Electronic Communications of the EASST
Volume 26 (2010)

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 Ehrig[1], **Annegret Habel**[2] **and Leen Lambers**[3]

[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.

Theorem	without ACs	with NACs	with ACs
Local Church-Rosser	[EK76, Ehr79, EHKP91, EEPT06]	[HHT96, LEPO08b]	this paper
Parallelism	[Kre77a, Kre77b, EHKP91, EEPT06]	[HHT96, LEPO08b]	this paper
Concurrency	[ER80, Hab80, EHKP91, EEPT06]	[LEPO08b]	this paper

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.

Theorem + Shift-Lemmas for ACs $\Rightarrow$ Theorem for rules with ACs

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

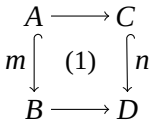
Our considerations are based on weak adhesive HLR categories, i.e. categories based on objects of many kinds of structures which are of interest in computer science and mathematics, e.g. Petri-nets, (hyper)graphs, and algebraic specifications, together with their corresponding morphisms and with specific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

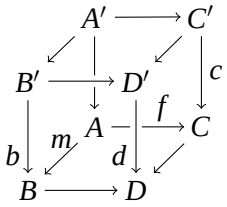
Definition 2 (Weak Adhesive HLR Category) A category $\mathscr{C}$ with a morphism class $\mathscr{M}$ is a *weak adhesive HLR category*, if the following properties hold:

1. $\mathscr{M}$ is a class of monomorphisms closed under isomorphisms, composition, and decomposition, i.e., for morphisms f and g, $f \in \mathscr{M}$, g isomorphism (or vice versa) implies $g \circ f \in \mathscr{M}$; $f, g \in \mathscr{M}$ implies $g \circ f \in \mathscr{M}$; and $g \circ f \in \mathscr{M}$, $g \in \mathscr{M}$ implies $f \in \mathscr{M}$.

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

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





Fact 1 ([EEPT06]) *The category* Graphs *with class $\mathscr{M}$ 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 $\mathscr{M}$ is the class of all monomorphisms. By [EEPT06], the category $\langle \text{PTNets}, \mathscr{M} \rangle$ of place/transition nets and the category $\langle \text{Spec}, \mathscr{M}_{strict} \rangle$ of algebraic specificatons are weak adhesive HLR, but not adhesive.*

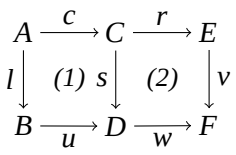
Weak adhesive HLR-categories have a number of nice properties, called HLR properties.

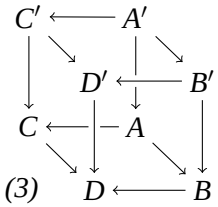
Lemma 1 (Properties of weak adhesive HLR categories [LS04, EEPT06]) *For a weak adhesive HLR-category $\langle \mathscr{C}, \mathscr{M} \rangle$, the following properties hold:*

1. *Pushouts along $\mathscr{M}$-morphisms are pullbacks.*

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

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 $\mathscr{M}$, 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.*



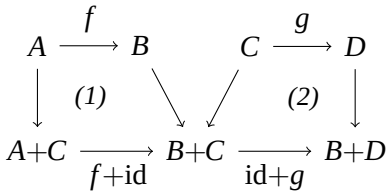


4. *Uniqueness of pushout complements. Given morphisms $c\colon A \to C$ in $\mathscr{M}$ and $s\colon C \to D$, then there is, up to isomorphism, at most one B with $l\colon A \to B$ and $u\colon B \to D$ such that diagram (1) is a pushout.*

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

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

Remark 2 (Binary coproducts) *In a weak adhesive HLR category $\langle \mathscr{C}, \mathscr{M} \rangle$ with binary coproducts, the binary coproducts are compatible with $\mathscr{M}$ in the sense that $f, g \in \mathscr{M}$ implies $f+g \in \mathscr{M}$. In fact, PO (1) in the figure below with $f \in \mathscr{M}$ implies $(f+\mathrm{id}) \in \mathscr{M}$ and PO (2) with $g \in \mathscr{M}$ implies $(\mathrm{id}+g) \in \mathscr{M}$, but now $(f+g) = (\mathrm{id}+g) \circ (f+\mathrm{id}) \in \mathscr{M}$ by closure under composition.*



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

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

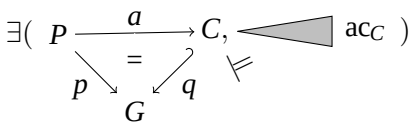
3 Rules with Application Conditions

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

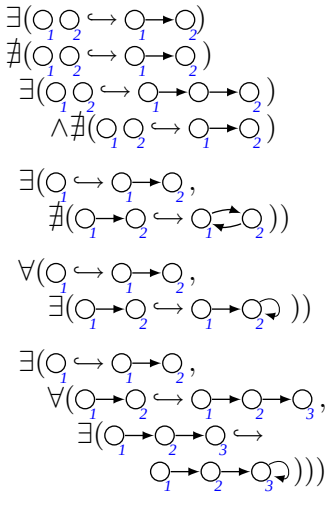
Application conditions are defined as in [HP09], Definition 4. Syntactically, application conditions may be seen as a tree of morphisms equipped with certain logical symbols such as quantifiers and connectives.

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 $\exists(a,\mathrm{ac}_C)$, where $a\colon P\to C$ 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 $\wedge_{i\in I}c_i$ are conditions over P. $\exists a$ abbreviates $\exists(a,\mathrm{true})$, $\forall(a,\mathrm{ac}_C)$ abbreviates $\neg\exists(a,\neg\mathrm{ac}_C)$, and $\nexists$ abbreviates $\neg\exists$.



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

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 $\nexists a$. Examples of nested application conditions are given below.*



There is an edge from the image of 1 to the image of 2.

There is no edge from the image of 1 to the image of 2.

There is a directed path of length 2, but not of length 1, from the image of 1 to the image of 2.

There is a proper edge outgoing from the image of 1 without edge in converse direction.

For every proper edge outgoing from the image of 1, the target has a loop.

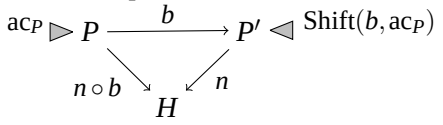
For the image of node 1, there exists an outgoing edge such that, for all edges outgoing from the target, the target has a loop.

In the presence of an $\mathscr{M}$-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 $\mathscr{M}$-morphism $I\hookrightarrow G$ satisfies $\exists(a,c)$.*

Remark 4 *In general, one could choose a satisfiability notion, i.e. a class of morphisms $\mathscr{M}'$, and require that the morphism q in Definition 4 is in $\mathscr{M}'$. Examples are $\mathscr{A}$- and $\mathscr{M}$-satisfiability [HP06] where $\mathscr{A}$ and $\mathscr{M}$ are the classes of all morphisms and all monomorphisms, respectively.*

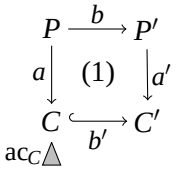
Application conditions can be shifted over morphisms into corresponding application conditions over the codomain of the morphism.

Lemma 2 (Shift of Application Conditions over Morphisms) *Let $\langle \mathscr{C}, \mathscr{M} \rangle$ be a weak adhesive HLR category with $\mathscr{E}$-$\mathscr{M}$-factorization. There is a transformation* Shift *such that, for all application conditions* ac_P *over P and all morphisms $b\colon P \to P'$, $n\colon P' \to H$, $n \circ b \models \mathrm{ac}_P \Leftrightarrow n \models \mathrm{Shift}(b, \mathrm{ac}_P)$.*



The Shift-construction is based on jointly epimorphic pairs of morphisms. A morphism pair (e_1, e_2) with $e_i\colon A_i \to B$ $(i = 1,2)$ is *jointly epimorphic* if, for all morphisms $g, h\colon B \to C$ 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 Shift-construction see [LEPO08b, HP09].

