Manipulation of Graphs, Algebras and Pictures

Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday

Parallelism and Concurrency Theorems for Rules with Nested Application Conditions

Hartmut Ehrig, Annegret Habel and Leen Lambers

23 pages
Parallelism and Concurrency Theorems for Rules with Nested Application Conditions

Hartmut Ehrig¹, Annegret Habel² and Leen Lambers³

¹ ehrig@cs.tu-berlin.de
Technische Universität Berlin, Germany
² Annegret.Habel@informatik.uni-oldenburg.de
Carl v. Ossietzky Universität Oldenburg, Germany
³ Leen.Lambers@hpi.uni-potsdam.de
Hasso-Plattner-Institut für Softwaresystemtechnik, Potsdam, Germany

Abstract: We present Local Church-Rosser, Parallelism, and Concurrency Theorems for rules with nested application conditions in the framework of weak adhesive HLR categories including different kinds of graphs. The proofs of the statements are based on the corresponding statements for rules without application conditions and two Shift-Lemmas, saying that nested application conditions can be shifted over morphisms and rules.

Keywords: High-level transformation systems, weak adhesive HLR categories, parallelism, concurrency, nested application conditions, negative application conditions.

1 Introduction

Graph replacement systems have been studied extensively and applied to several areas of computer science [Roz97, EEKR99, EKMR99] and were generalized to high-level replacement (HLR) systems [EHKP91] and weak adhesive HLR systems [EEHP06, EEPT06], based on adhesive categories [LS05]. Application conditions restrict the applicability of a rule. Originally, they were defined in [EH86], specialized to negative application conditions (NACs) [HHT96], and generalized to nested application conditions (ACs) [HP05].

The Local Church-Rosser, Parallelism, and Concurrency Theorems are well-known theorems for graph replacement systems on rules without application conditions [EK76, Kre77a, Kre77b, Ehr79, ER80, Hab80] and are generalized to high-level replacement (HLR) systems [EHKP91] and rules with negative application conditions [LEPO08b]. Nested application conditions (ACs) were introduced in [HP05] and intensively studied in [HP09]. They generalize the well-known negative application conditions (NACs) in the sense of [HHT96, LEPO08b]. Furthermore, nested application conditions in the category of graphs are expressively equivalent to first order formulas on graphs. In this paper, we generalize the theorems to weak adhesive HLR systems on rules with nested application conditions.
The proofs of the theorems are based on the corresponding theorems for weak adhesive HLR systems on rules without application conditions in [EEPT06] and facts on nested application conditions in [HP09], saying that application conditions can be shifted over morphisms and rules.

The paper is organized as follows: In Sections 2 and 3, we review the definitions of a weak adhesive HLR category, nested conditions, and rules. In Section 4, we state and prove the Local Church-Rosser, Parallelism, and Concurrency Theorems for rules with nested application conditions. The concepts are illustrated by examples in the category of graphs with the class $\mathcal{M}$ of all injective graph morphisms. A conclusion including further work is given in Section 5.

## 2 Graphs and High-level Structures

We recall the basic notions of directed, labeled graphs [Ehr79, CMR+97] and generalize them to high-level structures [EHKP91]. The idea behind the consideration of high-level structures is to avoid similar investigations for similar structures such as Petri-nets and hypergraphs.

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

**Definition 1 (Graphs and Graph Morphisms)** Let $\mathcal{C} = (\mathcal{C}_V, \mathcal{C}_E)$ be a fixed, finite label alphabet. A graph over $\mathcal{C}$ is a system $G = (V_G, E_G, s_G, t_G, l_G, m_G)$ consisting of two finite sets $V_G$ and $E_G$ of nodes (or vertices) and edges, source and target functions $s_G, t_G: E_G \rightarrow V_G$, and two labeling functions $l_G: V_G \rightarrow \mathcal{C}_V$ and $m_G: E_G \rightarrow \mathcal{C}_E$. A graph with an empty set of nodes is empty and denoted by $\emptyset$. A graph morphism $g: G \rightarrow H$ consists of two functions $g_V: V_G \rightarrow V_H$ and $g_E: E_G \rightarrow E_H$ that preserve sources, targets, and labels, that is, $s_H \circ g_E = g_V \circ s_G, t_H \circ g_E = g_V \circ t_G, l_H \circ g_V = l_G$, and $m_H \circ g_E = m_G$. 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 $h: H \rightarrow M$ consists of the composed functions $h_V \circ g_V$ and $h_E \circ g_E$. The category having graphs as objects and graph morphisms as arrows is called Graphs.

Our considerations are based on weak adhesive HLR categories, i.e. categories based on objects of many kinds of structures which are of interest in computer science and mathematics, e.g. Petri-nets, (hyper)graphs, and algebraic specifications, together with their corresponding morphisms and with specific properties. Readers interested in the category-theoretic background of these concepts may consult e.g. [EEPT06].

**Definition 2 (Weak Adhesive HLR Category)** A category $\mathcal{C}$ with a morphism class $\mathcal{M}$ is a weak adhesive HLR category, if the following properties hold:
1. \( \mathcal{M} \) is a class of monomorphisms closed under isomorphisms, composition, and decomposition, i.e., for morphisms \( f \) and \( g \), \( f, g \in \mathcal{M} \), \( g \) isomorphism (or vice versa) implies \( g \circ f \in \mathcal{M} \); \( f, g \in \mathcal{M} \) implies \( g \circ f \in \mathcal{M} \); and \( g \circ f \in \mathcal{M} \), \( g \in \mathcal{M} \) implies \( f \in \mathcal{M} \).

2. \( \mathcal{C} \) has pushouts and pullbacks along \( \mathcal{M} \)-morphisms, i.e. pushouts and pullbacks, where at least one of the given morphisms is in \( \mathcal{M} \), and \( \mathcal{M} \)-morphisms are closed under pushouts and pullbacks, i.e. given a pushout (1) as in the figure below, \( m \in \mathcal{M} \) implies \( n \in \mathcal{M} \) and, given a pullback (1), \( n \in \mathcal{M} \) implies \( m \in \mathcal{M} \).

3. Pushouts in \( \mathcal{C} \) along \( \mathcal{M} \)-morphisms are weak VK-squares, i.e. for any commutative cube in \( \mathcal{C} \) where we have the pushout with \( m \in \mathcal{M} \) and \( (f \in \mathcal{M} \) or \( b, c, d \in \mathcal{M} \)) in the bottom and the back faces are pullbacks, it holds: the top is pushout iff the front faces are pullbacks.

Fact 1 ([EEPT06]) The category Graphs with class \( \mathcal{M} \) of all injective graph morphisms is a weak adhesive HLR category. Further examples of weak adhesive HLR categories are the categories of hypergraphs with all injective hypergraph morphisms, place-transition nets with all injective net morphisms, and algebraic specifications with all strict injective specification morphisms.

Remark 1 Adhesive categories [LS04, EEPT06] are special cases of (weak) adhesive HLR categories, where, in addition, the class \( \mathcal{M} \) is the class of all monomorphisms. By [EEPT06], the category \( \langle \text{PTNets, } \mathcal{M} \rangle \) of place/transition nets and the category \( \langle \text{Spec, } \mathcal{M}_{\text{strict}} \rangle \) of algebraic specifications are weak adhesive HLR, but not adhesive.

