Frm Ptr Nts t Grph Trnsfrmtn Sstms*

A Contribution in Honour of Hans-Jörg Kreowski

Paolo Baldan, Andrea Corradini, Fabio Gadducci, and Ugo Montanari

Abstract. Hans-Jörg Kreowski was among the first researchers to point out that P/T Petri nets can be interpreted as instances of Graph Transformation Systems, a fact now considered folklore. We elaborate on this observation, discussing how several different models of Petri nets can be encoded faithfully into Graph Transformation Systems. The key idea we pursue is that the net encoding is uniquely determined, and distinct net models are mapped to alternative approaches to graph transformation.

1 Introduction

The success of Petri nets as specification formalism for concurrent or distributed systems is due (among other things) to the fact that they can describe in a natural way the evolution of systems whose states have a distributed nature. For example, in a Place/Transition net like the one depicted in Fig. 1, a state of the system is represented by a marking, i.e., a set of tokens distributed among a set of places. Hence the state is intrinsically distributed, thus allowing for an easy explicit representation of phenomena like *mutual exclusion*, *concurrency*, *causality*, and *non-determinism*. Nets and their semantics are therefore a reference point for any formalism intended to describe concurrent and distributed systems, and thus also for Graph Transformation Systems (GTSs).

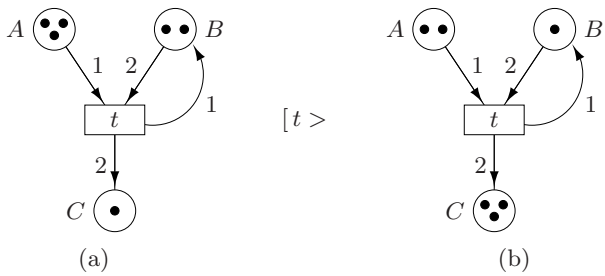


Fig. 1. (a) A marked P/T net. (b) The marking after the firing of transition t.

Indeed, it belongs to the folklore that Graph Transformation Systems can be seen as a generalisation of Petri nets. The first formalization of this intuition, to our knowledge, was proposed by Hans-Jörg Kreowski in [1] using the double-pushout (DPO) approach, and it is illustrated in Fig. 2. The marked net

* Research partially supported by the EU FP6-IST IP 16004 SENSORIA.

F. Drewes, A. Habel, B. Hoffmann, D. Plump (Eds.): Manipulation of Graphs, Algebras and Pictures. Essays Dedicated to Hans-Jörg Kreowski on the Occasion of His 60th Birthday, pp. 39–58, 2009.

of Fig. 1(a) is represented in Fig. 2 by the graph $Kr(M_0)$ having three kinds of nodes (for transitions, places, and tokens, respectively) and where edges connect either places and transitions (modelling the causal dependency relation) or tokens and places (determining the place where a token lies). Transition t is represented by rule $Kr(t)$ (the top row of the figure): The rule does not modify the topological structure of the net (nodes and edges corresponding to places, transitions and causal dependency relation are also in the interface), but only deletes and creates the nodes representing tokens together with the edges connecting them to places. It is easy to check that the rule is applicable to graph $Kr(M_0)$ (the gluing conditions are satisfied), and since the two squares in the figure are pushouts, that $Kr(M_0) \overset{Kr(t)}{\Longrightarrow} Kr(M_1)$; moreover, the derived graph $Kr(M_1)$ represents the marking M_1, such that $M_0 \,[t\rangle\, M_1$.

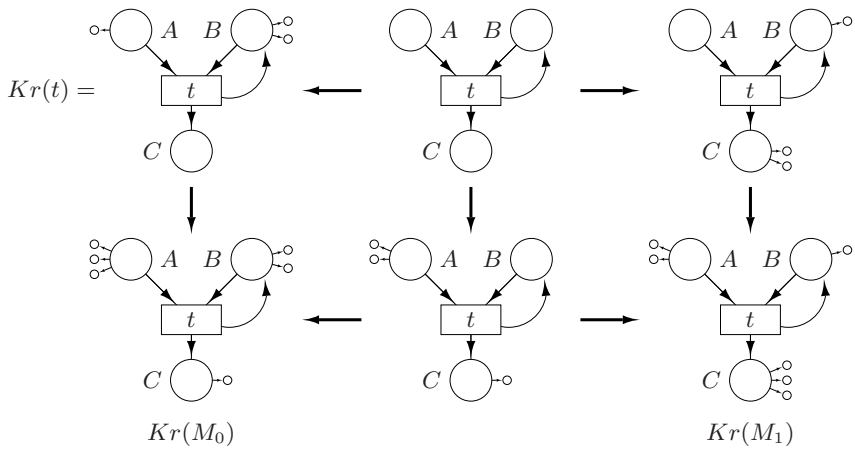


Fig. 2. Encoding of nets as grammars according to Kreowski.

Several encodings of Petri nets as GTSs have been proposed since then, and it is impossible even to summarize them here: for some of the earliest, see [2] and the references therein. In this paper we elaborate on this idea, starting from the observation that P/T nets are only one (a noticeable one) among the alternative models of Petri net which have been proposed along the years. Sticking to "low level" Petri nets, other models of nets may allow at most one token at a time in a place, as for *Condition/Event (C/E) nets* [3] or *Elementary Net Systems (*ENS*)* [4], and correspondingly a transition can fire only if the post-conditions are empty. In the so-called *Consume-Produce-Read (*CPR*) nets* [5], more permissively, the transition can fire anyhow, but the token produced on a place is "coalesced" with a possibly pre-existing token [5]. Orthogonally, nets of all kinds can be equipped with *read* or *inhibitor arcs*, specifying that the presence or the absence of a token on a place is necessary for firing, but it does not affect the

result [6–10]. Another type of arcs, called *reset arcs* [11], allows to specify that the firing of a transition deletes all the tokens, if any, from a given place.

What about representing these models of nets as GTSs? In principle, all of them can be encoded using DPO rewriting, because the latter is Turing complete [12]. We prefer to follow a different approach, which on the one hand allows us to keep the encoding very simple for all the models of nets mentioned above, and on the other hand exploits the fact that also for GTSs alternative formalisms have been proposed. From the GTS side we shall stick to the family of algebraic approaches, among which we consider the classical single- and double-pushout approaches [13, 14], and the less known Subobject Transformation Systems [15]. The latter basically consists of rewriting in the lattice of subgraphs of a given graph, and it turns out to be the natural framework for encoding net models which allow at most one token on a place (where a state is a subset of places).

We encode nets using a very simple kind of graphs, containing nodes and unary edges only. A marking of a net is represented by a set of edges, one for each token, each attached to a node representing a place. It is thus reminiscent of the encoding by Kreowski discussed above, even if the transitions are not represented explicitly in the states: They are encoded only as rules of the GTS. Interestingly, inhibitor and reset arcs can be encoded exactly in the same way: The different behaviour is determined by the choice of the GTS approach.

The following table summarizes the results we shall present. For each of the three basic net models, we indicate the GTS approach that can be used to encode it in presence of read, inhibitor and/or reset arcs: note that we do not allow for nets which include both inhibitor and reset arcs.

