Towards a Petri net Model for Graph Transformation Systems

Lorenzo Capra

Abstract—**Graph transformation systems (GTS) have been successfully proposed as a general, theoretically sound model for concurrency. Petri nets (PN), on the other side, are a central and intuitive formalism for concurrent or distributed systems, well supported by a number of analysis techniques/tools. Some PN classes have been shown to be instances of GTS. In this paper, we change perspective presenting an operational semantics of GTS in terms of Symmetric Nets, a well-known class of Coloured Petri nets featuring a structured syntax that outlines model symmetries. Some practical exploitations of the proposed operational semantics are discussed. In particular, a recently developed structural calculus for SN is used to validate graph rewriting rules in a symbolic way.**

Index Terms—**Formal models, Graph Transformation Systems, Symmetric Nets.**

I. Introduction

Graph transformation systems (GTS) are a general, well established formal model for concurrency. Petri nets (PN) [1], on the other side, are a reference model for any formalism meant to describe concurrent or distributed systems, including GTS. Their success is mostly due to the fact that they can describe in a natural way the evolution of systems whose states have a distributed nature (this maps to the notion of *marking*), and the availability of a number of tools/techniques supporting the editing/analysis of PN models.

It is well known that GTS are a generalization of some PN classes, as shown by Kreowsky in its pioneering work [2] using the double-pushout approach. Basically, the idea is to represent a marked PN as a graph with three different types of nodes (for places, transitions, and tokens) and describe the firing of a PN transition thorough a rule (derivation). Since then, several encodings of PN classes in terms of GTS have been presented, among which Place/Transitions nets, Condition/Event nets, Elementary Net Systems, Consume-Produce-Read nets. Some net variants with extra features such as read/reset/inhibitor arcs have been also encoded. It is impossible to exhaustively list all these proposals, let us refer to [3] for the earliest and [4],[5] for more recent ones.

In this paper we consider the relationship between GTS and PN from a new perspective: we provide a formalization of Graph Transformation Systems (GTS) based on Symmetric Nets (SN) (earlier known as Well-formed Nets, or WN) [6], a type of Coloured Petri nets [7],[8] featuring a particular syntax that outlines model symmetries and is exploited both in state-space based and structural analysis. The idea is simple: each rule (derivation) of a GTS corresponds to a SN transition which is properly connected to a couple of SN places whose marking encodes a graph. In the paper we refer to simple directed graphs, even if the approach might be generalized to any category of (hyper)graphs.

Lorenzo Capra is with the Dipartimento di Informatica, Università degli Studi di Milano, Milano (MI), 20133 Italy, e-mail: capra@di.unimi.it

The advantages of this approach are numerous, and the aim of the paper is to illustrate some of them though some examples: we can exploit well established tools supporting the editing/analysis of SN, like the GreatSPN package [9]; an operational interleaving semantics for GTS is provided in a natural way building the state-transition system of a SN; a compact state-transition system -called symbolic reachability graph [10], in which states (markings) representing isomorphic graphs are folded, can be directly derived once an initial symbolic graph encoding is set; some recent advances in SN (symbolic) structural analysis [11], [12], implemented in the SNExpression tool (www.di.unito.it/ depierro/SNex) [13] may be exploited to check some conditions ensuring rule well-definiteness, validate rules, and verify their potential concurrency. These concepts are instantiated on a few, though significant, examples of graph rewriting rules, available (in GreatSPN format) at https://github.com/lgcapra/GTS-SN.

The balance of the paper is as follows: Section II introduces SN and related notions; Section III presents the encoding of a GTS as a SN, and its operational semantics; symbolic structural conditions for rule well-definiteness are also set up; Section IV shows an application of SN structural calculus to check rule concurrency in GTS; finally, Section V contains the conclusion and describes ongoing work

II. Symmetric Nets

In this section we present the SN formalism and a few preliminary concepts and notations used in the sequel. We let the reader refer to [1] and [6] for a complete treatment of Petri nets and SNs, respectively.

A. Multisets

A multiset (or *bag*) over a domain D is a map $b : D \to \mathbb{N}$, where $b(d)$ is the *multiplicity* of d in b. The *support* $\overline{b}$ is $\{d \in D | b(d) > 0\}$: we write $d \in b$ to mean $d \in \overline{b}$. A multiset b may be denoted by a weighted formal sum of $\overline{b}$ elements where coefficients represent multiplicities. The *null* multiset (over a given domain), i.e., the multiset with an empty support, is denoted (with some overloading) $\emptyset$. The set of all bags over D is denoted $Bag[D]$. Let $b_1, b_2 \in Bag[D]$. The sum $(b_1 + b_2)$ and the difference $(b_1 - b_2)$ $(\in Bag[D])$ are defined as: $(b_1 + b_2)(d) = b_1(d) + b_2(d)$; $(b_1 - b_2)(d) = max(0, b_1(d) - b_2(d))$. Relational operators are similarly defined, e.g., $b_1 < b_2$ if and only if $\forall d$, $b_1(d) < b_2(d)$. The *scalar* product $k \cdot b_1$, $k \in \mathbb{N}$, is $b_1' \in Bag[D]$, s.t. $b_1'(d) = k \cdot b_1(d)$. Let $b_1 \in Bag[A]$, $b_2 \in Bag[B]$: the *Cartesian* product $b_1 \times b_2 \in Bag[A \times B]$ is defined as $(b_1 \times b_2)(\langle a, b\rangle) = b_1(a) \cdot b_2(b)$.

a) Multiset functions: All the operators on multisets straightforwardly extend to functions mapping to multisets. Let $f_1, f_2 : D \to Bag[D']$; if *op* is a binary operator on bags, then f_1 *op* f_2 is defined as f_1 *op* f_2 $(a) = f_1(a)$ *op* $f_2(a)$.