Weak adhesive HLR-categories have a number of nice properties, called HLR properties.

Lemma 1 (Properties of weak adhesive HLR categories [LS04, EEPT06]) For a weak adhesive HLR-category \( \langle \mathcal{C}, \mathcal{M} \rangle \), the following properties hold:

1. Pushouts along \( \mathcal{M} \)-morphisms are pullbacks.

2. \( \mathcal{M} \) pushout-pullback decomposition. If the diagram (1)+(2) in the figure below is a pushout, (2) a pullback, \( w \in \mathcal{M} \) and \( (l \in \mathcal{M} \) or \( c \in \mathcal{M} \)), then (1) and (2) are pushouts and also pullbacks.

3. Cube pushout-pullback decomposition. Given the commutative cube (3) in the figure below, where all morphisms in the top and the bottom are in \( \mathcal{M} \), the top is pullback, and the front faces are pushouts, then the bottom is a pullback iff the back faces of the cube are pushouts.
4. Uniqueness of pushout complements. Given morphisms $c: A \to C$ in $\mathcal{M}$ and $s: C \to D$, then 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.

In the following, we consider weak adhesive HLR categories with an $\mathcal{E} \cdot \mathcal{M}$ factorization and binary coproducts.

**Definition 3 ($\mathcal{E} \cdot \mathcal{M}$ Factorization)** A weak adhesive HLR category $\langle \mathcal{C}, \mathcal{M} \rangle$ has an $\mathcal{E} \cdot \mathcal{M}$ factorization for a given morphism class $\mathcal{E}$ if, for each morphism $f$, there is a decomposition, unique up to isomorphism, $f = m \circ e$ with $e \in \mathcal{E}$ and $m \in \mathcal{M}$.

**Remark 2 (Binary coproducts)** In a weak adhesive HLR category $\langle \mathcal{C}, \mathcal{M} \rangle$ with binary coproducts, the binary coproducts are compatible with $\mathcal{M}$ in the sense that $f, g \in \mathcal{M}$ implies $f + g \in \mathcal{M}$. In fact, if $f \in \mathcal{M}$ implies $(f + id) \in \mathcal{M}$ and $g \in \mathcal{M}$ implies $(id + g) \in \mathcal{M}$, but now $(f + g) = (id + g) \circ (f + id) \in \mathcal{M}$ by closure under composition.

The category Graphs with the classes $\mathcal{M}$ and $\mathcal{E}$ of all injective and surjective graph morphisms, respectively, satisfies the specific properties.

**Fact 2 ([EEPT06])** $\langle \text{Graphs}, \mathcal{M} \rangle$ has an $\mathcal{E} \cdot \mathcal{M}$ factorization and binary coproducts.

### 3 Rules with Application Conditions

We use the framework of weak adhesive HLR categories and introduce rules with application conditions for high-level structures like Petri nets, (hyper)graphs, and algebraic specifications.

**Assumption** We assume that $\langle \mathcal{C}, \mathcal{M} \rangle$ is a weak adhesive HLR category with an $\mathcal{E} \cdot \mathcal{M}$ factorization (used in Shift-Lemma 2) and binary coproducts (used in Definition 8).

Application conditions are defined as in [HP09], Definition 4. Syntactically, application conditions may be seen as a tree of morphisms equipped with certain logical symbols such as quantifiers and connectives.
Definition 4 (Nested Application Conditions) A nested application condition, short application condition, condition, or AC, $ac_P$ over an object $P$ is of the form true or $\exists (a, ac_C)$, where $a: P \to C$ is a morphism and $ac_C$ is an application condition over $C$. Moreover, Boolean formulas over conditions over $P$ are conditions over $P$: for conditions $c, c_i$ over $P$ with $i \in I$ (for all index sets $I$), $\neg c$ and $\bigwedge_{i \in I} c_i$ are conditions over $P$. $\exists a$ abbreviates $\exists (a, \text{true})$, $\forall (a, ac_C)$ abbreviates $\neg \exists (a, \neg ac_C)$, and $\not\exists$ abbreviates $\neg \exists$.

Every morphism satisfies true. A morphism $p: P \to G$ satisfies $\exists (a, ac_C)$ if there exists a morphism $q$ in $\mathcal{M}$ such that $q \circ a = p$ and $q \models ac_C$. The satisfaction of conditions over $P$ by morphisms with domain $P$ is extended to Boolean formulas over conditions in the usual way. We write $p \models ac_P$ to denote that the morphism $p$ satisfies $ac_P$. Two conditions $ac_P$ and $ac_P'$ over $P$ are equivalent, denoted by $ac_P \equiv ac_P'$, if for all morphisms $p: P \to G$, $p \models ac_P$ iff $p \models ac_P'$.

Remark 3 The definition of conditions generalizes those in [HHT96, HW95, KMP05, EEHP06]. In the context of rules, conditions are also called application conditions. Negative application conditions [HHT96, LEPO08b] correspond to nested application conditions of the form $\not\exists a$. Examples of nested application conditions are given below.

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\exists (Q_1 \leftarrow Q_2)$</td>
<td>There is an edge from the image of 1 to the image of 2.</td>
</tr>
<tr>
<td>$\not\exists (Q_1 \leftarrow Q_2)$</td>
<td>There is no edge from the image of 1 to the image of 2.</td>
</tr>
<tr>
<td>$\exists (Q_1 \leftarrow Q_2 \leftarrow Q_3)$</td>
<td>There is a directed path of length 2, but not of length 1, from the image of 1 to the image of 2.</td>
</tr>
<tr>
<td>$\not\exists (Q_1 \leftarrow Q_2 \leftarrow Q_3)$</td>
<td>There is a proper edge outgoing from the image of 1 without edge in reverse direction.</td>
</tr>
<tr>
<td>$\forall (Q_1 \leftarrow Q_2)$</td>
<td>For every proper edge outgoing from the image of 1, the target has a loop.</td>
</tr>
<tr>
<td>$\not\exists (Q_1 \leftarrow Q_2)$</td>
<td>For the image of node 1, there exists an outgoing edge such that, for all edges outgoing from the target, the target has a loop.</td>
</tr>
</tbody>
</table>

In the presence of an $\mathcal{M}$-initial object $I$ [HP09], conditions $\exists (a, c)$ over the initial object $I$ can be used to define constraints for objects $G$, namely $G$ satisfies $\exists (a, c)$ if the unique $\mathcal{M}$-morphism $I \hookrightarrow G$ satisfies $\exists (a, c)$.

Remark 4 In general, one could choose a satisfiability notion, i.e. a class of morphisms $\mathcal{M}'$, and require that the morphism $q$ in Definition 4 is in $\mathcal{M}'$. Examples are $\mathcal{M}'$ and $\mathcal{M}$-satisfiability [HP06] where $\mathcal{M}'$ and $\mathcal{M}$ are the classes of all morphisms and all monomorphisms, respectively.
Application conditions can be shifted over morphisms into corresponding application conditions over the codomain of the morphism.

