Appl Categor Struct (2008) 16:389–419
DOI 10.1007/s10485-008-9127-6

Subobject Transformation Systems

Andrea Corradini · Frank Hermann · Paweł Sobociński

Received: 13 October 2006 / Accepted: 22 January 2008 / Published online: 20 February 2008

Abstract Subobject transformation systems (STS) are proposed as a novel formal framework for the analysis of derivations of transformation systems based on the algebraic, double-pushout (DPO) approach. They can be considered as a simplified variant of DPO rewriting, acting in the distributive lattice of subobjects of a given object of an adhesive category. This setting allows a direct analysis of all possible notions of dependency between any two productions without requiring an explicit match. In particular, several equivalent characterizations of independence of productions are proposed, as well as a local Church–Rosser theorem in the setting of STS. Finally, we show how any derivation tree in an ordinary DPO grammar leads to an STS via a suitable construction and show that relational reasoning in the resulting STS is sound and complete with respect to the independence in the original derivation tree.

Keywords Graph transformation systems · Adhesive categories · Occurrence grammars

Mathematics Subject Classifications (2000) 18B35 · 68Q10 · 68Q42

Research partially supported by EU IST-2004-16004 SEnSOria and MIUR PRIN 2005015824 ART. The third author acknowledges the support of EPSRC grant EP/D066565/1.

A. Corradini (✉)
Dipartimento di Informatica, Università di Pisa, Pisa, Italy
e-mail: andrea@di.unipi.it

F. Hermann
Department of Electrical Engineering and Computer Science,
Technical University of Berlin, Berlin, Germany

P. Sobociński
ECS, University of Southampton, Southampton, UK

1 Introduction

Graph transformation systems (GTSs) [21] are a powerful specification formalism for concurrent and distributed systems, generalising another classical model of concurrency, namely Place/Transition Petri nets [19]. The concurrent behaviour of GTSs has been thoroughly studied and a consolidated theory of concurrency is now available. In particular, several constructions associated with the concurrent semantics of nets, such as processes and unfoldings, have been extended to GTSs. Intuitively, a deterministic process of a Petri net is a partial-order representation of a concurrent computation of the net, while the unfolding represents a branching and acyclic structure of all its possible concurrent computations. Interestingly, the processes and the unfolding of a net are themselves Petri nets of a particular kind, called *occurrence nets*. The generalization of these notions to GTSs (see, e.g., [1, 3, 7, 20]) follows a similar pattern, introducing *occurrence grammars* as a partial-order representation of GTSs computations.

Recently, several constructions and results of the classical theory of the algebraic double-pushout (DPO) approach to graph transformation have been extended to rewriting in arbitrary *adhesive categories* [10, 16]. In a joint effort with other researchers, the authors are working on the generalization of the concurrent semantics of nets and GTS to this more abstract setting: first results concerning deterministic processes appeared in [2].

It turns out that a key ingredient in the definition of processes and unfoldings for a given transformation system, is the analysis of the relationships that arise between the occurrences of rules in the possible computations of the system. Such relations include the *parallel* and *sequential independence*, deeply studied in the classical theory of the algebraic approaches to graph rewriting [8, 11]; the standard *causality* and *conflict* relations; the *asymmetric conflict*, studied for formalisms able to model read-only operations; and the less known *co-causality*, *disabling* and *co-disabling*, introduced in [2]. Actually, such relations are meaningful for restricted classes of transformation systems only, including the *occurrence systems* (nets and grammars) mentioned above.

Even if the possible ways in which the rules of an occurrence system can be related are quite well understood, to our knowledge a systematic study of this topic is still missing: This is the main goal of the present paper. To this aim, we introduce the notion of a *Subobject Transformation System (STS)*, which can be understood, intuitively, as a DPO rewriting system in the category of subobjects of a given object of an adhesive category (corresponding to the *type graph* in the typed approaches to DPO rewriting [7]). In this framework, the usual pushout and pullback constructions are replaced by union and intersection of subobjects. Thus, in general, one can work with a set-theoretical syntax rather than with a categorical one.

Known examples of STSs in the area of DPO graph transformation systems are the graph processes as defined in [3, 7], and the unfolding presented in [1], but STSs are more general, because no acyclicity constraint is enforced in their definition. Interestingly, it turns out that STSs in category **Set** with rules having empty interfaces correspond precisely to *Elementary Net Systems* [22], a class of Petri nets widely studied in the literature. However, the present paper focuses on the core of the new theory only, while the precise relationships among STSs and the mentioned

computational models already proposed in the literature is left as a topic of future investigation.

After introducing STSs in Section 2, we exploit them in Section 3 in order to identify the possible basic relations among rules, and we use these to define other derived relations which are shown to coincide with those introduced in the literature for STSs arising as processes of DPO rewriting systems. Here we shall rely on a useful auxiliary graphical "Venn-diagram" like notation for reasoning about dependency relations. The zones of the diagrams are not in general subobjects and in order to reason about them formally we introduce the notion of *region* which is, roughly, a complement of a subobject. The basic theory of regions allows us to show that reasoning with the aid of the Venn-diagrams is sound. Next, in Section 4, we discuss the conditions under which two productions of an STS have to be considered as independent, and we characterize this relation in several equivalent ways; a local Church–Rosser theorem closes the section. In Section 5 we present a colimit construction that builds an STS from a given derivation tree of a DPO system, generalizing the construction of the process of a linear derivation proposed in [3, 7]. Finally in Section 6 we show that the analysis of the relationships among rule occurrences in the derivation tree can be reduced faithfully to the analysis of such relationships in the generated STS. In the concluding section we list some topics of future research.

2 Subobject Transformation Systems

As mentioned above, Subobject Transformation Systems can be considered, conceptually, as transformation systems based on the DPO approach in the category of subobjects $\mathbf{Sub}(T)$ of some object T of a category $\mathbf{C}$. We shall assume that $\mathbf{C}$ is an adhesive category: this ensures that $\mathbf{Sub}(T)$ is a distributive lattice. The choice is justified by the intended use of STSs as a formal framework for analysing derivations of DPO systems (as detailed in Sections 5 and 6), which themselves are defined over adhesive categories in [16].

Definition 1 (Adhesive categories) A category is called *adhesive* if

- It has pushouts along monos;
- It has pullbacks;
- Pushouts along monos are *Van Kampen* (VK) squares.

Referring to Fig. 1, a VK square is a pushout (1) which satisfies the following property: if we draw a commutative cube (2) which has (1) as its bottom face and whose back faces are pullbacks then the front faces of the cube are pullbacks if and only if its top face is a pushout.

Recall that given a category $\mathbf{C}$ and an object $T \in \mathbf{C}$, the *category of subobjects* $\mathbf{Sub}(T)$ is defined to be the full subcategory of the slice category $\mathbf{C}/T$ with objects the monomorphisms into T. We denote an object $a : A \rightarrowtail T$ of $\mathbf{Sub}(T)$ simply as A, leaving the monomorphism implicit. Notice that $\mathbf{Sub}(T)$ is a preorder; there is at

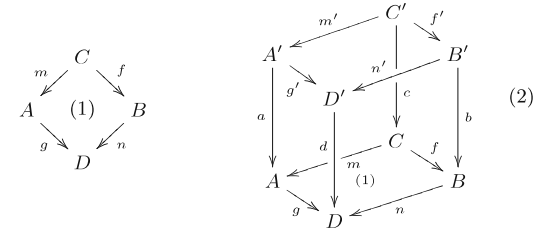


Fig. 1 A pushout square (*1*) and a commutative cube (*2*)

most one arrow between any two objects A and B, denoted $A \subseteq B$. In particular, this implies that all diagrams (in $\mathbf{Sub}(T)$) are commutative.

If $\mathbf{C}$ has pullbacks then $\mathbf{Sub}(T)$ has binary products (also called *intersections*): the product of two subobjects is given by the diagonal of the pullback square of the two morphisms in $\mathbf{C}$. If $\mathbf{C}$ is adhesive, then $\mathbf{Sub}(T)$ also has binary coproducts (*unions*): the coproduct of two subobjects is given by the mediating morphism from the pushout in $\mathbf{C}$ of the projections of their intersection [16]. Furthermore, in this case $\mathbf{Sub}(T)$ is a *distributive* lattice, i.e. products distribute over coproducts, and vice-versa. We shall denote the product and coproduct in $\mathbf{Sub}(T)$ by $\cap$ and $\cup$, respectively.

Throughout the paper we shall often use Venn diagrams to depict subobjects, representing union and intersection in the usual way. For example, in Fig. 2a subobject A is represented by the area delimited by the left rounded square (made of "zones" a and c), B is represented by b and c, $A \cap B$ by c, and $A \cup B$ by a, c and b. It is worth stressing that this graphical representation of subobjects is sound because $\mathbf{Sub}(T)$ is distributive. In fact, for example, in Fig. 2b subobjects $A \cap (B \cup C)$ and $(A \cap B) \cup (A \cap C)$, which are equal by distributivity, are both represented by the area made of d, e, and f; similarly, $A \cup (B \cap C)$ $(= (A \cup B) \cap (A \cup C))$ is represented by a, d, e, f and g.

Notice that category $\mathbf{Sub}(T)$ in general is not adhesive, even if $\mathbf{C}$ is. In fact, let $A \subseteq B$ be an arrow in $\mathbf{Sub}(T)$ which is not an isomorphism; then the pushout object of the span $B \supseteq A \subseteq B$ is easily shown to be B itself, but the resulting square is not a pullback, contradicting the fact that pushouts along monos in an adhesive category are pullbacks [16].

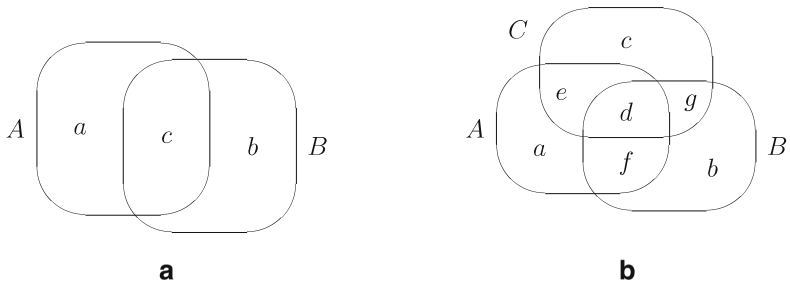


Fig. 2 Venn diagrams representing two (**a**) and three (**b**) subobjects, their unions and intersections

Definition 2 (Subobject transformation systems) A *Subobject Transformation System* (STS) over an adhesive category $\mathbf{C}$ is a triple $\mathcal{S} = \langle T, P, \pi \rangle$, where $T \in \mathbf{C}$ is a *type object*, P is a set of *production names*, $\pi : P \to \mathbf{Sub}(T)^{\cdot \leftarrow \cdot \rightarrow \cdot}$ maps a production name p to a *production* $L_p \supseteq K_p \subseteq R_p$, i.e., a span in $\mathbf{Sub}(T)$, where L_p, K_p and R_p are called, respectively, the *left-hand side*, the *interface* and the *right-hand side* of p.

A production will often be denoted simply as a triple $\langle L_p, K_p, R_p \rangle$, implicitly assuming that $K_p \subseteq L_p \cap R_p$.

Definition 3 (Direct derivations) Let $\mathcal{S} = \langle T, P, \pi \rangle$ be a Subobject Transformation System, $\pi(q) = \langle L, K, R \rangle$ be a production, and let G be an object of $\mathbf{Sub}(T)$. Then there is a *direct derivation from G to G' using q*, written $G \Rightarrow^q G'$, if $G' \in \mathbf{Sub}(T)$ and there exists an object $D \in \mathbf{Sub}(T)$ such that:

(a) $L \cup D \cong G$; (c) $D \cup R \cong G'$;
(b) $L \cap D \cong K$; (d) $D \cap R \cong K$.

If such an object D exists, we shall refer to it as the *context of G w.r.t. q* or as the *context of the direct derivation $G \Rightarrow^q G'$*. As we shall see later in Proposition 7, given a production q and a subobject G, a context of G w.r.t. q is unique up to isomorphism, if it exists.

Given subobjects L and R, considered as left- and right-hand sides of a production, there is a canonical choice for the interface K, namely $K = L \cap R$.

Definition 4 (Pure STS) We shall say that an STS is *pure* when $K = L \cap R$ in all of its productions.

Intuitively, in a pure STS no production deletes and produces again the same part of a subobject: this terminology is adapted from the theory of Elementary Net Systems, where a system which does not contain transitions with a self-loop is called "pure".

