Unfolding Semantics of Graph Transformation [1]

Paolo Baldan [a,*], Andrea Corradini [b], Ugo Montanari [b], Leila Ribeiro [c]

[a] *Dipartimento di Matematica Pura e Applicata, Università di Padova, Italia*

[b] *Dipartimento di Informatica, Università di Pisa, Italia*

[c] *Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil*

Abstract

Several attempts have been made of extending to graph grammars the unfolding semantics originally developed by Winskel for (safe) Petri nets, but only partial results were obtained. In this paper we fully extend Winskel's approach to single-pushout grammars providing them with a categorical concurrent semantics expressed as a coreflection between the category of (semi-weighted) graph grammars and the category of prime algebraic domains, which factorises through the category of occurrence grammars and the category of asymmetric event structures. For general, possibly non semi-weighted single-pushout grammars, we define an analogous functorial concurrent semantics, which, however, is not characterised as an adjunction. Similar results can be obtained for double-pushout graph grammars, under the assumptions that nodes are never deleted.

Key words: Graph transformation systems, unfolding, concurrency, non determinism, event structures, Petri nets, read arcs
2000 MSC: 68Q42, 68Q55

* Corresponding author.
Email addresses: baldan@math.unipd.it (Paolo Baldan), andrea@di.unipi.it (Andrea Corradini), ugo@di.unipi.it (Ugo Montanari), leila@inf.ufrgs.br (Leila Ribeiro).

[1] Research partially supported by the CNPq-CNR Project IQ-Mobile II, by the EU FET-GC Project IST-2001-32747 agile, and by the EC RTN 2-2001-00346 SegraVis, MIUR project PRIN 2005015824 ART, CRUI/DAAD Vigoni "Models based on Graph Transformation Systems: Analysis and Verification".

1 Introduction

The *theory of graph grammars* (or of *graph rewriting systems*) studies a variety of formalisms which extend the theory of formal languages in order to deal with structures more general than strings, like graphs and maps. A graph grammar allows one to describe finitely a (possibly infinite) collection of graphs, i.e., those graphs which can be obtained from a start graph through repeated applications of graph productions. Each production can be applied to a graph by replacing an occurrence of its left-hand side with its right-hand side. The form of graph productions and the mechanisms stating how a production can be applied to a graph and what the resulting graph is, depend on the specific graph rewriting formalism. The handbook [37] presents a comprehensive introduction to the theory of several approaches to graph rewriting.

Since many (natural or artificial) distributed structures can be represented (at a suitable level of abstraction) by graphs, and graph productions act on those graphs with local transformations, it is quite obvious that graph rewriting systems are potentially interesting for the study of the concurrent transformation of structures. In particular, it belongs to the folklore (see [13] and the references therein) that Petri nets can be regarded as graph rewriting systems that act on a restricted kind of graphs, namely discrete, labelled graphs (that can be considered as sets of tokens labelled by places). Conversely, graph rewriting systems generalise Petri nets not only because they allow for arbitrary (also non-discrete) graphs, but also because they allow for the specification of context-dependent operations, where part of the state is read but not consumed. Their greater expressiveness is also witnessed by the fact that, differently from ordinary Place/Transition Petri nets, most approaches to graph rewriting systems (including those considered in this paper) are Turing complete.

In recent years, various concurrent semantics for graph rewriting systems have been proposed in the literature, some based on the above mentioned correspondence with Petri nets, trying to generalise to graph grammars classical models originally developed for nets, like Goltz-Reisig process semantics [20] and Winskel's unfolding semantics [39]. This work was mainly concerned with the so-called *algebraic approaches* to graph rewriting, where graphs are viewed as objects of a category and the notion of rewriting is defined in terms of suitable diagrams in that category. Such approaches include the *double-pushout* (DPO) approach [19,16], where the application of a rule is modelled by *two* pushout diagrams in a category of graphs and *total* graph morphisms, and the *single-pushout* (SPO) approach [27,18], where a rule application is described instead as a *single* pushout in a category of graphs and *partial* morphisms.

Recall that, building on [33], the seminal work [39] proposes a concurrent se-

mantics of safe Petri nets by means of a chain of coreflections leading from the category of safe nets to the category of prime algebraic domains. Given functors F and G, we write $F \vdash G$ when F is right adjoint to G. The same symbol is used, possibly rotated, in diagrams. The symbol $\hookrightarrow$ indicates inclusion functors, while $\sim$ is put between functors establishing equivalences of categories.

$$\textbf{Safe Nets} \underset{\mathcal{U}}{\overset{\perp}{\leftrightarrows}} \textbf{Occurrence Nets} \underset{\mathcal{E}}{\overset{\mathcal{N}}{\underset{\perp}{\leftrightarrows}}} \textbf{Prime Event Structures} \underset{\mathcal{L}}{\overset{\mathcal{P}}{\underset{\sim}{\leftrightarrows}}} \textbf{Domains}$$

The first step unfolds any (safe) net into an occurrence net, i.e., a branching acyclic net making explicit causality and conflict (nondeterministic choice points) between events in the net. The second step produces a *prime event structure* (PES) abstracting away the state and recording only the events and the relationships between events. Finally, the last step maps any PES into the corresponding prime algebraic domain of configurations. In [31] it is shown that an analogous construction works for the wider class of *semi-weighted* nets, i.e., P/T nets in which the initial marking is a set and transitions can generate at most one token in each place of their post-set.

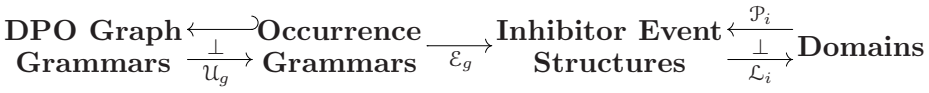
The above semantical framework has been generalised in [8] to the category of *semi-weighted contextual nets*. A contextual net is a Petri net where each transition, besides of a pre-set and a post-set which specify the tokens which are consumed and produced by the transition, may also have a *context*, i.e., places which must contain a token when the transition is fired. Tokens in the context may be considered to be accessed in a read-only way by the transition in that they are needed to enable the firing, but they are not consumed. The chain of coreflections proposed in [8] is the following:

$$\textbf{Semi-Weighted Contextual Nets} \underset{\mathcal{U}_a}{\overset{\perp}{\leftrightarrows}} \textbf{Occurrence Contextual Nets} \underset{\mathcal{E}_a}{\overset{\mathcal{N}_a}{\underset{\perp}{\leftrightarrows}}} \textbf{Asymmetric Event Structures} \underset{\mathcal{L}_a}{\overset{\mathcal{P}_a}{\underset{\perp}{\leftrightarrows}}} \textbf{Domains}$$

The first step is analogous to the one for ordinary Petri nets. In the next step, prime event structures are replaced by *asymmetric event structures* (AES's). This happens because the presence of contexts introduces a kind of dependency between transitions, called *asymmetric conflict*, which cannot be described adequately by prime event structures. Asymmetric event structures form a proper coreflection with the category of domain, as depicted by the last adjunction.

Coming back to graph grammars, some important steps have been taken in the direction of developing an analogous semantical framework for algebraic graph transformation systems, but this program could not be considered as completed yet. More precisely, several constructions have been defined for

algebraic DPO graph grammars by the first three authors (see [1,5,7,2]), as summarised by the following diagram:

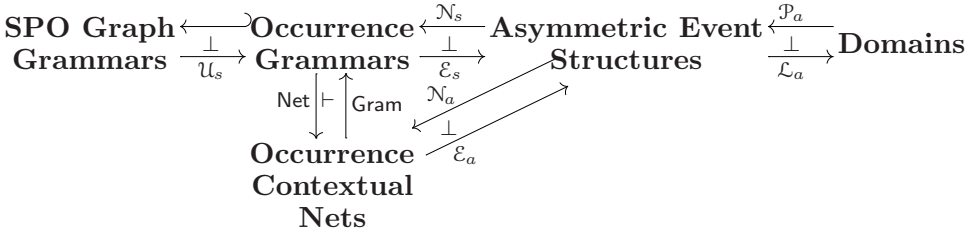


Even if at this level of abstraction it is not possible to see the relevant differences in the technical treatment of DPO grammars w.r.t. the much simpler case of Petri nets, still it is worth pointing at the evident differences between this chain of functors and the corresponding ones for nets.

Firstly, the categories of PES's and AES's are replaced by that of *inhibitor event structures* (IES's), an even more general class of event structures introduced in [2]. This is due to the fact that in DPO graph grammars, the possibility of applying a rule to a graph is subject to an application condition, the so-called *dangling condition*. By assuming a sort of conditional or-causality as a basic relation among events, IES's are able to model not only the asymmetric conflicts between events arising from the capability of preserving part of the state, but also the inhibiting effects related to the presence of the application condition for rules. The category of domains can be viewed as a coreflective subcategory of IES's (as shown by the last step of the chain) and thus one can also recover a semantics for DPO grammars in terms of domains and PES's.

Secondly, the functor from the category of occurrence grammars to the category of IES's *does not admit* a left adjoint establishing a coreflection between IES's and occurrence grammars, and thus the whole semantic transformation from DPO grammars to domains cannot be expressed as a coreflection.

In this paper we concentrate on the SPO approach to graph transformation. One of the main differences with respect to the DPO approach lies in the fact that there are no conditions on rule application, i.e., whenever a match of the left-hand side is found in a graph, the corresponding rule can always be applied. Building on the results briefly summarised above, we develop a coreflective unfolding semantics for semi-weighted SPO graph grammars defined through the following chain of coreflections:



To our knowledge, this is the first time that Winskel's chain of coreflections is

generalised to a category of graph grammars in a completely satisfactory way.

- The first coreflection in the above chain recasts in the framework of SPO grammars the unfolding construction of DPO grammars presented in [7], and it is conceptually close to the unfolding construction proposed in [35].
- Then, even if occurrence SPO grammars and occurrence contextual nets exhibit significant differences in their behaviour (the application of a rule of an occurrence grammar deletes, as a kind of side-effect, all the edges which would remain dangling due to the remotion of some nodes), we can formalise the relationship between the two classes of models as a categorical adjunction. A functor Net which maps any occurrence grammar to a contextual occurrence net (by forgetting the graphical structure of the state) is shown to admit a left-adjoint Gram (which freely generates the graphical structure).
- The above adjunction, composed with the coreflection between occurrence contextual nets and AES's already proved in [8], provides the step that was missing for DPO grammars, namely the left adjoint functor establishing a coreflection between the category of occurrence graph grammars and the category of asymmetric event structures.

  Note that, in particular, inhibitor event structures are not needed: due to the absence of application conditions for rules, the dependencies between events in SPO grammar computations can be expressed as causalities and asymmetric conflicts, and thus can be faithfully represented by using AES's.

As discussed in a concluding section, some of the results in this paper can be extended to larger categories of graph grammars. For example, provided that we stick to a smaller class of grammar morphisms, we can obtain a functorial concurrent semantics for general, possibly non-semi-weighted SPO graph grammars. A chain of functors maps any SPO grammar first to its unfolding, then to an AES and to a domain. In this setting we can still characterise the unfolding construction as a coreflection, but the coreflection to event structures and domains is lost.

As far as the DPO approach is concerned, we can obtain results analogous to those for SPO grammars if we consider the subclass of DPO graph grammars where productions never delete nodes. This class of grammars is still interesting from a modelling point of view since, as observed in [3,4], one can develop a theory of rewriting "up to isolated nodes" and in this setting the deletion of a node can be simulated faithfully by leaving such a node isolated. However, as discussed above, these results do not extend to general DPO graph grammars because of the presence of the dangling application condition for rules.

The rest of the paper is structured as follows. In Section 2 we review the basics of single-pushout graph grammars and we informally discuss their relationship with contextual nets. In Section 3 we define the notion of graph grammar mor-

phism we shall work with. In Section 4 we discuss the kind of dependencies arising between events in SPO graph grammars and we introduce the notion of occurrence graph grammar. The subcategory of occurrence graph grammars is given an alternative, more manageable characterisation, which allows also to get a tight link, formalised as an adjunction, with the category of occurrence contextual nets. In Section 6 we introduce the notion of a (nondeterministic) process for SPO graph grammars. Then, in Section 7, we present the unfolding construction for SPO graph grammars, which generates a process for the given grammar fully describing its behaviour. The unfolding is characterised, categorically, as a universal construction. In Section 8 we use the adjunction of Section 4 and some existing results for contextual nets in order complete the chain of coreflections from grammars to domains. In Section 9 we discuss some possible generalisations of our results and the relationships with the work on the DPO approach. Finally, some concluding remarks can be found in Section 10.

This paper elaborates on and extends the results on the concurrent semantics of algebraic graph grammars reported in the conference papers [7,6,9]. It also uses in an essential way the work on contextual nets in [8], for which the possible applications to graph grammars was indeed one of the main motivations.

2 Typed graph grammars

In this section we summarise the basics of graph grammars in the *single-pushout* (SPO) approach [27], an algebraic approach to graph rewriting alternative to the classical *double-pushout* (DPO) approach. Here we consider basic graph grammars, without any distinction between terminal and non-terminal symbols and without any high-level control mechanism. We remark that, even in this basic formulation, algebraic graph grammars are Turing complete (since they can simulate string rewriting).

The original SPO approach is adapted to deal with *typed graphs* [15,28], which are graphs labelled over a structure that is itself a graph, called the *graph of types*. Then some insights are provided on the relationship between typed graph grammars and Petri nets. Finally we introduce semi-weighted graph grammars, the class of grammars we shall work with in the paper.

2.1 Typed graph grammars

Given a partial function $f : A \rightarrowtail B$ we will denote by $dom(f)$ its *domain*, i.e., the set $\{a \in A \mid f(a) \text{ is defined}\}$. Furthermore we will write $f^{\leftarrow} : dom(f) \to A$

for the (total) inclusion of the domain of f into A and $f^{\rightarrow} : dom(f) \to B$ for the (total) restriction of f to $dom(f)$. Let $f, g : A \rightarrowtail B$ be two partial functions. We will write $f \leq g$ when $dom(f) \subseteq dom(g)$ and $f(x) = g(x)$ for all $x \in dom(f)$.

Definition 1 (graphs, partial graph morphism) *A (directed, unlabelled)* graph *is a tuple $G = \langle N_G, E_G, \mathsf{s}_G, \mathsf{t}_G \rangle$, where N_G and E_G are the sets of* nodes *and* edges *of G, and $\mathsf{s}_G, \mathsf{t}_G : E_G \to N_G$ are its* source *and* target *functions. Graph G is* discrete *if E_G is empty.*

A partial graph morphism *$f : G \rightarrowtail H$ is a pair of partial functions $f = \langle f_N : N_G \rightarrowtail N_H, f_E : E_G \rightarrowtail E_H \rangle$ such that (see Fig. 1.(a)):*

$$\mathsf{s}_H \circ f_E \leq f_N \circ \mathsf{s}_G \quad \text{and} \quad \mathsf{t}_H \circ f_E \leq f_N \circ \mathsf{t}_G. \ (*)$$

We denote by **PGraph** *the category of (directed, unlabelled) graphs and partial graph morphisms. A morphism is called* total *if both components are total, and the corresponding subcategory of* **PGraph** *is denoted by* **Graph**.

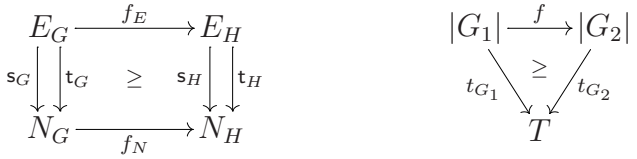
Notice that, according to condition (*), if f is defined over an edge then it must be defined both on its source and target nodes: this ensures that the domain of f is a well-formed graph. The inequalities in condition (*) ensure that *any* subgraph of a graph G can be the domain of a partial morphism $f : G \rightarrowtail H$. Instead, the stronger (apparently natural) conditions $\mathsf{s}_H \circ f_E = f_N \circ \mathsf{s}_G$ and $\mathsf{t}_H \circ f_E = f_N \circ \mathsf{t}_G$ would have imposed f to be defined over an edge whenever it is defined either on its source or on its target node.

Given a graph G we will sometimes write $x \in G$ to say that x is a node or edge in G, i.e., $x \in N_G \cup E_G$.

Definition 2 (typed graph) *Given a graph T, a* typed graph *G over T is a graph $|G|$, together with a total morphism $t_G : |G| \to T$. A* partial morphism *between T-typed graphs $f : G_1 \rightarrowtail G_2$ is a partial graph morphism $f : |G_1| \rightarrowtail |G_2|$ consistent with the typing, i.e., such that $t_{G_1} \geq t_{G_2} \circ f$ (see Fig. 1.(b)). A typed graph G is called* injective *if the typing morphism t_G is injective. The category of T-typed graphs and partial typed graph morphisms is denoted by* T-**PGraph**.

Given a partial typed graph morphism $f : G_1 \rightarrowtail G_2$, we denote by $dom(f)$ the domain of f typed in the obvious way. Also the notation $f^{\leftarrow}$ and $f^{\rightarrow}$ is extended to partial (typed) graph morphisms.

Definition 3 (graph production and direct derivation) *Fixing a graph T of types, a* (T-typed graph) production *q is an injective partial typed graph morphism $L_q \overset{r_q}{\rightarrowtail} R_q$. It is called* consuming *if the morphism is not total. The typed graphs L_q and R_q are called the* left-hand side *and the* right-hand side



(a) (b)

Fig. 1. Diagrams for partial graph and typed graph morphisms.

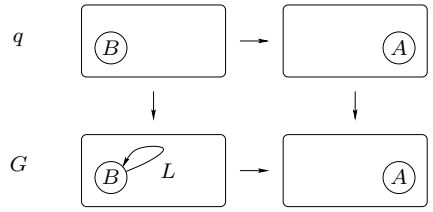
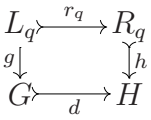


Fig. 2. Side-effects in SPO rewriting.

of the production, respectively.

Given a typed graph G and a match*, i.e., a total injective morphism $g : L_q \to G$, we say that there is a* direct derivation δ from G to H using q (based on g)*, written $\delta : G \Rightarrow_q H$, if the following is a pushout square in T-***PGraph***.*



Roughly speaking, the rewriting step removes from the graph G the image of the items of the left-hand side which are not in the domain of r_q, namely $g(L_q - dom(r_q))$, adding the items of the right-hand side which are not in the image of r_q, namely $R_q - r_q(dom(r_q))$. The items in the image of $dom(r_q)$ are "preserved" by the rewriting step (intuitively, they are accessed in a "read-only" manner).

A relevant difference with respect to the DPO approach is that here there is no *dangling condition* [16] preventing a rule to be applied whenever its application would leave dangling edges. In fact, as a consequence of the way pushouts are constructed in T-**PGraph**, when a node is deleted by the application of a rule also all the edges having such node as source or target are deleted by the rewriting step, as a kind of *side-effect*. For instance, consider production q in the top row of Fig. 2. Nodes are represented as circles, while edges are represented as directed arrows. The production consumes node B and produces node A, i.e., the associated graph morphism is the empty one. Then, production q can be applied to the graph G in the same figure. As a result both node B and the loop edge L are removed.

Notice that in the definition of direct derivation we consider *injective* matches only. As discussed later in Section 9.2 this does not affect the expressiveness of the formalism and it is technically convenient for the purposes of this paper.

Definition 4 (typed graph grammar and derivation) *A* (T-typed) SPO graph grammar $\mathcal{G}$ *is a tuple* $\langle T, G_s, P, \pi \rangle$*, where* G_s *is the* (typed) start graph*,* P *is a set of* production names*, and* π *is a function which associates to each name* $q \in P$ *a production* $\pi(q)$*. A graph grammar is* consuming *if all the productions in the range of* π *are consuming. A* derivation *in* $\mathcal{G}$ *is a sequence of direct derivations beginning from the start graph, i.e.,* $\rho = \{G_{i-1} \Rightarrow_{q_{i-1}} G_i\}_{i \in \{1,\ldots,n\}}$*, with* $G_0 = G_s$*.*

In the paper we will consider *consuming* graph grammars only. This restriction is essential to obtain a meaningful semantics combining concurrency and nondeterminism. In fact, the presence of non-consuming productions, which can be applied without deleting any item, would lead to an unbounded number of concurrent events with the same causal history. This would not fit with the approach to concurrency (see, e.g., [20,39]) where events in computations are identified with their causal history (formally, the unfolding construction of Section 7 would not work).

For a graph grammar $\mathcal{G}$ we denote by $Elem(\mathcal{G})$ the set $N_T \cup E_T \cup P$. As a convention, for each production name q the corresponding production $\pi(q)$ will be $L_q \overset{r_q}{\rightarrowtail} R_q$. Without loss of generality, we will assume that the injective partial morphism r_q is a partial inclusion (i.e., that $r_q(x) = x$ whenever defined). Moreover we assume that the domain of r_q, which is a subgraph of both $|L_q|$ and $|R_q|$ is the only intersection of these two graphs, i.e., that $|L_q| \cap |R_q| = dom(r_q)$, componentwise. Since in this paper we work only with typed notions, we will usually omit the qualification "typed", and, sometimes, we will not indicate explicitly the typing morphisms.

Example 5 *As an example let us consider the grammar* $\mathcal{SR}$ *in Fig. 3. The grammar is intended to represent a simple system where an unbounded number of processes can be created. Processes can then establish a connection through a communication manager in order to exchange a message.*

The items of the type graph represent the entities in the system and their possible relations. Node G *is the process generator, which can produce any number of processes, while node* M *is the* communication manager*, which manages the connection requests coming from processes. Any process can be in three states:* idle*, represented by node* P*,* sender*, represented by node* S *and* receiver*, represented by node* R*. The message to be sent is represented as a self-loop edge* ms *over the sender process, while a received message is represented by a self-loop edge* mr *over the receiver process. The communication requests are represented by edges* snd *and* rcv *which connect the sender and receiver*

processes to the communication manager. A sender and a receiver engaged in a communication are connected by a c-edge.

