Parallelism and Concurrency Theorems for Rules with Nested Application Conditions

Hartmut Ehrig, Annegret Habel, and Leen Lambers

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 [1,2,3] and were generalized to high-level replacement (HLR) systems [4] and weak adhesive HLR systems [5,6]. Application conditions restrict the applicability of a rule. Originally, they were defined in [7], specialized to negative application conditions (NACs) [8], and generalized to nested application conditions (ACs) [9].

The Local Church-Rosser, Parallelism, and Concurrency Theorems are well-known theorems for graph replacement systems on rules without application conditions [10,11,12,13,14,15] and are generalized to high-level replacement (HLR) systems [4] and rules with negative application conditions [16]. Nested application conditions (ACs) were introduced in [9] and intensively studied in [17]. They generalize the well-known negative application conditions (NACs) in the sense of [8,16] and 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	[10,13,4,6]	[8,16]	this paper
Parallelism	[11,12,4,6]	[8,16]	this paper
Concurrency	[14,15,4,6]	[16]	this paper

The proofs of the theorems are based on the corresponding theorems for weak adhesive HLR systems on rules without application conditions in [6] and facts on nested application conditions in [17], saying that application conditions can be shifted over morphisms and rules.

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

F. Drewes, A. Habel, B. Hoffmann, D. Plump (Eds.): Manipulation of Graphs, Algebras and Pictures. Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday, pp. 109–133, 2009.

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 [13,18] and generalize them to high-level structures [4]. 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.

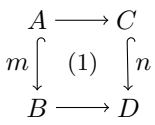
Definition 1 (graphs and graph morphisms). Let $\mathrm{C} = \langle \mathrm{C_V}, \mathrm{C_E} \rangle$ be a fixed, finite label alphabet. A *graph* over C is a system $G = (\mathrm{V}_G, \mathrm{E}_G, \mathrm{s}_G, \mathrm{t}_G, \mathrm{l}_G, \mathrm{m}_G)$ consisting of two finite sets V_G and E_G 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, $\mathrm{s}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{s}_G$, $\mathrm{t}_H \circ g_\mathrm{E} = g_\mathrm{V} \circ \mathrm{t}_G$, $\mathrm{l}_H \circ g_\mathrm{V} = \mathrm{l}_G$, and $\mathrm{m}_H \circ g_\mathrm{E} = \mathrm{m}_G$. A morphism g is *injective* (*surjective*) if g_V and g_E are injective (surjective), and an *isomorphism* if it is both injective and surjective. The *composition* $h \circ g$ of g with a morphism $h \colon H \to M$ consists of the composed functions $h_\mathrm{V} \circ g_\mathrm{V}$ and $h_\mathrm{E} \circ g_\mathrm{E}$.

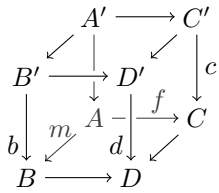
Our considerations are based on weak adhesive HLR categories, i.e. categories based on objects of many kinds of structures which are of interest in computer science and mathematics, e.g. Petri-nets, (hyper)graphs, and algebraic specifications, together with their corresponding morphisms and with specific properties. Readers interested in the category-theoretic background of these concepts may consult e.g. [6].

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 $g \circ f \colon f \in \mathcal{M}$, 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.



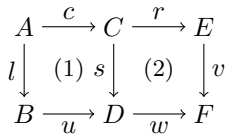


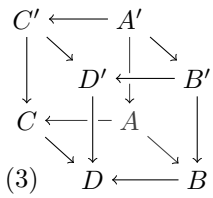
Fact 1. The category $\langle \text{Graphs}, \text{Inj} \rangle$ of graphs with class Inj of all injective graph morphisms is a weak adhesive HLR category [6].

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 [6]. Weak adhesive HLR-categories have a number of nice properties, called HLR properties [4].

Fact 2 (properties of weak adhesive HLR categories [19,6]). 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.



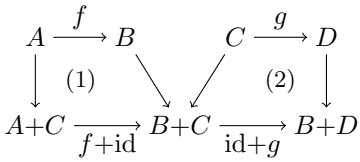


4. Uniqueness of pushout complements. Given morphisms $c\colon A \to C$ in $\mathcal{M}$ and $s\colon C \to D$, then there is, up to isomorphism, at most one B with $l\colon A \to B$ and $u\colon B \to D$ such that diagram (1) is a pushout.

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

Definition 3 (epi-$\mathcal{M}$ factorization). A weak adhesive HLR category $\langle \mathcal{C}, \mathcal{M} \rangle$ has an *epi-$\mathcal{M}$ factorization* if, for every morphism, there is an epi-mono factorization with monomorphism in $\mathcal{M}$ and this decomposition is unique up to isomorphism.

Remark 1 (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, PO (1) in the figure below with $f \in \mathcal{M}$ implies $(f+\mathrm{id}) \in \mathcal{M}$ and PO (2) with $g \in \mathcal{M}$ implies $(\mathrm{id}+g) \in \mathcal{M}$, but now $(f+g) = (\mathrm{id}+g) \circ (f+\mathrm{id}) \in \mathcal{M}$ by closure under composition.[1em]



[1em] For the category $\langle \mathrm{Graphs}, \mathrm{Inj} \rangle$ of graphs with class Inj of all injective graph morphisms, these specific properties are satisfied.

Fact 3. $\langle \mathrm{Graphs}, \mathrm{Inj} \rangle$ has an epi-Inj factorization and binary coproducts [6].

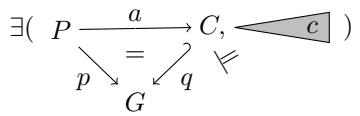
3 Conditions and rules

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

Assumption 1. We assume that $\langle \mathcal{C}, \mathcal{M} \rangle$ is a weak adhesive HLR category with an epi-$\mathcal{M}$ factorization and binary coproducts.

Conditions are defined as in [9,17]. Syntactically, the conditions may be seen as a tree of morphisms equipped with certain logical symbols such as quantifiers and connectives.

Definition 4 (conditions). A *(nested) condition* over an object P is of the form true or $\exists(a, c)$, where $a\colon P \to C$ is a morphism and c is a 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 $\exists(a, \mathrm{true})$, $\forall(a, c)$ abbreviates $\neg\exists(a, \neg c)$. Every morphism *satisfies* true. A morphism $p\colon P \to G$ *satisfies* a condition $\exists(a, c)$ if there exists a morphism q in $\mathcal{M}$ such that $q \circ a = p$ and $q \models c$.



The satisfaction of conditions over P by morphisms with domain P is extended to Boolean formulas over conditions in the usual way. We write $p \models c$ to denote that the morphism p satisfies c. Two conditions c and c' over P are *equivalent*, denoted by $c \equiv c'$, if for all morphisms p with domain P, $p \models c$ iff $p \models c'$.

Remark 2. The definition of conditions generalizes those in [8,20,21,5]. In the context of rules, conditions are also called *application conditions*. Negative application conditions [8,16] correspond to nested application conditions of the form $\nexists a$. Examples of nested application conditions are given in Figure 1.

