1}) \wedge \mathrm{L}(\rho^{-1}, \mathrm{A}(\rho^{-1}, d))).$$

Proof. Let ρ be an arbitrary rule and d be an arbitrary condition.

$$\begin{aligned}
\mathrm{Sp}(\rho, d) &\equiv \mathrm{C}_\exists(\mathrm{Def}(\rho^{-1}) \wedge \mathrm{L}(\rho^{-1}, \mathrm{A}(\rho^{-1}, d))) && \text{(Definition Sp)} \\
&\equiv \neg\mathrm{C}_\forall(\mathrm{Def}(\rho^{-1}) \Rightarrow \neg\mathrm{L}(\rho^{-1}, \mathrm{A}(\rho^{-1}, d))) && \text{(Definition } \mathrm{C}_\exists, \mathrm{C}_\forall, \Rightarrow, \vee) \\
&\equiv \neg\mathrm{C}_\forall(\mathrm{Def}(\rho^{-1}) \Rightarrow \mathrm{L}(\rho^{-1}, \mathrm{A}(\rho^{-1}, \neg d))) && \text{(Definition A, L)} \\
&\equiv \neg\mathrm{Wlp}(\rho^{-1}, \neg d) && \text{(Definition, Correctness Wlp)} \\
&\equiv \mathrm{sp}(\rho, d) && \text{(Fact 10)}
\end{aligned}$$

Consequently, Sp is a strongest postcondition of ρ relative to d. □

The implication problem of conditions, seen as a tautology or satisfiability problem, is covered in Orejas *et al.* (2008), Orejas (2008), Pennemann (2008a; 2008b), although the first two papers consider conditions as in Ehrig *et al.* (2006b). Alternatively, in the next section we consider a translation of graph conditions into first-order graph formulas, which means we can apply existing first-order theorem provers and satisfiability solvers.

8. Expressiveness of graph conditions

We are interested in classifying nested conditions, that is, we want to classify the kind of graph properties that can be expressed by graph conditions, and we want to know if the following problems are decidable: whether or not a graph condition is satisfiable at all (the satisfiability problem); whether or not a graph condition is always valid (the tautology problem). To this end, we compare graph conditions and first-order formulas on graphs (Courcelle 1990; Courcelle 1997) and show that the concepts are expressively equivalent. More precisely, we show that there are transformations from $\mathscr{A}$-satisfiable graph conditions into equivalent graph formulas and *vice versa*, in a way similar to Rensink (2004). However, we consider graphs with parallel edges, that is, multiple, distinguishable edges, which may have the same label. Together with the transformations between $\mathscr{M}$- and $\mathscr{A}$-satisfiability, we get the required result. As an additional result, the transformations enable the use of existing first-order tools to tackle the tautology and satisfiability problem of graph conditions.

For simplicity, we consider graphs with a common labelling function for nodes and edges, while maintaining the disjointness of the node and edge alphabet. Note that all the reasoning can also be carried out for graphs with separate labelling functions. The definition of first-order graph formulas is similar to that in Courcelle (1990) and Courcelle (1997): we allow quantification over nodes and edges, consider a tertiary incidence relation and introduce a unary predicate for each label. For a fixed label alphabet $\mathrm{C} = \langle \mathrm{C_V}, \mathrm{C_E} \rangle$ with $\mathrm{C_V} \cap \mathrm{C_E} = \varnothing$, the induced signature $\Sigma = (\varnothing, \{\mathrm{lab}_b \mid b \in \mathrm{C}\} \cup \{\mathrm{inc}, =\})$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

contains a unary predicate symbol lab_b for every label b, a tertiary predicate symbol inc and a binary predicate symbol $=$.

Definition 12 (first-order graph formulas). Let Var be an infinite, countable set of variables. The set of all (*first-order graph*) *formulas* over Σ is inductively defined as follows: for $b \in \mathrm{C}$ and $x, y, z \in \mathrm{Var}$, $\mathrm{lab}_b(x)$, $\mathrm{inc}(x, y, z)$ and $x = y$ are formulas over Σ. For formulas F, F_j $(j \in J)$ over Σ and $x \in \mathrm{Var}$, true, $\neg F$, $\wedge_{j\in J} F_j$, and $\exists x\ F$ are formulas over Σ. Additionally, false abbreviates $\neg$true, $\vee_{j\in J} F_j$ abbreviates $\neg\wedge_{j\in J} \neg F_j$, $F \Rightarrow G$ abbreviates $\neg F \vee G$, $\forall x\ F$ abbreviates $\neg\exists x\ \neg F$, edge(x) abbreviates $\exists y\ \exists z$ inc(x, y, z), and node(x) abbreviates $\neg$edge(x). For a formula F, Free(F) denotes the set of all *free* variables of F. A formula is *closed* if Free$(F) = \varnothing$, that is, if F does not contain free variables.

The semantics of graph formulas are given in terms of a domain of values and an interpretation of the (non-logical) symbols of Σ.

Definition 13 (semantics of graph formulas). For a non-empty graph G, let (D_G, I_G) be the induced Σ-*structure* consisting of a non-empty domain $\mathrm{D}_G = \mathrm{V}_G + \mathrm{E}_G$ and the interpretation I_G of the predicate symbols with $I_G(\mathrm{lab}_b)(d) = \mathit{true}$ if and only if $\mathrm{l}_G(d) = b$, $I_G(\mathrm{inc})(e, u, v) = \mathit{true}$ if and only if $e \in \mathrm{E}_G$, $\mathrm{s}_G(e) = u$ and $\mathrm{t}_G(e) = v$, and $I_G(=)(d, d') = \mathit{true}$ if and only if $d = d'$. The *semantic* $G[\![F]\!](\sigma)$ of a formula F over Σ in the graph G under the assignment $\sigma : \mathrm{Var} \to \mathrm{D}_G$ is inductively defined by:

$G[\![p(x_1, \ldots, x_n)]\!](\sigma) = I_G(p)(\sigma(x_1), \ldots, \sigma(x_n))$ for an n-ary predicate p.
$G[\![\exists x\ F]\!](\sigma) = \mathit{true}$ if and only if there exists $d \in \mathrm{D}_G$ such that $G[\![F]\!](\sigma\{x/d\}) = \mathit{true}$, where $\sigma\{x/d\}$ is the modified assignment with $\sigma\{x/d\}(x) = d$ and $\sigma\{x/d\}(y) = \sigma(y)$ otherwise.

The semantics is extended to the operators true, $\neg$ and $\wedge$ in the usual way.

A graph G *satisfies* a formula F, denoted by $G \models F$, if and only if for all assignments $\sigma : \mathrm{Var} \to \mathrm{D}_G$, $G[\![F]\!](\sigma) = \mathit{true}$.

