Constraints and Application Conditions: From Graphs to High-Level Structures

Hartmut Ehrig[1], Karsten Ehrig[1], Annegret Habel[2], and Karl-Heinz Pennemann[2]

[1] Technische Universität Berlin, Germany
{ehrig,karstene}@cs.tu-berlin.de
[2] Carl v. Ossietzky Universität Oldenburg, Germany
{habel,k.h.pennemann}@informatik.uni-oldenburg.de

Abstract. Graph constraints and application conditions are most important for graph grammars and transformation systems in a large variety of application areas. Although different approaches have been presented in the literature already there is no adequate theory up to now which can be applied to different kinds of graphs and high-level structures. In this paper, we introduce an improved notion of graph constraints and application conditions and show under what conditions the basic results can be extended from graph transformation to high-level replacement systems. In fact, we use the new framework of adhesive HLR categories recently introduced as combination of HLR systems and adhesive categories. Our main results are the transformation of graph constraints into right application conditions and the transformation from right to left application conditions in this new framework.

1 Introduction

According to the requirements of several application areas the rules of a graph grammar have been equipped in [4] by a very general notion of application conditions. In a subsequent paper [8], the notion of application conditions is restricted to contextual conditions like the existence or non-existence of certain nodes and edges or certain subgraphs in the given graph. In [9], the authors introduce graphical consistency constraints, also called graph constraints, that express very basic conditions on graphs as e.g. the existence or uniqueness of certain nodes and edges in a graphical way.

Basic results for graph constraints and application conditions have been shown in [9, 10] first for the single and later in the double pushout approach for different kinds of graphs. Unfortunately there is no adequate theory up to now which can be applied not only to graphs but also to high-level structures in the sense of [5].

A new version of high-level replacement systems, called adhesive HLR systems, has been introduced in [6] combining HLR systems in the sense of [5] and adhesive categories (see [11]). This new framework has been used not only to reformulate

the basic results like local Church Rosser, Parallelism and Concurrency Theorem from [5], but also to present an improved version of the Embedding Theorem [3] and the local Confluence Theorem, known as Critical Pair Lemma [12]. Moreover it can be applied to all kinds of graphs and Petri nets satisfying the HLR1 and HLR2 conditions in [5] and also to typed attributed graphs in [7].

In this paper we use adhesive HLR categories and systems to improve and generalize the basic notions and results for constraints and application conditions from graphs to high-level structures. For this purpose we present an improved notion of graph constraints, based on positive and negative atomic constraints, and of application conditions, based on atomic conditional conditions. In our main theorems we show how to transform constraints into right application conditions, and right into left application conditions in the framework of adhesive HLR systems. As additional condition we only need finite coproducts and a suitable E-M-factorization which is valid in all our example categories.

The paper is organized as follows. In section 2 we present our improved notions of graph constraints and application conditions. In section 3 we give a short introduction of adhesive HLR categories together with some basic properties. Then we generalize graph constraints and application conditions to the framework of adhesive HLR categories. In section 4, we present the main results for graphs and high-level structures and give several illustrating examples for graphs and place transition nets. A conclusion including further work is given in section 5.

2 Constraints and application conditions for graphs

In the following, we assume that the reader is familiar with the notions of graphs and graph morphisms, see e.g. [3, 2]. Graph constraints, first investigated by [9], allow to express basic conditions on graphs as e.g. the existence or uniqueness of certain nodes and edges in a graphical way.

Definition 1 (graph constraint). An *atomic graph constraint* is of the form $\mathrm{PC}(a)$ or $\mathrm{NC}(a)$ where $a\colon P \to C$ is an arbitrary graph morphism. It is said to be a *positive* or *negative* atomic graph constraint, respectively. A *graph constraint* is a Boolean formula over atomic graph constraints, i.e. every atomic graph constraint is a graph constraint and, for every graph constraint c, $\neg c$ is a graph constraint and, for every index set I and every family $(c_i)_{i\in I}$ of graph constraints, $\wedge_{i\in I} c_i$ and $\vee_{i\in I} c_i$ are graph constraints. A graph G *satisfies* $\mathrm{PC}(a)$ ($\mathrm{NC}(a)$), written $G \models \mathrm{PC}(a)$ ($\mathrm{NC}(a)$), if for every injective morphism $p\colon P \to G$ there exists (does not exist) an injective morphism $q\colon C \to G$ such that $q \circ a = p$.

Fig. 1. Satisfiability of atomic constraints

A graph G *satisfies* a graph constraint $\neg c$, written $G \models \neg c$, if and only if G does not satisfy the graph constraint c. G *satisfies* $\wedge_{i\in I} c_i$ ($\vee_{i\in I} c_i$), written $G \models \wedge_{i\in I} c_i$ ($\vee_{i\in I} c_i$), if and only if G satisfies all (some) graph constraints c_i with $i \in I$.

Example 1. Examples of graph constraints:

$\mathrm{PC}(\bigcirc\,\bigcirc \to \bigcirc)$	There exists at most one node.
$\mathrm{PC}(\bigcirc \to \circlearrowright)$	Every node has a loop.
$\mathrm{NC}(\bigcirc \to \circlearrowright)$	The graph is loop-free.
$\neg\mathrm{PC}(\bigcirc \to \circlearrowright)$	There exists a node without loop.
$\mathrm{PC}(\emptyset \to \circlearrowright)$	There exists a node with a loop.
$\mathrm{NC}(\emptyset \to \circlearrowright)$	There exists no node with a loop.
$\bigwedge_{k=1}^{\infty} \mathrm{NC}(\emptyset \to \mathrm{C}_k)$	The graph is acyclic (C_k denotes a cycle of length k).

Remark. The definition of graph constraints generalizes the one in [9], because we allow negative atomic constraints and non-injective a.

Fact . If a is non-injective and $G \models \mathrm{PC}(a)$, then there is no injective $p\colon P \to G$.

Proof. Assume there is an injective $p\colon P \to G$. Then $G \models \mathrm{PC}(a)$ implies the existence of an injective $q\colon C \to G$ with $q \circ a = p$. This implies a injective (contradiction).

Fact . If a is non-injective, then $G \models \mathrm{NC}(a)$.

Proof. Assume $G \not\models \mathrm{NC}(a)$. Then there exist injective $p\colon P \to G$ and $q\colon C \to G$ with $q \circ a = p$. The injectivity of p implies the injectivity of a (contradiction).

Application conditions for graph replacement rules were first introduced in [4]. In a subsequent paper [8], a special kind of application conditions were considered which can be represented in a graphical way. In particular, contextual conditions like the existence or non-existence of certain nodes and edges or certain subgraphs in the given graph can be expressed. In [9] so-called conditional application conditions were introduced.

Definition 2 (application condition over a graph). An *(conditional) atomic application condition* over a graph L is of the form $\mathrm{P}(x, \vee_{i\in I} x_i)$ or $\mathrm{N}(x, \wedge_{i\in I} x_i)$ where $x\colon L \to X$ is an arbitrary graph morphism and $x_i\colon X \to C_i$ with $i \in I$ are injective graph morphisms. It is said to be a *positive* or *negative* atomic application condition, respectively. An *application condition* over L is a Boolean formula over atomic application conditions over L, i.e. every atomic application condition is an application condition and, for every application condition acc, $\neg$ acc is an application condition and, for every index set I and every family $(\mathrm{acc}_i)_{i\in I}$ of application conditions, $\wedge_{i\in I}\ \mathrm{acc}_i$ and $\vee_{i\in I}\ \mathrm{acc}_i$ are application conditions.