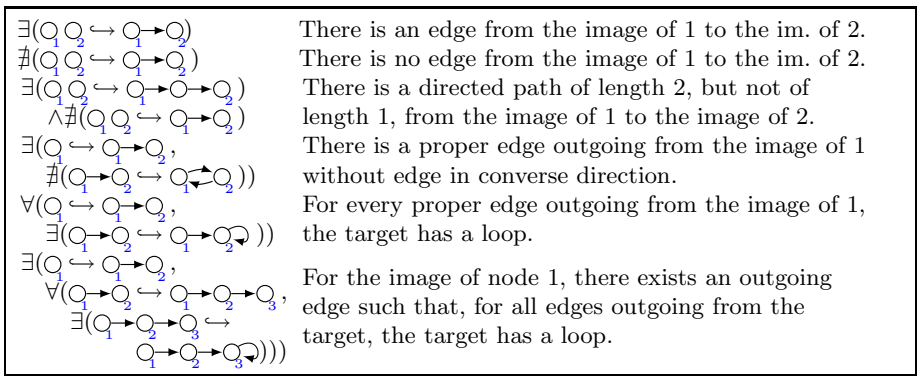


Fig. 1. Nested application conditions

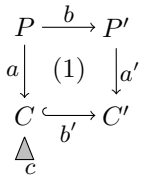
In the presence of an $\mathcal{M}$-initial object I [17], conditions $\exists(a, c)$ with morphism $a\colon I \to C$ can be used to define *constraints* for objects G, namely G satisfies $\exists(a, c)$ if the initial morphism i_G satisfies $\exists(a, c)$.

Remark 3. 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 $\mathcal{A}$- and $\mathcal{M}$-satisfiability [22] where $\mathcal{A}$ and $\mathcal{M}$ are the classes of all morphisms and all monomorphisms, respectively.

Conditions can be shifted over morphisms into corresponding conditions over the codomain of the morphism. We present a Shift-construction 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 $e_2(a_2) = b$.

Definition 5 (shift of conditions over morphisms). Let $\langle \mathcal{C}, \mathcal{M} \rangle$ be a weak adhesive HLR category with epi-$\mathcal{M}$-factorization. The transformation Shift is

inductively defined as follows:

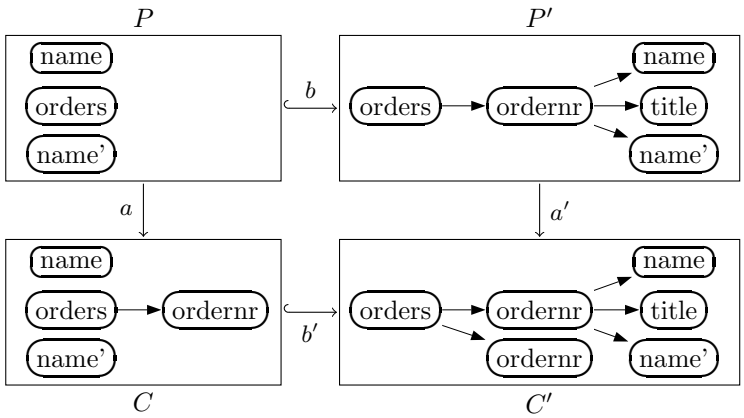


$\mathrm{Shift}(b, \mathrm{true}) = \mathrm{true}$.
$\mathrm{Shift}(b, \exists(a, c)) = \bigvee_{(a',b')\in\mathcal{F}} \exists(a', \mathrm{Shift}(b', c))$
with $\mathcal{F} = \{(a', b') \mid (a', b')$ jointly epimorphic, $b' \in \mathcal{M}$, and (1) commutes$\}$.

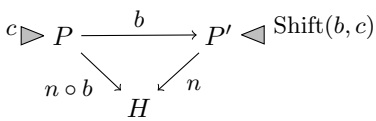
For Boolean formulas over conditions, Shift is extended in the usual way: For conditions c, c_i with $i \in I$ (for all index sets I), $\mathrm{Shift}(b, \neg c) = \neg\mathrm{Shift}(b, c)$ and $\mathrm{Shift}(b, \wedge_{i\in I} c_i) = \wedge_{i\in I}\mathrm{Shift}(b, c_i)$.

Remark 4. In the special case that $\mathcal{F}$ is empty, the result of the transformation is false. For previous versions of the Shift-construction see [16,17].

Example 1. Given the morphism $b\colon P \to P'$ below, the condition $\exists\, a$ is shifted into the 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'. The condition can be simplified to true because $\exists\, \mathrm{id}_{P'}$ is equivalent to true. The condition $\nexists\, a$ is shifted into the condition $\mathrm{Shift}(b, \nexists\, a) = \neg\mathrm{Shift}(b, \exists\, a) \equiv \neg\,\mathrm{true} \equiv \mathrm{false}$.



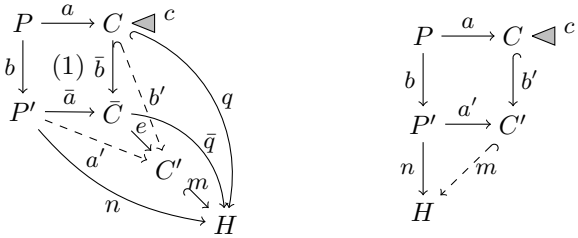
Lemma 1 (shift of conditions over morphisms). Let $\langle \mathcal{C}, \mathcal{M}\rangle$ be a weak adhesive HLR category with epi-$\mathcal{M}$-factorization. Then, for all conditions c over P and all morphisms $b\colon P \to P'$, $n\colon P' \to H$, $n \circ b \models c \Leftrightarrow n \models \mathrm{Shift}(b, c)$.



Proof. The statement is proved by structural induction.
Basis. For the condition true, the equivalence holds trivially.
Inductive step. For a condition of the form $\exists(a, c)$, we have to show

$$n \circ b \models \exists(a, c) \Leftrightarrow n \models \text{Shift}(b, \exists(a, c)).$$

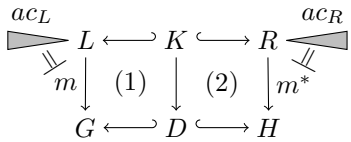
"$\Rightarrow$": Let $n \circ b \models \exists(a, c)$. By definition of satisfiability, there is some $q \in \mathcal{M}$ with $q \circ a = n \circ b$ and $q \models c$. Let $(\bar{a}, \bar{b})$ be the pushout in (1) in the left diagram below. By the universal property of pushouts, there is an induced morphism $\bar{q}\colon \bar{C} \to H$ such that $q = \bar{q} \circ \bar{b}$ and $n = \bar{q} \circ \bar{a}$. By epi-$\mathcal{M}$ factorization of $\bar{q}$, $\bar{q} = m \circ e$ with epimorphism e and monomorphism $m \in \mathcal{M}$. Define now $a' = e \circ \bar{a}$ and $b' = e \circ \bar{b}$. 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 $\langle \bar{a}, \bar{b} \rangle$ is jointly epimorphic and e is an epimorphism, (a', b') is jointly epimorphic. Thus, $(a', b') \in \mathcal{F}$. By the inductive hypothesis, $q = m \circ b' \models c \Leftrightarrow m \models \text{Shift}(b', c)$. Now $n \models \exists(a', \text{Shift}(b', c))$ and, by definition of Shift, $n \models \exists(b, \text{Shift}(a, c))$.



"$\Leftarrow$": Let $n \models \text{Shift}(b, \exists(a, c))$. By definition of Shift, there is some $(a', b') \in \mathcal{F}$ with $b' \in \mathcal{M}$ such that $n \models \exists(a', \text{Shift}(b', c))$. By definition of satisfiability, there is some $m \in \mathcal{M}$ such that $m \circ a' = n$ and $m \models \text{Shift}(b', c)$. By the inductive hypothesis, $m \models \text{Shift}(b', c) \Leftrightarrow m \circ b' \models c$. Now $m \circ b' \in \mathcal{M}$, $m \circ b' \circ a = n \circ b$ (see the right diagram above), and $m \circ b' \models c$, i.e., $n \circ b \models \exists(a, c)$. □

Rules are defined as in [5,17]. They are specified by a span of $\mathcal{M}$-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 [13,18].