	Read arcs	Read + Inhibitor	Read + Reset
P/T nets	DPO or SPO	DPO	SPO
ENS	STS or $\text{STS}^{\sqsubseteq}$	STS	$\text{STS}^{\sqsubseteq}$
CPR nets	STS_m or $\text{STS}_m^{\sqsubseteq}$	STS_m	$\text{STS}_m^{\sqsubseteq}$

Table 1. Summary of the proposed encodings.

The few variants of the STS approach referred to in the table will be introduced later on. The encodings of P/T Petri nets with read, inhibitor and reset arcs as GTSs were originally discussed in [16]. The present paper provides a systematic view of such encodings, viewing them in a much more general framework which recomprises Elementary Net Systems and CPR nets.

The paper is structured as follows. Section 2 presents the three GTS approaches we deal with in our work, and it is complemented in Section 3 by the kinds of nets for which we present an encoding. Section 4 discusses these encodings, and the correspondence between alternative net models and GTS approaches. Section 5 draws some conclusions and offers pointers to future works.

2 Algebraic approaches to graph transformation

This section introduces some basic notions concerning the algebraic formalisms for graph rewriting considered in the paper. We concentrate on *typed Graph Transformations Systems* (GTSs), both in the *single-pushout* (SPO) [13, 17] and the *double-pushout* (DPO) [14, 18] approach, and on *Subobject Transformation Systems* (STSs) [15]. Typed rewriting is a well-established variant of the classical proposals where rewriting takes place on so-called typed graphs, i.e., graphs labelled over a structure which is itself a graph [19, 20].

2.1 Graphs and graph morphisms

We introduce here the basic concepts concerning graphs and their morphisms. For the sake of simplicity, our introduction to GTSs will deal with unary hypergraphs only, since they are just what is needed for the encoding of Petri nets that we are going to present. Indeed, all the remarks in this section could be generalized to any kind of (hyper-)graphs or, albeit with some additional care, to any *adhesive* category [21]. Similarly, the encodings presented later would work in standard categories of (hyper-)graphs.

Given a partial function $f : A \rightarrowtail B$ we denote by $dom(f)$ its *domain*, i.e., the set $\{a \in A \mid f(a) \text{ is defined}\}$. Let $f, g : A \rightarrowtail B$ be two partial functions. We write $f \leq g$ when $dom(f) \subseteq dom(g)$ and $f(x) = g(x)$ for all $x \in dom(f)$.

Definition 1 (graph and graph morphism). *A* (unary) graph G *is a triple* $G = (V_G, E_G, c_G)$*, where* V_G *is a set of nodes,* E_G *is a set of edges and* $c_G : E_G \to V_G$ *is a function mapping each edge to the node it is connected to.*

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* $c_H \circ f_E \leq f_N \circ c_G$ *(see Fig. 3.(a))*

We denote by **PGraph** *the category of (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 if a partial graph morphism f is defined over an edge, then it must be defined on the node the edge is connected to: This ensures that the domain of f is a well-formed graph.

Definition 2 (subgraph lattice). *A graph* G *is a* subgraph *of* H*, written* $G \subseteq H$*, if* $N_G \subseteq N_H$*,* $E_G \subseteq E_H$*, and the inclusions form a graph morphism. The set of subgraphs of* H *ordered by inclusion form a distributive lattice, denoted* $\mathbf{Sub}(H)$*, where the meet* $\cap$ *and the join* $\cup$ *are defined as component-wise intersection and union, respectively.*

Given graphs H and $G \subseteq H$, we will write, a bit informally, $H \setminus G$ to denote the set of items (nodes and edges) of H which do not belong to G.

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 morphisms $f : |G_1| \rightarrowtail |G_2|$ consistent with the

typing, i.e., such that $t_{G_1} \geq t_{G_2} \circ f$ (see Fig. 3.(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**.

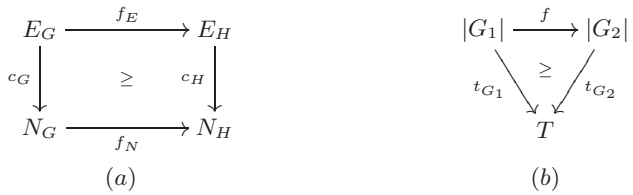


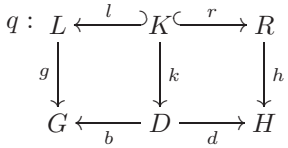
Fig. 3. Diagrams for partial graph and typed graph morphisms.

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. Given a subgraph G of T, i.e., an element of $\mathbf{Sub}(T)$, we often consider it as a graph typed over T by the inclusion. Since we work only with typed notions, we usually omit the qualification "typed".

2.2 Double-pushout rewriting

Chosen a type graph T, a *(T-typed)* DPO *rule* $q = (L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$ is a pair of *injective* (total, T-typed) graph morphisms $l : K \hookrightarrow L$ and $r : K \hookrightarrow R$, where $|L|$, $|K|$ and $|R|$ are finite graphs. The graphs L, K, and R are called the *left-hand side*, the *interface*, and the *right-hand side* of the rule, respectively.

Definition 3 (DPO direct derivation). *Given a graph G, a* DPO *rule q, and a* match *(i.e., a total graph morphism) $g : L \to G$, a* DPO direct derivation from G to H using q (based on g) *exists, written $G \Rightarrow_q^{\mathrm{DPO}} H$, if the diagram*



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

Given an injective morphism $l : K \hookrightarrow L$ and a match $g : L \to G$ as in the above diagram, their *pushout complement* (i.e., a graph D with morphisms k and b such that the left square is a pushout) exists if and only if the *gluing condition* is satisfied. This consists of two parts:

– the *identification condition*, requiring that if two distinct nodes or edges of L are mapped by g to the same image, then both are in the image of l;

– the *dangling condition*, stating that no edge in $G \setminus g(L)$ should be connected to a node in $g(L \setminus l(K))$ (because otherwise the application of the rule would leave such an edge "dangling").

2.3 Single-pushout rewriting

Chosen a type graph T, a *(T-typed)* SPO *rule* $q = (L \overset{r}{\rightarrowtail} R)$ is an injective partial typed graph morphism $r : L \rightarrowtail R$. The graphs L and R are called the *left-hand side* and the *right-hand side* of the rule, respectively.

Definition 4 (SPO direct derivation). *Given a graph G, an* SPO *rule q, and a* match *(i.e., a total graph morphism) $g : L \to G$, we say that there is an* SPO direct derivation from G to H using q (based on g), *written $G \Rightarrow_q^{\mathrm{SPO}} H$, if the following is a pushout square in T-***PGraph**.

$$\begin{array}{ccc} L & \overset{r}{\rightarrowtail} & R \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ G & \underset{d}{\rightarrowtail} & H \end{array}$$

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, namely $g(L \setminus dom(r))$, adding the items of the right-hand side which are not in the image of r, namely $R \setminus r(dom(r))$. The items in the image of $dom(r)$ 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* 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 connected to such node are deleted by the rewriting step, as a kind of side-effect. For instance, rule q in the top row of Fig. 4, which consumes node B, can be applied to the graph G in the same figure. As a result both node B and edge L are removed.