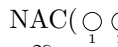
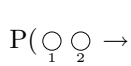
Fig. 2. Satisfiability of atomic application conditions

A match $m\colon L \to G$ *satisfies* $\mathrm{acc}_L = \mathrm{P}(x, \vee_{i\in I} x_i)$ $(\mathrm{N}(x, \wedge_{i\in I} x_i))$, written $m \models \mathrm{acc}_L$, if for all injective morphisms $p\colon X \to G$ with $p \circ x = m$ there exists (does not exist) $i \in I$ and an injective morphism $q_i\colon C_i \to G$ with $q_i \circ x_i = p$.

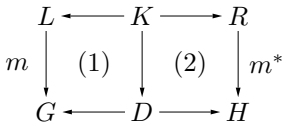
A match $m\colon L \to G$ *satisfies* an application condition of the form $\neg\ \mathrm{acc}$, written $m \models \neg\ \mathrm{acc}$, if and only if m does not satisfy the application condition acc. A match m *satisfies* $\wedge_{i\in I}\ \mathrm{acc}_i$ $(\vee_{i\in I}\ \mathrm{acc}_i)$, written $m \models \wedge_{i\in I}\ \mathrm{acc}_i$ $(\vee_{i\in I}\ \mathrm{acc}_i)$, if and only if m satisfies all (some) acc_i with $i \in I$.

Remark. The definition of an application condition slightly generalizes the ones in [8, 9]. Let us consider the well-known negative application condition $\mathrm{NAC}(x)$, where $x\colon L \to X$ is a graph morphism. A match $m\colon L \to G$ *satisfies* $\mathrm{NAC}(x)$, written $m \models \mathrm{NAC}(x)$, if there does not exist an injective morphism $p\colon X \to G$ with $p \circ x = m$. $\mathrm{NAC}(x)$ is equivalent to $\mathrm{P}(x, \vee_{i\in I} x_i)$ for $I = \emptyset$ and hence a special case of positive atomic application conditions.

Example 2. Examples of application conditions and their meaning for an injective match $m\colon L \to G$:

$\mathrm{NAC}($  $)$	There is no edge from node $m(1)$ to node $m(2)$.
$\bigwedge_{k=1}^{\infty} \mathrm{NAC}($  $)$	There is no path connecting node $m(1)$ and $m(2)$. (P_k denotes a path of length k)
$\mathrm{P}($  $)$	If there is an edge from $m(1)$ to $m(2)$, then there also is an edge from $m(2)$ to $m(1)$.

A *rule* $p = \langle L \leftarrow K \rightarrow R \rangle$ consists of two injective graph morphisms with a common domain K. Given a rule p and a graph morphism $K \to D$, a *direct derivation* consists of two pushouts (1) and (2). We write $G \Rightarrow_{p,m,m^*} H$ and say that $m\colon L \to G$ is the match and $m^*\colon R \to H$ is the comatch of p in H.



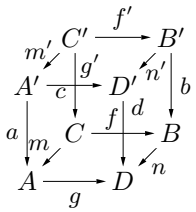
Definition 3 (application condition for a rule). An *application condition* $A(p) = (A_L, A_R)$ for a rule $p = \langle L \leftarrow K \rightarrow R \rangle$ consists of a left application condition A_L over L and a right application condition A_R over R. A *direct derivation* $G \Rightarrow_{p,m,m^*} H$ satisfies an application condition $A(p) = (A_L, A_R)$, if

$$m \models A_L \quad \text{and} \quad m^* \models A_R.$$

3 Constraints and application conditions for high-level structures

The main idea of high-level replacement systems is to generalize the concepts of graph replacement from graphs to all kinds of structures which are of interest in Computer Science and Mathematics. In the following, we will consider constraints and application conditions in adhesive HLR-categories (see [6]) and prove our transformation results on this general level.

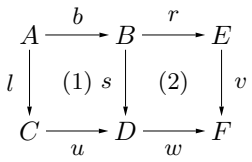
Definition 4 (adhesive HLR-category). A category $\mathbf{C}$ with a morphism class M is called *adhesive HLR category*, if 1) M is a class of monomorphisms closed under compositions and decompositions ($g \circ f \in M$, $g \in M$ implies $f \in M$), 2) $\mathbf{C}$ has pushouts and pullbacks along M-morphisms, i.e. pushouts and pullbacks, where at least one of the given morphisms is in M, and M-morphisms are closed under pushouts and pullbacks, and 3) pushouts in $\mathbf{C}$ along M-morphisms are VK-squares, i.e. for any commutative cube in $\mathbf{C}$ where we have the pushout with $m \in M$ in the bottom and the back faces are pullbacks, it holds: the top is pushout $\Leftrightarrow$ the front faces are pullbacks.



Example 3. All examples of adhesive categories defined in [11] are adhesive HLR categories for the class M of all monomorphisms. As shown in [11] this includes the categories **Sets** of sets, **Graphs** of graphs and several variants of graphs like typed, labelled and hypergraphs. Moreover this includes the category **PT-Net** of place transition nets considered in [5] already. The following categories are important examples of adhesive HLR categories where M is not the class of all monomorphisms: the category $\langle \mathbf{AGraphs_{ATG}}, M \rangle$ of typed attributed graphs with type graph ATG and class M of all injective morphisms with isomorphism on the data part is (see [7]), the category $\langle \mathbf{AHL\text{-}Net}, M \rangle$ of algebraic high level nets with class M of all strict injective net morphisms, and the category $\langle \mathbf{Spec}, M \rangle$ of algebraic specifications with class M of all strict injective specification morphisms [5].

Fact (HLR properties of adhesive HLR categories). Given an adhesive HLR-category $\langle \mathbf{C}, M \rangle$, the following HLR conditions are satisfied.

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



3. Uniqueness of pushout complements for M-morphisms: Given $b: A \to B$ in M and $s: B \to D$ then there is up to isomorphism at most one C with $l: A \to C$ and $u: C \to D$ such that diagram (1) is a pushout.

Proof. See [6, 11].

General assumption. In the following, we assume that $\langle \mathbf{C}, M \rangle$ is an adhesive HLR category with binary coproducts and epi-M-factorizations, that is, for every morphism there is an epi-mono-factorization with monomorphism in M.

We will consider structural constraints and application conditions in our general framework. Structural constraints, short constraints, correspond to graph constraints in section 2, but not necessarily to logical constraints defined by predicate logic.

Definition 5 (constraints). An *atomic constraint* is of the form $\mathrm{PC}(a)$ or $\mathrm{NC}(a)$ where $a: P \to C$ is an arbitrary morphism. $\mathrm{PC}(a)$ is said to be *positive* and $\mathrm{NC}(a)$ *negative*. A *constraint* is a Boolean formula over atomic constraints. An object G *satisfies* $\mathrm{PC}(a)$ ($\mathrm{NC}(a)$), written $G \models \mathrm{PC}(a)$ ($\mathrm{NC}(a)$), if for every morphism $p: P \to G$ in M there exists (does not exist) a morphism $q: C \to G$ in M such that $q \circ a = p$ (see figure 1). Satisfiability of arbitrary constraints is defined in the usual way (see definition 1).