Example 21. The first-order graph formula

$$F = \mathrm{node}(u) \wedge \mathrm{lab}_0(u) \Rightarrow \exists v \exists e\ \mathrm{node}(v) \wedge \mathrm{lab}_1(v) \wedge \mathrm{inc}(e, u, v) \wedge \mathrm{lab}_a(e)$$

has the meaning 'Whenever there is a node with label 0, there exists an edge with label a from this node to a node with label 1'.

For automated theorem proving we require an exact axiomatisation of the above structures to restrict considerations to directed, totally labelled graphs – see Courcelle (1997) for unlabelled graphs.

Fact 11 (axiomatisation). For an alphabet $\mathrm{C} = \langle \mathrm{C_V}, \mathrm{C_E} \rangle$ with $\mathrm{C_V} \cap \mathrm{C_E} = \varnothing$, the class of structures (D_G, I_G) over Σ are exactly those that satisfy the following properties:

(1) $\forall e\ \forall x\ \forall y\ (\mathrm{inc}(e, x, y) \Rightarrow \neg\exists u\ \exists v\ (\mathrm{inc}(x, u, v) \vee \mathrm{inc}(y, u, v)))$.
(2) $\forall e\ \forall x\ \forall y\ \forall x'\ \forall y'\ ((\mathrm{inc}(e, x, y) \wedge \mathrm{inc}(e, x', y')) \Rightarrow (x = x' \wedge y = y'))$.
(3) $\forall x\ \wedge_{b,d\in \mathrm{C_V}, b\neq d}\ \neg(\mathrm{lab}_b(x) \wedge \mathrm{lab}_d(x))$ and $\forall x\ \wedge_{b,d\in \mathrm{C_E}, b\neq d}\ \neg(\mathrm{lab}_b(x) \wedge \mathrm{lab}_d(x))$.
(4) $\forall x\ \mathrm{node}(x) \Leftrightarrow \vee_{b\in \mathrm{C_V}}\ \mathrm{lab}_b(x)$ and $\forall x\ \mathrm{edge}(x) \Leftrightarrow \vee_{b\in \mathrm{C_E}}\ \mathrm{lab}_b(x)$.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

The meaning of the axioms is:

(1) (target and source) nodes cannot be edges;
(2) an edge has at most one source and one target;
(3) an element has at most one label; and
(4) every node has a node label, every edge has an edge label.

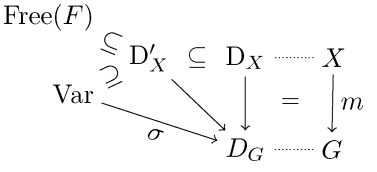
Every other statement, such as 'An edge has source and target nodes' (otherwise, it is not an edge), is implicit.

There is a transformation from first-order graph formulas into graph conditions.

Theorem 10 (from formulas to conditions). There is a transformation Cond from first-order graph formulas to graph conditions such that, for all first-order formulas F over Σ and all graphs G,

$$G \models F \iff G \models_{\mathscr{A}} \mathrm{Cond}(F).$$

Before we give a construction of the transformation Cond, we shall make some preliminary observations. We consider formulas on directed, labelled graphs with multiple parallel edges, so edges are handled as individuals. If F is a *rectified* formula, that is, distinct quantifiers bind occurrences of distinct variables, the variables of F can be represented by isolated nodes and edges in the graphs of a constructed condition. Let X be such a graph. If the set $\mathrm{D}'_X = \mathrm{Iso}_X + \mathrm{E}_X$ of all isolated nodes and all edges in X is a subset of the set Var of variables, then every graph morphism $m: X \to G$ into a non-empty graph G induces an assignment $\sigma: \mathrm{Var} \to \mathrm{D}_G$ such that $m = \sigma[\mathrm{D}'_X]$, that is, $m(x) = \sigma(x)$ for each $x \in \mathrm{D}'_X$. Conversely, an assignment $\sigma: \mathrm{Var} \to \mathrm{D}_G$ induces a mapping $\mathrm{D}'_X \to \mathrm{D}_G$ that may be extended to a graph morphism $m: X \to G$ with $m = \sigma[\mathrm{D}'_X]$.



The key idea of the transformation Cond is to represent existential quantification in a formula by disjunction over all possible choices, that is, nodes or edges with all their possible labels. Some of these branches may later become unsatisfiable, depending on the lab predicates occurring.

Construction. Assume that F is a closed and rectified formula over Σ (otherwise, consider the universal closure of F and rename the variables). The graph condition is given by $\mathrm{Cond}(F) = \mathrm{Cond}(\varnothing, F)$, where $\varnothing$ denotes the empty graph. For a formula F over Σ and a graph X with $\mathrm{Free}(F) \subseteq \mathrm{D}'_X \subseteq \mathrm{Var}$, the graph condition $\mathrm{Cond}(X, F)$ is constructed as follows:

— $\mathrm{Cond}(X, \mathrm{lab}_b(x))$
  $=$ true if $\mathrm{l}_X(x) = b$, and false otherwise.
— $\mathrm{Cond}(X, \mathrm{inc}(x, y, z))$
  $= \exists(X \to X[\mathrm{s}_X(x) = y]) \wedge \exists(X \to X[\mathrm{t}_X(x) = z])$ if $x \in \mathrm{E}_X$ and $y, z \in \mathrm{V}_X$, and false otherwise.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

A graph $X[x=y]$ is obtained from X by identifying the elements x and y.

— $\mathrm{Cond}(X, x=y)$
$= \exists(X \to X[x=y])$ if and only if x and y are identifiable, that is, $(x, y \in \mathrm{V}_X$ or $x, y \in \mathrm{E}_X$, $\mathrm{s}_X(x)=\mathrm{s}_X(y)$, $\mathrm{t}_X(x)=\mathrm{t}_X(y))$ and $\mathrm{l}_X(x)=\mathrm{l}_X(y)$, and false otherwise.

— $\mathrm{Cond}(X, \exists x\, F)$
$= \vee_{b\in \mathrm{C_V}} \exists(X \to Y, \mathrm{Cond}(Y, F)) \vee \vee_{b\in \mathrm{C_E}, d, d' \in \mathrm{C_V}} \exists(X \to Z, \mathrm{Cond}(Z, F))$ where $Y = X + \textcircled{b}_x$ is obtained from X by adding a node x with label b and $Z = X + \textcircled{d} \xrightarrow[x]{b} \textcircled{d'}$ by adding an edge x with label b together with a d-labelled source and a d'-labelled target.

— $\mathrm{Cond}(X, F)$ is extended to Boolean formulas with the operators true, $\neg$, $\wedge$ as usual.