(Advance online publication: 20 November 2019)

Analogously if *op* is a unary operator: e.g., $\overline{f_1}$ is a function $D \to 2^{D'}$ such that $\overline{f_1}(a) = \overline{f_1(a)}$. As for relational operators we have, e.g., $f_1 < f_2$ if and only if $\forall a, f_1(a) < f_2(a)$. Let $f_1 : D \to Bag[A]$, $f_2 : D \to Bag[B]$, and so forth: the product $f_1 \times f_2 \ldots : D \to Bag[A \times B \ldots]$ is defined: $f_1 \times f_2 \ldots (d) = f_1(d) \times f_2(d) \ldots$. In the following a function-tuple $\langle f_1, f_2, \ldots \rangle$ will denote the Cartesian product $f_1 \times f_2 \ldots$. Let $f : D \to Bag[D']$: its transpose $f^t : D' \to Bag[D]$ is defined as: $f^t(x)(y) = f(y)(x), \forall x \in D', y \in D$.

Bag functions are *linearly* extended: $f^* : Bag[D] \to Bag[D']$ is defined as $f^*(b) = \sum_{x \in b} b(x) \cdot f(x)$. The composition operator is extended accordingly: let $h : A \to Bag[B]$, $g : B \to Bag[C]$, then $g \circ h : A \to Bag[C]$ is defined as $g \circ h(a) = g^*(h(a))$. We will use the same symbol for a function and its linear extension.

When it will be clear from the context, the symbol $\emptyset$ will be used to denote the constant null multiset function.

B. Symmetric Nets

Symmetric Nets (SN) [6], are a high-level Petri Net formalism featuring a particular syntax for places, transitions, and arc annotations: such syntax has been devised to make the symmetries present in model's structure and behaviour explicit. This formalism is thus convenient from the point of view of model representation as well as from that of its analysis. Efficient methods have been proposed to perform SN state-space based analysis [10], or structural analysis [11],[12]. Many of these algorithms have been implemented in GreatSPN [9], whereas the most recent developments on structural analysis have been implemented in SNexpression (www.di.unito.it/ depierro/SNex).

SN are a particular flavour of *Colored* Petri nets (PN), originally introduced in [8]. Like in any Petri net, the SN underlying structure is a kind of (finite) directed bipartite graph, where the set of nodes is $P \cup T$, P and T being non-empty, disjoint sets, whose elements are called *places* and *transitions*, drawn as circles and bars, respectively. The former represent system state variables, whereas the latter events causing (local) state changes: what characterizes Petri nets in fact is a distributed notion of state, called *marking*. As in any high-level PN model, both places and transitions are associated with (colour) domains. Edges are annotated by (colour) functions mapping the domain of the incident transition to the domain of the incident place.

This section introduces the SN formalism exemplifying some key concepts by means of the models used in the rest.

1) Colour Domains: SN *places* are associated with a *color domain* (*cd*) defining the type of tokens a place may hold. A color domains is a Cartesian product of finite, non-empty, pair-wise disjoint *basic color classes*, denoted by capital letters (e.g., C). Basic color classes may be partitioned into *static subclasses* (denoted by capital letters with a subscript, e.g., C_1), or, in alternative, *circularly* ordered.

The SN models defined later build on a single basic color class: N=$\{nd_i\}$. The place color domains are N and E = $N \times N$ (or N^2).

Transitions have a color domain as well, since they specify parametric events. The color domain of a transition is implicitly determined by transition's parameters (*variables*) that annotate incident edges and transition's guard, denoted in this paper by lower-case letters with a subscript, e.g. c_i. By convention, the letter used for a variable implicitly defines its type, i.e., the color class denoted by the corresponding capital letter. Subscripts are used to distinguish variables of a given type associated with a transition. As an example, the colour domain of transition `R1` (Figure 1a) is $N \times N \times N$.

2) Transition guards: Transitions may have *guards*, consisting of *boolean predicates* defined on transition domains: $[c_1 = (\neq) c_2]$ is true when the same/a different color is assigned to c_1 and c_2; $[c_1 \in C_j]$ is true when the color assigned to c_1 belongs to subclass C_j. The default/implicit guard is the constant *true*.

A *transition instance* is a pair (t, b), where b (*binding*) is an assignment of colors to the transition's variables. For instance, a possible binding for `R1` is $n_1 = nd_2$, $n_2 = nd_1$, $n_3 = nd_3$. A transition instance is *valid* if it satisfies the transition's guard. From now on with transition color domain we will mean the set of valid transition instances.

3) Marking: A *marking* $\mathbf{m}$ provides a distributed notion of system state. Formally, a marking maps every place to a multiset on its domain: $\mathbf{m}(p) \in Bag[cd(p)]$ is the marking of place p. The multiset elements are called *tokens*.

4) Arc Functions: An arc form a place p to a transition t is said an *input* arc, whereas one in the opposite direction is called *output* arc. A place and a transition may be also connected by an *inhibitor arc*, drawn with an ending small circle instead of an arrow. Arcs are annotated by corresponding *arc functions*, denoted by $I[p,t]$, $O[p,t]$ and $H[p,t]$, respectively. An arc function is a map $F : cd(t) \to Bag[cd(p)]$, formally expressed as a linear combination:

$$F = \sum_i \lambda_i . T_i, \ \lambda_i \in \mathbb{N}, \tag{1}$$

where $T_i = \langle f_1, \ldots, f_k \rangle$ is a tuple of *class functions*.