The next technical lemma will be used several times along the paper: It provides a simple set-theoretical syntax for expressing the fact that a commutative square in $\mathbf{Sub}(T)$ is a pushout in $\mathbf{C}$.

Lemma 5 *Assume that $\mathbf{C}$ is an adhesive category, that $T \in \mathbf{C}$, and that (†) is a square in $\mathbf{Sub}(T)$ (equivalently, it is a commutative square of monos in $\mathbf{C}$ and there is a monomorphism $D \to T$). Then the following are equivalent:*

(1) (†) *is a pushout in* $\mathbf{C}$
(2) $B \cap C \cong A$ *and* $D \cong B \cup C$
(3) $B \cap C \subseteq A$ *and* $D \subseteq B \cup C$.

$$\begin{array}{ccc} A & \rightarrowtail & B \\ \downarrow & (\dagger) & \downarrow \\ C & \rightarrowtail & D \end{array}$$

Proof

(1 ⇒ 2) The square (†) is a pushout in $\mathbf{C}$ along a mono, therefore a pullback [16]. Since $B \cap C$ is also a pullback of $C \to D$ and $B \to D$ (given that there is a morphism $D \to T$), we have $B \cap C \cong A$ and thus $D \cong B \cup C$.

(2 ⇔ 3) The isomorphisms imply the inclusions. For the other implication, it is sufficient to observe that, given (†), $A \subseteq B \cap C$ and $B \cup C \subseteq D$ hold by the universal properties of $\cap$ and $\cup$, respectively.
(1 ⇐ 2) Diagram (†) is a pushout in $\mathbf{C}$ by the characterisation of unions of subobjects in adhesive categories. □

It is instructive to consider the relationship between a direct derivation in an STS and the usual notion of a DPO direct derivation in an adhesive category. It is possible to make this comparison, since one can consider a production $L_p \supseteq K_p \subseteq R_p$ as the underlying span of monomorphisms in $\mathbf{C}$.

We shall say that there is a *contact situation* for a production $\langle L, K, R \rangle$ at a subobject $G \supseteq L \in \mathbf{Sub}(T)$ if $G \cap R \not\subseteq L$. Intuitively this means that part of the subobject G is created but not deleted by the production: if we were allowed to apply the production at this match via a DPO direct derivation, the resulting object would contain the common part twice and consequently the resulting morphism to T would not be a monomorphism; i.e., the result would not be a subobject of T. The next result clarifies the relationship between the definition of direct derivation in $\mathbf{Sub}(T)$ and the standard definition used in the DPO approach [8]: essentially, the two derivations coincide if there is no contact.

Proposition 6 (STS derivations are contact-free double pushouts) *Let $\mathcal{S} = \langle T, P, \pi \rangle$ be an STS over an adhesive category $\mathbf{C}$, $\pi(q) = \langle L, K, R \rangle$ be a production, and G be an object of $\mathbf{Sub}(T)$. Then $G \Rightarrow^q G'$ iff $L \subseteq G$, $G \cap R \subseteq L$, and there is an object D in $\mathbf{C}$ such that the following diagram consists of two pushouts in $\mathbf{C}$.*

$$\begin{array}{ccccc} L & \overset{l}{\leftarrowtail} & K & \overset{r}{\rightarrowtail} & R \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle k} & (2) & \downarrow{\scriptstyle n} \\ G & \underset{f}{\leftarrowtail} & D & \underset{g}{\rightarrowtail} & G' \end{array}$$

Proof

(⇒) Suppose that $G \Rightarrow^q G'$. Then by Definition 3 there exists an object $D \in \mathbf{Sub}(T)$ such that (a) $L \cup D \cong G$ and (b) $L \cap D \cong K$, so clearly $L \subseteq G$ and by the conclusion of Lemma 5, the left square (1) is a pushout in $\mathbf{C}$. Furthermore, (c) $D \cup R \cong G'$ and (d) $D \cap R \cong K$, and thus (2) is a pushout in $\mathbf{C}$ as well. The fact that $G \cap R \subseteq L$ can be shown as follows, using (a) and (d):

$$G \cap R \overset{(a)}{\cong} (L \cup D) \cap R \overset{(*)}{\cong} (L \cap R) \cup (D \cap R) \overset{(d)}{\cong} (L \cap R) \cup K \subseteq L$$

where $(*)$ holds by distributivity of $\mathbf{Sub}(T)$.
(⇐) Suppose that the squares (1) and (2) are pushouts in $\mathbf{C}$, $L \subseteq G$ (3) and $G \cap R \subseteq L$ (4). Since $G \in \mathbf{Sub}(T)$ and (3), all arrows of (1) are in $\mathbf{Sub}(T)$ and by Lemma 5 we have (b) $L \cap D \cong K$ and (a) $L \cup D \cong G$.

Clearly $K \subseteq D \cap R$. Now $D \cap R \cong D \cap G \cap R \subseteq_{(4)} D \cap L \cong K$ and thus we conclude that condition (d): $K \cong D \cap R$ holds. Because square (2) is a pushout it follows that (c): $D \cup R \cong G'$. □

The following example shows that in the presence of a contact situation, a double-pushout diagram in **C** does not correspond in general to a direct derivation in the STS. More precisely, let **C** be the (adhesive) category of sets and functions, and let $T = \{\bullet\}$ be a singleton set. Then the top span is a production in $\mathbf{Sub}(T)$, and arrow m is in $\mathbf{Sub}(T)$ as well, but condition $G \cap R \subseteq L$ is not satisfied. The double-pushout diagram can be completed in **Set** as shown, but the resulting set G' is not a subobject of T.

$$\begin{array}{ccccc} L = \emptyset & \xleftarrow{l} & K = \emptyset & \xrightarrow{r} & R = \{\bullet\} \\ {\scriptstyle m}\downarrow & (1) & \downarrow{\scriptstyle k} & (2) & \downarrow{\scriptstyle n} \\ G = \{\bullet\} & \xleftarrow[f]{} & D = \{\bullet\} & \xrightarrow[g]{} & G' = \{\bullet, \bullet\} \end{array}$$

As a consequence of the fact that a direct derivation in an STS implies a direct derivation in the standard DPO approach, we can immediately derive several properties of a derivation; in particular, we prove its determinacy below.

Proposition 7 (Determinacy of STS derivations) *Suppose that $\mathcal{S} = \langle T, P, \pi \rangle$ is an STS over an adhesive category, q is a production, and G is an object of* $\mathbf{Sub}(T)$. *Then the context of G w.r.t. q is unique up to isomorphism, if it exists. As a consequence the target of a direct derivation is determined uniquely up to isomorphism: if $G \Rightarrow^q G'$ and $G \Rightarrow^q G''$ then $G' \cong G''$.*

Proof If D is a context of G w.r.t. q, by Proposition 6 it is a pushout complement of $K \subseteq L$ and $L \subseteq G$ in **C**. The statement follows from the uniqueness up to isomorphism of pushout complements along monos in adhesive categories [16].

3 Relations among Productions

The theory of the DPO approach to the transformation of graphs includes several results and constructions which aim at a higher level of abstraction in the analysis of the computations (concretely, linear sequences of double-pushout diagrams) of a system. For example, typically one does not want to consider as distinct two derivations which differ only in the order in which "independent" productions are applied: this led to the definition of *shift equivalence* [15], and more recently to notions and constructions borrowed from the theory of Petri nets, including the definition of processes [7] for DPO systems and the unfolding construction [1, 20].

A key ingredient in the definitions of equivalences on derivations and in the aforementioned constructions such as processes and unfoldings is the analysis of the relationships which hold among the occurrences of rules in the possible computations of the given system. Such relations include for example the classical *parallel* and *sequential independence*, *causality* and *conflict*, *asymmetric conflict*, and the less known *co-causality*, *disabling* and *co-disabling*, recently introduced in [2].

Typically, these relationships are defined over *production occurrences* of the original system with respect to either a given derivation (as for sequential independence or causality), or a branching structure of derivations (like conflict and asymmetric

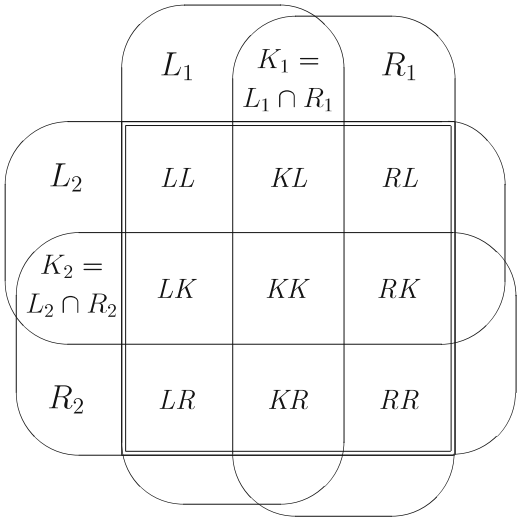
conflict), and they are determined by looking at the way the production occurrences overlap. Consider a simple example: if an item x in an occurrence graph grammar is generated by production q_1 (i.e., $x \in R_1 \setminus K_1$) and it is consumed by q_2 ($x \in L_2 \setminus K_2$), then q_1 (directly) causes q_2.

In this section we present a complete analysis of the relationships that may hold among the productions of a Subobject Transformation System. For the rest of the paper, we shall assume every STS to be *pure*, i.e., such that $K = L \cap R$ for each production, leaving a deeper study of non-pure systems as a topic of future work. For the goals of the present paper this assumption is not a limitation, because the STSs arising as representation of computations of a DPO system, including processes and unfoldings, are always pure: this is proved explicitly in Section 6 for the STS obtained from a derivation tree of a DPO system, as described in Section 5.

Recall that given a pure STS $\mathcal{S} = \langle T, P, \pi \rangle$ each production name $q \in P$ is associated with a production, which is a triple of subobjects $\pi(q) = \langle L_q, K_q, R_q \rangle$ with $K_q = L_q \cap R_q$. The Venn diagram of Fig. 3 shows the way two productions $q_1 = \langle L_1, K_1, R_1 \rangle$ and $q_2 = \langle L_2, K_2, R_2 \rangle$ can overlap.

In general, the intersection of q_1 and q_2, i.e., the subobject $(L_1 \cup R_1) \cap (L_2 \cup R_2)$ marked with a double border in the diagram, is composed of nine zones, denoted XY for $X, Y \in \{L, K, R\}$. Productions q_1 and q_2 can be considered as completely independent if their intersection is preserved by both productions, i.e., if it is contained in subobject $KK = K_1 \cap K_2$: this notion of independence will be formalized in Section 4. Each zone (but for KK) determines a *basic relation* between the two productions, which holds iff "the zone is not empty"; for example, the non-emptiness of RL would witness that q_1 causes q_2. Notice however that but for KK, the remaining zones in the intersection of q_1 and q_2 are not subobjects in general, because $\mathbf{Sub}(T)$ might not be

Fig. 3 Intersection of two pure productions



a boolean lattice. As a consequence we introduce *regions*, which correspond in Venn diagrams to the zones which are "complements" of subobjects.

Definition 8 (Region) Let $\equiv$ be the smallest equivalence relation on $\mathbf{Sub}(T) \times \mathbf{Sub}(T)$ which contains the following, for all subobjects U, V, W such that $W \cap U \subseteq V$, and Z such that $Z \subseteq U \cup V \cup W$:

$$(U, V) \equiv (U \cup Z, V \cup W)$$

A *region* is an equivalence class of pairs of subobjects (U, V) with respect to $\equiv$; we shall write $U \setminus V$ for the region containing (U, V).

In a Venn diagram we can identify a zone a (which may not be a subobject) with a region in the following way. We first identify a subobject A which includes a such that the complement of a in A, in the geometrical sense, corresponds to a subobject C; if such an A exists, next we take any subobject B such that its intersection with A is C. Then we say that a corresponds to region $A \setminus B$. For example, in Fig. 2a, zone a represents region $A \setminus B$ and b represents $B \setminus A$. By exploiting the equivalence relation of Definition 8 it can be shown that the region identified in the way described above is unique: for example, in Fig. 3, applying the construction to zone RL we would get region $R_1 \cap L_2 \setminus K_1 \cup K_2$, but also region $R_1 \cap L_2 \setminus L_1 \cup R_2$, which are provably the same.

We will call *basic regions* those represented, in Fig. 3, by XY for $X, Y \in \{L, K, R\}$, excluding KK which is not a region. Non-basic regions are for example $RL{+}RK$ and $KL{+}RK$, corresponding to $R_1 \cap L_2 \setminus K_1$ and $(K_1 \cap L_2) \cup (K_2 \cap R_1) \setminus K_1 \cap K_2$, respectively. Instead, for example $RL{+}KK$ does not represent a region, because it is not the complement of a subobject.