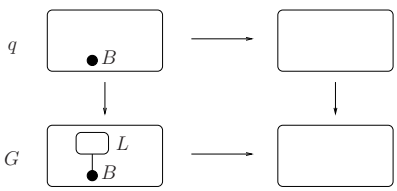


Fig. 4. Side-effects in SPO rewriting.

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 5 (valid match). *A match $g : L \to G$ is called* valid *when for any $x, y \in |L|$, if $g(x) = g(y)$ then $x, y \in dom(r)$.*

Conceptually, a match is not valid if it requires a single resource to be consumed twice, or to be consumed and preserved at the same time. In the paper we consider only *valid* derivations: This is needed to have a resource-conscious interpretation for derivations, i.e., where a resource is consumed at most once.

We close this section noting that for each DPO rule we can easily construct an SPO rule, which behaves like the original one when the dangling condition is satisfied. Clearly, the converse construction is possible as well.

Definition 6 (from DPO to SPO rules, and vice versa). *Let $q = (L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$ be a T-typed* DPO *rule. Then, the associated T-typed* SPO *rule, denoted by $\mathcal{S}(q)$, is given by the partial graph morphism $r \circ l^* : L \rightarrowtail R$, where $l^* : L \rightarrowtail K$ is the partial inverse of l, defined in the obvious way.*

Vice versa, for a T-typed SPO *rule $q = (L \overset{r}{\rightarrowtail} R)$, the associated* DPO *rule is defined as $\mathcal{D}(q) = (L \hookleftarrow dom(r) \overset{r}{\hookrightarrow} R)$.*

2.4 Subgraph Transformation Systems

In the typed approaches to graph transformation, the type graph plays a role analogous to the set of places in Petri nets. In particular, the constraint that a place can contain at most one token can be translated into the requirement that the typing morphism is injective. Both the DPO and the SPO approaches can be equipped with side conditions that guarantee that only injectively typed graphs are generated during rewriting, but this condition is built-in in the instance of the *Subobject Transformation System* approach [15] that we present here.

In the original formulation, the framework where rewriting is defined is the distributive lattice of subobjects of a fixed object of an adhesive category. Such generality is unnecessary here, and we instantiate the definitions to the case where the category of concern is **Graph**, which is indeed adhesive. As a consequence, in the following we read "STS" as *Subgraph* Transformation Systems.

Chosen a type graph T, a *(T-typed)* STS *rule* q is a triple $q = \langle L, K, R \rangle$, where $L, K, R \in \mathbf{Sub}(T)$, $K \subseteq L$ and $K \subseteq R$. The graphs L, K and R are called the *left-hand side*, the *interface* and the *right-hand side* of the rule, respectively.

Definition 7 (STS direct derivation). *Given a graph G in $\mathbf{Sub}(T)$ and an* STS *rule $q = \langle L, K, R \rangle$, there is an* STS direct derivation from G to H using q, *written $G \Rightarrow_q^{\text{STS}} H$, if $H \in \mathbf{Sub}(T)$ and there exists $D \in \mathbf{Sub}(T)$ such that*

$$(i)\ \ L \cup D = G; \qquad (iii)\ D \cup R = H;$$
$$(ii)\ L \cap D = K; \qquad (iv)\ \ D \cap R = K.$$

If such a graph D exists, we shall refer to it as the *context* of the direct derivation $G \Rightarrow_q^{\text{STS}} H$.

It is instructive to consider the relationship between an STS direct derivation and a DPO direct derivation as introduced above. First observe that $\mathbf{Sub}(T)$ can be seen as a category where the arrows are the inclusions, and a rule $\langle L, K, R\rangle$ can be seen as a span $q = (L \supseteq K \subseteq R)$, i.e., a pair of arrows in $\mathbf{Sub}(T)$. Next, we shall say that there is a *contact situation* for a rule $\langle L, K, R\rangle$ at a subgraph $G \supseteq L \in \mathbf{Sub}(T)$ if $G \cap R \not\subseteq L$. Intuitively, this means that some items of the subgraph G are created but not deleted by the rule: If we were allowed to apply the rule at this match via a DPO direct derivation, the resulting object would contain the common part twice and consequently the resulting morphism to T would not be injective; i.e., the result would not be a subgraph of T. The next result, presented in [15], shows that an STS direct derivation is also a DPO direct derivation if no contact occurs.

Proposition 1 (STS derivations are contact-free double pushouts). *Let G and H be graphs in $\mathbf{Sub}(T)$ and $q = \langle L, K, R\rangle$ be an STS rule. Then $G \Rightarrow_q^{\mathrm{STS}} H$ if and only if $L \subseteq G$, $G \cap R \subseteq L$, and $G \Rightarrow_q^{\mathrm{DPO}} H$, i.e., if there is a graph $D \in T$-$\mathbf{Graph}$ such that the diagram below forms two pushouts in T-$\mathbf{Graph}$.*

$$\begin{array}{ccccc} L & \xleftarrow{\supseteq} & K & \xrightarrow{\subseteq} & R \\ {\scriptstyle\subseteq}\downarrow & (1) & \downarrow & (2) & \downarrow \\ G & \xleftarrow{\quad} & D & \xrightarrow{\quad} & H \end{array}$$

In the last result we used the fact that an STS rule can be considered as a T-typed DPO rule, considering the inclusions as arrows in **Graph**. Conversely, a T-typed DPO rule $q = (L \overset{l}{\hookleftarrow} K \overset{r}{\hookrightarrow} R)$ induces an STS rule $\mathcal{I}(q)$ obtained by considering the images of $|L|$, $|K|$ and $|R|$ in the type graph, i.e., $\mathcal{I}(q) = \langle t_L(|L|), t_K(|K|), t_R(|R|)\rangle$.

2.5 Other kinds of STSs

We introduce here three variations of the definition of STS direct derivation, obtained by slightly changing the properties verified by the context graph D.

The first definition is reminiscent of the *sesqui-pushout* approach [22], and it leads to an SPO-like approach for STS, where rules can be applied regardless of the dangling condition, removing, as a side-effect, those edges which would remain dangling.

Definition 8 ($\mathrm{STS}^{\sqsubseteq}$ direct derivation). *Given a graph G in $\mathbf{Sub}(T)$ and an STS rule $q = \langle L, K, R\rangle$, there is an $\mathrm{STS}^{\sqsubseteq}$* direct derivation from G to H using q, *written $G \Rightarrow_q^{\mathrm{STS}^{\sqsubseteq}} H$, if $H \in \mathbf{Sub}(T)$ and*

(ii)′ D is the largest subgraph of G such that $L \cap D = K$;
(iii) $D \cup R = H$;
(iv) $D \cap R = K$.

Dropping the first condition of Definition 7 and imposing the "largest subgraph" requirement in (ii) implies that some items of $G \setminus L$ may not occur in D, as when deleting a node forces the deletion of incident edges in the SPO approach.

The next variants drop the requirement $D \cap R = K$. This allows for some overlap between the items preserved in the context D and those newly introduced by R: The injectivity of the typing forces these items to be coalesced, similarly to what happens in CPR nets. This is done for STSs both in DPO and SPO style.