Definition 6 (application condition over an object). An *atomic application condition* over an object L is of the form $\mathrm{P}(x, \vee_{i \in I} x_i)$ or $\mathrm{N}(x, \wedge_{i \in I} x_i)$ where $x: L \to X$ is an arbitrary morphism and $x_i: X \to C_i$ with $i \in I$ are morphisms in M. It is said to be a *positive* or *negative* atomic application condition, respectively. An *application condition* over L is a Boolean formula over atomic application conditions over L. A match $m: L \to G$ *satisfies* $\mathrm{acc}_L = \mathrm{P}(x, \vee_{i \in I} x_i)$ ($\mathrm{N}(x, \wedge_{i \in I} x_i)$), written $m \models \mathrm{acc}_L$, if for all morphisms $p: X \to G$ in M with $p \circ x = m$ there exists (does not exist) $i \in I$ and a morphism $q_i: C_i \to G$ in M with $q_i \circ x_i = p$ (see figure 2). Satisfiability of arbitrary application conditions is defined in the usual way (see definition 2).

The special case of negative atomic application conditions $\mathrm{NAC}(x)$ and general application conditions for rules (see definition 3) are defined as in the graph case.

General Remark. In the case $I = \emptyset$ we have $\mathrm{P}(x, \vee_{i \in I} x_i) \equiv \mathrm{NAC}(x)$, where $m \models \mathrm{NAC}(x)$ means that there is no $p \in M$ with $p \circ x = m$. Moreover we have $\mathrm{N}(x, \wedge_{i \in I} x_i) \equiv \text{true}$ for $I = \emptyset$, because $m^* \models N(x, \wedge_{i \in \emptyset} x_i) \Leftrightarrow \forall p (p \in M \wedge p \circ x = m^* \Rightarrow \neg(\exists i \in I = \emptyset ...)) \Leftrightarrow \forall p \text{ true} \Leftrightarrow \text{true}$.

4 Main results for graphs and high-level structures

In the following, we will show that arbitrary constraints can be transformed into right application conditions and that right application conditions can be transformed in left application conditions. We first show that positive and negative atomic constraints can be transformed into right application conditions.

Lemma 1 (transformation of positive atomic constraints into right application conditions). *Given a positive atomic constraint* $\mathrm{PC}(a)$ *with* $a\colon P \to C$ *and comatch* $m^*\colon R \to H$*. Then there is a right application condition* $T(\mathrm{PC}(a))$ *such that* $m^* \models T(\mathrm{PC}(a)) \Leftrightarrow H \models \mathrm{PC}(a)$*.*

Construction. Let $T(\mathrm{PC}(a))$ be the right application condition

$$T(\mathrm{PC}(a)) = \wedge_S \mathrm{P}(R \xrightarrow{s} S, \vee_{i\in I}(S \overset{t_i \circ t}{\rightarrow} T_i))$$

1. The conjunction $\wedge_S$ ranges over all "gluings" S of R and P in figure 3(a). More precisely over all triples $\langle S, s, p\rangle$ with arbitrary $s\colon R \to S$ and $p\colon P \to S$ in M such that the pair $\langle s, p\rangle$ is jointly epimorphic. For each such triple $\langle S, s, p\rangle$ we construct the pushout (1) of p and a leading to $t\colon S \to T$ and $q\colon C \to T$.
2. The disjunction $\vee_{i\in I}$ ranges over all $S \overset{t_i \circ t}{\rightarrow} T_i$ with epimorphism t_i such that $t_i \circ t$ and $t_i \circ q$ are in M. For $I = \emptyset$ we have $T(\mathrm{PC}(a)) = \wedge_S \mathrm{NAC}(R \xrightarrow{s} S)$.

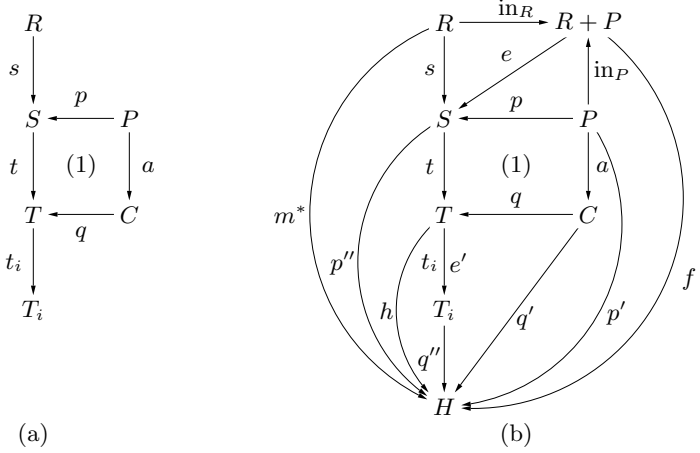
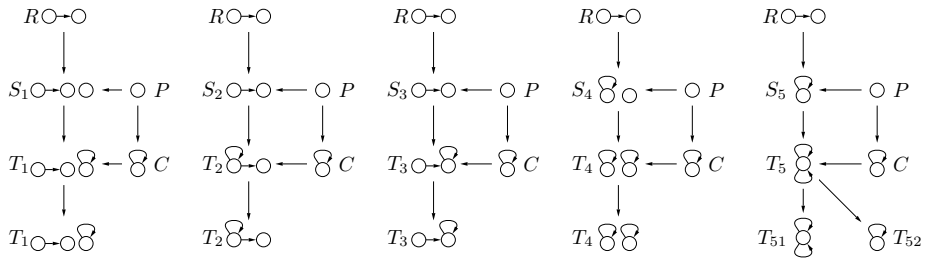


Fig. 3. Construction of $T(\mathrm{PC}(a))$/Correspondence of $T(\mathrm{PC}(a))$ and $\mathrm{PC}(a)$

Proof. See appendix A.

Example 4. Consider the positive atomic graph constraint $\mathrm{PC}(\bigcirc \to \bigcirc\!\!\supset)$ (see example 1) and the rule $p = \langle \bigcirc\,\bigcirc \leftarrow \bigcirc\,\bigcirc \to \bigcirc\!\!\rightarrow\!\!\bigcirc \rangle$. According to the construction in lemma 1 the graph constraint can be transformed into the following conjunction of right positiv atomic application conditions $\wedge_{i=1}^{4}\mathrm{P}(\bigcirc\!\!\rightarrow\!\!\bigcirc \to S_i$, $S_i \to T_i) \wedge \mathrm{P}(\bigcirc\!\!\rightarrow\!\!\bigcirc \to S_5, \vee_{j=1}^{2} S_5 \to T_{5j})$ with S_i, T_i, T_{5j} as shown below. The condition expresses the positiv atomic application condition "Every node outside (see T_1, T_4) and inside (see T_2, T_3, T_{51}, T_{52}) the comatch must have a loop.", where S_1, S_2, S_3 correspond to injective and S_4, S_5 to non-injective comatches. Altogether this condition means that for each comatch $m^* : R \to H$ each node of H must have a loop, which is equivalent to $H \models \mathrm{PC}(\bigcirc \to \bigcirc\!\!\supset)$.