Definition 9 (Basic regions of intersection of productions) Let $q_1 = \langle L_1, K_1, R_1 \rangle$ and $q_2 = \langle L_2, K_2, R_2 \rangle$ be two productions of an STS. Then the *basic regions* of their intersection (see Fig. 3) are the following, with $X, Y \in \{L, R\}$:

- $XY = X_1 \cap Y_2 \setminus K_1 \cup K_2$,
- $KX = K_1 \cap X_2 \setminus K_2$, and
- $XK = X_1 \cap K_2 \setminus K_1$.

Our main tool for the analysis of regions is a simple notion of emptiness; since regions are equivalence classes we must show that this notion of emptiness is independent from the chosen element of a class.

Definition 10 (Emptiness) A region $U \setminus V$ is said to be empty when $U \subseteq V$.

Lemma 11 *The notion of emptiness is well-defined.*

Proof We show that

$$U \subseteq V \quad \Leftrightarrow \quad U \cup Z \subseteq V \cup W$$

assuming that (1) $W \cap U \subseteq V$ and (2) $Z \subseteq U \cup V \cup W$. In fact, if $U \subseteq V$, then $U \cup Z \subseteq$ [by (2)] $U \cup V \cup W \subseteq$ [by assumption] $V \cup W$. Vice-versa, $U \cong U \cap (U \cup Z) \subseteq$ [by assumption] $U \cap (V \cup W) \cong$ [by distributivity] $(U \cap V) \cup (U \cap W) \subseteq$ [by (1)] V. □

In general, "union" of regions is difficult to define—however, if we can find representatives of regions in a particular format then we can easily show that their composition (intuitively, disjoint union) behaves as expected with respect to emptiness. For example, suppose that $U_1 \supseteq U_2 \supseteq U_3$ are subobjects. Then region $R = U_1 \setminus U_3$ is the composition of regions $R_1 = U_1 \setminus U_2$ and $R_2 = U_2 \setminus U_3$: the next result shows that, as expected, region R is empty if and only if both R_1 and R_2 are empty.

Lemma 12 (Region composition) *Given a sequence of subobjects $U_1 \supseteq U_2 \supseteq \cdots \supseteq U_n$ with $n \geq 2$, region $U_1 \setminus U_n$ is empty if and only if region $U_k \setminus U_{k+1}$ is empty for all $k \in \{1, \ldots, n-1\}$.*

Proof Let (1): $U_1 \supseteq U_2 \supseteq \cdots \supseteq U_n$ be subobjects. $U_1 \setminus U_n$ is empty $\Leftrightarrow$ $U_1 \subseteq U_n \Leftrightarrow_{[(1)]} U_1 = U_n \Leftrightarrow_{[(1)]} U_1 = U_2 = \cdots = U_n \Leftrightarrow_{[(1)]} U_1 \subseteq U_2 \subseteq \ldots \subseteq U_n \Leftrightarrow U_k \setminus U_{k+1}$ is empty for all $k \in \{1, \ldots, n-1\}$. □

We shall now introduce the basic relations among productions formally, providing a notation and two equivalent characterizations for each of them. Before that, notice that three basic regions (LK, LR, KR) do not introduce new dependencies, as they are obtained by switching the roles of q_1 and q_2: thus there remain five basic relationships.

Definition 13 (Basic relations) Let *conflict* ($\curlywedge$), *deactivation* ($<_d$), *write causality* ($<_{wc}$), *read causality* ($<_{rc}$), and *backwards conflict* ($\curlyvee$), be defined as shown in Fig. 4. For each relation, we give three definitions: in terms of a particular *non*-inclusion of subobjects, in terms of a certain commutative diagram in **C** *not* being a pushout and by the *non*-emptiness of a basic region.

For each relation the three definitions are easily shown to be equivalent. Consider for example $q_1 \curlywedge q_2$: the area LL represents, according to Definition 9, region $L_1 \cap L_2 \setminus K_1 \cup K_2$. Thus by Definition 10 it is not empty iff $L_1 \cap L_2 \nsubseteq K_1 \cup K_2$. Furthermore, by Lemma 5, this holds if and only if the diagram in the fourth column is not a pushout, because clearly $L_1 \cup L_2 \cong (L_1 \cup K_2) \cup (K_1 \cup L_2)$. As indicated by our notation and easily seen from the definition, the two conflicts are symmetric, while the three forms of causality are not.

It is instructive to consider what the five relations mean in particular settings, say for instance in **Graph**. Fixing an ambient graph T, the objects of $\mathbf{Sub}(T)$ are the subgraphs of T. In the following, we refer to the individual vertices or edges of T as *elements*. In this setting, the basic relations can be characterised as follows:

Conflict: $q_1 \curlywedge q_2$ precisely when q_1 consumes an element, which is also consumed by q_2. Both productions are then competing with each other for their application.

Name	Symbol	Inequation	Diagram in **C**	Non-empty region
Conflict	$q_1 \between q_2$	$L_1 \cap L_2 \not\subseteq K_1 \cup K_2$	$\begin{array}{ccc} K_1 \cup K_2 & \rightarrowtail & L_1 \cup K_2 \\ \downarrow & \neg\text{PO} & \downarrow \\ K_1 \cup L_2 & \rightarrowtail & L_1 \cup L_2 \end{array}$	LL
Deactivation	$q_1 <_d q_2$	$K_1 \cap L_2 \not\subseteq K_2$	$\begin{array}{ccc} K_2 & \rightarrowtail & L_2 \\ \downarrow & \neg\text{PO} & \downarrow \\ K_1 \cup K_2 & \rightarrowtail & K_1 \cup L_2 \end{array}$	KL
Write causality	$q_1 <_{wc} q_2$	$R_1 \cap L_2 \not\subseteq K_1 \cup K_2$	$\begin{array}{ccc} K_1 \cup K_2 & \rightarrowtail & R_1 \cup K_2 \\ \downarrow & \neg\text{PO} & \downarrow \\ K_1 \cup L_2 & \rightarrowtail & R_1 \cup L_2 \end{array}$	RL
Read causality	$q_1 <_{rc} q_2$	$R_1 \cap K_2 \not\subseteq K_1$	$\begin{array}{ccc} K_1 & \rightarrowtail & R_1 \\ \downarrow & \neg\text{PO} & \downarrow \\ K_1 \cup K_2 & \rightarrowtail & R_1 \cup K_2 \end{array}$	RK
Backward conflict	$q_1 \curlyvee q_2$	$R_1 \cap R_2 \not\subseteq K_1 \cup K_2$	$\begin{array}{ccc} K_1 \cup K_2 & \rightarrowtail & R_1 \cup K_2 \\ \downarrow & \neg\text{PO} & \downarrow \\ K_1 \cup R_2 & \rightarrowtail & R_1 \cup R_2 \end{array}$	RR

Fig. 4 Relations between productions in STS; $\pi(q_i) = \langle L_i, K_i, R_i \rangle$ for $i \in \{1, 2\}$

Deactivation: $q_1 <_d q_2$ precisely when q_1 preserves an element, which is consumed by q_2. Therefore q_2 "deactivates" q_1 meaning that q_1 is not applicable afterward.

Write causality: $q_1 <_{wc} q_2$ precisely when q_1 produces an element, which is consumed by q_2.

Read causality: $q_1 <_{rc} q_2$ precisely when q_1 produces an element, which is used but not consumed by q_2.

Backwards conflict: $q_1 \curlyvee q_2$ precisely when q_1 produces an element, which is also produced by q_2.

Given a production q with $\pi(q) = \langle L, K, R \rangle$, we write q^{op} for the *inverse production* $\langle R, K, L \rangle$: notice that it follows immediately from Definition 3 that $G \Rightarrow^q G'$ if and only if $G' \Rightarrow^{q^{\text{op}}} G$. Using this "swapping" of the left- and right-hand sides of the production, we can derive a number of useful equivalences among the basic relations.

Lemma 14 (Laws for relations)

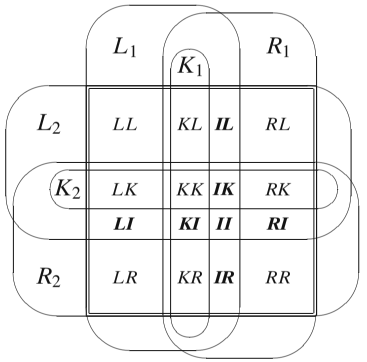
$<_{rc}$ and $<_d$	$<_{wc}$, $\between$, and $\curlyvee$
(a) $q_1 <_{rc} q_2$	(e) $q_1 <_{wc} q_2$
(b) $\Leftrightarrow q_1 <_{rc} q_2^{\text{op}}$	(f) $\Leftrightarrow q_2^{\text{op}} <_{wc} q_1^{\text{op}}$
(c) $\Leftrightarrow q_2 <_d q_1^{\text{op}}$	(g) $\Leftrightarrow q_1^{\text{op}} \between q_2$
(d) $\Leftrightarrow q_2^{\text{op}} <_d q_1^{\text{op}}$;	(h) $\Leftrightarrow q_1 \curlyvee q_2^{\text{op}}$;

Proof The equivalences follow immediately from the definitions. □

The equivalences just listed show that using the $\_^{\text{op}}$ operator on productions, relations $\curlyvee$, $\between$ and $<_{wc}$ are mutually definable, and so are $<_{rc}$ and $<_d$.

Proposition 15 (Completeness) *The inverse production operator* $\_^{\text{op}}$ *together with any pair of relations in* $\{\curlyvee, \between, <_{wc}\} \times \{<_{rc}, <_d\}$ *form a complete basis to model all possible relations of Fig. 4.*

It is worth mentioning here that considering possibly *non-pure* STSs, the set of basic relations would be larger. In fact, in this case K_i could be a proper subobject of $L_i \cap R_i$, for $i \in \{1, 2\}$, as depicted in the Venn diagram below, and therefore seven new regions would arise in the intersection of two productions, denoted $\boldsymbol{IX}$ or $\boldsymbol{XI}$ for $X \in \{\boldsymbol{L}, \boldsymbol{K}, \boldsymbol{R}\}$. For example, the region marked $\boldsymbol{IL}$ is not empty if production q_1 consumes and produces again an item that is consumed by q_2. However, as discussed in the concluding section, the relevance of such new dependency relations is not clear, and their study goes beyond the goal of the present paper.



Let us consider now a few more complex relations among productions that have been introduced in [2]. We show that, as expected, they can be defined easily by exploiting the basic relationships of Definition 13. For example, the (compound) *causality* relation is built up by taking the union of the *read causality* and the *write causality* relations.

Definition 16 (Compound relations) Let *causality* ($<$), *disabling* ($\ll$), *co-causality* ($<_{\text{co}}$) and *co-disabling* ($\ll_{\text{co}}$) be defined as shown in Fig. 5. For each relation, the equational definition in the third column is the one introduced in [2], while the diagrammatical one in the fourth column and the non-emptiness requirement for the non-basic region in the fifth column are equivalent by the considerations after Definition 13.

Name	Symbol	Inequation	Diagram	Non-empty region
Causality	$q_1 < q_2$	$R_1 \cap L_2 \not\subseteq K_1$	$K_1 \rightarrowtail R_1$; $K_1 \cup L_2 \rightarrowtail R_1 \cup L_2$; $\neg$PO	$RL{+}RK$
Disabling	$q_1 \ll q_2$	$L_1 \cap L_2 \not\subseteq K_2$	$K_2 \rightarrowtail L_2$; $L_1 \cup K_2 \rightarrowtail L_1 \cup L_2$; $\neg$PO	$LL{+}KL$
Co-causality	$q_1 <_{co} q_2$	$L_2 \cap R_1 \not\subseteq K_2$	$K_2 \rightarrowtail L_2$; $R_1 \cup K_2 \rightarrowtail R_1 \cup L_2$; $\neg$PO	$KL{+}RL$
Co-disabling	$q_1 \ll_{co} q_2$	$R_1 \cap R_2 \not\subseteq K_1$	$K_1 \rightarrowtail R_1$; $K_1 \cup R_2 \rightarrowtail R_1 \cup R_2$; $\neg$PO	$RK{+}RR$

Fig. 5 Compound relations; $\pi(q_i) = \langle L_i, K_i, R_i \rangle$ for $i \in \{0, 1\}$

Lemma 17 (Characterization of compound relations) *Each one of the compound relations of Fig. 5 can be obtained as the disjunction of two basic relations of Fig. 4, as follows:*

- (*Causality*) $q_1 < q_2 \;\Leftrightarrow\; q_1 <_{rc} q_2 \;\vee\; q_1 <_{wc} q_2$;
- (*Disabling*) $q_1 \ll q_2 \;\Leftrightarrow\; q_1 <_{d} q_2 \;\vee\; q_2 \not\perp q_1$;
- (*Co-causality*) $q_1 <_{co} q_2 \;\Leftrightarrow\; q_1 <_{d} q_2 \;\vee\; q_2 <_{wc} q_1$;
- (*Co-disabling*) $q_1 \ll_{co} q_2 \;\Leftrightarrow\; q_1 <_{wc} q_2 \;\vee\; q_1 \not\top q_2$.