**Lemma 2 (Shift of Application Conditions over Morphisms)** Let \( \langle \mathcal{C}, \mathcal{M} \rangle \) be a weak adhesive HLR category with \( \mathcal{E} \cdot \mathcal{M} \)-factorization. There is a transformation \( \text{Shift} \) such that, for all application conditions \( ac_P \) over \( P \) and all morphisms \( b: P \to P' \), \( n: P' \to H \), \( n \circ b = ac_P \iff n = \text{Shift}(b, ac_P) \).

\[
\begin{array}{c}
\text{ac}_P \triangleright P \xrightarrow{b} P' \xleftarrow{n} \text{Shift}(b, ac_P) \\
n \circ b \xrightarrow{H} n
\end{array}
\]

The \( \text{Shift} \)-construction is based on jointly epimorphic pairs of morphisms. A morphism pair \( (e_1, e_2) \) with \( e_i: A_i \to B \) \( (i = 1, 2) \) is jointly epimorphic if, for all morphisms \( g, h: B \to C \) with \( g \circ e_i = h \circ e_i \) for \( i = 1, 2 \), we have \( g = h \). In the case of graphs, “jointly epimorphic” means “jointly surjective”: a morphism pair \((e_1, e_2)\) is jointly surjective, if for each \( b \in B \) there is a preimage \( a_1 \in A_1 \) with \( e_1(a_1) = b \) or \( a_2 \in A_2 \) with \( e_2(a_2) = b \). For previous versions of the \( \text{Shift} \)-construction see [LEPO08b, HP09].

**Construction** The transformation \( \text{Shift} \) is inductively defined as follows:

\[
\begin{array}{c}
P \xrightarrow{b} P' \\
\text{Shift}(b, \exists(a, ac_C)) = \bigvee_{(d', b') \in \mathcal{F}} \exists(d', \text{Shift}(b', ac_C)) \quad \text{with} \\
\mathcal{F} = \{(d', b') \mid (d', b') \text{ jointly epimorphic, } b' \in \mathcal{M}, \text{ (1) commutes}\} \\
\end{array}
\]

For Boolean formulas over application conditions, \( \text{Shift} \) is extended in the usual way: For application conditions \( ac_i, ac_i \) with \( i \in I \) (for all index sets \( I \)), \( \text{Shift}(b, \neg ac_i) = \neg \text{Shift}(b, ac_i) \) and \( \text{Shift}(b, \land_{i \in I} ac_i) = \land_{i \in I} \text{Shift}(b, ac_i) \).

**Example 1** Given the morphism \( b: P \to P' \) below, the application condition \( \exists a \) is shifted into the application condition \( \text{Shift}(b, \exists a) = \exists d' \lor \exists d'' \lor \exists \text{id}_P \) where \( d' \) is the morphism depicted in the figure below and \( d'' \) obtained from \( d' \) by identifying the nodes with label \( \text{ordernr} \) in \( C' \); it can be simplified to true because \( \exists \text{id}_P \) is equivalent to true. The application condition \( \nexists a \) is shifted into \( \text{Shift}(b, \nexists a) = \neg \text{Shift}(b, \exists a) \equiv \neg \text{true} \equiv \text{false} \).

\[
\begin{array}{c}
P \xrightarrow{b} P' \\
\end{array}
\]
Proof of Lemma 2. The statement is proved by structural induction. Basis. For the condition true, the equivalence holds trivially. Inductive step. For a condition of the form \( \exists (a,ac_C) \), we have to show \( n \circ b \models \exists (a,ac_C) \iff n \models \text{Shift}(b,\exists (a,ac_C)) \).

Only if. Let \( n \circ b \models \exists (a,ac_C) \). By definition of satisfiability, there is some \( q \in \mathcal{M} \) with \( q \circ a = n \circ b \) and \( q \models ac_C \). Let \( (\overline{a},\overline{b}) \) be the pushout in (1) in the left diagram below. By the universal property of pushouts, there is an induced morphism \( \overline{q} : \overline{C} \to H \) such that \( q = \overline{q} \circ \overline{b} \) and \( n = \overline{q} \circ \overline{a} \). By \( \mathcal{E} \)-factorization of \( \overline{q}, \overline{q} = m \circ e \) with \( e \in \mathcal{E} \) and \( m \in \mathcal{M} \). Define now \( a' = e \circ \overline{a} \) and \( b' = e \circ \overline{b} \). Then the diagram \( PP'CC' \) commutes. Since \( \mathcal{M} \) is closed under decomposition, \( q = m \circ b' \in \mathcal{M}, m \in \mathcal{M} \) implies \( b' \in \mathcal{M} \). Since \( (\overline{a},\overline{b}) \) is jointly epimorphic and \( e \in \mathcal{E}, (a',b') \) is jointly epimorphic. Thus, \((a',b') \in \mathcal{F}\). By inductive hypothesis, \( q = m \circ b' \models ac_C \iff m \models \text{Shift}(b',ac_C) \). Now \( n \models \exists (a',\text{Shift}(b',ac_C)) \) and, by definition of Shift, \( n \models \exists (b,\text{Shift}(a,ac_C)) \).

If. Let \( n \models \text{Shift}(b,\exists (a,ac_C)) \). Then there is some \((a',b') \in \mathcal{F}\) with \( b' \in \mathcal{M}\) such that \( n \models \exists (a',\text{Shift}(b',ac_C)) \) and some \( m \in \mathcal{M} \) such that \( m \circ a' = n \) and \( m \models \text{Shift}(b',ac_C) \). By inductive hypothesis, \( m \models \text{Shift}(b',ac_C) \iff m \circ b' \models ac_C \). Now \( m \circ b' \in \mathcal{M}, m \circ b' \circ a = n \circ b \) (see the right diagram above), and \( m \circ b' \models ac_C \), i.e., \( n \circ b \models \exists (a,ac_C) \).

\[ \square \]

Rules [EEHP06, HP09] are specified by a span of \( \mathcal{M} \)-morphisms \( \langle L \leftarrow K \rightarrow R \rangle \) with a left and a right application condition. We consider the classical semantics based on the double-pushout construction [Ehr79, CMR+97].

Definition 5 (Rules) A rule \( \rho = \langle p, ac_L, ac_R \rangle \) consists of a plain rule \( p = \langle L \leftarrow K \rightarrow R \rangle \) with \( K \leftarrow L \) and \( K \rightarrow R \) in \( \mathcal{M} \) and two application conditions \( ac_L \) and \( ac_R \) over \( L \) and \( R \), respectively. \( L \) and \( R \) are called the left- and the right-hand side of \( p \) and \( K \) the interface; \( ac_L \) and \( ac_R \) are the left and right application condition of \( p \).

\[ \begin{array}{c}
  \text{acl} \downarrow L \quad \downarrow \text{acL} \\
  \quad \quad \quad \quad \quad \quad \quad \quad \quad \quad (1) \\
  m \downarrow \quad \downarrow \text{acR} \\
  G \quad \quad \quad \quad \quad \quad D \quad \downarrow H
\end{array} \]

A direct derivation consists of two pushouts (1) and (2) such that \( m \models ac_L \) and \( m^* \models ac_R \). We write \( G \Rightarrow_{\rho,m,m^*} H \) and say that \( m : L \rightarrow G \) is the match of \( \rho \) in \( G \) and \( m^* : R \rightarrow H \) is the comatch of \( \rho \) in \( H \). We also write \( G \Rightarrow_{\rho,m} H \) or \( G \Rightarrow_{\rho} H \) to express that there is an \( m^* \) or there are \( m \) and \( m^* \), respectively, such that \( G \Rightarrow_{\rho,m,m^*} H \).