Example 22. Let $\mathrm{C_V} = \{0, 1\}$ and $\mathrm{C_E} = \{a, b\}$. The first-order graph formula $F = \exists x\ \mathrm{lab}_a(x)$ with the meaning 'There exists an item with label a' is transformed into the graph condition

$$
\begin{aligned}
& \mathrm{Cond}(\exists x\ \mathrm{lab}_a(x)) = \mathrm{Cond}(\varnothing, \exists x\ \mathrm{lab}_a(x)) \\
= \quad & \vee_{m\in \mathrm{C_V}} \exists(\varnothing \to \textcircled{m}_x,\ \mathrm{Cond}(\textcircled{m}_x, \mathrm{lab}_a(x))) \\
& \vee \vee_{k,n\in \mathrm{C_V}, m\in \mathrm{C_E}} \exists(\varnothing \to \textcircled{k} \xrightarrow[x]{m} \textcircled{n},\ \mathrm{Cond}(\textcircled{k} \xrightarrow[x]{m} \textcircled{n}, \mathrm{lab}_a(x))) \\
= \quad & \exists(\varnothing \to \textcircled{0},\ \mathrm{false}) \vee \exists(\varnothing \to \textcircled{1},\ \mathrm{false}) \\
& \vee \vee_{k,n\in \mathrm{C_V}} (\exists(\varnothing \to \textcircled{k} \xrightarrow[x]{a} \textcircled{n},\ \mathrm{true}) \vee \exists(\varnothing \to \textcircled{k} \xrightarrow[x]{b} \textcircled{n},\ \mathrm{false})) \\
\equiv \quad & \vee_{k,n\in \mathrm{C_V}} \exists(\varnothing \to \textcircled{k} \xrightarrow[x]{a} \textcircled{n})
\end{aligned}
$$

with the meaning 'There exists an edge with label a and arbitrarily labelled source and target'. This example shows that, in the case of totally labelled graphs, unspecifiedness is represented by disjunction over all possibilities, which may, at least temporarily, lead to rather large conditions. A remedy could be the consideration of conditions over partial and/or partially labelled graphs.

The proof of Theorem 10 depends on the following lemma.

Lemma 4. For all rectified formulas F over Σ, all graphs G, and all graphs X with $\mathrm{Free}(F) \subseteq \mathrm{D}'_X \subseteq \mathrm{Var}$ we have that for all morphisms $m: X \to G$ and all assignments $\sigma: \mathrm{Var} \to \mathrm{D}_G$ with $m = \sigma[\mathrm{D}'_X]$, $G[\![F]\!](\sigma) = \mathit{true}$ if and only if $m \models_{\mathscr{A}} \mathrm{Cond}(X, F)$.

Proof. The proof is by structural induction.

— **Basis:**
For $F = \mathrm{true}$, the statement is straightforward.
For atomic formulas, the statement follows directly from the definitions:

$$
\begin{aligned}
(1)\ G[\![\mathrm{lab}_b(x)]\!](\sigma) = \mathit{true} & \Leftrightarrow \mathrm{l}_G(\sigma(x)) = b && \text{(Definition of } [\![\,]\!]) \\
& \Leftrightarrow \mathrm{l}_G(m(x)) = b && (m = \sigma[\mathrm{D}'_X],\ *) \\
& \Leftrightarrow m \models_{\mathscr{A}} \mathrm{true} \text{ and } \mathrm{l}_X(x) = b && (m \text{ label-preserving}) \\
& \Leftrightarrow m \models_{\mathscr{A}} \mathrm{Cond}(X, \mathrm{lab}_b(x)) && \text{(Definition of Cond)}
\end{aligned}
$$

$* \ x \in \mathrm{Free}(\mathrm{lab}_b(x)) \subseteq \mathrm{D}'_X$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

(2) $G[\![\mathrm{inc}(x,y,z)]\!](\sigma) = \mathit{true}$

$\Leftrightarrow \sigma(x) \in \mathrm{E}_G,\ \mathrm{s}_G(\sigma(x)) = \sigma(y)$ and $\mathrm{t}_G(\sigma(x)) = \sigma(z)$ (Definition of $[\![\,]\!]$)

$\Leftrightarrow \mathrm{m}(x) \in \mathrm{E}_G,\ \mathrm{s}_G(m(x)) = m(y)$ and $\mathrm{t}_G(m(x)) = m(z)$ ($m = \sigma[\mathrm{D}'_X]$, $*$)

$\Leftrightarrow x \in \mathrm{E}_X,\ y,z \in \mathrm{V}_X,\ m(\mathrm{s}_X(x)) = m(y)$
and $m(\mathrm{t}_X(x)) = m(z)$ (m is a morphism)

$\Leftrightarrow m \models_{\mathscr{A}} \exists(X \to X[\mathrm{s}_X(x) = y]) \wedge \exists(X \to X[\mathrm{t}_X(x) = y])$
and $x \in \mathrm{E}_X,\ y,z \in \mathrm{V}_X$ (Definition of $\models_{\mathscr{A}}$)

$\Leftrightarrow m \models_{\mathscr{A}} \mathrm{Cond}(X, \mathrm{inc}(x,y,z))$ (Definition of Cond)

$*\ x,y,z \in \mathrm{Free}(\mathrm{inc}(x,y,z)) \subseteq \mathrm{D}'_X$

(3) $G[\![x = y]\!](\sigma) = \mathit{true}$

$\Leftrightarrow (\sigma(x), \sigma(y) \in \mathrm{V}_G$ or $\sigma(x), \sigma(y) \in \mathrm{E}_G)$ and $\sigma(x) = \sigma(y)$ (Definition of $[\![\,]\!]$)

$\Leftrightarrow m(x) = m(y)$ ($m = \sigma[\mathrm{D}'_X]$, $*$)

$\Leftrightarrow m \models_{\mathscr{A}} \exists(X \to X[x = y])$ and x, y are identifiable (Definition of $\models_{\mathscr{A}}$)

$\Leftrightarrow m \models_{\mathscr{A}} \mathrm{Cond}(X, x = y)$ (Definition of Cond)

$*\ x,y \in \mathrm{Free}(x = y) \subseteq \mathrm{D}'_X,\ m$ is a morphism

— **Hypothesis:**
Assume, the statement holds for rectified formulas F and F_j $(j \in J)$.