The typing functions for the productions and the start graph are represented by labelling any graph item with the corresponding item of the type graph. Functions from the left-hand side to the right-hand side of productions are partial inclusions represented by drawing the items in the domain as dashed circles/arrows.

Let us give a more detailed description of the productions. As mentioned above, an unbounded number of idle P-typed processes can be created by using production (Gen)*, which relies on the presence of a G-typed generator node.*

Any process P can connect to the communication manager, represented by a M-typed node, to send a message ms, as expressed by production (Send)*. The process changes its state, becoming an S-typed process and it connects to the communication manager via a snd edge.*

Production (Conn) *establishes a connection between a sender and a receiver, via a c-typed edge. Once a sender and a receiver are connected, the communication can take place as expressed by production* (Comm) *and the message is received, as represented by the mr-typed self-loop over the receiver.*

Finally, productions (EndS) *and* (EndR) *remove a sender or a receiver, respectively. Note that, by the* SPO *rewriting mechanism, when a sender or a receiver is removed, all edges having that node as source or target are removed as well. Conceptually, productions* (EndS) *and* (EndR) *can be applied after the communication has taken place, but also before a communication has been successfully completed, and in this case they model a communication failure.*

2.2 Relation with Petri nets

The reader who is familiar with Petri net theory can gain a solid intuition about grammar morphisms and many other definitions and constructions presented in this paper by referring to the relationship between Petri nets and (SPO) graph grammars. The correspondence between these two formalisms (see, e.g., [13] and references therein) relies on the basic observation that a P/T Petri net is essentially a rewriting system on a restricted kind of graphs, namely discrete, labelled graphs (that can be identified with sets of tokens labelled by places), with the net transitions playing the role of the productions. As a graph production can specify that part of the state must be present for the transformation to take place but is not consumed, an even tighter correspondence exists between graph grammars and *contextual nets* [32], also called nets with test arcs in [12], activator arcs in [22] or read arcs in [38], an

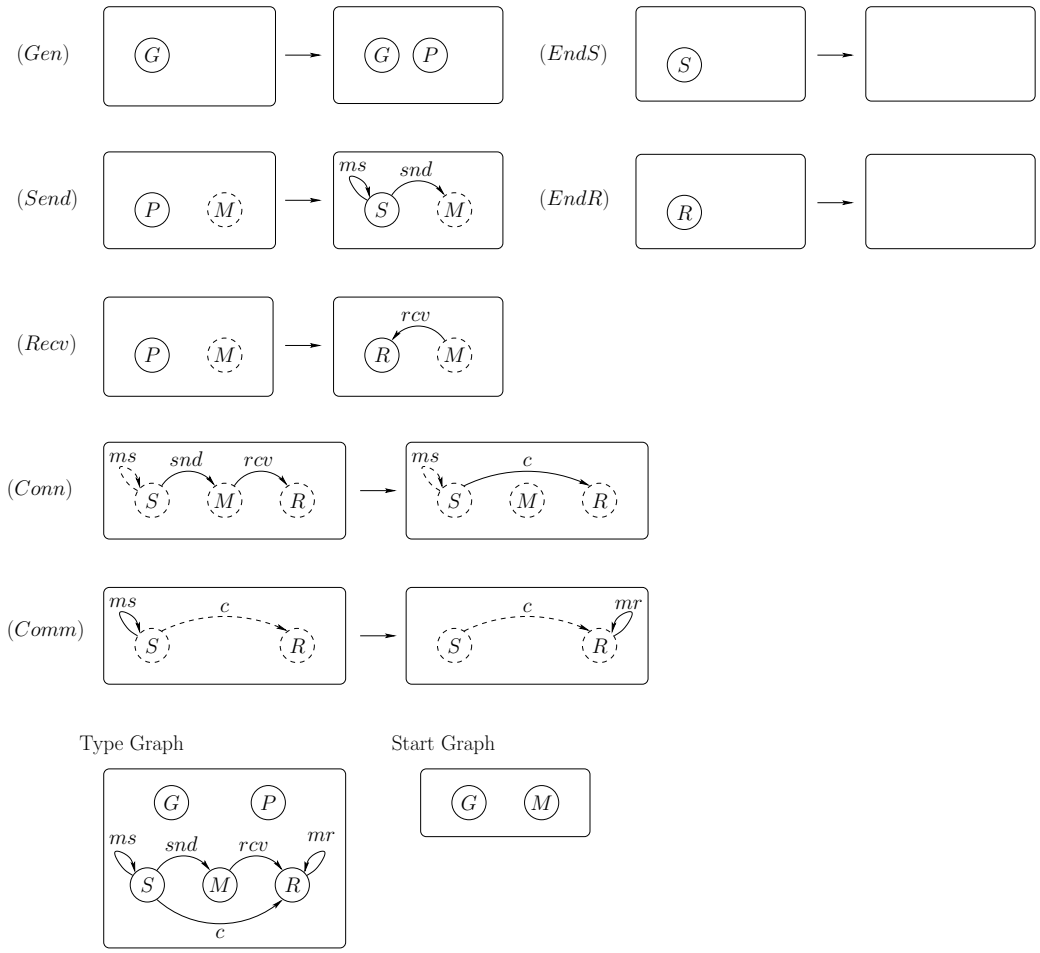


Fig. 3. The running example grammar $\mathcal{SR}$.

extension of ordinary nets with the possibility of checking for the presence of tokens which are not consumed.

To give the formal definition of contextual Petri nets we need some notation for multisets and multirelations. Let A be a set. The powerset of A is denoted by 2^A. A *multiset* of A is a function $M : A \to \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. The set of multisets of A is denoted by μA. The usual operations and relations on multisets, like multiset union $+$ or multiset difference $-$, are used. We write $a \in M$ if $M(a) > 0$ and $M \leq M'$ if $M(a) \leq M'(a)$ for all $a \in A$. Sometimes a subset $X \subseteq A$ will be seen as the multiset defined by $X(a) = 1$ if $a \in X$ and $X(a) = 0$, otherwise.

A *multirelation* $R : A \leftrightarrow B$ is a multiset of $A \times B$, i.e., a function $R : A \times B \to \mathbb{N}$. Intuitively, R relates elements $a \in A$ and $b \in B$ with multiplicity $R(a, b)$. We will limit our attention to *finitary* multirelations, namely multirelations R such that the set $\{b \in B \mid R(a, b) > 0\}$ is finite. The composition of two finitary multirelations $R : A \leftrightarrow B$ and $R' : B \leftrightarrow C$ is the (finitary) multirelation $R' \circ R : A \leftrightarrow C$ defined as $(R' \circ R)(a, c) = \sum_{b \in B} R(a, b) \cdot R'(b, c)$.

A multirelation R induces in an obvious way a (possibly partial) function $\mu R : \mu A \to \mu B$, defined as $\mu R(\sum_{a \in A} n_a \cdot a) = \sum_{b \in B} \sum_{a \in A} (n_a \cdot R(a,b)) \cdot b$.[2] A *relation* $R : A \times B$ is a subset of $A \times B$. It will be often identified with a multirelation $R : A \leftrightarrow B$ where multiplicities are bounded by one, namely $R(a,b) \leq 1$ for all $a \in A$ and $b \in B$.

Note that the composition of multirelations is not a generalisation of standard composition over relations, i.e., if we take two relations and we compose them as multirelation the result usually differs from their relational composition. As an example consider the relations $r_1 : \{a\} \times \{b,c\}$ defined by $r_1 = \{(a,b),(a,c)\}$ and $r_2 : \{b,c\} \times \{a\}$ defined by $r_1 = \{(b,a),(c,a)\}$. The relational composition of r_1 and r_2 is the identity on $\{a\}$, while their composition of as multirelations results in a proper multirelation $M : \{a\} \leftrightarrow \{a\}$ defined by $M(a,a) = 2$.

Definition 6 ((contextual) Petri nets) *A* (marked) contextual Petri net *is a tuple* $N = \langle S, Tr, F, C, m \rangle$*, where*

- S *is a set of* places*;*
- Tr *is a set of* transitions*;*
- $F = \langle F_{pre}, F_{post} \rangle$ *is a pair of multirelations from* Tr *to* S*;*
- C *is a relation between* Tr *and* S*, called the* context*;*
- $m \in \mu S$ *is a multiset called the* initial marking.

We assume, as usual, that $S \cap Tr = \emptyset$. A contextual Petri net is called a *Petri net, tout court*, if the context relation C is empty. The functions from μTr to μS induced by the multirelations F_{pre} and F_{post} are denoted by ${}^\bullet(\ )$ and $(\ )^\bullet$, respectively. If $A \in \mu Tr$ is a multiset of transitions, ${}^\bullet A$ is called its *pre-set*, while $A^\bullet$ is called its *post-set*. Moreover, by $\underline{A}$ we denote the *context* of A, defined as $\underline{A} = \{s \in S \mid \exists t \in Tr \,.\, t \in A \wedge C(t,s)\}$. Note that the context of A, although defined as a set, will be often used as a multiset. This choice will allow us to simplify the presentation.

An analogous notation is used to denote the functions from S to 2^{Tr} defined as, for $s \in S$, ${}^\bullet s = \{t \in Tr \mid F_{post}(t,s) > 0\}$, $s^\bullet = \{t \in Tr \mid F_{pre}(t,s) > 0\}$, and $\underline{s} = \{t \in Tr \mid C(t,s)\}$.

A finite multiset of transitions A is enabled at a marking M, if M contains the pre-set of A and an additional set of tokens which covers the context of A.

Definition 7 (token game) *Let* N *be a contextual net, and let* M *be a* marking *of* N*, i.e., a multiset of places* $M \in \mu S$*. A finite multiset* $A \in \mu Tr$ *is*

[2] The function μR is partial since infinite coefficients are disallowed in multisets. For instance, given the multirelation $R : \mathbb{N} \leftrightarrow \{0\}$ with $R(n,0) = 1$ for all $n \in \mathbb{N}$, then μR is undefined on the multiset $\sum_{n \in \mathbb{N}} 1 \cdot n$.

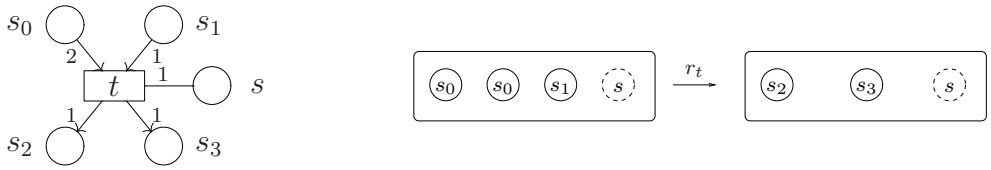


Fig. 4. A contextual Petri net transition and a corresponding SPO production.

enabled *at* M *if* ${}^\bullet A + \underline{A} \leq M$. *The* transition relation *between markings is defined as*

$$M\,[A\rangle\,M' \qquad \textit{if} \qquad A \textit{ is enabled at } M \textit{ and } M' = M - {}^\bullet A + A^\bullet.$$

Step and firing sequences, as well as reachable markings, are defined in the usual way. Note that if two transitions share a common context place s, then a single token in s is sufficient to enable their concurrent firing, i.e., a token can be "read" concurrently by several transitions.

Coming back to the relationship between nets and graph grammars, observe first that, quite obviously, a multiset $M \in \mu A$ can be seen as a set X_M equipped with a ("labelling" or "typing") function $l : X_M \to A$ and satisfying $|f^{-1}(a)| = M(a)$ for all $a \in A$ (this defines X_M up to isomorphism). Indeed, this is the way a marking (i.e., a multisets of places) is usually depicted in Petri net theory, as a set of *tokens* distributed among (or typed over) the places.

Fig. 4 shows a contextual Petri net transition t and its encoding as an SPO production r_t. Transition t has pre-set ${}^\bullet t = 2 \cdot s_0 + s_1$, post-set $t^\bullet = s_2 + s_3$, and context $\underline{t} = s$, depicted as a non-directed arc. Correspondingly, production r_t consumes two nodes typed over s_0 and one node typed over s_1, produces two nodes typed over s_2 and s_3, respectively, and preserves one node typed over s, the only element in $dom(r_t)$.

This encoding satisfies the basic properties which one would expect: production r_t can be applied to a discrete S-typed graph $\langle X_m, l : X_m \to S\rangle$ representing marking m, if and only if transition t is enabled at m and, in this case, the single-pushout construction and the firing of the transition produce equivalent resulting states.

It is worth noting that in this encoding of transitions as SPO productions, the restriction to consuming graph grammars corresponds, in the theory of Petri nets, to the common requirement that transitions must have non-empty pre-sets.

As recalled in the Introduction, the coreflection results have been proved for specific subclasses of (contextual) Petri nets, namely for *safe* nets in [39], and for *semi-weighted* (contextual) nets in [31,8]. We introduce here corresponding classes of graph grammars: the reasons why some of the presented results (mainly the characterisation of the semantics as a coreflection) do not hold for more general grammars will be explained later.

Definition 8 (safe and semi-weighted nets) *A contextual Petri net* $N = \langle S, Tr, F, C, m\rangle$ *is* safe *if (a.1) each marking* m' *reachable from* m *is a set, i.e.,* $m'(s) \leq 1$ *for all* $s \in S$*, and (a.2) the multi-relations* $F_{pre}, F_{post} : Tr \leftrightarrow S$ *are relations, i.e., the pre-set and post-set of each transition are sets.* [3]

A net N *is* semi-weighted *[31] if (b.1) the initial marking is a set, and (b.2) the multi-relation* $F_{post} : Tr \leftrightarrow S$ *is a relation.*

Since clearly (a.1) implies (b.1) and (a.2) implies (b.2), it follows that any safe net is semi-weighted. Notice also that it can be checked statically if a net is semi-weighted, just looking at the transitions and initial marking, while for safety one must consider all possible computations, because of condition (a.1).

In the encoding of nets as graph grammars sketched in the previous section, a marking is represented as a set (discrete graph) typed over the set of places. Requiring that a marking is a set, rather than a proper multiset, is equivalent to requiring the injectivity of the typing function. As general SPO productions act on general (possibly non-discrete) graphs, we can generalise smoothly the above definition to grammars as follows.

Definition 9 (safe and semi-weighted SPO graph grammars) *A grammar* $\mathcal{G} = \langle T, G_s, P, \pi\rangle$ *is* safe *if (a.1) for all* H *such that* $G_s \Rightarrow^* H$*,* H *is injective, and (a.2) for each production* $q \in P$*, the left- and right-hand side graphs* L_q *and* R_q *are injective.*

Grammar $\mathcal{G}$ *is* semi-weighted *if (b.1) the start graph* G_s *is injective, and (b.2) for each production* $q \in P$*, for any* x, y *in* $|R_q| - |L_q|$ *if* $t_{R_q}(x) = t_{R_q}(y)$ *then* $x = y$*, i.e., the right-hand side graph* R_q *is injective on the "produced part"* $|R_q| - |L_q|$*.*

[3] Often condition (a.2) is not required, but if a safe net contains a transition t with a proper multiset as pre-set or post-set, by (a.1) t will not be enabled in any reachable marking. Condition (a.2) requires that N does not contain such useless transitions.

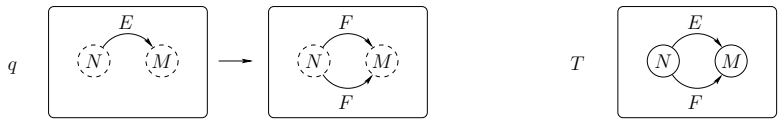


Fig. 5. A production of a graph grammar which is not semi-weighted.

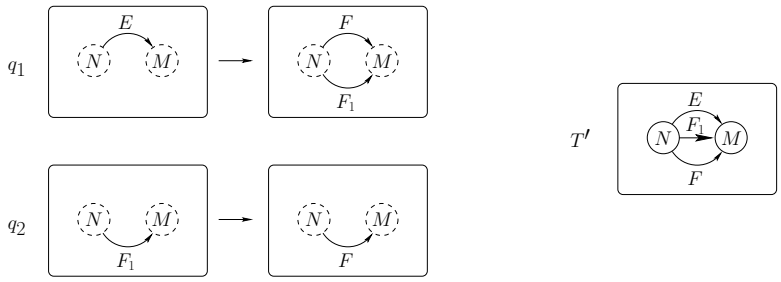


Fig. 6. Transforming a general grammar into a semi-weighted one.

Not surprisingly, it is possible to show that if we encode a Petri net N as a grammar, as sketched in Section 2.2, then N is a semi-weighted net if and only if the corresponding grammar is semi-weighted. An example of semi-weighted graph grammar is given by grammar $\mathcal{SR}$ in Fig. 3, since all productions of $\mathcal{SR}$ are injectively typed. Instead grammar $\mathcal{SR}$ is not safe since reachable graphs can be non-injective.

It is worth observing that semi-weighted graph grammars are much more expressive than safe graph grammars. In particular, it can be shown that any graph grammar $\mathcal{G}$ can be "encoded" as a semi-weighted graph grammar $\mathcal{G}'$. Roughly, any production of $\mathcal{G}$ which generates a non-injective graph is replaced by a production which generates the same graph, injectively typed over temporary items. Then an additional set of productions (finite if the right-hand sides of the productions in $\mathcal{G}$ are finite) retypes such items, one at a time. For example, consider a graph grammar $\mathcal{G}$ containing the production q depicted in Fig. 5, typed over the graph T in the same figure. In the encoding we extend the type graph with a temporary type F_1, thus obtaining the type graph T' in Fig. 6. Production q is replaced by the productions q_1 and q_2 in the same figure. When the right-hand side of a production is non-injective on produced nodes, the transformation becomes more tricky but it is still feasible.

It is possible to show that given any T-typed graph G, we have that G is reachable in the original grammar $\mathcal{G}$ if and only if it is reachable in the semi-weighted grammar $\mathcal{G}'$.

3 Graph grammar morphisms

In this section SPO graph grammars are considered as the objects of a category **GG**. This is done by defining a notion of SPO grammar morphism, which recasts in this setting the morphisms for DPO grammars introduced in [14,7], which in turn were defined as a generalisation of (contextual) Petri net morphisms.

Several notions of morphisms have been considered for Petri nets. Often they origin from an algebraic view of Petri nets which considers a Petri net as the signature of a multisorted algebra, the sorts being the places (see, e.g., [39,29,30]). More general classes of morphisms have been considered, e.g., in [10], with the aim of providing a satisfactory categorical solution to the synthesis problem. Here we focus on the notion originally proposed in [39] for the development of the unfolding approach, and extended to contextual nets in [8]. It ensures the existence of products, which can be interpreted as asynchronous compositions, and of some coproducts, modelling nondeterministic choice [40].

A morphism between two contextual nets [8] maps transitions and places of the first net into transitions and multisets of places of the second net, respectively, in such a way that the initial marking as well as the pre-set, post-set and context of each transition are "preserved".

Definition 10 (contextual net morphism) *Let $N_i = \langle S_i, Tr_i, F_i, C_i, m_i\rangle$ ($i \in \{0,1\}$) be contextual nets. A* morphism *$h : N_0 \to N_1$ is a pair $h = \langle h_T, h_S\rangle$, where $h_T : Tr_0 \rightarrowtail Tr_1$ is a partial function and $h_S : S_0 \leftrightarrow S_1$ is a finitary multirelation such that*

(1) $\mu h_S(m_0)$ is defined and $\mu h_S(m_0) = m_1$;
(2) for each transition $t \in Tr_0$, $\mu h_S({}^\bullet t)$, $\mu h_S(t^\bullet)$ and $\mu h_S(\underline{t})$ are defined, and
  (a) $\mu h_S({}^\bullet t) = {}^\bullet \mu h_T(t)$;
  (b) $\mu h_S(t^\bullet) = \mu h_T(t)^\bullet$;
  (c) $\mu h_S(\underline{t}) = \underline{\mu h_T(t)}$.

We denote by **CN** *the category having contextual nets as objects and contextual net morphisms as arrows.*

Note that in item (2.c) above, the context $\underline{t}$ of t, which is a set, is considered as a multiset. Observe also that $\mu h_T(t) = h_T(t)$ when $h_T(t)$ is defined, and $\mu h_T(t) = \emptyset$ otherwise. In the last case, by the definition above, the places in the pre-set, post-set and context of t are forced to be mapped to the empty set, i.e., $\mu h_S({}^\bullet t + t^\bullet + \underline{t}) = \emptyset$.

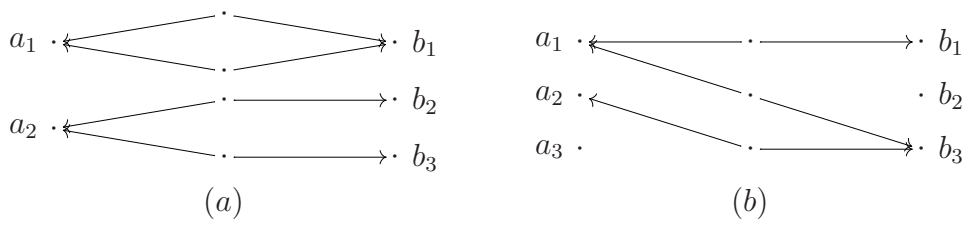


Fig. 7. The (semi-abstract) spans for the multirelations (a) $R_1(a_1, b_1) = 2$, $R_1(a_2, b_2) = 1$, $R_1(a_2, b_3) = 1$ and (b) $R_2(a_1, b_1) = 1$, $R_2(a_1, b_3) = 1$, $R_2(a_2, b_3) = 1$ (Pairs which are not mentioned are mapped to 0).