The concept of rules is completely symmetric.

7 / 23 Volume 26 (2010)
**Fact 3 (Inverse rule)** For every rule \( \rho = (p, ac_L, ac_R) \) with \( p = (L \leftarrow K \leftarrow R) \), the rule \( \rho^{-1} = (p^{-1}, ac_R, ac_L) \) with \( p^{-1} = (R \leftarrow K \leftarrow L) \) is the inverse rule of \( \rho \). For every direct derivation \( G \Rightarrow_{\rho, m, m'} H \), there is a direct derivation \( H \Rightarrow_{\rho^{-1}, m', m} G \) via the inverse rule.

**Notation** In the case of graphs, a rule (schema) \( (L \leftarrow K \leftarrow R) \) with discrete interface \( K \) is shortly depicted by \( L \Rightarrow R \), where the nodes of \( K \) are indexed in the left-and the right-hand side of the rule (schema). A negative application condition of the form \( \neg(L \leftarrow L') \) is integrated in the left-hand side of a rule (schema) by crossing the part \( L' - L \). E.g. the rule (schema) \( (p, ac_L) \) with

\[
\begin{align*}
p &= \left\langle \begin{array}{c}
\text{authors} \\
\text{authors} \\
\text{name}
\end{array} \right\rangle \\
ac_L &= \neg\left\langle \begin{array}{c}
\text{authors} \\
\text{authors} \\
\text{name}
\end{array} \right\rangle
\end{align*}
\]

is depicted by

\[
\begin{array}{c}
\text{authors} \\
\text{name} \\
\text{authors} \\
\text{name}
\end{array}
\]

Moreover, the grey edge with labels \(+, -\) in the rule (schema) RegisterBook(catnr, ordernr) in the figure below represents the conjunction of the negative application conditions “There does not exist a \( +\)-labelled edge” and “There does not exist a \( -\)-labelled edge”.

**Example 2** In the figure on the next page, rules (rule schemata) with left application conditions are given, corresponding more or less to the operations of the small library system originally investigated in [EK80].

Right application conditions of rules can be shifted into corresponding left application conditions and vice versa.

**Lemma 3 (Shift of Application Conditions over Rules [HP09])** There are transformations \( L \) and \( R \) such that, for every right application condition \( ac_R \) and every left application condition \( ac_L \) of a rule \( \rho \) and every direct derivation \( G \Rightarrow_{\rho, m, m'} H \), \( m \models L(\rho, ac_R) \Leftrightarrow m^* \models ac_R \) and \( m \models ac_L \Leftrightarrow m^* \models R(\rho, ac_L) \).

**Construction** The transformation \( L \) is inductively defined as follows:

\[
\begin{array}{c}
L(\rho, \text{true}) = \text{true} \\
L(\rho, \exists(a, ac_X)) = \exists(b, L(\rho^*, ac_X)) \text{ if } \langle r, a \rangle \text{ has a pushout complement (1) and } \rho^* = \langle Y \leftarrow Z \rightarrow X \rangle \text{ is the derived rule by constructing the pushout (2).} \\
L(\rho, \exists(a, ac_X)) = \text{false, otherwise.}
\end{array}
\]

For Boolean formulas over application conditions, \( L \) is extended in the usual way: For application conditions \( ac_i \) with \( i \in I \), \( L(b, \neg ac) = \neg L(b, ac) \) and \( L(b, \land_{i \in I} ac_i) = \land_{i \in I} L(b, ac_i) \). The transformation \( R \) is given by \( R(\rho, ac_L) = L(\rho^{-1}, ac_L) \).
Example 3  Given the rule (schema) $\rho = \text{OrderBook}(\text{ordernr}, \text{name}, \text{title}, \text{name}')$ in the upper row of the figure below, the right application condition $\exists(R \rightarrow X)$ is shifted over $\rho$ into the left application condition $\exists(L \rightarrow Y)$. 

![Diagram](image-url)
In the following, we define the equivalence of rules and the equivalence of application conditions with respect to a rule. The equivalence with respect to a rule is more restrictive than the unrestricted one in Definition 4.

Definition 6 (Equivalence) Two rules $\rho$ and $\rho'$ are equivalent, denoted by $\rho \equiv \rho'$, if the relations $\Rightarrow_\rho$ and $\Rightarrow_{\rho'}$ are equal. For a rule $\rho$, two left (right) application conditions $ac$ and $ac'$ are $\rho$-equivalent, denoted by $ac \equiv_\rho ac'$, if the rules obtained from $\rho$ by adding the application condition $ac$ and $ac'$, respectively, are equivalent.

There is a close relationship between the transformations $L$ and $R$: For every rule $\rho$, Shift of a condition over the rule to the left and then over the rule to the right is $\rho$-equivalent to the original condition.

Lemma 4 (L and R) For every rule $\rho$ and every application condition $ac$ over $R$, the right-hand side of the plain rule of $\rho$, the application conditions $R(\rho, L(\rho, ac))$ and $ac$ are $\rho$-equivalent: $R(\rho, L(\rho, ac)) \equiv_\rho ac$.

Proof. By the Shift-Lemma 3, for every direct derivation $G \Rightarrow_{\rho, m, m'} H, m^* \models R(\rho, L(\rho, ac))$ $\iff m \models L(\rho, ac) \iff m^* \models ac$, i.e., $R(\rho, L(\rho, ac))$ and $ac$ are $\rho$-equivalent.

Remark 5 In general, the application conditions $R(\rho, L(\rho, ac))$ and $ac$ are not equivalent in the sense of Definition 4. E.g., for the rule $\rho = (p, true, ac)$ with $p = (\emptyset \leftarrow \emptyset \leftarrow \emptyset)$ and $ac = \exists Q \rightarrow Q \rightarrow Q$, $L(\rho, \neg ac) = \neg L(\rho, ac) = \neg false \equiv true$ and $R(\rho, L(\rho, \neg ac)) = R(\rho, true) = true \neq \neg ac$.

There is a nice interchange result of Shift and L saying that, for a rule $\rho$, the shift of a right application condition over a rule and a match is $\rho$-equivalent to the shift of the application condition over the comatch and the rule induced by the match.

Lemma 5 (Shift and L) For every direct derivation $L^* \Rightarrow_{\rho, k, k'} R^*$ via a rule $\rho$ and every application condition $ac$, Shift$(k, L(\rho, ac)) \equiv_{\rho^*} L(\rho^*, Shift(k^*, ac))$, where $\rho^*$ denotes the rule derived from $\rho$ and $k$. A corresponding statement holds for Shift and R.

Proof. Let $G \Rightarrow_{\rho, l, l'} H$ be a direct derivation, $m = l \circ k$ and $m^* = l^* \circ k^*$. By Shift-Lemmas 2 and 3, we have $l \models Shift(k, L(\rho, ac)) \iff m \models L(\rho, ac) \iff m^* \models ac_R \iff l^* \models Shift(k^*, ac) \iff l \models L(\rho^*, Shift(k^*, ac))$. 

