UNIVERSITÀ CA' FOSCARI DI VENEZIA

Dipartimento di Informatica

Technical Report Series in Computer Science

Rapporto di Ricerca CS-2005-2

Marzo 2005

Paolo Baldan, Andrea Corradini, Ugo Montanari, Leila Ribeiro

Concurrency and Nondeterminism in Graph Rewriting: From Graph Grammars to Asymmetric Event Structures and Backwards

Dipartimento di Informatica, Università ca' Foscari di Venezia
via Torino, 155 – 30172 Mestre-Venezia, Italy.
TEL: +39 041 2348411 FAX: +39 041 2348419

Concurrency and Nondeterminism in Graph Rewriting: From Graph Grammars to Asymmetric Event Structures and Backwards*

Paolo Baldan[1], Andrea Corradini[2], Ugo Montanari[2], Leila Ribeiro[3]

[1] Dipartimento di Informatica, Università Ca' Foscari di Venezia, Italia.
[2] Dipartimento di Informatica, Università di Pisa, Italia.
[3] Instituto de Informática, Universidade Federal do Rio Grande do Sul, Brazil
`baldan@dsi.unive.it, {andrea, ugo}@di.unipi.it, leila@inf.ufrgs.br`

Abstract. The problem of extending to graph grammars the unfolding semantics originally developed by Winskel for (safe) Petri nets has been faced several times along the years, both for the single-pushout and double-pushout approaches to graph rewriting, 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 safe (actually, more generally, of semi-weighted) graph grammars and the category of prime algebraic domains.

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 [32] presents a comprehensive introduction to the theory of the most important 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 [12] 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.

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 [19] and Winskel's unfolding semantics [34]. 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 [18, 15], 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 [25, 17], where a rule application is described instead as a *single* pushout in a category of graphs and *partial* morphisms.

* Research partially supported by the CNPq-CNR Project IQ-MOBILE, by the EU FET-GC Project IST-2001-32747 AGILE, and by the EC RTN 2-2001-00346 SEGRAVIS.

Recall that, building on [29], the seminal work [34] proposes a concurrent semantics 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.

$$\begin{tikzcd} \textbf{Safe Nets} \arrow[r, shift right, "\mathcal{U}"'] \arrow[r, phantom, "\perp"] & \textbf{Occurrence Nets} \arrow[l, hook', shift right] \arrow[r, shift right, "\mathcal{E}"'] \arrow[r, phantom, "\perp"] & \textbf{Prime Event Structures} \arrow[l, shift right, "\mathcal{N}"'] \arrow[r, shift right, "\mathcal{L}"'] \arrow[r, phantom, "\sim"] & \textbf{Domains} \arrow[l, shift right, "\mathcal{P}"'] \end{tikzcd}$$

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 [27] 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:

$$\begin{tikzcd} \textbf{Semi-Weighted Contextual Nets} \arrow[r, shift right, "\mathcal{U}_a"'] \arrow[r, phantom, "\perp"] & \textbf{Contextual Occurrence Nets} \arrow[l, hook', shift right] \arrow[r, shift right, "\mathcal{E}_o"'] \arrow[r, phantom, "\perp"] & \textbf{Asymmetric Event Structures} \arrow[l, shift right, "\mathcal{N}_o"'] \arrow[r, shift right, "\mathcal{L}_a"'] \arrow[r, phantom, "\perp"] & \textbf{Domains} \arrow[l, shift right, "\mathcal{P}_a"'] \end{tikzcd}$$

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 among 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:

$$\begin{tikzcd} \textbf{DPO Graph Grammars} \arrow[r, shift right, "\mathcal{U}_g"'] \arrow[r, phantom, "\perp"] & \textbf{Occurrence Grammars} \arrow[l, hook', shift right] \arrow[r, "\mathcal{E}_g"'] & \textbf{Inhibitor Event Structures} \arrow[r, shift right, "\mathcal{L}_i"'] \arrow[r, phantom, "\perp"] & \textbf{Domains} \arrow[l, shift right, "\mathcal{P}_i"'] \end{tikzcd}$$

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 is found the corresponding rule can always be applied.

Building on the results briefly summarised above, we present a coreflective unfolding semantics for semi-weighted SPO graph grammars defined through the following chain of coreflections:

$$\begin{tikzcd}[column sep=large] \begin{array}{c}\textbf{SPO Graph}\\ \textbf{Grammars}\end{array} \arrow[r, shift right=1.5ex, "\mathcal{U}_s"'] \arrow[r, phantom, "\perp"] & \begin{array}{c}\textbf{Occurrence}\\ \textbf{Grammars}\end{array} \arrow[l, shift right=1.5ex, hook'] \arrow[r, shift right=1.5ex, "\mathcal{E}_s"'] \arrow[r, phantom, "\perp"] & \begin{array}{c}\textbf{Asymmetric Event}\\ \textbf{Structures}\end{array} \arrow[l, shift right=1.5ex, "\mathcal{N}_s"'] \arrow[r, shift right=1.5ex, "\mathcal{L}_a"'] \arrow[r, phantom, "\perp"] & \textbf{Domains} \arrow[l, shift right=1.5ex, "\mathcal{P}_a"'] \end{tikzcd}$$

To our knowledge, this is the first time that Winskel's chain of coreflections has been generalised to a category of graph grammars in a completely satisfactory way. In particular, this construction differs from and improves that for DPO graph grammars, discussed above, because of the following facts:

- Due to the absence of application conditions for rules, asymmetric event structures are sufficient to represent the dependencies between events in SPO graph grammars.
  This suggests the existence of a tight relationship between occurrence SPO grammars and the occurrence contextual nets mentioned above, as for both the dependencies among transitions/rules are suitably modelled by AES's. Indeed, this relationship is formalised in the paper by defining a functor from occurrence SPO graph grammars to occurrence contextual nets, which is then exploited to inherit for graph grammars several results already proved in [8] for contextual nets.
- A novel construction, inspired by the work on contextual nets, allows to generate a canonical occurrence SPO graph grammar from any asymmetric event structure. This 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.
- As far as the first and third coreflections in the above chain are concerned, the first one essentially rephrases in the framework of SPO grammars the unfolding construction of DPO grammars presented in [7], and it is conceptually very close to the unfolding construction proposed in [31]; the third one instead is borrowed from [8].

As discussed in a concluding section, the results in this paper do not generalise easily to larger categories of graph grammars. For example, considering general, possibly not semi-weighted SPO graph grammars, we can still characterise the unfolding construction as a coreflection, provided that we stick to a smaller class of grammar morphisms. However, in this case the remaining part of the chain of coreflections leading 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. As discussed above these results do not extend to general DPO graph grammars because of the presence of the dangling application condition for rules. However they can provide some suggestions for improving the treatment of this more complex case.

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 morphism 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 with the category of occurrence contextual nets. In Section 5 we introduce the notion of a (nondeterministic) process for SPO graph grammars. Then, in Section 6, 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 7 we complete the chain of coreflections from grammars to domains, showing how any occurrence grammar can be abstracted to an asymmetric event structure and, vice versa, how a canonical occurrence grammar can be generated from any asymmetric event structure. In Section 8 we discuss some possible generalisations of our results and the relationships with the work on the DPO approach. Finally Section 9 draws some conclusions.

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 [25], an algebraic approach to graph rewriting alternative to the classical *double-pushout* (DPO) approach. The original SPO approach is adapted to deal with *typed graphs* [14, 26], 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 $G = \langle N_G, E_G, s_G, t_G \rangle$, where N_G and E_G are the sets of* nodes *and* edges *of G, and $s_G, t_G : E_G \to N_G$ 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)):*

$$s_H \circ f_E \leq f_N \circ s_G \quad \text{and} \quad t_H \circ f_E \leq f_N \circ t_G. \text{ (*)}$$

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 $s_H \circ f_E = f_N \circ s_G$ and $t_H \circ f_E = f_N \circ t_G$ 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., $x \in N_G \cup E_G$.

$$\begin{tikzcd}
E_G \arrow[rr, "f_E"] \arrow[d, shift right, "s_G"'] \arrow[d, shift left, "t_G"] & \geq & E_H \arrow[d, shift right, "s_H"'] \arrow[d, shift left, "t_H"] \\
N_G \arrow[rr, "f_N"'] & & N_H
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{|G_1|} \arrow[rr, "f"] \arrow[dr, "t_{G_1}"'] & \geq & {|G_2|} \arrow[dl, "t_{G_2}"] \\
& T &
\end{tikzcd}$$

(a) (b)

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

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). *Fixed 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 *of the production, respectively.*

Given a typed graph G and a match *(i.e., a total 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**.*

$$\begin{tikzcd}
L_q \arrow[r, tail, "r_q"] \arrow[d, "g"'] & R_q \arrow[d, tail, "h"] \\
G \arrow[r, tail, "d"'] & H
\end{tikzcd}$$

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* [15] 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, production q in the top row of Fig. 2, which consumes node B, can be applied to the graph G in the same figure. As a result both node B and the loop edge L are removed.
Even if the category **PGraph** has all pushouts, still we will consider a condition which corresponds to the *identification condition* of the DPO approach.

Definition 4 (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 by a rewriting step only to be read.

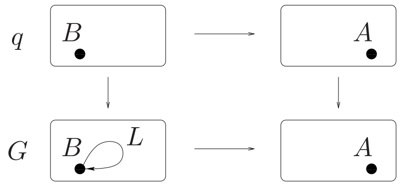


Fig. 2. Side-effects in SPO rewriting.