— **Step:**
For formulas of the form $\exists x\ F$, the proof uses the induction hypothesis:

(4) $G[\![\exists x\ F]\!](\sigma) = \mathit{true}$

$\Leftrightarrow \exists o \in \mathrm{D}_G.\ G[\![F]\!](\sigma\{x/o\}) = \mathit{true}$ (Definition $[\![\,]\!]$)

$\Leftrightarrow \exists o \in \mathrm{V}_G.\ \exists b \in \mathrm{C}_{\mathrm{V}}.\ \mathrm{l}_G(o) = b$ and $G[\![F]\!](\sigma\{x/o\}) = \mathit{true}$ or
$\exists o \in \mathrm{E}_G.\ \exists b \in \mathrm{C}_{\mathrm{E}}.\ \mathrm{l}_G(o) = b$ and $G[\![F]\!](\sigma\{x/o\}) = \mathit{true}$ (Assignment)

$\Leftrightarrow \exists b \in \mathrm{C}_{\mathrm{V}}.\ \exists m'.\ m = m' \circ X \to Y$ and $m' \models_{\mathscr{A}} \mathrm{Cond}(Y, F)$ or
$\exists b \in \mathrm{C}_{\mathrm{E}}.\ \exists d, d' \in \mathrm{C}_{\mathrm{V}}.\ \exists m'.\ m = m' \circ X \to Z,\ m' \models_{\mathscr{A}} \mathrm{Cond}(Z, F)$
(Hypothesis, $m' = \sigma\{x/o\}[\mathrm{D}'_X]$)

$\Leftrightarrow \exists b \in \mathrm{C}_{\mathrm{V}}.\ m \models_{\mathscr{A}} \exists(X \to Y,\ \mathrm{Cond}(Y, F))$ or
$\exists b \in \mathrm{C}_{\mathrm{E}}.\ \exists d, d' \in \mathrm{C}_{\mathrm{V}}.\ m \models_{\mathscr{A}} \exists(X \to Z,\ \mathrm{Cond}(Z, F))$ (Definition $\models_{\mathscr{A}}$)

$\Leftrightarrow m \models_{\mathscr{A}} \mathrm{Cond}(X, \exists x\ F)$ (Definition Cond)

where $Y = X + \overset{b}{\underset{x}{\bigcirc}}$ is obtained from X by adding a node x with label b and $Z = X + \overset{d}{\bigcirc} \xrightarrow[x]{b} \overset{d'}{\bigcirc}$ by adding an edge x with label b together with d-labelled source and d'-labelled target nodes. The rectifiedness of F guarantees that x does not already exists. For formulas built with the operators $\neg$ and $\wedge$, the proof of the statement is straightforward and uses the induction hypothesis.

This completes the proof. □

Proof of Theorem 10. For all closed, rectified formulas F over Σ and all graphs G, we have that $G \models F$ if and only if for all assignments $\sigma : \mathrm{Var} \to \mathrm{D}_G$, $G[\![F]\!](\sigma) = \mathit{true}$ (Definition of $\models$) if and only if for all morphisms $i_G : \varnothing \to G$, $i_G \models_{\mathscr{A}} \mathrm{Cond}(\varnothing, F)$ (Lemma 4, for $\mathrm{Free}(F) = \varnothing = \mathrm{D}'_X$ and $X = \varnothing$) if and only if $G \models_{\mathscr{A}} \mathrm{Cond}(\varnothing, F)$ (Definition $\models_{\mathscr{A}}$). □

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Conversely, there is a transformation from graph conditions into first-order graph formulas.

Theorem 11 (from conditions to formulas). There exists a transformation Form from graph conditions to first-order graph formulas such that for all graph conditions c over the empty graph $\varnothing$ and all graphs G,

$$G \models_{\mathscr{A}} c \iff G \models \mathrm{Form}(c).$$

For the construction of the transformation Form we consider morphisms and conditions in a certain normal form such that the identities of the nodes and edges in a condition can be associated with variables, that is, no unnecessary identity changes of elements take place within a condition. A morphism $a: P \to C$ is said to be *identity preserving* if $y \in C$ implies ($y = a(y)$ or ($y \notin a(P)$ and $y \notin P$)) elementwise for nodes and edges. This is no restriction as every condition may be transformed into an equivalent condition by subsequently replacing all morphisms not in this form by morphisms with this property.

Fact 12. Every identity-preserving graph morphism a with domain P is either the identity $\mathrm{id}: P \to P$ or can be decomposed into morphisms that, starting from P, subsequently add nodes, add edges, identify nodes and identify edges. There may be multiple decompositions for a given morphism.

The key idea of the transformation Form is to decompose the condition $\exists(a, c')$ into a nested condition $\exists(a_1, \ldots, \exists(a_n, \exists(\mathrm{id}, c'))\ )$ such that in each morphism a_j, $1 \leqslant j \leqslant n$, one element is added or two elements are identified.

Construction. For conditions based on identity-preserving graph morphisms, Form is defined as follows:

$$\begin{aligned}
\mathrm{Form}(\mathrm{true}) &= \mathrm{true}\\
\mathrm{Form}(\exists(\mathrm{id}, c')) &= \mathrm{Form}(c')\\
\mathrm{Form}(\exists([vb]a, c')) &= \exists v\ (\mathrm{node}(v) \wedge \mathrm{lab}_b(v) \wedge \mathrm{Form}(\exists(a, c')))\\
\mathrm{Form}(\exists([euvb]a, c')) &= \exists e\ (\mathrm{inc}(e, u, v) \wedge \mathrm{lab}_b(e) \wedge \mathrm{Form}(\exists(a, c')))\\
\mathrm{Form}(\exists([u = v]a, c')) &= (u = v \wedge \mathrm{Form}(\exists(a, c')))\\
\mathrm{Form}(\exists([e = e']a, c')) &= (e = e' \wedge \mathrm{Form}(\exists(a, c')))
\end{aligned}$$

where the morphisms $[vb]a$, $[euvb]a$, $[u = v]a$ and $[e = e']a$ can be decomposed into some morphism $a: P \to C$ after $[vb]P \to P$ (addition of a node labelled b), $[euvb]P \to P$ (addition of an edge labelled b to some nodes u and v), $[u = v]P \to P$ (identification of nodes) and $[e = e']P \to P$ (identification of edges), respectively. Form is extended for the operators $\neg, \wedge$ as usual. Note, Form is not unique, but is well defined.

Remark. For every graph condition c over P, $\mathrm{Free}(\mathrm{Form}(c)) \subseteq \mathrm{D}_P \subseteq \mathrm{Var}$, that is, all free variables of the constructed formula $\mathrm{Form}(c)$ correspond to elements in P.

Example 23. Let $\mathrm{C_V} = \{0, 1\}$ and $\mathrm{C_E} = \{b, d\}$. The graph condition

$$\forall(\varnothing \to \textcircled{0}_u,\ \exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b,e} \textcircled{1}_v))$$