Definition 6 (rules). A *rule* $\rho = \langle p, ac_L, ac_R \rangle$ consists of a plain rule $p = \langle L \hookleftarrow K \hookrightarrow R \rangle$ with $K \hookrightarrow L$ and $K \hookrightarrow R$ in $\mathcal{M}$ 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.



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 $m\colon L \to G$ 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 $G \Rightarrow_{\rho,m,m^*} H$.

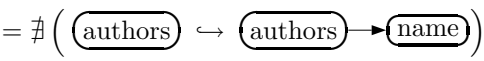
The concept of rules is completely symmetric.

Fact 4. For $\rho = \langle p, ac_L, ac_R \rangle$ with $p = \langle L \hookleftarrow K \hookrightarrow R \rangle$, $\rho^{-1} = \langle p^{-1}, ac_R, ac_L \rangle$ 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.

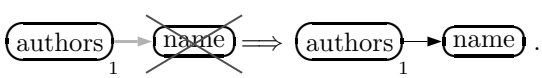
Notation. In the case of graphs, a rule $\langle L \hookleftarrow K \hookrightarrow R \rangle$ with discrete interface K is shortly depicted by $L \Rightarrow R$, where the nodes of K are indexed in the left- and the right-hand side of the rule. A negative application condition of the form $\nexists(L \hookrightarrow L')$ is integrated in the left-hand side of a rule by crossing the part $L'-L$ out. E.g. the rule

$p =$ 

with

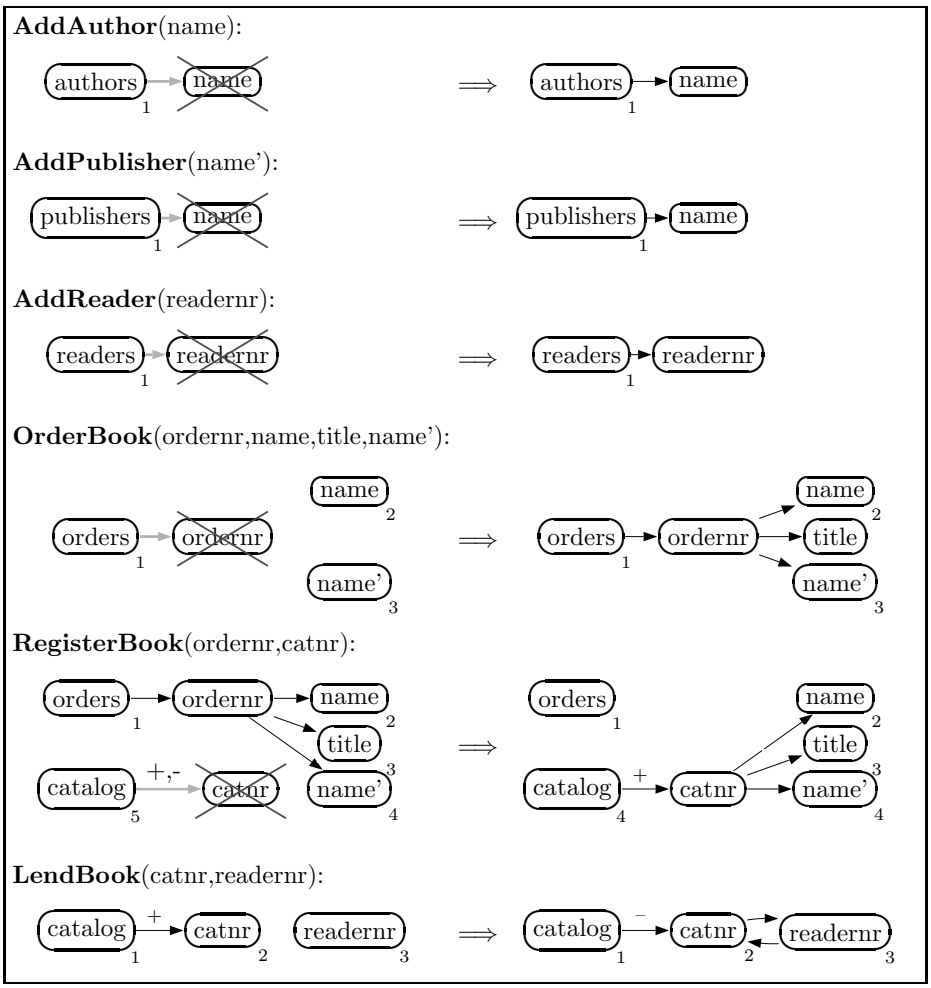
$ac_L = \nexists$ 

is depicted by



A conjunction $\bigwedge_i \nexists(L_i \hookrightarrow L'_i)$ of negative application conditions is represented by coloring the parts $L'_i - L_i$ in grey and crossing them out. A grey edge with labels $l_1, \ldots, l_n$ represents the conjunction of the negative application conditions "There does not exist an l_i-labelled edge" for $i = 1, \ldots, n$.

Example 2. In the figure below, rules with left application conditions are given, corresponding more or less to the operations of the small library system originally investigated in [23].

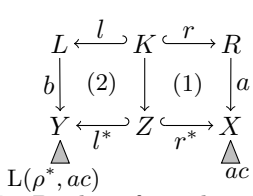


By Theorem 6 in [17], right application conditions of rules can be shifted into corresponding left application conditions and vice versa.

Lemma 2 (shift of conditions over rules). There are transformations L and R of application conditions such that, for every right application condition ac_R and every left application condition ac_L of a rule ρ and every direct derivation $G \Rightarrow_{\rho,m,m^*} H$, $m \models \mathrm{L}(\rho, ac_R) \Leftrightarrow m^* \models ac_R$ and $m \models ac_L \Leftrightarrow m^* \models \mathrm{R}(\rho, ac_L)$.

$$\begin{array}{ccccc} & \mathrm{L}(\rho, ac_R) & & & ac_R \\ & \blacktriangleright & & & \blacktriangleleft \\ L & \hookleftarrow & K & \hookrightarrow & R \\ m \downarrow & (1) & \downarrow & (2) & \downarrow m^* \\ G & \hookleftarrow & D & \hookrightarrow & H \end{array}$$

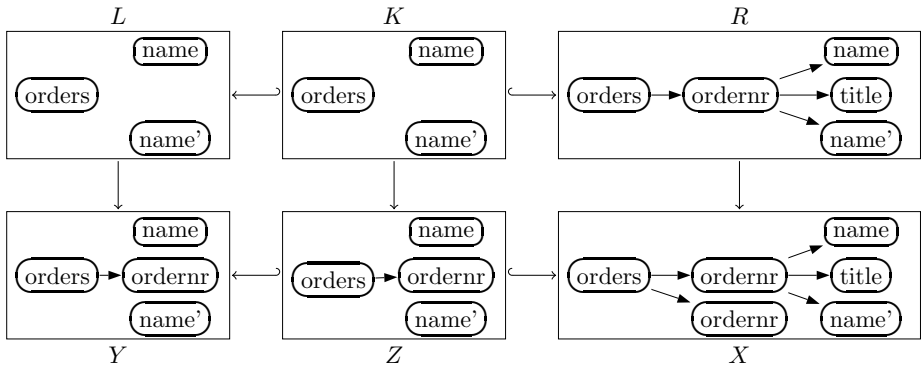
Construction. The transformation L is inductively defined as follows:



$\mathrm{L}(\rho, \mathrm{true}) = \mathrm{true}$
$\mathrm{L}(\rho, \exists(a, ac)) = \exists(b, \mathrm{L}(\rho^*, ac))$ 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).
$\mathrm{L}(\rho, \exists(a, ac)) = \mathrm{false}$, otherwise.