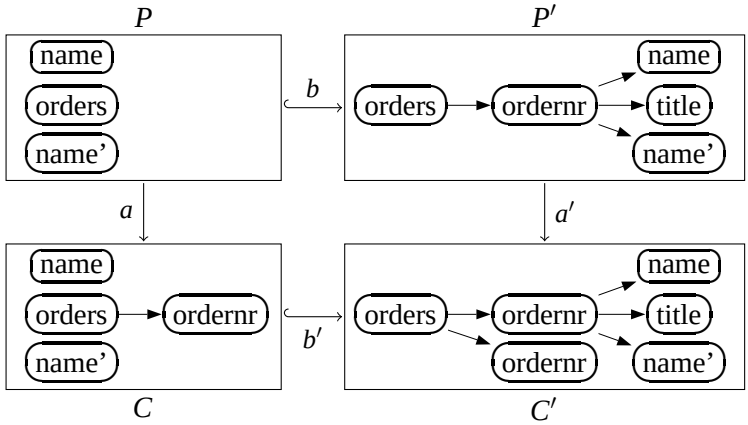
Construction The transformation Shift is inductively defined as follows:



$\mathrm{Shift}(b, \mathrm{true}) = \mathrm{true}$.
$\mathrm{Shift}(b, \exists(a, \mathrm{ac}_C)) = \bigvee_{(a',b') \in \mathscr{F}} \exists(a', \mathrm{Shift}(b', \mathrm{ac}_C))$ with
$\mathscr{F} = \{(a', b') \mid (a', b')$ jointly epimorphic, $b' \in \mathscr{M}$, (1) commutes$\}$
$\mathrm{Shift}(b, \exists(a, \mathrm{ac}_C)) = \mathrm{false}$ if $\mathscr{F}$ is empty.

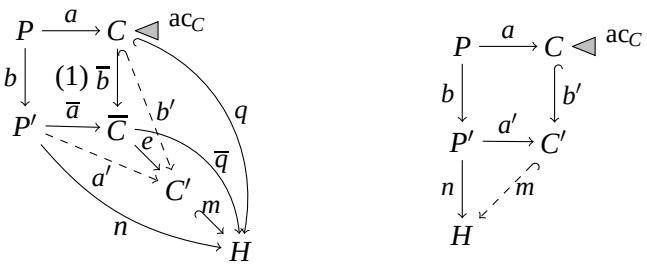
For Boolean formulas over application conditions, Shift is extended in the usual way: For application conditions $\mathrm{ac}, \mathrm{ac}_i$ with $i \in I$ (for all index sets I), $\mathrm{Shift}(b, \neg \mathrm{ac}) = \neg \mathrm{Shift}(b, \mathrm{ac})$ and $\mathrm{Shift}(b, \wedge_{i \in I} \mathrm{ac}_i)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

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



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 $\exists(a,\mathrm{ac}_C)$, we have to show $n\circ b \models \exists(a,\mathrm{ac}_C) \Leftrightarrow n \models \mathrm{Shift}(b,\exists(a,\mathrm{ac}_C))$.

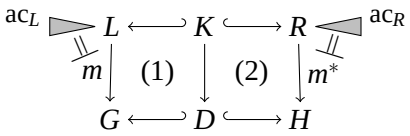
Only if. Let $n\circ b \models \exists(a,\mathrm{ac}_C)$. By definition of satisfiability, there is some $q \in \mathscr{M}$ with $q\circ a = n\circ b$ and $q \models \mathrm{ac}_C$. Let $(\overline{a},\overline{b})$ be the pushout in (1) in the left diagram below. By the universal property of pushouts,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



If. Let $n \models \mathrm{Shift}(b,\exists(a,\mathrm{ac}_C))$. Then there is some $(a',b') \in \mathscr{F}$ with $b' \in \mathscr{M}$ such that $n \models \exists(a',\mathrm{Shift}(b',\mathrm{ac}_C))$ and some $m \in \mathscr{M}$ such that $m\circ a' = n$ and $m \models \mathrm{Shift}(b',\mathrm{ac}_C)$. By inductive hypothesis, $m \models \mathrm{Shift}(b',\mathrm{ac}_C) \Leftrightarrow m\circ b' \models \mathrm{ac}_C$. Now $m\circ b' \in \mathscr{M}$, $m\circ b'\circ a = n\circ b$ (see the right diagram above), and $m\circ b' \models \mathrm{ac}_C$, i.e., $n\circ b \models \exists(a,\mathrm{ac}_C)$. □

Rules [EEHP06, HP09] are specified by a span of $\mathscr{M}$-morphisms $\langle L \hookleftarrow K \hookrightarrow R\rangle$ 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* $\rho = \langle p, \mathrm{ac}_L, \mathrm{ac}_R\rangle$ consists of a plain rule $p = \langle L \hookleftarrow K \hookrightarrow R\rangle$ with $K \hookrightarrow L$ and $K \hookrightarrow R$ in $\mathscr{M}$ 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.



A *direct derivation* consists of two pushouts (1) and (2) such that $m \models \mathrm{ac}_L$ and $m^* \models \mathrm{ac}_R$. We write $G \Rightarrow_{\rho,m,m^*} H$ and say that $m\colon L \to G$ is the match of ρ in G and $m^*\colon R \to H$ is the comatch of ρ 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 $G \Rightarrow_{\rho,m,m^*} H$.

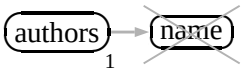
The concept of rules is completely symmetric.

Fact 3 (Inverse rule) *For every rule* $\rho = \langle p, \mathrm{ac}_L, \mathrm{ac}_R \rangle$ *with* $p = \langle L \hookleftarrow K \hookrightarrow R \rangle$, *the rule* $\rho^{-1} = \langle p^{-1}, \mathrm{ac}_R, \mathrm{ac}_L \rangle$ *with* $p^{-1} = \langle R \hookleftarrow K \hookrightarrow L \rangle$ *is the* inverse rule *of* ρ. *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) $\langle L \hookleftarrow K \hookrightarrow R \rangle$ 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 $\nexists(L \hookrightarrow L')$ is integrated in the left-hand side of a rule (schema) by crossing the part $L'-L$ out. E.g. the rule (schema) $\langle p, \mathrm{ac}_L \rangle$ with

$p = \langle$ authors $\hookleftarrow$ authors $\hookrightarrow$ authors $\to$ name $\rangle$ and

$\mathrm{ac}_L = \nexists($ authors $\hookrightarrow$ authors $\to$ name $)$ is depicted by

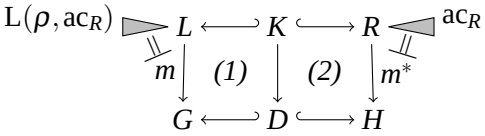


Moreover, the grey edge with labels $+,-$ in the rule (schema) RegisterBook$(\mathrm{catnr}, \mathrm{ordernr})$ in the figure below represents the conjunction of the negative application conditions "There does not exist a $+$-labelled edge" and "There does not exist a $-$-labelled edge".

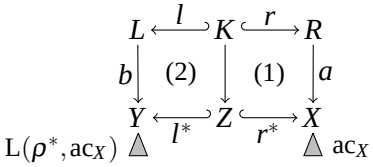
Example 2 *In the figure on the next page, rules (rule schemata) with left application conditions are given, corresponding more or less to the operations of the small library system originally investigated in [EK80].*

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* ρ *and every direct derivation* $G \Rightarrow_{\rho,m,m^*} H$, $m \models \mathrm{L}(\rho, \mathrm{ac}_R) \Leftrightarrow m^* \models \mathrm{ac}_R$ *and* $m \models \mathrm{ac}_L \Leftrightarrow m^* \models \mathrm{R}(\rho, \mathrm{ac}_L)$.