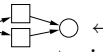
Lemma 2 (transformation of negative atomic constraints into application conditions). *Given a negative constraint* $\mathrm{NC}(a)$ *with* $a\colon P \to C$ *and comatch* $m^*\colon R \to H$. *Then there is an application condition* $T(\mathrm{NC}(a))$ *such that* $m^* \models T(\mathrm{NC}(a)) \Leftrightarrow H \models \mathrm{NC}(a)$.

Construction. Let $T(\mathrm{NC}(a))$ be the following right application condition

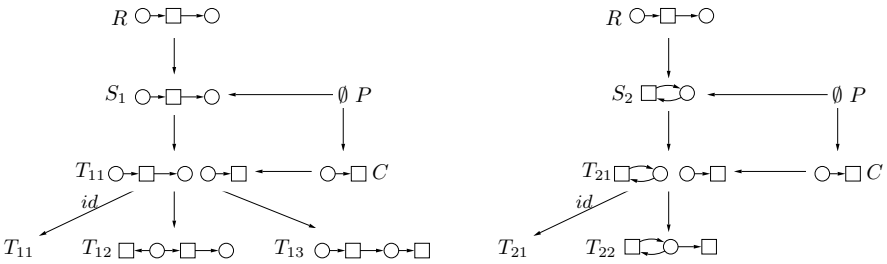
$$T(\mathrm{NC}(a)) = \wedge_S \mathrm{N}(R \xrightarrow{s} S, \wedge_{i\in I}(S \xrightarrow{t_i \circ t} T_i)),$$

where the morphisms $s\colon R \to S$ and $t_i \circ t\colon S \to T_i$ are the same as in the construction of lemma 1 and $m^* \models T(\mathrm{NC}(a))$ is now defined by: For all $\langle S, s, p\rangle$ as given in the construction, all $p''\colon S \to H$ in M with $p'' \circ s = m^*$ there is *no* $i \in I$ and $q''\colon T_i \to H$ in M with $q'' \circ t_i \circ t = p''$. For $I = \emptyset$ we have, according to the general remark after definition 6, $m^* \models T(\mathrm{NC}(a)) \Leftrightarrow$ true.

Proof. See appendix A.

Example 5. According to example 3 we now consider place transition nets. Consider the negative atomic net constraint $\mathrm{NC}(\emptyset \to \bigcirc\!\!\rightarrow\!\!\square)$. H satisfies this constraint if H contains no subnet of the form $\bigcirc\!\!\rightarrow\!\!\square$, where we call such a place a "sink place". Consider the rule $p = \langle$  $\leftarrow \bigcirc\,\bigcirc \to \bigcirc\!\!\rightarrow\!\!\square\!\!\rightarrow\!\!\bigcirc \rangle$. According to the construction in lemma 2 the constraint can be transformed into the application condition $\mathrm{N}(R \to S_1, \wedge_{i=1}^{3} S_1 \to T_{1i}) \wedge \mathrm{N}(R \to S_2, \wedge_{i=1}^{2} S_2 \to T_{2i})$

where R, S_1, S_2, T_{ij} are given below. The condition means "No sink place is allowed to be outside or inside the comatch, e.g. no sink place is allowed in H.", where S_1 takes care of injective and S_2 of non-injective comatches $m^*: R \to H$.



The transformation in lemma 1 and 2 can be extended to arbitrary constraints.

Theorem 1 (transformation of constraints into application conditions). *Given a constraint c and a comatch $m^*: R \to H$. Then there is an application condition $T(c)$ such that $m^* \models T(c) \Leftrightarrow H \models c$.*

Proof. For atomic constraints, the transformation is given in the proof of lemma 1 and 2, respectively. For arbitrary constraints, the transformation is inductively defined as follows: $T(\neg c) = \neg T(c)$, $T(\wedge_{i\in I} c_i) = \wedge_{i\in I} T(c_i)$ and $T(\vee_{i\in I} c_i) = \vee_{i\in I} T(c_i)$. Now the proof of the statement is straightforward.

In the following, we will show that arbitrary right application conditions can be transformed into left application conditions. For this purpose, we first show that right positive and then right negative atomic application conditions can be transformed into corresponding left atomic application conditions.

Lemma 3 (transformation from right positive atomic to left positive application conditions). *Given a rule $p = \langle L \leftarrow K \rightarrow R \rangle$ and a right positive atomic application condition acc_R then there is a left positive atomic application condition acc_L such that for all direct derivations $G \Rightarrow_{p,m,m^*} H$ we have:*

$$m \models \mathrm{acc}_L \Leftrightarrow m^* \models \mathrm{acc}_R.$$

Construction. Let $\mathrm{acc}_R = \mathrm{P}(R \xrightarrow{x} X, \vee_{i\in I}(X \xrightarrow{x_i} C_i))$ be a right positive atomic application condition in figure 4. Then we construct a left positive atomic application condition $\mathrm{acc}_L = p^{-1}(\mathrm{acc}_R) = \mathrm{P}(L \xrightarrow{y} Y, \vee_{i\in I'}(Y \xrightarrow{y_i} D_i))$ with $I' \subseteq I$ or $p^{-1}(\mathrm{acc}_R) = \mathrm{true}$ as follows:

1. If the pair $\langle r: K \to R, x: R \to X \rangle$ has a pushout complement, define $y: L \to Y$ by two pushouts (1) and (2), otherwise $p^{-1}(\mathrm{acc}_R) = \mathrm{true}$.
2. For each $i \in I$, if the pair $\langle r^*: Z \to X, x_i: X \to C_i \rangle$ has a pushout complement, then $i \in I'$ and $y_i: Y \to D_i$ is defined by two pushouts (3) and (4), otherwise $i \notin I'$. Since pushout complements of M-morphisms (if they exist) are unique, the construction yields a unique result up to isomorphism.

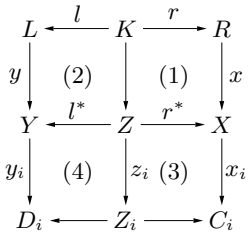
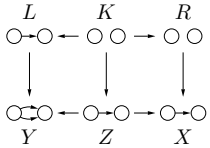


Fig. 4. Transformation of application conditions

Proof. See appendix A.

Example 6. Consider the rule $p = \langle \bigcirc_1 \to \bigcirc_2 \leftarrow \bigcirc_1\ \bigcirc_2 \rightarrow \bigcirc_1\ \bigcirc_2 \rangle$ and – according to the remark after definition 2 – the right application condition $\mathrm{acc}_R = \mathrm{NAC}(\bigcirc_1\ \bigcirc_2 \rightarrow \bigcirc_1 \to \bigcirc_2)$ (see example 2), meaning that an edge between the nodes in the comatch must not exist. According to the construction of lemma 3 with $I = \emptyset$ the right atomic application condition is transformed into the left atomic application condition $\mathrm{acc}_L = \mathrm{NAC}(\bigcirc_1 \to \bigcirc_2 \rightarrow \bigcirc_1 \rightrightarrows \bigcirc_2)$ with $I = \emptyset$ meaning that two parallel edges between the nodes in the match must not exist.