Proof We shall prove the statement for *causality*: The other cases of compound relations are similar. In terms of regions, the statement can be read equivalently as *region RL+RK is not empty iff either RL or RK is not empty*, and thus *region RL+RK is empty iff RL and RK are empty*. Now let $U_1 = R_1 \cap L_2$, $U_2 = (K_1 \cap L_2) \cup (R_1 \cap K_2)$ and $U_3 = K_1 \cap L_2$. It is straightforward to check that RL represents $U_1 \setminus U_2$, RK represents $U_2 \setminus U_3$, and $RL{+}RK$ represents $U_1 \setminus U_3$; furthermore since $U_1 \supseteq U_2 \supseteq U_3$, we can conclude by Lemma 12.

It is instructive to compare the given proof with the following one, which uses the equivalent definitions given by diagrams in **C** not being pushouts.

($\Rightarrow$) By contraposition this direction is equivalent to

$$q_1 \not<_{rc} q_2 \text{ and } q_1 \not<_{wc} q_2 \text{ implies } q_1 \not< q_2.$$

Since $q_1 \not<_{rc} q_2$ and $q_1 \not<_{wc} q_2$ mean that the two corresponding diagrams of Fig. 4 are pushouts, they can be composed to form the pushout in Fig. 5, as illustrated.

$$\begin{array}{ccc} K_1 & \rightarrowtail & R_1 \\ \downarrowtail & (1) & \downarrowtail \\ K_1 \cup K_2 & \rightarrowtail & R_1 \cup K_2 \\ \downarrowtail & (2) & \downarrowtail \\ K_1 \cup L_2 & \rightarrowtail & R_1 \cup L_2 \end{array}$$

($\Leftarrow$) Again using contraposition, let the diagram $(1 + 2)$ be a pushout; we have to show that both diagrams (1) and (2) are pushouts. Since $(1 + 2)$ is a pushout, using Lemma 5 for the first step and distributivity for the second, we have:

$$(\dagger)\; K_1 \cong (K_1 \cup L_2) \cap R_1 \cong (R_1 \cap K_1) \cup (R_1 \cap L_2) \cong K_1 \cup (R_1 \cap L_2).$$

Now, concerning diagram (1): $(K_1 \cup K_2) \cup R_1 \cong R_1 \cup K_2$ and using $(\dagger)$ for the last step: $(K_1 \cup K_2) \cap R_1 = K_1 \cup (R_1 \cap K_2) \subseteq K_1 \cup (R_1 \cap L_2) \cong K_1$. Using Lemma 5 (1) is a pushout in **C**.

Analogously for diagram (2): $(K_1 \cup L_2) \cup (R_1 \cup K_2) \cong R_1 \cup L_2$ and with $(\dagger)$ for the last step: $(K_1 \cup L_2) \cap (R_1 \cup K_2) \cong K_1 \cup (K_1 \cap K_2) \cup (R_1 \cap L_2) \cup K_2 \cong K_1 \cup K_2$, thus with Lemma 5 also (2) is a pushout in **C**. □

4 Independence in Subobject Transformation Systems

Based on the relations among productions introduced above, we develop here a theory of independence for STSs which follows the outline of the classical theory of the DPO approach. Interestingly, in our formal framework there is a single notion of independence among productions, which corresponds to both *parallel* and *sequential independence* of the DPO approach, as made precise in Section 6.

Two productions of an STS are *independent* if their respective applications do not interfere: from the discussion before Definition 13 this holds if their intersection is contained in $K_1 \cap K_2$. All along this section, we assume that $\mathcal{S} = \langle T, P, \pi \rangle$ is an arbitrary but fixed pure STS, and that for each production name $q_i \in P$, the corresponding production is $\pi(q_i) = \langle L_i, K_i, R_i \rangle$.

Definition 18 (Independence of productions in STS) Two productions q_1 and q_2 of $\mathcal{S}$ are *independent*, denoted $q_1 \lozenge q_2$, if

$$(L_1 \cup R_1) \cap (L_2 \cup R_2) \subseteq (K_1 \cap K_2)$$

Therefore, by definition, two productions are independent if the compound region $LL{+}KL{+}RL{+}LK{+}RK{+}LR{+}KR{+}RR$ is empty. By exploiting Lemma 12 we show that this holds if and only if all the basic regions are empty, i.e., iff the productions are not related via any of the basic causality or conflict relations listed in Fig. 4, nor by any of the symmetric variations.

Theorem 19 *Two productions q_1 and q_2 of $\mathcal{S}$ are independent if and only if all basic regions in $(L_1 \cup R_1) \cap (L_2 \cup R_2)$ are empty.*

Proof Let q_1 and q_2 be two productions and $C_1 = L_1 \cup R_1$, $C_2 = L_2 \cup R_2$. Consider the following sequence of nine subobjects $U_1 \supseteq U_2 \supseteq \cdots \supseteq U_9$, where for each subobject we list the zones of $C_1 \cap C_2$ that identify it:

$U_1 = C_1 \cap C_2$	(all zones),
$U_2 = (R_1 \cup R_2) \cap U_1$	$(KL, RL, LK, KK, RK, LR, KR, RR)$,
$U_3 = (K_1 \cup R_2) \cap U_1$	$(KL, LK, KK, RK, LR, KR, RR)$,
$U_4 = (C_1 \cap K_2) \cup (K_1 \cap C_2) \cup (R_1 \cap R_2)$	(KL, LK, KK, RK, KR, RR),
$U_5 = (K_1 \cup K_2) \cap U_1$	(KL, LK, KK, RK, KR),
$U_6 = [(K_1 \cap R_2) \cup K_2] \cap U_1$	(LK, KK, RK, KR),
$U_7 = (K_1 \cap R_2) \cup (R_1 \cap K_2)$	(KK, RK, KR),
$U_8 = (K_1 \cap R_2)$	(KK, KR), and
$U_9 = K_1 \cap K_2$	(KK).

The eight basic regions of the Venn diagram, that we redraw below for the reader's convenience, are identified as follows:

$$LL = U_1 \setminus U_2,\ KL = U_5 \setminus U_6,\ RL = U_2 \setminus U_3,\ LK = U_6 \setminus U_7,$$
$$RK = U_7 \setminus U_8,\ LR = U_3 \setminus U_4,\ KR = U_8 \setminus U_9,\ \text{ and } \ RR = U_4 \setminus U_5.$$

It is easy to check that this is consistent with Definition 9. For example, considering zone KR, we have $(U_8, U_9) \equiv (K_1 \cap R_2, K_2)$ according to Definition 8. Lemma 12 applies to the chain of subobjects $U_1, \ldots, U_9$. Therefore, $U_1 \setminus U_9 = C_1 \cap C_1 \setminus K_1 \cap K_2$ is empty if and only if $U_1 \setminus U_2, \ldots,$ and $U_8 \setminus U_9$ are empty, which are the basic regions. □

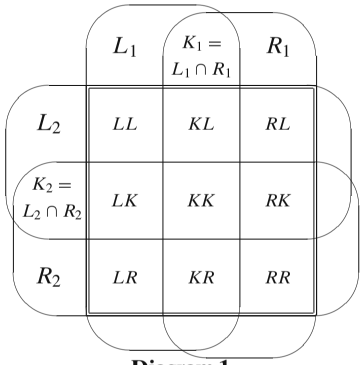


Diagram 1

Intersection of two pure productions

The next lemma shows that if two productions of an STS are applicable to the same subobject, then in order to check that they are independent it is enough to consider only a subset of the possible relations among them. We also provide an alternative

characterization of independence, which is analogous to the classical definition of parallel independence in the DPO approach (see Definition 31).

Lemma 20 (Characterization of independence in STSs (I)) *Suppose that there are direct derivations $G \Rightarrow^{q_1} G_1$ and $G \Rightarrow^{q_2} G_2$ in $\mathcal{S}$. Then the following are equivalent:*

(1) $q_1 \Diamond q_2$
(2) $\neg(q_1 \not\downarrow q_2) \ \wedge\ \neg(q_1 \not\uparrow q_2) \ \wedge\ q_1 \not<_d q_2 \ \wedge\ q_2 \not<_d q_1$
(3) $L_1 \subseteq D_2 \ \wedge\ L_2 \subseteq D_1 \ \wedge\ \neg(q_1 \not\uparrow q_2)$, *where D_1 and D_2 are the contexts of the first and of the second direct derivations, respectively.*

Proof

(1 ⇒ 2) Immediate by Theorem 19.
(1 ⇒ 2) We have to show that if (1) $\neg(q_1 \not\downarrow q_2)$, (2) $\neg(q_1 \not\uparrow q_2)$, (3) $q_1 \not<_d q_2$ and (4) $q_2 \not<_d q_1$ then $(L_1 \cup R_1) \cap (L_2 \cup R_2) \subseteq (K_1 \cap K_2)$.

Consider Diagram 1 and the relations of Fig. 4: (1) implies that region LL is empty, and similarly (2) for RR, (3) for KL and (4) for LK. Furthermore, since $G \Rightarrow^{q_1} G_1$ by hypothesis, by Proposition 6 we have $G \cap R_1 \subseteq L_1$, and since $G \Rightarrow^{q_2} G_2$, we know that $L_2 \subseteq G$; thus we infer $L_2 \cap R_1 \subseteq L_1$, which implies that the non-basic region $RL{+}RK$ is empty. By Lemma 12 this implies that both RL and RK are empty; in fact, let $U_1 = R_1 \cap L_2$, $U_2 = (K_1 \cup K_2) \cap (R_1 \cap L_2)$, and $U_3 = K_1 \cap L_2$: we have that $U_1 \supseteq U_2 \supseteq U_3$, $RL = U_1 \setminus U_2$, $RK = U_2 \setminus U_3$, and $RL + RK = U_1 \setminus U_3$. A symmetric argument, switching q_1 and q_2, shows that LR and KR are empty as well. Therefore the eight basic regions of Definition 9 are all empty, and we conclude by Theorem 19.

(1 ⇒ 3) By hypothesis and Definition 3 we know that $G \cong L_1 \cup D_1$ and $G \cong L_2 \cup D_2$, and that $L_i \cap D_i \cong K_i$ for $i \in \{1, 2\}$. Thus object G can be seen as composed of nine zones as drawn in Diagram 2, where, for example, L_1 is made of the first two columns, D_1 of the second and third ones, and K_1, their intersection, of the mid column. From $q_1 \Diamond q_2$ it follows $L_1 \cap L_2 \subseteq K_1 \cap K_2$, thus region $LL{+}LK{+}KL$ is empty. Now let $U_1 = L_1 \cap L_2$, $U_2 = L_1 \cap K_2$, and $U_3 = K_1 \cap K_2$.

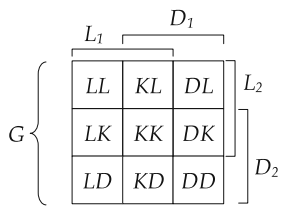


Diagram 2

Decomposition of object G

Since $U_1 \supseteq U_2 \supseteq U_3$, $LK + LL + KL = U_1 \setminus U_3$, $LL + KL = U_1 \setminus U_2$, and $LK = U_2 \setminus U_3$, by Lemma 12 we conclude that $LL{+}KL$ is empty, and thus $L_1 \subseteq D_2$. A similar argument shows that $L_2 \subseteq D_1$.

(3 $\Rightarrow$ 2) Since $L_1 \subseteq D_2$ and $L_2 \subseteq D_1$, we have $L_1 \cap L_2 \subseteq D_1 \cap D_2$. Thus in Diagram 2 region $LL{+}KL{+}LK$ is empty, and so are regions LL, KL and LK by an easy application of Lemma 12. This implies (see Fig. 4) that $\neg(q_1 \between q_2)$, $q_1 \not<_d q_2$ and $q_2 \not<_d q_1$. □

Similar characterizations of independence can be provided if the two productions can be applied in sequence: By using the inverse of a production, the proof is reduced to that of the previous lemma.