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

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



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 7 (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 $ac \equiv_\rho ac'$, 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.

Fact 5 (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 $\mathrm{R}(\rho, \mathrm{L}(\rho, ac))$ and ac are ρ-equivalent: $\mathrm{R}(\rho, \mathrm{L}(\rho, ac)) \equiv_\rho ac$.

Proof. By the Shift-Lemma 2, for every direct derivation $G \Rightarrow_{\rho,m,m^*} H$, $m^* \models \mathrm{R}(\rho, \mathrm{L}(\rho, ac)) \Leftrightarrow m \models \mathrm{L}(\rho, ac) \Leftrightarrow m^* \models ac$, i.e., the application conditions $\mathrm{R}(\rho, \mathrm{L}(\rho, ac))$ and ac are ρ-equivalent. □

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

Furthermore, 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 3 (Shift **and** L**).** For every direct derivation $L^* \Rightarrow_{\rho,k,k^*} R^*$ via a rule ρ and every application condition ac, $\mathrm{Shift}(k, \mathrm{L}(\rho, ac)) \equiv_{\rho^*} \mathrm{L}(\rho^*, \mathrm{Shift}(k^*, ac))$, 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 1 and 2, we have $l \models \mathrm{Shift}(k, \mathrm{L}(\rho, ac)) \Leftrightarrow m \models \mathrm{L}(\rho, ac) \Leftrightarrow m^* \models ac_R \Leftrightarrow l^* \models \mathrm{Shift}(k^*, ac) \Leftrightarrow l \models \mathrm{L}(\rho^*, \mathrm{Shift}(k^*, ac))$.

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

□

As a consequence of Shift-Lemma 2, every rule can be transformed into an equivalent one with true right application condition. A rule of the form $\langle p, ac_L, \mathrm{true} \rangle$ 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 Left from rules into rules with left application condition such that, for every rule ρ, ρ, and $\mathrm{Left}(\rho)$ are equivalent.

Proof. For a rule $\rho = \langle p, ac_L, ac_R \rangle$, the transformation Left is defined by $\mathrm{Left}(\rho) = \langle p, ac_L \wedge \mathrm{L}(\rho, ac_R) \rangle$. By Definition 6, Shift-Lemma 2, and the defi-

nition of Left,

$$\begin{aligned} G \Rightarrow_{\rho,m,m^*} H &\Leftrightarrow G \Rightarrow_{p,m,m^*} H \wedge m \models ac_L \wedge m^* \models ac_R \\ &\Leftrightarrow G \Rightarrow_{p,m,m^*} H \wedge m \models ac_L \wedge m \models \mathrm{L}(\rho, ac_R) \\ &\Leftrightarrow G \Rightarrow_{p,m,m^*} H \wedge m \models ac_L \wedge \mathrm{L}(\rho, ac_R) \\ &\Leftrightarrow G \Rightarrow_{\mathrm{Left}(\rho),m,m^*} H, \end{aligned}$$

i.e., the rules ρ and $\mathrm{Left}(\rho)$ are equivalent. □

4 Local Church-Rosser, Parallelism, and Concurrency

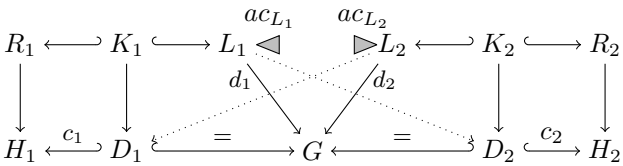
In this section, we present Local Church-Rosser, Parallelism, and Concurrency Theorems for rules with application conditions. The proofs of the statements are based on the corresponding statements for rules *without application conditions* [6] and Shift-Lemmas 1 and 2, saying that application conditions can be shifted over morphisms and rules.

First, we study parallel and sequential independence of direct derivations leading to the Local Church-Rosser and Parallelism Theorems for rules with application conditions. By Corollary 1, we may assume that the rules are rules with left application condition.

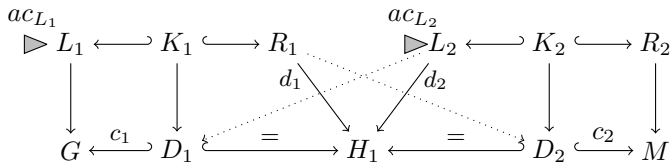
Assumption 2. In the following, let $\rho_1 = \langle p_1, ac_{L_1} \rangle$ and $\rho_2 = \langle p_2, ac_{L_2} \rangle$ be rules with $p_i = \langle L_i \hookleftarrow K_i \hookrightarrow R_i \rangle$ for $i = 1, 2$.

Roughly speaking, 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 [24].

Definition 8 (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 $L_1 D_2 G$ 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 $d_2 \colon R_1 \to D_2$ and $d_1 \colon L_2 \to D_1$ such that the triangles $R_1 D_2 H_1$ and $L_2 D_1 H_1$ commute, $m_1'^* = c_2 \circ d_2 \models \mathrm{R}(\rho_1, ac_{L_1})$ and $m_2 = c_1 \circ d_1 \models ac_{L_2}$.

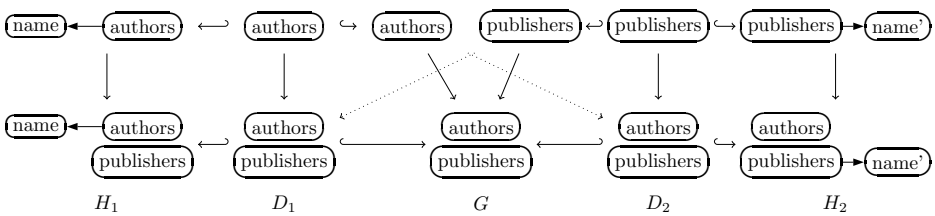


Two direct derivations that are not parallel (sequentially) independent, are called *parallel (sequentially) dependent.*

By definition, parallel and sequential independence are closely related.

Fact 6 (parallel and sequential independence are closely related). 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.

Example 4. The two direct derivations $H_1 \Leftarrow_{\rho_1} G \Rightarrow_{\rho_2} H_2$ via the rules $\rho_1 =$ **AddAuthor**(name) and $\rho_2 =$ **AddPublisher**(name′) are parallel independent.



In the proofs of the Local Church-Rosser, Parallelism and Concurrency Theorems, we proceed as follows: (1) We switch from derivations with ACs to the corresponding derivations without ACs, (2) use the results for derivations without ACs, and (3) lift the results without ACs to ACs.

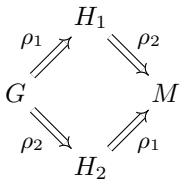
derivations with ACs	$\Longrightarrow$	result with ACs
$\downarrow$		$\uparrow$
derivations without ACs	$\Longrightarrow$	result without ACs

Fact 7 (Every derivation with ACs induces a derivation without ACs). For every direct derivation $G \Rightarrow_{\rho,m} H$ via the rule $\rho = \langle p, ac \rangle$, there is a direct derivation $G \Rightarrow_{p,m} H$ via the plain rule p, called the *underlying direct derivation without ACs.*

Fact 8 (independence with ACs implies independence without ACs). Parallel (sequential) independence of direct derivations implies parallel (sequential) independence of the underlying direct derivations without ACs.