Remark. 1. Dually we can construct from acc_L a right atomic application condition $\mathrm{acc}_R = p(\mathrm{acc}_L)$ such that $m \models \mathrm{acc}_L \Leftrightarrow m^* \models p(\mathrm{acc}_L)$.
2. For $I = \emptyset$, acc_R is a negative atomic application condition, i.e. $\mathrm{acc}_R \Leftrightarrow \mathrm{NAC}(x)$. In this case $p^{-1}(\mathrm{acc}_R)$ is either true or also a negative atomic application condition, i.e. $p^{-1}(\mathrm{acc}_R) \Leftrightarrow \mathrm{NAC}(y)$. For $I \neq \emptyset$, acc_R is a "real" atomic application condition and $p^{-1}(\mathrm{acc}_R)$ may be either true, a negative atomic application condition (if $I' = \emptyset$) or also a "real" atomic application condition.
3. Since x_i ($i \in I$) and also r and r^* are in M, also z_i and y_i are in M in (3) and (4) respectively (M-morphisms are closed under pushouts and pullbacks).

Lemma 4 (transformation from right negative atomic to left negative application conditions). *Given a rule $p = \langle L \leftarrow K \rightarrow R \rangle$ and a right negative atomic application condition acc_R then there is a left negative atomic application condition acc_L such that for all direct derivations $G \Rightarrow_{p,m,m^*} H$ we have: $m \models \mathrm{acc}_L \Leftrightarrow m^* \models \mathrm{acc}_R$.*

Construction. Let $\mathrm{acc}_R = \mathrm{N}(R \xrightarrow{x} X, \wedge_{i \in I}(X \xrightarrow{x_i} C_i))$ be a right negative atomic application condition. Then we construct a left negative atomic application condition $\mathrm{acc}_L = p^{-1}(\mathrm{acc}_R)$ as follows:

1. If $I = \emptyset$ then $\mathrm{acc}_R = \mathrm{true}$ and we define $\mathrm{acc}_L = \mathrm{true}$.

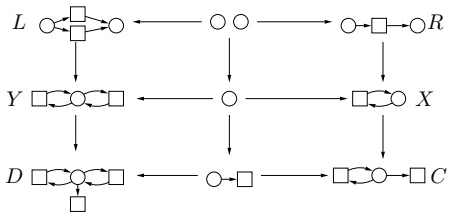
2. If $I \neq \emptyset$ and $\langle r, x \rangle$ has no pushout complement then $\text{acc}_L = \text{true}$.
3. If $I \neq \emptyset$ and $\langle r, x \rangle$ has a pushout complement then define $y: L \to Y$ by two pushouts (1) and (2) in figure 4. Moreover for each $i \in I$ if $\langle r^*, x_i \rangle$ has a pushout complement then $i \in I'$ and $y_i: Y \to D_i$ is defined by pushouts (3) and (4) in figure 4, otherwise $i \notin I'$. Now define

$$\text{acc}_L = p^{-1}(\text{acc}_R) = \text{N}(L \xrightarrow{y} Y, \wedge_{i \in I'}(Y \xrightarrow{y_i} D_i))$$

where $\text{acc}_L = \text{true}$ in the case $I' = \emptyset$.

Proof. See appendix A.

Example 7. Consider the same rule p as in example 5 and the following right negative atomic application condition $\text{acc}_R = \text{N}(R \to X, X \to C)$ which corresponds to $\text{N}(R \to S_2, S_2 \to T_{22})$ in example 5. $H \models acc_R$ means that for each non-injective comatch $m^* : R \to H$ the place in $m^*(R)$ must not be a sink place. According to the general construction in figure 4 we obtain the following left negative atomic application condition $\text{acc}_L = \text{N}(L \to Y, Y \to D)$. $G \models \text{acc}_L$ means that for each non-injective match $m : L \to G$ the place in $m(L)$ must not be a sink place. Note that a non-injective match $m : L \to G$ can only identify the places, because otherwise the gluing condition would be violated.



The transformation in lemma 3 and 4 can be extended to arbitrary right application conditions.

Theorem 2 (transformation from right to left application conditions). *Given a rule $p = \langle L \leftarrow K \to R \rangle$ and right application condition acc_R then there is a left application condition acc_L such that for all direct derivations $G \Rightarrow_{p,m,m^*} H$ we have: $m \models \text{acc}_L \Leftrightarrow m^* \models \text{acc}_R$.*

Proof. For right atomic application conditions, the transformation is defined as in the proof of lemma 3 and 4, respectively. For arbitrary right application conditions, the transformation is defined as follows: $p^{-1}(\neg\, \text{acc}_R) = \neg p^{-1}(\text{acc}_R)$, $p^{-1}(\wedge_{i \in I}\, \text{acc}_{iR}) = \wedge_{i \in I} p^{-1}(\text{acc}_{iR})$, and $p^{-1}(\vee_{i \in I}\, \text{acc}_{iR}) = \vee_{i \in I} p^{-1}(\text{acc}_{iR})$. Now the proof of the statement is straightforward.

5 Conclusion

In the present paper we have introduced a general notion of constraints and application conditions that is more expressive than previous ones in the graph

case and is formulated now for high-level structures in the new framework of adhesive HLR categories (see [6]). It is shown that constraints can be transformed into right application conditions for rules and that right application conditions can be transformed into left ones. As a consequence, we have a mechanism to integrate constraints into rules and to ensure that the constraints remain satisfied.

Further topics could be the followings.

(1) Extension of the notions of constraints and application conditions: Although the constraints are more general than the ones in [9], there are constraints which cannot be expressed up to now. E.g. the constraints like "Every node has an outgoing or incoming edge" and "There exists a node such that all outgoing edges are labelled by a" could be expressed if one would extend the concepts by alternative or conditional atomic graph constraints and existential satisfaction of graph constraints. Also the extension of constraints from statical (propositional logic) to dynamical (temporal logic) constraints [10] and a transformation between logical and graphical constraints (e.g. OCL-Constraints [1]) is interesting.

(2) Extensions of the theory: In [8], the local Church Rosser theorems I and II are proved for single-pushout rules with negative atomic application conditions. These results are also valid for the double-pushout rules with arbitrary application conditions in high-level structures provided that the notion of independence is extended such that the induced matches satisfy the corresponding application conditions. Moreover, it would be important to generalize the results in [6] to rules with application conditions.

(3) Applications of the theory: The theory can be applied already not only to graph transformations over labelled graphs (see [3, 2]) but also to several variants of graphs like typed attributed graphs and hypergraphs and also to Petri nets (see [5, 6]) and examples 4-7. For building up Petri nets satisfying some net constraints, one could integrate the constraints as application conditions into the rules. Another important application of adhesive HLR categories and corresponding systems is typed attributed graph transformation as presented in [7], where also a slight extension of the general assumption in section 3 seems to be useful.

A Appendix

In this appendix we present the proofs of all lemmata given in section 4.