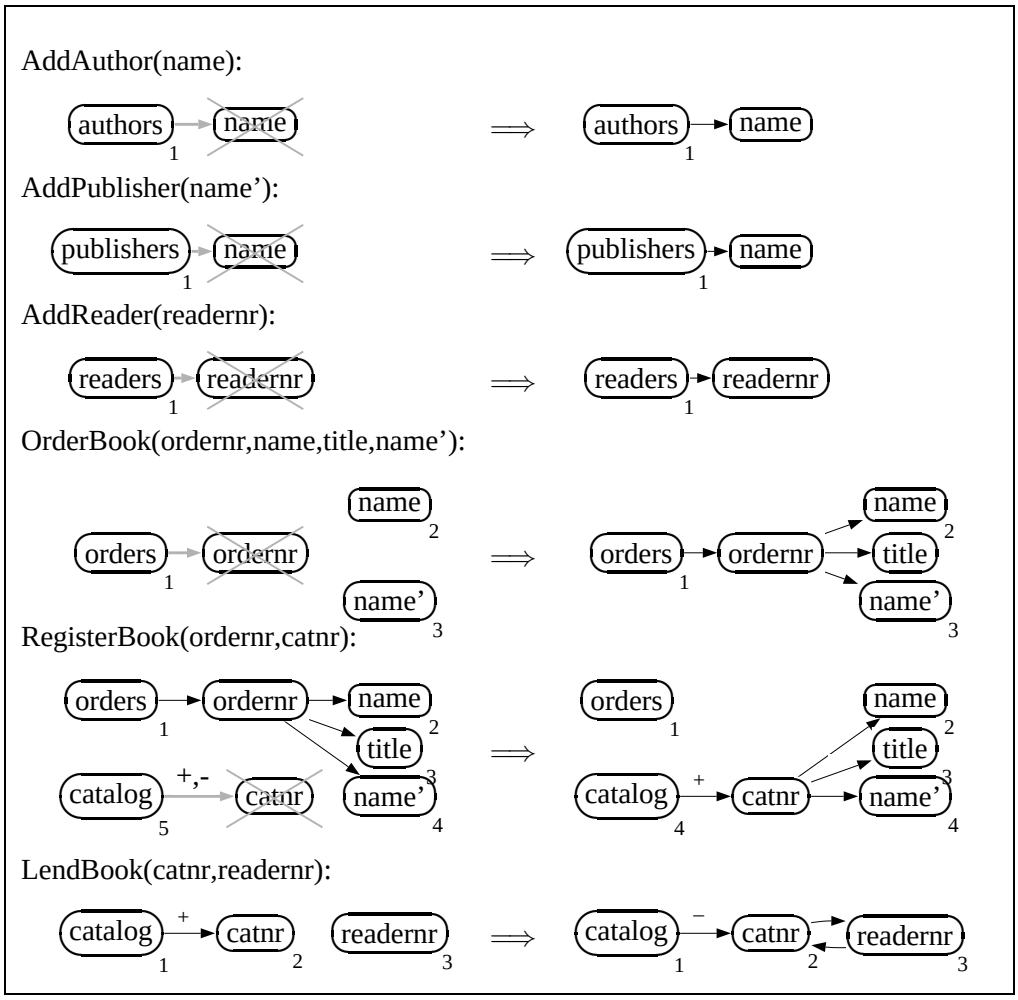


Construction The transformation L is inductively defined as follows:

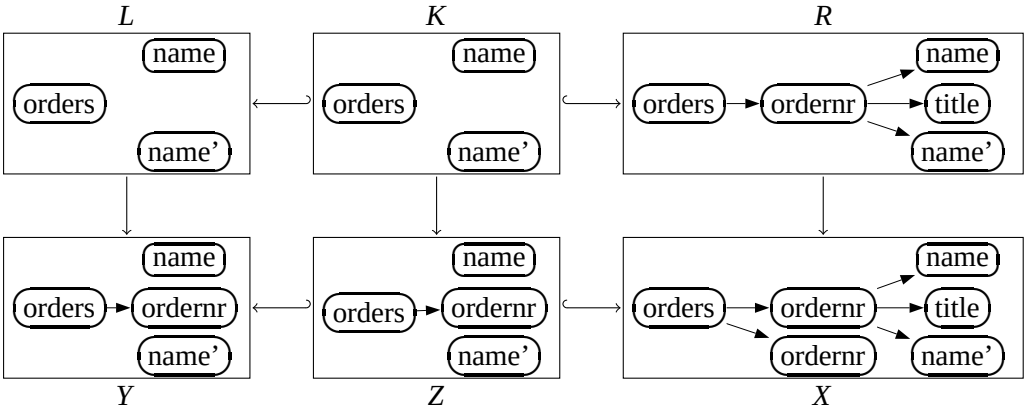


$\mathrm{L}(\rho, \mathrm{true}) = \mathrm{true}$
$\mathrm{L}(\rho, \exists(a, \mathrm{ac}_X)) = \exists(b, \mathrm{L}(\rho^*, \mathrm{ac}_X))$ if $\langle r, a \rangle$ has a pushout complement (1) and $\rho^* = \langle Y \leftarrow Z \rightarrow X \rangle$ is the derived rule by constructing the pushout (2).
$\mathrm{L}(\rho, \exists(a, \mathrm{ac}_X)) = \mathrm{false}$, otherwise.

For Boolean formulas over application conditions, L is extended in the usual way: For application conditions $\mathrm{ac}, \mathrm{ac}_i$ with $i \in I$, $\mathrm{L}(b, \neg \mathrm{ac}) = \neg \mathrm{L}(b, \mathrm{ac})$ and $\mathrm{L}(b, \wedge_{i \in I} \mathrm{ac}_i) = \wedge_{i \in I} \mathrm{L}(b, \mathrm{ac}_i)$. The transformation R is given by $\mathrm{R}(\rho, \mathrm{ac}_L) = \mathrm{L}(\rho^{-1}, \mathrm{ac}_L)$.



Example 3 *Given the rule (schema)* $\rho = \mathrm{OrderBook}(\mathrm{ordernr}, \mathrm{name}, \mathrm{title}, \mathrm{name}')$ *in the upper row of the figure below, the right application condition* $\nexists(R \to X)$ *is shifted over* ρ *into the left application condition* $\nexists(L \to Y)$.



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 ρ and ρ' are *equivalent*, denoted by $\rho \equiv \rho'$, if the relations $\Rightarrow_\rho$ and $\Rightarrow_{\rho'}$ are equal. For a rule ρ, two left (right) application conditions ac and ac$'$ are *ρ-equivalent*, denoted by $\text{ac} \equiv_\rho \text{ac}'$, if the rules obtained from ρ 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 ρ, Shift of a condition over the rule to the left and then over the rule to the right is ρ-equivalent to the original condition.

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

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

Remark 5 *In general, the application conditions* $\mathrm{R}(\rho,\mathrm{L}(\rho,\text{ac}))$ *and* ac *are not equivalent in the sense of Definition 4. E.g., for the rule* $\rho = \langle p, \text{true}, \text{ac}\rangle$ *with* $p = \langle \emptyset \hookleftarrow \emptyset \hookrightarrow \bigcirc_1 \rangle$ *and* $\text{ac} = \exists(\bigcirc_1 \to \bigcirc_1 \to \bigcirc)$, $\mathrm{L}(\rho,\neg\text{ac}) = \neg\mathrm{L}(\rho,\text{ac}) = \neg\text{false} \equiv \text{true}$ *and* $\mathrm{R}(\rho,\mathrm{L}(\rho,\neg\text{ac})) = \mathrm{R}(\rho,\text{true}) = \text{true} \not\equiv \neg\text{ac}$.