In order to extend the correspondence between graph grammars and nets discussed in Section 2.2 uniformly to morphisms, we will define grammar morphisms in such a way that they essentially coincide with contextual net morphisms if restricted to grammars which act on discrete graphs only. Keeping this goal in mind, and recalling that the type graph of a graph grammar corresponds conceptually to the set of places of a Petri net, in a morphism we shall relate the type graphs of the source and target grammar by a *semi-abstract span* in the category of graphs. Indeed, as shown, for example, in [11], in the category of finite sets and functions, semi-abstract spans correspond one-to-one with multirelations, and this correspondence lifts to span/multirelation composition. Similarly, as discussed in [36], given a category whose morphisms are seen as total maps, left injective (semi-abstract) spans over such category can be interpreted as partial maps.

Definition 11 (spans) *Let* $\mathbf{C}$ *be a category. A* (concrete) span $f : A \leftrightarrow B$ *in* $\mathbf{C}$ *is a pair of arrows* $f = \langle f^L, f^R \rangle$ *with* $f^L : X_f \to A$ *and* $f^R : X_f \to B$. *Objects* A *and* B *are called the* source *and the* target *of the span,* X_f *is the* support*, and* f^L, f^R *are called the* left leg *and the* right leg *of* f*, respectively. The span* f *will be sometimes denoted as* $\langle f^L, X_f, f^R \rangle$*, explicitly showing its support.*

Consider the equivalence $\sim$ *over the set of spans with the same source and target defined, for* $f, f' : A \leftrightarrow B$*, as* $f \sim f'$ *if there exists an isomorphism* $k : X_f \to X_{f'}$ *such that* $f'^L \circ k = f^L$ *and* $f'^R \circ k = f^R$ *(see Fig. 8.(a)). The isomorphism class of a span* f *is denoted by* $[f]$ *and called a* semi-abstract span*.*

The word "semi" in the term "semi-abstract span" reminds that only the support of the span is taken up to isomorphism, while the source and target objects are concrete.

Fig. 7 shows two spans in **Set** and the corresponding multirelations.

Definition 12 (category of spans) *Let* $\mathbf{C}$ *be a category with pullbacks. The category* $\mathbf{Span}(\mathbf{C})$ *has the same objects as* $\mathbf{C}$ *and semi-abstract spans in* $\mathbf{C}$ *as*

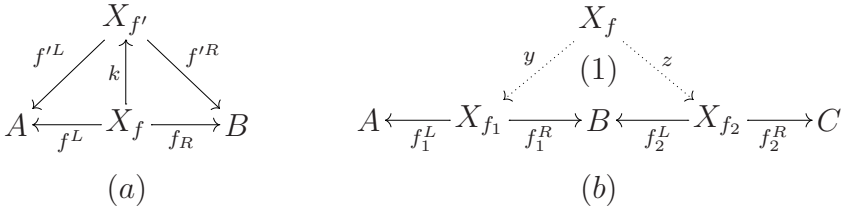


Fig. 8. Equivalence and composition of spans.

arrows. More precisely, a semi-abstract span $[f]$ *is an arrow from the source to the target of* f*. The composition of two semi-abstract spans* $[f_1] : A \leftrightarrow B$ *and* $[f_2] : B \leftrightarrow C$ *is the (equivalence class of a) span* f *constructed as in Fig. 8.(b) (i.e.,* $f^L = f_1^L \circ y$ *and* $f^R = f_2^R \circ z$*), where the square (1) is a pullback. The identity on an object* A *is the equivalence class of the span* $\langle id_A, id_A \rangle$*, where* id_A *is the identity of* A *in* $\mathbf{C}$*.*

It can be shown that composition is well-defined, namely it does not depend on the particular choice of the representatives, and that it is associative.

Recall that relations can be identified with multirelations $R : A \leftrightarrow B$ where multiplicities are bounded by one, i.e., $R(a, b) \leq 1$ for all $a \in A$ and $b \in B$. The corresponding condition on a span $f : A \leftrightarrow B$ is the injectivity of function $\langle f^L, f^R \rangle$ from X_f to $A \times B$, which implies the existence of at most one path between any two elements $a \in A$ and $b \in B$. For instance, the span in Fig. 7.(a) is not relational, while that in Fig. 7.(b) is relational. This yields to the following definition, for a general category $\mathbf{C}$:

Definition 13 (relational span) *Let* $\mathbf{C}$ *be a category. A span* $f : A \leftrightarrow B$ *in* $\mathbf{C}$ *is called* relational *if* $\langle f^L, f^R \rangle : X_f \to A \times B$ *is mono.*

In other words $f : A \leftrightarrow B$ is relational if given any object C and pair of arrows $g, h : C \to X_f$, if $f^R \circ g = f^R \circ h$ and $f^L \circ g = f^L \circ h$ then $g = h$. Along the paper we shall sometimes use the following equivalent condition, for a span $f : A \leftrightarrow B$ in **Set** or in **Graph**:

$$\forall x, y \in X_f.\ x \neq y \Rightarrow f^R(x) \neq f^R(y) \ \vee \ f^L(x) \neq f^L(y).$$

In general, relational spans do not compose [11], in the sense that if $[f_1] : A \leftrightarrow B$ and $[f_2] : B \leftrightarrow C$ are two semi-abstract relational spans, then their composition $[f] : A \leftrightarrow C$, as for Definition 12, is not necessarily relational. [4] However, there are interesting classes of relational spans which do compose. Observe in fact that a span $f : A \leftrightarrow B$ is certainly relational when either its left or its right leg is mono. Such spans correspond one-to-one with partial

[4] The apparent inconsistency of this statement with the fact that relations do compose, can be explained by recalling that the composition of relations, seen as specific multirelations, does not coincide with the standard composition of relations.

morphisms from A to B or backward. In fact, a partial morphism $g : A \to B$ can be identified with the span

$$A \hookleftarrow dom(g) \xrightarrow{g} B$$

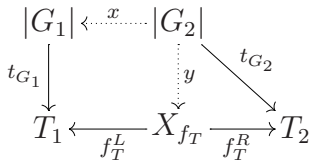
and similarly a partial morphism $h : B \to A$ can be represented as the span

$$A \xleftarrow{h} dom(h) \hookrightarrow B$$

where unlabelled arrows are inclusions.

We have seen that a multiset of a set A can be regarded as a set labelled on A, and that a multirelation $R : A \leftrightarrow B$ can be identified with a semi-abstract span from A to B. The next definition presents a categorical analogue of constructing the image of a multiset through a multirelation, namely of the function $\mu R : \mu A \to \mu B$. The definition is given for graphs, but it could be generalised to any category with pullbacks.

Definition 14 (pullback-retyping relation) *Let $[f_T] : T_1 \leftrightarrow T_2$ be a semi-abstract span in* **Graph**, *let G_1 be a T_1-typed graph, and let G_2 be a T_2-typed graph. Then G_1 and G_2 are* related by pullback-retyping (via $[f_T]$) *if there exist morphisms $x : |G_2| \to |G_1|$ and $y : |G_2| \to X_{f_T}$ such that the square in the following diagram is a pullback:*



In this case we will write $f_T\{x, y\}(G_1, G_2)$, or simply $f_T(G_1, G_2)$ if we are not interested in morphisms x and y.

We are now ready to define grammar morphisms. Besides the component specifying the multirelation between the type graphs, a morphism from $\mathcal{G}_1$ to $\mathcal{G}_2$ includes a (partial) mapping between production names. Furthermore a third component explicitly relates the (untyped) graphs underlying corresponding productions of the two grammars, as well as the graphs underlying the start graphs.

Definition 15 (grammar morphism) *Let $\mathcal{G}_i = \langle T_i, G_{s_i}, P_i, \pi_i \rangle$ $(i \in \{1, 2\})$ be graph grammars. A morphism $f : \mathcal{G}_1 \to \mathcal{G}_2$ is a triple $\langle [f_T], f_P, \iota_f \rangle$ where*

- *$[f_T] : T_1 \leftrightarrow T_2$ is a semi-abstract span in* **Graph**, *called the* type-span*;*
- *$f_P : P_1 \to P_2 \cup \{\emptyset\}$ is a total function, where $\emptyset$ is a new production name (not in P_2), with associated production $\emptyset \rightarrowtail \emptyset$;*

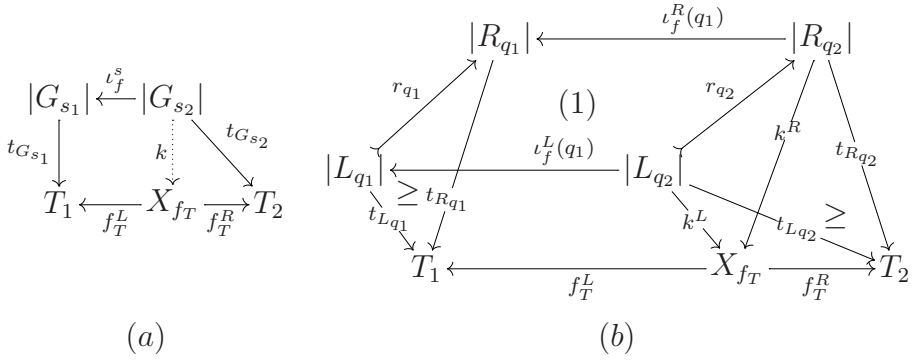


Fig. 9. Diagrams for SPO grammar morphisms.

- ι_f *is a family* $\{\iota_f(q_1) \mid q_1 \in P_1\} \cup \{\iota_f^s\}$ *of morphisms in* **Graph** *such that* $\iota_f^s : |G_{s_2}| \to |G_{s_1}|$ *and for each* $q_1 \in P_1$*, if* $f_P(q_1) = q_2$*, then* $\iota_f(q_1)$ *is a pair* $\langle \iota_f^L(q_1) : |L_{q_2}| \to |L_{q_1}|, \iota_f^R(q_1) : |R_{q_2}| \to |R_{q_1}| \rangle$.

such that the following conditions are satisfied:

(1) Preservation of the start graph.
There exists a morphism k *such that* $f_T\{\iota_f^s, k\}(G_{s_1}, G_{s_2})$*, i.e., the diagram in Fig. 9.(a) commutes and the square is a pullback.*

(2) Preservation of productions.
For each $q_1 \in P_1$*, with* $q_2 = f_P(q_1)$*, there exist morphisms* k^L *and* k^R *such that the square (1) in Fig. 9.(b) commutes, and* $f_T\{\iota_f^Y(q_1), k^Y\}(Y_{q_1}, Y_{q_2})$ *for* $Y \in \{L, R\}$.

The morphism f *is called* relational *if the type component* f_T *is relational.*

It is worth noticing that, for technical convenience, the partial mapping on production names is represented as a total mapping by enriching the target set with a distinguished element $\emptyset$, representing "undefinedness". In this way the condition asking the preservation of productions (Condition 2) faithfully rephrases the condition that the pre- and post-set of a transition on which the morphism is undefined are necessarily mapped to the empty multiset (see after Definition 10).

Definition 16 (category of graph grammars) *We denote by* **GG** *the category where objects are* SPO *graph grammars and arrows are graph grammar morphisms. By* **SGG** *we denote the full subcategory of* **GG** *having semi-weighted graph grammars as objects.*

As in [35,1,14] one can show that grammar morphisms are "simulations", namely that if $f : \mathcal{G}_1 \to \mathcal{G}_2$ is a graph grammar morphism, then every derivation ρ_1 in $\mathcal{G}_1$ is related by pullback-retyping via $[f_T]$ to a derivation ρ_2 in $\mathcal{G}_2$. As already observed, as a consequence of the partial arbitrariness in the choice of the pullback components, such correspondence, differently from [14], is not

"functional".

Lemma 17 *Let $f : \mathcal{G}_1 \to \mathcal{G}_2$ be a graph grammar morphism, and let $\delta_1 : G_1 \Rightarrow_{q_1} H_1$ be a direct derivation in $\mathcal{G}_1$. Then there exists a corresponding direct derivation $\delta_2 : G_2 \Rightarrow_{f_P(q_1)} H_2$ in $\mathcal{G}_2$, such that $f_T(G_1, G_2)$ and $f_T(H_1, H_2)$.*

The proof follows the same outline of that for the corresponding result for the DPO approach (Lemma 5.37 in [1]) and thus it is omitted.

4 Occurrence graph grammars and their morphisms

In this section we introduce occurrence grammars. Through the unfolding construction, they will be used to provide a static description of the computations of a given graph grammar, recording the events (production applications) which can appear in all possible derivations and the dependency relations among them. Next, the full subcategory of **GG** including all occurrence graph grammars is given an alternative, simpler characterisation. This will be used, in particular, in the next section to formalise the connection between occurrence grammars and occurrence contextual nets

4.1 Occurrence grammars

Analogously to what happens for Petri nets, occurrence grammars are *safe* grammars, where the dependency relations between productions satisfy suitable acyclicity and well-foundedness requirements. While for nets it suffices to take into account only the causality and conflict relations, for grammars the fact that a production application not only consumes and produces, but also preserves a part of the state leads to a form of asymmetric conflict between productions. Quite interestingly, as we shall discuss later, there is no need of taking into account the dependencies between events related to the side-effects of rule applications, i.e., the deletion of an edge caused by the deletion of its source or target node.

Recall that in a safe grammar (Definition 9) each graph G reachable from the start graph is injectively typed, and thus we can identify it with the corresponding subgraph $t_G(|G|)$ of the type graph. With this identification, a production can only be applied to the subgraph of the type graph which is the image via the typing morphism of its left-hand side. Thus, according to its typing, we can think that a production *produces*, *preserves* or *consumes* items of the type graph, and using a net-like language, we speak of pre-set,

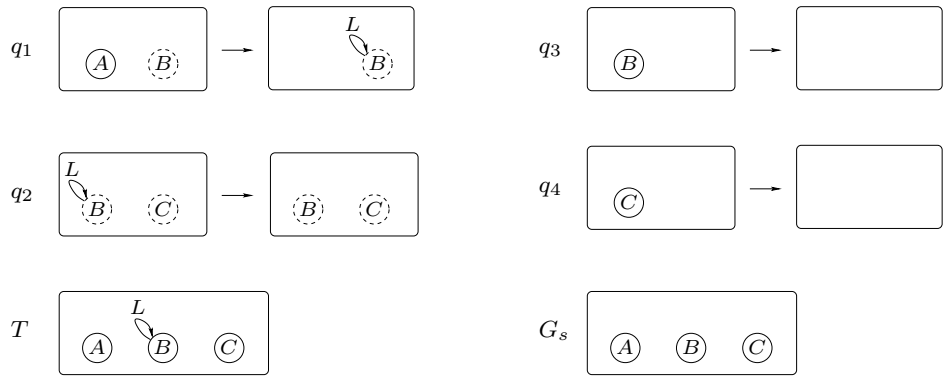


Fig. 10. A safe SPO graph grammar $\mathcal{G}$.

context and post-set of a production, correspondingly. Actually, it is worth mentioning that the next definition captures the intuition just sketched only for safe grammars, but for technical reasons we state it for arbitrary graph grammars.

Definition 18 (pre-set, post-set and context of a production) *Let $\mathcal{G} = \langle T, G_s, P, \pi \rangle$ be a graph grammar. For any production $q \in P$ we define its* pre-set ${}^\bullet q$, context $\underline{q}$ *and* post-set $q^\bullet$ *as the following subsets of $E_T \cup N_T$:*

$$ {}^\bullet q = t_{L_q}(|L_q| - |dom(r_q)|) \qquad \underline{q} = t_{L_q}(|dom(r_q)|) $$
$$ q^\bullet = t_{R_q}(|R_q| - r_q(|dom(r_q)|)). $$

Symmetrically, for each item $x \in T$ we define the following subsets of P: ${}^\bullet x = \{q \in P \mid x \in q^\bullet\}$, $x^\bullet = \{q \in P \mid x \in {}^\bullet q\}$, $\underline{x} = \{q \in P \mid x \in \underline{q}\}$.

In order to illustrate these concepts, let us consider the simple grammar $\mathcal{G}$ in Fig. 10, which can be easily seen to be safe. Then ${}^\bullet q_1 = \{A\}$, $\underline{q_1} = \{B\}$ and $q_1{}^\bullet = \{L\}$, while ${}^\bullet B = \emptyset$, $\underline{B} = \{q_1, q_2\}$ and $B^\bullet = \{q_3\}$.

Also the causality and the asymmetric conflict relations defined below are meaningful only for safe grammars, but it is technically convenient to introduce them for arbitrary graph grammars.

Definition 19 (causality relation) *The* causality relation *of a grammar $\mathcal{G}$ is the binary relation $<$ over $Elem(\mathcal{G})$ defined as the least transitive relation satisfying: for any node or edge $x \in T$, and for productions $q, q' \in P$*

(1) if $x \in {}^\bullet q$ then $x < q$;
(2) if $x \in q^\bullet$ then $q < x$;
(3) if $q^\bullet \cap \underline{q'} \neq \emptyset$ then $q < q'$.

As usual $\leq$ is the reflexive closure of $<$. Moreover, for $x \in Elem(\mathcal{G})$ we denote by $\lfloor x \rfloor$ the set of causes of x in P, namely $\{q \in P : q \leq x\}$.

The first two clauses of the definition of relation $<$ are obvious. The third one formalises the fact that if an item is generated by q and it is preserved by q', then q', to be applied, requires that q had already been applied.

Notice that the fact that an item is preserved by q and consumed by q', i.e., $\underline{q} \cap {}^\bullet q' \neq \emptyset$ (e.g., item $C \in \underline{q_2} \cap {}^\bullet q_4$ in grammar $\mathcal{G}$ of Fig. 10), does not imply $q < q'$. Actually, the dependency between the two productions is a kind of *asymmetric conflict* (see [8,34,25]). The application of q' prevents q from being applied, so that q can never follow q' in a derivation. However, the converse is not true, since q *can* be applied before q'. Equivalently, when both q and q' occur in a derivation then q must precede q'.

Definition 20 (asymmetric conflict) *The* asymmetric conflict relation *of a grammar $\mathcal{G}$ is the binary relation $\nearrow$ over the set of productions, defined by:*

(1) if $\underline{q} \cap {}^\bullet q' \neq \emptyset$ then $q \nearrow q'$;
(2) if ${}^\bullet q \cap {}^\bullet q' \neq \emptyset$ and $q \neq q'$ then $q \nearrow q'$;
(3) if $q < q'$ then $q \nearrow q'$.

Condition 1 is justified by the discussion above. Condition 2 essentially expresses the fact that the ordinary symmetric conflict is encoded, in this setting, as an asymmetric conflict in both directions. Finally, since $<$ represents a global order of execution, while $\nearrow$ determines an order of execution only locally to each computation, it is natural to impose $\nearrow$ to be an extension of $<$ (Condition 3).

Notice that if a set of productions forms a cycle of asymmetric conflicts $q_0 \nearrow q_1 \nearrow \ldots \nearrow q_n \nearrow q_0$, then such productions cannot appear in the same computation, otherwise the application of each production should precede the application of the production itself; this fact can be naturally interpreted as a form of n-ary conflict.

Definition 21 (conflict) *The* conflict *relation $\# \subseteq 2^P$ associated to a grammar $\mathcal{G}$ is defined as:*

$$\frac{q_0 \nearrow q_1 \nearrow \ldots \nearrow q_n \nearrow q_0}{\#\{q_0, q_1, \ldots, q_n\}} \qquad \frac{\#(A \cup \{q\}) \quad q \leq q'}{\#(A \cup \{q'\})}$$

where A denotes a generic finite subset of P. We use the infix notation $q\#q'$ for $\#\{q, q'\}$.

As already mentioned, the side-effects of production applications can be disregarded when analysing the dependency relations between events. This fact, which is later formalised by showing the tight relation between concurrency and reachability (see Proposition 35) can be intuitively understood as follows:

Causality. Assume that production q produces an edge e, and q' deletes e as side-effect (because it deletes its source or its target). At a first glance we could think that q' should causally depend on q. However, even if q' consumes the resource e produced by q, the application of q is not necessary to make q' applicable, since q' does not explicitly require the presence of e. Hence q' does not causally depend on q. For instance, referring to grammar $\mathcal{G}$ in Fig. 10, the application of q_3 after q_1 deletes node B and edge L as side-effect. However q_3 does not depend on q_1 since it can be applied already to the start graph.

Asymmetric conflict. Also asymmetric conflict (called *weak conflict* in [35]) can be defined disregarding the mentioned side-effects. This is basically due to the fact that when a production uses (consumes or preserves) an edge, it must use necessarily the corresponding source and target nodes as well, and therefore dependencies related to side-effects are subsumed by those induced by explicitly used items. E.g., consider again grammar $\mathcal{G}$ in Fig. 10. Observe that, after the application of q_1 to the start graph, production q_3 prevents q_2 from being applied since it deletes, as side-effect, edge L which is needed by q_2. However, to consume L, production q_2 must preserve or consume node B (actually, it preserves it) and thus the "ordinary" definition of asymmetric conflict already tells us that $q_2 \nearrow q_3$.

An *occurrence grammar* is an acyclic grammar which represents, in a branching structure, several possible computations beginning from its start graph and using each production at most once. Recall that a relation $R \subseteq X \times X$ is *finitary* if for any $x \in X$, the set $\{y \in X \mid R(y, x)\}$ is finite.