Definition 5 (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.,* $\rho = \{G_{i-1} \Rightarrow_{q_{i-1}} G_i\}_{i \in \{1,\dots,n\}}$*, with* $G_0 = G_s$*. A derivation is* valid *if so are all the matches in its direct derivations.*

In the paper we will consider only *consuming* graph grammars and *valid* derivations. The restriction to consuming grammars 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., [19, 34]) where events in computations are identified with their causal history (formally, the unfolding construction would not work). Considering only valid derivations is needed to have a computational interpretation which is resource-conscious, i.e., where a resource can be consumed only once. (From a technical point of view, considering non-valid derivations we would not get the adjunction between occurrence grammars and event structures in Theorem 3.)

For a graph grammar $\mathcal{G}$ we denote by $Elem(\mathcal{G})$ the set $N_T \cup E_T \cup P$. We will assume that for each production name q the corresponding production $\pi(q)$ is $L_q \stackrel{r_q}{\rightarrowtail} R_q$. 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.

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 relation between Petri nets and (SPO) graph grammars. The correspondence between these two formalisms (see, e.g., [12] 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* [28], also called nets with test arcs in [11], activator arcs in [21] or read arcs in [33], an 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. 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 $M \leq M'$ 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 $\mu R(\sum_{a \in A} n_a \cdot a) = \sum_{b \in B} \sum_{a \in A} (n_a \cdot R(a,b)) \cdot b$.[4] A *relation* $R : A \leftrightarrow B$ is a multirelation R where multiplicities are bounded by one, namely $R(a,b) \leq 1$ for all $a \in A$ and $b \in B$.

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 $\underline{A} = \{s \in S \mid \exists t \in Tr \,.\, A(t) > 0 \wedge C(t,s)\}$. The same notation is used to denote the functions from S to $\mathbf{2}^{Tr}$ 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* enabled *at* M *if* ${}^\bullet A + \underline{A} \leq M$*. The* transition relation *between markings is defined as*

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

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") function $l : X_M \to A$ and satisfying $|f^{-1}(a)| = M(a)$ for all $a \in A$ (this defines X_M up to isomorphisms). 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 labelled over) the places.

Fig. 3 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 labelled by s_0 and one node labelled by s_1, produces two nodes labelled by s_2 and s_3, respectively, and preserves one node labelled by s, the only element in $dom(r_t)$.

This encoding satisfies the basic properties which one would expect: production r_t can be applied, via a *valid* match, to a discrete labelled 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-set.

[4] 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 $\sum_{n \in \mathbb{N}} 1 \cdot n$.

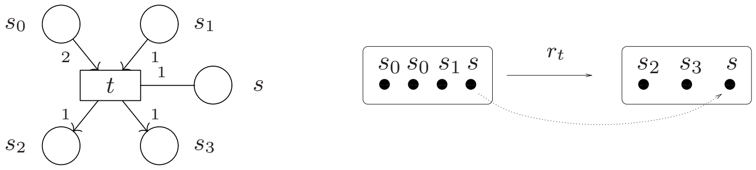


Fig. 3. A contextual Petri net transition and a corresponding SPO production.

2.3 Semi-weighted grammar

As recalled in the Introduction, the coreflection results have been proved for specific subclasses of (contextual) Petri nets, namely for *safe* nets in [34], and for *semi-weighted* (contextual) nets in [27, 8]. We introduce here corresponding classes of graph grammars: the reasons why the presented results 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 (i) each marking* m' *reachable from* m *is a set, i.e.,* $m'(s) \leq 1$ *for all* $s \in S$*, and (ii) 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.*[5]

Net N *is* semi-weighted *[27] if (iii) the initial marking is a set, and (iv) the multi-relation* $F_{post} : Tr \leftrightarrow S$ *is a relation.*

Since clearly (i) implies (iii) and (ii) implies (iv), 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 should consider all possible computations, because of condition (i).

In the encoding of nets as graph grammars sketched in the previous section, a marking is represented as a set (discrete graph) labelled 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 labelling function. As general SPO productions act on general graphs (possibly non-discrete), 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 (i) for all* H *such that* $G_s \Rightarrow^* H$*,* H *is injective, and (ii) 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 (iii) the start graph* G_s *is injective, and (iv) for each production* $q \in P$*, the right-hand side graph* R_q *is injective.*

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.

3 Graph grammar morphisms

In this section the class of SPO graph grammars is turned into 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 [13, 7], which in turn were defined as a generalisation of (contextual) Petri net morphisms.

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".

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

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.*

Observe 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., $\mu h_S({}^\bullet t + t^\bullet + \underline{t}) = \emptyset$.

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 discussed, for example, in [10], 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.

Definition 11 (spans). *Let* **C** *be a category. A* (concrete) span $f : A \leftrightarrow B$ *in* **C** *is a pair of coinitial 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. 5.(a)). The isomorphism class of a span* f *is denoted by* $[f]$ *and called a* semi-abstract span*.*

Fig. 4 shows two spans in **Set** and the corresponding multirelations.

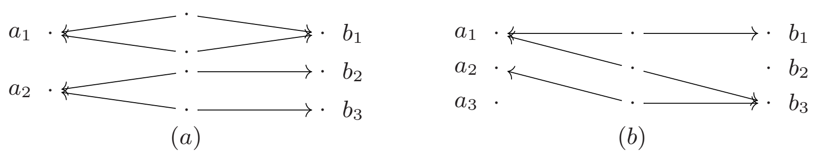


Fig. 4. The (semi-abstract) spans for the multirelations (a) $R_1(a_1, b_1) = 2$, $R_1(a_2, b_2) = 1$, $R_1(a_2, b_3) = 1$ and (b) $R_2(a_1, b_1) = 1$, $R_2(a_1, b_3) = 1$, $R_2(a_2, b_3) = 1$ (Pairs which are not mentioned are mapped to 0).

Definition 12 (category of spans). *Let* **C** *be a category with pullbacks. The category* $\mathbf{Span}(\mathbf{C})$ *has the same objects of* **C** *and semi-abstract spans on* **C** *as 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. 5.(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* **C***.*

$$\begin{tikzcd}
 & X_{f'} \arrow[dl, "f'^L"'] \arrow[dr, "f'^R"] & \\
A & X_f \arrow[l, "f^L"] \arrow[r, "f_R"'] \arrow[u, "k"] & B
\end{tikzcd}$$

(a)

$$\begin{tikzcd}
 & & X_f \arrow[dl, dotted, "y"'] \arrow[dr, dotted, "z"] \arrow[d, phantom, "(1)"] & & \\
A & X_{f_1} \arrow[l, "f_1^L"] \arrow[r, "f_1^R"'] & B & X_{f_2} \arrow[l, "f_2^L"] \arrow[r, "f_2^R"'] & C
\end{tikzcd}$$

(b)

Fig. 5. Equivalence and composition of spans.

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. 4.(a) is not relational, while that in Fig. 4.(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 [10], 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.[6] 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 right leg is mono. Such spans correspond one-to-one with partial morphisms from A to B and backward. In fact, a partial morphism $g : A \rightharpoonup B$ can be identified with the span

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

and similarly a partial morphism $h : B \rightharpoonup A$ 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

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

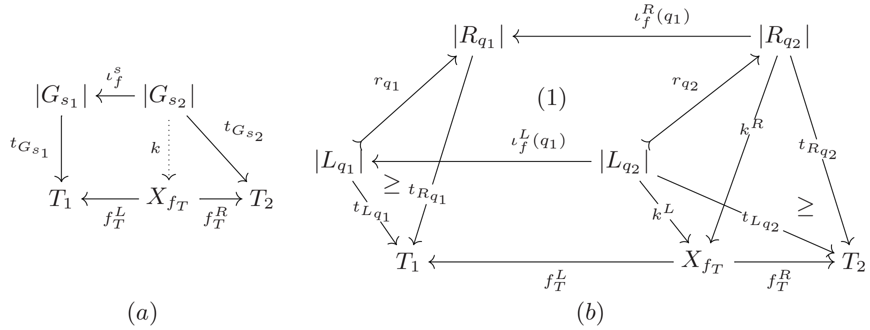


Fig. 6. Diagrams for SPO grammar morphisms.

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:*

$$\begin{tikzcd}
{|G_1|} \arrow[d, "t_{G_1}"'] & {|G_2|} \arrow[l, dotted, "x"'] \arrow[d, dotted, "y"] \arrow[dr, "t_{G_2}"] & \\
T_1 & X_{f_T} \arrow[l, "f_T^L"] \arrow[r, "f_T^R"'] & T_2
\end{tikzcd}$$

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* $\emptyset \rightarrowtail \emptyset$*;*
- 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. 6.(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. 6.(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 [31, 1, 13] 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 [13], is not "functional".

Lemma 1. *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 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 is used, in particular, to establish a tight connection between occurrence grammars and occurrence contextual nets, which allows to "inherit" for graph grammars several useful results already proved in [8] for 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 safely think that a production *produces*, *preserves* or *consumes* items of the type graph, and using a net-like language, we speak of pre-set, 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 17 (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 $E_T \cup N_T$:*

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

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

For instance, for grammar $\mathcal{G}$ in Fig. 7, $^\bullet q_1 = \{A\}$, $\underline{q_1} = \{B\}$ and $q_1{}^\bullet = \{L\}$, while $^\bullet B = \emptyset$, $\underline{B} = \{q_1, q_2\}$ and $B^\bullet = \{q_3\}$.