There is a nice interchange result of Shift and L saying that, for a rule ρ, the shift of a right application condition over a rule and a match is ρ-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 ρ and every application condition* ac, $\mathrm{Shift}(k,\mathrm{L}(\rho,\text{ac})) \equiv_{\rho^*} \mathrm{L}(\rho^*,\mathrm{Shift}(k^*,\text{ac}))$, *where ρ^* denotes the rule derived from ρ and k. A corresponding statement holds for* Shift *and* R.

$$\begin{array}{ccccc} L & \hookleftarrow & K & \hookrightarrow & R \\ {\scriptstyle k}\downarrow & (11) & \downarrow & (21) & \downarrow{\scriptstyle k^*} \\ L^* & \hookleftarrow & K^* & \hookrightarrow & R^* \end{array}$$

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 $l \models \mathrm{Shift}(k,\mathrm{L}(\rho,\text{ac})) \Leftrightarrow m \models \mathrm{L}(\rho,\text{ac}) \Leftrightarrow m^* \models \text{ac}_R \Leftrightarrow l^* \models \mathrm{Shift}(k^*,\text{ac}) \Leftrightarrow l \models \mathrm{L}(\rho^*,\mathrm{Shift}(k^*,\text{ac}))$.

$$\begin{array}{ccccc} L & \hookleftarrow & K & \hookrightarrow & R \\ {\scriptstyle k}\downarrow & (11) & \downarrow & (21) & \downarrow{\scriptstyle k^*} \\ L^* & \hookleftarrow & K^* & \hookrightarrow & R^* \\ {\scriptstyle l}\downarrow & (12) & \downarrow & (22) & \downarrow{\scriptstyle l^*} \\ G & \hookleftarrow & D & \hookrightarrow & H \end{array} \qquad m \text{ (left)}, \quad m^* \text{ (right)}$$

□

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 $\langle p, \mathrm{ac}_L, \mathrm{true}\rangle$ is said to be a rule with left application condition and is abbreviated by $\langle p, \mathrm{ac}_L\rangle$.

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

Proof. For a rule $\rho = \langle p, \mathrm{ac}_L, \mathrm{ac}_R\rangle$, $\mathrm{Left}(\rho) = \langle p, \mathrm{ac}_L \wedge \mathrm{L}(\rho, \mathrm{ac}_R)\rangle$. By Definition 5, Shift-Lemma 3, and the definition of Left, the rules ρ and Left(ρ) are equivalent: $G \Rightarrow_{\rho,m,m^*} H$ iff $G \Rightarrow_{p,m,m^*} H \wedge m \models \mathrm{ac}_L \wedge m^* \models \mathrm{ac}_R$ iff $G \Rightarrow_{p,m,m^*} H \wedge m \models \mathrm{ac}_L \wedge m \models \mathrm{L}(\rho, \mathrm{ac}_R)$ iff $G \Rightarrow_{p,m,m^*} H \wedge m \models \mathrm{ac}_L \wedge \mathrm{L}(\rho, \mathrm{ac}_R)$ iff $G \Rightarrow_{\mathrm{Left}(\rho),m,m^*} H$. □

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	$\Longrightarrow$	result with ACs
$\downarrow$		$\uparrow$
derivations without ACs	$\Longrightarrow$	result without ACs

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

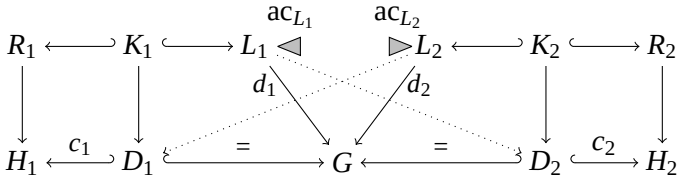
In the following, we study parallel and sequential independence of direct derivations for rules with application conditions. By Corollary 1, we may assume that the rules are rules with left application condition.

Assumption *Let $\rho_1 = \langle p_1, \mathrm{ac}_{L_1}\rangle$ and $\rho_2 = \langle p_2, \mathrm{ac}_{L_2}\rangle$ be rules with $p_i = \langle L_i \hookleftarrow K_i \hookrightarrow R_i\rangle$ for $i = 1, 2$.*

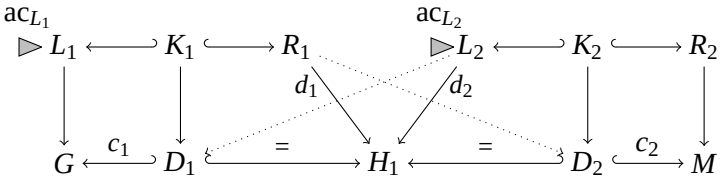
Two direct derivations are parallel (sequentially) independent if the underlying direct derivations without application conditions are parallel (sequentially) independent and the induced

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 $d_2 \colon L_1 \to D_2$ and $d_1 \colon L_2 \to D_1$ such that the triangles L_1D_2G and L_2D_1G commute, $m_1' = c_2 \circ d_2 \models \mathrm{ac}_{L_1}$, and $m_2' = c_1 \circ d_1 \models \mathrm{ac}_{L_2}$.

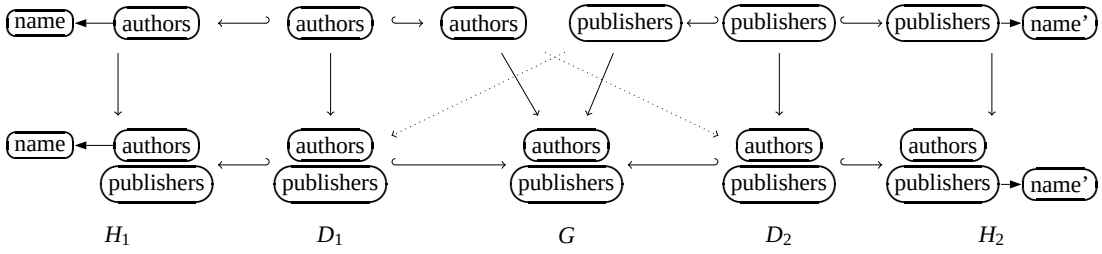


Two direct derivations $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ are *sequentially independent* if there are morphisms $d_2 \colon R_1 \to D_2$ and $d_1 \colon L_2 \to D_1$ such that the triangles $R_1D_2H_1$ and $L_2D_1H_1$ commute, $m_1'^{*} = c_2 \circ d_2 \models \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})$ and $m_2 = c_1 \circ d_1 \models \mathrm{ac}_{L_2}$.



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* $\rho_1 = \mathrm{AddAuthor}(\mathrm{name})$ *and* $\rho_2 = \mathrm{AddPublisher}(\mathrm{name}')$ *are parallel independent.*



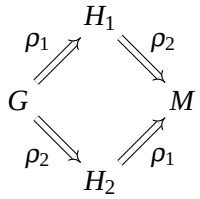
Fact 5 (Independence with ACs implies independence without ACs) *Parallel (sequential) independence of direct derivations implies parallel (sequential) independence of the underlying direct derivations without ACs.*

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* $H_1 \Rightarrow_{\rho_1^{-1},m_1^*} G \Rightarrow_{\rho_2,m_2} H_2$ *are sequentially independent, where* m_1^* *is the comatch of* ρ_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 $H_1 \Leftarrow_{\rho_1,m_1} G \Rightarrow_{\rho_2,m_2} H_2$, there are an object M and direct derivations $H_1 \Rightarrow_{\rho_2,m_2'} M \Leftarrow_{\rho_1,m_1'} H_2$ such that $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ and $G \Rightarrow_{\rho_2,m_2} H_2 \Rightarrow_{\rho_1,m_1'} M$ are sequentially independent. Given two sequentially independent direct derivations $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$, there are an object H_2 and direct derivations $G \Rightarrow_{\rho_2,m_2} H_2 \Rightarrow_{\rho_1,m_1'} M$ such that $H_1 \Leftarrow_{\rho_1,m_1} G \Rightarrow_{\rho_2,m_2} H_2$ are parallel independent.*