Definition 22 (occurrence grammar) *An* occurrence grammar *is a safe grammar*
$\mathcal{O} = \langle T, G_s, P, \pi \rangle$ *such that*

(1) the causality relation $<$ is irreflexive, its reflexive closure $\leq$ is a partial order, and, for any $q \in P$, the set $\lfloor q \rfloor$ is finite and the asymmetric conflict $\nearrow$ is acyclic on $\lfloor q \rfloor$;
(2) the start graph G_s is the set $Min(\mathcal{O})$ of minimal elements of $\langle Elem(\mathcal{O}), \leq \rangle$ [5] (with the graphical structure inherited from T and typed by the inclusion);
(3) any item x in T is created by at most one production in P, i.e., $|{}^\bullet x| \leq 1$;

An occurrence grammar is deterministic *if relation $\nearrow^+$, the transitive closure of $\nearrow$, is finitary and irreflexive.*

[5] Notice that $Min(\mathcal{O}) \subseteq N_T \cup E_T$, i.e., it does not contain productions, since the grammar is consuming.

We denote by **OGG** *the full subcategory of* **GG** *with occurrence grammars as objects.*

Since the start graph of an occurrence grammar $\mathcal{O}$ is determined by $Min(\mathcal{O})$, we often do not mention it explicitly. One can show that, given a grammar $\mathcal{G}$ where all productions are injectively typed, if $\mathcal{G}$ satisfies (1)-(3) above then it is safe, and thus it is an occurrence grammar.

Intuitively, conditions (1)–(3) recast in the framework of graph grammars the analogous conditions of occurrence nets (actually of occurrence contextual nets [8]). In particular, in Condition (1), the acyclicity of asymmetric conflict on $\lfloor q \rfloor$ corresponds to the requirement of irreflexivity for the conflict relation in occurrence nets. A simple example of occurrence grammar is given by grammar $\mathcal{G}$ in Fig. 10. A more complex example can be found in Fig. 11 (for the moment, ignore how this grammar is related to grammar $\mathcal{SR}$ in Fig. 3).

As in the case of Petri nets, reachable states can be characterised in terms of a concurrency relation.

Definition 23 (concurrent graph) *Let $\mathcal{O} = \langle T, P, \pi \rangle$ be an occurrence grammar. A subgraph G of T is called* concurrent*, written $conc(G)$, if*

(1) $\nearrow_G$, the asymmetric conflict restricted to $\bigcup_{x \in G} \lfloor x \rfloor$, is acyclic and finitary;
(2) $\neg(x < y)$ for all $x, y \in G$.

We will see later that a subgraph G of T is concurrent if and only if it is a subgraph of a graph reachable from the start graph by means of a derivation which applies all the productions in $\bigcup_{x \in G} \lfloor x \rfloor$ exactly once in any order compatible with $\nearrow$.

4.2 An alternative characterisation of occurrence grammar morphisms

We next provide an alternative, much simpler characterisation of morphisms between occurrence grammars which will be useful in the sequel.

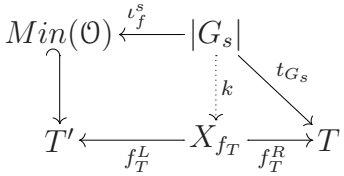
We first prove a basic property, which will be also pivotal for expressing the unfolding as a universal construction (Theorem 45). This is a key point where the restriction to semi-weighted grammars plays a role, since the lemma fails to hold for arbitrary grammars (see Section 9.1).

Lemma 24 *Let $\mathcal{G} = \langle T, G_s, P, \pi \rangle$ be a semi-weighted grammar, let $\mathcal{O} = \langle T', G'_s, P', \pi' \rangle$ be an occurrence grammar and let $f : \mathcal{O} \to \mathcal{G}$ be a grammar morphism. Then the type span $[f_T]$ of the morphism is relational.*

PROOF. Recall (Definition 13) that the span $f_T : T' \leftrightarrow T$ is relational if $\langle f_T^L, f_T^R \rangle : X_{f_T} \to T \times T'$ is mono. In turn, in the categories **Set** and **Graph** this amounts to say

$$\forall x, y \in X_{f_T}.\ x \neq y \Rightarrow f_T^R(x) \neq f_T^R(y) \ \vee \ f_T^L(x) \neq f_T^L(y).$$

We proceed by contraposition. Consider $x, y \in X_{f_T}$ such that $f_T^L(x) = f_T^L(y) = z'$ and $f_T^R(x) = f_T^R(y) = z$. Since $\mathcal{O}$ is an occurrence grammar, necessarily z' is in the start graph or in the post-set of some production. Let us assume that $z' \in Min(\mathcal{O})$. By definition of grammar morphism, there exists a morphism $k : |G_s| \to X_{f_T}$ such that the following diagram commutes and the square is a pullback, where the unlabelled arrow is an inclusion:



The fact that $f_T^L(x) = f_T^L(y) = z'$ and $z' \in Min(\mathcal{O})$ implies that there are $x'', y'' \in |G_s|$ such that $k(x'') = x$ and $k(y'') = y$. Recalling that the triangle on the right commutes we have

$$t_{G_s}(x'') = f_T^R(k(x'')) = f_T^R(x) = f_T^R(y) = f_T^R(k(y'')) = t_{G_s}(y'').$$

Since the graph G_s is injectively typed, we conclude that $x'' = y''$, and thus we deduce the desired equality $x = k(x'') = k(y'') = y$. Similar reasoning applies if the item z belong the post-set of some production, since the grammar is semi-weighted and thus the right-hand sides of the productions are injectively typed on produced items. $\Box$

Observe that, as an immediate consequence of the above lemma, if $f : \mathcal{O} \to \mathcal{G}$ is a grammar morphism, where $\mathcal{G}$ is a semi-weighted grammar and $\mathcal{O}$ is an occurrence grammar, then the morphism k, such that $f_T\{\iota_f^s, k\}(G_s, G_{s'})$ (see Definition 15, condition (1)) is uniquely determined. Similarly, for each $q \in P$, with $q' = f_P(q)$, the morphisms k^L and k^R such that $f_T\{\iota_f^X(q), k^X\}(X_q, X_{q'})$ for $X \in \{L, R\}$ (see Definition 15, condition (2)) are uniquely determined.

Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be occurrence grammars and let $f : \mathcal{O}_1 \to \mathcal{O}_2$ be a morphism. By Lemma 24, $[f_T]$ is relational. Therefore we can safely replace the span and the ι components of the morphism with the corresponding relation between the items of T_1 and T_2, namely with

$$\{(x_1, x_2) \mid \exists x \in X_{f_T}.\ f_T^L(x) = x_1 \ \wedge \ f_T^R(x) = x_2\}$$

This intuition, which is formalised in the rest of this section, will allow us to present an equivalent but more manageable notion of morphisms between occurrence grammars. First we introduce the notion of relation between graphical structures.

Definition 25 (graph relation) *A* graph relation τ *over graphs* $G_1 = \langle N_1, E_1, \mathsf{s}_1, \mathsf{t}_1\rangle$ *and* $G_2 = \langle N_2, E_2, \mathsf{s}_2, \mathsf{t}_2\rangle$*, written* $\tau : G_1 \times G_2$*, is a subgraph* $\tau \subseteq G_1 \times G_2$ *of the cartesian product of* G_1 *and* G_2*. Thus* $\tau = \langle N_\tau, E_\tau, \mathsf{s}_\tau, \mathsf{t}_\tau\rangle$*, where* $N_\tau \subseteq N_1 \times N_2$*,* $E_\tau \subseteq E_1 \times E_2$*,* $\mathsf{s}_\tau(\langle e_1, e_2\rangle) = \langle \mathsf{s}_1(e_1), \mathsf{s}_2(e_2)\rangle$*, and similarly for* t_τ*. For* $x_1 \in N_1 \cup E_1$ *and* $x_2 \in N_2 \cup E_2$*, we write* $\tau(x_1, x_2)$ *if* $\langle x_1, x_2\rangle \in N_\tau \cup E_\tau$*.*

Quite obviously, graph relations are one-to-one with relational morphisms in **Span**(**Graph**). The isomorphism is defined as follows:

- Given a *graph relation* $\tau : G_1 \times G_2$, let *Span*(τ) be the relational span *Span*$(\tau) = G_1 \stackrel{\pi_1}{\leftarrow} [\tau] \stackrel{\pi_2}{\rightarrow} G_2$, where π_1 and π_2 are the projections on the first and second component, respectively.
- Given a *relational span* $f : G_1 \stackrel{f^L}{\leftarrow} [X_f] \stackrel{f^R}{\rightarrow} G_2$, let *Rel*$(f) : G_1 \times G_2$ be the graph relation such that *Rel*$(f)(x_1, x_2)$ if there exists $x \in X_f$ such that $f^L(x) = x_1$ and $f^R(x) = x_2$, for $x_1 \in G_1, x_2 \in G_2$.

It is immediate to see that *Span*$(\cdot)$ and *Rel*$(\cdot)$ are well-defined and inverse to each other. A graph relation over G_1 and G_2 naturally induces a relation between sets of items of G_1 and G_2, as given in the next definition. Interestingly, this relation coincides with the pullback-retyping relation induced by the corresponding relational span.

Definition 26 (injective image through a graph relation) *Let* $\tau : G_1 \times G_2$ *be a graph relation. Given two subsets* X_i *of items (edges and nodes) of* G_i *(*$i \in \{1, 2\}$*), we write* $\tau(X_1, X_2)$ *and we say that* X_2 is the injective image of X_1 through τ*, if*

(1) for any $x_1 \in X_1$ *and* $x_2 \in G_2$*, if* $\tau(x_1, x_2)$ *then* $x_2 \in X_2$*;*
(2) for any $x_2 \in X_2$ *there exists a unique* $x_1 \in X_1$ *such that* $\tau(x_1, x_2)$*.*

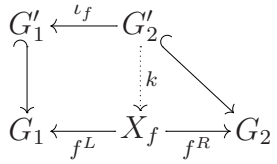
Similarly, given two subgraphs G'_i *of* G_i *(for* $i \in \{1, 2\}$*) we will write* $\tau(G'_1, G'_2)$ *whenever* $\tau(N_1 \cup E_1, N_2 \cup E_2)$*.*

The relation τ *is said to be* injective *on* G'_1 *if for any* $x_1, y_1 \in G'_1$*, if* $\tau(x_1, z)$ *and* $\tau(y_1, z)$ *then* $x_1 = y_1$*.*

Observe that the notation used for the the injective image through a graph relation is the same as that for the pullback-retyping relation in Definition 14. This abuse of notation is motivated by the result below.

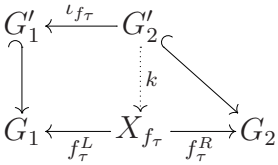
Lemma 27 (injective image is pullback-retyping) *Given a relational span $f : G_1 \leftrightarrow G_2$ and two subgraphs G'_1 and G'_2 of G_1 and G_2 respectively, we have that $f(G'_1, G'_2)$ (with G'_i seen as graphs typed over G_i by the inclusion) if and only if $Rel(f)(G'_1, G'_2)$.*

PROOF. ($\Rightarrow$) Let $f : G_1 \leftrightarrow G_2$ be a relational span, and G'_1, G'_2 be subgraphs of G_1 and G_2, respectively. If $f\{\iota_f, k\}(G'_1, G'_2)$ for some morphisms ι_f and k, we know that the following diagram can be constructed



where the square is a pullback. Exploiting the fact that, by Lemma 24, the morphism k is uniquely determined, we can easily conclude that $Rel(f)(G'_1, G'_2)$.

($\Leftarrow$) Suppose that $\tau : G_1 \times G_2$ is a graph relation and that $\tau(G'_1, G'_2)$. To simplify the notation, let f_τ denote the corresponding relational span $Span(\tau) : G_1 \leftrightarrow G_2$ and let us show that $f_\tau(G'_1, G'_2)$. By Definition 25, we know that for any item $x_2 \in G'_2$ there exists a unique $x_1 \in G'_1$ such that $\tau(x_1, x_2)$: this defines a mapping $\iota_{f_\tau} : G'_2 \to G'_1$ which is easily shown to be a well-defined graph morphism. Next, define $k : G'_2 \to X_{f_\tau}(= \tau)$ as $k(x_2) = \langle \iota_{f_\tau}(x_2), x_2 \rangle$ for all $x_2 \in G'_2$. Then a straight calculation allows to deduce that the diagram below is a pullback.



Coming back to the desired implication, assume that $Rel(f)(G'_1, G'_2)$. By the above consideration we have that $Span(Rel(f))(G'_1, G'_2)$. Recalling that $Span(Rel(f)) = f$, we conclude $f(G'_1, G'_2)$. □

We are now ready to present the equivalent formulation of the category of occurrence grammars.

Definition 28 (Occurrence grammars and relational morphisms) *Let* **OGGRel** *be the category where objects are occurrence grammars and morphisms are pairs $f = \langle f_P, f_\tau \rangle : \mathcal{O}_1 \to \mathcal{O}_2$ where $f_P : P_1 \to P_2$ is a function*

and $f_\tau : T_1 \times T_2$ *is a graph relation, such that*

(1) $f_\tau(|G_{s2}|, |G_{s1}|)$
(2) for any $q_1 \in P_1$*, if* $f_P(q_1) = q_2$ *then*
  (a) $f_\tau({}^\bullet q_1, {}^\bullet q_2)$*;*
  (b) $f_\tau(\underline{q_1}, \underline{q_2})$*;*
  (c) $f_\tau(\overline{q_1}^\bullet, \overline{q_2}^\bullet)$

Theorem 29 *The categories* **OGG** *and* **OGGRel** *are isomorphic.*

PROOF. Let F : **OGG** $\rightarrow$ **OGGRel** and G : **OGGRel** $\rightarrow$ **OGG** be the functors defined as follows:

- both F and G are the identity on objects;
- given a morphism $f : \mathcal{O}_1 \rightarrow \mathcal{O}_2$ in **OGG**, $f = \langle f_P : P_1 \rightarrow P_2, f_T : T_1 \leftrightarrow T_2, \iota_f \rangle$, then $F(f) = \langle f_P : P_1 \rightarrow P_2, \mathit{Rel}(f_T) : T_1 \times T_2 \rangle$;
- given a morphism $h : \mathcal{O}_1 \rightarrow \mathcal{O}_2$ in **OGGRel**, $h = \langle h_P : P_1 \rightarrow P_2, h_\tau : T_1 \times T_2 \rangle$ then $G(h) = \langle h_P : P_1 \rightarrow P_2, \mathit{Span}(h_\tau) : T_1 \leftrightarrow T_2, \iota \}$, where the components of ι on the start graph and on the left- and right-hand sides of the productions are uniquely determined, as in the proof of ($\Leftarrow$) in Lemma 27, by exploiting conditions (1) and (2) of Definition 28.

The well-definedness of F and G can be proved easily by exploiting Lemma 27. The fact that they are inverse to each other follows immediately because so are $\mathit{Rel}(\cdot)$ and $\mathit{Span}(\cdot)$. $\Box$

5 Relating occurrence grammars and occurrence contextual nets

In this section we show that the relationship between graph grammars and contextual nets sketched in Section 2 can be made much more tight if we restrict to occurrence grammars and occurrence contextual nets. Formally, we show that there exists an adjunction between the corresponding categories. This is one of the key steps of the paper, since it will allow to exploit some results in [8] for completing the chain of coreflections from the category of grammars to the category of domains.

5.1 Occurrence contextual nets

Recall from [8] that in a contextual net N causality $<_N$ and asymmetric conflict $\nearrow_N$ are defined exactly as for graph grammars (see Definitions 19

and 20).

Definition 30 (occurrence contextual net) *An* occurrence contextual net *is a safe contextual net $N = \langle S, Tr, F, C, m\rangle$ such that*

(1) the causality relation $<_N$ is irreflexive, its reflexive closure $\leq_N$ is a partial order, and, for any $t \in Tr$, the set $\lfloor t \rfloor$ is finite and the asymmetric conflict $\nearrow_N$ is acyclic on $\lfloor t \rfloor$;
(2) the initial marking m is the set of minimal places w.r.t. $\leq_N$, i.e., $m = \{s \in S : {}^\bullet s = \emptyset\}$;
(3) each place $s \in S$ is in the post-set of at most one transition, i.e., $|{}^\bullet s| \leq 1$.

An occurrence contextual net is deterministic *if relation $\nearrow_N^+$, the transitive closure of $\nearrow_N$, is finitary and irreflexive.*

As in the case of graph grammars, since the initial marking of an occurrence contextual net is determined by its structure we will often omit to mention it.

Occurrence contextual nets determine a full subcategory of the category **CN** of contextual nets. The morphisms in this subcategory can be described in a slightly simpler way than general contextual net morphisms (Definition 10), essentially because the multirelation among places can be shown to be a proper relation [8].

Definition 31 (category of occurrence nets) *Given two occurrence contextual nets N_0 and N_1, a morphism $h : N_0 \to N_1$ is a pair $h = \langle h_T, h_S\rangle$, where $h_T : Tr_0 \rightarrowtail Tr_1$ is a partial function and $h_S : S_0 \times S_1$ is a relation such that*

(1) $h_S(m_0, m_1)$ and
(2) for each $t \in Tr$,
(i) $h_S({}^\bullet t, {}^\bullet h_T(t))$, (ii) $h_S(t^\bullet, h_T(t)^\bullet)$ (iii) $h_S(\underline{t}, \underline{h_T(t)})$

where the image of a set through a relation is defined as for graphs (see Definition 26). We denote by **OCN** *the category of occurrence contextual nets with the above morphisms.*

5.2 From occurrence grammars to occurrence contextual nets

We next define a functor Net that maps each occurrence graph grammar to an occurrence contextual net. This is done by forgetting the graphical structure of the state, i.e., by considering each graph as an unstructured collection of nodes and edges. Although in the transformation the graphical structure is lost, due to the specific properties of occurrence graph grammars we will still

get a tight relationship between the reachable states in the two models. In the next section we will prove indeed that functor Net has a left adjoint.

Definition 32 (from occurrence grammars to contextual nets) *Let* $\mathsf{Net} : \mathbf{OGGRel} \to \mathbf{OCN}$ *be the functor defined as follows:*

- *for any occurrence grammar* $\mathcal{O} = \langle T, P, \pi \rangle$, *define* $\mathsf{Net}(\mathcal{O}) = \langle N_T \cup E_T, P, {}^\bullet(.), \underline{(.)}, (.)^\bullet \rangle$
- *for any occurrence grammar morphism* $f : \mathcal{O}_1 \to \mathcal{O}_2$ *in* **OGGRel**, *with* $f = \langle f_P : P_1 \to P_2, f_\tau : T_1 \times T_2 \rangle$, *define* $\mathsf{Net}(f) = \langle f_P, N_{f_\tau} \cup E_{f_\tau} \rangle$.

In words, the net associated to a grammar has nodes and edges of the type graph as places, and productions as transitions, with the pre-, context and post-set functions defined exactly as for the grammar. Since the defining conditions of occurrence grammars and of their morphisms are analogous, it is straightforward to check that the functor Net is well-defined. Also, since causality, asymmetric conflict and concurrency are defined in $\mathcal{O}$ and in $\mathsf{Net}(\mathcal{O})$ exactly in the same way, and thus a subgraph of the type graph in $\mathcal{O}$ is concurrent iff the corresponding set of places in $\mathsf{Net}(\mathcal{O})$ is concurrent, several results for graph grammars can be inherited from contextual nets.

Corollary 33 *Let* $\mathcal{O}_1$ *and* $\mathcal{O}_2$ *be occurrence grammars and let* $f : \mathcal{O}_1 \to \mathcal{O}_2$ *be a grammar morphism. Then*

(1) Morphisms preserve concurrency *(Corollary 5.2 in [8])*
If G_i *is a subgraph of* T_i *for* $i \in \{1, 2\}$, $conc(G_1)$ *and* $f_\tau(G_1, G_2)$ *implies* $conc(G_2)$. *Furthermore concurrent items cannot be identified by a morphism, i.e., for all concurrent items* $x, y \in T_1$, *if* $f_\tau(x, z)$ *and* $f_\tau(y, z)$ *for some* $z \in T_2$, *then* $x = y$.

(2) Pre-sets and contexts are concurrent *(Proposition 5.1 in [8])*
For any production $q \in P_1$, $conc({}^\bullet q \cup \underline{q})$.

(3) Morphisms properties *(Theorem 5.1 in [8])*
For any $q_1, q_1' \in P_1$ *such that* $f_P(q_1) \neq \emptyset \neq f_P(q_1')$
(a) $\lfloor f_P(q_1) \rfloor \subseteq f_P(\lfloor q_1 \rfloor)$;
(b) $(f_P(q_1) = f_P(q_1')) \ \wedge \ (q_1 \neq q_1') \quad \Rightarrow \quad q_1 \#_1 q_1'$;
(c) $f_P(q_1) \nearrow_2 f_P(q_1') \quad \Rightarrow \quad (q_1 \nearrow_1 q_1') \ \vee \ (q_1 \#_1 q_1')$.

We can also establish a tight connection between derivations in an occurrence grammar $\mathcal{O}$ and firing sequences in the corresponding occurrence contextual net $\mathsf{Net}(\mathcal{O})$. Given a marking m of $\mathsf{Net}(\mathcal{O})$, i.e., a set of items (nodes or edges) in the type graph T of $\mathcal{O}$, let us denote by $graph(m)$ the greatest subgraph of T which includes all the nodes in m and a subset of the edges in m, i.e., $graph(m)$ is obtained by viewing m as a graph, removing the dangling edges. Formally, $graph(m) = \langle N, E, s, t \rangle$ where $N = m \cap N_T$, $E = m \cap \{e \in E_T \mid \mathsf{s}(e), \mathsf{t}(e) \in N\}$

and the source and target functions are $\mathsf{s} = \mathsf{s}_{T|E}$ and $\mathsf{t} = \mathsf{t}_{T|E}$.