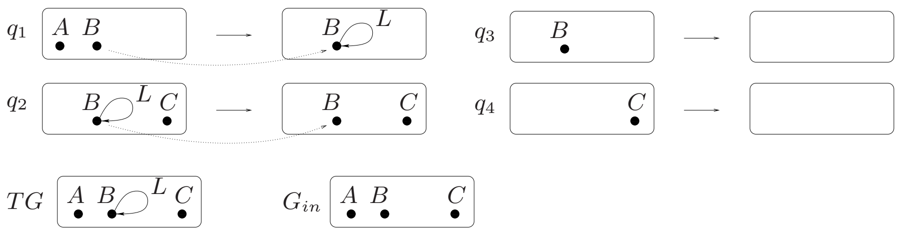


Fig. 7. A safe SPO graph grammar $\mathcal{G}$.

Also the causality relation and the asymmetric conflict relation we are going to define are meaningful only for safe grammars, but it is technically convenient to define them for arbitrary graph grammars.

Definition 18 (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. 7), does not imply $q < q'$. Actually, the dependency between the two productions is a kind of *asymmetric conflict* (see [8, 30, 24]). The application of q' prevents q from being applied, so that q can never follow q' in a derivation (or equivalently when both q and q' occur in a derivation then q must precede q'). But the converse is not true, since q *can* be applied before q'.

Definition 19 (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 20 (conflict). *The* conflict *relation $\# \subseteq \mathbf{2}^P_{fin}$ associated to a grammar $\mathcal{G}$ is defined as:*

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

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 1) 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 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. 7, 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 [31]) 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. 7. 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 consumed 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 21 (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 $\langle Elem(\mathcal{O}), \leq \rangle$ [7] (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, namely $|{}^{\bullet}x| \leq 1$;*

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

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

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

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. An example of occurrence grammar is our running example $\mathcal{G}$ in Fig. 7.

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

Definition 22 (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 iff 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 2). 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 8.1).

Lemma 2. *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:

$$\begin{tikzcd}
Min(\mathcal{O}) \arrow[d, hook] & {|G_s|} \arrow[l, "\iota_f^s"'] \arrow[d, dotted, "k"] \arrow[dr, "t_{G_s}"] & \\
T' & X_{f_T} \arrow[l, "f_T^L"] \arrow[r, "f_T^R"'] & T
\end{tikzcd}$$

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 $x = k(x'') = k(y'') = y$ that is what we wanted to prove. A 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. $\square$

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 2, $[f_T]$ 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 23 (graph relation). *A* graph relation τ *over graphs* $G_1 = \langle N_1, E_1, s_1, t_1\rangle$ *and* $G_2 = \langle N_2, E_2, s_2, t_2\rangle$*, 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* $\tau = \langle N_\tau, E_\tau, s_\tau, t_\tau\rangle$*, where* $N_\tau \subseteq N_1 \times N_2$*,* $E_\tau \subseteq E_1 \times E_2$*,* $s_\tau(\langle e_1, e_2\rangle) = \langle s_1(e_1), s_2(e_2)\rangle$*, 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 $\mathit{Span}(\tau) = G_1 \stackrel{\pi_1}{\leftarrow} [\tau] \stackrel{\pi_2}{\rightarrow} G_2$, 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 $\mathit{Rel}(f) : G_1 \times G_2$ be the graph relation such that $\mathit{Rel}(f)(x_1, x_2)$ iff 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 24 (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 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* $x_1 = y_1$*.*

Lemma 3 (image through graph relation 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*

$$f(G_1', G_2') \text{ (with } G_i' \text{ seen as graphs typed over } G_i \text{ by the inclusion)} \quad \text{iff} \quad \mathit{Rel}(f)(G_1', G_2')$$

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. In order to simplify the notation, let us denote *Rel*(f) by τ_f. 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

$$\begin{tikzcd}
G'_1 \arrow[d, hook] & G'_2 \arrow[l, "\iota_f"'] \arrow[d, dotted, "k"] \arrow[dr, hook] & \\
G_1 & X_f \arrow[l, "f^L"] \arrow[r, "f^R"'] & G_2
\end{tikzcd}$$

where the square is a pullback and, by Lemma 2, the morphism k is uniquely determined.

Let us show that $\tau_f(G'_1, G'_2)$, i.e., that (1) for any $x_1 \in G'_1$, if $\tau_f(x_1, x_2)$ for $x_2 \in G_2$ then $x_2 \in G'_2$, and (2) for any $x_2 \in G'_2$, there exists a unique $x_1 \in G'_1$ such that $\tau_f(x_1, x_2)$.

1. Let $x_1 \in G'_1$, $x_2 \in G_2$ and assume that $\tau_f(x_1, x_2)$. Then there exists $x \in X_f$ such that $f^L(x) = x_1$ and $f^R(x) = x_2$. Since the square is a pullback, there exists $x'_2 \in G'_2$ such that $k(x'_2) = x$. Thus $x_2 = f^R(k(x'_2)) = x'_2$ and we can conclude $x_2 \in G'_2$.
2. For $x_2 \in G'_2$, take $x_1 = f^L(k(x_2))$. Then we have $\tau_f(x_1, x_2)$. Furthermore x_1 is unique. In fact, let $x'_1 \in G'_1$ such that $\tau_f(x_1, x_2)$. Then, by construction, there exists $x' \in X_f$ such that $f^L(x') = x'_1$ and $f^R(x'_1) = x_2$. Since the square in the diagram above is a pullback, there exists $x'_2 \in G'_2$ such that $k(x'_2) = x'$ and hence $x'_2 = f^R(k(x'_2)) = x_2$. Therefore, $x'_1 = f^L(x') = f^L(k(x'_2)) = f^L(k(x_2)) = x_1$, as desired.

($\Leftarrow$) 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 $\mathit{Span}(\tau) : G_1 \leftrightarrow G_2$ and let us show that $f_\tau(G'_1, G'_2)$. By Definition 23, 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 $x_2 \in G'_2$. Consider now the following diagram

$$\begin{tikzcd}
G'_1 \arrow[d, hook] & G'_2 \arrow[l, "\iota_{f_\tau}"'] \arrow[d, dotted, "k"] \arrow[dr, hook] & \\
G_1 & X_{f_\tau} \arrow[l, "f_\tau^L"] \arrow[r, "f_\tau^R"'] & G_2
\end{tikzcd}$$

The diagram clearly commutes. To prove that the square is a pullback, take $x_1 \in G'_1$ and $x \in X_{f_\tau}$ such that $f_\tau^L(x) = x_1$. Then we must find a corresponding element in G'_2 which "projects" over x_1 and x, respectively. To this aim take $x_2 = f_\tau^R(x) \in G_2$. Since $\tau(x_1, x_2)$ by hypothesis, $x_2 \in G'_2$ and therefore, by construction, $k(x_2) = x$ and $\iota_{f_\tau}(x_2) = x_1$, as desired. It is also easy to prove that such x_2 is unique and thus we conclude that $f_\tau(G'_1, G'_2)$.

Coming back to the desired implication, assume that $\mathit{Rel}(f)(G'_1, G'_2)$. By the above proof we deduce that $\mathit{Span}(\mathit{Rel}(f))(G'_1, G'_2)$. Recalling that $\mathit{Span}(\mathit{Rel}(f)) = f$, we conclude, as desired, that $f(G'_1, G'_2)$. $\square$

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

Definition 25 (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) $f_\tau(q_1{}^\bullet, q_2{}^\bullet)$

Theorem 1. *The categories* **OGG** *and* **OGGRel** *are isomorphic.*

Proof. Let $F : \mathbf{OGG} \to \mathbf{OGGRel}$ and $G : \mathbf{OGGRel} \to \mathbf{OGG}$ be the functors defined as follows:

- both F and G are the identity on objects;
- given a morphism $f : \mathcal{O}_1 \to \mathcal{O}_2$ in **OGG**, $f = \langle f_P : P_1 \to P_2, f_T : T_1 \leftrightarrow T_2, \iota_f \rangle$, then $F(f) = \langle f_P : P_1 \to P_2, Rel(f_T) : T_1 \times T_2 \rangle$;
- given a morphism $h : \mathcal{O}_1 \to \mathcal{O}_2$ in **OGGRel**, $h = \langle h_P : P_1 \to P_2, h_\tau : T_1 \times T_2 \rangle$ then $G(h) = \langle h_P : P_1 \to P_2, Span(h_\tau) : T_1 \leftrightarrow T_2, \iota \}$, 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 3, by exploiting conditions (1) and (2) of Definition 25.

The well-definedness of F and G can be proved easily by exploiting Lemma 3. The fact that they are inverse to each other follows immediately because so are $Rel(\cdot)$ and $Span(\cdot)$. $\square$

4.3 Relating occurrence grammars and occurrence contextual nets

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. 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 18 and 19).

Definition 26 (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 27 (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 $t \in Tr$,*

$$\text{(i) } h_S({}^\bullet t, {}^\bullet h_T(t)), \qquad \text{(ii) } h_S(t^\bullet, h_T(t)^\bullet) \qquad \text{(iii) } h_S(\underline{t}, \underline{h_T(t)})$$

where the image of a set through a relation is defined as for graphs (see Definition 24). We denote by **OCN** *the category of occurrence contextual nets with the above morphisms.*

We are now ready to formalise the relationship between occurrence grammars and occurrence contextual nets. We define a functor that maps each occurrence graph grammar to an occurrence contextual net. This is done by considering each graph as an unstructured collection of nodes and edges, to be read as a marking of the contextual net. 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.