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

with the meaning 'For every node with label 0, there exists an outgoing b-labelled edge to a node with label 1' is transformed into the following first-order formula:

$$
\begin{aligned}
&\mathrm{Form}(\neg\exists(\varnothing \to \textcircled{0}_u,\ \neg\exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v)))\\
&\quad = \neg\mathrm{Form}(\exists(\varnothing \to \textcircled{0}_u,\ \neg\exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v)))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \mathrm{Form}(\exists(\textcircled{0}_u \to \textcircled{0}_u,\ \neg\exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v))))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \mathrm{Form}(\neg\exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v)))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \neg\mathrm{Form}(\exists(\textcircled{0}_u \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v)))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \neg\exists v\ (\mathrm{node}(v) \wedge \mathrm{lab}_1(v) \wedge \mathrm{Form}(\exists(\textcircled{0}_u\ \textcircled{1}_v \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v))))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \neg\exists v\ (\mathrm{node}(v) \wedge \mathrm{lab}_1(v) \wedge \exists e\ (\mathrm{inc}(e,u,v)\\
&\qquad\qquad \wedge \mathrm{lab}_b(e) \wedge\ \mathrm{Form}(\exists(\textcircled{0}_u \xrightarrow{b} \textcircled{1}_v \to \textcircled{0}_u \xrightarrow{b} \textcircled{1}_v)))))\\
&\quad = \neg\exists u\ (\mathrm{node}(u) \wedge \mathrm{lab}_0(u) \wedge \neg\exists v\ (\mathrm{node}(v) \wedge \mathrm{lab}_1(v) \wedge \exists e\ (\mathrm{inc}(e,u,v)\\
&\qquad\qquad \wedge \mathrm{lab}_b(e) \wedge \mathrm{true})))\\
&\quad \equiv \forall u\ ((\mathrm{node}(u) \wedge \mathrm{lab}_0(u)) \Rightarrow \exists v\ \exists e\ (\mathrm{node}(v) \wedge \mathrm{lab}_1(v) \wedge \mathrm{inc}(e,u,v) \wedge \mathrm{lab}_b(e))).
\end{aligned}
$$

with the same meaning 'For every node u with label 0, there is a node v with label 1 and a b-labelled edge from u to v'.

The proof of Theorem 11 depends on the following lemma.

Lemma 5. For all conditions c over P and all graphs G we have that for all graph morphisms $m: P \to G$ and all assignments $\sigma: \mathrm{Var} \to \mathrm{D}_G$ with $m = \sigma[\mathrm{D}_P]$,

$$m \models_{\mathscr{A}} c \text{ iff } G[\![\mathrm{Form}(c)]\!](\sigma) = \mathit{true}.$$

Proof. The proof is by structural induction.

— **Basis:**
For $c = \mathrm{true}$, we have $m \models \mathrm{true} \Leftrightarrow G[\![\mathrm{true}]\!](\sigma) = \mathit{true}$.

— **Hypothesis:**
Assume the statement holds for condition c'.

— **Step:**
For conditions of the form $c = \exists(a, c')$, the statement is proved by induction over the decomposition of morphisms. If the morphism a is the identity $\mathrm{id}: P \to P$, we note that for all morphisms $m: P \to G$ and all assignments $\sigma: \mathrm{Var} \to \mathrm{D}_G$ with $m = \sigma[\mathrm{D}_P]$ we have $m \models_{\mathscr{A}} \exists(\mathrm{id}, c') \Leftrightarrow m \models_{\mathscr{A}} c' \Leftrightarrow G[\![\mathrm{Form}(c')]\!](\sigma) = \mathit{true}$ (hypothesis for c').
Now we assume that the statement holds for a condition $\exists(a, c')$ and prove that the statement holds for all conditions $\exists(a', c')$ with extended morphism $a' = a[vb]$, $a[euvb]$, $a[u{=}v]$, and $a[e{=}e']$:

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

(1) $m \models_{\mathscr{A}} \exists([vb]a, c')$
$\Leftrightarrow m \models_{\mathscr{A}} \exists([vb]P \to P,\ \exists(a, c'))$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \exists m'.\ m = m' \circ [vb]P \to P$ and $m' \models_{\mathscr{A}} \exists(a, c')$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \exists o \in \mathrm{D}_G.\ \sigma' = \sigma\{v/o\}$ and $G[\![\mathrm{node}(v)]\!](\sigma') = \mathit{true}$
and $G[\![\mathrm{lab}_b(v)]\!](\sigma') = \mathit{true}$
and $G[\![\mathrm{Form}(\exists(a, c'))]\!](\sigma') = \mathit{true}$ (Hypothesis, $\sigma'[\mathrm{D}_P] = m'$)
$\Leftrightarrow G[\![\exists v\ (\mathrm{node}(v) \wedge \mathrm{lab}_b(v) \wedge \mathrm{Form}(\exists(a, c')))]\!](\sigma) = \mathit{true}$ (Definition $[\![\,]\!]$)
$\Leftrightarrow G[\![\mathrm{Form}(\exists([vb]a, c'))]\!](\sigma) = \mathit{true}$ (Definition Form)
$\Leftrightarrow G \models \mathrm{Form}(\exists([vb]a, c'))$ (Definition $\models$)

(2) $m \models_{\mathscr{A}} \exists([euvb]a, c')$
$\Leftrightarrow m \models_{\mathscr{A}} \exists([euvb]P \to P,\ \exists(a, c'))$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \exists m'.\ m = m' \circ [euvb]P \to P$ and $m' \models_{\mathscr{A}} \exists(a, c')$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \exists o \in \mathrm{D}_G.\ \sigma' = \sigma\{e/o\}$ and $G[\![\mathrm{inc}(e, u, v)]\!](\sigma') = \mathit{true}$
and $G[\![\mathrm{lab}_b(e)]\!](\sigma') = \mathit{true}$
and $G[\![\mathrm{Form}(\exists(a, c'))]\!](\sigma') = \mathit{true}$ (Hypothesis, $\sigma'[\mathrm{D}_P] = m'$)
$\Leftrightarrow G[\![\exists e\ (\mathrm{inc}(e, u, v) \wedge \mathrm{lab}_b(e) \wedge \mathrm{Form}(\exists(a, c')))]\!](\sigma) = \mathit{true}$ (Definition $[\![\,]\!]$)
$\Leftrightarrow G[\![\mathrm{Form}(\exists([euvb]a, c'))]\!](\sigma) = \mathit{true}$ (Definition Form)
$\Leftrightarrow G \models \mathrm{Form}(\exists([euvb]a, c'))$ (Definition $\models$)

(3) $m \models_{\mathscr{A}} \exists([u{=}v]a, c')$
$\Leftrightarrow m \models_{\mathscr{A}} \exists([u{=}v]P \to P,\ \exists(a, c'))$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \exists m'.\ m = m' \circ [u{=}v]P \to P$ and $m' \models_{\mathscr{A}} \exists(a, c')$ (Definition $\models_{\mathscr{A}}$)
$\Leftrightarrow \sigma(u) = \sigma(v)$ and $G[\![u = v]\!](\sigma) = \mathit{true}$
and $G[\![\mathrm{Form}(\exists(a, c'))]\!](\sigma) = \mathit{true}$ (Hypothesis, $\sigma[\mathrm{D}_P] = m'$)
$\Leftrightarrow G[\![(u = v \wedge \mathrm{Form}(\exists(a, c')))]\!](\sigma) = \mathit{true}$ (Definition $[\![\,]\!]$)
$\Leftrightarrow G[\![\mathrm{Form}(\exists([u{=}v]a, c'))]\!](\sigma) = \mathit{true}$ (Definition Form)
$\Leftrightarrow G \models \mathrm{Form}(\exists([u{=}v]a, c'))$ (Definition $\models$)

(4) $m \models_{\mathscr{A}} \exists([e{=}e']a, c') \Leftrightarrow G \models \mathrm{Form}(\exists([e{=}e']a, c'))$ (similar to (3)).