Definition 9 (STS$_m$ and STS$_m^{\sqsubseteq}$ direct derivations). *Given a graph G in* $\mathbf{Sub}(T)$ *and an* STS *rule $q = \langle L, K, R\rangle$, there is an* STS$_m$ direct derivation from G to H using q, *written* $G \Rightarrow_q^{\text{STS}_m} H$, *if* $H \in \mathbf{Sub}(T)$ *and there exists* $D \in \mathbf{Sub}(T)$ *such that*

$$(i)\ \ L \cup D = G; \qquad (iii)\ D \cup R = H.$$
$$(ii)\ L \cap D = K;$$

Analogously, there is an STS$_m^{\sqsubseteq}$ direct derivation from G to H using q, *written* $G \Rightarrow_q^{\text{STS}_m^{\sqsubseteq}} H$, *if* $H \in \mathbf{Sub}(T)$ *and*

$(ii)'$ *D is the largest subgraph of G such that $L \cap D = K$;*
(iii) $D \cup R = H$.

Figure 5 shows the differences among the various kinds of STS direct derivations introduced in Definitions 7, 8 and 9. The type graph T contains two nodes, $\circ$ and $\bullet$, and one edge connected to $\circ$; all the elements of $\mathbf{Sub}(T)$ (the subgraphs of T) are depicted, with the obvious inclusions. The arrows show all the possible direct derivations using the STS rule $q = \langle\{\circ\}, \emptyset, \{\bullet\}\rangle$ and the approaches introduced in Definitions 7, 8 and 9.

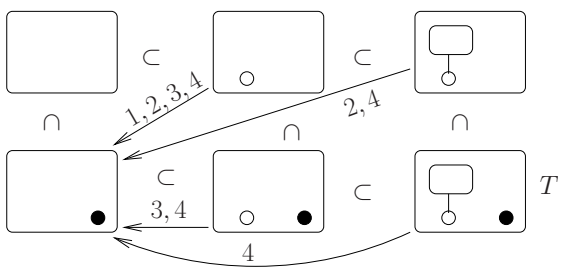


Fig. 5. Examples of the various kinds of STS direct derivations. Arrows represent direct derivations among elements of $\mathbf{Sub}(T)$ using rule $q = \langle\{\circ\}, \emptyset, \{\bullet\}\rangle$ and the following approaches: $1 = \text{STS}, 2 = \text{STS}^{\sqsubseteq}, 3 = \text{STS}_m, 4 = \text{STS}_m^{\sqsubseteq}$.

2.6 Graph grammars

In the previous sections we presented six different definitions of direct derivation, each of which determines a different algebraic approach to graph transformation. For each one of those approaches, a graph grammar contains a type graph, a start graph, a set of rule names, and a mapping from rule names to corresponding rules. Clearly, the precise definition of start graph and of rule depends on the chosen approach.

Definition 10 (graph grammar). *A* KND graph grammar*, where* $\text{KND} \in \{\text{DPO}, \text{SPO}, \text{STS}, \text{STS}^{\sqsubseteq}, \text{STS}_m, \text{STS}_m^{\sqsubseteq}\}$*, is a tuple* $\mathcal{G} = \langle T, G_s, P, \pi \rangle$*, where* $T \in \mathbf{Graph}$ *is the* type graph*,* P *is a set of* rule names*,* π *is a function which associates a* KND *rule*[1] *to each rule name in* P*, and* G_s *is the* start graph*, which has to be consistent with* KND*. That is,* G_s *is a* T*-typed graph if* $\text{KND} \in \{\text{DPO}, \text{SPO}\}$*, and* $G_s \in \mathbf{Sub}(T)$ *in all other cases.*

A *derivation* over a KND grammar $\mathcal{G}$ is a sequence of KND direct derivations using rules in P, starting from the start graph, namely $\rho = \{G_{i-1} \Rightarrow_{p_{i-1}}^{\text{KND}} G_i\}_{i \in \{1,\ldots,n\}}$, with $G_0 = G_s$.

3 Enriched Petri nets

In this section we introduce some basic extensions of Petri nets, namely, nets with read, inhibitor and reset arcs. A study of the expressiveness of these kinds of arcs, along with a comparison with other extensions proposed in the literature, like priorities, exclusive-or transitions and switches, is carried out in [23, 24].

To give the formal definition of these generalised nets we need some notation for sets and multisets. Given a set X we write $\mathbf{2}^X$ for the powerset of X and $X^{\oplus}$ for the free commutative monoid over X, with monoid operation $\oplus$, whose elements will be referred to as *multisets* over X. Given a multiset $M \in X^{\oplus}$, with $M = \bigoplus_{x \in X} M_x \cdot x$, for $x \in X$ we will write $M(x)$ to denote the coefficient M_x. Moreover, we denote by $[\![M]\!]$ the underlying subset of X, defined as $[\![M]\!] = \{x \in X \mid M(x) > 0\}$. With little abuse of notation, we will write $x \in M$ iff $x \in [\![M]\!]$.

Given $M, M' \in X^{\oplus}$ we write $M \leq M'$ when $M(x) \leq M'(x)$ for all $x \in X$. In this case the *multiset difference* $M' \ominus M$ is the multiset M'' such that $M \oplus M'' = M'$. For $Y \subseteq X$ and $M \in X^{\oplus}$, we denote by $M[Y]$ the restriction of M to Y, i.e., $M[Y](x) = M(x)$ if $x \in Y$, and $M[Y](x) = 0$ otherwise. Finally, the symbol $\emptyset$ denotes the empty multiset.

3.1 Place/Transition nets

We are now ready to define the enriched P/T nets considered in the paper. Besides ordinary flow arcs and read arcs, the nets are endowed with so-called "distinguished arcs" (represented by the ${}^{\circledcirc}(.)$ function below), which will be interpreted either as inhibitor or reset arcs in the token game.

[1] To be precise, for $\text{KND} \in \{\text{STS}^{\sqsubseteq}, \text{STS}_m, \text{STS}_m^{\sqsubseteq}\}$, a KND rule is an STS rule.

Definition 11 (enriched P/T nets). *An* enriched (marked) P/T Petri net *is a tuple* $N = \langle S, Tr, {}^\bullet(.), (.)^\bullet, \underline{(.)}, {}^\circledcirc(.), m\rangle$, *where*

- S *is a set of* places*;*
- Tr *is a set of* transitions*;*
- ${}^\bullet(.), (.)^\bullet : Tr \to S^\oplus$ *are functions mapping each transition to its pre-set and post-set, respectively;*
- $\underline{(.)} : Tr \to \mathbf{2}^S$ *is a function mapping each transition to its* context*;*
- ${}^\circledcirc(.) : Tr \to \mathbf{2}^S$ *is a function mapping each transition to its* distinguished set *of places, such that for all* $t \in Tr$, $({}^\bullet t \oplus \underline{t} \oplus t^\bullet)[\,{}^\circledcirc t] = \emptyset$ *(i.e., no token in* ${}^\circledcirc t$ *can be either read, consumed or produced by* t*);*
- $m \in S^\oplus$ *is a multiset called the* initial marking.