Definition 28 (from occurrence grammars to occurrence contextual nets). *Let* 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* $\mathbf{OGGRel}$*, 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 1. *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 having as nodes those in m and as edges 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 $N = m \cap N_T$, $E = m \cap \{e \in E_T \mid s(e), t(e) \in N\}$ and the source and target functions are $s = s_{T|E}$ and $t = t_{T|E}$.

Lemma 4. *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* $\mathcal{O}$ *of the kind*

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

Proof. The proof proceeds by induction on n.

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

$(\mathsf{n} \to \mathsf{n} + 1)$ 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

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

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 of the edges which do not appear in $graph(m_n)$ since they would be dangling. This is obvious, since to use (read or consume) an edge, production q_{n+1} must use also all of its attached nodes.

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 $graph(m_{n+1})$. $\square$

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 1. *Let $\mathcal{O} = \langle T, P, \pi \rangle$ be any occurrence grammar. A subgraph of T is concurrent iff 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. 7. 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 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 processes* 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 computations 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 [19]. 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 29 (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 30 (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 coherent with the basic theory of concurrency for the SPO approach to graph grammars and, in particular, with the notion of shift-equivalence for derivations [32] and with the concurrent derivations of [22, 31]. 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 valid 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. These results can be proved by adapting the analogous result for the DPO approach relating (non-concatenable) graph processes and derivation traces (see, e.g., [5]).

6 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 31 (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 $G \xleftarrow{m} L_q \xhookleftarrow{r_q^{\leftarrow}} dom(r_q)$ and $dom(r_q) \xhookrightarrow{r_q} R_q$.

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 32 (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:*

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

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 33 (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 valid 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.

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 8.1 we will see how, suitably restricting graph grammar morphisms to still interesting subclasses (including, for instance, the morphisms of [20]) 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 2 (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 folding morphism as in Definition 33. 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:*

$$\begin{tikzcd}
\mathcal{U}_s(\mathcal{G}) \arrow[r, "\varphi"] & \mathcal{G} \\
\mathcal{O} \arrow[u, dotted, "h"] \arrow[ur, "g"'] &
\end{tikzcd}$$

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 $\iota_h = \{\iota_h(q_o) \mid q_o \in P_o\} \cup \{\iota_h^s\}$ 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 $\langle g_T^L, X_{g_T}, g_T^R \rangle : T_o \to T$ be an arbitrary but fixed representative of $[g_T]$, and let $\langle id, T', \varphi_T^R \rangle : T' \to T$ be a representative of $[\varphi_T]$ (where, since φ is strong, we can choose the identity of T' as left component).

$$\begin{tikzcd}
& & X_{g_T} \arrow[lld, "g_T^L"'] \arrow[ld, dotted, "id"] \arrow[rd, dotted, "h_T^R"] \arrow[rrd, "g_T^R"] & & \\
T_o & X_{g_T} \arrow[l, "g_T^L"] \arrow[r, "h_T^R"'] & T' & T' \arrow[l, "id"] \arrow[r, "\varphi_T^R"'] & T
\end{tikzcd}$$

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 $\langle g_T^L, X_{g_T}, h_T^R \rangle$ for some h_T^R, 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 $\iota_\varphi^s : |G_s| \to |G'_s|$ is the identity by Definition 33; thus, since $\varphi \circ h = g$ implies $\iota_h^s \circ \iota_\varphi^s = \iota_g^s$, we deduce that $\iota_h^s = \iota_g^s$. 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

$$\begin{tikzcd}
\mathcal{U}_s(\mathcal{G}) \arrow[r, "\varphi"] & \mathcal{G} \\
\mathcal{O}^{[n]} \arrow[u, dotted, "h^{[n]}"] \arrow[ur, "g^{[n]}"'] &
\end{tikzcd}$$

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 2, k is uniquely determined.

$$\begin{tikzcd}
|G_{s_o}| \arrow[d, "id"'] & |G_s| \arrow[l, "\iota_g^s"'] \arrow[d, dotted, "k"] \arrow[dr, "t_{G_s}"] & \\
|G_{s_o}| & X_{g_T}{}^{[0]} \arrow[l, "g_T^{L[0]}"] \arrow[r, "g_T^{R[0]}"'] & T
\end{tikzcd}$$

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.

$$\begin{tikzcd}
|G_{s_o}| \arrow[d, "id"'] & |G_s| \arrow[l, "\iota_g^s"'] \arrow[d, dotted, "k"] \arrow[dr, "t_{G'_s}"] & \\
|G_{s_o}| & X_{g_T}{}^{[0]} \arrow[l, "g_T^{L[0]}"] \arrow[r, "h_T^{R[0]} = t_{G'_s} \circ k^{-1}"'] & T'
\end{tikzcd}$$

($\mathbf{n \to n{+}1}$) 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.

$$\begin{tikzcd}
|L_{q_o}| \arrow[d, "t_{L_{q_o}}"'] & |L_q| \arrow[l, "\iota_g^L(q_o)"'] \arrow[d, dotted, "k^L"] \arrow[dr, "t_{L_q}"] & \\
T_o{}^{[n]} & X_{g_T}{}^{[n]} \arrow[l, "g_T^{L[n]}"] \arrow[r, "g_T^{R[n]}"'] & T
\end{tikzcd}$$

Moreover, since $\mathcal{G}$ is a semi-weighted grammar, by Lemma 2, g_T is relational and thus the arrow k^L is uniquely determined. By Definition 32, $depth(x) \leq n$ 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.

$$\begin{tikzcd}
{|L_{q_o}|} \arrow[d, "t_{L_{q_o}}"'] & {|L_q|} \arrow[l, "\iota_g^L(q_o)"'] \arrow[d, dotted, "k^L"] \arrow[dr, "m=h_T^{R^{[n]}} \circ k^L"] & \\
{T_o}^{[n]} & {X_{g_T}}^{[n]} \arrow[l, "g_T^{L^{[n]}}"] \arrow[r, "h_T^{R^{[n]}}"'] & T'
\end{tikzcd}$$

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 1.(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 valid 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 1.(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 and hence valid.

Since $m : L_q \to \langle T', \varphi_T^R \rangle$ is a valid 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.

$$\begin{tikzcd}
{|R_{q_o}|} \arrow[d, "t_{R_{q_o}}"'] & {|R_q|} \arrow[l, "\iota_g^R(q_o)"'] \arrow[d, dotted, "k^R"] \arrow[dr, "t_{R_q}"] & \\
{T_o}^{[n+1]} & {X_{g_T}}^{[n+1]} \arrow[l, "g_T^{L^{[n+1]}}"] \arrow[r, "g_T^{R^{[n+1]}}"'] & T
\end{tikzcd}$$

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. $\square$

7 Event structure semantics for SPO graph grammars

In this section we show that asymmetric event structures, a generalisation of prime event structures introduced in [8], provide a suitable setting for defining an event structure semantics for SPO graph grammars. After reviewing the basics of asymmetric event structures, we show that any occurrence SPO grammar can be mapped to an asymmetric event structure via a functorial construction. Furthermore, a left adjoint functor, back from asymmetric event structures to occurrence grammars, can be defined, associating a canonical occurrence grammar to any asymmetric event structure.

7.1 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 more 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 pre-asymmetric event structures. Then asymmetric event structures will be defined as special pre-asymmetric event structures satisfying a suitable condition of "saturation".

Definition 34 (asymmetric event structure). *A* pre-asymmetric 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 $\epsilon, \epsilon' \in \mathcal{E}$:*

$$(a)\ \epsilon < \epsilon' \quad \Rightarrow \quad \epsilon \nearrow \epsilon', \qquad\qquad (b)\ \nearrow \text{ is acyclic in } \lfloor \epsilon \rfloor,$$

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 conflict is inherited through causality, a typical property also of PES's.

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

Definition 35 (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 $\epsilon \nearrow' \epsilon'$ iff $(\epsilon \nearrow \epsilon')$ 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 36 (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 domains (*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**.

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

7.2 From occurrence grammars to AES's

Given any occurrence grammar, the corresponding asymmetric event structure is readily obtained by taking the production names as events. Causality and asymmetric conflict are the relations defined in Definitions 18 and 19.

Definition 37 (AES for an occurrence grammar). *Let $\mathcal{O} = \langle T, P, \pi \rangle$ be an occurrence grammar. The* AES *associated to $\mathcal{O}$, denoted $\mathcal{E}_s(\mathcal{O})$, is the saturation of the pre-*AES *$\langle P, \leq, \nearrow \rangle$, with $\leq$ and $\nearrow$ as in Definitions 18 and 19.*

The above construction naturally gives rise to a functor.