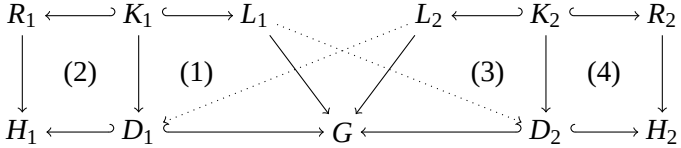


Proof. Let $H_1 \Leftarrow_{\rho_1,m_1} G \Rightarrow_{\rho_2,m_2} H_2$ be parallel independent. Then the underlying direct derivations without ACs are parallel independent. By the Local Church-Rosser Theorem without ACs [EEPT06], there are an object M and direct derivations $H_1 \Rightarrow_{p_2,m_2'} M \Leftarrow_{p_1,m_1'} H_2$ such that $G \Rightarrow_{p_1,m_1} H_1 \Rightarrow_{p_2,m_2'} M$ and $G \Rightarrow_{p_2,m_2} H_2 \Rightarrow_{p_1,m_1'} M$ are sequentially independent. By assumption, $m_i, m_i' \models \mathrm{ac}_{L_i}$ for $i = 1,2$. Thus, there are direct derivations $H_1 \Rightarrow_{\rho_2,m_2'} M \Leftarrow_{\rho_1,m_1'} H_2$ with ACs. Let $R_1 \to \overline{D_2}$ and $L_2 \to D_1$ be the morphisms in the figure below. Then $R_1 \to \overline{D_2} \to H_1 = m_1^*$ and $L_2 \to D_1 \to H_1 = m_2'$. By Shift-Lemma 3, $R_1 \to \overline{D_2} \to M = m_1'^* \models \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})$ and $L_2 \to D_1 \to G = m_2 \models \mathrm{ac}_{L_2}$. Thus, the derivation $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ is sequentially independent. Analogously, the second derivation is sequentially independent.

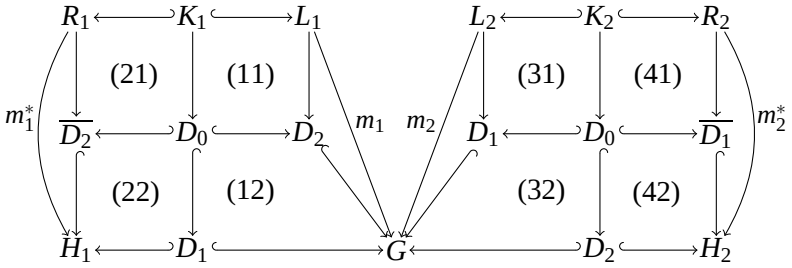
Vice versa, let $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ be sequentially independent. Then the underlying direct derivations without ACs are sequentially independent. By the Local Church-Rosser Theorem without ACs [EEPT06], there are an object H_2 and direct derivations $G \Rightarrow_{p_2,m_2} H_2 \Rightarrow_{p_1,m_1'} M$ such that $H_1 \Leftarrow_{p_1,m_1} G \Rightarrow_{p_2,m_2} H_2$ are parallel independent. By assumption, we know that $m_1, m_1' \models \mathrm{ac}_{L_1}$, $m_2 \models \mathrm{ac}_{L_2}$ (by Shift-Lemma 3, $m_1'^* \models \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})$ implies $m_1' \models \mathrm{ac}_{L_1}$). Thus, $G \Rightarrow_{\rho_2,m_2} H_2 \Rightarrow_{\rho_1,m_1'} M$ is a derivation with ACs. Let $L_2 \to D_1$ and $L_1 \to D_2$ in the figure below be the morphisms with $L_1 \to D_2 \to G = L_1 \to G$ and $L_2 \to D_1 \to G = L \to G$. Then $L_1 \to D_2 \to H_2 = m_1'$ and $L_2 \to D_1 \to H_1 = m_2' \models \mathrm{ac}_{L_2}$. Thus, the direct derivations $H_1 \Leftarrow_{p_1,m_1} G \Rightarrow_{p_2,m_2} H_2$ 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].

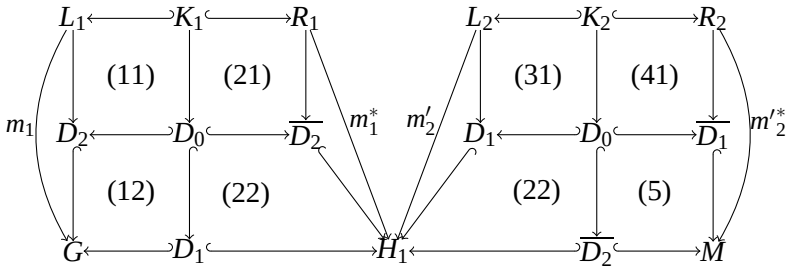
Proof Sketch. Let $H_1 \Leftarrow_{p_1,m_1} G \Rightarrow_{p_2,m_2} H_2$ be parallel independent. Then there are morphisms $L_1 \to D_2$ and $L_2 \to D_1$ such that the triangles L_1D_2G andL_2D_1G in the figure below commute.

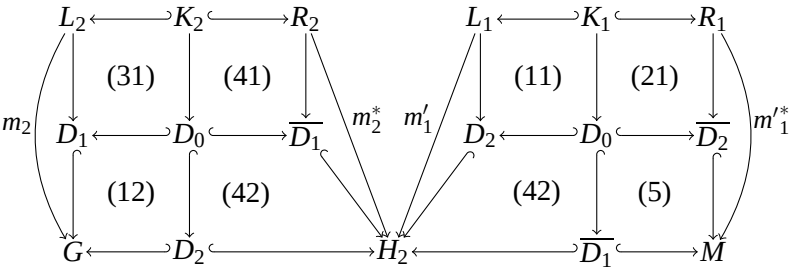


The morphisms are used for the decomposition of the pushouts (i) into pushouts (i1),(i2) for $i = 1, \ldots, 4$.



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 $H_1 \Rightarrow_{p_2,m_2'} M \Leftarrow_{p_1,m_1'} H_2$ 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.

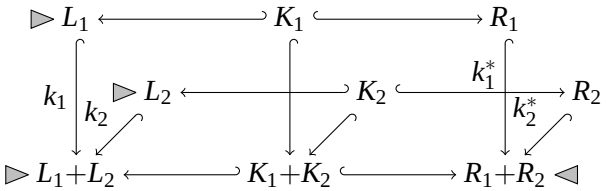




□

Next, we present the construction of a parallel rule of rules with application conditions. As in [EEPT06], we have to assume that $\langle \mathscr{C}, \mathscr{M} \rangle$ has binary coproducts. The application condition of the parallel rule $\rho_1 + \rho_2$ guarantees that, whenever the parallel rule is applicable, the rules ρ_1 and ρ_2 are applicable and, after the application of ρ_1, the rule ρ_2 is applicable and, after the application of ρ_2, the rule ρ_1 is applicable.

Definition 8 (Parallel Rule and Derivation) The *parallel rule* of ρ_1 and ρ_2 is the rule $\rho_1 + \rho_2 = \langle p, \mathrm{ac}_L, \mathrm{ac}_R \rangle$ where $p = p_1 + p_2$ is the parallel rule of p_1 and p_2, $\mathrm{ac}_L = \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{Shift}(k_2, \mathrm{ac}_{L_2})$, and $\mathrm{ac}_R = \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, \mathrm{ac}_{L_2}))$.



A direct derivation via a parallel rule is called *parallel* direct derivation or parallel derivation, for short.