Now we present a Local Church-Rosser Theorem for rules with application conditions. It generalizes the well-known Local Church-Rosser Theorems for rules without application conditions [6] and with negative application conditions [24].

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.

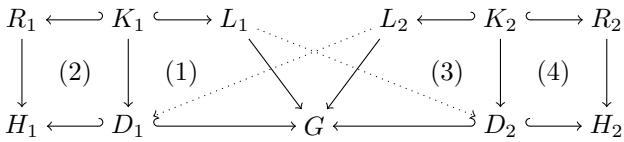


Proof. Let $H_1 \Leftarrow_{\rho_1,m_1} G \Rightarrow_{\rho_2,m_2} H_2$ be parallel independent. Then the underlying direct derivations without ACs are parallel independent. By the Local Church-Rosser Theorem without ACs [6], there are an object M and direct derivations $H_1 \Rightarrow_{p_2,m_2'} M \Leftarrow_{p_1,m_1'} H_2$ 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, $m_i, m_i' \models ac_{L_i}$ 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 $R_1 \to \bar{D}_2$ and $L_2 \to D_1$ be the morphisms in Figure 2. Then $R_1 \to \bar{D}_2 \to H_1 = m_1^*$ and $L_2 \to D_1 \to H_1 = m_2'$. By Shift-Lemma 2, $R_1 \to \bar{D}_2 \to M = m_1'^* \models \mathrm{R}(\rho_1, ac_{L_1})$ and $L_2 \to D_1 \to G = m_2 \models ac_{L_2}$. 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 [6], 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 $m_1, m_1' \models ac_{L_1}$, $m_2 \models ac_{L_2}$ (by Shift-Lemma 2, $m_1'^* \models \mathrm{R}(\rho_1, ac_{L_1})$ implies $m_1' \models ac_{L_1}$). 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 Figure 2 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 $L_2 \to D_1 \to H_1 = m_2' \models ac_{L_2}$. 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 a part of the proof of Local Church-Rosser Theorem for rules without ACs is given oriented at the one in [30].

Sketch of proof. 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 $L_1 D_2 G$

andL_2D_1G in the figure below commute.



The morphisms are used for the decomposition of the pushouts (i) into pushouts (i1),(i2) for $i = 1, \ldots, 4$ (Figure 2.1). The pushouts can be rearranged as in Figure 2.2 and 2.3. Furthermore, diagram (5) is constructed as pushout. 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. □

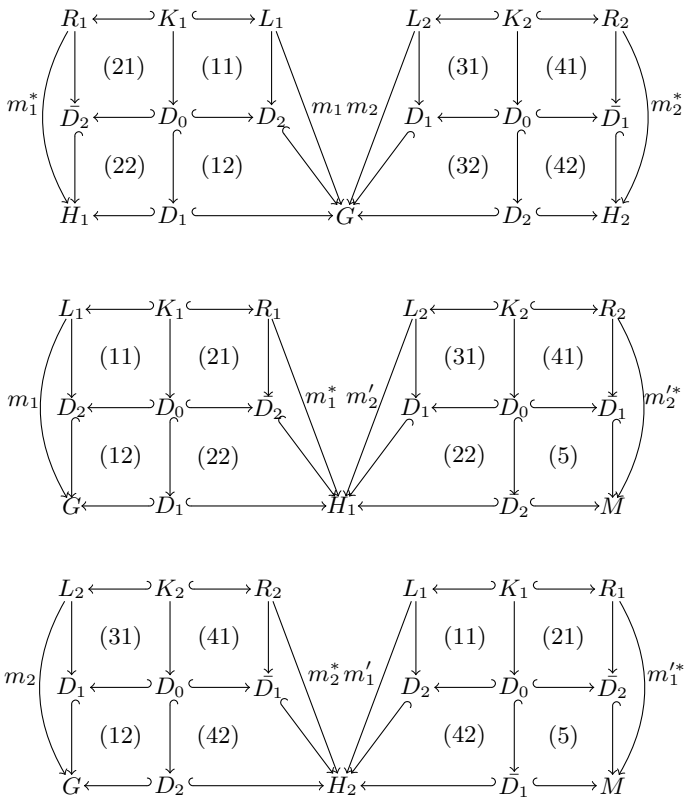
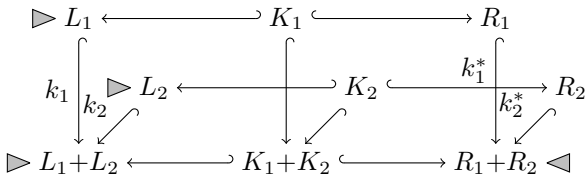


Fig. 2. Decomposition and composition

Next, we present the construction of a parallel rule of rules with application conditions. It generalizes the construction of a parallel rule of rules without application conditions [6] and makes use of the Shift of application conditions over morphisms and rules (see Shift-Lemmas 1 and 2). As in [6], 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 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 9 (parallel rule and derivation). The *parallel rule* of ρ_1 and ρ_2 is the rule $\rho_1 + \rho_2 = \langle p, ac'_L \rangle$ where $p = p_1 + p_2$ is the parallel rule of p_1 and p_2, and $ac_{L'} = ac_L \wedge \mathrm{L}(\rho_1 + \rho_2, ac_R)$, where

$$ac_L = \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{Shift}(k_2, ac_{L_2})$$
$$ac_R = \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, ac_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, ac_{L_2})).$$



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

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

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

and the application conditions

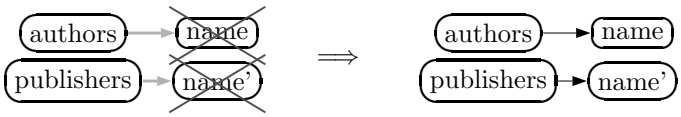
$$ac_L = \nexists\left(\text{publishers}\ \ \text{authors} \to \text{name}\right) \wedge \nexists\left(\text{authors}\ \ \text{publishers} \to \text{name'}\right)$$

$$ac_R = \nexists\left(\begin{matrix} & & \text{name} \\ \text{authors} & \to & \text{name} \\ \text{publishers} & \to & \text{name'} \end{matrix}\right) \qquad \wedge \nexists\left(\begin{matrix} \text{authors} & \to & \text{name} \\ \text{publishers} & \to & \text{name'} \\ & & \text{name'} \end{matrix}\right)$$

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 author nodes with label name", and "Afterwards, there do not exist two publisher nodes with label name′". 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 ρ yields the application condition ac_L.

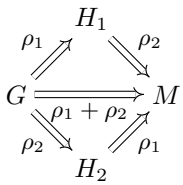
Thus, the parallel rule is equivalent to the rule with left application condition depicted below.

AddAuthorPublisher(name, name′):



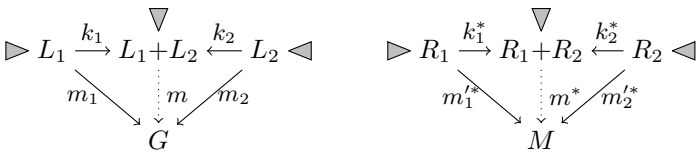
The connection between sequentially independent direct derivations and parallel direct derivations is expressed by the Parallelism Theorem. We present the Parallelism Theorem for rules with application conditions. It generalizes the well-known Parallelism Theorems for rules without application conditions [6] and with negative application conditions [16].