Since all identity-preserving graph morphisms may be decomposed into graph morphisms that subsequently add nodes, add edges, identify nodes and identify edges, the statement holds for all conditions of the form $\exists(a, c')$.

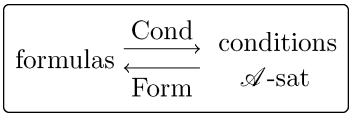
For conditions built with the operators $\neg$, $\wedge$, the proof of the statement is straightforward. □

Proof of Theorem 11. For all conditions c over $\varnothing$ and all graphs G, we have that $G \models_{\mathscr{A}} c$ if and only if for all morphisms $i_G : \varnothing \to G$, $i_G \models_{\mathscr{A}} c$ (Definition $\models_{\mathscr{A}}$) if and only if for all assignments $\sigma : \mathrm{Var} \to \mathrm{D}_G$, $G[\![\mathrm{Form}(c)]\!](\sigma) = \mathit{true}$ (Lemma 5, for $P = \varnothing$ and $\mathrm{Free}(\mathrm{Form}(c)) = \varnothing = \mathrm{D}_P$) if and only if $G \models \mathrm{Form}(c)$ (Definition $\models$). □

Theorems 10 and 11 give us the equivalence of graph conditions under $\mathscr{A}$-satisfiability and first-order graph formulas.

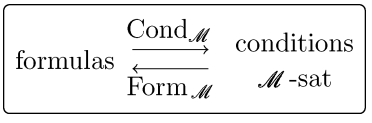
Corollary 7 ($\mathscr{A}$: equivalence of graph conditions and graph formulas). $\mathscr{A}$-satisfiable graph conditions and first-order graph formulas are expressively equivalent:

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

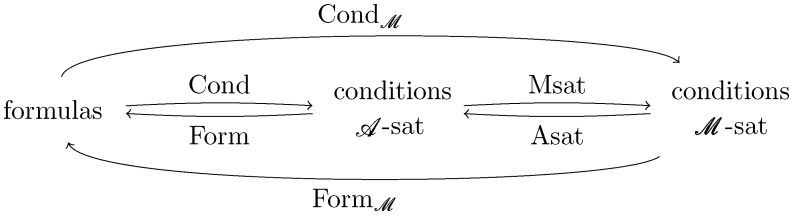


The standard semantics for high-level conditions is $\mathscr{M}$-satisfiability. By Fact 4 and Theorems 1 and 2, $\mathscr{A}$-satisfiable graph conditions can be transformed into $\mathscr{M}$-satisfiable graph conditions and *vice versa*. Therefore, $\mathscr{M}$-satisfiable graph conditions and first-order graph formulas are expressively equivalent also.

Corollary 8 ($\mathscr{M}$: equivalence of graph conditions and graph formulas). $\mathscr{M}$-satisfiable graph conditions and first-order graph formulas are expressively equivalent:



Proof. The statement is an immediate consequence of Theorems 10, 11 together with Theorems 1, 2, which allow us to switch between $\mathscr{A}$- and $\mathscr{M}$-satisfiability. Define $\mathrm{Cond}_{\mathscr{M}} = \mathrm{Msat} \circ \mathrm{Cond}$ and $\mathrm{Form}_{\mathscr{M}} = \mathrm{Form} \circ \mathrm{Asat}$. By Theorems 10 and 1, for all formulas F over Σ and all graphs G, $G \models F \Leftrightarrow G \models_{\mathscr{A}} \mathrm{Cond}(F) \Leftrightarrow G \models \mathrm{Msat}(\mathrm{Cond}(F)) = \mathrm{Cond}_{\mathscr{M}}(F)$. By Theorems 11 and 2, for all formulas F over Σ and all graphs G, we have $G \models c \Leftrightarrow G \models_{\mathscr{A}} \mathrm{Asat}(c) \Leftrightarrow G \models \mathrm{Form}(\mathrm{Asat}(c)) = \mathrm{Form}_{\mathscr{M}}(c)$:



□

Conditions and formulas are *finite* if the index set J of every conjunction $\wedge_{j \in J}$ and every disjunction $\vee_{j \in J}$ is finite. In the following, we want to strengthen the statement of Corollary 8 by showing that every finite graph condition can be translated into a finite first-order graph formula and *vice versa*. We first state the additional requirements for a weak adhesive HLR category to ensure the effectiveness of the constructions in this paper.

Assumption 2. Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with:

– a *finite number of epimorphisms for every given domain*, that is, for all objects P, there exist only finitely many epimorphisms with domain P up to isomorphism; and

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

– a *finite number of matches*, that is, for every morphism $l: K \hookrightarrow L$ and every object G, there exist only finitely many morphisms $m: L \to G$ up to isomorphism such that $\langle l, m \rangle$ has a pushout complement.

A finite number of matches implies finitely many morphisms between every pair of objects. The category (**Graphs**, *Inj*) satisfies the above requirements. As an example, we show that Asat and Msat yield finite results for finite inputs.

Fact 13 (transformation of finite conditions). Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with a finite number of epimorphisms for every given domain. For every finite condition c, the conditions Msat(c) and Asat(c) of Theorems 1 and 2 are finite, respectively.

Proof. For Msat(c) we have that as $\mathscr{C}$ has a finite number of epimorphisms for a given domain, the set $\mathscr{E}$ in the construction of Msat(e, c) is finite for every epimorphism e, and the set $\mathscr{E}$ in the construction of Msat(c) is finite. For Asat(c) we have that as $\mathscr{C}$ has a finite number of epimorphisms for a given domain, the set $\mathscr{E}$ in the construction of inM_C is finite for every object $C \in \mathscr{C}$, so Asat(c) is finite. □

Every finite graph condition can be translated into a finite first-order graph formula and *vice versa*.

Fact 14 (equivalence of finite graph conditions and finite graph formulas). $\mathscr{M}$-satisfiable finite graph conditions, $\mathscr{A}$-satisfiable finite graph conditions and finite first-order graph formulas are expressively equivalent.

Proof. By Corollaries 7 and 8, it is enough to show that all involved transformations preserve finiteness. For every finite graph condition c, the graph formula Form(c) is finite since for every graph morphism, there exists a finite decomposition such that each morphism either adds a node, adds an edge, identifies two nodes or identifies two edges. For every finite first-order graph formula F, the graph condition Cond(F) is finite since the disjunctions in the case Cond($X, \exists x\ F$) are finite, as the label alphabet $\mathrm{C} = \mathrm{C_V} \cup \mathrm{C_E}$ is finite. As the category (**Graphs**, *Inj*) satisfies Assumption 2, we have by Fact 13 that for every finite graph condition c, the graph conditions Msat(c) and Asat(c) are finite. □

By the undecidability of first-order graph formulas, we get that there are no effective procedures for deciding if a given graph condition is satisfiable at all, satisfied by every graph or whether a given graph condition implies another given graph condition.