Example 5 *The parallel rule (schema) of* AddAuthor(name) *and* AddPublisher(name′) *is the rule (schema) with the plain rule (schema)*

$$p = \left\langle \begin{array}{c}\text{authors}\\ \text{publishers}\end{array} \hookleftarrow \begin{array}{c}\text{authors}\\ \text{publishers}\end{array} \hookrightarrow \begin{array}{c}\text{authors}\to\text{name}\\ \text{publishers}\to\text{name'}\end{array} \right\rangle$$

and the application conditions

$$\text{ac}_L = \nexists\left(\text{publishers}\ \ \text{authors}\to\text{name}\right) \wedge \nexists\left(\text{authors}\ \ \text{publishers}\to\text{name'}\right)$$

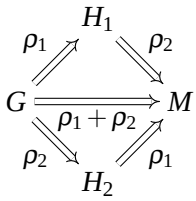
$$\text{ac}_R = \nexists\left(\begin{array}{ll}&\text{name}\\ \text{authors}&\text{name}\\ \text{publishers}&\text{name'}\end{array}\right) \wedge \nexists\left(\begin{array}{ll}\text{authors}&\text{name}\\ \text{publishers}&\text{name'}\\ &\text{name'}\end{array}\right)$$

requiring that "There does not exist an author node with label name*", "There does not exist a publisher node with label* name′*", "Afterwards, there do not exist two an author node with two* name *nodes", and "Afterwards, there do not exist a publisher node with two* 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* ac_R *over the rule (schema)* ρ *yields the application condition* ac_L*. Thus, the parallel rule (schema) is equivalent to the one with left application condition depicted below.*

AddAuthorPublisher(name, name′)*:*

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* $G \Rightarrow_{\rho_2,m_2} H_2 \Rightarrow_{\rho_1,m_1'} M$*.*

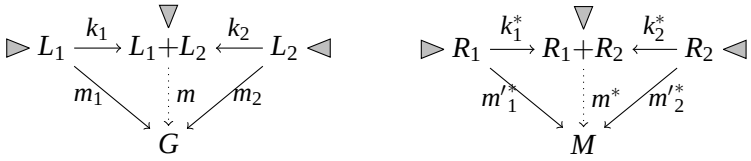


Proof. By Definition 8 and Shift-Lemmas 2 and 3, we have

$$m \models \mathrm{ac}_L \text{ and } m^* \models \mathrm{ac}_R \text{ iff } m_i, m_i' \models \mathrm{ac}_{L_i} \text{ for } i = 1,2. \tag{1}$$

This may be seen as follows.

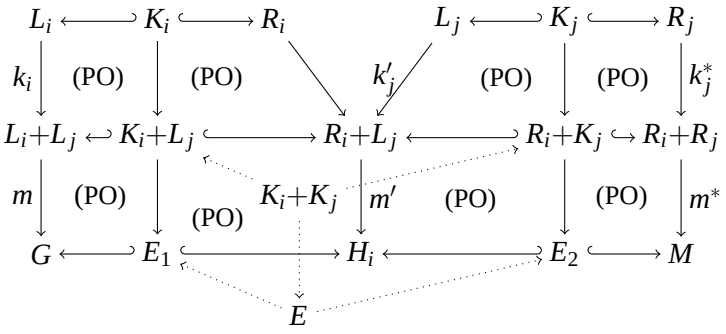
$$\begin{array}{lcl} m \models \mathrm{ac}_L & \Leftrightarrow & m \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{Shift}(k_2, \mathrm{ac}_{L_2}) \\ & \Leftrightarrow & m_1 \models \mathrm{ac}_{L_1} \text{ and } m_2 \models \mathrm{ac}_{L_2} \\ m^* \models \mathrm{ac}_R & \Leftrightarrow & m^* \models \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, \mathrm{ac}_{L_2})) \\ & \Leftrightarrow & m'^*_1 \models \mathrm{R}(\rho_1, \mathrm{ac}_{L_1}) \text{ and } m'^*_2 \models \mathrm{R}(\rho_2, \mathrm{ac}_{L_2}) \\ & \Leftrightarrow & m_1' \models \mathrm{ac}_{L_1} \text{ and } m_2' \models \mathrm{ac}_{L_2} \end{array}$$



If $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ 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, $m_i, m_i' \models \mathrm{ac}_{L_i}$ for $i = 1,2$ and, by Statement (1), $m \models \mathrm{ac}_L$ and $m^* \models \mathrm{ac}_R$, i.e., $G \Rightarrow_{p_1+p_2,m} M$ satisfies ACs. If $G \Rightarrow_{\rho_1+\rho_2,m} M$ 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, $m \models \mathrm{ac}_L$ and $m^* \models \mathrm{ac}_R$ and, by Statement (1), $m_i, m_i' \models \mathrm{ac}_{L_i}$ for $i = 1,2$, i.e., the sequentially independent direct derivations satisfy ACs. □

Shift operations over parallel rules can be sequentialized into a sequence of shifts over induced rules.

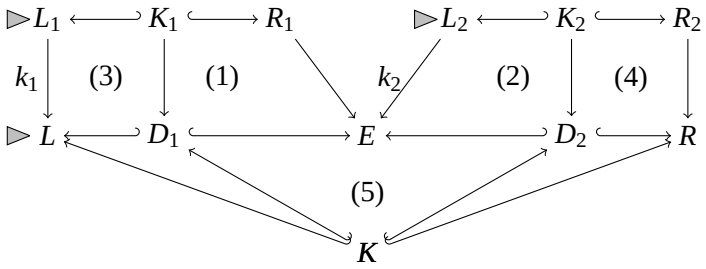
Lemma 6 (Shift over Parallel Rules) *For every parallel rule $\rho = \rho_1 + \rho_2$, every right application condition* ac *for ρ, and $i, j \in \{1,2\}$ with $i \neq j$, we have* $\mathrm{L}(\rho, \mathrm{ac}) \equiv_\rho \mathrm{L}(\rho_i^*, \mathrm{L}(\rho_j^*, \mathrm{ac}))$ *where ρ_i^* is induced by ρ_i and k_i and ρ_j^* is induced by ρ_j and k_j'.*



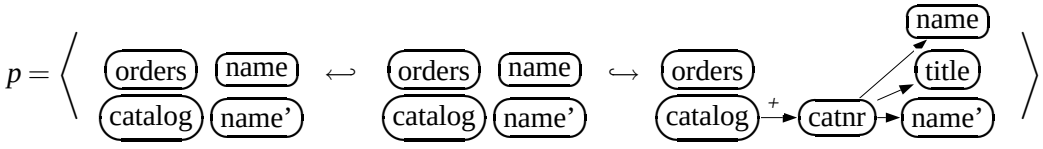
Proof. By the Parallelism Theorem, for every direct derivation $G \Rightarrow_{\rho,m,m^*} M$ there are direct derivations $G \Rightarrow_{\rho_i,m_i} H_i \Rightarrow_{\rho_j,m_j} M$. By analysis arguments as in the proof of the Parallelism Theorem [EEPT06], there are direct derivations $G \Rightarrow_{\rho_i^*,m} H_i \Rightarrow_{\rho_j^*,m'} M$ depicted below. By the Shift-Lemma 3, $m \models \mathrm{L}(\rho,\mathrm{ac}) \Leftrightarrow m^* \models \mathrm{ac} \Leftrightarrow m' \models \mathrm{L}(\rho_j^*,\mathrm{ac}) \Leftrightarrow m \models \mathrm{L}(\rho_i^*,\mathrm{L}(\rho_j^*,\mathrm{ac}))$, i.e, the application conditions $\mathrm{L}(\rho,\mathrm{ac})$ and $\mathrm{L}(\rho_i^*,\mathrm{L}(\rho_j^*,\mathrm{ac}))$ are ρ-equivalent. □

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

Definition 9 (E-concurrent Rule) Let $\mathscr{E}'$ be a class of morphism pairs with the same codomain. Given two rules ρ_1 and ρ_2, an object E with morphisms $e_1\colon R_1 \to E$ and $e_2\colon L_2 \to E$ is an *E-dependency relation* for ρ_1 and ρ_2 if $(e_1,e_2) \in \mathscr{E}'$ and the pushout complements (1) and (2) over $K_1 \hookrightarrow R_1 \to E$ and $K_2 \hookrightarrow L_2 \to E$ in the figure below exist. Given such an E-dependency relation for ρ_1 and ρ_2, the *E-concurrent rule* of ρ_1 and ρ_2 is the rule $\rho_1 *_E \rho_2 = \langle p, \mathrm{ac}_L \rangle$ where $p = p_1 *_E p_2$ is E-concurrent rule of p_1 and p_2 with pushouts (3), (4) and pullback (5), $\rho_1^* = \langle L \hookleftarrow D_1 \hookrightarrow E \rangle$ is the rule derived by ρ_1 and k_1, and $\mathrm{ac}_L = \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{L}(\rho_1^*, \mathrm{Shift}(k_2, \mathrm{ac}_{L_2}))$.