A class-C function f is a map $cd(t) \to Bag[C]$, expressed in turn as a linear combination of elementary functions:

$$f = \sum_h \alpha_h . e_h, \ \alpha_h \in \mathbb{Z}, \tag{2}$$

where (referring to class C) $e_h \in \{c_j, c_j{+}{+}, C_q, All\}$:

- c_j (previously called variable) is actually a *projection*, i.e, given a tuple of colours in $cd(t)$ maps to the j^{th} occurrence of color C; if class C is ordered, then $++c_j$ denotes the successor $\mathrm{mod}_{|C|}$ of the element that c_j maps to;
- C_q and *All* are *diffusion* (or constant) functions mapping any color in $cd(t)$ to $\sum_{x \in C_q} 1 \cdot x$ and $\sum_{x \in C} 1 \cdot x$, respectively.

Scalars α_h in (2) must be such that no negative coefficients result from the evaluation of f_i for any legal binding of t. Both function-tuples and class-functions may be suffixed by a guard defined on $cd(t)$, acting as a filter: $f[g](a) = f(a)$ if $g(a)$, otherwise $f[g](a) = \emptyset$. If t has an associated guard $g(t)$ then we assume $g(t)$ spans over surrounding arc functions.

AS an example of arc function, consider the function on the inhibitor arc connecting transition `R2` to place `Edge` (Figure 1b). The transition's domain is $cd(\mathtt{R2}) = N$, because only variable n_1 occurs in incident edges. The evaluation of this function on a given $nd_i \in N$ results in the (multi)set composed of all pairs with the first element equal to nd_i and all pairs with the 2nd element equal to nd_i and the first one other than nd_i.

The only basic class used in the SN models of the paper is neither partitioned nor ordered. Arc functions, moreover, map to multisets with multiplicities ≤ 1. i.e., sets.

5) SN Execution: The interleaving semantics of a SN is defined by the *firing rule*. Assuming that missing arcs (of any type) between SN nodes are arcs annotated by the null function $\emptyset$, an instance (t, b) is *enabled* in marking $\mathbf{m}$ iff:

- $\forall p \in P$: $\mathrm{I}[p,t](b) \leq \mathbf{m}(p)$
- $\forall p \in P,\ x \in \mathrm{H}[p,t](b)$: $\mathrm{H}[p,t](b)(x) > \mathbf{m}(p)(x)$

An instance (t, b) enabled in $\mathbf{m}$ may *fire* by withdrawing from each input place p the bag $\mathrm{I}[p,t](b)$ and adding to each output place p the bag $\mathrm{O}[p,t](b)$. We get a new marking $\mathbf{m}'$: $\forall p: \mathbf{m}'(p) = \mathbf{m}(p) - \mathrm{I}[p,t](b) + \mathrm{O}[p,t](b)$.

We say that $\mathbf{m}'$ is *reachable* from $\mathbf{m}$ through (t, b), and this is denoted $\mathbf{m}[t, b > \mathbf{m}'$.

Once an *initial marking* $\mathbf{m}_0$ of a SN is set, it is possible to build the state-transition system (often called *reachability graph*, or RG) describing a SN model's behaviour. The RG is a (edge-labelled) directed multi-graph inductively defined as follows: $\mathbf{m}_0 \in RG$; if $\mathbf{m} \in RG$, and $\mathbf{m}[t, b > \mathbf{m}'$, also $\mathbf{m}' \in RG$ and there is an edge $\langle \mathbf{m}, \mathbf{m}' \rangle$ with label (t, b).

If a *symbolic* initial marking is set, a quotient graph called *symbolic reachability graph* is directly built, that retains all the information of the ordinary reachability graph.

III. Encoding GTS in SN

In this section we show how to encode a Graph Transformation Systems through Symmetric nets. In particular graph rewriting *rules* are formalized in terms of SN, and illustrated through a few examples. For the sake of simplicity we refer to simple directed graphs, even if this approach may be extended to any category of (hyper)graphs.

A *directed graph* (form now on simply graph) is composed of a (finite) set N of nodes and a set $E \subseteq N \times N$ of edges.

A. Graph encoding

Graph encoding through SN builds on a couple of places, `Node`, `Edge`, whose associated colour domain are the basic colour class $\mathrm{N} = \{nd_i\}$, and the product $\mathrm{E} = \mathrm{N} \times \mathrm{N}$, respectively. We assume that class N holds enough nodes to cover all possible evolutions of a graph.

A graph $G_1 = (N_1, E_1)$ is straightforwardly encoded by a SN marking, denoted $\mathbf{m}_{G_1}$: letting l be an injective labelling $N_1 \to \mathrm{N}$, $\mathbf{m}_{G_1}(\mathtt{Node}) = \sum_{n \in N_1} 1 \cdot l(n)$, $\mathbf{m}_{G_1}(\mathtt{Edge}) = \sum_{\langle n_1, n_2 \rangle \in E_1} 1 \cdot \langle l(n_1), l(n_2) \rangle$.

The other way round, a SN marking $\mathbf{m}$ is a *graph-encoding* if and only if both $\mathbf{m}(\mathtt{Node})$ and $\mathbf{m}(\mathtt{Edge})$ are multisets where multiplicities are ≤ 1 (i.e., sets) and any colour nd_i occurring in $\mathbf{m}(\mathtt{Edge})$ also occurs in $\mathbf{m}(\mathtt{Node})$.

B. Graph rewriting rules

A graph rewriting rule (or derivation) is formalized by a SN transition $\mathtt{R}_i$ properly connected to places `Node` and `Edge`. The colour domain of $\mathtt{R}_i$ depends on how many variables (projections) n_i occur on the incident arcs and transition's guard: in general, $cd(\mathtt{R}_i) = \mathrm{N}^k$, $k > 0$.