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. Let $G \Rightarrow_{\rho_1,m_1} H_1 \Rightarrow_{\rho_2,m_2'} M$ be sequentially independent. Then the underlying derivation without ACs is sequentially independent and, by the Parallelism Theorem without ACs [6], there is a parallel derivation $G \Rightarrow_{p_1+p_2,m} M$. By Shift-Lemmas 1 and 2, (*) $m \models ac_L$ and $m^* \models ac_R$ if and only if $m_i, m_i' \models ac_{L_i}$ for $i = 1, 2$. This may be seen as follows:

$$\begin{aligned}
m \models ac_L &\Leftrightarrow m \models \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{Shift}(k_2, ac_{L_2}) \\
&\Leftrightarrow m_1 \models ac_{L_1} \text{ and } m_2 \models ac_{L_2} \\
m^* \models ac_R &\Leftrightarrow m^* \models \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, ac_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, ac_{L_2})) \\
&\Leftrightarrow m_1'^* \models \mathrm{R}(\rho_1, ac_{L_1}) \text{ and } m_2'^* \models \mathrm{R}(\rho_2, ac_{L_2}) \\
&\Leftrightarrow m_1' \models ac_{L_1} \text{ and } m_2' \models ac_{L_2}
\end{aligned}$$



By assumption, $m_i, m_i' \models ac_{L_i}$ for $i = 1, 2$. By (∗), $m \models ac_L$ and $m^* \models ac_R$, i.e., $G \Rightarrow_{p_1+p_2,m} M$ satisfies ACs. Vice versa, let $G \Rightarrow_{\rho_1+\rho_2,m} M$ be a parallel

derivation. Then there is an underlying parallel derivation without ACs, and, by the Parallelism Theorem without ACs [6], 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, $m \models ac_L$ and $m^* \models ac_R$. By $(*)$, $m_i, m_i' \models ac_{L_i}$ 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.

Fact 9 (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 $\mathrm{L}(\rho, ac) \equiv_\rho \mathrm{L}(\rho_i^*, \mathrm{L}(\rho_j^*, ac))$ 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 [6], there are direct derivations $G \Rightarrow_{\rho_i^*,m} H_i \Rightarrow_{\rho_j^*,m'} M$ depicted in Figure 3. By the Shift-Lemma 2, $m \models \mathrm{L}(\rho, ac) \Leftrightarrow m^* \models ac \Leftrightarrow m' \models \mathrm{L}(\rho_j^*, ac) \Leftrightarrow m \models \mathrm{L}(\rho_i^*, \mathrm{L}(\rho_j^*, ac))$, i.e, the application conditions $\mathrm{L}(\rho, ac)$ and $\mathrm{L}(\rho_i^*, \mathrm{L}(\rho_j^*, ac))$ are ρ-equivalent. □

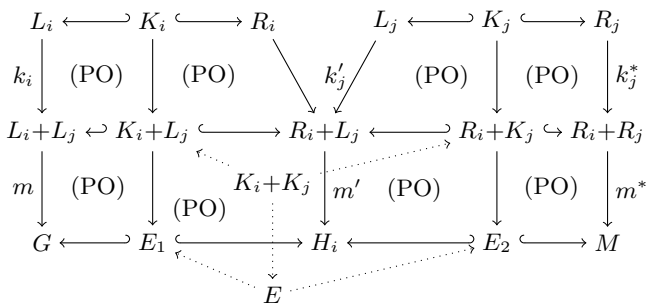


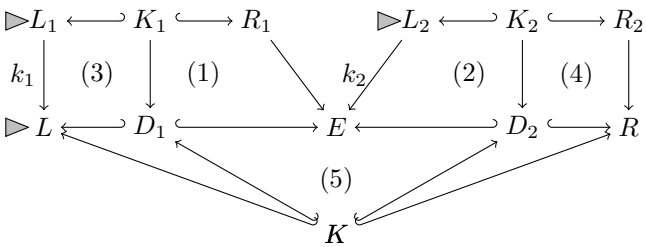
Fig. 3. Sequentialization of a parallel derivation

Finally, we present the construction of a concurrent rule for rules with application conditions. It generalizes the construction of concurrent rules for rules without application conditions [6] and makes use of shifting of application conditions over morphisms and rules (see Shift-Lemmas 1 and 2).

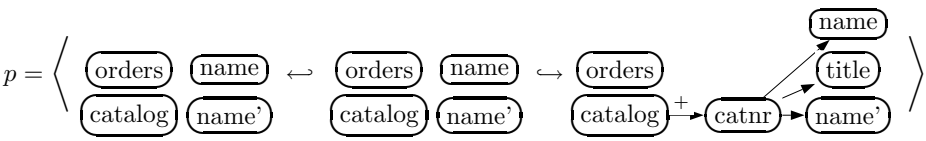
Definition 10 (E-concurrent rule). Let $\mathcal{E}'$ be a class of morphism pairs with the same codomain. Given two rules ρ_1 and ρ_2, an object E with morphisms $e_1 \colon R_1 \to E$ and $e_2 \colon L_2 \to E$ is an *E-dependency relation* for ρ_1 and ρ_2 if $(e_1, e_2) \in$

$\mathcal{E}'$ 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 $\rho_1 *_E \rho_2 = \langle p, ac_L \rangle$ 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

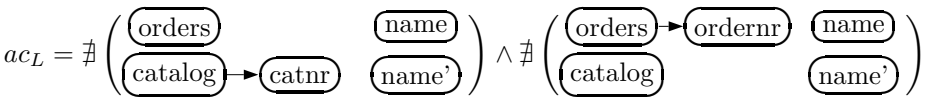
$$ac_L = \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{L}(\rho_1^*, \mathrm{Shift}(k_2, ac_{L_2}).$$



Example 6. The E-concurrent rule of ρ_1=**OrderBook**(ordernr, name, title, name′) and ρ_2 = **RegisterBook**(ordernr, 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

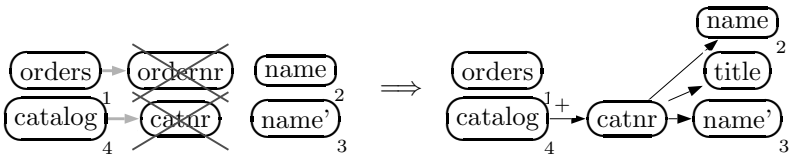


with the left application condition



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 may be depicted as follows.

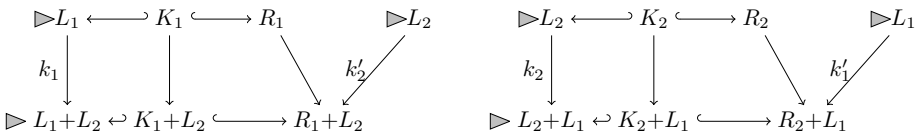
Order; **RegisterBook**(ordernr, catnr, name, title, name′):



The non-existence of a node with label catnr guarantees that, whenever the E-concurrent rule of ρ_1 and ρ_2 is applicable, then the rule ρ_1 with ordernr is applicable and, afterwards, the rule ρ_2 with catnr is applicable.

For rules without ACs, the parallel rule is a special case of the concurrent rule [6]. 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.