Lemma 34 *Let $\mathcal{O}$ be an occurrence grammar. Consider a firing sequence in the corresponding contextual net $\mathsf{Net}(\mathcal{O})$, $m_0\,[q_0\rangle\, m_1\,[q_1\rangle\, \ldots\, [q_n\rangle\, m_n$, where m_0 is the set of minimal places (edges and nodes of the start graph of $\mathcal{O}$). Then there is a derivation in $\mathcal{O}$*

$$min(\mathcal{O}) \rightarrow^{q_0} \ldots\ldots \rightarrow^{q_{n-1}} graph(m_n).$$

PROOF. The proof proceeds by induction on n.

($\mathsf{n} = \mathsf{0}$) Trivial.

($\mathsf{n} \rightarrow \mathsf{n}+\mathsf{1}$) Let $m_0\,[q_0\rangle\, m_1\,[q_1\rangle\, \ldots\, [q_n\rangle\, m_n\,[q_{n+1}\rangle\, m_{n+1}$ be a firing sequence of $\mathsf{Net}(\mathcal{O})$. By inductive hypothesis there is

$$min(\mathcal{O}) \rightarrow^{q_0} \ldots\ldots \rightarrow^{q_{n-1}} graph(m_n).$$

To deduce that the production q_{n+1} can be applied to $graph(m_n)$ we only need to show that the firing $m_n\,[q_{n+1}\rangle\, m_{n+1}$ does not use (consume or read) any edge $e \in m_n$ which does not belong $graph(m_n)$. In fact, to use (read or consume) an edge $e \in m_n$, production q_{n+1} must also use its source and target nodes $\mathsf{s}(e)$ and $\mathsf{t}(e)$, which therefore must belong to m_n: this ensures, by the above definition, that e belongs to $graph(m_n)$.

Therefore production q_{n+1} can be applied to $graph(m_n)$, and it produces a new graph, which is the structure defined as $graph(m_n) - {}^\bullet q_{n+1} \cup q_{n+1}{}^\bullet$ (componentwise), where, furthermore, all dangling edges are removed. Since $\mathsf{Net}(\mathcal{O})$ is acyclic, it is easy to conclude that this structure is exactly $graph(m_{n+1})$. □

From the above result and the fact that in an occurrence contextual net a subset of places is concurrent if and only if it is a subset of a reachable marking (Proposition 5.2 in [8]) we deduce that the following holds.

Proposition 35 *Let $\mathcal{O} = \langle T, P, \pi\rangle$ be any occurrence grammar. A subgraph of T is concurrent if and only if it is the subgraph of a reachable graph.*

Notice that, differently from Petri nets, it is not always the case that a concurrent subset of items is coverable, i.e. that it can be extended to a well-defined reachable state (graph). Given an occurrence grammar, we only know that any *well-defined* subgraph of the type graph whose items are concurrent is the subgraph of a reachable graph. For instance, consider a simple grammar having only the production q in the top of Fig. 2. The graphs are supposed to be typed over the graph T of Fig. 10. Then the set $\{A, L\}$ is concurrent,

but it is not coverable: there is no reachable graph including both the items A and L.

5.3 From occurrence contextual nets to occurrence grammars

In this section we prove that functor Net admits a left adjoint. This will be a key result for completing the chain of coreflections from the category of grammars to the category of domains.

Definition 36 (from occurrence contextual nets to grammars) *Let* $\mathsf{Gram} : \mathbf{OCN} \to \mathbf{OGGRel}$ *be the functor defined as follows.*

Let $N = \langle S, Tr, F, C\rangle$ *be an occurrence contextual net. The corresponding* SPO *occurrence graph grammar is* $\mathsf{Gram}(O) = \langle T, P, \pi\rangle$*, where:*

- $T = \langle N, E, \mathsf{s}, \mathsf{t}\rangle$ *is the type graph, with:*
  - $N = S$;
  - $E = \left\{ (s, s_1, s_2) \in S^3 \mid \begin{array}{l} |{}^\bullet s_i| \leq |{}^\bullet s| \text{ and } {}^\bullet s \neq {}^\bullet s_i \Rightarrow {}^\bullet s \subseteq \underline{s_i} \text{ for } i \in \{1,2\} \\ \underline{s} \subseteq \underline{s_1} \cap \underline{s_2}, \quad s^\bullet \subseteq (\underline{s_1} \cup s_1{}^\bullet) \cap (\underline{s_2} \cup s_2{}^\bullet) \end{array} \right\}$;
  - $\mathsf{s}(s, s_1, s_2) = s_1 \quad$ *and* $\quad \mathsf{t}(s, s_1, s_2) = s_2$.
- $P = Tr$ *is the set of production names, and for any* $t \in P$ *the corresponding production* $\pi(t) = L_t \rightarrowtail R_t$ *is defined as follows:*
  - $|L_t| = \langle {}^\bullet t \cup \underline{t}, \{(s, s_1, s_2) \mid s \in {}^\bullet t \cup \underline{t}\}, \mathsf{s}, \mathsf{t}\rangle$
  - $|R_t| = \langle t^\bullet \cup \underline{t}, \{(s, s_1, s_2) \mid s \in t^\bullet \cup \underline{t}\}, \mathsf{s}, \mathsf{t}\rangle$

  with the graphical structure inherited from T*, i.e., the source and the target of an edge* (s, s_1, s_2) *are* s_1 *and* s_2 *respectively. The typing is the inclusion and the (partial) inclusion of* L_t *in* R_t *is the obvious one.*

Given an occurrence contextual net morphism $f = \langle f_T, f_S\rangle : N_1 \to N_2$*, its image is* $\mathsf{Gram}(f) = \langle f_T, f_\tau\rangle : \mathsf{Gram}(N_1) \to \mathsf{Gram}(N_2)$ *with* $f_\tau : T_1 \times T_2$ *defined by*

- $(f_\tau)_N = f_S$;
- $(f_\tau)_E = \{((s, s_1, s_2), (s', s_1', s_2')) \mid (s, s'), (s_1, s_1'), (s_2, s_2') \in f_S\}$.

Intuitively, the image of an occurrence net N is a graph grammar where transitions become productions, and any place can be both a node and an edge connecting two nodes. More precisely, the set of nodes in the type graph T coincides with the set of places of N. An edge in the type graph is a triple $e = (s, s_1, s_2)$, which arises by viewing s as an edge connecting nodes s_1 and s_2. The presence of such edges is subject to requirements which arise from the specific features of the SPO rewriting mechanism. First, if edge e is in the start graph then also the source and target nodes must be in the start graph. Hence

the requirement $|{}^\bullet s_i| \leq |{}^\bullet s|$, from which if ${}^\bullet s = \emptyset$ then ${}^\bullet s_i = \emptyset$. Moreover, either ${}^\bullet s_i = {}^\bullet s$ or ${}^\bullet s \subseteq \underline{s_i}$, since the production which generates an edge must generate or preserve the source/target nodes. Any production which preserves an edge must also preserve its source/target nodes, hence $\underline{s} \subseteq \underline{s_1} \cap \underline{s_2}$. Any production which consumes the edge must preserve or consume the source/target nodes, hence $\underline{s} \subseteq (\underline{s_1} \cup {}^\bullet s_1) \cap (\underline{s_2} \cup {}^\bullet s_2)$.

The production $\pi(t)$ consumes (reads or produces, respectively) a node s or an edge (s, s_1, s_2) if transition t consumes (reads or produces, respectively) place s.

The following proposition states some simple, but useful properties of $\mathsf{Gram}(N)$. They are exploited, in particular, to show that Gram is a well-defined functor.

Lemma 37 (Properties of $\mathsf{Gram}(N)$) *For any occurrence contextual net N the following holds:*

(1) The structure $\mathsf{Gram}(N)$ is a well-defined SPO *grammar.*
(2) For any $s \in S$, the sets ${}^\bullet s$, $\underline{s}$ and $s^\bullet$ in $\mathsf{Gram}(N)$ are the same as ${}^\bullet s$, $\underline{s}$ and $s^\bullet$ in N. Similarly, if $e = (s, s_1, s_2)$ is an edge in the type graph of $\mathsf{Gram}(N)$ then ${}^\bullet e$, $\underline{e}$ and $e^\bullet$ in $\mathsf{Gram}(N)$ are the same as ${}^\bullet s$, $\underline{s}$ and $s^\bullet$ in N.
(3) Causality and asymmetric conflict in $\mathsf{Gram}(N)$ and in N coincide.

PROOF.

1. The fact that $\mathsf{Gram}(N)$ is a well-defined grammar is almost obvious. We only explicitly note that, by construction, for any production t in $\mathsf{Gram}(N)$ the left- and right-hand side graphs L_t and R_t are well-defined. For instance, let us prove that L_t is a well-defined graph. Let $e = (s, s_1, s_2)$ be an edge in L_t and let us show that its source s_1 and target s_2 are in L_t as well. Since $e \in L_t$, by construction we have $s \in {}^\bullet t \cup \underline{t}$, or, equivalently, $t \in \underline{s} \cup s^\bullet$. From the definition of edges, we have that $\underline{s} \subseteq \underline{s_1} \cap \underline{s_2}$ and $s^\bullet \subseteq (\underline{s_1} \cup s_1{}^\bullet) \cap (\underline{s_2} \cup s_2{}^\bullet)$. Hence $\underline{s} \cup s^\bullet \subseteq (\underline{s_1} \cup s_1{}^\bullet) \cap (\underline{s_2} \cup s_2{}^\bullet)$ and thus $t \in \underline{s_i} \cup s_i{}^\bullet$ for $i \in \{1, 2\}$. Therefore $s_1, s_2 \in L_t$. In a similar way, if $e \in R_t$ we can show that $s_1, s_2 \in R_t$, and thus R_t is a well-defined graph.

2, 3. Immediate by construction. □

Proposition 38 (Well-definedness of Gram) $\mathsf{Gram} : \mathbf{OCN} \to \mathbf{OGGRel}$ *is a well-defined functor.*

PROOF. Let N be an occurrence contextual net. The fact that $\mathsf{Gram}(N)$ satisfies properties (1)-(3) of Definition 22 easily follows by items (2) and (3) of Lemma 37.

Given an occurrence contextual net morphism $f = \langle f_T, f_S \rangle : N \to N'$, let us show that $\mathsf{Gram}(f) = \langle f_T, f_\tau \rangle : \mathsf{Gram}(N) \to \mathsf{Gram}(N')$ is a well-defined relational grammar morphism.

Let us show, for instance, that $f_\tau(|G_s|, |G_s{}'|)$. For nodes this is trivial, since Lemma 37(2) implies that the set of nodes in the start graph coincides with the initial marking of the original net and, by construction, the component of f_τ on nodes is f_S. Hence it remains to show that

- given edges $e \in |G_s|$ and $e' \in T'$ such that $f_\tau(e, e')$ we have $e' \in |G'_s|$;
- given an edge $e' \in |G'_s|$ there exists a unique $e \in |G_s|$ such that $f_\tau(e, e')$.

For the first part, let $e = (s, s_1, s_2)$ in $|G_s|$ and $e' = (s', s'_1, s'_2) \in T'$ such that $f_\tau(e, e')$. This means that $f_S(s, s')$, $f_S(s_1, s'_1)$ and $f_S(s_2, s'_2)$. Note that by construction s, s_1 and s_2 are in the initial marking of N. Thus s', s'_1 and s'_2 are in the initial marking of N' and therefore (s', s'_1, s'_2) is in the start graph of $\mathsf{Gram}(N')$.

For the second part, let $e' = (s', s'_1, s'_2)$ in $|G'_s|$. This means that s', s'_1, s'_2 are in the initial marking of N'. Hence, by Condition 1 in Definition 31, there are unique s, s_1 and s_2 in the initial marking of N such that $f_S(s, s')$, $f_S(s_1, s'_1)$ and $f_S(s_2, s'_2)$ and therefore there is a unique edge $e = (s, s_1, s_2) \in |G_s|$ such that $f_\tau(e, e')$. $\square$

We finally show that functor Gram is left-adjoint to functor Net.

Theorem 39 (adjunction between OGGRel and OCN) *The functor* Gram *is left adjoint to* Net. *The component at* N *of the unit of the adjunction* $\eta_N : N \to \mathsf{Net}(\mathsf{Gram}(N))$ *is defined as* $\eta_N = \langle id_{Tr}, \eta_S \rangle$, *where:*

$$\eta_S = \{(s, s) \mid s \in S\} \cup \{(s, (s, s_1, s_2)) \mid s \in S,\ (s, s_1, s_2) \in T_{\mathsf{Gram}(N)}\}$$

PROOF. Consider an occurrence contextual net $N = \langle S, Tr, F, C \rangle$. Then let $\mathsf{Gram}(N) = \langle T, P, \pi \rangle$ be the occurrence graph grammar as defined in Definition 36.

First observe that $\eta_N : N \to \mathsf{Net}(\mathsf{Gram}(N))$, as defined above, is a well-defined occurrence contextual net morphism. In fact, it is easy to see that, if m_N denotes the intial marking of net N (determined as the set of minimal

places with respect to $\leq_N$), then

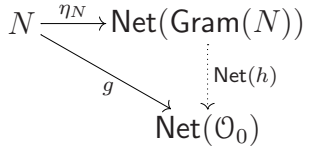
$$m_{\mathsf{Net}(\mathsf{Gram}(N))} = m \cup \{(s, s_1, s_2) \in T_{\mathsf{Gram}(N)} \mid s \in m\}.$$

Using this fact it is immediate to prove that η_N preserves the initial marking, i.e., that $\eta_S(m_N, m_{\mathsf{Net}(\mathsf{Gram}(N))})$. Similarly, for any transition t we have

$$\begin{gathered} {}^\bullet\eta_T(t) = {}^\bullet t \cup \{(s, s_1, s_2) \mid s \in {}^\bullet t\} \quad \underline{\eta_T(t)} = \underline{t} \cup \{(s, s_1, s_2) \mid s \in \underline{t}\} \\ \eta_T(t)^\bullet = t^\bullet \cup \{(s, s_1, s_2) \mid s \in t^\bullet\} \end{gathered}$$

which imply $\eta_s({}^\bullet t, {}^\bullet\eta_T(t))$, $\eta_s(\underline{t}, \underline{\eta_T(t)})$ and $\eta_s(t^\bullet, \eta_T(t)^\bullet)$, as desired.

In order to conclude, we must show the that for any occurrence grammar $\mathcal{O}_0 = \langle T_0, P_0, \pi_0 \rangle$ and for any morphism $g : N \to \mathsf{Net}(\mathcal{O}_0)$ there exists a unique morphism $h : \mathsf{Gram}(N) \to \mathcal{O}_0$, such that the following diagram commutes:



Let h be defined as follows:

- The component on production is $h_P = g_T$;
- The graph relation h_τ is
  - on nodes: $h_\tau(s, n)$ if $h_S(s, n)$
  - on edges: $h_\tau((s, s_1, s_2), e)$ if $h_S(s, n)$, $h_S(s_1, \mathsf{s}(e))$ and $h_S(s_2, \mathsf{t}(e))$.

It is easy to see that morphism h, if it exists, must be defined as above. Moreover a long but straightforward calculation, mainly based on Lemma 37, allows to show that h is actually a well-defined relational morphism for occurrence grammars. □

6 Graph processes

In the theory of Petri nets the notion of occurrence net is strictly related to that of process. A (deterministic) net process is a (deterministic) occurrence net with a suitable morphism to the original net. Similarly, in this paper, as it happens for the DPO approach [7], occurrence grammars are the basis for defining a notion of *graph process* for SPO grammars.

A *(nondeterministic) graph process* is aimed at representing in a unique "branching" structure several possible computations of a grammar. The underlying occurrence grammar makes explicit the causal structure of such com-

putations since each production can be applied at most once and each items of the type graph can be "filled" at most once. Via the morphism to the original grammar, productions and items of the type graph in the occurrence grammar can be thought of, respectively, as instances of applications of productions and instances of items generated in the original grammar by such applications. Actually, to allow for such an interpretation, some further restrictions must be imposed on the process morphism. Recall that process morphisms in Petri net theory must map places into places (rather than into multisets of places) and must be total on transitions [20]. Similarly, for graph process morphisms the left leg of the type-span is required to be an isomorphism in such a way that the type-span can be thought of simply as a graph morphism. Furthermore a process morphism cannot map a production to the empty production, a requirement corresponding to totality.

Definition 40 (strong morphism) *A grammar morphism $f : \mathcal{G}_1 \to \mathcal{G}_2$ is called* strong *if $f_T^L : X_f \to T_1$ is an isomorphism and $f_P(q_1) \neq \emptyset$, for any $q_1 \in P_1$.*

Hereafter we will always choose as concrete representative of the type-span of a strong grammar morphism f, a span f_T such that the left component f_T^L is the identity id_{T_1}.

It is not difficult to verify that, if f is a strong morphism then, by Condition 1 of the definition of grammar morphism (Definition 15), $\iota_f^s : |G_{s_2}| \to |G_{s_1}|$ is an isomorphism. Similarly, by Condition 2, for each production $q_1 \in P_1$, $\iota_f(q_1)$ is a pair of isomorphisms, namely each production q_1 of $\mathcal{G}_1$ is mapped to a production q_2 of $\mathcal{G}_2$ whose untyped components are isomorphic.

Definition 41 (graph process) *Let $\mathcal{G}$ be a graph grammar. A* graph process *of $\mathcal{G}$ is a strong grammar morphism $\chi : \mathcal{O}_\chi \to \mathcal{G}$, where $\mathcal{O}_\chi$ is an occurrence grammar. A graph process is* deterministic *if so is the underlying occurrence grammar.*

We will denote by T_χ, G_{s_χ}, P_χ and π_χ the components of the occurrence grammar O_χ underlying a process χ.

In a deterministic process, the requirement that $\nearrow^+$ is a finitary and irreflexive ensures that all the productions of $\mathcal{O}$ can be applied in a single (possibly infinite) derivation starting from $Min(\mathcal{O})$, in any order compatible with $\nearrow$. In particular, if $\mathcal{O}$ has a finite number of productions, let $Max(\mathcal{O})$ denote the graph obtained by taking the set of maximal items of $Elem(\mathcal{O})$ and removing all the edges which would be dangling. Then any derivation in $\mathcal{O}$ applying all productions leads from $Min(\mathcal{O})$ to $Max(\mathcal{O})$.

Our notion of process can be shown to be compatible with the basic theory of concurrency for the SPO approach to graph grammars and, in particular,

with the notion of shift-equivalence for derivations [18] and with the concurrent derivations of [23,35]. More precisely, starting from the above considerations, it can be shown that, working up to isomorphism, deterministic finite processes are in bijective correspondence with shift-equivalent classes of derivations. Roughly, a deterministic process χ corresponds to a full class of shift-equivalent derivations, starting from the graph $Min(\mathcal{O}_\chi)$ and ending into the graph $Max(\mathcal{O}_\chi)$, typed by the restrictions of χ_T. This result can be proved by adapting the analogous result for the DPO approach relating graph processes and derivation traces ([15,5]).

7 Unfolding construction

This section introduces the unfolding construction which, applied to an SPO grammar $\mathcal{G}$, produces a nondeterministic occurrence grammar $\mathcal{U}_s(\mathcal{G})$ describing the behaviour of $\mathcal{G}$. The unfolding is equipped with a strong grammar morphism $\varphi_\mathcal{G}$ to the original grammar, making it a nondeterministic process of $\mathcal{G}$. Then, the unfolding construction for semi-weighted graph grammars is shown to be functorial, right adjoint to the inclusion of **OGG** into **SGG**, thus establishing a coreflection between the two categories.

The idea of the unfolding construction is to begin with the start graph of the grammar, and to apply in all possible ways its productions to concurrent subgraphs, recording in the unfolding each occurrence of production and each new graph item generated in the rewriting process, both enriched with the corresponding causal history.

A basic ingredient of the unfolding construction is the *gluing* operation. It can be seen as a "partial application" of a rule to a given match, in the sense that it generates the new items as specified by the production (i.e., items of right-hand side not in the image), but items that should have been deleted are not affected: intuitively, this is because such items may still be used by another production in the nondeterministic unfolding.

Definition 42 (gluing) *Let $q = r_q : L_q \rightarrowtail R_q$ be a production, G a graph and $m : L_q \to G$ a graph morphism. We define, for any symbol $*$, the* gluing of G and R_q, according to m and marked by $*$, *denoted by $glue_*(q, m, G)$, as the graph $\langle N, E, s, t\rangle$, where:*

$$N = N_G \cup m_*(N_{R_q}) \qquad\qquad E = E_G \cup m_*(E_{R_q})$$

with m_* defined by:

$$m_*(x) = \begin{cases} m(x) \text{ if } x \in dom(r_q); \\ \langle x, * \rangle \text{ otherwise.} \end{cases}$$

The source and target functions and the typing are inherited from G and R_q.

The gluing operation keeps unchanged the identity of the items already in G, and records in each newly added item from R_q the given symbol $*$. Notice that (assuming that symbol $*$ is "fresh") the gluing, as just defined, is a concrete definition of a pushout object of the arrows $G \overset{m}{\leftarrow} L_q \overset{r_q\leftarrow}{\hookleftarrow} dom(r_q)$ and $dom(r_q) \overset{r_q}{\hookrightarrow} R_q$.

As described below, the unfolding of a grammar is obtained as the limit of a chain of occurrence grammars, each approximating the unfolding up to a certain causal depth.

Definition 43 (depth) *Let $\mathcal{O} = \langle T, P, \pi \rangle$ be an occurrence grammar. The function depth : $Elem(\mathcal{O}) \to \mathbb{N}$ is defined inductively as follows:*

$$\begin{aligned} &depth(x) = 0 && \text{for } x \in |G_s| = Min(\mathcal{O}); \\ &depth(q) = \max\{depth(x) \mid x \in {}^\bullet q \cup \underline{q}\} + 1 && \text{for } q \in P; \\ &depth(x) = depth(q) && \text{for } x \in q^\bullet. \end{aligned}$$

It is not difficult to prove that *depth* is a well-defined total function, since infinite descending chains of causality are disallowed in occurrence grammars. Moreover, given an occurrence grammar $\mathcal{O}$, the grammar containing only the items of *depth* less than or equal to n, denoted by $\mathcal{O}^{[n]}$, is a well-defined occurrence grammar.