Proof of Lemma 1. 1. Let $m^* \models T(\mathrm{PC}(a))$. We have to show $H \models \mathrm{PC}(a)$, i.e. for all morphisms $p' \colon P \to H$ in M there is a morphism $q' \colon C \to H$ in M with $q' \circ a = p'$. Given a morphism $p' \colon P \to H$ in M and a comatch $m^* \colon R \to H$, we construct the coproduct $R + P$ with injections in_R and in_P in figure 3(b). By the universal property of coproducts, there is a unique morphism $f \colon R + P \to H$ with $f \circ \mathrm{in}_R = m^*$ and $f \circ \mathrm{in}_P = p'$. Now let $f = p'' \circ e$ be an epi-mono factorization of f with epimorphism e and monomorphism p'' in M, and define $s = e \circ \mathrm{in}_R$

and $p = e \circ \text{in}_P$. Then the pair $\langle s, p \rangle$ is jointly epimorphic, because e is an epimorphism, and p is in M, because $p'' \circ p = p'' \circ e \circ \text{in}_P = f \circ \text{in}_P = p'$ is in M (M-morphisms are closed under decomposition). Hence $\langle S, s, p \rangle$ belongs to the conjunction $\wedge_S$ of $T(\text{PC}(a))$. Moreover we have $p'' \circ s = p'' \circ e \circ \text{in}_R = f \circ \text{in}_R = m^*$ with monomorphism p'' in M.

In the case $I \neq \emptyset$, $m^* \models T(\text{PC}(a))$ implies existence of $i \in I$ and $q'' \colon T_i \to H$ in M with $q'' \circ t_i \circ t = p''$. Now let $q' = q'' \circ t_i \circ q$ then q' is in M, because q'' is in M by construction and $t_i \circ q$ is in M by step 2 in the construction (M-morphisms are closed under decompositions). Finally we have $H \models \text{PC}(a)$, because $q' \circ a = q'' \circ t_i \circ q \circ a = q'' \circ t_i \circ t \circ p = p'' \circ p = p'$.

In the case $I = \emptyset$, the existence of $p'' \in M$ with $p'' \circ s = m^*$ contradicts $m^* \models T(\text{PC}(a)) = \wedge_S \text{NAC}(s)$. Hence our assumption to have a $p' \colon P \to H$ in M is false, which implies $H \models \text{PC}(a)$.

2. Let $H \models \text{PC}(a)$. We have to show $m^* \models T(\text{PC}(a))$, i.e. for all triples $\langle S, s, p \rangle$ constructed in step 1 and all monomorphisms $p'' \colon S \to H$ in M with $p'' \circ s = m^*$ we have to find $i \in I$ and a morphism $q'' \colon T_i \to H$ in M with $q'' \circ t_i \circ t = p''$. Given $\langle S, s, p \rangle$ and p'' in M as above we define $p' = p'' \circ p \colon P \to H$. Then p' is in M, because p and p'' are in M, and $H \models \text{PC}(a)$ implies $q' \colon C \to H$ in M with $q' \circ a = p'$. Hence $p'' \circ p = p' = q' \circ a$. The universal property of pushouts implies the existence of a unique morphism $h \colon T \to H$ with $h \circ t = p''$ and $h \circ q = q'$. Now let $h = q'' \circ e'$ be a epi-mono factorization of h with epimorphism e' and monomorphism q'' in M. Then $q'' \circ e' \circ t = h \circ t = p''$ in M implies $e' \circ t$ is in M and $q'' \circ e' \circ q = h \circ q = q'$ in M implies $e' \circ q$ in M (M closed under decompositions). Hence according to construction step 2 $e' \circ t$ belongs to the family $(S \stackrel{t_i \circ t}{\to} T_i)_{i \in I}$ of $T(\text{PC}(a))$ such that $e' = t_i \colon T \to T_i$ for some $i \in I$. In the case $I \neq \emptyset$ we have q'' in M and $q'' \circ t_i \circ t = q'' \circ e' \circ t = h \circ t = p''$ implies $m^* \models T(\text{PC}(a))$. In the case $I = \emptyset$ we have a contradiction which means that our assumption to have $p'' \in M$ with $p'' \circ s = m^*$ is false. This implies $m^* \models \wedge_S \text{NAC}(s) = T(\text{PC}(a))$.

Remark. The proof of lemma 1 in both directions does not require that $a \colon P \to C$ is in M. If a is not in M, however, then t is not in M. (In fact, p in M in pushout (1) implies (1) pushout and pullback such that t in M would imply a in M). Hence there is no t_i in M s.t. $t_i \circ t$ is in M. This implies $I = \emptyset$, s.t. $T(\text{PC}(a)) = \wedge_S \text{NAC}(s)$. Hence we have for a not in M or $T = \emptyset$ the equivalence

$$m^* \models \wedge_S \text{NAC}(s) \Leftrightarrow H \models \text{PC}(a).$$

Proof of Lemma 2. 1. Let $m^* \models T(\text{NC}(a))$ and $I \neq \emptyset$. The claim $H \not\models \text{NC}(a)$ implies the existence of morphisms $p' \colon P \to H$ and $q' \colon C \to H$ in M with $q' \circ a = p'$ and will lead to a contradiction. Given $p' \colon P \to H$ in M as above and m^* we construct the coproduct $R + P$. This leads to a unique $f \colon R + P \to H$ with $f \circ \text{in}_R = m^*$ and $f \circ \text{in}_p = p'$. Now let $f = p'' \circ e$ an epi-mono-factorization of f with epimorphism e and monomorphism p'' in M and define $s = e \circ \text{in}_R$ and $p = e \circ \text{in}_P$. Similar to part 1 of the proof of lemma 1 we have that $\langle S, s, p \rangle$ belongs to the family $\wedge_S$ of $T(\text{NC}(a))$ and we have $p'' \circ p = p'' \circ e \circ \text{in}_P = f \circ \text{in}_P = p'$.

Moreover we have $p'' \circ s = p'' \circ e \circ \text{in}_R = f \circ \text{in}_R = m^*$ with monomorphism p'' in M and $q' \circ a = p'$. Now pushout (1) in the figure 3, implies existence of $h\colon T \to H$ with $h \circ t = p''$ and $h \circ q = q'$. Now let $h = q'' \circ e'$ epi-mono-factorization of h with epimorphism e' and monomorphism q'' in M. Then, by the decomposition property of M, $p'' = h \circ t = q'' \circ e' \circ t$ in M implies $e' \circ t$ in M and $q' = h \circ q = q'' \circ e' \circ q$ implies $e' \circ q$ in M. Hence $e' \circ t$ belongs to the family $t_i \circ t$ in the construction of $T(\text{NC}(a))$. Hence there is $i \in I$ with $e' = t_i\colon T \to T_i$ and q'' in M with $q'' \circ t_i \circ t = q'' \circ e' \circ t = h \circ t = p''$.

Our assumption $m^* \models T(\text{NC}(a))$ implies that for all $\langle S, s, p\rangle$ as in the construction and all $p''\colon S \to H$ in M with $p'' \circ s = m^*$ as given above there is no $i \in I$ and $q''\colon T_i \to H$ in M with $q'' \circ t_i \circ t = p''$. This is a contradiction to existence of $i \in I$ and q'' constructed above.