---

Festschrift H.-J. Kreowski 10 / 23
As a consequence of Shift-Lemma 3, every rule can be transformed into an equivalent one with true right application condition. A rule of the form \( \langle p, ac_L, \text{true} \rangle \) is said to be a rule with left application condition and is abbreviated by \( \langle p, ac_L \rangle \).

**Corollary 1 (Rules with Left Application Condition)** There is a transformation \( \text{Left} \) from rules into rules with left application condition such that, for every rule \( \rho \), the rules \( \rho \) and \( \text{Left}(\rho) \) are equivalent.

**Proof.** For a rule \( \rho = \langle p, ac_L, ac_R \rangle \), \( \text{Left}(\rho) = \langle p, ac_L \land L(\rho), ac_R \rangle \). By Definition 5, Shift-Lemma 3, and the definition of Left, the rules \( \rho \) and \( \text{Left}(\rho) \) are equivalent:

\[
G \Rightarrow \rho, m, m^* H \iff G \Rightarrow p, m, m^* H \land m \models ac_L \land m^* \models ac_R \iff G \Rightarrow p, m, m^* H \land m \models L(\rho), ac_R \iff G \Rightarrow L(\rho), m, m^* H.
\]

4 Local Church-Rosser, Parallelism, and Concurrency

In this section, we present Local Church-Rosser, Parallelism, and Concurrency Theorems for rules with application conditions generalizing the well-known theorems for rules without application conditions [EEPT06] and with negative application conditions [LEO06]. The proofs of the statements are based on the corresponding statements for rules without application conditions [EEPT06] and Shift-Lemmas 2 and 3, saying that application conditions can be shifted over morphisms and rules. The structure of the proofs is as follows: We switch from derivations with application conditions to the corresponding derivations without application conditions, use the results for derivations without application conditions, and lift the results without application conditions to application conditions.

| derivations with ACs | \( \Rightarrow \) | result with ACs |
| derivations without ACs | \( \Rightarrow \) | result without ACs |

**Fact 4 (Every derivation with ACs induces a derivation without ACs)** For every direct derivation \( G \Rightarrow p, m H \), there is a direct derivation \( G \Rightarrow p, m H \) via the plain rule \( p \), called the underlying direct derivation without ACs.

In the following, we study parallel and sequential independence of direct derivations for rules with application conditions. By Corollary 1, we may assume that the rules are rules with left application condition.

**Assumption** Let \( \rho_1 = \langle p_1, ac_{L_1} \rangle \) and \( \rho_2 = \langle p_2, ac_{L_2} \rangle \) be rules with \( p_i = \langle L_i \leftarrow K_i \leftarrow R_i \rangle \) for \( i = 1, 2 \).

Two direct derivations are parallel (sequentially) independent if the underlying direct derivations without application conditions are parallel (sequentially) independent and the induced
matches satisfy the corresponding application conditions. For rules with negative application conditions, the definition corresponds to the one in [LEO06].

**Definition 7 (Parallel and Sequential Independence)** Two direct derivations \( H_1 \leftarrow \rho_1, m_1 \) \( G \Rightarrow \rho_2, m_2 \) \( H_2 \) are **parallel independent** if there are morphisms \( d_2 : L_1 \to D_2 \) and \( d_1 : L_2 \to D_1 \) such that the triangles \( L_1 D_2 G \) and \( L_2 D_1 G \) commute, \( m_1' = c_2 \circ d_2 \models ac_{L_1} \), and \( m_2' = c_1 \circ d_1 \models ac_{L_2} \).

Two direct derivations \( G \Rightarrow \rho_1, m_1 \) \( H_1 \) \( \Rightarrow \rho_2, m_2 \) \( M \) are **sequentially independent** if there are morphisms \( d_2 : R_1 \to D_2 \) and \( d_1 : L_2 \to D_1 \) such that the triangles \( R_1 D_2 H_1 \) and \( L_2 D_1 H_1 \) commute, \( m_1^{*} = c_2 \circ d_2 \models R(\rho_1, ac_{L_1}) \) and \( m_2 = c_1 \circ d_1 \models ac_{L_2} \).

Two direct derivations that are not parallel (sequentially) independent, are called **parallel (sequentially) dependent**.

**Example 4** The two direct derivations \( H_1 \leftarrow \rho_1 \) \( G \Rightarrow \rho_2 \) \( H_2 \) via \( \rho_1 = \text{AddAuthor}(\text{name}) \) and \( \rho_2 = \text{AddPublisher}(\text{name}') \) are parallel independent.

**Fact 5 (Independence with ACs implies independence without ACs)** **Parallel (sequential) independence of direct derivations implies parallel (sequential) independence of the underlying direct derivations without ACs.**

By definition, parallel and sequential independence are closely related.

**Fact 6 (Parallel and sequential independence)** Two direct derivations \( H_1 \leftarrow \rho_1, m_1 \) \( G \Rightarrow \rho_2, m_2 \) \( H_2 \) are parallel independent iff the two direct derivations \( H_1 \Rightarrow \rho_1, m_1 \) \( G \Rightarrow \rho_2, m_2 \) \( H_2 \) are sequentially independent, where \( m_1' \) is the comatch of \( \rho_1 \) in \( H_1 \).
Now we present a Local Church-Rosser Theorem for rules with application conditions.

**Theorem 1** (Local Church-Rosser Theorem) Given two parallel independent direct derivations $H_1 \leftrightarrow p_{1,m_1} \Rightarrow p_{2,m_2} \Rightarrow H_2$, there are an object $M$ and direct derivations $H_1 \Rightarrow p_{2,m_2} \Rightarrow M \leftarrow p_{1,m_1} \Rightarrow H_2$ such that $G \Rightarrow p_{1,m_1} \Rightarrow H_1 \Rightarrow p_{2,m_2} \Rightarrow M$ and $G \Rightarrow p_{2,m_2} \Rightarrow H_2 \Rightarrow p_{1,m_1} \Rightarrow M$ are sequentially independent. Given two sequentially independent direct derivations $G \Rightarrow p_{1,m_1} \Rightarrow H_1 \Rightarrow p_{2,m_2} \Rightarrow M$, there are an object $H_2$ and direct derivations $G \Rightarrow p_{2,m_2} \Rightarrow H_2 \Rightarrow p_{1,m_1} \Rightarrow M$ such that $H_1 \leftrightarrow p_{1,m_1} \Rightarrow G \Rightarrow p_{2,m_2} \Rightarrow H_2$ are parallel independent.