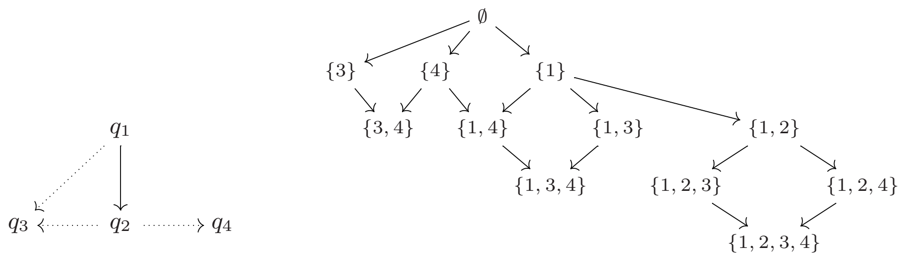
Proposition 2. *For any morphism $h : \mathcal{O}_0 \to \mathcal{O}_1$ between occurrence grammars, let $\mathcal{E}_s(h)(q) = h_P(q)$ if $h_P(q) \neq \emptyset$ and $\mathcal{E}_s(h)(q)$ undefined, otherwise. Then $\mathcal{E}_s$:* **OGGRel** $\to$ **AES** *is a well-defined functor.*

Proof. Well-definedness of $\mathcal{E}_s$ easily follows from Corollary 1(3). Alternatively, it derives from the observation that $\mathcal{E}_s = \mathcal{E}_a \circ \mathsf{Net}$, where $\mathcal{E}_a$: **OCN** $\to$ **AES** is the functor mapping each occurrence contextual net to the corresponding asymmetric event structure, as defined in [8] (Definition 7.29). □

For instance, Fig. 8 shows the AES (and the domain of its configurations) associated to the occurrence grammar $\mathcal{G}$ in Fig. 7. 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".

7.3 From AES's to occurrence grammars

Any AES is identified with a canonical occurrence grammar, via a free construction that mimics Winskel's one. Given an asymmetric event structure $\mathcal{A}$, the corresponding grammar has the events of $\mathcal{A}$ as production names, while the graph items are freely generated in order to induce the right kind of dependencies between events. More specifically, first the graph nodes are freely generated according to the dependencies in $\mathcal{A}$. Then for any pair of nodes, edges connecting the two nodes



(a) (b)

Fig. 8. The (a) AES and (b) domain of configurations for $\mathcal{G}$ of Fig. 7.

are freely generated according to the dependencies in $\mathcal{A}$ and the specific restrictions of the SPO rewriting mechanism. Below $A, B, \ldots$ range over generic sets of events and x over sets of events of cardinality at most 1 (singletons or the empty set). Moreover, by $x < \epsilon$, if $x = \{\epsilon'\}$ we mean that $\epsilon' < \epsilon$, while the relation is intended to hold always if $x = \emptyset$ (i.e. $\emptyset < \epsilon$, for any event ϵ).

Definition 38 (SPO occurrence grammar for an AES). *Let $\mathcal{A} = \langle \mathcal{E}, \leq, \nearrow \rangle$ be an* AES. *The corresponding* SPO *occurrence graph grammar, denoted by $\mathcal{N}_s(\mathcal{A}) = \langle T, P, \pi \rangle$, is defined as follows:*

- *The* type graph $T = \langle N, E, s, t \rangle$ *is defined as below.*
  - *Nodes:*
    $$N = \left\{ \langle x, A, B \rangle : \begin{array}{l} \forall \epsilon \in A \cup B.\ x < \epsilon, \\ \forall \epsilon_a \in A.\ \forall \epsilon_b \in B.\ \epsilon_a \nearrow \epsilon_b, \\ \forall \epsilon_b, \epsilon'_b \in B.\ \epsilon_b \neq \epsilon'_b \Rightarrow \epsilon_b \nearrow \epsilon'_b \end{array} \right\};$$
  - *Edges:*
    $$E = \left\{ \langle x, A, B, n_1, n_2 \rangle : \begin{array}{l} n_i = \langle x_i, A_i, B_i \rangle \in N, \\ \forall \epsilon \in A \cup B.\ x < \epsilon, \\ \forall \epsilon_a \in A.\ \forall \epsilon_b \in B.\ \epsilon_a \nearrow \epsilon_b, \\ \forall \epsilon_b, \epsilon'_b \in B.\ \epsilon_b \neq \epsilon'_b \Rightarrow \epsilon_b \nearrow \epsilon'_b \\ \\ |x_i| \leq |x| \text{ and } x \neq x_i \Rightarrow x \subseteq A_i \text{ for } i \in \{1,2\} \\ A \subseteq A_1 \cap A_2 \\ B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2) \end{array} \right\};$$
  - *Source and target functions:*
    $$s(\langle x, A, B, n_1, n_2 \rangle) = n_1 \quad \textit{and} \quad t(\langle x, A, B, n_1, n_2 \rangle) = n_2.$$
- *The set of productions $P = \mathcal{E}$, and for any event $\epsilon \in \mathcal{E}$ the corresponding production $\pi(\epsilon) = L_\epsilon \rightarrowtail R_\epsilon$ is defined as follows:*
  - $|L_\epsilon| = \{n = \langle x, A, B \rangle,\ e = \langle x, A, B, n_1, n_2 \rangle \mid \epsilon \in A \cup B\}$
  - $|R_\epsilon| = \{n = \langle x, A, B \rangle,\ e = \langle x, A, B, n_1, n_2 \rangle \mid \epsilon \in x \cup A\}$

  with the graphical structure inherited from T. The typing is the inclusion and the (partial) inclusion of L_ϵ in R_ϵ is the obvious one.

A node in the type graph T is a triple $n = \langle x, A, B \rangle$. The set x might contain the event which generates the node n or might be empty if the node is in the start graph, A is the set of events which preserve the node n and B is the set of events which consume n. Clearly, the event in x, if any, must be a cause for every event in $A \cup B$, the events in A must be in asymmetric conflict with the events in B, and the events in B must be pairwise in conflict (represented as an asymmetric conflict in both directions, i.e., $\epsilon_b \nearrow \epsilon'_b$ and $\epsilon'_b \nearrow \epsilon_b$ for $\epsilon_b, \epsilon'_b \in B$).

An edge in the type graph is a tuple $e = \langle x, A, B, n_1, n_2 \rangle$. The meaning of x, A, B is the same as for nodes. The components n_1 and n_2 are intended to represent the source and target nodes of edge e. They are subject to requirements which arise from the specific features of the SPO rewriting mechanism. First, if edge e is in the start graph (i.e., $x = \emptyset$) then also the source and target nodes n_i must be in the start graph (i.e., $x_i = \emptyset$), as imposed by condition $|x_i| \leq |x|$. Moreover, either $x_i = x$ or $x \subseteq A_i$, since the event which produces an edge must produce or preserve the source/target nodes. Any event which preserves the edge must also preserve the source/target nodes, hence $A \subseteq A_1 \cap A_2$. Any event which consumes the edge must preserve or consume the source/target nodes, hence $B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2)$.

Proposition 3 (Well-definedness of $\mathcal{N}_s$). *For any* AES $\mathcal{A}$ *the structure* $\mathcal{N}_s(\mathcal{A})$ *is a well-defined occurrence grammar.*

Proof. The fact that $\mathcal{N}_s(\mathcal{A})$ satisfies properties (1)-(3) of Definition 21 easily follows by construction and the proof can be adapted from the one for contextual nets (Corollary 7.1 in [8]). The only delicate point is the well-definedness of the left-hand side and right-hand side graphs of productions $\pi(\epsilon) = L_\epsilon \rightarrowtail R_\epsilon$ for $\epsilon \in \mathcal{E}$. For instance, let us prove that L_ϵ is a well-defined graph. Let $e = \langle x, A, B, n_1, n_2 \rangle$ be an edge in L_ϵ and let us show that its source n_1 and target n_2 are in L_ϵ as well. Since $e \in L_\epsilon$, by construction, $\epsilon \in A \cup B$. From the definition of edges, we have that $A \subseteq A_1 \cap A_2$ and $B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2)$. Hence $A \cup B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2)$ and thus $\epsilon \in A_i \cup B_i$ for $i \in \{1, 2\}$. Therefore $n_1, n_2 \in L_\epsilon$.

Similarly, if $e \in R_\epsilon$ we can show that $n_1, n_2 \in R_\epsilon$, and thus R_ϵ is a well-defined graph. ☐

Observe that constructing the canonical grammar associated to an AES and then considering the corresponding AES we obtain again the original structure.

Proposition 4. *Let* $\mathcal{A}$ *be an* AES*. Then* $\mathcal{E}_s(\mathcal{N}_s(\mathcal{A})) = \mathcal{A}$.

Proof. The result follows immediately by construction, since the graph items in $\mathcal{N}_s(\mathcal{A})$ are generated to induce exactly the dependencies between events which were originally present in $\mathcal{A}$. ☐

We conclude with the main result, stating that the construction of the occurrence grammar associated to an AES is functorial and such functor is left adjoint to $\mathcal{E}_s$. This establishes a coreflection between **OGG** and **AES**. For any AES $\mathcal{A}$, the component at $\mathcal{A}$ of the unit of the adjunction is the identity.

Given an AES $\mathcal{A} = \langle \mathcal{E}, \leq, \nearrow \rangle$ and an event $\epsilon \in \mathcal{E}$, we define the set of *consequences* $\lceil\{\epsilon\}\rceil$, as follows (considering the singleton $\{\epsilon\}$ instead of ϵ itself will later simplify the notation).

$$\lceil\{\epsilon\}\rceil = \{\epsilon' \in \mathcal{E} \mid \epsilon < \epsilon'\}.$$

This function is extended also to the empty set, by $\lceil\emptyset\rceil = \mathcal{E}$. We use the same notation for occurrence grammars, referring to the underlying AES.

The next technical lemma will be used in the coreflection proof. It immediately follows from the analogous result for contextual nets (Lemma 7.2 in [8]) by exploiting the relation established by functor Net in Section 4.3.