Lemma 21 (Characterization of independence in STSs (II)) *Suppose that there are direct derivations $G \Rightarrow^{q_1} G_1 \Rightarrow^{q_2} G_2$ in $\mathcal{S}$. Then the following are equivalent:*

(1) $q_1 \lozenge q_2$
(2) $q_1 \not<_{rc} q_2 \ \wedge\ q_1 \not<_{wc} q_2 \ \wedge\ q_2 \not<_{wc} q_1 \ \wedge\ q_1 \not<_d q_2$
(3) $R_1 \subseteq D_2 \ \wedge\ L_2 \subseteq D_1 \ \wedge\ q_2 \not<_{wc} q_1$, *where D_1 and D_2 are the contexts of the first and of the second direct derivations, respectively.*

Proof Observe that $G \Rightarrow^{q_1} G_1$ iff $G_1 \Rightarrow^{q_1^{\mathrm{op}}} G$. Furthermore, by Lemma 14 we have

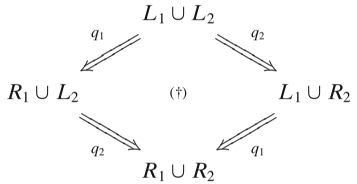
$$
\begin{array}{ll}
q_1 \not<_{rc} q_2 \Leftrightarrow q_2 \not<_d q_1^{\mathrm{op}} & \text{using } (a) \Leftrightarrow (c) \\
q_1 \not<_{wc} q_2 \Leftrightarrow \neg(q_1^{\mathrm{op}} \between q_2) & \text{using } (e) \Leftrightarrow (g) \\
q_2 \not<_{wc} q_1 \Leftrightarrow \neg(q_1^{\mathrm{op}} \curlyvee q_2) & \text{using } (e) \Leftrightarrow (h) \\
q_1 \not<_d q_2 \Leftrightarrow q_1^{\mathrm{op}} \not<_d q_2. & \text{using } (c) \Leftrightarrow (d).
\end{array}
$$

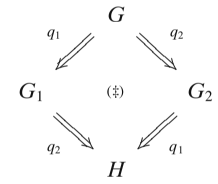
Then the statement follows by Lemma 20, observing that $q_1^{\mathrm{op}} \lozenge q_2$ holds, by definition, if and only if $q_1 \lozenge q_2$. □

The next result rephrases in the setting of Subobject Transformation Systems the well-known local Church–Rosser theorem of the DPO approach.

Theorem 22 (Local Church–Rosser for STSs) *Let q_1 and q_2 be two independent productions of $\mathcal{S}$. Then:*

(1) *There are direct derivations as in diagram (†) below.*
(2) *If there are direct derivations $G \Rightarrow^{q_1} G_1$ and $G \Rightarrow^{q_2} G_2$, then there is an object H in* **Sub**(T) *and direct derivations $G_1 \Rightarrow^{q_2} H$ and $G_2 \Rightarrow^{q_1} H$, as in diagram (‡) below.*
(3) *If there are direct derivations $G \Rightarrow^{q_1} G_1 \Rightarrow^{q_2} H$, then there is an object G_2 in* **Sub**(T) *and direct derivations $G \Rightarrow^{q_2} G_2$ and $G_2 \Rightarrow^{q_1} H$, as in diagram (‡).*





Proof

(1) It is easy to check that $(L_1 \cup L_2) \Rightarrow^{q_1} (R_1 \cup L_2)$ with context $D_1 \stackrel{\mathrm{def}}{=} L_2 \cup K_1$. In fact, the conditions (a – d) of Definition 3 reduce to
   (a) $L_1 \cup (L_2 \cup K_1) \cong L_1 \cup L_2$, obvious because $K_1 \subseteq L_1$;
   (b) $L_1 \cap (L_2 \cup K_1) \cong (L_1 \cap L_2) \cup (L_1 \cap K_1) \cong K_1$, by distributivity and independence;
   (c) $(L_2 \cup K_1) \cup R_1 \cong R_1 \cup L_2$, obvious because $K_1 \subseteq R_1$;
   (d) $(L_2 \cup K_1) \cap R_1 \cong (L_2 \cap R_1) \cup (K_1 \cap R_1) \cong K_1$, by distributivity and independence.

The other direct derivations are similar, using as contexts (clockwise) $L_1 \cup K_2$, $K_1 \cup R_2$, and $R_1 \cup K_2$.

(2) Let $H \stackrel{\mathrm{def}}{=} (D_1 \cap D_2) \cup R_1 \cup R_2$, where D_1 and D_2 are the contexts of the first and of the second direct derivations, respectively.

Let us show that $G_1 \Rightarrow^{q_2} H$: the proof that $G_2 \Rightarrow^{q_1} H$ is analogous. Let $D_2' \stackrel{\mathrm{def}}{=} (D_1 \cap D_2) \cup R_1$: we show that conditions (a – d) of Definition 3 hold for context D_2'.

   (a) $[L_2 \cup D_2' \cong G_1]$: We have $L_2 \cup D_2' = L_2 \cup ((D_1 \cap D_2) \cup R_1)$ by definition, and (*) $L_2 \cup (D_1 \cap D_2) \cong D_1$ by inspecting Diagram 2 (using $L_2 \subseteq D_1$, which follows by Lemma 20); thus we get $L_2 \cup D_2' \cong D_1 \cup R_1$, but $D_1 \cup R_1 \cong G_1$ by (c) of $G \Rightarrow^{q_1} G_1$, so we are done. More explicitly, (*) follows from $L_2 \cup (D_1 \cap D_2) \cong (L_2 \cup D_1) \cap (L_2 \cup D_2) \cong_{\{by\ L_2 \subseteq D_1\}} D_1 \cap G \cong D_1$.
   (b) $[L_2 \cap D_2' \cong K_2]$: Expanding D_2' and distributing we get $L_2 \cap D_2' = L_2 \cap ((D_1 \cap D_2) \cup R_1) \cong (L_2 \cap D_1 \cap D_2) \cup (L_2 \cap R_1)$; the statement follows observing that (†) $L_2 \cap D_1 \cap D_2 \cong_{\{by\ (b)\ of\ G \Rightarrow^{q_2} G_2\}} K_2 \cap D_1 \cong_{\{by\ L_2 \subseteq D_1\}} K_2$, and that by independence we have $L_2 \cap R_1 \subseteq K_1 \cap K_2 \subseteq K_2$.
   (c) $[D_2' \cup R_2 \cong H]$: Obvious, by expanding the definitions of H and D_2'.
   (d) $[D_2' \cap R_2 \cong K_2]$: Expanding D_2' and distributing we get $D_2' \cap R_2 \cong (D_1 \cap D_2 \cap R_2) \cup (R_1 \cap R_2)$; analogous to (†), the first argument of the union is K_2, and by independence the second one is included in K_2, allowing us to conclude.

(3) This point reduces to the previous one by observing that $G \Rightarrow^{q_1} G_1$ if and only if $G_1 \Rightarrow^{q_1^{\mathrm{op}}} G$, and $q_1 \lozenge q_2$ if and only if $q_1^{\mathrm{op}} \lozenge q_2$. □

5 From Derivation Trees to Subobject Transformation Systems

Here we shall outline an application of the theory of STS developed in the previous sections. In particular, in this section we show that starting with a derivation tree in an arbitrary adhesive grammar $\mathcal{G}$, we obtain an STS via a familiar colimit construction, that can be considered as a generalization to the non-deterministic case of the synthesis of a process from a linear derivation [7]. As shown in the next section, we are then able to apply the local analysis using relations between productions in the resulting STS in order to completely characterise all the independence in the original derivation tree.

In order to illustrate how the transformation from a derivation tree for $\mathcal{G}$ to an STS works, it is helpful to consider a concrete example. Suppose that $\mathcal{G}$ is an adhesive grammar containing productions q_1, q_2 and q_3, and that we have a derivation tree as illustrated in Fig. 6. Each step (direct derivation) in the original derivation tree α leads to a new production in the STS $Prc(\alpha)$. The type graph T of $Prc(\alpha)$ is obtained from the derivation by computing a certain colimit—for finite trees, this type of colimit exists in adhesive categories as it can be obtained by constructing successive pushouts. The objects $G_i \in \mathbf{C}$ can now be considered as subobjects of T.

As shown in the next section, the STS $Prc(\alpha)$ derived from a derivation tree α satisfies several properties, which correspond closely to those of occurrence grammars, as introduced in the traditional definition of processes for transformation systems like Petri nets and graph grammars [3, 12].

We begin by introducing the category $\mathbf{DerTree}(\mathcal{G})$ of derivation trees of an adhesive grammar G. The objects of this category are words of objects of $\mathbf{C}$ and arrows are forests of derivation trees. Given an arbitrary object $S \in \mathbf{C}$, it is possible to show that the construction sketched in the previous paragraph gives rise to a functor

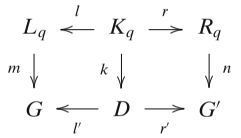
$$Prc\colon S/\mathbf{DerTree}(\mathcal{G}) \to \mathbf{STS}$$

where **STS** is the category of subobject transformation systems and their morphisms, defined by suitably restricting the usual notion of typed grammar morphisms. However, since the functorial property of the construction is not relevant for the main results of the next section, we will present the construction on objects only.

The definition of $\mathbf{DerTree}(\mathcal{G})$ uses the definition of adhesive grammars, as specified in [2, 16]; however, we do not a priori assume that our grammars are typed. An extension to typed grammars is straightforward.

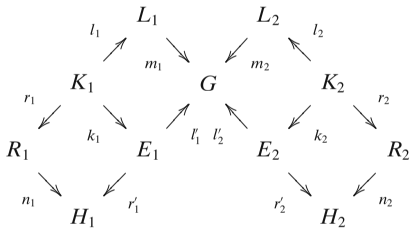
Definition 23 (Adhesive grammars) Let $\mathbf{C}$ be an adhesive category. A *production* is a span of monomorphisms $L \hookleftarrow K \hookrightarrow R$ in $\mathbf{C}$. An *adhesive grammar* over $\mathbf{C}$ is a pair $\mathcal{G} = \langle P, \pi \rangle$, where P is a set of *production names*, and π is a function which maps any $q \in P$ to a production $L_q \hookleftarrow K_q \hookrightarrow R_q$.

Definition 24 (Direct derivation) Let $\mathcal{G} = \langle P, \pi \rangle$ be a grammar, let $q \in P$, let $G, G' \in ob(\mathbf{C})$, and $m : L_q \rightarrowtail G$ be a monomorphism, called a *match*. Then q *rewrites* G *to* G' *at* m if there exists a diagram, illustrated below, consisting of two pushouts. We shall write $G \overset{q,m}{\Longrightarrow} G'$ as shorthand for such a diagram.



The derivation trees of an adhesive grammar $\mathcal{G}$ will be obtained compositionally by putting together building blocks called $\mathcal{G}$*-fans*.

Definition 25 ($\mathcal{G}$-fan) Given an adhesive grammar $\mathcal{G}$ and $G, H_1, \ldots, H_k$ ($k \geq 1$) objects of $\mathbf{C}$, a $\mathcal{G}$-fan φ from G to $H_1 \ldots H_k$, written $\varphi\colon G \to H_1 \ldots H_k$, is a diagram consisting of (one-step) direct derivations from G to H_i, for each $i \in [k] \overset{def}{=} \{1, \ldots, k\}$. As an example, we illustrate a fan $\varphi\colon G \to H_1 H_2$ below:



In simplified graphical notation, we shall denote such a fan as shown in the leftmost diagram of Fig. 7. We shall write $\mathrm{ar}(\varphi)$ for the number of productions appearing in a fan φ. Moreover, we shall abuse notation by referring to φ as a function $\varphi : [\mathrm{ar}(\varphi)] \to P$, where P is the set of productions of $\mathcal{G}$. Thus, if φ consists of two direct derivations $G \overset{q_1,m_1}{\Longrightarrow} H_1$ and $G \overset{q_2,m_2}{\Longrightarrow} H_2$ from left to right, we have $\mathrm{ar}(\varphi) = 2$, $\varphi(1) = q_1$ and $\varphi(2) = q_2$.

We shall use $\mathcal{G}$-fans to construct a strict monoidal category of derivation trees, $\mathbf{DerTree}(\mathcal{G})$. We first need to recall the notion of a *tensor scheme* [14] and the associated notion of a free monoidal category on a tensor scheme.[1]

A tensor scheme $\mathcal{T}$ consists of a set V of vertices, a set E of edges, and functions $s, t : E \to V^*$, where V^* is the free monoid (the set of finite words) on V. Every tensor scheme leads to a free strict[2] monoidal category $\mathbf{C}$—see [14] for details. Intuitively, the objects of $\mathbf{C}$ can be seen as finite words (i.e., the product in V^* is interpreted as $\otimes$ in $\mathbf{C}$) in V and the arrows of $\mathbf{C}$ are generated freely from the basic edges in E. Concretely, the arrows can be seen as certain equivalence classes or as certain string diagrams; see also [23].