Fact 10. The parallel rule $\rho_1{+}\rho_2 = \langle p_1{+}p_2, ac_L, ac_R\rangle$ and the rule $\langle p_1{+}p_2, ac_{L_{12}} \wedge ac_{L_{21}}\rangle$ obtained from the $R_1{+}L_2$-concurrent rule $\langle p_1 + p_2, ac_{L_{12}}\rangle$ of ρ_1 and ρ_2 and the $R_2{+}L_1$-concurrent rule $\langle p_2 + p_1, ac_{L_{21}}\rangle$ of ρ_2 and ρ_1 are equivalent.



Proof. For every parallel derivation $G \Rightarrow_{\rho_1+\rho_2,m,m^*} M$ (see Figure 3) and $i, j \in \{1, 2\}$ with $i \neq j$, we have

$$
\begin{array}{lll}
(***)\ m^* \models \mathrm{Shift}(k_j^*, \mathrm{R}(\rho_j, ac_{L_j})) & & \\
\quad\Leftrightarrow \quad m^* \models \mathrm{R}(\rho_j^*, \mathrm{Shift}(k_j, ac_{L_j})) & & \text{(Lemma 3)} \\
\quad\Leftrightarrow \quad m \models \mathrm{L}(\rho_j^*, \mathrm{R}(\rho_j^*, \mathrm{Shift}(k_j, ac_{L_j}))) & & \text{(Shift-Lemma 2)} \\
\quad\Leftrightarrow \quad m \models \mathrm{Shift}(k_j, ac_{L_j})) & & \text{(Fact 5)}
\end{array}
$$

By the definitions and statement (***),

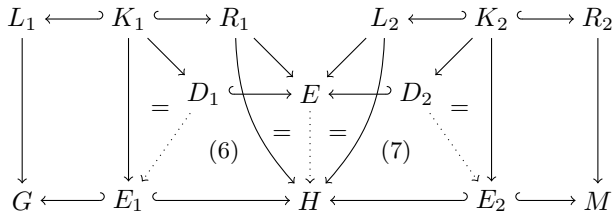
$$
\begin{array}{lll}
\quad m \models ac_L \text{ and } m^* \models ac_R & & \\
\Leftrightarrow m \models \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{Shift}(k_2, ac_{L_2}) \text{ and} & & \\
\quad m^* \models \mathrm{Shift}(k_1^*, \mathrm{R}(\rho_1, ac_{L_1})) \wedge \mathrm{Shift}(k_2^*, \mathrm{R}(\rho_2, ac_{L_2})) & & \text{(Definition 9)} \\
\Leftrightarrow m \models \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{L}(\rho_1^*, \mathrm{Shift}(k_2', ac_{L_2})) \text{ and} & & \\
\quad m \models \mathrm{Shift}(k_2, ac_{L_2}) \wedge \mathrm{L}(\rho_2^*, \mathrm{Shift}(k_1', ac_{L_1})) & & (***) \\
\Leftrightarrow m \models ac_{L_{12}} \wedge ac_{L_{21}} & & \text{(Definition 10)}
\end{array}
$$

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.

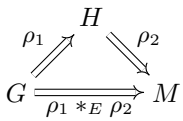
Definition 11 (E-concurrent and E-related derivation). A direct derivation via an E-concurrent rule is called *E-concurrent* direct derivation or E-concurrent derivation, for short. A derivation $G \Rightarrow_{\rho_1} H \Rightarrow_{\rho_2} M$ is *E-related* if there are morphisms $E \to H$, $D_1 \to E_1$, 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. It generalizes the well-known Concurrency Theorems for rules without application conditions [6] and with negative application conditions [16].

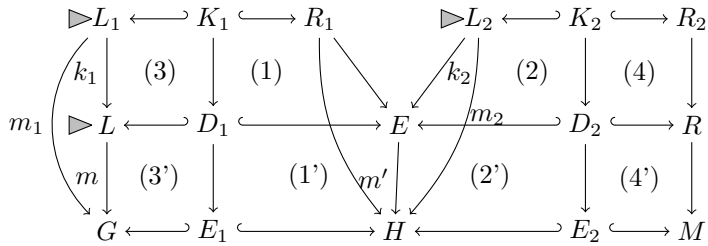
Theorem 3 (Concurrency). Let E be a dependency relation for ρ_1 and ρ_2. For every E-related derivation $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$, there is an E-concurrent derivation $G \Rightarrow_{\rho_1 *_E \rho_2,m} M$. Vice versa, for every E-concurrent derivation $G \Rightarrow_{\rho_1 *_E \rho_2,m} M$, there is an E-related derivation $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$.



Proof. Let $G \Rightarrow_{\rho_1,m_1} H \Rightarrow_{\rho_2,m_2} M$ be E-related. Then the underlying derivation without ACs is E-related and, by the Concurrency Theorem without ACs [6], there is an E-concurrent derivation $G \Rightarrow_{p_1 * p_2,m} M$. By Shift-Lemmas 1 and 2, (**) $m_1 \models ac_{L_1}$ and $m_2 \models ac_{L_2}$ iff $m \models ac_L$. This may be seen as follows:

$$\begin{aligned}
& m_1 \models ac_{L_1} \text{ and } m_2 \models ac_{L_2} \\
\Leftrightarrow\ & m \models \mathrm{Shift}(k_1, ac_{L_1}) \text{ and } m' \models \mathrm{Shift}(k_2, ac_{L_2}) \\
\Leftrightarrow\ & m \models \mathrm{Shift}(k_1, ac_{L_1}) \text{ and } m \models \mathrm{L}(p_1^*, \mathrm{Shift}(k_2, ac_{L_2})) \\
\Leftrightarrow\ & m \models \mathrm{Shift}(k_1, ac_{L_1}) \wedge \mathrm{L}(p_1^*, \mathrm{Shift}(k_2, ac_{L_2})) = ac_L.
\end{aligned}$$

By assumption, $m_i \models ac_{L_i}$ for $i = 1, 2$. By (**), $m \models ac_L$, i.e. the E-concurrent derivation satisfies ACs.



Vice versa, let $G \Rightarrow_{\rho,m} M$ be an E-concurrent derivation, then the underlying direct derivation without ACs is E-concurrent, and, by the Concurrency Theorem without ACs [6], there is an E-related derivation $G \Rightarrow_{p_1,m_1} H \Rightarrow_{p_2,m_2} M$. By assumption, $m \models ac_L$. By (**), $m_1 \models ac_{L_1}$ and $m_2 \models ac_{L_2}$, 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 [16], for rules with nested application conditions. The proofs are based on the corresponding theorems for rules without application conditions [6] and two Shift-Lemmas [17], 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 epi-$\mathcal{M}$-factorization and binary coproducts.

statement	requirements
Local Church-Rosser	Shift 1 & 2
Parallelism	Shift 1 & 2, binary coproducts
Concurrency	Shift 1 & 2
Shift 1	epi-$\mathcal{M}$-factorization
Shift 2	–

Further topics might be the following:

- **Amalgamation Theorem for rules with ACs.** It would be important to generalize the AmalgamationTheorem [25,18] 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 [26,13,27,28,6,29] 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 [30].