Lemma 5. *Let* $\mathcal{O}_0$ *and* $\mathcal{O}_1$ *be occurrence graph grammars and let* $h : \mathcal{O}_0 \to \mathcal{O}_1$ *be a morphism. For all* $x_0 \in T_0$ *and* $x_1 \in T_1$*, if* $h_T(x_0, x_1)$ *then*

1. $h_P({}^\bullet x_0) = {}^\bullet x_1$;
2. $x_0{}^\bullet = h_P^{-1}(x_1{}^\bullet) \cap \lceil {}^\bullet x_0 \rceil$;
3. $\underline{x_0} = h_P^{-1}(\underline{x_1}) \cap \lceil {}^\bullet x_0 \rceil$.

Theorem 3 (coreflection between OGGRel and AES). *The construction* $\mathcal{N}_s$ *extends to a functor that is left adjoint to* $\mathcal{E}_s$.

Proof. Consider any asymmetric event structure $\mathcal{A} = \langle \mathcal{E}, \leq, \nearrow \rangle$. Then let $\mathcal{N}_s(\mathcal{A}) = \langle T, G_s, P, \pi \rangle$ be the occurrence graph grammar as defined in Definition 38. We must show the that for any occurrence grammar $\mathcal{O}_0 = \langle T_0, G_{s0}, P_0, \pi_0 \rangle$ and for any morphism $g : \mathcal{A} \to \mathcal{E}_s(\mathcal{O}_0)$ there exists a unique morphism $h : \mathcal{N}_s(\mathcal{A}) \to \mathcal{O}_0$, such that the following diagram commutes:

$$\begin{tikzcd}
\mathcal{A} \arrow[r, "id_{\mathcal{A}}"] \arrow[dr, "g"'] & \mathcal{E}_s(\mathcal{N}_s(\mathcal{A})) = \mathcal{A} \arrow[d, dotted, "\mathcal{E}_s(h)"] \\
& \mathcal{E}_s(\mathcal{O}_0)
\end{tikzcd}$$

First notice that, by the above requirement, the component $h_P : P \to P_0$ on productions is uniquely determined by g:

$$h_P = g.$$

We must show, now, the existence and uniqueness of a graph relation $h_\tau : T \times T_0$ such that $\langle h_P, h_\tau \rangle$ is a morphism in **OGGRel**.

Existence
Guided by the result in Lemma 5, let us consider the graph relation h_τ identified by the properties in such lemma, specialised to our case, namely:

- h_τ over nodes:
  For all $n = \langle x, A, B \rangle \in T$ and $n_0 \in T_0$, we let $h_\tau(n, n_0)$ iff
  1. $h_P(x) = {}^\bullet n_0$
  2. $A = h_P^{-1}(\underline{n_0}) \cap \lceil x \rceil$
  3. $B = h_P^{-1}(n_0{}^\bullet) \cap \lceil x \rceil$
- h_τ over edges:
  For all $e = \langle x, A, B, n_1, n_2 \rangle \in T$ and $e_0 \in T_0$, we let $h_\tau(e, e_0)$ iff $h_\tau(n_1, s(e_0))$, $h_\tau(n_2, t(e_0))$ and conditions (1), (2), (3) above are satisfied.

Let us show that the pair $h = \langle h_P, h_\tau \rangle$ is indeed a morphism in **OGGRel**, by verifying the conditions on the preservation of the start graph and of the pre-set, post-set and context of productions in Definition 25.

Preservation of the start graph: $h_\tau(G_s, G_{s0})$
We prove the result first for the set of nodes and then for the edges. Hereafter, the symbols n and e (possibly with subscripts) ranges over nodes and edges, respectively.

(Nodes) Let $n = \langle x, A, B \rangle \in |G_s|$, $n_0 \in T_0$ and assume that $h_\tau(n, n_0)$. Since x is minimal, by construction $x = \emptyset$. Hence, by definition of h_τ, ${}^\bullet n_0 = \emptyset$ and thus $n_0 \in |G_{s0}|$.
Vice versa, given $n_0 \in |G_{s0}|$ we must find a unique $n \in G_s$ such that $h_\tau(n, n_0)$. Let

$$A = h_P^{-1}(\underline{n_0}) \qquad \text{and} \qquad B = h_P^{-1}(n_0{}^\bullet).$$

By definition of AES-morphism, since $q_0 \nearrow q'_0$ for all $q_0, q'_0 \in n_0{}^\bullet$, $q_0 \neq q'_0$, we have that $q \nearrow q'$ for all $q, q' \in B$, $q \neq q'$. Similarly, since $q_0 \nearrow q'_0$ for all $q_0 \in \underline{n_0}$, $q'_0 \in n_0{}^\bullet$, we deduce that $q \nearrow q'$ for all $q \in A$ and $q' \in B$. Hence there is a node $n = \langle \emptyset, A, B \rangle \in |G_s|$ and $h_\tau(n, n_0)$. Uniqueness of n follows immediately from the definition of h_τ.

(Edges) Take an edge $e = \langle x, A, B, n_1, n_2 \rangle \in |G_s|$. If $h_\tau(e, e_0)$ for some $e_0 \in T_0$, then, reasoning as above for nodes, we conclude that $e_0 \in |G_{s0}|$. Vice versa, let $e_0 \in |G_{s0}|$ and let us prove that there exists a unique $e \in |G_s|$ such that $h_\tau(e, e_0)$. Since the start state G_{s0} is a well defined graph $s(e_0), t(e_0) \in |G_{s0}|$. Therefore by the proof on nodes above, there are unique $n_1, n_2 \in |G_s|$ such that $h_\tau(n_1, s(e_0))$ and $h_\tau(n_2, t(e_0))$. Define

$$A = h_P^{-1}(\underline{e_0}) \qquad \text{and} \qquad B = h_P^{-1}(e_0{}^\bullet).$$

and let $e = \langle \emptyset, A, B, n_1, n_2 \rangle$. Let us prove that e is a well-defined edge, showing that the properties required over edges in the construction of $\mathcal{N}_s(\mathcal{A})$ (see Definition 38) hold.

1. $|x_i| \leq |x|$ and $x \neq x_i \Rightarrow x \subseteq A_i$
   Trivial, noticing that $x = x_1 = x_2 = \emptyset$.
2. $A \subseteq A_1 \cap A_2$
   Recall that $A = h_P^{-1}(\underline{e_0})$, $A_1 = h_P^{-1}(\underline{s(e_0)})$ and $A_2 = h_P^{-1}(\underline{t(e_0)})$. From the fact that a production which preserves an edge must preserve also its source and target nodes we have that $\underline{e_0} \subseteq \underline{s(e_0)}$ and $\underline{e_0} \subseteq \underline{t(e_0)}$. Thus the desired result immediately follows.
3. $B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2)$
   Recall that $B = h_P^{-1}(e_0{}^\bullet)$, $A_1 = h_P^{-1}(\underline{s(e_0)})$ and $B_1 = h_P^{-1}(s(e_0)^\bullet)$. Since a production which consumes an edge must preserve or consume its source and target nodes we have that $e_0{}^\bullet \subseteq \underline{s(e_0)} \cup s(e_0)^\bullet$. Hence $B \subseteq (A_1 \cup B_1)$. In the same way, $B \subseteq (A_2 \cup B_2)$ an thus we conclude.

Now, by construction $e \in |G_s|$ and $h_\tau(e, e_0)$. As in the previous case, uniqueness of e follows immediately from the definition of h_τ.

Preservation of the pre-, post-, context of productions

Now, let $q \in P$ be any production. We must show that (a) $h_\tau({}^\bullet q, {}^\bullet h_P(q))$, (b) $h_\tau(\underline{q}, \underline{h_P(q)})$, (c) $h_\tau(q^\bullet, h_P(q)^\bullet)$. This is done first for nodes and then for edges, separately.

(Nodes) a. $h_\tau({}^\bullet q, {}^\bullet h_P(q))$.

Let $n = \langle x, A, B\rangle \in {}^\bullet q$ and let $h_\tau(n, n_0)$. Then, $q \in B$ and thus, by definition of h_τ, $h_P(q) \in n_0{}^\bullet$, or equivalently $n_0 \in {}^\bullet h_P(q)$.
For the converse, let $n_0 \in {}^\bullet h_P(q)$ and let us look for a node $n \in {}^\bullet q$ such that $h_\tau(n, n_0)$. Define $x = h_P^{-1}({}^\bullet n_0) \cap \lfloor q \rfloor$. Since $\mathcal{O}_0$ is an occurrence grammar $|{}^\bullet n_0| \leq 1$ and thus $|x| \leq 1$ (more precisely $x = \emptyset$ if $n_0 \in |G_{s0}|$, otherwise, x contains the unique $q' \leq q$, such that $h_P(q') = q_0$, with ${}^\bullet n_0 = \{q_0\}$). Consider

$$A = h_P^{-1}(\underline{n_0}) \cap \lceil x \rceil \quad \text{and} \quad B = h_P^{-1}(n_0{}^\bullet) \cap \lceil x \rceil.$$

As above, since $q_0 \nearrow q_0'$ for all $q_0, q_0' \in n_0{}^\bullet$, $q_0 \neq q_0'$ and $q_0 \nearrow q_0'$ for all $q_0 \in \underline{n_0}$, $q_0' \in n_0{}^\bullet$, the defining properties of AES-morphisms allow to deduce that $n = \langle x, A, B\rangle \in T$ is a well defined node and, by definition of h_τ, we have $h_\tau(n, n_0)$. Clearly $q \in \lceil x \rceil$, thus $q \in B$ and therefore $n \in {}^\bullet q$ and $n_0 \in h_\tau({}^\bullet q)$. Again, uniqueness of n follows by the definition of h_τ.

b. $h_\tau(\underline{q}, \underline{h_P(q)})$.
   Analogous to the previous case.

c. $h_\tau(q^\bullet, h_P(q)^\bullet)$.
   If $n = \langle x, A, B\rangle \in q^\bullet$, that is $x = \{q\}$, and $h_\tau(n, n_0)$, then, by definition of h_τ, we have $n_0 \in h_P(q)^\bullet$.
   For the converse, let $n_0 \in h_P(q)^\bullet$ and consider

$$A = h_P^{-1}(\underline{n_0}) \cap \lceil \{q\} \rceil \quad \text{and} \quad B = h_P^{-1}(n_0{}^\bullet) \cap \lceil \{q\} \rceil.$$

   Then $n = \langle \{q\}, A, B\rangle \in q^\bullet$ is a well-defined node and, by definition of h_τ, we have $h_\tau(n, n_0)$ and n is uniquely determined in $q^\bullet$.