We assume, as usual, that $S \cap Tr = \emptyset$. We shall denote with ${}^\bullet(.), (.)^\bullet, \underline{(.)}$ and ${}^\circledcirc(.)$ also the functions from S to $\mathbf{2}^{Tr}$ defined as, for $s \in S$, ${}^\bullet s = \{t \in Tr \mid s \in t^\bullet\}$, $s^\bullet = \{t \in Tr \mid s \in {}^\bullet t\}$, $\underline{s} = \{t \in Tr \mid s \in \underline{t}\}$, and ${}^\circledcirc s = \{t \in Tr \mid s \in {}^\circledcirc t\}$.

A state of a P/T net is defined as a *marking*, that is, a set of tokens distributed over the places. Formally, a marking M is a multiset of places, i.e., $M \in S^\oplus$. The *token game* determines when a transition t is *enabled* at a given marking, and, if enabled, what marking is reached after firing the transition. For a transition t to be enabled at a marking M, it is necessary for M to contain the pre-set of t and an additional set of tokens which covers the context of t. Additional conditions for enabledness, as well as the result of firing, depend on the interpretation given to the distinguished arcs: As anticipated, we interpret them either as *inhibitor arcs* or as *reset arcs*, obtaining the classes of nets below.

Definition 12 (inhibitor and reset P/T nets). *An* inhibitor P/T net *is an enriched P/T net* $\langle S, Tr, {}^\bullet(.), (.)^\bullet, \underline{(.)}, {}^\circledcirc(.), m\rangle$ *where the distinguished arcs are interpreted as inhibitor arcs. Given a marking* $M \in S^\oplus$ *and a transition* $t \in Tr$, t *is* i-enabled *if* ${}^\bullet t \oplus \underline{t} \leq M$ *and* $M[\,{}^\circledcirc t] = \emptyset$ *(i.e.,* M *contains no token in any place of* ${}^\circledcirc t$*). The* inhibitor transition relation *between markings is defined as*

$$M\,[t\rangle_i\,M' \qquad \text{if} \qquad t \text{ is i-enabled at } M \text{ and } M' = (M \ominus {}^\bullet t) \oplus t^\bullet.$$

A reset P/T net *is an enriched P/T net where the distinguished arcs are interpreted as reset arcs. Given* $M \in S^\oplus$ *and* $t \in Tr$, t *is* r-enabled *if* ${}^\bullet t \oplus \underline{t} \leq M$. *The* reset transition relation *is defined as*

$$M\,[t\rangle_r\,M' \qquad \text{if} \qquad t \text{ is r-enabled at } M \text{ and } M' = ((M \ominus {}^\bullet t) \oplus t^\bullet) \ominus M[\,{}^\circledcirc t]$$

(i.e., the firing of t *deletes all the tokens from places in* ${}^\circledcirc t$*: Such places are certainly empty after the firing, because they cannot belong to the post-set of* t*).*

For a transition t, if the distinguished set ${}^\circledcirc t$ is empty the two alternative enabling conditions coincide, as well as the induced transition relations on markings. In the following, we call *contextual Petri nets* the class of nets such that all its transitions have the distinguished set empty.

Firing sequences and reachable markings are defined in the usual way.

Example 1. An example of an enriched P/T net N can be found in the left part of Fig. 6. Graphically, transitions are connected to context places by undirected arcs and to distinguished places by dotted undirected arcs.

Starting from the initial marking $s_0 \oplus s_1 \oplus s_2 \oplus s_4$, a possible firing sequence is $t_1; t_2$ leading to the marking $s_2 \oplus s_3 \oplus 2s_4 \oplus s$.

If we first fire t_2, the net reaches the marking $s_0 \oplus s_2 \oplus s_4 \oplus s$. Now, if N is seen as an inhibitor P/T net, the presence of a token in s inhibits t_1 which cannot fire. If, instead, N is seen as a reset P/T net, transition t_1 can fire and, as a consequence, place s is emptied, producing the marking $s_2 \oplus s_3 \oplus 2s_4$.

3.2 Elementary nets

Let us call *elementary* a net where the states are defined as *(sub)sets* of places, rather than *multisets* of places as for P/T nets. Thus elementary nets comprise several net models proposed in the literature, including C/E nets [3], Elementary Net Systems [4], Consume-Produce-Read nets [5] and others.

An *enriched elementary (marked) net* $\langle S, Tr, {}^\bullet(.), (.)^\bullet, \underline{(.)}, {}^\circledcirc(.), m\rangle$ is defined as an enriched P/T net in Definition 11, requiring ${}^\bullet(.), (.)^\bullet : Tr \to \mathbf{2}^S$ and $m \in \mathbf{2}^S$ (i.e., ${}^\bullet t$ and $t^\bullet$ for all $t \in Tr$, as well as the initial marking m, are sets rather than multisets). Furthermore, besides the disjointness condition on the distinguished places, that is formulated as $({}^\bullet t \cup \underline{t} \cup t^\bullet) \cap {}^\circledcirc t = \emptyset$, it is required that no token in $\underline{t}$ is consumed or produced, i.e., $({}^\bullet t \cup t^\bullet) \cap \underline{t} = \emptyset$ for all $t \in Tr$.

Both inhibitor and reset elementary nets are easily defined, interpreting the distinguished arcs as expected. However, since the states are subsets of places, the enabling condition and the transition relation must ensure that the marking reached by firing a transition is a set. This is obtained in a different way by the two models of nets that we introduce: ENSs require a stronger enabling condition w.r.t. P/T nets, while CPR nets, intuitively, change the transition relation by allowing to merge tokens of the marking with those produced by the transition.

Definition 13 (inhibitor and reset Elementary Net Systems). *An* inhibitor ENS *is an enriched elementary net* $\langle S, Tr, {}^\bullet(.), (.)^\bullet, \underline{(.)}, {}^\circledcirc(.), m\rangle$ *where the distinguished arcs are interpreted as inhibitor arcs. Given a marking* $M \subseteq S$ *and a transition* $t \in Tr$*,* t *is* ie-enabled *if* ${}^\bullet t \cup \underline{t} \subseteq M$*,* $M \cap {}^\circledcirc t = \emptyset$*, and* $(M \setminus {}^\bullet t) \cap t^\bullet = \emptyset$*. The* ie-transition relation *between markings is defined as*

$$M\,[t\rangle_{ie}\,M' \qquad \text{if} \qquad t \text{ is ie-enabled at } M \text{ and } M' = (M \setminus {}^\bullet t) \cup t^\bullet.$$

A reset ENS *is an enriched elementary net where the distinguished arcs are interpreted as reset arcs. Given* $M \subseteq S$ *and* $t \in Tr$*,* t *is* re-enabled *if* ${}^\bullet t \cup \underline{t} \subseteq M$ *and* $(M \setminus {}^\bullet t) \cap t^\bullet = \emptyset$*. The* re-transition relation *is defined as*

$$M\,[t\rangle_{re}\,M' \qquad \text{if} \qquad t \text{ is re-enabled at } M \text{ and } M' = ((M \setminus {}^\bullet t) \cup t^\bullet) \setminus {}^\circledcirc t.$$

The condition $(M \setminus {}^\bullet t) \cap t^\bullet = \emptyset$ ensures that there is "no contact", i.e., t can produce a token only if it is not in M, or if it is deleted by t itself. As a consequence the $\cup$ operator in the definition of M' is actually a disjoint union.