Example 6 *The E-concurrent rule (schema) of* $\rho_1 = \mathrm{OrderBook}(\mathrm{ordernr},\mathrm{name},\mathrm{title},\mathrm{name}')$ *and* $\rho_2 = \mathrm{RegisterBook}(\mathrm{ordernr},\mathrm{catnr})$ *according to the dependency relation E, being the right-hand side E of ρ_1 and the left-hand side of ρ_2, is the rule (schema)*

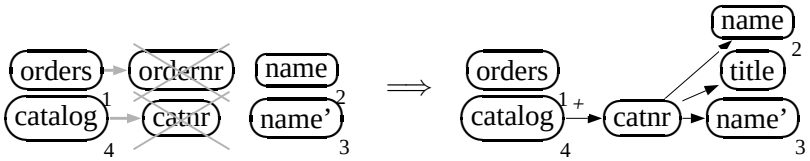


with the left application condition

$$\mathrm{ac}_L = \nexists\left(\begin{array}{c}\text{orders}\\ \text{catalog}\end{array}\rightarrow\text{catnr}\quad\begin{array}{c}\text{name}\\ \text{name'}\end{array}\right)\wedge\nexists\left(\begin{array}{c}\text{orders}\\ \text{catalog}\end{array}\rightarrow\text{ordernr}\quad\begin{array}{c}\text{name}\\ \text{name'}\end{array}\right)$$

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.*

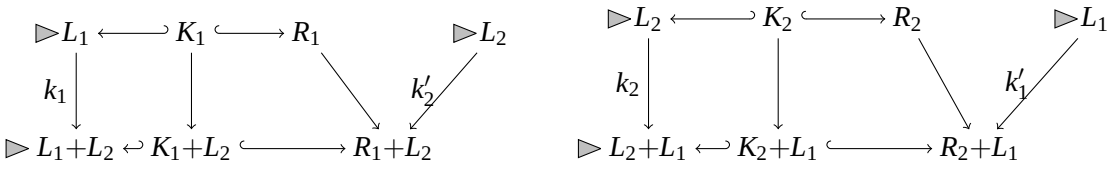
Order; RegisterBook(ordernr, catnr, name, title, name′)*:*



The non-existence of a node with label catnr *guarantees that, whenever the E-concurrent rule (schema) of ρ_1 and ρ_2 is applicable, then ρ_1 with* ordernr *is applicable and, afterwards, ρ_2 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 & Concurent Rules) *The parallel rule $\rho_1+\rho_2 = \langle p_1+p_2, \mathrm{ac}_L, \mathrm{ac}_R\rangle$ and the rule $\langle p_1+p_2, \mathrm{ac}_{L_{12}} \wedge \mathrm{ac}_{L_{21}}\rangle$ obtained from the R_1+L_2-concurrent rule $\langle p_1+p_2, \mathrm{ac}_{L_{12}}\rangle$ of ρ_1 and ρ_2 and the R_2+L_1-concurrent rule $\langle p_2+p_1, \mathrm{ac}_{L_{21}}\rangle$ of ρ_2 and ρ_1 are equivalent.*



Proof. For plain rules p_1 and p_2, the parallel rule p_1+p_2 and the concurrent rules $p_1 *_{R_1+L_2} p_2$ and $p_2 *_{R_2+L_1} p_1$ are equivalent [EEPT06]. By Lemma 5, Shift-Lemmas 2 and 3, and Lemma 4, $m^* \models \mathrm{Shift}(k_j^*, \mathrm{R}(\rho_j, \mathrm{ac}_{L_j})) \Leftrightarrow m^* \models \mathrm{R}(\rho_j^*, \mathrm{Shift}(k_j, \mathrm{ac}_{L_j})) \Leftrightarrow m \models \mathrm{L}(\rho_j^*, \mathrm{R}(\rho_j^*, \mathrm{Shift}(k_j, \mathrm{ac}_{L_j})) \Leftrightarrow m \models \mathrm{Shift}(k_j, \mathrm{ac}_{L_j}))$, i.e.,

$$m^* \models \mathrm{Shift}(k_j^*, \mathrm{R}(\rho_j, \mathrm{ac}_{L_j})) \Leftrightarrow m \models \mathrm{Shift}(k_j, \mathrm{ac}_{L_j})). \tag{2}$$

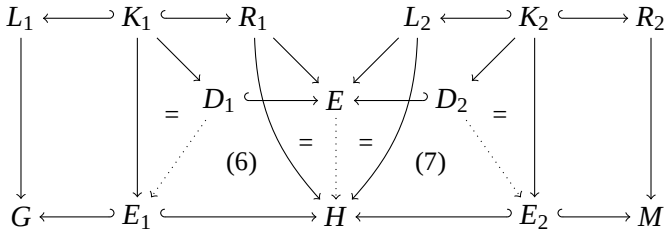
By Definition 8, Statement (2), and Definition 9 we have

$$\begin{array}{lll} & m & \models \mathrm{ac}_L \text{ and } m^* \models \mathrm{ac}_R \\ \Leftrightarrow & m & \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{Shift}(k_2, \mathrm{ac}_{L_2}) \text{ and} \\ & m^* & \models \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, \mathrm{ac}_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, \mathrm{ac}_{L_2})) \\ \Leftrightarrow & m & \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{L}(\rho_1^*, \mathrm{Shift}(k_2', \mathrm{ac}_{L_2})) \text{ and} \\ & m & \models \mathrm{Shift}(k_2, \mathrm{ac}_{L_2}) \wedge \mathrm{L}(\rho_2^*, \mathrm{Shift}(k_1', \mathrm{ac}_{L_1})) \\ \Leftrightarrow & m & \models \mathrm{ac}_{L_{12}} \wedge \mathrm{ac}_{L_{21}} \end{array}$$

i.e., the parallel rule and the rule constructed from the concurrent rules are equivalent. □

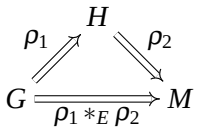
We consider E-concurrent derivations via E-concurrent rules and E-related derivations via pairs of rules.

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 $E \to H$, $D_1 \to E_1$, and $D_2 \to E_2$ as shown below such that the triangles R_1EH, L_2EH, $K_1D_1E_1$, and $K_2D_2E_2$ in the figure below commute and the diagrams (6) and (7) are pushouts.



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

Theorem 3 (Concurrency) *Let E be a dependency relation for ρ_1 and ρ_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 $G \Rightarrow_{\rho_1 *_E \rho_2, m} M$. Vice versa, for every E-concurrent derivation $G \Rightarrow_{\rho_1 *_E \rho_2, m} M$, there is an E-related derivation $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$.*



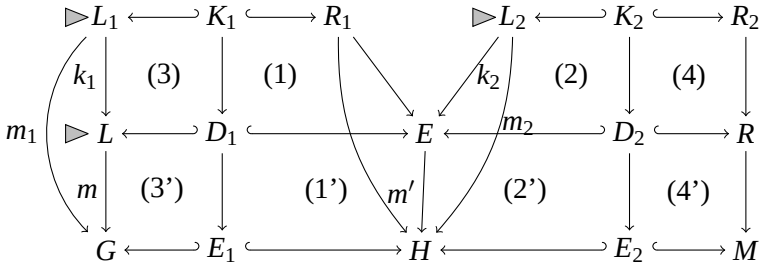
Proof. By Definition 9 and Shift-Lemmas 2 and 3, we have

$$m_1 \models \mathrm{ac}_{L_1} \text{ and } m_2 \models \mathrm{ac}_{L_2} \text{ iff } m \models \mathrm{ac}_L. \tag{3}$$

This may be seen as follows.