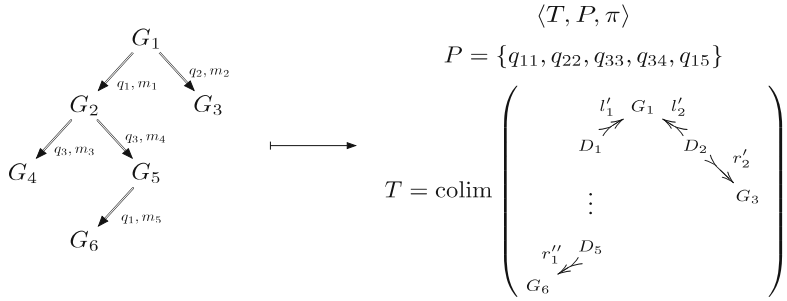


Fig. 6 Obtaining an STS from a derivation tree

[1] Tensor schemes are closely related to Petri nets in the sense of [18], see [9].

[2] The tensor product is associative "on the nose": $(A \otimes B) \otimes C = A \otimes (B \otimes C)$.

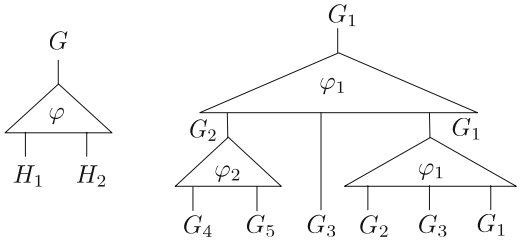


Fig. 7 A $\mathcal{G}$-fan $\varphi : G \to H_1 H_2$ and an arrow $G_1 \to G_4 \otimes G_5 \otimes G_3 \otimes G_2 \otimes G_3 \otimes G_1$ in **DerTree**($\mathcal{G}$)

For the purposes of the following definition, we assume that the underlying category **C** of $\mathcal{G}$ is a small category. Size plays a role because we shall construct a tensor scheme with the objects of **C** as its *set* of vertices. As usual, however, one could redefine the notion of tensor scheme appropriately (depending on the underlying set theory) so that the construction makes sense for an arbitrary category.

Definition 26 (DerTree($\mathcal{G}$)) Given an adhesive grammar $\mathcal{G}$ over small **C**, let $\mathcal{T}_\mathcal{G}$ denote the tensor scheme with set of vertices the objects of **C** and its edges the $\mathcal{G}$-fans. By **DerTree**($\mathcal{G}$) we denote the free strict monoidal category over $\mathcal{T}_\mathcal{G}$.

For example, for a fan $\varphi : G \to H_1 H_2$ seen as an edge of $\mathcal{T}_\mathcal{G}$ we have $s(\varphi) = G$ and $t(\varphi) = H_1 H_2$. The objects of **DerTree**($\mathcal{G}$) are finite words of objects of **C**. The arrows $G_1 \dots G_n \to H_1 \dots H_m$ are tensor products of arrows $G_1 \to H_1 \dots H_{i_1}$, $\dots G_n \to H_{i_{n-1}+1} \dots H_{i_n}$ where $i_n = m$. Such basic ingredients are constructed out of fans, an example is given in Fig. 7. Tensor product is here of course just putting such diagrams side-by-side.

One can also think of arrows of **DerTree**($\mathcal{G}$) as concrete derivation trees, constructed at each level from concrete fan derivation diagrams as illustrated in Definition 25. Indeed, this will be our usual approach. Finally, although we have defined **DerTree**($\mathcal{G}$) as a strict monoidal category, it would be, perhaps, more natural to define it as a free multicategory, see [17] for an introductory account. The following lemma relates our presentation of **DerTree**($\mathcal{G}$) with concrete derivation diagrams in **C**.

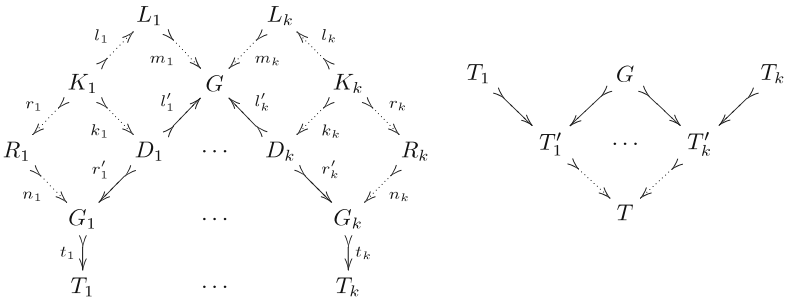
Lemma 27 *Every arrow α in* **DerTree**($\mathcal{G}$) *gives rise to a canonical diagram Der(α) in* **C** *which witnesses the derivation tree.*

Proof Recall from Definition 25 that a $\mathcal{G}$-fan is a concrete diagram in **C**. An arrow in **DerTree**($\mathcal{G}$) is a formal construction which combines fans. Using the fact that **DerTree**($\mathcal{G}$) is freely constructed from the fans, any arrow α can be deconstructed (usually not uniquely) as a $\alpha = \alpha_n \dots \alpha_1$ where each α_i is of the form $\mathrm{id}_{X'} \otimes \varphi_i \otimes \mathrm{id}_{X''}$ and φ_i is a fan. Since the codomain of each α_i agrees with the domain of each α_{i+1}, we can construct the diagram iteratively, starting with the fan φ_1, at each step pasting the corresponding fan at the correct place according to the domain and the codomain of the αs. It is clear that any decomposition of α gives rise to the same diagram. □

Given an object $S \in \mathbf{C}$, the slice category $S/$**DerTree**($\mathcal{G}$) has as objects the derivation trees from S and as arrows extensions of such trees. We shall show that each derivation tree naturally leads to an STS.

Theorem 28 *Suppose that Der(α) is the canonical diagram of a derivation tree $\alpha \in S/$***DerTree**($\mathcal{G}$). *Let T denote Colim(Der(α)). Then the canonical morphism $S \to T$ is mono. Moreover, for each fan φ in α and $i \in [\mathrm{ar}(\varphi)]$, the canonical morphisms $L_{\varphi(i)} \to T$, $K_{\varphi(i)} \to T$ and $R_{\varphi(i)} \to T$ are mono.*

Proof First note that all arrows in *Der*(α) are mono, because productions are pairs of monos, matches are mono, and monos are stable under pushouts in adhesive categories. We proceed by simple induction on any decomposition $\alpha = \alpha_n \dots \alpha_1$. The base case is trivial. For the inductive step, suppose that $\alpha_i = \mathrm{id}_{X'} \otimes \varphi_i \otimes \mathrm{id}_{X''}$ and $\varphi_i : G \to G_1 \dots G_k$. Let β_i be the derivation diagram corresponding to the derivation tree at G_i. By the inductive hypothesis the morphisms $G_i \to T_i = Colim(Der(\beta_i))$ are mono, and the canonical morphisms from each production appearing in those derivation trees to T_i are mono. Given the above, we shall construct an object T which is the colimit of the diagram below left.

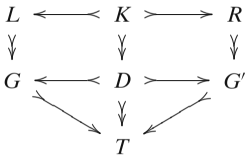


To calculate the colimit of such a diagram it is enough to consider the solid morphisms, because all squares are pushouts and colimits compose. Since the fan is of finite arity, we can calculate the colimit by constructing successive pushouts. Indeed, for each $i \in [k]$ let T'_i be the pushout of l'_i and $t_i r'_i$. We obtain the solid part of the diagram above right, all morphisms are mono since monos are stable under pushouts in adhesive categories. Finally, T is constructed by taking repeated pushouts in the obvious way. Clearly, $G \to T$ is mono, and since each of the morphisms $T_i \to T$ are mono, the canonical morphisms from the productions to T obtained by postcomposition with $T_i \to T$ are mono as well. Finally, it is clear that the resulting object is the colimit of the original diagram (and of the entire derivation diagram) because colimits commute. □

The STS associated with a derivation tree has the colimit of the derivation diagram as type object, and one production for each direct derivation in the tree.

Definition 29 (Derived STS) Let $\mathcal{G}$ be an adhesive grammar. Let $S \in \mathbf{C}$ be arbitrary. Recall that the objects of $S/\mathbf{DerTree}(\mathcal{G})$ are finite derivation trees from S. Let α be such a derivation tree. The derived STS is $Prc(\alpha) = \langle T, P, \pi \rangle$ where

- $T = Colim(Der(\alpha))$;
- $P = \sum_{\varphi \in \alpha} \mathrm{ar}(\varphi)$—note that we fix the coproduct in **Set** and order the fans φ so that $\langle i, j \rangle \in P$ is the ith production of the jth fan. The ordering is immaterial;
- $\pi(i, j) = \pi_{\mathcal{G}}(\varphi_j(i))$;
- The typing for the productions is canonically:



Using the conclusion of Theorem 28, $Prc(\alpha)$ is an STS.

6 Analysing Derivations in the STS

The goal of this section is to show that the construction of the derived STS for a finite derivation tree in an adhesive grammar gives us a setting where the local reasoning about the independence of productions using subobject inclusion and intersection, developed in Section 4, is fully abstract with respect to corresponding relations among direct derivations in the original tree.

We first show how derivations of a given diagram $Der(\alpha)$ are related to derivations in the derived STS $Prc(\alpha)$. In particular, we will show that all productions in $Prc(\alpha)$ are pure, that, as expected, every derivation of $Der(\alpha)$ (a linear path in the derivation tree) is also a derivation of $Prc(\alpha)$, and that there are no backward conflicts in $Prc(\alpha)$. This is shown in the following proposition.

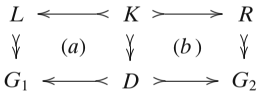
Proposition 30 (Properties of the derived STS) *Let $\mathcal{G}$ be an adhesive grammar, let α be a derivation tree in $\mathcal{G}$ with root S ($\alpha \in S/\mathbf{DerTree}(\mathcal{G})$), and let $Prc(\alpha)$ be its derived STS. Then:*

(1) *$Prc(\alpha)$ is pure.*
(2) *For each direct derivation $G_1 \overset{q,m}{\Longrightarrow} G_2$ in α, let q' be the corresponding production in $Prc(\alpha)$. Then $G_1 \Rightarrow^{q'} G_2$.*
(3) *Let $G_0 \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m_2}{\Longrightarrow} \cdots G_{n-1} \overset{q_n,m_n}{\Longrightarrow} G_n$ be a derivation in α with $n \geq 2$, and let $q'_1, q'_2, \ldots, q'_n$ be the corresponding productions in $Prc(\alpha)$. Then $q'_n \not\prec_{wc} q'_1$.*
(4) *Let q'_1 and q'_2 be two distinct productions of $Prc(\alpha)$. Then $\neg(q'_1 \curlyvee q'_2)$.*

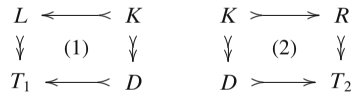
Proof The proofs of the four statements follow a common pattern, based on the following observations: let Δ be the "bottom part" of the diagram of derivations $Der(\alpha)$; i.e., Δ is obtained by deleting from $Der(\alpha)$ all of the double-pushout

diagrams, leaving only their lower spans. Clearly, a colimit for Δ is also a colimit for $Der(\alpha)$. Now notice that Δ is simply connected, and it consists of objects which are either the source of exactly two arrows (the "D"s), or are the target of a finite number of arrows (the "G"s). Taking out of Δ any single "D" we obtain two simply connected diagrams enjoying the same properties of Δ.

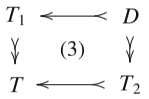
Now consider any double-pushout diagram within $Der(\alpha)$, illustrated below: Δ contains only its lower span.



By deleting from Δ object D and the outgoing arrows, we obtain Δ_1 and Δ_2: let T_1 and T_2 be the corresponding colimits. By Theorem 28 there are monos $G_1 \to T_1$ and $G_2 \to T_2$. Since they are mono and squares (a) and (b) are pullbacks, the following diagrams are pullbacks as well:

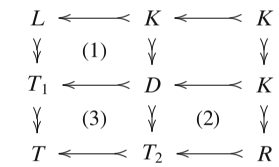


Furthermore, the colimit T of the original diagram Δ is obtained by the following pushout, which is also a pullback as we are in an adhesive category:



We can proceed now with the proof of the four statements.