This is the main difference with respect to CPR nets, where the "no contact" condition is omitted, and the arguments of $\cup$ in the definition of the follower marking might not be disjoint.

Definition 14 (inhibitor and reset CPR nets). *An* inhibitor CPR net *is an enriched elementary net where for a marking $M \subseteq S$ and a transition $t \in Tr$, t is* ic-enabled *if ${}^\bullet t \cup \underline{t} \subseteq M$ and $M \cap {}^\circledcirc t = \emptyset$; the* ic-transition relation *is defined as*

$$M\,[t\rangle_{ic}\,M' \qquad \text{if} \qquad t \text{ is ic-enabled at } M \text{ and } M' = (M \setminus {}^\bullet t) \cup t^\bullet.$$

A reset CPR net *is an enriched elementary net where for $M \subseteq S$ and $t \in Tr$, t is* rc-enabled *if ${}^\bullet t \cup \underline{t} \subseteq M$; the* rc-transition relation *is defined as*

$$M\,[t\rangle_{rc}\,M' \qquad \text{if} \qquad t \text{ is rc-enabled at } M \text{ and } M' = ((M \setminus {}^\bullet t) \cup t^\bullet) \setminus {}^\circledcirc t.$$

Example 2. Observe that the net N in Fig. 6 can be seen as an ENS. In this case, starting from the initial marking $\{s_0, s_1, s_2, s_4\}$ the transition t_1 cannot fire due to a contact situation in s_4, hence the only possible firing sequence is t_2.

If we interpret N as a CPR net, then t_1 can fire and the reached marking is $\{s_1, s_2, s_3, s_4\}$, where, intuitively, the token generated in s_4 is "merged" with the pre-existing one. In this state, t_2 can fire producing the marking $\{s_2, s_3,\ s_4, s\}$. If we start by firing t_2, as in the P/T case, t_1 is blocked or can fire (emptying place s), depending on whether we interpret N as an inhibitor or a reset CPR net.

4 From enriched nets to graph transformation systems

This section shows how enriched Petri nets can be encoded as graph grammars. Interestingly, the encoding is essentially the same for all kinds of nets: The different token game flavours are obtained by changing the approach to rewriting.

4.1 Encoding Petri nets as graph grammars

It is part of the folklore (see e.g. the discussion in [2] and the references therein) that (ordinary) Petri nets can be seen as a special kind of graph grammars. The simplest idea is that the marking of a net is represented as a graph with no edges typed over the places: A token in place s is a node typed over s. Then transitions are seen as rules which consume and produce nodes, as prescribed by their pre- and post-set. In this way, Petri nets exactly correspond to graph grammars acting over graphs containing only nodes, where rules preserve no item.

To make the encoding parametric with respect to the chosen class of Petri nets, here we consider a slightly different encoding, where edges, rather than nodes, play the role of tokens. Roughly, the idea of the encoding is the following:

– a place is represented as a node;

- tokens in a place are represented as unary edges connected to the corresponding node;
- a transition becomes a rule, which deletes the tokens in its pre-set, produces the post-set and preserves the tokens in its context; for any place in the distinguished set of t, the corresponding node is deleted and created again.

Note the chosen encoding for the distinguished set of t: In the DPO approach this will prevent the application of the rule if there is at least one token (edge) in the place, thus causing an inhibitor effect. In the SPO approach, the application of the rule will delete as a side-effect any edge possibly attached to the node, thus giving raise to a reset effect.

As a first step, we show how the set of places underlying an enriched net (either P/T or elementary) gives raise to a type graph. In all cases there will be a node s in the type graph for each place s in the net, and the number of edges incident on the node typed over s will represent the number of tokens in that place. Also the way in which markings are encoded as graphs does not depend on the specific kind of nets we are considering.

Definition 15 (type graph, markings). *Let S be a set of places. Then, the associated type graph T_S is (S, S, c), where $c(s) = s$ for all $s \in S$.*

Given a subset of places $S' \subseteq S$ and a marking $M \in S'^{\oplus}$, we define the graph $\mathbf{G}_S(S', M)$ as $(S', E(M), c)$, typed in the obvious way over T_S, such that $E(M) = \{\langle s, i\rangle \mid s \in [\![M]\!] \wedge 0 < i \leq M(s)\}$ and $c(\langle s, i\rangle) = s$ for all $\langle s, i\rangle \in E(M)$. We write simply $\mathbf{G}_S(M)$ for $\mathbf{G}_S(S, M)$.

So, each place contributes a node *and* an edge in the type graph T_S, and a marking can be regarded as a multiset of edges of the type graph.

We next introduce the encoding of net transitions into grammar rules. As mentioned above, the encoding is essentially independent of the kind of nets we are considering: The different firing behaviour will be obtained by changing the considered rewriting approach. Indeed, we next define the encoding of a transition as a DPO rule, but changing the rewriting approach (to SPO or STS) will just require a syntactical change in the presentation of the rule.

Definition 16 (net transitions as DPO rules). *Let t be a transition of an enriched P/T net with place set S. Then t is encoded as a T_S-typed DPO transition*

$$\mathbf{G}_S(t) = \mathbf{G}_S(X \cup {}^{\circledcirc}t, \underline{t} \oplus {}^{\bullet}t) \leftarrow \mathbf{G}_S(X, \underline{t}) \rightarrow \mathbf{G}_S(X \cup {}^{\circledcirc}t, \underline{t} \oplus t^{\bullet})$$

where $X = [\![{}^{\bullet}t \oplus \underline{t} \oplus t^{\bullet}]\!]$ and the left and right morphisms are inclusions.

The DPO rule $\mathbf{G}_S(t)$ corresponding to a transition t deletes the edges in its pre-set, preserves the edges in its context and produces the edges in its post-set. The nodes attached to edges in the pre-set, context and post-set (i.e., the set X) are preserved. Finally, the nodes corresponding to the places $s \in {}^{\circledcirc}t$ in the distinguished set of t are deleted and produced again.

It is now immediate to provide the encoding for the different kinds of Petri nets into graph grammars of the appropriate approach.

	Inhibitor	Reset
P/T nets	DPO $\mathbf{G}_S(t)$	SPO $\mathcal{S}(\mathbf{G}_S(t))$
ENS	STS $\mathcal{I}(\mathbf{G}_S(t))$	$\text{STS}^{\sqsubseteq}$ $\mathcal{I}(\mathbf{G}_S(t))$
CPR nets	STS_m $\mathcal{I}(\mathbf{G}_S(t))$	$\text{STS}_m^{\sqsubseteq}$ $\mathcal{I}(\mathbf{G}_S(t))$

Table 2. Encoding Petri nets as graph grammars.