As expected, an occurrence grammar $\mathcal{O}$ is the (componentwise) union of its subgrammars $\mathcal{O}^{[n]}$, of depth n, for all n. Moreover it is not difficult to see that if $g : \mathcal{O} \to \mathcal{G}$ is a grammar morphism, then for any $n \in \mathbb{N}$, g restricts to a morphism $g^{[n]} : \mathcal{O}^{[n]} \to \mathcal{G}$. In particular, if $T^{[n]}$ denotes the type graph of $\mathcal{O}^{[n]}$, then the type-span of $g^{[n]}$ will be the equivalence class of

$$T^{[n]} \xleftarrow{g_T^{L\,[n]}} X^{[n]} \xrightarrow{g_T^{R\,[n]}} T_{\mathcal{G}}$$

where $X^{[n]} = \{x \in X_g \mid g_T^L(x) \in T^{[n]}\}$. Vice versa each morphism $g : \mathcal{O} \to \mathcal{G}$ is uniquely determined by its truncations at finite depths.

The unfolding of a graph grammar is thus obtained as the limit of a chain of occurrence grammars, each approximating the unfolding up to a certain

causal depth.

Definition 44 (unfolding) *Let $\mathcal{G} = \langle T, G_s, P, \pi \rangle$ be a semi-weighted graph grammar. We inductively define, for each n, an occurrence grammar $\mathcal{U}_s(\mathcal{G})^{[n]} = \langle T^{[n]}, P^{[n]}, \pi^{[n]} \rangle$ and a pair of mappings $\varphi^{[n]} = \langle {\varphi_T}^{[n]} : T^{[n]} \to T, {\varphi_P}^{[n]} : P^{[n]} \to P \rangle$. Then the* unfolding $\mathcal{U}_s(\mathcal{G})$ *and the* folding morphism $\varphi_{\mathcal{G}} : \mathcal{U}_s(\mathcal{G}) \to \mathcal{G}$ *are the occurrence grammar and strong grammar morphism defined as the componentwise union of $\mathcal{U}_s(\mathcal{G})^{[n]}$ and $\varphi^{[n]}$, respectively.*

Since each morphism $\varphi^{[n]}$ is strong, assuming that the left component of the type-span ${\varphi_T}^{[n]}$ is the identity on $T^{[n]}$ we only need to define the right component $\varphi_T^{R[n]} : T^{[n]} \to T$, which, by the way, makes $\langle T^{[n]}, \varphi_T^{R[n]} \rangle$ a T-typed graph.

($\mathbf{n = 0}$) *The components of the grammar $\mathcal{U}_s(\mathcal{G})^{[0]}$ are $T^{[0]} = |G_s|$, $P^{[0]} = \pi^{[0]} = \emptyset$. Morphism $\varphi^{[0]} : \mathcal{U}_s(\mathcal{G})^{[0]} \to \mathcal{G}$ is defined by $\varphi_T^{R[0]} = t_{G_s}$, ${\varphi_P}^{[0]} = \emptyset$, and ${\iota^{[0]}}^s = id_{|G_s|}$.*

($\mathbf{n \to n+1}$) *The occurrence grammar $\mathcal{U}_s(\mathcal{G})^{[n+1]}$ is obtained by extending $\mathcal{U}_s(\mathcal{G})^{[n]}$ with all the possible production applications to concurrent subgraphs of its type graph. More precisely, let $M^{[n]}$ be the set of pairs $\langle q, m \rangle$ such that $q \in P$ is a production in $\mathcal{G}$, $m : L_q \to \langle T^{[n]}, \varphi_T^{R[n]} \rangle$ is an injective match and $m(|L_q|)$ is a* concurrent *subgraph of $T^{[n]}$. Then $\mathcal{U}_s(\mathcal{G})^{[n+1]}$ is the occurrence grammar resulting after performing the following steps for each $\langle q, m \rangle \in M^{[n]}$.*

- *Add to $P^{[n]}$ the pair $\langle q, m \rangle$ as a new production name and extend ${\varphi_P}^{[n]}$ so that ${\varphi_P}^{[n]}(\langle q, m \rangle) = q$. Intuitively, $\langle q, m \rangle$ represents an occurrence of q, where the match m is needed to record the "history".*
- *Extend the type graph $T^{[n]}$ by adding to it a copy of each item generated by the application q, marked by $\langle q, m \rangle$ (in order to keep trace of the history). The morphism $\varphi_T^{R[n]}$ is extended consequently. Formally, the T-typed graph $\langle T^{[n]}, \varphi_T^{R[n]} \rangle$ is replaced by $glue_{\langle q,m \rangle}(q, m, \langle T^{[n]}, \varphi_T^{R[n]} \rangle)$.*
- *The production $\pi^{[n]}(\langle q, m \rangle)$ has the same untyped components of $\pi(q)$ and the morphisms $\iota^{[n]}(\langle q, m \rangle)$ are identities, that is $\iota(\langle q, m \rangle) = \langle id_{|L_q|}, id_{|R_q|} \rangle$. The typing of the left-hand side is determined by m, and each item x in $|R_q| - r_q(|dom(r_q)|)$ is typed over the corresponding new item $\langle x, \langle q, m \rangle \rangle$ of the type graph.*

It is not difficult to verify that for each n, $\mathcal{U}_s(\mathcal{G})^{[n]}$ is a (finite depth) occurrence grammar, and $\mathcal{U}_s(\mathcal{G})^{[n]} \subseteq \mathcal{U}_s(\mathcal{G})^{[n+1]}$, componentwise. Therefore $\mathcal{U}_s(\mathcal{G})$ is a well-defined occurrence grammar. Similarly for each $n \in \mathbb{N}$ we have that $\varphi^{[n]}$ is a well-defined morphism from $\mathcal{U}_s(\mathcal{G})^{[n]}$ to $\mathcal{G}$ and $\varphi^{[n]}$ coincides with the

restriction of $\varphi^{[n+1]}$ to $\mathcal{U}_s(\mathcal{G})^{[n]}$. This induces a unique morphism $\varphi_{\mathcal{G}} : \mathcal{U}_s(\mathcal{G}) \to \mathcal{G}$.

The deterministic gluing construction ensures that, at each step, the order in which productions are applied does not influence the final result of the step. Moreover if a production is applied twice at the same match (even if in different steps), the generated items are identical and thus they appear only once in the unfolding.

It is possible to show that the unfolding construction applied to an occurrence grammar yields a grammar which is isomorphic to the original one. For instance, the unfolding of grammar $\mathcal{G}$ in Fig. 10 is (up to isomorphism) the grammar $\mathcal{G}$ itself.

By unfolding the running example grammar $\mathcal{SR}$ of Fig. 3 we obtain the (infinite) occurrence grammar which is partially represented in Fig. 11. The folding morphism $f : \mathcal{U}_s(\mathcal{SR}) \to \mathcal{SR}$ is defined as follows:

- f_P maps each production of the kind $name_i$ or $name_{i,j}$ in $\mathcal{U}_s(\mathcal{SR})$ to production $name$ in $\mathcal{SR}$;
- the span f_T relates any graph item $name_i$ or $name_{i,j}$ in $\mathcal{U}_s(\mathcal{SR})$ to the item $name$ in $\mathcal{SR}$.

The unfolding construction has been defined, up to now, only at the "object level". We next face the problem of characterising the unfolding as a coreflection between suitable categories of graph grammars and of occurrence grammars. More specifically the unfolding construction is extended to a functor $\mathcal{U}_s : \mathbf{SGG} \to \mathbf{OGG}$ that is right adjoint to the inclusion functor $\mathcal{I}_s : \mathbf{OGG} \to \mathbf{SGG}$.

The restriction to semi-weighted graph grammars is essential for the above categorical result when one uses general morphisms. However, in Section 9.1 we will see how, suitably restricting graph grammar morphisms to still interesting subclasses (including, for instance, the morphisms of [21]) it is possible to restore the characterisation of the unfolding as a universal construction for general, possibly non semi-weighted, grammars.

Occurrence grammars are safe, and therefore *a fortiori* semi-weighted grammars. Thus there exists an inclusion functor $\mathcal{I}_s : \mathbf{OGG} \to \mathbf{SGG}$. The next theorem shows that the unfolding of a grammar $\mathcal{U}_s(\mathcal{G})$ and the folding morphism $\varphi_{\mathcal{G}}$ are cofree over $\mathcal{G}$. Therefore $\mathcal{U}_s$ extends to a functor that is right adjoint of $\mathcal{I}_s$ and thus establishes a coreflection between **SGG** and **OGG**.

Theorem 45 (coreflection between SGG and OGG) *Let $\mathcal{G}$ be a semi-weighted grammar, let $\mathcal{U}_s(\mathcal{G})$ be its unfolding and let $\varphi : \mathcal{U}_s(\mathcal{G}) \to \mathcal{G}$ be the*

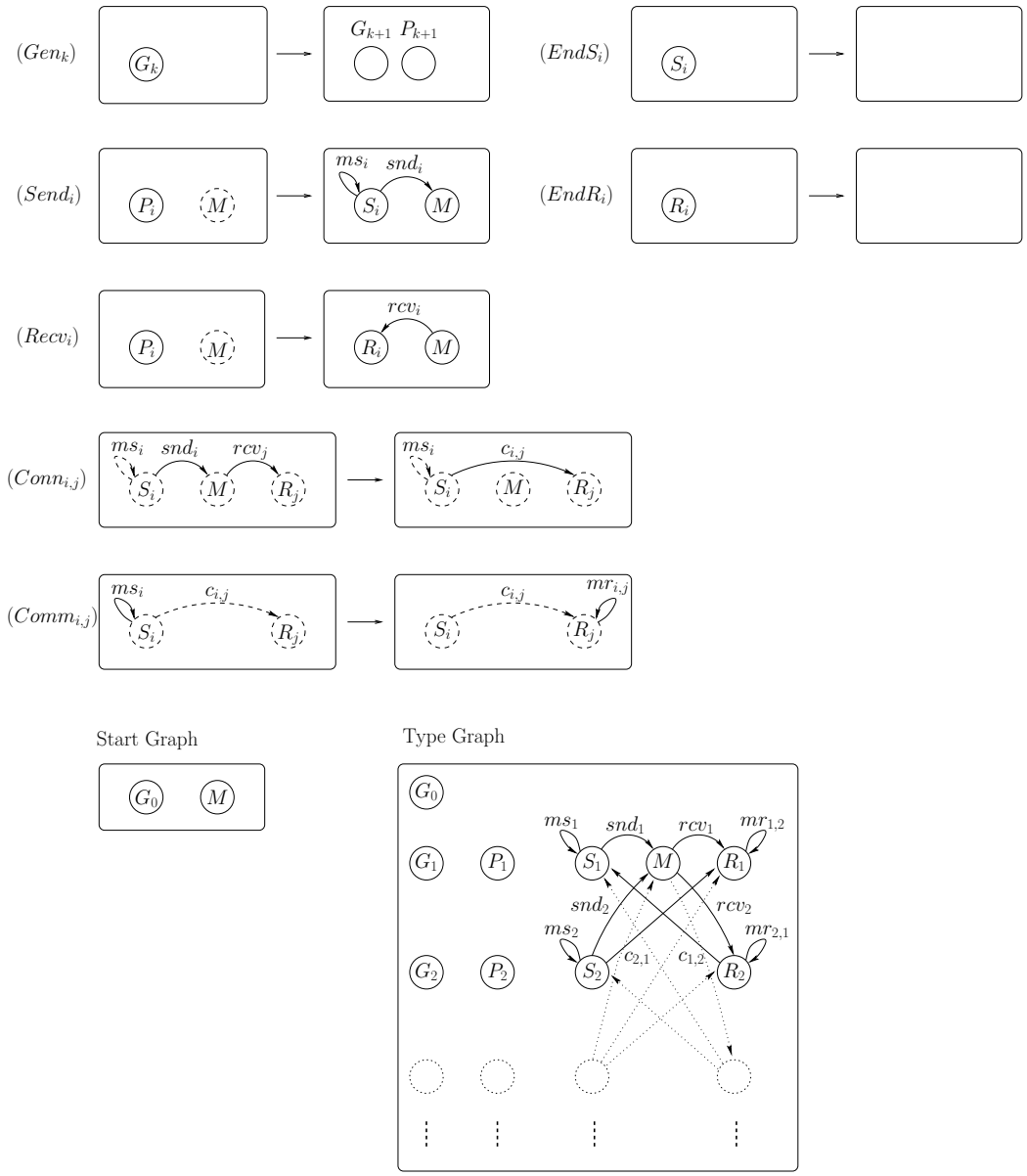
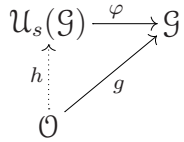


Fig. 11. Unfolding of the grammar $\mathcal{SR}$ in Fig. 3 ($k \geq 0$ and $i, j > 0$).

folding morphism as in Definition 44. Then for any occurrence grammar $\mathcal{O}$ *and for any morphism* $g : \mathcal{O} \to \mathcal{G}$ *there exists a unique morphism* $h : \mathcal{O} \to \mathcal{U}_s(\mathcal{G})$ *such that the following diagram commutes:*

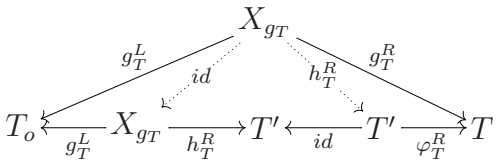


Therefore $\mathcal{I}_s \dashv \mathcal{U}_s$.

PROOF. To avoid a cumbersome notation, let us fix the names of the components of the various grammars. Let $\mathcal{G} = \langle T, G_s, P, \pi \rangle$, $\mathcal{U}_s(\mathcal{G}) = \langle T', G'_s, P', \pi' \rangle$, and $\mathcal{O} = \langle T_o, G_{s_o}, P_o, \pi_o \rangle$.

According to Definition 15, a morphism $h : \mathcal{O} \to \mathcal{U}_s(\mathcal{G})$ is determined by a semi-abstract span $[h_T] : T_o \to T'$, a function $h_P : P_o \to P'$, and a family of morphisms $\iota_h = \{\iota_h(q_o) \mid q_o \in P_o\} \cup \{\iota^s_h\}$ satisfying suitable requirements.

As a first step, we show that both the left component of $[h_T]$ and the family ι_h are uniquely determined by the condition $\varphi \circ h = g$ and by the properties of the folding morphism φ. In fact, let in the following diagram $\langle g^L_T, X_{g_T}, g^R_T \rangle : T_o \to T$ be an arbitrary but fixed representative of $[g_T]$, and let $\langle id, T', \varphi^R_T \rangle : T' \to T$ be a representative of $[\varphi_T]$ (where, since φ is strong, we can choose the identity of T' as left component).

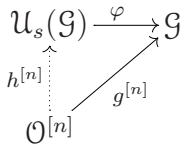


Then it is easily shown that for any semi-abstract span $[h_T] : T_o \to T'$ such that $[\varphi_T] \circ [h_T] = [g_T]$ we can choose a representative of the form $\langle g^L_T, X_{g_T}, h^R_T \rangle$ for some h^R_T, because the inner square becomes a pullback. This shows that, without any loss of generality, we can assume that the left components of $[h_T]$ and $[g_T]$ coincide.

As far as the family of morphisms ι_h is concerned, recall that morphism $\iota^s_\varphi : |G_s| \to |G'_s|$ is the identity by Definition 44; thus, since $\varphi \circ h = g$ implies $\iota^s_h \circ \iota^s_\varphi = \iota^s_g$, we deduce that $\iota^s_h = \iota^s_g$. The same holds for the components of $\iota_h(q_o)$ for any production $q_o \in P_o$, by observing that $\iota_h(q_o) \circ \iota_\varphi(h_P(q_o)) = \iota_g(q_o)$ must hold, and that $\iota_\varphi(q')$ is a pair of identities for each $q' \in P'$.

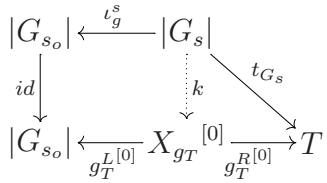
Existence

We will show inductively that for each $n \in \mathbb{N}$ we can find a morphism $h^{[n]}$ such that the diagram

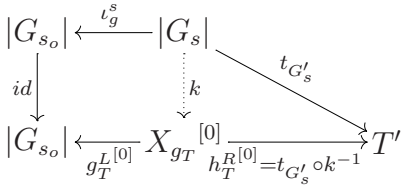


commutes, and such that $h^{[n+1]}$ extends $h^{[n]}$. Then the morphism h we are looking for will be the componentwise union of the chain of morphisms $\{h^{[n]}\}_{n \in \mathbb{N}}$.

($\mathbf{k = 0}$) By definition, the occurrence grammar $\mathcal{O}^{[0]}$ consists of the start graph of $\mathcal{O}$ only, typed identically on the type graph, with no productions, i.e., $\mathcal{O}^{[0]} = \langle |G_{s_o}|, \emptyset, \emptyset \rangle$. By the considerations above, to determine morphism $h^{[0]} : \mathcal{O}^{[0]} \to \mathcal{U}_s(\mathcal{G})$ we only have to provide the right component $h_T^{R[0]} : {X_{g_T}}^{[0]} \to T'$ of $[{h_T}^{[0]}]$. Moreover, to be a well defined grammar morphism, $h^{[0]}$ must preserve the start graph. By condition (1) of Definition 15 applied to $g^{[0]} : \mathcal{O}^{[0]} \to \mathcal{G}$, there is a morphism $k : |G_s| \to {X_{g_T}}^{[0]}$ such that the diagram below commutes, and the square is a pullback. Furthermore, by the pullback properties k is an isomorphism, and, since $\mathcal{G}$ is a semi-weighted grammar, by Lemma 24, k is uniquely determined.

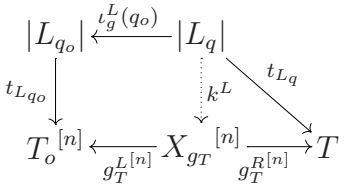


Now we define $h_T^{R[0]} = t_{G'_s} \circ k^{-1}$, completing the definition of $h^{[0]}$. The next diagram shows that $h^{[0]}$ satisfies the requirement of preservation of the start graph. The fact that $g^{[0]} = \varphi \circ h^{[0]}$ easily follows by construction.

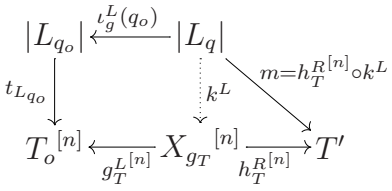


($\mathbf{n \to n{+}1}$) We have to define morphism $h^{[n+1]} : \mathcal{O}^{[n+1]} \to \mathcal{U}_s(\mathcal{G})$ by extending $h^{[n]}$ to the items of $T_o \cup P_o$ of depth equal to $n{+}1$. Without any loss of generality we assume that there is just one production q_o in $\mathcal{O}^{[n+1]}$ with $depth(q_o) = n{+}1$ (the general case can be carried out in a completely analogous way). To ensure $\varphi_P \circ {h_P}^{[n+1]} = {g_P}^{[n+1]}$, the production q_o must be mapped to a production q' in $\mathcal{U}_s(\mathcal{G})$, which is an occurrence of the production $q = g_P(q_o)$ of $\mathcal{G}$. In other words, q' will be $\langle q, m \rangle$, with $m : L_q \to \langle T', \varphi_T^R \rangle$ a match satisfying suitable conditions.

The defining conditions of grammar morphisms, applied to $g^{[n]} : \mathcal{O}^{[n]} \to \mathcal{G}$, ensure the existence of a morphism k^L, such that the diagram below commutes, where the square is a pullback.



Moreover, since $\mathcal{G}$ is a semi-weighted grammar, by Lemma 24, g_T is relational and thus the arrow k^L is uniquely determined. By Definition 43, $depth(x) \leq n$ for all $x \in t_{L_{q_o}}(|L_{q_o}|) = {}^\bullet q_o \cup \underline{q_o}$, and thus $h^{[n]}$ is defined on the pre-set and on the context of q_o. Therefore we can construct the following diagram.



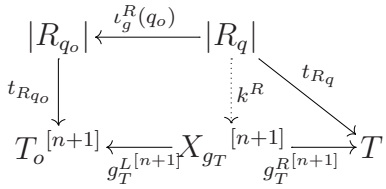
Notice that $m = h_T^{R[n]} \circ k^L$ can be seen as a T-typed graph morphism from L_q to $\langle T', \varphi_T^R \rangle$. In fact, it satisfies $\varphi_T^R \circ m = \varphi_T^R \circ h_T^{R[n]} \circ k^L = g_T^{R[n]} \circ k^L = t_{L_q}$. Moreover, recalling that $h_T{}^{[n]} = \langle g_T^{L[n]}, h_T^{R[n]} \rangle$, by the diagram above we have that $h_T{}^{[n]}(L_{q_o}, \langle |L_q|, m \rangle)$. Since by definition of occurrence grammar $t_{L_{q_o}}(|L_{q_o}|)$ is a concurrent subgraph of T_o, by Corollary 33.(1), we can conclude that $m(|L_q|)$ is a concurrent subgraph of T'. Let us prove that, in addition, the mapping m is a well-defined injective match. First observe that for $x, y \in |L_q|$

$$m(x) = m(y) \quad \Rightarrow \quad k^L(x) = k^L(y). \tag{$\dagger$}$$

In fact, assume that $m(x) = m(y)$, let $x' = k^L(x)$ and $y' = k^L(y)$ and suppose $x' \neq y'$. From the fact that $m(x) = m(y)$ we deduce $h_T^{R[n]}(x') = h_T^{R[n]}(y')$, and therefore, since $h^{[n]}$ is relational, $g_T^{L[n]}(x') \neq g_T^{L[n]}(y')$. Now observe that, by commutativity of the square in the diagram above, $g_T^{L[n]}(x')$, $g_T^{L[n]}(y') \in t_{L_{q_o}}(|L_{q_o}|)$ and moreover $h_T{}^{[n]}(g_T^{L[n]}(x'), z)$, $h_T{}^{[n]}(g_T^{L[n]}(y'), z)$, where $z = m(x) = m(y)$. But according to Corollary 33.(1) this would imply that $t_{L_{q_o}}(|L_{q_o}|)$ is not concurrent, contradicting the definition of occurrence grammar. Hence, as desired, it must be $k^L(x) = k^L(y)$. Now, by definition of occurrence grammar, the typing t_{L_q} is injective and thus, by ($\dagger$), we conclude that m is injective, as desired.