![Diagram](image)

**Proof.** Let $H_1 \leftrightarrow p_{1,m_1} \Rightarrow H_2$ be parallel independent. Then the underlying direct derivations without ACs are parallel independent. By the Local Church-Rosser Theorem without ACs [EEPT06], there are an object $M$ and direct derivations $H_1 \Rightarrow p_{2,m_2} \Rightarrow M \leftarrow p_{1,m_1} \Rightarrow H_2$ such that $G \Rightarrow p_{1,m_1} \Rightarrow H_1 \Rightarrow p_{2,m_2} \Rightarrow M$ and $G \Rightarrow p_{2,m_2} \Rightarrow H_2 \Rightarrow p_{1,m_1} \Rightarrow M$ are sequentially independent. By assumption, $m_{1}, m'_{1} = \text{ac}_{L_{i}}$ for $i = 1, 2$. Thus, there are direct derivations $H_1 \Rightarrow p_{2,m_2} \Rightarrow M \leftarrow p_{1,m_1} \Rightarrow H_2$ with ACs. Let $R_1 \rightarrow D_2$ and $L_2 \rightarrow D_1$ be the morphisms in the figure below. Then $R_1 \rightarrow D_2 \rightarrow H_1 = m'_{1}$ and $L_2 \rightarrow D_1 \rightarrow H_1 = m'_{2}$. By Shift-Lemma 3, $R_1 \rightarrow D_2 \rightarrow M = m''_{1} \models R(p_1, \text{ac}_{L_{1}})$ and $L_2 \rightarrow D_1 \rightarrow G = m_2 \models \text{ac}_{L_2}$ Thus, the derivation $G \Rightarrow p_{1,m_1} \Rightarrow H_1 \Rightarrow p_{2,m_2} \Rightarrow M$ is sequentially independent. Analogously, the second derivation is sequentially independent.

Vice versa, let $G \Rightarrow p_{1,m_1} \Rightarrow H_1 \Rightarrow p_{2,m_2} \Rightarrow M$ be sequentially independent. Then the underlying direct derivations without ACs are sequentially independent. By the Local Church-Rosser Theorem without ACs [EEPT06], there are an object $H_2$ and direct derivations $G \Rightarrow p_{2,m_2} \Rightarrow H_2 \Rightarrow p_{1,m_1} \Rightarrow M$ such that $H_1 \leftrightarrow p_{1,m_1} \Rightarrow G \Rightarrow p_{2,m_2} \Rightarrow H_2$ are parallel independent. By assumption, we know that $m_1, m'_{1} = \text{ac}_{L_{1}}, m_2 \models \text{ac}_{L_{2}}$ (by Shift-Lemma 3, $m''_{1} \models R(p_1, \text{ac}_{L_{1}})$ implies $m'_{1} \models \text{ac}_{L_{2}}$). Thus, $G \Rightarrow p_{2,m_2} \Rightarrow H_2 \Rightarrow p_{1,m_1} \Rightarrow M$ is a derivation with ACs. Let $L_2 \rightarrow D_1$ and $L_1 \rightarrow D_2$ in the figure below be the morphisms with $L_1 \rightarrow D_2 \rightarrow G = L_1 \rightarrow G$ and $L_2 \rightarrow D_1 \rightarrow G = L_2 \rightarrow G$. Then $L_1 \rightarrow D_2 \rightarrow H_2 = m'_1$ and $L_2 \rightarrow D_1 \rightarrow H_1 = m'_2 \models \text{ac}_{L_2}$. Thus, the direct derivations $H_1 \leftrightarrow p_{1,m_1} \Rightarrow G \Rightarrow p_{2,m_2} \Rightarrow H_2$ become parallel independent. The statement also can be proved with the help of the first statement and Fact 6.

For clarifying the notations, a sketch of a part of the proof of Local Church-Rosser Theorem for rules without ACs is given oriented at the one in [HMP01].

**Proof Sketch.** Let $H_1 \leftrightarrow p_{1,m_1} \Rightarrow p_{2,m_2} \Rightarrow H_2$ be parallel independent. Then there are morphisms $L_1 \rightarrow D_2$ and $L_2 \rightarrow D_1$ such that the triangles $L_1 D_2 G$ and $L_2 D_1 G$ in the figure below commute.

![Diagram](image)
The morphisms are used for the decomposition of the pushouts (i) into pushouts (i1),(i2) for $i = 1, \ldots, 4$.

The pushouts can be rearranged as in the figures below. Furthermore, the diagrams (22) and (42) are constructed as pushouts. Since the composition of pushouts yields pushouts, we obtain direct derivations $H_1 \Rightarrow p_1, m_1^1 M \leftarrow p_1, m_1^1 H_2$ such that the direct derivations $G \Rightarrow p_1, m_1 H_1 \Rightarrow p_2, m_2 M$ and $G \Rightarrow p_2, m_2 H_2 \Rightarrow p_1, m_1 M$ are sequentially independent.

Next, we present the construction of a parallel rule of rules with application conditions. As in [EEPT06], we have to assume that $\langle \mathcal{C}, \mathcal{M} \rangle$ has binary coproducts. The application condition of the parallel rule $\rho_1 + \rho_2$ guarantees that, whenever the parallel rule is applicable, the rules $\rho_1$ and $\rho_2$ are applicable and, after the application of $\rho_1$, the rule $\rho_2$ is applicable and, after the application of $\rho_2$, the rule $\rho_1$ is applicable.

**Definition 8** (Parallel Rule and Derivation) The parallel rule of $\rho_1$ and $\rho_2$ is the rule $\rho_1 + \rho_2 = \langle p, ac_L, ac_R \rangle$ where $p = p_1 + p_2$ is the parallel rule of $p_1$ and $p_2$, $ac_L = \text{Shift}(k_1, ac_{L_1}) \land \text{Shift}(k_2, ac_{L_2})$, and $ac_R = \text{Shift}(k_1^*, R(\rho_1, ac_{L_1})) \land \text{Shift}(k_2^*, R(\rho_2, ac_{L_2}))$. 

\[ \text{Definition 8 (Parallel Rule and Derivation)} \]
A direct derivation via a parallel rule is called parallel direct derivation or parallel derivation, for short.

**Example 5** The parallel rule (schema) of \( \text{AddAuthor}(\text{name}) \) and \( \text{AddPublisher}(\text{name}') \) is the rule (schema) with the plain rule (schema)

\[
p = \langle \begin{array}{c}
\text{authors} \\
\text{publishers}
\end{array} \leftarrow \begin{array}{c}
\text{authors} \\
\text{publishers}
\end{array} \rightarrow \begin{array}{c}
\text{authors} \\
\text{name}
\end{array} \rangle
\]

and the application conditions

\[
ac_L = \exists \left( \begin{array}{c}
\text{authors} \\
\text{name}
\end{array} \right) \land \exists \left( \begin{array}{c}
\text{publishers} \\
\text{name}'
\end{array} \right)
\]

\[
ac_R = \exists \left( \begin{array}{c}
\text{authors} \\
\text{name}'
\end{array} \right) \land \exists \left( \begin{array}{c}
\text{publishers} \\
\text{name'}
\end{array} \right)
\]

requiring that “There does not exist an author node with label \( \text{name} \)”, “There does not exist a publisher node with label \( \text{name}' \)”, “Afterwards, there do not exist two an author node with two \( \text{name} \) nodes”, and “Afterwards, there do not exist a publisher node with two \( \text{name}' \) nodes”. Here an author node is a node which is connected with the node with label authors by a directed edge. Shifting the application condition \( \text{ac}_R \) over the rule (schema) \( p \) yields the application condition \( \text{ac}_L \). Thus, the parallel rule (schema) is equivalent to the one with left application condition depicted below.

\( \text{AddAuthorPublisher}(\text{name}, \text{name}') \):

\[
\begin{array}{c}
\text{authors} \\
\text{publishers}
\end{array} \rightarrow \begin{array}{c}
\text{name}
\end{array} \leftarrow \begin{array}{c}
\text{name'}
\end{array}
\]

The connection between sequentially independent direct derivations and parallel direct derivations is expressed by the Parallelism Theorem for rules with application conditions.