Definition 17. *An enriched Petri net $N = \langle S, Tr, F, C, D, m\rangle$ of one of the six types of nets presented in Definitions 12, 13 and 14 is encoded as a* KND *graph grammar $\mathcal{G}(N) = \langle T, G_s, P, \pi\rangle$ where*

- $T = T_S$
- $P = Tr$
- $G_s = \mathbf{G}_S(m)$

Moreover KND *and the* KND *rule $\pi(t)$ associated to $t \in P$ are defined, according to the type of the net, as shown in Table 2.*

Obviously, the encoding also works for contextual nets (see the first column of Table 1 in the Introduction).

It can be shown that the encoding preserves the firing relation and reachability, in the sense specified by next theorem.

Theorem 1. *Let N be an enriched Petri net of one of the types introduced in Section 3, let* KND *be the type of grammar corresponding to the type of N according to Table 2, and let M be a marking of N. If $M\,[t\rangle\,M'$ in N then $\mathbf{G}_S(M) \Rightarrow_t^{\text{KND}} \mathbf{G}_S(M')$ in the* KND *graph grammar $\mathcal{G}(N)$; vice versa, if $\mathbf{G}_S(M) \Rightarrow_t^{\text{KND}} G'$ in the* KND *graph grammar $\mathcal{G}(N)$ then $M\,[t\rangle\,M''$ in N with $\mathbf{G}_S(M'') = G'$.*

4.2 Examples

In order to provide some more intuition, we next briefly discuss the encoding for the various classes of Petri nets.

P/T Petri nets. As shown in Table 2, the behaviour of P/T Petri nets is faithfully captured by standard DPO or SPO graph grammars.

Inhibitor nets. When N is a P/T inhibitor net, $\mathcal{G}(N)$ is a DPO graph grammar, where the effects of the dangling condition are used to encode inhibitor arcs. As an example, the net in Fig. 6, seen as an inhibitor P/T net, is encoded by the grammar in the same figure, interpreted as a DPO grammar. Observe that since place $s \in {}^{\circledcirc}t_1$, i.e., s inhibits transition t_1, the rule associated with t_1 deletes and produces again the node corresponding to s. In this way the presence of tokens in place s, represented by edges connected to such node, will inhibit the rule because of the dangling condition.

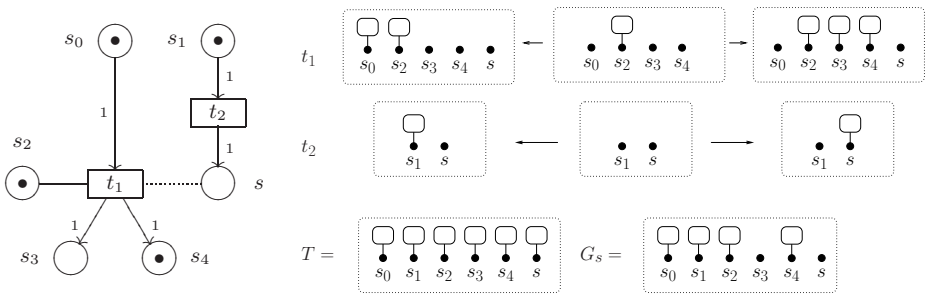


Fig. 6. An enriched Petri net N and the corresponding DPO grammar.

Reset nets. In the case of a P/T reset net N, the encoding $\mathcal{G}(N)$ is an SPO grammar and the side-effects related to node deletion turn out to capture precisely the behaviour of reset arcs. As an example the net in Fig. 6, seen as a reset P/T net, is encoded by the grammar in the same figure, seen as an SPO grammar (by transforming the rules using the function $\mathcal{S}(.)$). The fact that rule t_1 deletes and produces again the node s determines, as side effect, the deletion of all edges connected to such node, representing tokens in place s.

Contextual nets. For contextual P/T nets, i.e., P/T nets where ${}^{\circledcirc}t = \emptyset$ for all t, the rules of the corresponding grammar never delete nodes. Hence, the SPO and the DPO approaches are interchangeable. In particular, ordinary P/T net transitions t, such that $\underline{t} = {}^{\circledcirc}t = \emptyset$, are represented by rules with an interface containing only nodes (see the rule corresponding to t_2 in Fig. 6).

Elementary nets. As shown in Table 2, ENSs are encoded as STSs. As an example, let us consider again the net N in Fig. 6, which can be interpreted as an ENS interpreting, correspondingly, the grammar on the right as an STS.

Observe that, even though there is a match of the rule t_1 in the start graph G_s, i.e., the left-hand side of the rule is a subgraph of G_s, the rule cannot be applied, because there is a contact situation. More precisely, referring to Fig. 7, condition (iv) of Definition 7 (namely, $D \cap R = K$) is not satisfied, as the

intersection between the right-hand side of t_1 and the context graph D contains the edge connected to s_4 which is not in K.

If we interpret N as a CPR net and correspondingly the grammar as an STS_m, then the diagram in Fig. 7 is a legal derivation: in fact conditions $(i - iii)$ of Definition 8 are satisfied, while condition (iv) is not required anymore.

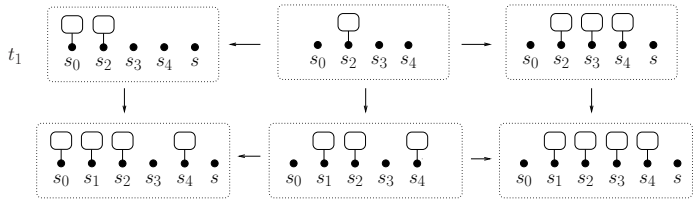


Fig. 7. A STS_m derivation which is not a legal STS one.

5 Concluding remarks and further works

In this paper we discussed the encoding of different Petri net models into Graph Transformation Systems. Our aim was of a methodological nature, and its accomplishments are summarized by the taxonomy proposed in Tables 1 and 2. Intuitively, the results can be synthesized by the slogan "encode the net once", that is, a Petri net is always encoded essentially in the same way, while different net models correspond to alternative approaches to graph transformation.

A relevant issue, which has been neglected in the present paper, concerns the study of concurrency in Petri nets and in their graph grammar counterparts. Admittedly, there is a shortcoming as far as inhibitor nets are considered (already noted in [16]): If two transitions are inhibited by the same place s, their encodings as DPO rules cannot be executed in parallel, since both rules delete and produce again the node corresponding to s. For instance, in the inhibitor net in Fig. 8,

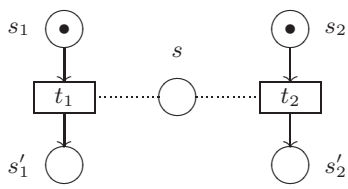


Fig. 8. An inhibitor net N_I: Transitions can fire concurrently. In $\mathcal{G}(N_I)$ they cannot.

the two transitions t_1 and t_2 can fire concurrently. However, in the corresponding DPO grammar the rules associated to t_1 and t_2 delete and generate again the same node s and thus they are forced to be executed sequentially. In general terms, we would like to see how to perform a technology transfer between the less-explored models of nets and GTSs, in order to address the issue of concurrent computations in these yet not fully developed formalisms.