Since $m : L_q \to \langle T', \varphi_T^R \rangle$ is an injective match and $m(|L_q|)$ is concurrent, by definition of unfolding $q' = \langle q, m \rangle$ is a production name in P'. Then the production component $h_P{}^{[n+1]}$ of the morphism $h^{[n]}$ can be defined by extending $h_P{}^{[n]}$ with $h_P{}^{[n+1]}(q_o) = q'$. The diagram above shows that, with this extension, the left-hand side of the production is preserved. Now, it can be seen that there is a unique way of extending the type-span $h_T{}^{[n]}$ to take into account also the right-hand side of production q'. In fact, consider the diagram below

expressing, for morphism $g^{[n+1]}$, the preservation of the right-hand side of q_o.



To complete the definition of $h^{[n]}$ we must define the right component $h_T^{R[n+1]} : X_{g_T}{}^{[n+1]} \to T'$, extending $h_T^{R[n]}$ on the items which are in $X = X_{g_T}{}^{[n+1]} - X_{g_T}{}^{[n]}$. Now one can verify that k^R establishes an isomorphism between X and $|R_q| - r_q(dom(r_q))$. Then the condition requiring that $h_T{}^{[n+1]}$ preserves the right-hand side of q_o forces us to define, for each $x \in X$, $h_T^{R[n+1]}(x) = t_{L_{q'}}(k^{R^{-1}}(x))$.

The fact that $g^{[n]} = \varphi \circ h^{[n]}$ easily follows by construction.

Uniqueness
Uniqueness follows from the fact that at each step we are forced to define the morphism h as we have done to ensure commutativity. □

8 Event structure semantics for SPO graph grammars

In this section, after reviewing the basics of asymmetric event structures, we discuss how, combining the adjunction between occurrence graph grammars and occurrence nets and the results proved for contextual nets in [8], we can obtain a coreflective asymmetric event structure semantics for SPO graph grammars.

8.1 Asymmetric event structures

Asymmetric event structures [8] are a generalisation of prime event structures where the conflict relation is allowed to be non-symmetric. As already mentioned, this is needed to give a faithful representation of dependencies between events in formalisms such as string, term, graph rewriting and contextual nets, where a rule may preserve a part of the state, in the sense that part of the state is necessary for applying the rule, but it is not affected by the application. In this setting the symmetric binary conflict is no longer a primitive relation, but it is represented via "cycles" of asymmetric conflict. As a consequence, PES's can be identified with a special subclass of asymmetric event structures, namely those where all conflicts are actually symmetric.

We next review some basics of asymmetric event structures. For a wider treatment we refer the reader to [8]. For technical reasons we first introduce asymmetric pre-event structures. Then asymmetric event structures will be defined as special asymmetric pre-event structures satisfying a suitable condition of "saturation".

Definition 46 (asymmetric event structure) *A* asymmetric pre-event structure (pre-AES) *is a tuple* $\mathcal{A} = \langle \mathcal{E}, \leq, \nearrow \rangle$*, where* $\mathcal{E}$ *is a set of* events *and* $\leq$, $\nearrow$ *are binary relations on* $\mathcal{E}$ *called* causality *and* asymmetric conflict*, respectively, such that:*

(1) $\leq$ *is a partial order and* $\lfloor \epsilon \rfloor = \{\epsilon' \in \mathcal{E} \mid \epsilon' \leq \epsilon\}$ *is finite for all* $\epsilon \in \mathcal{E}$*;*
(2) $\nearrow$ *satisfies, for all* $\epsilon, \epsilon' \in \mathcal{E}$*:*
(a) $\epsilon < \epsilon' \quad \Rightarrow \quad \epsilon \nearrow \epsilon'$*,* *(b)* $\nearrow$ *is acyclic in* $\lfloor \epsilon \rfloor$*,*

where, as usual, $\epsilon < \epsilon'$ *means* $\epsilon \leq \epsilon'$ *and* $\epsilon \neq \epsilon'$*.*

An asymmetric event structure *(*AES*) is a pre-*AES *which satisfies:*

(3) for any $\epsilon, \epsilon' \in \mathcal{E}$*, if* $\nearrow$ *is cyclic in* $\lfloor \epsilon \rfloor \cup \lfloor \epsilon' \rfloor$ *then* $\epsilon \nearrow \epsilon'$*.*

The asymmetric conflict relation $\nearrow$ determines an order of execution locally to each computation: if $\epsilon \nearrow \epsilon'$ and ϵ, ϵ' occur in the same computation then ϵ must precede ϵ'. Therefore a set of events $\epsilon_1 \nearrow \epsilon_2 \nearrow \ldots \nearrow \epsilon_n \nearrow \epsilon_1$ forming a cycle of asymmetric conflict can never occur in the same computation, a fact that can be naturally interpreted as a kind of conflict over sets of events. Condition (3) above ensures that, in an AES, this kind of conflict is inherited through causality, a typical property also of PES's.

Any pre-AES can be "saturated" to produce an AES.

Definition 47 (saturation) *Given a pre-*AES $\mathcal{A} = \langle \mathcal{E}, \leq, \nearrow \rangle$*, its* saturation*, denoted by* $\overline{\mathcal{A}}$*, is the* AES $\langle \mathcal{E}, \leq, \nearrow' \rangle$*, where* $\nearrow'$ *is defined as* $\epsilon \nearrow' \epsilon'$ *if* $(\epsilon \nearrow \epsilon')$ *or* $\nearrow$ *is cyclic in* $\lfloor \epsilon \rfloor \cup \lfloor \epsilon' \rfloor$*.*

It is immediate to prove that for any $\mathcal{A}$ its saturation $\overline{\mathcal{A}}$ is a well-defined AES.

Definition 48 (category of AES's) *Let* $\mathcal{A}_0$ *and* $\mathcal{A}_1$ *be two* AES*'s. An* AES-morphism $f : \mathcal{A}_0 \to \mathcal{A}_1$ *is a partial function* $f : \mathcal{E}_0 \rightarrowtail \mathcal{E}_1$ *such that, for all* $\epsilon_0, \epsilon_0' \in \mathcal{E}_0$*, assuming that* $f(\epsilon_0)$ *and* $f(\epsilon_0')$ *are defined,*

(1) $\lfloor f(\epsilon_0) \rfloor \subseteq f(\lfloor \epsilon_0 \rfloor)$*;*
(2) (a) $f(\epsilon_0) \nearrow_1 f(\epsilon_0') \quad \Rightarrow \quad \epsilon_0 \nearrow_0 \epsilon_0'$*;*
(b) $(f(\epsilon_0) = f(\epsilon_0')) \wedge (\epsilon_0 \neq \epsilon_0') \quad \Rightarrow \quad \epsilon_0 \nearrow_0 \epsilon_0'$*.*

We denote by **AES** *the category having asymmetric event structures as objects and* AES*-morphisms as arrows.*

The notion of configuration extends smoothly from PES's to AES's. A configuration is a subset C of events, closed under causality (i.e., $\lfloor \epsilon \rfloor \subseteq C$ for any $\epsilon \in C$) such that $\nearrow^+$ is well-founded and finitary in C (well-foundedness of asymmetric conflict implies its acyclicity which, in turn, corresponds to the absence of conflicts). The main novelty is the fact that the computational order between configurations is not simply set-inclusion. In fact, a configuration C can be extended with an event ϵ' only if for any event $\epsilon \in C$, it does not hold that $\epsilon' \nearrow \epsilon$ (since, in this case, ϵ would prevent the execution of ϵ'). The set of configurations of an AES with such a computational order is a finitary coherent prime algebraic domain (*domain*, for short). The corresponding functor from **AES** to **Dom**, the category of domains, has a left adjoint which maps each domain to the corresponding PES (each PES can be seen as a special AES where conflict is symmetric). Hence Winskel's equivalence between **PES**, the category of prime event structures, and **Dom** generalises to a coreflection between **AES** and **Dom**.

$$\mathbf{AES} \underset{\mathcal{L}_a}{\overset{\mathcal{P}_a}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{Dom}$$

As mentioned in the introduction, AES's have been introduced to provide a coreflective concurrent semantics for contextual nets. In particular, the paper [8] establishes a coreflection between the category of occurrence contextual nets and **AES**

$$\mathbf{OCN} \underset{\mathcal{E}_a}{\overset{\mathcal{N}_a}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{AES}$$

The functor $\mathcal{E}_a$ maps any occurrence contextual net N to the AES obtained by saturating the pre-AES consisting of the set of transitions, endowed with causality and asymmetric conflict. Given an occurrence net morphism $f = \langle f_T, f_S \rangle : N \to N'$ we have $\mathcal{E}_a(f) = f_T$. The component at an AES $\mathcal{A}$ of the unit $\kappa : 1 \to \mathcal{E}_a \circ \mathcal{N}_a$ is the identity.

8.2 Occurrence grammars and AES's

The adjunction between **OGGRel** and **OCN** in Section 5, can be composed with the coreflection between **OCN** and **AES** mentioned above, thus leading to an adjunction between **OGGRel** and **AES**. Let us denote by $\mathcal{E}_s : \mathbf{OGGRel} \to \mathbf{AES}$ the functor defined as $\mathcal{E}_a \circ \mathsf{Net}$ and by $\mathcal{N}_s : \mathbf{AES} \to \mathbf{OGGRel}$ the functor defined as $\mathsf{Gram} \circ \mathcal{N}_a$. According to this definition, given an occurrence grammar $\mathcal{O}$, the corresponding AES is obtained by saturating the pre-AES consisting of the set of production names of $\mathcal{O}$, endowed with the relation of causality and asymmetric conflict as defined in Definitions 19 and 20.

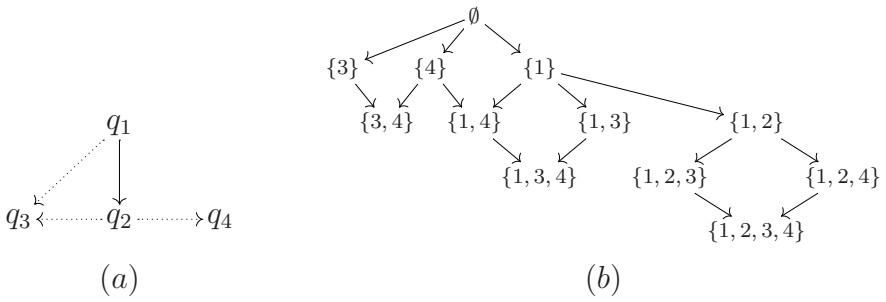


Fig. 12. The (a) AES and (b) domain of configurations for $\mathcal{G}$ of Fig. 10.

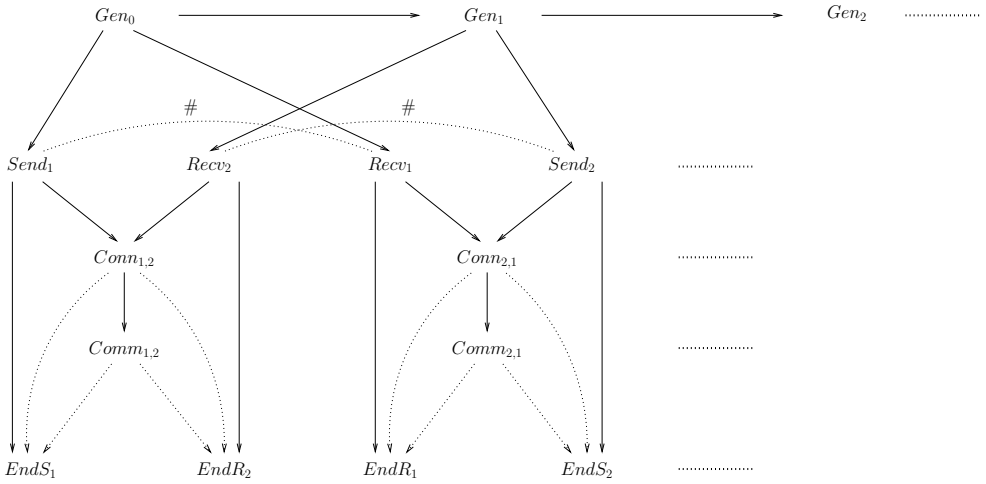


Fig. 13. Event structure of the example grammar.

For instance, Fig. 12 shows the AES (and the domain of its configurations) associated to the occurrence grammar $\mathcal{G}$ in Fig. 10. In the AES straight and dotted arrows represent causality and asymmetric conflict, respectively. In any configuration the event corresponding to q_i is written as "i".

The AES corresponding to grammar $\mathcal{SR}$ in Fig. 3 can be found in Fig. 13. Note that any *Send* and *Recv* event is caused by the *Gen* event generating the corresponding process. A process cannot be a sender and a receiver at the same time, hence the corresponding events are in conflict. A *Conn* event requires (and thus is caused by) a *Send* and a *Recv* events, which must be performed by different processes. Any *Conn* event is a cause for a *Comm* event, which establishes the communication. Observe that *Conn* and *Comm* events can be prevented by the execution of events *EndS* and *EndR* (formally, because the latter events delete the process node which is preserved by the former). Hence events *EndS* and *EndR* are in asymmetric conflict with events *Conn* and *Comm*.

The adjunction between **OGGRel** and **AES** is actually a coreflection, as

expressed by the following corollary.

Corollary 49 (coreflection between OGGRel and AES) *The functor* $\mathcal{N}_s = \mathsf{Gram} \circ \mathcal{N}_a\mathsf{Gram} \circ \mathcal{N}_a : \mathbf{AES} \to \mathbf{OGGRel}$ *is left-adjoint to* $\mathcal{E}_s = \mathcal{E}_a \circ \mathsf{Net} : \mathbf{OGGRel} \to \mathbf{AES}$ *and they establish a coreflection between* **OGGRel** *and* **AES**.

PROOF. The fact that $\mathcal{N}_s$ is left-adjoint to $\mathcal{E}_s$ follows from the fact that the composition of left-adjoint functors is a left adjoint.

Furthermore, let $\kappa : 1 \dot{\to} \mathcal{E}_a \circ \mathcal{N}_a$ be the unit of the coreflection between **OCN** and **AES** in [8] (which is the identity), and let $\eta : 1 \dot{\to} \mathsf{Net} \circ \mathsf{Gram}$ be the unit of the adjunction **OGGRel** and **OCN**, as defined in Section 5. Then the unit of the adjunction $\mathcal{N}_s \dashv \mathcal{E}_s$ at an AES $\mathcal{A}$ turns out to be $\mathcal{E}_a(\eta_{\mathcal{N}_a(\mathcal{A})}) \circ \kappa_{\mathcal{A}}$, which is the identity. Therefore the adjunction is a coreflection. □

9 Unfolding semantics of other classes of grammars

In this section we present some possible extensions of the work in this paper, first to general, non-semi-weighted SPO graph grammars, next to SPO grammars with possibly non-injective matches, and finally to graph grammars in the DPO approach. We discuss which results can be generalised to each of these three classes of grammars, and under which assumptions.

9.1 Unfolding semantics of general SPO grammars

A natural question regards the possibility of extending the results in this paper to the full category **GG** of SPO graph grammars. Here we show that considering general, possibly non semi-weighted, graph grammars the result characterising the unfolding as a coreflection fails. However considering a restricted, still meaningful, subclass of grammar morphisms, the construction in the paper can be easily adapted in order to provide a functorial concurrent semantics for the full class of SPO graph grammars. In this setting, the unfolding can be again characterised as a coreflection, while, unfortunately the adjunction with domains is lost.

First, we notice that in the characterisation of the unfolding as a coreflection (Theorem 45) the restriction to semi-weighted grammars plays a basic role. In fact, in the proof of such theorem, the uniqueness of morphism h relies on Lemma 24 which in turn requires the grammar $\mathcal{G}$ to be semi-weighted. Unfortunately the problem does not reside in our proof technique: the cofreeness of

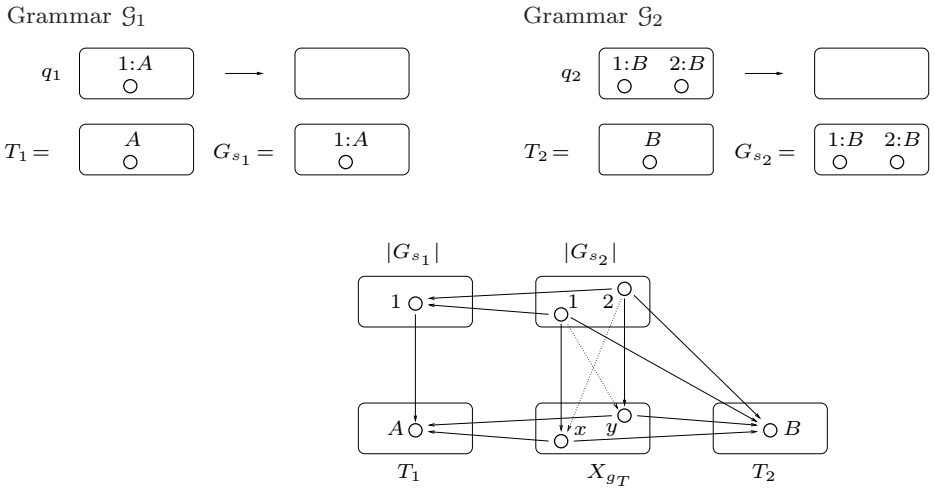


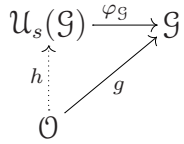
Fig. 14. The grammars $\mathcal{G}_1$ and $\mathcal{G}_2$, and the pullback-retyping diagram for their start graphs.

the unfolding of $\mathcal{U}_s(\mathcal{G})$ and of the folding morphism $\varphi_{\mathcal{G}}$ over $\mathcal{G}$ may really fail to hold if the grammar $\mathcal{G}$ is not semi-weighted.

For instance, consider grammars $\mathcal{G}_1$ and $\mathcal{G}_2$ in Fig. 14, where typed graphs are represented by decorating their items with pairs "concrete identity:type". The grammar $\mathcal{G}_2$ is not semi-weighted since the start graph is not injectively typed, while $\mathcal{G}_1$ is clearly an occurrence grammar. The unfolding $\mathcal{U}_s(\mathcal{G}_2)$ of the grammar $\mathcal{G}_2$, according to Definition 44, is defined as follows. The start graph and type graph of $\mathcal{U}_s(\mathcal{G}_2)$ coincide with $|G_{s_2}|$. Furthermore, $\mathcal{U}_s(\mathcal{G}_2)$ contains two productions $q_2' = \langle q_2, m' \rangle$ and $q_2'' = \langle q_2, m'' \rangle$, which are two occurrences of q_2 corresponding to the two possible different matches $m', m'' : L_{q_2} \to G_{s_2}$ (the identity and the swap).

Observe that there exists a morphism $g : \mathcal{G}_1 \to \mathcal{G}_2$ which is not relational, i.e., the property in Lemma 24 fails to hold. The component g_P on productions is defined by $g_P(q_1) = q_2$, while the type span g_T is defined as follows: X_{g_T} is a discrete graph with two nodes x and y, $g_T^L(x) = g_T^L(y) = A$ and $g_T^L(x) = g_T^L(y) = B$ (see the bottom row of the diagram in Fig. 14). Consider the pullback-retyping diagram in Fig. 14, expressing the preservation of the start graph for morphism g (Condition (1) of Definition 15). Notice that there are two possible different morphisms k and k' from $|G_{s_2}|$ to X_{g_T} (represented via plain and dotted arrows, respectively) such that the diagram commutes and the square is a pullback. Now, it is not difficult to see that, correspondingly, we can construct two different morphisms $h_i : \mathcal{G}_1 \to \mathcal{U}_s(\mathcal{G}_2)$ ($i \in \{1, 2\}$), such that $\varphi_{\mathcal{G}_2} \circ h_i = g$, the first one mapping production q_1 into q_2' and the second one mapping q_1 into q_2''. An immediate consequence of this fact is the impossibility of extending $\mathcal{U}_s$ on morphisms, in order to obtain a functor which is right adjoint to the inclusion $\mathcal{I} : \mathbf{OGG} \to \mathbf{GG}$.

The above considerations, besides giving a negative result, also suggest a way to partially overcome the problem. An inspection of the proof of Theorem 45 reveals that the only difficulty which prevents us to extend the result is the non uniqueness of the morphisms k, k^L and k^R in the pullback-retyping diagram. In other words, if we consider any morphism $g : \mathcal{O} \to \mathcal{G}$ such that $[g_T]$ is *relational* then we can prove, as in Theorem 45, the existence of a unique morphism $h : \mathcal{O} \to \mathcal{U}_s(\mathcal{G})$ making the following diagram commute:



The coreflection result can now be restored by limiting our attention to any (non full) subcategory $\widehat{\mathbf{GG}}$ of $\mathbf{GG}$, where objects are general graph grammars, but all morphisms have a relational span as type component. The only thing to prove is that the unique morphism h constructed in the proof of Theorem 45 is indeed an arrow in $\widehat{\mathbf{GG}}$. As worked out in details in [1] for the DPO case, the generalisation to extended occurrence grammars (which are called simply occurrence grammars there) does not cause additional complications.