References

1. Rozenberg, G. ed.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations. World Scientific (1997)
2. Ehrig, H. Engels, G. Kreowski, H.J. Rozenberg, G. eds.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 2: Applications, Languages and Tools. World Scientific (1999)
3. Ehrig, H. Kreowski, H.J. Montanari, U. Rozenberg, G. eds.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 3: Concurrency, Parallelism, and Distribution. World Scientific (1999)
4. Ehrig, H. Habel, A. Kreowski, H.J. Parisi-Presicce, F.: Parallelism and concurrency in high level replacement systems. Mathematical Structures in Computer Science **1** (1991) 361–404
5. Ehrig, H. Ehrig, K. Habel, A. Pennemann, K.H.: Theory of constraints and application conditions: From graphs to high-level structures. Fundamenta Informaticae **74(1)** (2006) 135–166
6. Ehrig, H. Ehrig, K. Prange, U. Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs of Theoretical Computer Science. Springer, Berlin (2006)

7. Ehrig, H. Habel, A.: Graph grammars with application conditions. In Rozenberg, G. Salomaa, A. eds.: The Book of L. Springer, Berlin (1986) 87–100
8. Habel, A. Heckel, R. Taentzer, G.: Graph grammars with negative application conditions. Fundamenta Informaticae **26** (1996) 287–313
9. Habel, A. Pennemann, K.H.: Nested constraints and application conditions for high-level structures. In: Formal Methods in Software and System Modeling. Volume 3393 of LNCS. Springer (2005) 293–308
10. Ehrig, H. Kreowski, H.J.: Parallelism of manipulations in multidimensional information structures. In: Mathematical Foundations of Computer Science. Volume 45 of LNCS. Springer (1976) 284–293
11. Kreowski, H.J.: Manipulationen von Graphmanipulationen. PhD thesis, Technical University of Berlin (1977)
12. Kreowski, H.J.: Transformations of derivation sequences in graph grammars. In: Fundamentals of Computation Theory. Volume 56 of LNCS. Springer (1977) 275–286
13. Ehrig, H.: Introduction to the algebraic theory of graph grammars. In: Graph-Grammars and Their Application to Computer Science and Biology. Volume 73 of LNCS. Springer (1979) 1–69
14. Ehrig, H. Rosen, B.K.: Parallelism and concurrency of graph manipulations. Theoretical Computer Science **11** (1980) 247–275
15. Habel, A.: Concurrency in Graph-Grammatiken. Technical Report 80-11, Technical University of Berlin (1980)
16. Lambers, L. Ehrig, H. Prange, U. Orejas, F.: Parallelism and concurrency in adhesive high-level replacement systems with negative application conditions. In: Workshop on Applied and Computational Category Theory (ACCAT 2007). Volume 2003 of ENTCS. Elsevier (2008) 43–66
17. Habel, A. Pennemann, K.H.: Correctness of high-level transformation systems relative to nested conditions. Mathematical Structures in Computer Science **19** (2009) 245–296
18. Corradini, A. Montanari, U. Rossi, F. Ehrig, H. Heckel, R. Löwe, M.: Algebraic approaches to graph transformation. Part I: Basic concepts and double pushout approach. In: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1. World Scientific (1997) 163–245
19. Lack, S. Sobociński, P.: Adhesive categories. In: Foundations of Software Science and Computation Structures (FOSSACS'04). Volume 2987 of LNCS. Springer (2004) 273–288
20. Heckel, R. Wagner, A.: Ensuring consistency of conditional graph grammars — a constructive approach. In: SEGRAGRA '95. Volume 2 of ENTCS. (1995) 95–104
21. Koch, M. Mancini, L.V. Parisi-Presicce, F.: Graph-based specification of access control policies. Journal of Computer and System Sciences **71** (2005) 1–33
22. Habel, A. Pennemann, K.H.: Satisfiability of high-level conditions. In: Graph Transformations (ICGT 2006). Volume 4178 of LNCS. Springer (2006) 430–444
23. Ehrig, H. Kreowski, H.J.: Applications of graph grammar theory to consistency, synchronization and scheduling in data base systems. Information Systems **5** (1980) 225–238
24. Lambers, L. Ehrig, H. Orejas, F.: Conflict detection for graph transformation with negative application conditions. In: Graph Transformations (ICGT 2006). Volume 4178 of LNCS. Springer (2006) 61–76
25. Boehm, P. Fonio, H.R. Habel, A.: Amalgamation of graph transformations: A synchronization mechanism. Journal of Computer and System Sciences **34** (1987) 377–408

26. Ehrig, H.: Embedding theorems in the algebraic theory of graph grammars. In: Fundamentals of Computation Theory. Volume 56 of LNCS. Springer (1977) 245–255
27. Plump, D.: Hypergraph rewriting: Critical pairs and undecidability of confluence. In: Term Graph Rewriting: Theory and Practice. John Wiley, New York (1993) 201–213
28. Plump, D.: 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. Volume 3838 of LNCS. Springer (2005) 280–308
29. Lambers, L. Ehrig, H. Prange, U. Orejas, F.: Embedding and confluence of graph transformations with negative application conditions. In: Graph Transformations (ICGT 2008). Volume 5214 of LNCS. Springer (2008) 162–177
30. Habel, A. Müller, J. Plump, D.: Double-pushout graph transformation revisited. Mathematical Structures in Computer Science **11** (2001) 637–688

Prof. Dr. Hartmut Ehrig

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin
D-10587 Berlin (Germany)
ehrig@cs.tu-berlin.de
http://tfs.cs.tu-berlin.de/˜ehrig

Hartmut Ehrig knows Hans-Jörg Kreowski since 1970 when he was one of the most engaged students in Hartmut's seminar on *Kategorien und Automaten* at the Mathematical Department of TU Berlin. This seminar was a great success, leading to a textbook with the same title, published 1971 by Walter de Gruyter. In 1974 followed a joint international book *Universal Theory of Automata*, published by Teubner, which was mainly based on Hans-Jörg's Diploma thesis. Meanwhile, Hartmut had become assistant professor at the new Department of Computer Science at TU Berlin, and hired Hans-Jörg as an assistant. The main focus of their joint work switched from Categorical Automata Theory to Graph Transformation, based on the DPO-approach, and Algebraic Specification, following the initial algebra approach of the ADJ-group at IBM Yorktown Heights. A very important contribution in the first area was Hans-Jörg's doctoral thesis *Manipulationen von Graphmanipulationen*, on the concurrent semantics of graph transformation systems. In the area of algebraic specifications, their joint research focussed on parametrized specifications and parameter passing, which led to the well-known ACT-approach and the algebraic specification language Act One. With respect to both areas, this period was most successful for them, with interesting contributions to important conferences and publications in the Springer LNCS series and several well-known journals. Meanwhile, Hans-Jörg finished his habilitation thesis in Berlin. In 1982, he accepted a call for a professorship in Bremen, where he built up a strong research group in the areas of Algebraic Specification and Graph Transformation. Since that time the research groups in Berlin and Bremen have been working together with great success, especially in the European

Research Projects CompuGraph, Compass, GetGraTS, AppliGraph, and SeGraVis.

Prof. Dr. Annegret Habel

Carl v. Ossietzky Universität Oldenburg
Fachbereich Informatik
D-26111 Oldenburg (Germany)
Annegret.Habel@informatik.uni-oldenburg.de
http://theoretica.informatik.uni-oldenburg.de/~habel

Annegret Habel was the first doctoral student of Hans-Jörg Kreowski. She joined his team as a research associate in 1986. Having received her doctoral degree in 1989, she continued to work in his team as an assistant professor until 1995, when she was offered a professorship in Hildesheim, and later moved to Oldenburg.

Leen Lambers

Institut für Softwaretechnik und Theoretische Informatik
Technische Universität Berlin
D-10587 Berlin (Germany)
leen@cs.tu-berlin.de
http://tfs.cs.tu-berlin.de/~leen