References

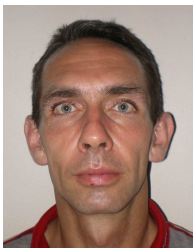
1. Kreowski, H.J.: A comparison between Petri nets and graph grammars. In Noltemeier, H., ed.: Proceedings of the Workshop on Graphtheoretic Concepts in Computer Science. Volume 100 of LNCS, Springer Verlag (1981) 306–317
2. Corradini, A.: Concurrent graph and term graph rewriting. In Montanari, U., Sassone, V., eds.: Proceedings of CONCUR'96. Volume 1119 of LNCS, Springer Verlag (1996) 438–464
3. Bernardinello, L., de Cindio, F.: A survey of basic net models and modular net classes. In Rozenberg, G., ed.: Advances in Petri Nets: The DEMON Project. Volume 609 of LNCS, Springer Verlag (1992) 304–351
4. Rozenberg, G., Engelfriet, J.: Elementary Net Systems. In Reisig, W., Rozenberg, G., eds.: Lectures on Petri Nets I: Basic Models. Volume 1491 of LNCS, Springer Verlag (1996) 12–121
5. Bonchi, F., Brogi, A., Corfini, S., Gadducci, F.: On the use of behavioural equivalences for web services' development. Fundamenta Informaticae **89** (2008) 479–510
6. Christensen, S., Hansen, N.D.: Coloured Petri nets extended with place capacities, test arcs and inhibitor arcs. In Ajmone-Marsan, M., ed.: Proceedings of ICAPTN'93. Volume 691 of LNCS, Springer Verlag (1993) 186–205
7. Montanari, U., Rossi, F.: Contextual nets. Acta Informatica **32** (1995) 545–596
8. Janicki, R., Koutny, M.: Semantics of inhibitor nets. Information and Computation **123** (1995) 1–16
9. Vogler, W.: Efficiency of asynchronous systems and read arcs in Petri nets. In: Proceedings of ICALP'97. Volume 1256 of LNCS, Springer Verlag (1997) 538–548
10. Agerwala, T., Flynn, M.: Comments on capabilities, limitations and "correctness" of Petri nets. Computer Architecture News **4** (1973) 81–86
11. Araki, T., Kasami, T.: Some decision problems related to the reachability problem for Petri nets. Theoretical Computer Science **3** (1977) 85–104
12. Habel, A., Plump, D.: Computational completeness of programming languages based on graph transformation. In Honsell, F., Miculan, M., eds.: Proceedings of FoSSaCS'01. Volume 2030 of LNCS, Springer Verlag (2001) 230–245
13. Löwe, M.: Algebraic approach to single-pushout graph transformation. Theoretical Computer Science **109** (1993) 181–224
14. Ehrig, H., Pfender, M., Schneider, H.: Graph-grammars: an algebraic approach. In Book, R., ed.: Switching and Automata Theory, IEEE Computer Society Press (1973) 167–180
15. Corradini, A., Hermann, F., Sobociński, P.: Subobject transformation systems. Applied Categorical Structures **16** (2008) 389–419
16. Baldan, P., Corradini, A., Montanari, U.: Relating SPO and DPO graph rewriting with Petri nets having read, inhibitor and reset arcs. In Ehrig, H., Padberg, J., Rozenberg, G., eds.: Proceedings of PNGT'04. Volume 127(2) of ENTCS, Elsevier (2005) 5–28

17. Ehrig, H., Heckel, R., Korff, M., Löwe, M., Ribeiro, L., Wagner, A., Corradini, A.: Algebraic Approaches to Graph Transformation II: Single Pushout Approach and comparison with Double Pushout Approach. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations. World Scientific (1997)
18. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M.: Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach. In Rozenberg, G., ed.: Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations. World Scientific (1997)
19. Corradini, A., Montanari, U., Rossi, F.: Graph processes. Fundamenta Informaticae **26** (1996) 241–265
20. Löwe, M., Korff, M., Wagner, A.: An Algebraic Framework for the Transformation of Attributed Graphs. In Sleep, M., Plasmeijer, M., van Eekelen, M., eds.: Term Graph Rewriting: Theory and Practice. Wiley, London (1993) 185–199
21. Lack, S., Sobociński, P.: Adhesive and quasiadhesive categories. Theoretical Informatics and Applications **39** (2005) 511–546
22. Corradini, A., Heindel, T., Hermann, F., König, B.: Sesqui-pushout rewriting. In Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G., eds.: Proceedings of ICGT 2006. Volume 4187 of LNCS, Springer Verlag (2006) 30–45
23. Peterson, J.: Petri Net Theory and the Modelling of Systems. Prentice-Hall (1981)
24. Lakos, C., Christensen, S.: A general systematic approach to arc extensions for Coloured Petri Nets. In Valette, R., ed.: Proceedings of ICAPTN'94. Volume 815 of LNCS, Springer Verlag (1994) 338–357

Prof. Dr. Paolo Baldan

Dipartimento di Matematica Pura e Applicata
Università di Padova
I-35121 Padova (Italy)
baldan@math.unipd.it
http://www.math.unipd.it/~baldan

Paolo Baldan met Hans-Jörg Kreowski for the first time when he was a fresh doctoral student joining the GetGraTS project (General Theory of Graph Transformation Systems) in 1997. Since then, Hans-Jörg's work on concurrency in graph transformation and on the relation between Petri nets and graph rewriting has been a reference for Paolo's research on this subject.



Prof. Dr. Andrea Corradini

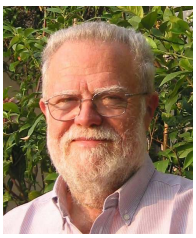
Dipartimento di Informatica
Università di Pisa
I-56127 Pisa (Italy)
andrea@di.unipi.it
http://www.di.unipi.it/~andrea

Andrea Corradini met Hans-Jörg Kreowski for the first time in Bremen, at the 4th International Workshop on Graph-Grammars and Their Application to Computer Science (1990). Along the years, they worked as partners in the European projects CompuGraph I and II, GetGraTS, and AppliGraph, but also in the IFIP Working Group 1.3. So they co-chaired the first International Conference on Graph Transformation in Barcelona (2002). Andrea has worked on several research topics on which Hans-Jörg had worked before him, notably on notions of equivalence of graph derivations, and encodings of Petri Nets as graph transformation systems.

Prof. Dr. Fabio Gadducci

Dipartimento di Informatica
Università di Pisa
I-56127 Pisa (Italy)
gadducci@di.unipi.it
http://www.di.unipi.it/~gadducci

Fabio Gadducci met Hans-Jörg Kreowski for the first time in Volterra, at the 1995 Joint CompuGraph/SemaGraph Workshop. The two then met at many further occasions, while Fabio held a grant of the GetGraTS project, dealing with issues related to concurrency for graph transformation. Fabio often benefitted from Hans-Jörg's earlier work, and also from his advice.



Prof. Dr. Ugo Montanari

Dipartimento di Informatica
Università di Pisa
I-56127 Pisa (Italy)
ugo@di.unipi.it
http://www.di.unipi.it/~ugo

Ugo Montanari knows Hans-Jörg Kreowski since his first encounter with the double-pushout graph transformation community, on the 2nd International Workshop on Graph-Grammars and Their Application to Computer Science (1982). Hans-Jörg's work on concurrency has been the basis of lots of contributions by his colleagues in Pisa and himself.