$$\begin{array}{ll} & m_1 \models \mathrm{ac}_{L_1} \text{ and } m_2 \models \mathrm{ac}_{L_2} \\ \Leftrightarrow & m \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \text{ and } m' \models \mathrm{Shift}(k_2, \mathrm{ac}_{L_2}) \\ \Leftrightarrow & m \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \text{ and } m \models \mathrm{L}(p_1^*, \mathrm{Shift}(k_2, \mathrm{ac}_{L_2})) \\ \Leftrightarrow & m \models \mathrm{Shift}(k_1, \mathrm{ac}_{L_1}) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(k_2, \mathrm{ac}_{L_2})) = \mathrm{ac}_L. \end{array}$$

If $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$ 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, $m_i \models \mathrm{ac}_{L_i}$ for $i = 1,2$ and, by Statement 3, $m \models \mathrm{ac}_L$, i.e., E-concurrent derivation $G \Rightarrow_{\rho,m} M$ satisfies ACs. If $G \Rightarrow_{\rho,m} M$ 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 $G \Rightarrow_{p_1,m_1} H \Rightarrow_{p_2,m_2} M$. By assumption, $m \models \mathrm{ac}_L$, and by Statement (3), $m_i \models \mathrm{ac}_{L_i}$ 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 $\langle \mathscr{C}, \mathscr{M} \rangle$ is a weak adhesive HLR category with an $\mathscr{E}$-$\mathscr{M}$-factorization and binary coproducts.

statement	requirements
Local Church-Rosser	Shift 2 & 3
Parallelism	Shift 2 & 3, binary coproducts
Concurrency	Shift 2 & 3
Shift 2	epi-$\mathscr{M}$-factorization
Shift 3	–

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

[BFH87] P. Boehm, H.-R. Fonio, A. Habel. Amalgamation of Graph Transformations: A Synchronization Mechanism. *Journal of Computer and System Sciences* 34:377–408, 1987.

[CMR+97] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, M. 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, pp. 163–245. World Scientific, 1997.

[EEHP06] H. Ehrig, K. Ehrig, A. Habel, K.-H. Pennemann. Theory of Constraints and Application Conditions: From Graphs to High-Level Structures. *Fundamenta Informaticae* 74(1):135–166, 2006.

[EEKR99] H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg (eds.). *Handbook of Graph Grammars and Computing by Graph Transformation*. Volume 2: Applications, Languages and Tools. World Scientific, 1999.

[EEPT06] H. Ehrig, K. Ehrig, U. Prange, G. Taentzer. *Fundamentals of Algebraic Graph Transformation*. EATCS Monographs of Theoretical Computer Science. Springer, Berlin, 2006.

[EH86] H. Ehrig, A. Habel. Graph Grammars with Application Conditions. In Rozenberg and Salomaa (eds.), *The Book of L*. Pp. 87–100. Springer, 1986.

[EHKP91] H. Ehrig, A. Habel, H.-J. Kreowski, F. Parisi-Presicce. Parallelism and Concurrency in High Level Replacement Systems. *Mathematical Structures in Computer Science* 1:361–404, 1991.

[EHP02] H. Ehrig, A. Habel, F. Parisi-Presicce. Basic Results for Two Types of High-Level Replacement Systems. In *General Theory of Graph Transformations (GETGRATS)*. ENTCS 51. 2002.

[Ehr77] H. Ehrig. Embedding Theorems in the Algebraic Theory of Graph Grammars. In *Fundamentals of Computation Theory*. LNCS 56, pp. 245–255. 1977.

[Ehr79] H. Ehrig. Introduction to the Algebraic Theory of Graph Grammars. In *Graph-Grammars and Their Application to Computer Science and Biology*. LNCS 73, pp. 1–69. 1979.

[EK76] H. Ehrig, H.-J. Kreowski. Parallelism of Manipulations in Multidimensional Information Structures. In *Mathematical Foundations of Computer Science*. LNCS 45, pp. 284–293. 1976.

[EK80] H. Ehrig, H.-J. Kreowski. Applications of Graph Grammar Theory to Consistency, Synchronization and Scheduling in Data Base Systems. *Information Systems* 5:225–238, 1980.

[EKMR99] H. Ehrig, H.-J. Kreowski, U. Montanari, G. Rozenberg (eds.). *Handbook of Graph Grammars and Computing by Graph Transformation*. Volume 3: Concurrency, Parallelism, and Distribution. World Scientific, 1999.

[ER80] H. Ehrig, B. K. Rosen. Parallelism and Concurrency of Graph Manipulations. *Theoretical Computer Science* 11:247–275, 1980.

[Hab80] A. Habel. Concurrency in Graph-Grammatiken. Technical report 80-11, Technical University of Berlin, 1980.

[HHT96] A. Habel, R. Heckel, G. Taentzer. Graph Grammars with Negative Application Conditions. *Fundamenta Informaticae* 26:287–313, 1996.

[HMP01] A. Habel, J. Müller, D. Plump. Double-Pushout Graph Transformation Revisited. *Mathematical Structures in Computer Science* 11(5):637–688, 2001.

[HP05] A. Habel, K.-H. Pennemann. Nested Constraints and Application Conditions for High-Level Structures. In *Formal Methods in Software and System Modeling*. LNCS 3393, pp. 293–308. 2005.

[HP06] A. Habel, K.-H. Pennemann. Satisfiability of High-Level Conditions. In *Graph Transformations (ICGT 2006)*. LNCS 4178, pp. 430–444. 2006.

[HP09] A. Habel, K.-H. Pennemann. Correctness of High-Level Transformation Systems Relative to Nested Conditions. *Mathematical Structures in Computer Science* 19:245–296, 2009.

[HW95] R. Heckel, A. Wagner. Ensuring Consistency of Conditional Graph Grammars — A Constructive Approach. In *Joint COMPUGRAPH/SEMAGRAPH Workshop on Graph Rewriting and Computation (SEGRAGRA'95)*. ENTCS 2, pp. 95–104. 1995.

[KMP05] M. Koch, L. V. Mancini, F. Parisi-Presicce. Graph-based Specification of Access Control Policies. *Journal of Computer and System Sciences* 71:1–33, 2005.

[Kre77a] H.-J. Kreowski. *Manipulationen von Graphmanipulationen*. PhD thesis, Technical University of Berlin, 1977.

[Kre77b] H.-J. Kreowski. Transformations of Derivation Sequences in Graph Grammars. In *Fundamentals of Computation Theory*. LNCS 56, pp. 275–286. 1977.

[LEO06] L. Lambers, H. Ehrig, F. Orejas. Conflict detection for graph transformation with negative application conditions. In *Graph Transformations (ICGT 2006)*. LNCS 4178, pp. 61–76. 2006.

[LEPO08a] L. Lambers, H. Ehrig, U. Prange, F. Orejas. Embedding and Confluence of Graph Transformations with Negative Application Conditions. In *Graph Transformations (ICGT 2008)*. LNCS 5214, pp. 162–177. 2008.

[LEPO08b] L. Lambers, H. Ehrig, U. Prange, F. Orejas. Parallelism and Concurrency in Adhesive High-Level Replacement Systems with Negative Application Conditions. In *Workshop on Applied and Computational Category Theory (ACCAT 2007)*. ENTCS 2003, pp. 43–66. Elsevier, 2008.

[LS04] S. Lack, P. Sobociński. Adhesive Categories. In *Foundations of Software Science and Computation Structures (FOSSACS'04)*. LNCS 2987, pp. 273–288. 2004.

[LS05] S. Lack, P. Sobociński. Adhesive and Quasiadhesive Categories. *Theoretical Informatics and Application* 39(2):511–546, 2005.

[Plu93] D. Plump. Hypergraph Rewriting: Critical Pairs and Undecidability of Confluence. In *Term Graph Rewriting: Theory and Practice*. Pp. 201–213. John Wiley, 1993.

[Plu05] D. Plump. Confluence of Graph Transformation Revisited. In *Processes, Terms and Cycles: Steps on the Road to Infinity: Essays Dedicated to Jan Willem Klop on the Occasion of His 60th Birthday*. LNCS 3838, pp. 280–308. 2005.

[Roz97] G. Rozenberg (ed.). *Handbook of Graph Grammars and Computing by Graph Transformation*. Volume 1: Foundations. World Scientific, 1997.