(1) Let q' be a production of $Prc(\alpha)$ with $\pi(q') = \langle L, K, R \rangle$, and let $G_1 \overset{q,m}{\Longrightarrow} G_2$ be the corresponding direct derivation in $Der(\alpha)$. By the observations above, we can build the diagram below.



The upper right square is a pullback since $K \to D$ is mono. Since all interior squares are pullbacks, so is the outer one. Hence $K \cong L \cap R$ and thus the production q' is pure.

(2) By Proposition 6, we have to show that there is no contact, i.e., that $G_1 \cap R \subseteq L$. Consider the diagram below: the upper left square is a pullback since the diagram commutes and $G_1 \to T_1$ is mono. The upper right square is trivially a pullback.

$$\begin{array}{ccccc} G_1 & \leftarrowtail & D & \leftarrowtail & K \\ \downarrow & & \downarrow & & \downarrow \\ T_1 & \leftarrowtail & D & \leftarrowtail & K \\ \downarrow & (3) & \downarrow & (2) & \downarrow \\ T & \leftarrowtail & T_2 & \leftarrowtail & R \end{array}$$

All the interior squares are pullbacks, thus $G_1 \cap R \cong K \subseteq L$, as desired.

(3) We show that, given the hypotheses, $L_1 \cap R_n \cong K_1 \cap K_n \subseteq K_1 \cup K_n$, and thus $q'_n \not\asymp_{wc} q'_1$.
By the given derivation, in diagram Δ (the "bottom part" of $Der(\alpha)$), we have the following chain of spans:

$$G_0 \leftarrow D_1 \rightarrow G_1 \leftarrow D_2 \rightarrow \cdots G_{n-1} \leftarrow D_n \rightarrow G_n$$

By deleting D_1 and D_n we get three separated simply connected diagrams, with colimits T_1, T' and T_n. In the following diagram, the colimit T of Δ is computed via the lower left pushout, which is also a pullback because of adhesivity. The four upper right squares are all easily seen to be pullbacks, since $K_1 \to D_1$, $K_n \to D_n$ and $T' \to T$ are monos. Thus all the interior squares are pullbacks, meaning that the entire square is a pullback and $L_1 \cap R_n \cong K_1 \cap K_n$ as desired.

$$\begin{array}{ccccccc} R_n & \leftarrowtail & K_n & \leftarrowtail & K_n \cap D_1 & \leftarrowtail & K_n \cap K_1 \\ \downarrow & (2) & \downarrow & & \downarrow & & \downarrow \\ T_n & \leftarrowtail & D_n & \leftarrowtail & D_n \cap D_1 & \leftarrowtail & D_n \cap K_1 \\ \downarrow & (3) & \downarrow & & \downarrow & & \downarrow \\ T'_n & \leftarrowtail & T' & \leftarrowtail & D_1 & \leftarrowtail & K_1 \\ \downarrow & & \downarrow & (3) & \downarrow & (1) & \downarrow \\ T & \leftarrowtail & T'_1 & \leftarrowtail & T_1 & \leftarrowtail & L_1 \end{array}$$

(4) There are two cases to consider: *(a)* The two direct derivations corresponding to q'_1 and q'_2 belong to different branches of $Der(\alpha)$, and *(b)* they belong to the same linear derivation.
For the first case, since α is a tree we know that in $Der(\alpha)$ there are two minimal derivations

$$G_0 \overset{q_1,m_1}{\Longrightarrow} G_1 \cdots G_{n-1} \overset{q_n,m_n}{\Longrightarrow} G_n \text{ and } G_0 \overset{p_1,m'_1}{\Longrightarrow} G'_1 \cdots G'_{k-1} \overset{p_k,m'_k}{\Longrightarrow} G'_k$$

such that q'_1 corresponds to $G_{n-1} \overset{q_n,m_n}{\Longrightarrow} G_n$, and q'_2 to $G'_{k-1} \overset{p_k,m'_k}{\Longrightarrow} G'_k$, respectively. In diagram Δ we have the following chain of spans:

$$G'_k \leftarrow D'_k \rightarrow G'_{k-1} \cdots G'_1 \leftarrow D'_1 \rightarrow G_0 \leftarrow D_1 \rightarrow G_1 \cdots G_{n-1} \leftarrow D_n \rightarrow G_n$$

By deleting D'_k and D_n we get three separated simply connected diagrams, with colimits T_k, T' and T_n, respectively. The following diagram, where all interior squares are pullbacks, allows us to conclude that $R_{q'_1} \cap R_{q'_2} \cong K_{q'_1} \cap K_{q'_2}$, thus $R_{q'_1} \cap R_{q'_2} \subseteq K_{q'_1} \cup K_{q'_2}$, which means $\neg(q'_1 \curlyvee q'_2)$.

$$\begin{array}{ccccccc} R_{q'_1} = R_n & \leftarrowtail & K_n & \leftarrowtail & K_n \cap D'_k & \leftarrowtail & K_n \cap K'_k = K_{q'_1} \cap K_{q'_2} \\ \downarrow & (2) & \downarrow & & \downarrow & & \downarrow \\ T_n & \leftarrowtail & D_n & \leftarrowtail & D_n \cap D'_k & \leftarrowtail & D_n \cap K'_k \\ \downarrow & (3) & \downarrow & & \downarrow & & \downarrow \\ T'_n & \leftarrowtail & T' & \leftarrowtail & D'_k & \leftarrowtail & K'_k \\ \downarrow & & \downarrow & (3) & \downarrow & (2) & \downarrow \\ T & \leftarrowtail & T'_k & \leftarrowtail & T_k & \leftarrowtail & R'_k = R_{q'_2} \end{array}$$

For case *(b)* suppose, without loss of generality, that q'_1 and q'_2 are the productions of $Prc(\alpha)$ corresponding to the first and to the last direct derivations of the following derivation: $G_0 \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m_2}{\Longrightarrow} G_2 \cdots G_{n-1} \overset{q_n,m_n}{\Longrightarrow} G_n$: thus $R_{q'_1} = R_1$ and $R_{q'_2} = R_n$.
Now if $n = 2$, then we have $G_1 \cap R_2 \cong K_2$ by the proof of point (2), and since $R_1 \subseteq G_1$, we have $R_1 \cap R_2 \subseteq K_2 \subseteq K_1 \cup K_2$, thus $\neg(q'_1 \curlyvee q'_2)$.
If instead $n > 2$, consider the following chain of spans in Δ:

$$G_0 \leftarrow D_1 \rightarrow G_1 \leftarrow D_2 \rightarrow G_2 \cdots G_{n-1} \leftarrow D_n \rightarrow G_n$$

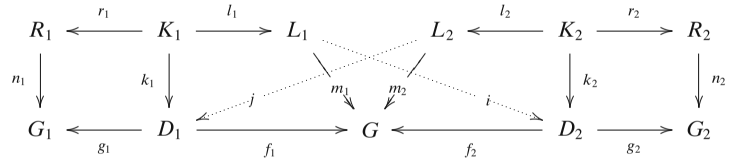
By deleting D_2 and D_n we get three separated simply connected diagrams, with colimits T_2, T' and T_n. Reasoning as in point (3) we build the following diagram, where all the interior squares are pullbacks:

$$\begin{array}{ccccccc} T_2 & \leftarrowtail & D_2 & \leftarrowtail & D_2 \cap D_n & \leftarrowtail & D_2 \cap K_n \\ \downarrow & (3) & \downarrow & & \downarrow & & \downarrow \\ T'_2 & \leftarrowtail & T' & \leftarrowtail & D_n & \leftarrowtail & K_n \\ \downarrow & & \downarrow & (3) & \downarrow & (2) & \downarrow \\ T & \leftarrowtail & T'_n & \leftarrowtail & T_n & \leftarrowtail & R_n \end{array}$$

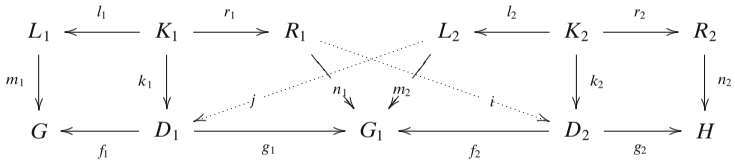
Therefore the outer square is a pullback, meaning that $T_2 \cap R_n \cong D_2 \cap K_n$. Now by Theorem 28, we know that R_1 maps injectively to the colimit T_2, and thus $R_1 \cap R_n \subseteq D_2 \cap K_n \subseteq K_n \subseteq K_1 \cup K_n$, as desired. □

In order to show that STSs can be used for reasoning about independence in derivation trees of adhesive grammars, we shall need to recall the standard notions of independence from the theory of DPO rewriting [8], namely sequential and parallel independence for graph transformation systems. Given the categorical nature of the definitions, the same definitions are used in the more general setting of transformation systems based on adhesive categories [16].

Definition 31 (Parallel and sequential independence) Let $\mathcal{G}$ be an adhesive graph grammar and q_1, q_2 be two of its productions: Two direct derivations $G \overset{q_1,m_1}{\Longrightarrow} G_1$ and $G \overset{q_2,m_2}{\Longrightarrow} G_2$ are *parallel independent* if there exist morphisms $i : L_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 \circ i = m_1$ and $f_1 \circ j = m_2$:



Furthermore, two direct derivations $G \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m_2}{\Longrightarrow} H$ are *sequential independent* if there exist morphisms $i : R_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 \circ i = n_1$ and $g_1 \circ j = m_2$:



The following theorem states that the construction of a derived STS for a finite derivation tree in an adhesive grammar $\mathcal{G}$, presented in Section 5, gives us a setting where the local reasoning about independence with subobjects, developed in Section 3, is fully abstract with respect to the independence in the original derivation.

Theorem 32 (Checking independence in the derived STS) *Suppose that $\mathcal{G}$ is an adhesive grammar. Let α be a derivation tree in $\mathcal{G}$ with root S ($\alpha \in S/\mathbf{DerTree}(\mathcal{G})$).*

1. *Let $G_1 \overset{q_1,m_1}{\Longrightarrow} G_2$, $G_1 \overset{q_2,m_2}{\Longrightarrow} G_3$ be two derivation steps in α, with $q_1 = \varphi_k(i)$, $q_2 = \varphi_k(j)$ for some fan φ_k in α, and let $q'_1 = \langle i, k\rangle$, $q'_2 = \langle j, k\rangle$ in $Prc(\alpha)$ be the corresponding productions in the derived STS. Then the following are equivalent:*
   (1) *$G_1 \overset{q_1,m_1}{\Longrightarrow} G_2$ and $G_1 \overset{q_2,m_2}{\Longrightarrow} G_3$ are parallel independent*
   (2) $q'_1 \lozenge q'_2$
   (3) $\neg(q'_1 \between q'_2) \wedge q'_1 \not<_d q'_2 \wedge q'_2 \not<_d q'_1$
2. *Let $G_1 \overset{q_1,m_1}{\Longrightarrow} G_2 \overset{q_2,m_2}{\Longrightarrow} G_3$ be two derivation steps in α, with $q_1 = \varphi_k(i)$, $q_2 = \varphi_l(j)$ for fans φ_k, φ_l in α and $q'_1 = \langle i, k\rangle$, $q'_2 = \langle j, l\rangle$ in $Prc(\alpha)$, then the following are equivalent:*
   (1) *$G_1 \overset{q_1,m_1}{\Longrightarrow} G_2 \overset{q_2,m_2}{\Longrightarrow} G_3$ are sequential independent*
   (2) $q'_1 \lozenge q'_2$
   (3) $q'_1 \not<_{rc} q'_2 \wedge q'_1 \not<_{wc} q'_2 \wedge q'_1 \not<_d q'_2$

Proof

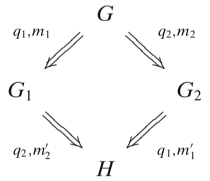
1. (1) $\Rightarrow$ (2) By parallel independence there are arrows $i : L_1 \to D_2$ and $j : L_2 \to D_1$ such that $f_2 \circ i = m_1$ and $f_1 \circ j = m_2$; i and j are mono because so are f_i and m_i for $i \in \{1, 2\}$. It is easy to show that i and j commute with the injections of their source and target objects in the colimit object T, thus $L_1 \subseteq D_2$ and $L_2 \subseteq D_1$ in $\mathbf{Sub}(T)$. Furthermore, by Proposition 30*(4)* it holds $\neg(q'_1 \between q'_2)$. Then we conclude by Lemma 20 (*(3) $\Rightarrow$ (1)*).
   (2) $\Rightarrow$ (1) by Lemma 20 ((1) $\Rightarrow$ (3)) we know that $L_1 \subseteq D_2$ and $L_2 \subseteq D_1$ in $\mathbf{Sub}(T)$. Thus there are monos $i : L_1 \to D_2$ and $j : L_2 \to D_1$; since $\mathbf{Sub}(T)$ is a preorder, i and j form commuting triangles with the cospans of injections $D_2 \to G_1 \leftarrow L_1$ and $D_1 \to G_1 \leftarrow L_2$, respectively. Thus the two direct derivations in $\mathcal{G}$ are parallel independent.
   (2) $\Leftrightarrow$ (3) Obvious by Lemma 20 ((1) $\Leftrightarrow$ (2)), observing that $\neg(q'_1 \between q'_2)$ by Proposition 30*(4)*.
2. The proof is analogous to that of the previous point, by exploiting Lemma 21 instead of Lemma 20, and point (3) of Proposition 30 instead of point (4). □

The formal framework we introduced allows us to provide a new proof of the well-known local Church–Rosser theorem for transformations with injective matches, by exploiting the corresponding theorem for STSs.