Corollary 9 (undecidability of graph conditions). The satisfiability, the tautology and the implication problems of (finite) graph conditions are undecidable.

Proof. Assume that the problems are decidable for (finite) graph conditions. By Corollary 8, we could construct for every (finite) first-order graph formula F_j ($i = 1, 2$) a (finite) graph condition $\mathrm{Cond}_{\mathscr{M}}(F_j)$ such that $G \models F_j$ if and only if $G \models \mathrm{Cond}_{\mathscr{M}}(F_j)$. Then the satisfiability problem of (finite) first-order graph formulas would be decidable, which gives a contradiction (Trakhtenbrot 1950; Courcelle 1990). □

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Symbol	Description	Reference
Msat	From $\mathscr{A}$- to $\mathscr{M}$-satisfiability	Theorem 1
Asat	From $\mathscr{M}$- to $\mathscr{A}$-satisfiability	Theorem 2
B, N	(Interface) transformation of conditions	Lemma 2
Q	From $\mathscr{A}$- to $\mathscr{M}$-matching	Theorem 3
R	From $\mathscr{M}$- to $\mathscr{A}$-matching	Theorem 4
A	From constraints to application conditions	Theorem 5
L	From right to left application conditions	Theorem 6
$C_\forall, C_\exists$	From application conditions to constraints	Theorem 7
Appl	Construction of 'gluing conditions'	Theorem 8
Cond	From formulas to graph conditions	Theorem 10
Form	From graph conditions to formulas	Theorem 11

Table 1. *Transformations of conditions and rules*

9. Conclusion

In this paper the concepts of constraints and application conditions, which were investigated in, for example, Heckel and Wagner (1995), Ehrig *et al.* (2006b) and Koch *et al.* (2005), are unified and generalised to nested conditions over graph-like structures, along the lines of Rensink (2004). As demonstrated by a case study on the handover protocol of mobile phones, graph conditions constitute a graphical and intuitive, yet precise, formalism for specifying structural properties of system states. The results of this paper are concerned with weak adhesive HLR systems with nested application conditions and are listed in Table 1. Notable results are the transformations between $\mathscr{M}$- and $\mathscr{A}$-satisfiability of conditions, Msat and Asat, as well as the transformations between $\mathscr{A}$- and $\mathscr{M}$-matching of rules, R and Q, respectively. These transformations prove that if the monomorphism class $\mathscr{M}$ is strictly closed under decomposition, then the satisfiability and matching concepts considered are expressively equivalent. Otherwise, the notions of $\mathscr{M}$-satisfiability and $\mathscr{M}$-matching are more expressive. Moreover, the transformations can be applied to switch between the user's preferred notions and the implemented notions of a rule-based transformation tool, and provide the basis for a data exchange between tools with different semantics. Finally, Msat and Asat enable the composition of complementary condition transformations in the case of $\mathscr{A}$-satisfiability, such as $L_{\mathscr{A}}$.

We have investigated a number of transformations between constraints and application conditions that can be used to develop correct transformation systems. This includes transformations: of constraints into application conditions (A); of right- into left application conditions and *vice versa* (L); and of application conditions in constraints ($C_\forall$ and $C_\exists$). Moreover, we have considered the construction of an application condition (Appl) that guarantees the applicability of the rule.

The above basic transformations enable the construction of constraint-guaranteeing and constraint-preserving application conditions with the aim of restricting the applicability of rules to yield correct transformation systems. Moreover, the basic transformations are used for the construction of weakest preconditions (Dijkstra 1976; Dijkstra and

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

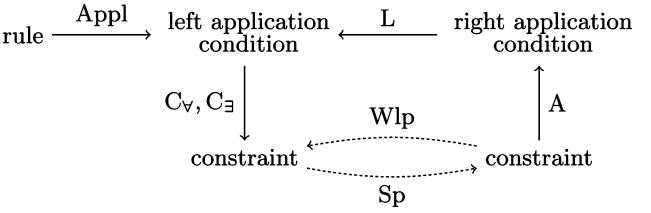


Figure 4. Basic transformations of conditions

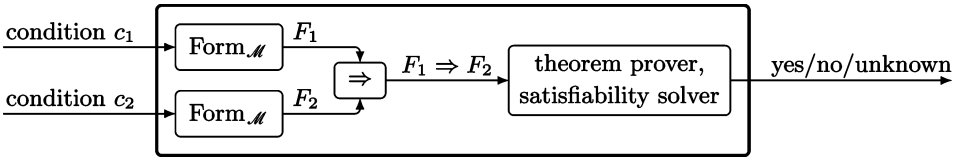


Figure 5. Semi-decider for the implication problem of $\mathscr{M}$-satisfiable graph conditions

Scholten 1989) and strongest postconditions with the aim of verifying the correctness of transformation systems – see Habel *et al.* (2006) and Azab *et al.* (2006). For instance, if a given precondition implies the weakest precondition of a program relative to a postcondition, the program under consideration is correct with respect to the pre- and postcondition (Habel *et al.* 2006).

Finally, we have investigated transformations between graph conditions and first-order graph formulas. A main result of this paper is that both concepts are expressively equivalent, that is, nested graph conditions exactly capture first-order graph properties. As a consequence, graph conditions inherit undecidability results for the satisfiability, tautology and implication problems. On the other hand, the transformation Form$_{\mathscr{M}}$ of graph conditions to graph formulas enables the use of existing first-order automated theorem provers and satisfiability solvers.

Further topics worthy of investigation include:

(1) **Generalisation of conditions.** This includes the generalisation of nested conditions to capture monadic second-order properties and a consideration of temporal logic.
(2) **Extensions of the theory.** This includes a consideration of conditions over partial and partially labelled graphs and an investigation of weakest preconditions for rules with external interface (Pennemann 2008a).
(3) **Implementation.** This includes:
   — the construction of weakest preconditions for transformation rules (*via* transformations $A, L, C_\forall$ and Appl);
   — the implementation of condition-based theorem provers (Orejas *et al.* 2008; Orejas 2008; Pennemann 2008*b*) and satisfiability solvers (Pennemann 2008a) to tackle the implication problem, and the application of existing theorem provers and satisfiability solvers (*via* Form$_{\mathscr{M}}$) as a comparison;
   — the implementation of semantic converters for rules and conditions to freely switch between $\mathscr{A}$-/$\mathscr{M}$-satisfiability and $\mathscr{A}$-/$\mathscr{M}$-matching) – see Azab *et al.* (2006) and Zuckschwerdt (2006).

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Acknowledgments

Many thanks are due to the referees for their careful reviewing of the paper and for suggesting several improvements.