2. Let $H \models \text{NC}(a)$ and $I \neq \emptyset$. The claim $m^* \not\models T(\text{NC}(a))$ will lead to a contradiction. For $I \neq \emptyset$, $m^* \not\models T(\text{NC}(a))$ implies the existence of $\langle S, s, p\rangle$ as in the construction of $T(\text{NC}(a))$, and existence of $p''\colon S \to H$ in M with $p'' \circ s = m^*$, existence of $i \in I$ with $t_i \circ t$, $t_i \circ q$ in M, and existence of $q''\colon T_i \to H$ in M with $q'' \circ t_i \circ t = p''$. Now let $p' = p'' \circ p$, then p, p'' in M implies p' in M. Further let $q' = q'' \circ t_i \circ q$, then q'', $t_i \circ q$ in M implies q' in M. This implies $q' \circ a = q'' \circ t_i \circ q \circ a = q'' \circ t_i \circ t \circ p = p'' \circ p = p'$. Hence we have in contradiction to $H \models \text{NC}(a)$ p' and q' in M with $q' \circ a = p'$. For $I = \emptyset$, $m^* \not\models T(\text{NC}(a))$ implies existence of $p'' \in M$ with $p'' \circ s \neq m^*$. But this contradicts $H \models \text{NC}(a)$.

3. Let $I = \emptyset$. Then $m^* \models T(\text{NC}(a)) \Leftrightarrow$ true because $T(\text{NC}(a)) =$ true in this case. The claim $H \not\models \text{NC}(a)$ leads according to part 1 of the proof to $I \neq \emptyset$ which contradicts $I = \emptyset$. Hence we have $H \models \text{NC}(a)$. This means we have for $I = \emptyset$ $\quad m^* \models T(\text{NC}(a)) \Leftrightarrow H \models \text{NC}(a) \Leftrightarrow$ true.

Remark. The proof of lemma 2 does not require that a is in M. If a is not in M we have again $I = \emptyset$ (as in remark after lemma 1).

Proof of Lemma 3. Let $G \underset{p,m,m^*}{\Longrightarrow} H$ be any direct derivation.

Case 1. The pair $\langle r\colon K \to R, x\colon R \to X\rangle$ has no pushout complement. Then $p^{-1}(\text{ acc}_R) =$ true and $m \models p^{-1}(\text{ acc}_R)$. We have to show $m^* \models \text{ acc}_R$. This is true, because there is no $p\colon X \to H$ with $p \in M$ and $p \circ x = m^*$. Otherwise, since the pair $\langle r, m^*\rangle$ has a pushout complement, the pair $\langle r, x\rangle$ would have a pushout complement in contradiction to case 1 (pushout-pullback decomposition, $r, p \in M$).

Case 2. The pair $\langle r\colon K \to R, x\colon R \to X\rangle$ has a pushout complement and $I \neq \emptyset$.

Case 2.1. $m \models p^{-1}(\text{ acc}_R)$. We have to show $m^* \models \text{ acc}_R$, i.e. given a morphism $p\colon X \to H$ in M with $p \circ x = m^*$ we have to find an $i \in I$ and a morphism $q\colon C_i \to H$ in M with $q \circ x_i = p$. From the double pushout for $G \Rightarrow_{p,m,m^*} H$ and $p \circ x = m^*$ we obtain the following decomposition in pushouts (1), (2), (5), (6): First (5) is constructed as pullback of p and d_1 leading to pushouts (1)

and (5) (pushout-pullback decomposition lemma, r, p in M), with same square (1) as in the construction because of uniqueness of pushout complements for M-morphisms. Then (2) is constructed as pushout and we have $p'\colon Y \to G$ with $p' \circ y = m$ and pushout (6) induced by the pushouts (2) and (2) + (6). Since p is in M, z and p' are in M (M-morphisms are closed under pullbacks and pushouts).
In the case $I' = \emptyset$ we have no $p\colon X \to H$ with $p \in M$ and $p \circ x = m^*$, because this would imply $p'\colon Y \to G$ with $p' \in M$ and $p' \circ y = m$ violating $m \models p^{-1}(\text{ acc}_R)$. Having no p with $p \circ x = m^*$, however, implies $m^* \models \text{ acc}_R$. In the case $I' \neq \emptyset$ we have by $m \models p^{-1}(\text{ acc}_R)$ an $i \in I' \subseteq I$ with $y_i\colon Y \to D_i$ and $q'\colon D_i \to G$ in M with $q' \circ y_i = p'$. Now we are able to decompose pushouts (6) and (5) into pushouts (4)+(8) and (3)+(7) respectively using the same technique as above now from left to right (pushout-pullback decomposition, $l^*, q' \in M$) leading to a morphism $q\colon C_i \to H$ in M with $q \circ x_i = p$. This implies $m^* \models \text{ acc}_R$.

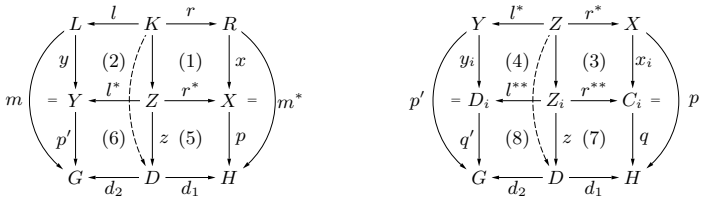


Fig. 5. Decomposition of pushouts

Case 2.2. $m^* \models \text{ acc}_R$. We have to show $m \models p^{-1}(\text{ acc}_R)$. Due to case 2 we have $p^{-1}(\text{ acc}_R) \neq \text{true}$. Hence for each morphism $p'\colon Y \to G$ in M with $p' \circ y = m$ we have to find an $i \in I'$ and a morphism $q'\colon D_i \to G$ in M with $q' \circ y_i = p'$. Given a morphism p' in M with $p' \circ y = m$ we can construct pushouts (1), (2), (5), (6) as above, where this time we first construct (6) as pullback leading in the right-hand side to a morphism $p\colon X \to H$ in M with $p \circ x = m^*$. Now $m^* \models \text{ acc}_R$ implies the existence of an $i \in I$ and a morphism $q\colon C_i \to H$ in M with $q \circ x_i = p$. Due to pushout (5) the pair $\langle r^*, p \rangle$ has a pushout complement, so that this is also true for $x_i\colon X \to C_i$ with $q \circ x_i = p$. Hence we have an $i \in I'$ and can decompose pushouts (5) and (6) into pushouts (3)+(7) and (4)+(8) from right to left leading to a morphism $q'\colon D_i \to G$ in M with $q' \circ y_i = p'$. This implies $m \models p^{-1}(\text{ acc}_R)$.

Case 3. The pair $\langle r, x \rangle$ has a pushout complement, but $I = \emptyset$.

Case 3.1. $m^* \not\models \text{ acc}_R = \text{NAC}(x)$ implies $p \in M$ with $p \circ x = m^*$. As shown in case 2.1 we obtain $p' \in M$ with $p' \circ y = m$ which implies $m \not\models \text{NAC}(y)$.

Case 3.2. $m \not\models p^{-1}(\text{ acc}_R) = \text{NAC}(y)$ implies in a similar way $m^* \not\models \text{NAC}(x)$ using the construction in case 2.2.

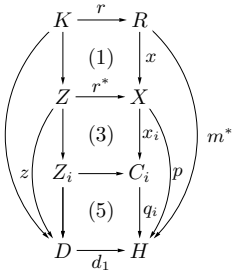
Proof of Lemma 4. Let $G \underset{p,m,m^*}{\Longrightarrow} H$ be any direct derivation. We have to show

$$(\star) \quad m \models \text{acc}_L \Leftrightarrow m^* \models \text{acc}_R.$$

Case 1. $I = \emptyset$. Then $\text{acc}_R = \text{acc}_L = \text{true}$ which implies $(\star)$.