Theorem 33 (Local Church–Rosser) *Let $\mathcal{G}$ be an adhesive grammar.*

(1) *If $G \overset{q_1,m_1}{\Longrightarrow} G_1$ and $G \overset{q_2,m_2}{\Longrightarrow} G_2$ are two parallel independent direct derivations, then there are an object H and direct derivations $G_1 \overset{q_2,m'_2}{\Longrightarrow} H$ and $G_2 \overset{q_1,m'_1}{\Longrightarrow} H$ such that $G \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m'_2}{\Longrightarrow} H$ and $G \overset{q_2,m_2}{\Longrightarrow} G_2 \overset{q_1,m'_1}{\Longrightarrow} H$ are sequential independent.*

(2) *If $G \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m'_2}{\Longrightarrow} H$ are two sequential independent direct derivations, then there are an object G_2 and direct derivations $G \overset{q_2,m_2}{\Longrightarrow} G_2 \overset{q_1,m'_1}{\Longrightarrow} H$ such that $G \overset{q_1,m_1}{\Longrightarrow} G_1$ and $G \overset{q_2,m_2}{\Longrightarrow} G_2$ are parallel independent:*



Proof We prove only point (1), because point (2) is completely analogous. Given $G \overset{q_1,m_1}{\Longrightarrow} G_1$ and $G \overset{q_2,m_2}{\Longrightarrow} G_2$, consider the derivation tree α in $G/\mathbf{DerTree}(\mathcal{G})$ based on this span of direct derivations. The derived STS of α, $Prc(\alpha)$, contains by construction only two productions, say q'_1 and q'_2, corresponding to the two direct derivations, with $G \Rightarrow^{q'_1} G_1$ and $G \Rightarrow^{q'_2} G_2$.

Since the original direct derivations are parallel independent by hypothesis, by Theorem 32.1 we have that $q'_1 \lozenge q'_2$, and by Theorem 22 there is a subobject H of

T, the colimit of $Der(\alpha)$, and direct derivations $G_1 \Rightarrow^{q'_2} H$ and $G_2 \Rightarrow^{q'_1} H$; from the proof of the theorem we also know that the contexts of these two direct derivations are $D'_2 = (D_1 \cap D_2) \cup R_1$ and $D'_1 = (D_1 \cap D_2) \cup R_2$. Therefore by Proposition 6 there are two double-pushout diagrams witnessing, up to isomorphisms, that $G_1 \overset{q_2,m'_2}{\Longrightarrow} H$ and $G_2 \overset{q_1,m'_1}{\Longrightarrow} H$, for matches m'_2 and m'_1 determined by the injections $L_2 \subseteq D_1 \subseteq G_1$ and $L_1 \subseteq D_2 \subseteq G_2$, which hold by hypothesis. Finally, notice that there are injections $R_1 \subseteq (D_1 \cap D_2) \cup R_1 = D'_2$ and $L_2 \subseteq D_1$, which make the resulting triangles commute because **Sub**(T) is a preorder. Thus $G \overset{q_1,m_1}{\Longrightarrow} G_1 \overset{q_2,m'_2}{\Longrightarrow} H$ is sequential independent, and so is $G \overset{q_2,m_2}{\Longrightarrow} G_2 \overset{q_1,m'_1}{\Longrightarrow} H$ by similar reasoning. □

7 Conclusions and Future Work

In this paper we have introduced subobject transformation systems (STS), a novel formal framework for the analysis of derivations of DPO transformation systems. They can be considered as a "distilled" variant of DPO rewriting, acting in the distributive lattice of subobjects of a given object of an adhesive category. In this setting the analysis of several conflict, causality, and independence relations among productions can be carried on using a set-theoretical syntax or simple geometric reasoning based on Venn diagrams, thus providing an alternative to the usual "diagram chasing" used in the algebraic approaches to rewriting. In particular, since every production in an STS has a unique match, in order to analyze how two different productions relate to each other, it is enough to look at the productions themselves.

We have presented several characterizations of independence of productions in *pure* STSs, as well as a local Church–Rosser theorem for them, also showing how the proof of the local Church–Rosser theorem for DPO transformation (with monic matches) in an adhesive category can be reduced to it. The characterisation can be considered complete, as we have analyzed all the possible ways in which causal dependency can arise between two productions. In particular, we have given a minimal set of basic relations and shown that relations which have been previously considered can be built up of the basic set.

As mentioned in the introduction, STSs over category **Set** with a few additional constraints are in a one-to-one correspondence with a particular class of Petri nets, called *Elementary Net Systems (ENS)* [22].[3] The formalization of the precise relationship between STSs and ENSs goes beyond the goal of the present paper, and will be a topic of future research. Nevertheless, let us stress the methodological value of this relationship: in the same way the theory of Place/Transition nets has been a constant source of inspiration during the last years for researchers working on both theoretical and more practical aspects of graph transformation systems (as witnessed for example by the various concurrent semantics proposed for GTSs, and by their application to the verification of such systems), we expect that also the theory of ENSs will provide challenging intuitions that could be generalized, at least in part, to the more abstract setting of STSs.

[3] Indeed, some terms we introduced are borrowed from the ENS terminology, like *pure* and *contact situation*.

In the paper we mainly considered *pure* systems, because so are the STSs arising as representation of the computations of DPO systems. We showed in Section 3 that for non-pure systems the set of basic relations would be larger. However, the theoretical or practical relevance of such systems is not clear, because often a self-loop, modelling the fact that a resource can be consumed and produced again, can be conveniently replaced with a read access to that resource. This analysis is left as a topic of future research, together with the study of some natural generalizations of the approach presented in this paper, including for example the handling of other algebraic approaches to rewriting, like the single-pushout and the sesqui-pushout approaches [6, 11].

In Section 5 we presented a construction that, given a *finite* derivation tree of a DPO system, builds an STS, a sort of *non-deterministic process*, which can be used to analyze the dependencies among production occurrences in the given derivation tree. On the other hand, the classical *unfolding construction* defined for Petri nets and GTSs in [24] and [4, 20], respectively, builds a specific, usually *infinite*, non-deterministic process, that represents all the derivations of the original system and enjoys an interesting universal property. We plan to capture the unfolding construction within our framework: To this aim, we intend to generalize the unfolding construction to an arbitrary adhesive grammar, possibly requiring some further properties on the underlying adhesive category.

In practice, often the grammars which are designed to model a given system are equipped with application conditions, as defined for example in [10, 13]. These conditions allow restricting the application of rules and hence, they model restricted control structures. Some preliminary results show that positive and negative application conditions can be handled by extra relations in an STS: they constitute a first step towards the generalization of the theory of STSs to this richer class of systems.

Occurrence grammars and Petri nets are already similar representations of a process, as they share the intuition of a causal relation and items, which can be produced and consumed. Petri nets offer a well founded theory for analysis and hence a transformation of an STS to an equivalent Petri net is an interesting challenge. Transformations for grammars without application conditions were already defined, e.g., in [1]. The integration of restricted negative application conditions were handled by in [5], but an integration of general application conditions as they are used in most practical examples would be of much more value. And indeed, the given definition of an STS and its relations combined with the mentioned extension for application conditions seems to be adequate to create an equivalent Petri net.

Acknowledgements We acknowledge Paolo Baldan and the anonymous referees for constructive comments on a preliminiary version of the paper.

References

1. Baldan, P.: Modelling concurrent computations: from contextual Petri nets to graph grammars. PhD dissertation, Department of Computer Science, University of Pisa, March. Available as technical report no. TD-1/00 (2000)
2. Baldan, P., Corradini, A., Heindel, T., König, B., Sobociński, P.: Processes for adhesive rewriting systems. In: Aceto, L., Ingólfsdóttir, A. (eds.) FoSSaCS, vol. 3921 of Lecture Notes in Computer Science, pp. 202–216. Springer Verlag (2006)

3. Baldan, P., Corradini, A., Montanari, U.: Concatenable graph processes: relating processes and derivation traces. In: Proc. of ICALP'98, vol. 1443 of Lecture Notes in Computer Science, pp. 283–295. Springer Verlag (1998)
4. Baldan, P., Corradini, A., Montanari, U.: Unfolding of double-pushout graph grammars is a coreflection. In: Ehrig, G., Engels, G., Kreowski, H.J., Rozenberg, G. (eds.) Proceedings of the International Workshop on Theory and Application of Graph Transformations, vol. 1764 of Lecture Notes in Computer Science, pp. 145–163. Springer Verlag (1999)
5. Baldan, P., König, B., Stürmer, I.: Generating test cases for code generators by unfolding graph transformation systems. In: Ehrig, H., Engels, G., Parisi-Presicce, F., Rozenberg, G. (eds.) ICGT'04, vol. 3256 of Lecture Notes in Computer Science, pp. 194–209. Springer Verlag (2004)
6. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In: Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G. (eds.) ICGT'06, vol. 4178 of Lecture Notes in Computer Science, pp. 30–45. Springer Verlag (2006)
7. Corradini, A., Montanari, U., Rossi, F.: Graph processes. Fund. Inform. **26**, 241–265 (1996)
8. 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: Rozenberg [21], Chapter 3 (1997)
9. Danos, V., Krivine, J., Sobociński, P.: General reversibility. In: Express '06, Electronic Notes in Theoretical Computer Science **175**(3), pp. 75–86. Elsevier (2007)
10. Ehrig, H., Ehrig, K., Prange, U., Taentzer, G.: Fundamentals of Algebraic Graph Transformation. EATCS Monographs in Theoretical Computer Science. Springer Verlag (2006)
11. Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic approaches to graph transformation II: single pushout approach and comparison with double pushout approach. In: Rozenberg [21], Chapter 4 (1997)
12. Golz, U., Reisig, W.: The non-sequential behaviour of Petri nets. Inf. Control **57**, 125–147 (1983)
13. Habel, A., Heckel, R., Taentzer, G.: Graph grammars with negative application conditions. Special issue of Fund. Inform. **26**(3,4), 287–313 (1996)
14. Joyal, A., Street, R.: The geometry of tensor calculus. I. Adv. Math. **88**, 55–112 (1991)
15. Kreowski, H.-J.: Manipulation von Graphmanipulationen. PhD thesis, Technische Universität Berlin (1977)
16. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. Theor. Inf. Appl. **39**(2), 511–546 (2005)
17. Leinster, T.: Higher Operads, Higher Categories. London Mathematical Lecture Notes. Cambridge University Press (2003)
18. Meseguer, J., Montanari, U.: Petri nets are monoids. Inform. and Comput. **88**, 105–155 (1990)
19. Reisig, W.: Petri Nets: An Introduction. EACTS Monographs on Theoretical Computer Science. Springer Verlag (1985)
20. Ribeiro, L.: Parallel Composition and Unfolding Semantics of Graph Grammars. PhD thesis, Technische Universität Berlin (1996)
21. Rozenberg, G. (ed.) Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations. World Scientific (1997)
22. Rozenberg, G., Engelfriet, J.: Elementary net systems. In: Reisig, W., Rozenberg, G. (eds.) Lectures on Petri Nets I: Basic Models, vol. 1491 of Lecture Notes in Computer Science, pp. 12–121. Springer Verlag (1996)
23. Street, R.: Higher categories, strings, cubes and simplex equations. Appl. Categ. Structures. **3**, 29–77 (1995)
24. Winskel, G.: Event structures. In: Petri Nets: Applications and Relationships to Other Models of Concurrency, vol. 255 of Lecture Notes in Computer Science, pp. 325–392. Springer Verlag (1987)