References

Adámek, J., Herrlich, H. and Strecker, G. (1990) *Abstract and Concrete Categories*, John Wiley.

Azab, K. and Habel, A. (2008) High-level programs and program conditions. In: Graph Transformations (ICGT 2008). To appear in *Springer-Verlag Lecture Notes in Computer Science*.

Azab, K., Habel, A., Pennemann, K.-H. and Zuckschwerdt, C. (2006) ENFORCe: A system for ensuring formal correctness of high-level programs. In: Proc. 3rd. Int. Workshop on Graph Based Tools (GraBaTs'06). *Electronic Communications of the EASST* **1**.

Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R. and Löwe, M. (1997) Algebraic approaches to graph transformation. Part I: Basic concepts and double pushout approach. In: *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, World Scientific 163–245.

Courcelle, B. (1990) Graph rewriting: An algebraic and logical approach. In *Handbook of Theoretical Computer Science*, volume B, Elsevier 193–242.

Courcelle, B. (1997) The expression of graph properties and graph transformations in monadic second-order logic. In: *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, World Scientific 313–400.

Dijkstra, E. W. (1975) *A Discipline of Programming*, Prentice-Hall.

Dijkstra, E. W. and Scholten, C. S. (1989) *Predicate Calculus and Program Semantics*, Springer-Verlag.

Ehrig, H. (1979) Introduction to the algebraic theory of graph grammars. In: Graph-Grammars and Their Application to Computer Science and Biology. *Springer-Verlag Lecture Notes in Computer Science* **73** 1–69.

Ehrig, H., Ehrig, K., Habel, A. and Pennemann, K.-H. (2006) Theory of constraints and application conditions: From graphs to high-level structures. *Fundamenta Informaticae* **74** (1) 135–166.

Ehrig, H., Ehrig, K., Prange, U. and Taentzer, G. (2006) *Fundamentals of Algebraic Graph Transformation*, EATCS Monographs of Theoretical Computer Science, Springer-Verlag.

Ehrig, H. and Habel, A. (1986) Graph grammars with application conditions. In: Rozenberg, G. and Salomaa, A. (eds.) *The Book of L*, Springer-Verlag 87–100.

Ehrig, H., Habel, A., Kreowski, H.-J. and Parisi-Presicce, F. (1991) Parallelism and concurrency in high level replacement systems. *Mathematical Structures in Computer Science* **1** 361–404.

Ehrig, H., Habel, A., Padberg, J. and Prange, U. (2006) Adhesive high-level replacement systems: A new categorical framework for graph transformation. *Fundamenta Informaticae* **74** 1–29.

Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A. and Corradini, A. (1997) Algebraic approaches to graph transformation. Part II: Single-pushout approach and comparison with double pushout approach. In: *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, World Scientific 247–312.

European Telecommunications Standards Institute (1998) Digital cellular telecommunications system (Phase 2+); Handover procedures. *Technical Specification*, ETSI TS 100 527 V7.0.0 1999-08 (GSM 03.09 version 7.0.0 Release 1998).

Habel, A., Heckel, R. and Taentzer, G. (1996) Graph grammars with negative application conditions. *Fundamenta Informaticae* **26** 287–313.

Habel, A., Müller, J. and Plump, D. (2001) Double-pushout graph transformation revisited. *Mathematical Structures in Computer Science* **11** (5) 637–688.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press

Habel, A. and Pennemann, K.-H. (2005) Nested constraints and application conditions for high-level structures. In: Formal Methods in Software and System Modeling. *Springer-Verlag Lecture Notes in Computer Science* **3393** 293–308.

Habel, A. and Pennemann, K.-H. (2006) Satisfiability of high-level conditions. In: Graph Transformations (ICGT 2006). *Springer-Verlag Lecture Notes in Computer Science* **4178** 430–444.

Habel, A., Pennemann, K.-H. and Rensink, A. (2006) Weakest preconditions for high-level programs. In: Graph Transformations (ICGT 2006). *Springer-Verlag Lecture Notes in Computer Science* **4178** 445–460.

Heckel, R. and Wagner, A. (1995) Ensuring consistency of conditional graph grammars – a constructive approach. In: Proc. Workshop on Graph Rewriting and Computation (SEGRAGRA'95). *Electronic Notes in Theoretical Computer Science* **2** 95–104.

Koch, M., Mancini, L. V. and Parisi-Presicce, F. (2005) Graph-based specification of access control policies. *Journal of Computer and System Sciences* **71** 1–33.

Koch, M. and Parisi-Presicce, F. (2002) Describing policies with graph constraints and rules. In: Graph Transformation (ICGT 2002). *Springer-Verlag Lecture Notes in Computer Science* **2505** 223–238.

Lack, S. and Sobociński, P. (2004) Adhesive categories. In: Proc. of Foundations of Software Science and Computation Structures (FOSSACS'04). *Springer-Verlag Lecture Notes in Computer Science* **2987** 273–288.

Löwe, M. (1993) Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science* **109** 181–224.

Orejas, F. (2008) Attributed graph constraints. In: Graph Transformations (ICGT'08). To appear in *Springer-Verlag Lecture Notes in Computer Science*.

Orejas, F., Ehrig, H. and Prange, U. (2008) A logic of graph constraints. In: Proc. Fundamental Approaches to Software Engineering (FASE'08). *Springer-Verlag Lecture Notes in Computer Science* **4961** 179–19.

Pennemann, K.-H. (2008a) An algorithm for approximating the satisfiability problem of high-level conditions. In: Proc. Graph Transformation for Verification and Concurrency (GT-VC'07). *Electronic Notes in Theoretical Computer Science* **213** 75–94.

Pennemann, K.-H. (2008b) Resolution-like theorem proving for high-level conditions. In: Graph Transformations (ICGT'08). To appear in *Springer-Verlag Lecture Notes in Computer Science*.

Plump, D. and Steinert, S. (2004) Towards graph programs for graph algorithms. In: Graph Transformations (ICGT'04). *Springer-Verlag Lecture Notes in Computer Science* **3256** 128–143.

Rensink, A. (2004) Representing first-order logic by graphs. In: Graph Transformations (ICGT'04). *Springer-Verlag Lecture Notes in Computer Science* **3256** 319–335.

Trakhtenbrot, B. A. (1950) The impossibility of an algorithm for the decision problem on finite classes (in Russian). *Doklady Akademii Nauk SSSR* **70** 569–572. (English translation in: Nine Papers on Logic and Quantum Electrodynamics (1963) *AMS Transl. Ser. 2* **23** 1–5.)

Zuckschwerdt, C. (2006) Ein System zur Transformation von Konsistenz in Anwendungsbedingungen (in German). Berichte aus dem Department für Informatik, Universität Oldenburg 11/06, 114 pages.

https://doi.org/10.1017/S0960129508007202 Published online by Cambridge University Press