**Theorem 2** (Parallelism) Given sequentially independent direct derivations \( G \Rightarrow \rho_{1,m_1} H_1 \Rightarrow \rho_{2,m_2} M \), there is a parallel derivation \( G \Rightarrow \rho_{1+\rho_{2,m}} M \). Given a parallel derivation \( G \Rightarrow \rho_{1+\rho_{2,m}} M \), there are two sequentially independent direct derivations \( G \Rightarrow \rho_{1,m_1} H_1 \Rightarrow \rho_{2,m_2} M \) and \( G \Rightarrow \rho_{2,m_2} H_2 \Rightarrow \rho_{1,m_1} M \).
Proof. By Definition 8 and Shift-Lemmas 2 and 3, we have

\[ m \models ac_L \text{ and } m^* \models ac_R \text{ iff } m_i, m'_i \models ac_L \text{ for } i = 1, 2. \]  

(1)

This may be seen as follows.

\[
\begin{align*}
  m \models ac_L & \iff m \models \text{Shift}(k_1, ac_{L_1}) \land \text{Shift}(k_2, ac_{L_2}) \\
  m_1 \models ac_{L_1} \text{ and } m_2 \models ac_{L_2} & \\
  m^* \models ac_R & \iff m^* \models \text{Shift}(k'_1, R(p_1, ac_{L_1})) \land \text{Shift}(k'_2, R(p_2, ac_{L_2})) \\
  m'_1 \models R(p_1, ac_{L_1}) \text{ and } m'_2 \models R(p_2, ac_{L_2}) \end{align*}
\]

If \( G \Rightarrow_{p_1, m_1} H_1 \Rightarrow_{p_2, m'_2} M \) is sequentially independent, then the underlying derivation without ACs is sequentially independent and, by the Parallelism Theorem without ACs [EEPT06], there is a parallel derivation \( G \Rightarrow_{p_1 + p_2, m} M \). By assumption, \( m_i, m'_i \models ac_L \text{ for } i = 1, 2 \) and, by Statement (1), \( m \models ac_L \text{ and } m^* \models ac_R \), i.e., \( G \Rightarrow_{p_1 + p_2, m} M \) satisfies ACs. If \( G \Rightarrow_{p_1 + p_2, m} M \) is a parallel derivation, then there is an underlying parallel derivation without ACs, and, by the Parallelism Theorem without ACs [EEPT06], there are sequentially independent direct derivations \( G \Rightarrow_{p_1, m_1} H_1 \Rightarrow_{p_2, m'_2} M \) and \( G \Rightarrow_{p_2, m_2} H_2 \Rightarrow_{p_1, m'_1} M \). By assumption, \( m \models ac_L \text{ and } m^* \models ac_R \) and, by Statement (1), \( m_i, m'_i \models ac_{L_i} \text{ for } i = 1, 2 \), i.e., the sequentially independent direct derivations satisfy ACs.

\[
\text{Shift operations over parallel rules can be sequentialized into a sequence of shifts over induced rules.}
\]

**Lemma 6** (Shift over Parallel Rules) For every parallel rule \( \rho = p_1 + p_2 \), every right application condition \( ac \) for \( \rho \), and \( i, j \in \{1, 2\} \) with \( i \neq j \), we have \( L(\rho, ac) \equiv \rho L(\rho^*_i, L(\rho^*_j, ac)) \) where \( \rho^*_i \) is induced by \( \rho_i \) and \( k_i \) and \( \rho^*_j \) is induced by \( \rho_j \) and \( k'_j \).
By the Parallelism Theorem, for every direct derivation \( G \Rightarrow_{\rho,m,m'} M \) there are direct derivations \( G \Rightarrow_{\rho,m} H_i \Rightarrow_{\rho,m_i} M \). By analysis arguments as in the proof of the Parallelism Theorem [EEPT06], there are direct derivations \( G \Rightarrow_{\rho,m} H_i \Rightarrow_{\rho,m'} M \) depicted below. By the Shift-Lemma 3, \( m \models L(\rho,ac) \iff m^* \models ac \iff m' \models L(\rho^*_2,ac) \), i.e., the application conditions \( L(\rho,ac) \) and \( L(\rho^*_2,L(\rho^*_1,ac)) \) are \( \rho \)-equivalent.

Finally, we present the construction of a concurrent rule for rules with application conditions.

**Definition 9** (E-concurrent Rule) Let \( \mathcal{E}' \) be a class of morphism pairs with the same codomain. Given two rules \( \rho_1 \) and \( \rho_2 \), an object \( E \) with morphisms \( e_1: R_1 \rightarrow E \) and \( e_2: R_2 \rightarrow E \) is an \( E \)-dependency relation for \( \rho_1 \) and \( \rho_2 \) if \( (e_1,e_2) \in \mathcal{E}' \) and the pushout complements (1) and (2) over \( K_1 \leftarrow R_1 \rightarrow E \) and \( K_2 \leftarrow R_2 \rightarrow E \) in the figure below exist. Given such an \( E \)-dependency relation for \( \rho_1 \) and \( \rho_2 \), the \( E \)-concurrent rule of \( \rho_1 \) and \( \rho_2 \) is the rule \( \rho_1 \ast E \rho_2 = \langle p,ac_L \rangle \) where \( p = \rho_1 \ast E \rho_2 \) is \( E \)-concurrent rule of \( \rho_1 \) and \( \rho_2 \) with pushouts (3), (4) and pullback (5), \( \rho_1^* = \langle L \leftarrow D_1 \rightarrow E \rangle \) is the rule derived by \( \rho_1 \) and \( k_1 \), and \( ac_L = \text{Shift}(k_1,ac_{L_1}) \wedge L(\rho^*_1,\text{Shift}(k_2,ac_{L_2})) \).

**Example 6** The \( E \)-concurrent rule (schema) of \( \rho_1 = \text{OrderBook}(\text{ordernr},\text{name},\text{title},\text{name}') \) and \( \rho_2 = \text{RegisterBook}(\text{ordernr},\text{catnr}) \) according to the dependency relation \( E \), being the right-hand side \( E \) of \( \rho_1 \) and the left-hand side of \( \rho_2 \), is the rule (schema)

\[
p = \begin{cases} \text{orders} & \text{name} \\ \text{catalog} & \text{name}' \end{cases} \leftarrow \begin{cases} \text{orders} & \text{name} \\ \text{catalog} & \text{name}' \end{cases} \leftarrow \begin{cases} \text{orders} & \text{title} \\ \text{catalog} & \text{catnr} \end{cases} \text{name}'
\]

with the left application condition
ac_L = \{ \begin{array}{c}
\text{orders} \\
\text{catalog} \\
\text{catnr} \\
\text{name}
\end{array}
\} \land \{ \begin{array}{c}
\text{orders} \\
\text{ordernr} \\
\text{name'}
\end{array}
\} requiring that “There does not exist a catalog node with label catnr” and “There does not exist an order node with label ordernr”. The E-concurrent rule (schema) may be depicted as follows.

Order; RegisterBook (ordernr, catnr, name, title, name'):

The non-existence of a node with label catnr guarantees that, whenever the E-concurrent rule (schema) of ρ₁ and ρ₂ is applicable, then ρ₁ with ordernr is applicable and, afterwards, ρ₂ with catnr is applicable.