The idea is simple: the input arc functions $\mathrm{I}[\mathtt{Node}, \mathtt{R}_i]$, $\mathrm{I}[\mathtt{Edge}, \mathtt{R}_i]$ (assumed non both null), and the inhibitor arc function $\mathrm{H}[\mathtt{Edge}, \mathtt{R}_i]$, when evaluated on an enabled instance of $\mathtt{R}_i$ in a graph-encoding marking $\mathbf{m}$, match a subgraph of the encoded graph which is rewritten according to the SN firing rule: the matched subgraph is atomically removed from the encoded graph and replaced with the subgraph yielded by evaluating the output arc functions on the same instance. Inhibitor arc functions, even if not directly involved in the firing, play a crucial role both in subgraph matching and in setting structural conditions for rule correctness.

Some representative examples of rules are shown in Figure 1. Rule 1a allows the transitive closure of a graph be incrementally computed. Rule 1b represents the removal of isolated nodes of a graph. Rule 1c may be used to derive a Kripke structure from a graph: in fact, a self-loop is created for nodes without successors. Rule 1d transforms a self-loop involving node nd_i into a pair of edges from/to nd_j, where nd_j is a new node.

C. Well defined Rules

We have to establish some conditions ensuring that a rewriting rule is *well-defined*, that is, *any* instance of the rule (transition) rewrites a (simple) directed graph into another one. By exploiting the calculus for SN introduced in [12], [11], it is possible to characterize these rules as structural conditions on the arc functions annotating the corresponding transition, that may be checked in a fully symbolic and automated way, e.g., by using the SNexpression (www.di.unito.it/ depierro/SNex) toolset.

The calculus for SN has been developed to check basic structural properties (conflict, causal connection, mutual exclusion) on SN without any net unfolding. It builds on the ability to solve in a symbolic way expressions whose terms are the elements of a language $\mathcal{L}$ and involving a specific set of functional operators (in this context, the difference, the composition, and the support). The terms of $\mathcal{L}$ are a small extension of the SN arc functions, but the language restriction used here exactly matches SN arc functions. The calculus has been implemented as a rewriting system that, given any structural expression, reduces it to a normal form in $\mathcal{L}$. In particular, if $e \equiv \emptyset$ then $e \to \emptyset$.

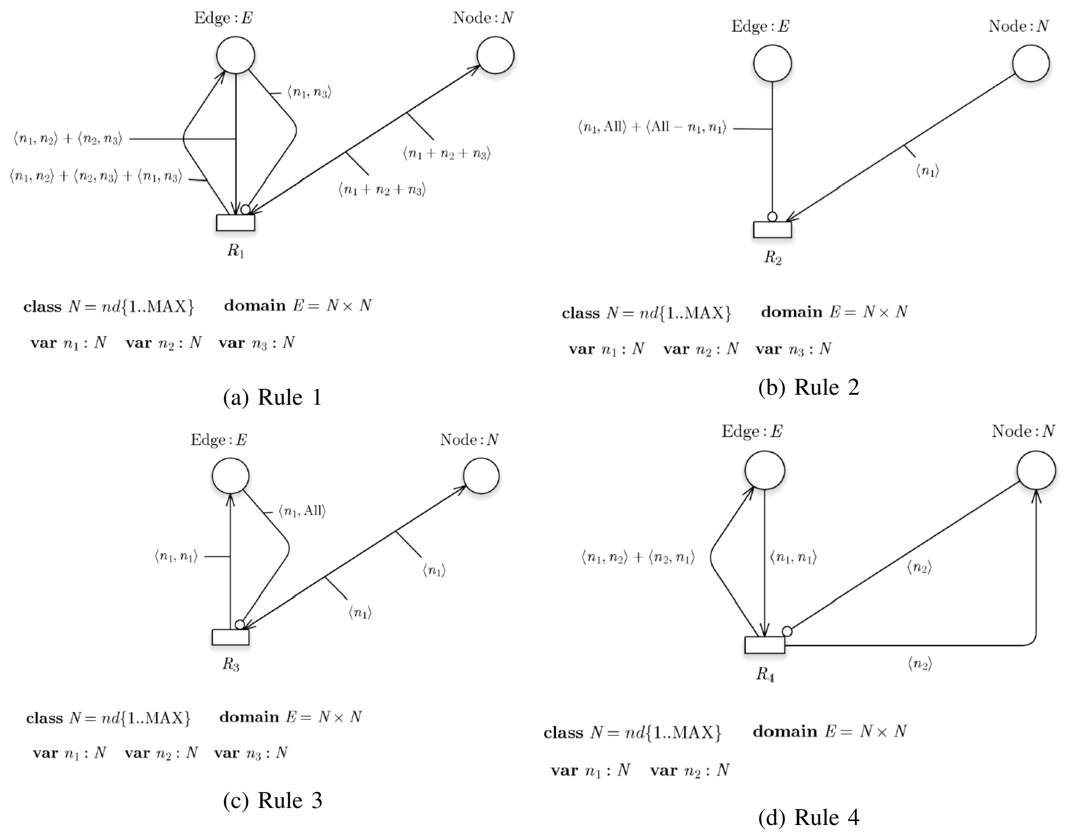
Hereinafter, the expressions $\mathrm{W}^+[p,t]$ and $\mathrm{W}^-[p,t]$ stand for $\mathrm{O}[p,t]-\mathrm{I}[p,t]$ and $\mathrm{I}[p,t]-\mathrm{O}[p,t]$, respectively: they map a transition instance (t, b) to the (multi)set of coloured tokens that (upon its firing) are added/withdrawn to/from place p.