Case 2. $I \neq \emptyset$ and $\langle r, x\rangle$ has no pushout complement then $\text{acc}_L = \text{true}$. We have to show $m^* \models \text{acc}_R$. Assume $m^* \not\models \text{acc}_R$. Then there is $p \in M$ with $p \circ x = m^*$. Since $\langle r, m^*\rangle$ has a pushout complement the pushout-pullback decomposition lemma with $r, p \in M$ implies that also $\langle r, x\rangle$ has a pushout complement. Contradiction. Hence $m^* \models \text{acc}_R$.

Case 3. Let $I = \emptyset$ and $I' = \emptyset$ and $\langle r, x\rangle$ has a pushout complement. In this case we have $\text{acc}_L = \text{true}$ and we have to show $m^* \models \text{acc}_R$. Assume $m^* \not\models \text{acc}_R$. Then there is $p \in M$ with $p \circ x = m^*$ and $i \in I$, $q_i \in M$ with $q_i \circ x_i = p$. This implies $m^* = q_i \circ x_i \circ x$. Since $\langle r, m^*\rangle$ has a pushout complement and $r, q_i, x_i \in M$ the pushout-pullback decomposition lemma implies that also $\langle r^*, q_i \circ x_i\rangle$ and $\langle r^*, x_i\rangle$ has a pushout complement. Contradiction to $I' = \emptyset$. Hence $m^* \models \text{acc}_R$.



Case 4. Let $I \neq \emptyset$ and $I' = \emptyset$ and $\langle r, x\rangle$ has a pushout complement. In this case we use the following negations:
a) $m \not\models \text{acc}_R = \text{N}(L \overset{y}{\to} Y, \wedge_{i \in I'}(Y \overset{y_i}{\to} D_i)) \Leftrightarrow$
$\exists p' \colon Y \to G$, $p' \in M$, $p' \circ y = m$ and $\exists i \in I'$ $\exists q_i' \colon D_i \to G$, $q_i' \in M$,$q_i' \circ y_i = p'$.
b) $m^* \not\models \text{acc}_R = \text{N}(R \overset{x}{\to} X, \wedge_{i \in I}(X \overset{x_i}{\to} C_i)) \Leftrightarrow$
$\exists p \colon X \to H$, $p \in M$, $p \circ x = m^*$ and $\exists i \in I$ $\exists q_i \colon C_i \to H$, $q_i \in M$, $q_i \circ x_i = p$.

Case 4.1. $m \not\models p^{-1}(\text{acc}_R) = \text{acc}_L$ and we have to show $m^* \not\models \text{acc}_R$. By $m \not\models \text{acc}_L$ we have $p' \in M$, $i \in I'$ and $q_i' \in M$ as given in a). From the double pushout of $G \Rightarrow_{p,m,m^*} H$ and $m = p' \circ y$ with $p' \in M$ we can construct pushouts (2), (6), (1), (5) in figure 5 by the pushout-pullback decomposition lemma with $l, p' \in M$ leading to the commutative diagram in figure 5 with $p \in M$, $p \circ x = m^*$. Using $q_i' \circ y_i = p'$ we are able to decompose the pushouts (6) and (5) to pushouts (4), (8) and (3), (7) (by the pushout-pullback decomposition lemma with $l^*, q_i' \in M$) leading to morphisms $q_i \colon C_i \to H$, $q_i \in M$ with $q_i \circ x_i = p$ (see figure 5). This implies $m^* \not\models \text{acc}_R$ as given in b).

Case 4.2. Let $m^* \not\models \text{acc}_R$ and we have to show $m \not\models \text{acc}_L$. By $m^* \not\models \text{acc}_R$ we have $p \in M$, $i \in I$, and $q_i \in M$ as given in b). From the double pushout of

$G \Rightarrow_{p,m,m^*} H$ and $m^* = p \circ x$ with $p \in M$ we can construct pushouts (1), (5), (2), (6) in figure 5 leading to $p \in M$ with and $p \circ x = m^*$. Using $q_i \circ x_i = p$ we are able to decompose the pushouts (5) and (6) to pushouts (3), (7) and (4), (8) in figure 5 with $q' \in M$ and $q' \circ y_i = p'$. This implies $m \not\models \text{acc}_R$ as given in a).

References

1. Paolo Bottoni, Manuel Koch, Manuel Parisi-Presicce, and Gabriele Taentzer. Consistency checking and visualization of ocl constraints. In *UML 2000*, volume 1939 of *Lecture Notes in Computer Science*, pages 294–308. Springer-Verlag, 2000.
2. Andrea Corradini, Ugo Montanari, Francesca Rossi, Hartmut Ehrig, Reiko Heckel, and Michael Löwe. Algebraic approaches to graph transformation. Part I: Basic concepts and double pushout approach. In *Handbook of Graph Grammars and Computing by Graph Transformation*, volume 1, chapter 3, pages 163–245. World Scientific, 1997.
3. Hartmut Ehrig. Introduction to the algebraic theory of graph grammars. In *Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *Lecture Notes in Computer Science*, pages 1–69. Springer-Verlag, 1979.
4. Hartmut Ehrig and Annegret Habel. Graph grammars with application conditions. In G. Rozenberg and A. Salomaa, editors, *The Book of L*, pages 87–100. Springer-Verlag, Berlin, 1986.
5. Hartmut Ehrig, Annegret Habel, Hans-Jörg Kreowski, and Francesco Parisi-Presicce. Parallelism and concurrency in high level replacement systems. *Mathematical Structures in Computer Science*, 1:361–404, 1991.
6. Hartmut Ehrig, Annegret Habel, Julia Padberg, and Ulrike Prange. Adhesive high-level replacement categories and systems. In *Graph Transformation (ICGT'04)*, Lecture Notes in Computer Science. Springer-Verlag, 2004.
7. Hartmut Ehrig, Ulrike Prange, and Gabriele Taentzer. Fundamental theory of typed attributed graph transformation. In *Graph Transformation (ICGT'04)*, Lecture Notes in Computer Science. Springer-Verlag, 2004.
8. Annegret Habel, Reiko Heckel, and Gabriele Taentzer. Graph grammars with negative application conditions. *Fundamenta Informaticae*, 26:287–313, 1996.
9. Reiko Heckel and Annika Wagner. Ensuring consistency of conditional graph grammars — a constructive approach. In *SEGRAGRA 95*, volume 2 of *Electronic Notes in Theoretical Computer Science*, pages 95–104, 1995.
10. Manuel Koch and Francesco Parisi-Presicce. Describing policies with graph constraints and rules. In *Graph Transformation (ICGT 2002)*, volume 2505 of *Lecture Notes in Computer Science*, pages 223–238. Springer-Verlag, 2002.
11. Stephen Lack and Paweł Sobociński. Adhesive categories. In *Proc. of Foundations of Software Science and Computation Structures (FOSSACS'04)*, volume 2987 of *Lecture Notes in Computer Science*, pages 273–288. Springer-Verlag, 2004.
12. Detlef Plump. Hypergraph rewriting: Critical pairs and undecidability of confluence. In *Term Graph Rewriting: Theory and Practice*, pages 201–213. John Wiley, New York, 1993.