The naive solution of taking *all* relational morphisms as arrows of $\widehat{\mathbf{GG}}$ does not work because they are not closed under composition. A possible appropriate choice is instead given by the category $\mathbf{GG}^R$, where the arrows are grammar morphisms f such that the left component f_T^L of the type span is mono.

Definition 50 (Category $\mathbf{GG}^R$) *We denote by $\mathbf{GG}^R$ the subcategory of $\mathbf{GG}$, where for any arrow f the left component f_T^L of the type span is mono. Furthermore we denote by $\mathbf{OGG}^R$ the full subcategory of $\mathbf{GG}^R$ having occurrence grammars as objects.*

By the properties of pullbacks, the arrows in $\mathbf{GG}^R$ are closed under composition and thus $\mathbf{GG}^R$ is a well-defined subcategory of $\mathbf{GG}$.

Theorem 51 (unfolding as coreflection - reprise) *The unfolding construction can be turned into a functor $\mathcal{U}_s^R : \mathbf{GG}^R \to \mathbf{OGG}^R$, having the inclusion $\mathcal{I}_s^R : \mathbf{OGG}^R \to \mathbf{GG}^R$ as left adjoint, establishing a coreflection between the two categories.*

PROOF. By the considerations above, the only thing to prove is that the morphism h constructed as in the proof of Theorem 45 is an arrow in $\mathbf{OGG}^R$. But this is obvious since, by construction $h_T^L = g_T^L$ and thus h_T^L is mono. □

Observe that although not completely general, the above results apply to

a remarkable class of grammar morphisms. In particular, they apply to the morphisms used in [21] where the type component of an arrow from $\mathcal{G}_1$ to $\mathcal{G}_2$ is a partial graph morphism from T_1 to T_2.

Now, a functor $\mathcal{E}_s^R : \mathbf{OGG}^R \to \mathbf{AES}$ can be defined straightforwardly, by composing the corresponding restrictions of the functors Net and $\mathcal{E}_a$.

Summing up, we have the following chain of functors providing a concurrent semantics to the full class of SPO grammars.

$$\mathbf{GG}^R \underset{\mathcal{U}_s^R}{\overset{\perp}{\hookleftarrow}} \mathbf{OGG}^R \xrightarrow[\mathcal{E}_s^R]{} \mathbf{AES} \underset{\mathcal{L}_a}{\overset{\mathcal{P}}{\underset{\sim}{\hookleftarrow}}} \mathbf{Dom}$$

Unfortunately, morphisms in the range of the functor $\mathcal{N}_s$, which maps each AES to a canonical occurrence grammar, are not in $\mathbf{GG}^R$ and thus the construction in Section 8 does not induce a functor in this restricted setting, i.e., $\mathcal{N}_s$ does not restrict to a functor from **AES** to $\mathbf{OGG}^R$. Hence the whole semantic transformation is not characterised as an adjunction.

9.2 Unfolding semantics of SPO grammars with non-injective matches

In this paper we considered SPO grammars with injective matches. As formally proved in [26,27], this choice does not affect the expressiveness of the formalism: for any SPO graph grammar dealing with general matches we can obtain a grammar with injective matches which is "essentially equivalent" to the original one (e.g., which generates the same graph language). The new grammar is obtained by replacing every production of the original grammar by a finite (if the left-hand sides of productions are finite) set of productions.

Still, one could wonder if something goes wrong when injectivity condition is relaxed. It can be seen that a similar theory can be developed by allowing matches to be non-injective, but only on preserved items (this formalises the intuition that a single item can be read with multiplicity greater than one).

Definition 52 (valid match) *A match $g : L_q \to G$ is called* valid *when for any $x, y \in |L_q|$, if $g(x) = g(y)$ then $x, y \in dom(r_q)$.*

Conceptually, a match is not valid if it specifies a use of the resources which is somehow inconsistent, i.e., it requires a single resource to be consumed twice, or to be consumed and preserved at the same time. In other words, a resource can be accessed twice (or, more generally, with multiplicity greater than one) by a rewriting step only to be read.

All the results in the paper can be easily adapted to SPO grammars with

valid matches, but we are forced to consider a more restrictive notion of semi-weightedness for grammars requiring productions to be injective not only on the produced but also on the preserved items.

Definition 53 (semi-weighted grammars) *A grammar $\mathcal{G}$ is* semi-weighted *if (i) the start graph G_s is injective, and (ii) for each production $q \in P$, the right-hand side graph R_q is injective.*

With this notion of semi-weightedness the encoding of general grammars into semi-weighted grammars presented at the end of Section 2 would not work (the rules used in the encoding can be non-injective on the preserved part) and it is unclear whether an encoding exists in this case.

Summing up, the choice of restricting to injective matches does not represent a limitation for the expressiveness of the formalism, and it is technically convenient since it allows to have a coreflective semantics for a larger class of grammars, where general grammars can be encoded.

9.3 Unfolding semantics of DPO grammars

Another natural question regards the possibility of exploiting the work in this paper to obtain analogous results for the DPO approach to graph rewriting. Recall that a DPO production consists of a span of *injective total* morphisms in T-**Graph**

$$L \overset{\phi_L}{\hookleftarrow} K \overset{\phi_R}{\hookrightarrow} R,$$

where L, K, R are T-typed graphs. To apply such rule to a T-typed graph G one must find a match of the production, i.e., a total graph morphism $\phi : L \to G$ such that a diagram

$$\begin{array}{ccccc} L & \xhookleftarrow{\phi_L} & K & \xhookrightarrow{\phi_R} & R \\ {\scriptstyle\phi}\downarrow & & \downarrow & & \downarrow \\ G & \xhookleftarrow{} & D & \xhookrightarrow{} & H \end{array}$$

can be constructed in T-**Graph**, where both squares are required to be pushouts. As for SPO grammars we will consider injective matches only.

Roughly, the effect of an SPO rule $r : L \rightarrowtail R$ is similar to that of the "corresponding" DPO rule $D(r) : L \overset{r^{\leftarrow}}{\hookleftarrow} dom(r) \overset{r^{\rightarrow}}{\hookrightarrow} R$. However, the fundamental difference in the DPO rewriting mechanism is the fact the left pushout square in the diagram exists only if the match satisfies an application conditions. Informally, without getting too much into technical details, according to the so-called *dangling condition* a production q cannot be applied to a match if its

application would remove some nodes and not the attached edges, or, in other word, if the application of the production would leave some dangling edges.

The DPO and SPO approaches are equivalent when we restrict to productions which do not delete nodes.

Definition 54 (node-preserving grammars) *An* SPO *grammar* $\mathcal{G}$ *is called* node-preserving *if for any* $q \in P$*, the production* $\pi(q) = r_q : L_q \rightarrowtail R_q$ *is total on nodes. Similarly, a* DPO *grammar is* node-preserving *if for any* $q \in P$*, in the production* $\pi(q) = L_q \overset{\phi_L}{\hookleftarrow} K_q \overset{\phi_R}{\hookrightarrow} R_q$ *the function* ϕ_L *is surjective on nodes.*

Let $\mathbf{GG}^{-}$ denote the full subcategory of node-preserving SPO graph grammars. It is immediate to see that $\mathbf{GG}^{-}$ is isomorphic to the category of node-preserving DPO graph grammars, with the morphisms defined in [7]. The isomorphism maps each SPO graph grammar $\mathcal{G} = \langle T, G_s, P, \pi \rangle$ to the DPO grammar $D(\mathcal{G}) = \langle T, G_s, P, \pi' \rangle$ where any production q of is transformed as described above, i.e., $\pi'(q) = D(\pi(q))$. In this transformation, SPO derivations corresponds to DPO derivations, and vice versa.

Now, it is not difficult to see that all the results in this paper can be reformulated for the subclass of node-preserving SPO/DPO graph grammars. In particular we have the chain of adjunctions:

$$\mathbf{SGG}^{-} \underset{\mathcal{U}'_s}{\overset{\perp}{\hookleftarrow}} \mathbf{OGG}^{-} \simeq \mathbf{OGGRel}^{-} \underset{\mathsf{Net}'}{\overset{\mathsf{Gram}'}{\underset{\perp}{\leftarrow}}} \mathbf{OCN} \underset{\mathcal{E}_a}{\overset{\mathcal{N}_a}{\underset{\perp}{\leftarrow}}} \mathbf{AES} \underset{\mathcal{L}_a}{\overset{\mathcal{P}}{\underset{\sim}{\leftarrow}}} \mathbf{Dom}$$

where $\mathbf{SGG}^{-}$, $\mathbf{OGG}^{-}$, $\mathbf{OGGRel}^{-}$ denotes the subcategories of node-preserving semi-weighted and occurrence graph grammars.

This is mostly trivial as all functors above, apart from Gram', are simply the restrictions of the corresponding functors defined in this paper. The only delicate point is to define functor Gram' in order to ensure that any occurrence contextual net is mapped to a node-preserving occurrence grammar. This requires to change Definition 36 in order to enforce this property. More precisely, the definition of the set of nodes of the type graph becomes:

$$N = \{s \in S : s^{\bullet} = \emptyset\}$$

Then, it can be shown that the functors Gram' and Net' establish a coreflection between $\mathbf{OGGRel}^{-}$ and $\mathbf{OCN}$.

When we consider general DPO graph grammars, unfortunately the situation is significantly more complex. For instance, consider the safe DPO grammar in Fig. 15, which is obtained form the SPO grammar in Fig. 10 by transform-

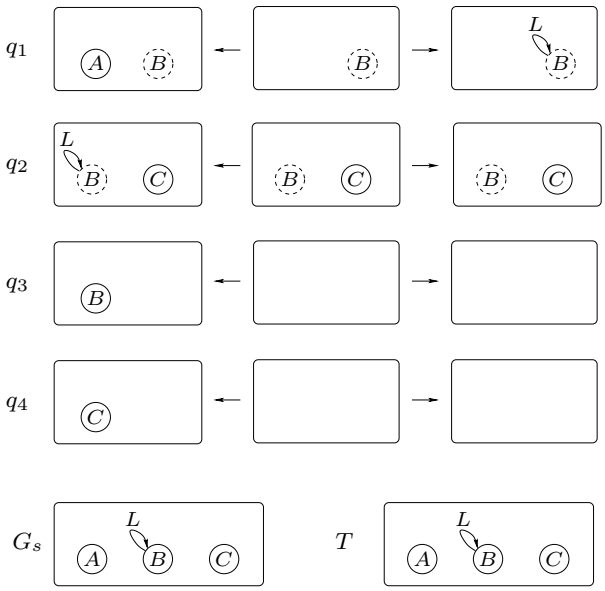


Fig. 15. A safe DPO graph grammar.

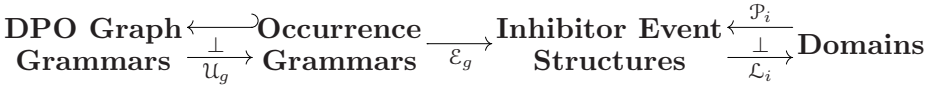
ing each SPO rule into the corresponding DPO rule, as explained above. The morphisms from the interface to the left-hand side and right-hand side of productions are inclusions represented by drawing the items in the domain as dashed circles/arrows.

Observe that production q_3, which simply removes the B-typed node, can be applied to the start graph. However, if q_1 is applied to the start graph, producing an edge L attached to B, then q_3 is "inhibited": it cannot be applied to the current graph since it would leave the L edge dangling. Then, the application of q_2, which remove the L-typed edge enables q_3 again.

As discussed in [1], this non-monotonic features of the enabling relation cannot be captured neither by a prime nor by an asymmetric event structure. The mentioned work, relying on the relationship between DPO grammars and inhibitor Petri nets, introduces a new class of event structures, the so-called *inhibitor event structures*, which properly generalises AES's (and many other event based models in the literature). The basic relation of an IES is a ternary relation which allows to express a kind of conditional or-causality. Roughly it specifies triples of the kind $(\{e'\}, e, \{e_1, \ldots, e_n\})$, which express the fact that if e' occur then event e can happen only after one among the events e_1, ..., e_n, i.e., if e' occur then e causally depends on the disjunction of $\{e_1, \ldots, e_n\}$. For instance, the dependency between the productions $\{q_1, q_2, q_3\}$ in the DPO grammar of Fig. 15 is expressed by the triple $(\{q_1\}, q_2, \{q_3\})$.

Relying on these structures one can define an unfolding construction characterised as a coreflection and a functor mapping any occurrence DPO grammar to the category of inhibitor event structures, as summarised in the diagram

below.

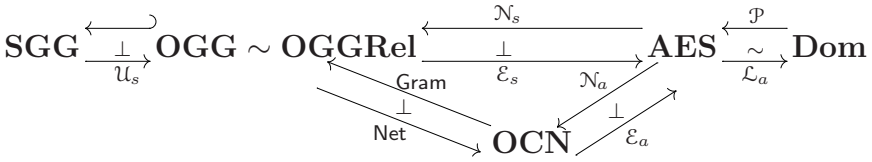


However the functor mapping each occurrence grammar to an event structure does not admit a left adjoint (see [1,2]), due to the greater expressiveness of inhibitor event structures needed to model the dependencies between events in DPO grammar derivations.

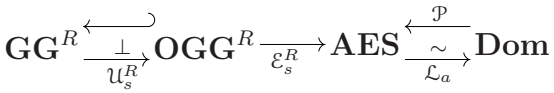
An idea to overcome this problem could be to view asymmetric event structures as a coreflective subcategory of inhibitor event structures and then to devise a construction which associates a canonical DPO grammar to any asymmetric event structure, but this goes beyond the aim of this paper.

10 Conclusions

We have defined a functorial concurrent semantics for SPO graph grammars, expressed as a chain of coreflections leading from a category of semi-weighted SPO graph grammars to the categories of asymmetric event structures and domains. The approach originally proposed by Winskel in the setting of Petri nets has been fully extended to SPO graph grammars, improving the previous proposals where some steps of the construction were lacking, notably, in the case of the DPO approach, the functor from event structures to occurrence grammars. The constructions and results in this paper are summarised by the diagram below.



For general, possibly not semi-weighted grammar, the paper shows how the above constructions can be adapted to get a chain of functors



where, unfortunately, not the whole chain is characterised as an adjunction.

The notions needed to define the unfolding naturally suggests a notion of graph process for SPO grammars, defined as a deterministic occurrence grammar

with a morphism to the original grammar. Although not worked out in this paper, it can be shown that the (abstract) processes correspond exactly the (abstract) equivalence classes of shift-equivalent derivations. Then this would establishes a link also with the *concurrent derivations* of [23], which in turn were characterised as special classes of graph grammars in [35].

The analogies between the first steps of the constructions for the SPO and DPO approaches (the proper unfolding constructions) suggest the possibility of developing a general theory of unfolding in abstract categories (e.g., high level replacement systems [17] or adhesive categories [24]). Some parts of the construction are rather concrete and not easy to recast in an abstract categorical setting, but still this represents a challenging topic of further investigation.

References

[1] P. Baldan. *Modelling concurrent computations: from contextual Petri nets to graph grammars*. PhD thesis, Department of Computer Science, University of Pisa, 2000. Available as technical report n. TD-1/00.

[2] P. Baldan, N. Busi, A. Corradini, and G.M. Pinna. Domain and event structure semantics for Petri nets with read and inhibitor arcs. *Theoretical Computer Science*, 323(1–3):129–189, 2004.

[3] P. Baldan, A. Corradini, and B. König. A static analysis technique for graph transformation systems. In K.G. Larsen and M. Nielsen, editors, *Proceedings of CONCUR 2001*, volume 2154 of *LNCS*, pages 381–395. Springer Verlag, 2001.

[4] P. Baldan, A. Corradini, and B. König. Verifying finite-state graph grammars: an unfolding-based approach. In P. Gardner and N. Yoshida, editors, *Proceedings of CONCUR 2004*, volume 3170 of *LNCS*, pages 83–98. Springer Verlag, 2004.

[5] P. Baldan, A. Corradini, and U. Montanari. Concatenable graph processes: relating processes and derivation traces. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 283–295. Springer Verlag, 1998.

[6] P. Baldan, A. Corradini, and U. Montanari. Unfolding and Event Structure Semantics for Graph Grammars. In W. Thomas, editor, *Proceedings of FoSSaCS '99*, volume 1578 of *LNCS*, pages 73–89. Springer Verlag, 1999.

[7] P. Baldan, A. Corradini, and U. Montanari. Unfolding of double-pushout graph grammars is a coreflection. In G. Ehrig, G. Engels, H.J. Kreowsky, and G. Rozemberg, editors, *TAGT'98 Conference Proceedings*, volume 1764 of *LNCS*, pages 145–163. Springer Verlag, 1999.

[8] P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures and processes. *Information and Computation*, 171(1):1–49, 2001.

[9] P. Baldan, A. Corradini, U. Montanari, and L. Ribeiro. Coreflective concurrent semantics for single-pushout graph grammars. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Proceedings of WADT'02*, volume 2755 of *LNCS*, pages 165–184. Springer Verlag, 2003.

[10] M. A. Bednarczyk and A. M. Borzyszkowski. General morphisms of Petri nets (extended abstract). In J. Wiedermann, P. van Emde Boas, and M. Nielsen, editors, *Proceedings of ICALP'99*, volume 1644, pages 190–199. Springer Verlag, 1999.

[11] R. Bruni and F. Gadducci. Some algebraic laws for spans (and their connections with multirelations). In W. Kahl, D.L. Parnas, and G. Schmidt, editors, *Proceedings of RelMiS 2001, Relational Methods in Software*, volume 44.3 of *ENTCS*, 2001. Also published as Technical Report, Bericht Nr. 2001-02, Fakultat fur Informatik, Universitat der Bundeswehr Munchen.

[12] S. Christensen and N. D. Hansen. Coloured Petri nets extended with place capacities, test arcs and inhibitor arcs. In M. Ajmone-Marsan, editor, *Applications and Theory of Petri Nets*, volume 691 of *LNCS*, pages 186–205. Springer Verlag, 1993.

[13] A. Corradini. Concurrent graph and term graph rewriting. In U. Montanari and V. Sassone, editors, *Proceedings of CONCUR'96*, volume 1119 of *LNCS*, pages 438–464. Springer Verlag, 1996.

[14] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and J. Padberg. The category of typed graph grammars and its adjunctions with categories of derivations. In J. Cuny, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Proceedings of the 5th International Workshop on Graph Grammars and their Application to Computer Science*, volume 1073 of *LNCS*. Springer Verlag, 1996.

[15] A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.

[16] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations*. World Scientific, 1997.

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

[18] H. Ehrig, R. Heckel, M. Korff, M. Löwe, L. Ribeiro, A. Wagner, and A. Corradini. Algebraic Approaches to Graph Transformation II: Single Pushout Approach and comparison with Double Pushout Approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations.* World Scientific, 1997.

[19] H. Ehrig, M. Pfender, and H.J. Schneider. Graph-grammars: an algebraic approach. In *Proceedings of IEEE Conf. on Automata and Switching Theory*, pages 167–180, 1973.

[20] U. Goltz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.

[21] R. Heckel, A. Corradini, H. Ehrig, and M. Löwe. Horizontal and vertical structuring of graph transformation systems. *Mathematical Structures in Computer Science*, 6(6):613–648, 1996.

[22] R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123:1–16, 1995.

[23] M. Korff. *Generalized graph structure grammars with applications to concurrent object-oriented systems*. PhD thesis, Technische Universität Berlin, 1996.

[24] S. Lack and P. Sobociński. Adhesive and quasiadhesive categories. *Theoretical Informatics and Applications*, 39(2):511–546, 2005.

[25] R. Langerak. *Transformation and Semantics for LOTOS*. PhD thesis, Department of Computer Science, University of Twente, 1992.

[26] M. Löwe. *Extended Algebraic Graph Transformation*. PhD thesis, Technical University of Berlin, 1990.

[27] M. Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109:181–224, 1993.

[28] M. Löwe, M. Korff, and A. Wagner. An Algebraic Framework for the Transformation of Attributed Graphs. In M.R. Sleep, M.J. Plasmeijer, and M.C. van Eekelen, editors, *Term Graph Rewriting: Theory and Practice*, pages 185–199. Wiley, London, 1993.

[29] J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88:105–155, 1990.

[30] J. Meseguer, U. Montanari, and V. Sassone. Process versus unfolding semantics for Place/Transition Petri nets. *Theoretical Computer Science*, 153(1-2):171–210, 1996.

[31] J. Meseguer, U. Montanari, and V. Sassone. On the semantics of Place/Transition Petri nets. *Mathematical Structures in Computer Science*, 7:359–397, 1997.

[32] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32(6):545–596, 1995.

[33] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.

[34] G. M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, 138(2):425–454, 1995.

[35] L. Ribeiro. *Parallel Composition and Unfolding Semantics of Graph Grammars*. PhD thesis, Technische Universität Berlin, 1996.

[36] E. Robinson and G. Rosolini. Categories of partial maps. *Information and Computation*, 79:95–130, 1988.

[37] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 1: Foundations*. World Scientific, 1997.

[38] W. Vogler. Efficiency of asynchronous systems and read arcs in Petri nets. In *Proceedings of ICALP'97*, volume 1256 of *LNCS*, pages 538–548. Springer Verlag, 1997.

[39] G. Winskel. Event Structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer Verlag, 1987.

[40] G. Winskel. Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72(3):197–238, 1987.