Two type of terms are used: functions mapping to multisets, and their supports, mapping to sets. According to the type of operands, '−','+' will denote the multiset difference/sum or the set difference/sum. These equivalences are exploited (with an obvious overloading of symbol '$\emptyset$'):

$$F \leq G \Leftrightarrow F - G \equiv \emptyset;\ \overline{F} \subseteq \overline{G} \Leftrightarrow \overline{F} - \overline{G} \equiv \emptyset.$$

Let `R` be the transition encoding a rule. The conditions below ensure that `R` is well defined:

1) $\mathrm{H}[\mathtt{Edge},\mathtt{R}] \leq \langle All, All \rangle \quad \wedge \quad \mathrm{H}[\mathtt{Node},\mathtt{R}] \leq \langle All \rangle$
2) $\mathrm{W}^+[\mathtt{Edge},\mathtt{R}] \leq \langle All, All \rangle$
3) $\mathrm{W}^+[\mathtt{Node},\mathtt{R}] \leq \mathrm{H}[\mathtt{Node},\mathtt{R}]$
4) let $NA = (\overline{\langle n_1 + n_2 \rangle \circ \mathrm{O}[\mathtt{Edge},\mathtt{R}]} - \overline{\langle n_1 + n_2 \rangle \circ \mathrm{I}[\mathtt{Edge},\mathtt{R}]})$ $- \overline{\mathrm{I}[\mathtt{Node},\mathtt{R}]} : \quad NA \subseteq \overline{\mathrm{O}[\mathtt{Node},\mathtt{R}]}$
5) $\overline{\mathrm{W}^+[\mathtt{Edge},\mathtt{R}]} - (\overline{\langle NA, All \rangle} + \overline{\langle All, NA \rangle}) \subseteq \overline{\mathrm{H}[\mathtt{Edge},\mathtt{R}]}$
6) $\overline{((\langle All - n_1, n_1 \rangle + \langle n_1, All \rangle) \circ \mathrm{W}^-[\mathtt{Node},\mathtt{R}])} - \overline{\mathrm{W}^-[\mathtt{Edge},\mathtt{R}]}$ $\subseteq \overline{\mathrm{H}[\mathtt{Edge},\mathtt{R}]}$


Edge : E
Node : N
$\langle n_1, n_3\rangle$
$\langle n_1, n_2\rangle + \langle n_2, n_3\rangle$
$\langle n_1, n_2\rangle + \langle n_2, n_3\rangle + \langle n_1, n_3\rangle$
$\langle n_1 + n_2 + n_3\rangle$
$\langle n_1 + n_2 + n_3\rangle$
R_1
class $N = nd\{1..\mathrm{MAX}\}$ domain $E = N \times N$
var $n_1 : N$ var $n_2 : N$ var $n_3 : N$
Edge : E
Node : N
$\langle n_1, \mathrm{All}\rangle + \langle \mathrm{All} - n_1, n_1\rangle$
$\langle n_1\rangle$
R_2
class $N = nd\{1..\mathrm{MAX}\}$ domain $E = N \times N$
var $n_1 : N$ var $n_2 : N$ var $n_3 : N$
Edge : E
Node : N
$\langle n_1, \mathrm{All}\rangle$
$\langle n_1, n_1\rangle$
$\langle n_1\rangle$
$\langle n_1\rangle$
R_3
class $N = nd\{1..\mathrm{MAX}\}$ domain $E = N \times N$
var $n_1 : N$ var $n_2 : N$ var $n_3 : N$
Edge : E
Node : N
$\langle n_1, n_2\rangle + \langle n_2, n_1\rangle$
$\langle n_1, n_1\rangle$
$\langle n_2\rangle$
$\langle n_2\rangle$
R_4
class $N = nd\{1..\mathrm{MAX}\}$ domain $E = N \times N$
var $n_1 : N$ var $n_2 : N$

(a) Rule 1

(b) Rule 2

(c) Rule 3

(d) Rule 4

Fig. 1: Examples of graph rewriting rules

Conditions 1,2) are related to simplicity: 1) means that inhibitor functions map to multisets with multiplicities ≤ 1, i.e., they only check for the *absence* of nodes/edges in a graph-encoding; 2) means that new edges are inserted with multiplicity 1; 3) avoids node duplication. Conditions 4-6) avoid (among others) the creation of dangling edges, and are a bit more complex, involving the composition operator: 4) means that nodes incident to newly added edges, but that do not exist yet (this set is denoted NA), must be contextually inserted: it builds on the assumption that if at any moment an edge does exist, also its incident nodes do; 5) is related again to simplicity: means that whenever a new edge is added, we must check its absence unless one of the incident nodes belongs to precomputed set NA; finally, 6) deals with node removal: the function on the inhibitor arc must ensure that for every withdrawn node nd_i there are no edges incident to nd_i, but for those that are contextually removed.

A few remarks have to be done. In conditions 4,6), the projections n_1, n_2 have domain $\mathrm{N} \times \mathrm{N}$. The use of support operator is due to the fact that the composition may result in multisets with multiplicities > 1. The *parametric* set A (holding those nodes recognized as existing, due to the rule's functions) is computed by separately considering the output and the input arc functions to/from place Edge, instead of considering $\mathrm{W}^{+}\overline{[\texttt{Edge}, \texttt{R}]}$: this way we get a more general condition, since $\overline{F} - \overline{G} \subseteq \overline{F - G}$.

We can thus state the following:

Property 1. *If a rule/transition* R *meets conditions 1-6), then the firing of any instance (*R*,b) in a graph-encoding marking generates a graph-encoding marking.*

D. Bringing rules together

A Graph Transformation System (or GTS) may be very simply defined by bringing together a set of well-defined rules (transitions) sharing places `Node` and `Edge`, and setting an initial graph-encoding marking. The induced state-transition system corresponds to the SN reachability graph.

As an example, consider the SN in Figure 2. It comes from the combination of Rules 1,3) described above. Given a graph G_0 encoded by the initial marling $\mathbf{m}_{G_0}$, the derived RG describes the sequence of transformations that G_0 undergoes by applying either Rule 1 or Rule 3. The resulting RG has an *absorbing* state, i.e. a dead home-state, which corresponds to the transitive closure of $\mathbf{m}_{G_0}$ where nodes without proper predecessors are sources/targets of self-loops.