(Edges)

a. $h_\tau({}^\bullet q, {}^\bullet h_P(q))$.
   Let $e = \langle x, A, B, n_1, n_2\rangle \in {}^\bullet q$ and let $h_\tau(e, e_0)$. Then we conclude, as for nodes, that $e_0 \in {}^\bullet h_P(q)$.
   For the converse, let $e_0 \in {}^\bullet h_P(q)$. Thus $s(e_0), t(e_0) \in {}^\bullet h_P(q) \cup \underline{h_P(q)}$. Therefore, by the proof for nodes above, there are unique $n_1, n_2 \in {}^\bullet q \cup \underline{q}$, with $n_i = \langle x_i, A_i, B_i\rangle$ such that $h_\tau(n_1, s(e_0))$ and $h_\tau(n_2, t(e_0))$. Define $x = h_P^{-1}({}^\bullet e_0) \cap \lfloor q \rfloor$ and

$$A = h_P^{-1}(\underline{e_0}) \cap \lceil x \rceil \qquad \text{and} \qquad B = h_P^{-1}(e_0{}^\bullet) \cap \lceil x \rceil.$$

   then we only need to show that $e = \langle x, A, B, n_1, n_2\rangle$ is an edge in T. The fact that $h_\tau(e, e_0)$ and the uniqueness of e will follow by construction. To this aim let us prove the properties required for edges in the construction of $\mathcal{N}_s(\mathcal{A})$ (see Definition 38):

(1) $|x_i| \leq |x|$ and $x \neq x_i \Rightarrow x \subseteq A_i$
Note that if $x = \emptyset$, then necessarily $x_i = \emptyset$. In fact, if $x = \emptyset$ then e_0 is in the start graph and thus also $s(e_0)$, $t(e_0)$ and hence n_1, n_2 are in the start graph. Thus $|x_i| \leq |x|$ holds.
For the second part, assume $x \neq x_1$ and let us show that $x \subseteq A_1$. By the above considerations, $x \neq x_1$ implies $x = \{q'\} \neq \emptyset$. Hence, by construction, $e_0 \in h_P(q')^\bullet$. Observe that $s(e_0) \notin h_P(q')^\bullet$, otherwise we would have $x_1 = \{q'\} = x$, contradicting the hypothesis. Hence, since a production which generates an edge must produce or preserve the source and target nodes, we must have $s(e_0) \in \underline{h_P(q')}$. Thus, by the results above on nodes, since $h_\tau(n_0, s(e_0))$, we deduce $n_1 \in \underline{q'}$. By definition of $\mathcal{N}_s(\mathcal{A})$, this means that $q' \in A_1$, i.e., $x \subseteq A_1$.
The proof that $x \neq x_2$ implies $x \subseteq A_2$ is totally analogous.

(2) $A \subseteq A_1 \cap A_2$
Recall that $A = h_P^{-1}(\underline{e_0}) \cap \lceil x \rceil$, $A_1 = h_P^{-1}(\underline{s(e_0)}) \cap \lceil x_1 \rceil$, $A_2 = h_P^{-1}(\underline{t(e_0)}) \cap \lceil x_2 \rceil$. From the fact that a production which preserve an edge must preserve also its source and target nodes we have that $\underline{e_0} \subseteq \underline{s(e_0)}$ and $\underline{e_0} \subseteq \underline{t(e_0)}$. Furthermore, by (1) above, $\lceil x \rceil \subseteq \lceil x_i \rceil$ $(i \in \{1, 2\})$. Hence the desired result immediately follows.

(3) $B \subseteq (A_1 \cup B_1) \cap (A_2 \cup B_2)$
Analogous to (2) above.

b. $h_\tau(\underline{q}, \underline{h_P(q)})$.
Analogous to the previous case.

c. $h_\tau(q^\bullet, h_P(q)^\bullet)$.
If $e = \langle x, A, B, n_1, n_2 \rangle \in q^\bullet$ and $h_\tau(e, e_0)$, we conclude as for nodes that $e_0 \in h_P(q)^\bullet$.
Vice versa, let $e_0 \in h_P(q)^\bullet$. Therefore $s(e_0), t(e_0) \in \underline{h_P(q)} \cup h_P(q)^\bullet$. Thus, by the proof for nodes above, there are unique $n_0, n_1 \in \underline{q} \cup q^\bullet$, with $n_i = \langle x_i, A_i, B_i \rangle$, such that $h_\tau(n_1, s(e_0))$ and $h_\tau(n_2, t(e_0))$. As in the proof for nodes, consider

$$A = h_P^{-1}(\underline{e}) \cap \lceil \{q\} \rceil \quad \text{and} \quad B = h_P^{-1}(e^\bullet) \cap \lceil \{q\} \rceil.$$

Then take $e = \langle \{q\}, A, B, n_1, n_2 \rangle \in q^\bullet$. We only need to show that e is a well-defined edge in T. The fact that $h_\tau(e, e_0)$ and the uniqueness of e will follow by construction.
The properties required for edges in the construction of $\mathcal{N}_s(\mathcal{A})$ (see Definition 38), can be proved as before, apart from (1) which is explicitly discussed below.

(1) $|x_i| \leq |\{q\}|$ and $x_i \neq \{q\} \Rightarrow \{q\} \subseteq A_i$.
The first part is trivially true since the x_i's are singletons and $|\{q\}| = 1$.
As for the second part, assume, for instance, $x_1 \neq \{q\}$. Hence $s(e_0) \notin h_P(q)^\bullet$, otherwise we would have by construction $x_1 = \{q\}$. As in the previous case (1) above, from this fact we deduce $s(e_0) \in \underline{h_P(q)}$ and thus $q \in A_1$, i.e., $\{q\} \subseteq A_1$, as desired. The case in which $x_2 \neq \{q\}$ is completely analogous.

This concludes the proof that h is a grammar morphism.

Uniqueness
Uniqueness of the graph relation h_τ such that $\langle h_P, h_\tau \rangle$ is an occurrence graph grammar morphism in **OGGRel** can be proved by resorting to Lemma 5.

More precisely, let $h'_\tau : T \times T_0$ be another graph relation such that $\langle h_P, h'_\tau \rangle$ is a morphism and $h'_\tau(n, n_0)$, then, necessarily, by Lemma 5, $h_\tau(n, n_0)$. Hence $h'_\tau \subseteq h_\tau$, componentwise.

As for the converse inclusion, let us treat separately the case of nodes and of edges.

(nodes) Let $h_\tau(n, n_0)$, with $n = \langle x, A, B \rangle$. Then, if $x = \emptyset$, by properties of occurrence grammar morphisms, $n_0 \in |G_{s0}|$. Therefore there must be $n' \in |G_s|$ such that $h'_\tau(n', n_0)$. But, by Lemma 5 and definition of h_τ, $\underline{n'} = h_P^{-1}(\underline{n_0}) = A$ and similarly $n'^\bullet = h_P^{-1}({n_0}^\bullet) = B$. Therefore $n' = \langle \emptyset, A, B \rangle = n$ and thus $h'_\tau(n, n_0)$. An analogous reasoning allows us to conclude when $x = \{q\}$.

(edges) Same as above, just remembering that h_T is a relational graph morphism and exploiting the result about nodes. More precisely, let $h_\tau(e, e_0)$, with $e = \langle x, A, B, n_1, n_2 \rangle$ and assume $x = \emptyset$. By definition of graph relation

$h_\tau(n_1, s(e'))$ and $h_\tau(n_2, t(e'))$ (†)

Moreover, as above we can deduce that there exists $e' = \langle \emptyset, A, B, m_1, m_2 \rangle \in T$ such that $h'_\tau(e', e_0)$, and, being h'_τ a graph relation

$h'_\tau(m_1, s(e'))$ and $h'_\tau(m_2, t(e'))$

which, by the result already proved for nodes, implies

$h_\tau(m_1, s(e'))$ and $h_\tau(m_2, t(e'))$ (‡)

By (†) and (‡), since, by definition of relational morphism, h_τ is injective on the initial graph we must have $n_i = m_i$ ($i \in \{1, 2\}$). Therefore $e = e'$ and thus $h'_\tau(e, e_0)$, as desired.

Also in this case an analogous reasoning allows us to conclude when $x = \{q\}$. □

8 Possible variations

In this section we present some possible extensions of the work in this paper, first to general, non semi-weighted SPO graph grammars and then to graph grammars in the DPO approach, discussing the results which can be generalised and the corresponding limitations.

8.1 Unfolding 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. It can be recovered at the price of considering a restricted, still meaningful, subclass of grammar morphisms. Unfortunately, in this way the adjunction with domains is lost.