For rules without ACs, the parallel rule is a special case of the concurrent rule [EEPT06]. For rules with ACs, in general, this is not the case: While the application conditions for the parallel rule must guarantee the applicability of the rules in each order, the application condition for the concurrent rule only must guarantee the applicability of the rules in the given order. Nevertheless, the parallel rule of two rules can be constructed from two concurrent rules of the rules, one for each order.

Lemma 7 (Parallel & Concurrent Rules) The parallel rule ρ₁+ρ₂ = ⟨p₁+p₂, ac_L, ac_R⟩ and the rule ⟨p₁+p₂, ac_L₁2 ac_L₁3⟩ obtained from the R₁+L₂-concurrent rule ⟨p₁+p₂, ac_L₁2⟩ of ρ₁ and ρ₂ and the R₂+L₁-concurrent rule ⟨p₂+p₁, ac_L₁3⟩ of ρ₂ and ρ₁ are equivalent.

Proof. For plain rules p₁ and p₂, the parallel rule p₁+p₂ and the concurrent rules p₁ *R₁+L₂ p₂ and p₂ *R₂+L₁ p₁ are equivalent [EEPT06]. By Lemma 5, Shift-Lemmas 2 and 3, and Lemma 4, m⁺ ≡ Shift(k⁺, R(ρ_j, ac_L_j)) ⇔ m⁺ ≡ R(ρ⁺, Shift(k_j, ac_L_j)) ⇔ m | L(ρ⁺, R(k⁺, Shift(k_j, ac_L_j))) ⇔ m | Shift(k_j, ac_L_j), i.e.,

m⁺ | Shift(k⁺, R(ρ_j, ac_L_j)) ⇔ m | Shift(k_j, ac_L_j). (2)

By Definition 8, Statement (2), and Definition 9 we have
We consider $E$-concurrent derivations via $E$-concurrent rules and $E$-related derivations via pairs of rules.

**Definition 10 (E-concurrent and E-related Derivation)** A direct derivation via an $E$-concurrent rule is called $E$-concurrent direct derivation or $E$-concurrent derivation, for short. A derivation $G \Rightarrow \rho_1 H \Rightarrow \rho_2 M$ is $E$-related if there are morphisms $E \rightarrow H$, $D_1 \rightarrow E_1$, and $D_2 \rightarrow E_2$ as shown below such that the triangles $R_1 EH$, $L_2 EH$, $K_1 D_1 E_1$, and $K_2 D_2 E_2$ in the figure below commute and the diagrams (6) and (7) are pushouts.

\[
\begin{array}{cccc}
L_1 & K_1 & R_1 & L_2 \\
\downarrow & \uparrow & \downarrow & \uparrow \\
D_1 & E & D_2 \\
G & \Rightarrow & H & \Rightarrow \\
(6) & \Rightarrow & \Rightarrow & (7) \\
E_1 & \leftarrow & E_2 & \leftarrow M
\end{array}
\]

Now we present a Concurrency Theorem for rules with application conditions.

**Theorem 3 (Concurrency)** Let $E$ be a dependency relation for $\rho_1$ and $\rho_2$. For every $E$-related derivation $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$, there is an $E$-concurrent derivation $G \Rightarrow_{\rho_1 \ast E \rho_2,m} M$. Vice versa, for every $E$-concurrent derivation $G \Rightarrow_{\rho_1 \ast E \rho_2,m} M$, there is an $E$-related derivation $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$.

**Proof.** By Definition 9 and Shift-Lemmas 2 and 3, we have

\[m_1 \models ac_{L_1} \text{ and } m_2 \models ac_{L_2} \text{ iff } m \models ac_L. \quad (3)\]

This may be seen as follows.

\[
\begin{align*}
m_1 &\models ac_{L_1} \text{ and } m_2 \models ac_{L_2} \\
\iff m &\models \text{Shift}(k_1,ac_{L_1}) \text{ and } m' \models \text{Shift}(k_2,ac_{L_2}) \\
\iff m &\models \text{Shift}(k_1,ac_{L_1}) \text{ and } m \models L(p_1',\text{Shift}(k_2,ac_{L_2})) \\
\iff m &\models \text{Shift}(k_1,ac_{L_1}) \text{ and } m \models L(p_1',\text{Shift}(k_2,ac_{L_2})) = ac_L.
\end{align*}
\]
If \( G \Rightarrow_{p_1,m_1} H \Rightarrow_{p_2,m_2} M \) is \( E \)-related, then the underlying derivation without ACs is \( E \)-related and, by the Concurrency Theorem without ACs \([EEPT06]\), there is an \( E \)-concurrent derivation \( G \Rightarrow_{p_1,p_2,m} M \). By assumption, \( m_i \models ac_L \) for \( i = 1,2 \) and, by Statement 3, \( m \models ac_L \), i.e., \( E \)-concurrent derivation \( G \Rightarrow_{p,m} M \) satisfies ACs. If \( G \Rightarrow_{p,m} M \) is an \( E \)-concurrent derivation, then the underlying direct derivation without ACs is \( E \)-concurrent, and, by the Concurrency Theorem without ACs \([EEPT06]\), there is an \( E \)-related derivation \( G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M \). By assumption, \( m \models ac_L \), and by Statement (3), \( m \models ac_L \) for \( i = 1,2 \), i.e., the \( E \)-related derivation satisfies ACs.

5 Conclusion

In this paper we present the well-known Local Church-Rosser, Parallelism, and Concurrency Theorems, known already for rules with negative application conditions \([LEPO08b]\), for rules with nested application conditions. The proofs are based on the corresponding theorems for rules without application conditions \([EEPT06]\) and two Shift-Lemmas \([HP09]\), saying that application conditions can be shifted over morphisms and rules and assume that \( \langle \mathcal{C}, \mathcal{M} \rangle \) is a weak adhesive HLR category with an \( E \)-\( \mathcal{M} \)-factorization and binary coproducts.

<table>
<thead>
<tr>
<th>statement</th>
<th>requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>Local Church-Rosser</td>
<td>Shift 2 &amp; 3</td>
</tr>
<tr>
<td>Parallelism</td>
<td>Shift 2 &amp; 3, binary coproducts</td>
</tr>
<tr>
<td>Concurrency</td>
<td>Shift 2 &amp; 3</td>
</tr>
<tr>
<td>Shift 2</td>
<td>epi-( \mathcal{M} )-factorization</td>
</tr>
<tr>
<td>Shift 3</td>
<td>–</td>
</tr>
</tbody>
</table>

Further topics might be the following:

- **Amalgamation Theorem** for rules with ACs. It would be important to generalize the Amalgamation Theorem \([BFH87, CMR+97]\) to weak adhesive HLR systems and rules with nested application conditions.

- **Embedding and Local Confluence Theorems** for rules with ACs. It would be important to generalize the Embedding and Local Confluence Theorems \([Ehr77, Ehr79, Plu93, Plu05, EEPT06, LEPO08a]\) to rules with nested application conditions.

- **Theory to rules with merging**. It would be important to generalize the theory to the case of merging as indicated in \([HMP01, EHP02]\).
Bibliography


Parallelism and Concurrency