Let $\mathbf{m}_{G_0}(\texttt{Edge}) = \langle nd_1, nd_2\rangle + \langle nd_1, nd_3\rangle + \langle nd_4, nd_1\rangle$, and $\mathbf{m}_{G_0}(\texttt{Node}) = \langle nd_1 + nd_2 + nd_3 + nd_4\rangle$: the corresponding RG holds 16 nodes, one of which absorbing; this final node encodes the graph:

$$\langle nd_1, nd_2\rangle + \langle nd_1, nd_3\rangle + \langle nd_4, nd_1\rangle +$$
$$\langle nd_2, nd_2\rangle + \langle nd_3, nd_3\rangle + \langle nd_4, nd_2\rangle + \langle nd_4, nd_3\rangle.$$

Symbolic State-transition System During the construction of the SN reachability graph some markings encoding *isomorphic* graphs may be reached. Consider the example above: from the initial marking, we can reach the two markings below by firing $\texttt{R}_1$ with the bindings $n_1 = nd_4$, $n_2 = nd_1$, $n_3 = nd_2$ and $n_1 = nd_4$, $n_2 = nd_1$, $n_3 = nd_3$:

$$i)\ \ \langle nd_1, nd_2\rangle + \langle nd_1, nd_3\rangle + \langle nd_4, nd_1\rangle + \langle nd_4, nd_2\rangle$$
$$ii)\ \ \langle nd_1, nd_2\rangle + \langle nd_1, nd_3\rangle + \langle nd_4, nd_1\rangle + \langle nd_4, nd_3\rangle$$

Observe that $i)$ and $ii)$ are isomorphic since can be obtained from one another by swapping nd_2 with nd_3.

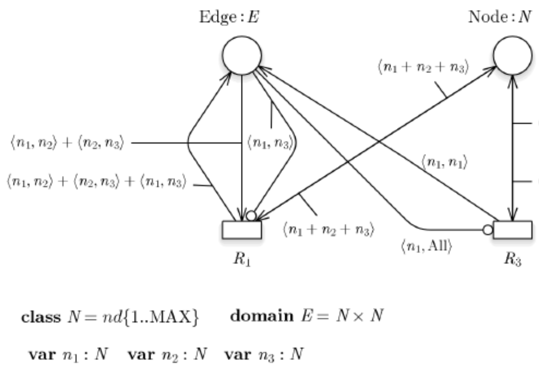

Edge : E
Node : N
$\langle n_1 + n_2 + n_3 \rangle$
$\langle n_1 \rangle$
$\langle n_1, n_2 \rangle + \langle n_2, n_3 \rangle$
$\langle n_1, n_3 \rangle$
$\langle n_1, n_1 \rangle$
$\langle n_1, n_2 \rangle + \langle n_2, n_3 \rangle + \langle n_1, n_3 \rangle$
$\langle n_1 \rangle$
$\langle n_1 + n_2 + n_3 \rangle$
$\langle n_1, \mathrm{All} \rangle$
R_1
R_3
class $N = nd\{1..\mathrm{MAX}\}$ **domain** $E = N \times N$
var $n_1 : N$ **var** $n_2 : N$ **var** $n_3 : N$

Fig. 2: a GTS composed of Rules 1,3

Recognizing isomorphic graph-encodings is for free in SN, if the initial marking is *symbolic*. A *symbolic marking* $\widehat{\mathbf{m}}$ [10] is an equivalence class of ordinary markings: $\{\mathbf{m}_1, \mathbf{m}_2\} \subseteq \widehat{\mathbf{m}}$ if and only if they correspond, up to a *permutation* on colour classes (preserving the possible partitions in subclasses)

A symbolic marking (or SM) is syntactically expressed using *dynamic subclasses* instead of ordinary colours. Dynamic subclasses define *parametric partitions* of basic colour classes: each dynamic subclass is associated with a colour class (or a static subclass, if the class is split) and has a cardinality. As an example, the initial symbolic marking encoding (among others) graph G_0 above is:

$$\widehat{\mathbf{m}}_0(\texttt{Edge}) = \langle znd_1, znd_{23} \rangle + \langle znd_4, znd_1 \rangle,$$
$$\widehat{\mathbf{m}}_0(\texttt{Node}) = \langle znd_1 + znd_{23} + znd_4 \rangle$$

where all symbols (dynamic subclasses) refer to class N, and $|znd_1| = |znd_4| = 1$, $|znd_{23}| = 2$. This symbolic marking represents six ordinary markings, including $\mathbf{m}_{G_0}$. A symbolic reachability graph (or SRG) is directly built from an initial symbolic marking, by means of a symbolic firing rule (and a canonical representative for SM). Skipping the technical details, a symbolic instance of $\mathtt{R}_1$ folding the two bindings above is enabled in $\widehat{\mathbf{m}}_0$; this symbolic instance may fire, leading to a new symbolic marking representing (among others) the ordinary markings $i)$ and $ii)$.

The SRG built (with the GreatSPN package) from $\widehat{\mathbf{m}}_0$ is a quotient-graph of the RG, retaining liveness and safety properties: in the simple example we are considering, the SRG holds 9 nodes plus an absorbing one, each encoding a class of isomorphic graphs. When huge graphs are encoded with SN, the reduction achieved with the SRG in terms of generated states/arcs may be dramatic (e.g., a symbolic instance of transition $\mathtt{R}_1$ may fold up to $|\mathrm{N}|^3$ ordinary instances), even if bringing a SM to a canonical form is comparable to checking graph isomorphism.