First, we notice that in the characterisation of the unfolding as a coreflection (Theorem 2) 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 2 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 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. 9, 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 33, 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 2 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, $g^L_T(x) = g^L_T(y) = A$ and $g^L_T(x) = g^L_T(y) = B$ (see the bottom row of the diagram in Fig. 9). Consider the pullback-retyping diagram in Fig. 9, 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 2 reveals that the only difficulty

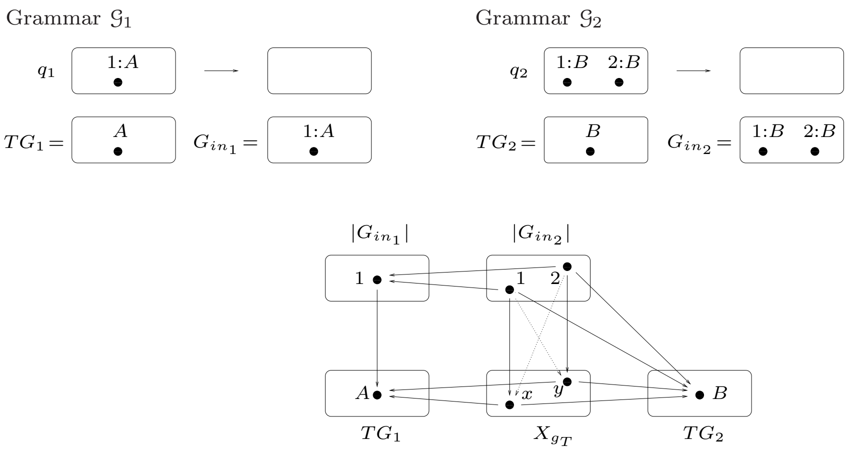


Fig. 9. The grammars $\mathcal{G}_1$ and $\mathcal{G}_2$, and the pullback-retyping diagram for their start graphs.

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 2, the existence of a unique morphism $h : \mathcal{O} \to \mathcal{U}_s(\mathcal{G})$ making the following diagram commute:

$$\begin{tikzcd} \mathcal{U}_s(\mathcal{G}) \arrow[r, "\varphi_{\mathcal{G}}"] & \mathcal{G} \\ \mathcal{O} \arrow[u, dotted, "h"] \arrow[ur, "g"'] & \end{tikzcd}$$

Dealing with general grammars, we must consider slightly more general notions of safe and occurrence grammar, allowing the preserved items of a production to be non-injectively typed.

Definition 39 (extended safe and occurrence grammar). *An* extended safe grammar *is a grammar $\mathcal{G} = \langle T, G_s, P, \pi \rangle$ satisfying (i) for all H such that $G_s \Rightarrow^* H$, H is injective, and (ii)for each $q \in P$, the typing t_{L_q} is injective on the "consumed part" $|L_q| - |dom(r_q)|$, and t_{R_q} is injective on the "produced part" $|R_q| - r_q(|dom(r_q)|)$.*

An extended occurrence grammar *is an extended safe grammar satisfying the conditions (1)-(3) in Definition 21. We denote by $\mathbf{OGG}_*$ the full subcategory of $\mathbf{GG}$ with extended occurrence grammars as objects.*

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 2 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 40 (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 4 (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 2 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 regard a remarkable class of grammar morphisms. In particular they comprise the morphisms adopted in [20] 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.

Unfortunately, morphisms in the range of the functor $\mathcal{N}_s$ mapping each AES to a canonical occurrence grammar are not in $\mathbf{GG}^R$, and thus the construction in Section 7 does not induce a functor in this restricted setting, i.e., $\mathcal{N}_s$ does not restrict to a functor from $\mathbf{AES}$ to $\mathbf{OGG}^R$.

8.2 Unfolding of DPO grammars

A second 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

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

can be constructed in T-**Graph**, where both squares are required to be pushouts.

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 the so-called 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 41 (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 valid 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 coreflections:

$$\begin{tikzcd}
\mathbf{SGG}^{-} \arrow[r, shift left, "\mathcal{U}'_s"'] \arrow[r, phantom, "\perp"] & \mathbf{OGG}^{-} \arrow[l, hook', shift left] \arrow[r, shift left, "\mathcal{E}'_s"'] \arrow[r, phantom, "\perp"] & \mathbf{AES} \arrow[l, shift left, "\mathcal{N}'_s"'] \arrow[r, shift left, "\mathcal{L}_a"'] \arrow[r, phantom, "\sim"] & \mathbf{Dom} \arrow[l, shift left, "\mathcal{P}"']
\end{tikzcd}$$

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

This is mostly trivial as all functors above, apart from $\mathcal{N}'_s$, are simply the restrictions of the corresponding functors defined in this paper. The only delicate point is to define functor $\mathcal{N}'_s$ in order to ensure that any AES is mapped to a node-preserving occurrence grammar. This requires to change Definition 38 in order to enforce this property. More precisely, the definition of the set of nodes of the type graph becomes:

$$N = \{\langle x, A, \emptyset\rangle : \forall \epsilon \in A.\ x < \epsilon\}$$

Then, it is not difficult to see that everything else goes through.

When we consider general DPO graph grammars, unfortunately the situation is sensibly more complex. For instance, consider the safe DPO grammar in Fig. 10, which is obtained form the SPO grammar in Fig. 7 by transforming each SPO rule into the corresponding DPO rule, as explained above.

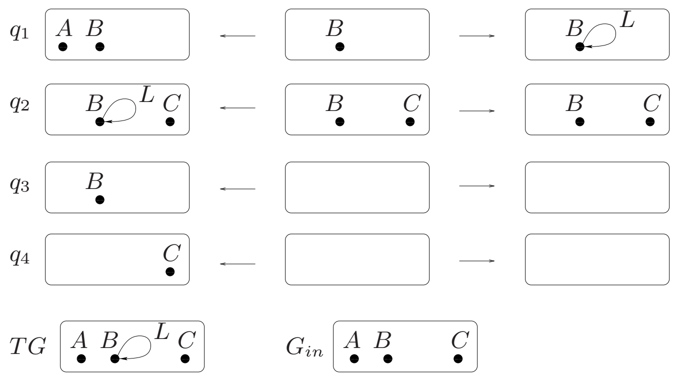


Fig. 10. A safe DPO graph grammar.

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 classical 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 $e_1, \ldots, e_n$, 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. 10 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.

$$
\begin{tikzcd}
\begin{array}{c}\textbf{DPO Graph}\\ \textbf{Grammars}\end{array} \arrow[r, shift right, "\mathcal{U}_g"'] \arrow[r, phantom, "\perp"] & \begin{array}{c}\textbf{Occurrence}\\ \textbf{Grammars}\end{array} \arrow[l, hook', shift right] \arrow[r, "\mathcal{E}_g"'] & \begin{array}{c}\textbf{Inhibitor Event}\\ \textbf{Structures}\end{array} \arrow[r, shift right, "\mathcal{L}_i"'] \arrow[r, phantom, "\perp"] & \textbf{Domains} \arrow[l, shift right, "\mathcal{P}_i"']
\end{tikzcd}
$$

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.

9 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.

$$
\begin{tikzcd}
\textbf{SGG} \arrow[r, shift right, "\mathcal{U}_s"'] \arrow[r, phantom, "\perp"] & \textbf{OGG} \sim \textbf{OGGRel} \arrow[l, hook', shift right] \arrow[r, shift right, "\mathcal{E}_s"'] \arrow[r, phantom, "\perp"] \arrow[dr, "\mathsf{Net}"'] & \textbf{AES} \arrow[l, shift right, "\mathcal{N}_s"'] \arrow[r, shift right, "\mathcal{L}_a"'] \arrow[r, phantom, "\sim"] & \textbf{Dom} \arrow[l, shift right, "\mathcal{P}"'] \\
& & \textbf{OCN} \arrow[u, "\mathcal{E}_a"'] &
\end{tikzcd}
$$

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 [22], which in turn were characterised as special classes of graph grammars in [31].

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 [16] or adhesive categories [23]). 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. Kreowski, 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. 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.
11. 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.
12. 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.
13. 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.
14. A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.
15. 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 Rozenberg [32], chapter 3.
16. 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.
17. 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 Rozenberg [32], chapter 4.
18. 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.
19. U. Golz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.
20. 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.
21. R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123:1–16, 1995.
22. M. Korff. *Generalized graph structure grammars with applications to concurrent object-oriented systems*. PhD thesis, Technische Universität Berlin, 1996.
23. S. Lack and P. Sobociński. Adhesive categories. In I. Walukiewicz, editor, *Proceedings of FoSSaCS'04*, volume 2987 of *LNCS*, pages 273–288. Springer Verlag, 2004.
24. R. Langerak. *Transformation and Semantics for LOTOS*. PhD thesis, Department of Computer Science, University of Twente, 1992.
25. M. Löwe. Algebraic approach to single-pushout graph transformation. *Theoretical Computer Science*, 109:181–224, 1993.
26. 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.
27. J. Meseguer, U. Montanari, and V. Sassone. On the semantics of Place/Transition Petri nets. *Mathematical Structures in Computer Science*, 7:359–397, 1997.

28. U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32(6), 1995.
29. M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.
30. G. M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, 138(2):425–454, 1995.
31. L. Ribeiro. *Parallel Composition and Unfolding Semantics of Graph Grammars.* PhD thesis, Technische Universität Berlin, 1996.
32. Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations.* World Scientific, 1997.
33. 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.
34. 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.