IV. Exploiting SN Structural Analysis

In Section III-C we have established some conditions on arc functions making a SN transition specify a well-defined graph rewriting rule. These conditions involve functional operators that can be solved in a fully automated/symbolic way through the SNexpression tool, implementing the computation of a base set of structural properties directly on SN models, without any unfolding. Each structural property may be expressed in terms of language $\mathcal{L}$, which is a small extension of arc functions.

Let us discuss now about the exploitation of these properties for validating rules, e.g., to figure out which rules of a GTS might concurrently apply. Concurrent graph rewriting issues have been widely tackled in literature: we do not want to go into the details of a theoretical discussion, rather we aim at showing the potential of SN structural analysis.

Symbolic structural relations are computed by properly combining arc functions through some operators: transpose, sum, difference, support, and composition. A relation is a map $\mathcal{R}(t, t') : cd(t') \rightarrow 2^{cd(t)}$ that when applied to an instance c' of t' gives the set of instances of t that are in such a relation with (t', c'). Symbolic relations build on a couple of auxiliary ones, involving a pair place/transition, both with arity $cd(p) \rightarrow 2^{cd(t)}$: $\mathrm{Rb}[t,p] = \overline{\mathrm{W}^-[p,t]}^t$ (*Removed by*), given a color c of p provides the set of instances of t that withdraw c from p; $\mathrm{Ab}[t,p] = \overline{\mathrm{W}^+[p,t]}^t$ (*Added by*), given a color c of p provides the set of instances of t that add c to p. Table I reports the definitions of base structural relations.

(Asymmetric) Structural Conflict: Two transition instances (t, c) and (t', c') are in conflict in a given marking $\mathbf{m}$ if the firing of the former disables the latter. The structural conflict (SC) relation defines the necessary conditions that *may* lead to an actual conflict in some marking. The symbolic relation $SC(t, t')$ maps a an instance c' of t' to the set of colour instances of t that may disable (t', c'): this happens either because (t, c) withdraws a token from an input place which is shared by the two transitions, or because it adds a token into an output place which is connected to t' through an inhibitor arc. These two cases are reflected in the SC formula, which is is obtained by summing up over all shared input places and shared output-inhibitor places. Observe that different instances of the *same* transition may be in conflict (auto conflict): the same expression can be used, but one must subtract from the set of conflicting instances the instance itself to which SC applies (using the identity function).

Structural Causal Connection: Two transition instances (t, c) and (t', c') are in causal connection if the firing of the former in a given marking $\mathbf{m}$ causes the enabling of the latter. The structural causal connection (SCC) relation defines the necessary conditions that may lead to an actual causal connection in some marking. The symbolic relation $SCC(t, t')$, when applied to an instance c' of t', provides the set of instances (t, c) that may cause the enabling of (t', c'). This happens if some output places of t are input places for t' and some input places of t are inhibitor places for t'.

Structural Mutual Exclusion: Two transition instances (t, c) and (t', c') are in (structural) mutual exclusion (SME) if the enabling of (t', c') in any $\mathbf{m}$ implies that (t, c) is not enabled, and viceversa. This situation arises when a place p does exist which is input for t and inhibitor for t', and the number of tokens (of any color) required in p for the enabling of t is greater than or equal to the upper bound on the number of tokens (of the same color) in p imposed by the inhibitor arc connecting p and t'. The (symmetric) symbolic relation $SME(t, t')$ maps an instance (t', c') to the set of instances of t that are surely disabled in any marking where (t', c') is enabled. If all functions on input and inhibitor arcs were mappings onto sets (i.e., on multisets with multiplicities ≤ 1), as in the SN models presented in this paper, then the SME relation corresponds to the expression in Table I ([12]), that applies also when t and t' coincide.

TABLE I: Symbolic Structural relations in SN

$SC(t,t')$	=	$\bigcup_p \mathrm{Rb}[t,p] \circ \overline{\mathrm{I}[t',p]} \ \cup \ \mathrm{Ab}[t,p] \circ \overline{\mathrm{H}[t',p]}$
$SCC(t,t')$	=	$\bigcup_p \mathrm{Ab}[t,p] \circ \overline{\mathrm{I}[t',p]} \ \cup \ \mathrm{Rb}[t,p] \circ \overline{\mathrm{H}[t',p]}$
$SME(t,t')$	=	$\bigcup_p \overline{\mathrm{I}[t,p]}^t \circ \overline{\mathrm{H}[t',p]} \ \cup \ \overline{\mathrm{H}[t,p]}^t \circ \overline{\mathrm{I}[t',p]}$

a) Application example: Structural relations can be used to validate the rules of a GTS formalized in terms of SN. In particular, it is possible to check which rules may concurrently apply, in the event a *true concurrent* semantics were used. Using the structural calculus for SN we can -in a way, parametrically (i.e., symbolically) partition the set of instances of a given transition (rule) on the basis of a given relation with the instances of the other (or the same) rule(s).

To illustrate these concepts, let us consider the GTS in Figure 2. The two rules are potentially in conflict due to place Edge, which is simultaneously an output place for one rule and an inhibitor place for the other. Instead, there are no potential conflicts due to the sharing of input places, since we can easily check that the expressions $\mathrm{Rb}[t,p]$ are null (by the way, a composition involving a null function results in $\emptyset$). As for the *added by* expressions, we got the following non-null entries (function supports are implicitly used):

$$\mathrm{Ab}[\mathtt{R}_1, \mathtt{Edge}] = \langle n_1, All, n_2 \rangle \quad \mathrm{Ab}[\mathtt{R}_3, \mathtt{Edge}] = \langle n_1 \rangle [n_1 = n_2]$$

The first expression says that a color (token) $\langle c_1, c_2 \rangle$ may be pushed into place Edge by *any* instance of $\mathtt{R}_1$ (a triplet of colours) whose 1st and 3rd elements are equal to c_1 and c_2, respectively. The other expression says that a color $\langle c_1, c_2 \rangle$, with $c_1 = c_2$, may be pushed into place Edge by the instance $\langle c_1 \rangle$ of $\mathtt{R}_3$. Then, according with Table I we obtain:

$$SC(\mathtt{R}_1, \mathtt{R}_3) = \langle n_1, All, n_2 \rangle \circ \langle n_1, All \rangle = \langle n_1, All, All \rangle$$
$$SC(\mathtt{R}_3, \mathtt{R}_1) = \langle n_1 \rangle [n_1 = n_2] \circ \langle n_1, n_3 \rangle = \langle n_1 \rangle [n_1 = n_3]$$

Again, the interpretation of these symbolic expressions is quite intuitive: $SC(\mathtt{R}_1, \mathtt{R}_3)$ says that an instance $\langle c_1 \rangle$ of Rule 3 might be in conflict with (i.e., disabled by) any instance of Rule 1 having color c_1 as first element; $SC(\mathtt{R}_3, \mathtt{R}_1)$ instead says that an instance $\langle c_1, c_2, c_3 \rangle$ of Rule 1, such that $c_1 = c_3$, might be in conflict with the instance $\langle c_1 \rangle$ of Rule 3.

The SC relation, however, outlines potential conflicts. This outcome may be refined by computing SME: in fact, we observe that place Edge is both input and inhibitor for $\mathtt{R}_1$, and inhibitor for $\mathtt{R}_3$. Then, according with Table I we obtain:

$$SME(\mathtt{R}_1, \mathtt{R}_3) = \langle All, n_1, All \rangle + \langle n_1, All, All \rangle$$
$$SME(\mathtt{R}_3, \mathtt{R}_1) = \langle n_1 \rangle + \langle n_2 \rangle$$

Notice that, according with the transpose rules and the symmetry of the relation: $SME(\mathtt{R}_3, \mathtt{R}_1)^t = SME(\mathtt{R}_1, \mathtt{R}_3)$. What is interesting, however, is that $SC(\mathtt{R}_1, \mathtt{R}_3) \subset SME(\mathtt{R}_1, \mathtt{R}_3)$ and $SC(\mathtt{R}_3, \mathtt{R}_1) \subset SME(\mathtt{R}_3, \mathtt{R}_1)$, i.e., potentially conflicting instances of Rule 1 and Rule 2 are mutually exclusive, therefore, Rule 1 and Rule 2 are potentially concurrent.

V. Conclusions and ongoing work

We have presented a formalization of Graph Transformation Systems (GTS) based on Symmetric Nets (SN), a type of Coloured Petri nets featuring a particular syntax that outlines model symmetries. Each GTS rule corresponds to a transition of a SN properly connected to a couple of places encoding a graph. The advantages of this approach are numerous: we can exploit well established tools supporting the editing/analysis of SN, like the GreatSPN package; an interleaving semantics for GTS is provided in a natural way by the state-transition system of a SN; a compact state-transition system in which markings representing isomorphic graphs are folded, can be directly derived from an initial symbolic graph encoding; some recent advances in SN structural analysis, implemented in the SNExpression tool, may be exploited to check some conditions ensuring rule well-definiteness, to validate rules, and to check their potential concurrency; in particular, a fully automated calculus of symbolic structural relations in SN models may be profitably used. All these concepts have been instantiated on a few, though significant, examples of graph rewriting rules, and a simple GTS.

Ongoing work is in two main directions. The presented approach is general, we are therefore extending the class of encodable graphs to multigraphs (this extension is for free, it only requires that some well-definiteness conditions on rules are relaxed), bipartite graphs, hypergraphs, and so forth. Some SN features not used in the paper might be needed: for example (think of bi-or three-partite graphs), partitioning the colour class of nodes in two or more subclasses

A more theoretical research line involves a comparison of the SN based approach with classical approaches to GTS, in particular the algebraic ones based on single/double pushout. We are firmly convinced that, under some quite general conditions, it is possible to characterize a SN rule as a pushout (in particular, a dpo) derivation.

References

[1] W. Reisig, *Petri Nets: An Introduction.* New York, NY, USA: Springer-Verlag New York, Inc., 1985.

[2] H.-J. Kreowski, "A comparison between petri-nets and graph grammars." vol. 100, 06 1980, pp. 306–317.

[3] A. Corradini, "Concurrent graph and term graph rewriting," 01 2006, pp. 438–464.

[4] P. Baldan, A. Corradini, F. Gadducci, and U. Montanari, "From petri nets to graph transformation systems," *ECEASST*, vol. 26, 01 2010.

[5] H. Ehrig and J. Padberg, "Graph grammars and petri net transformations," 01 2003, pp. 496–536.

[6] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "Stochastic well-formed colored nets and symmetric modeling applications," *IEEE Transactions on Computers*, vol. 42, no. 11, pp. 1343–1360, Nov 1993.

[7] K. Jensen and G. Rozenberg, Eds., *High-level Petri Nets: Theory and Application.* London, UK: Springer-Verlag, 1991.

[8] K. Jensen, *Coloured Petri Nets. Basic Concepts, Analysis Methods and Practical Use.* Volume 1, Basic Concepts. Monographs in Theoretical Computer Science, Springer-Verlag, 2nd corrected printing 1997. ISBN: 3-540-60943-1., 1997.

[9] S. Baarir, M. Beccuti, D. Cerotti, M. D. Pierro, S. Donatelli, and G. Franceschinis, "The GreatSPN tool: Recent enhancements," *SIGMETRICS Perform. Eval. Rev.*, vol. 36, no. 4, pp. 4–9, Mar. 2009.

[10] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad, "A symbolic reachability graph for coloured petri nets," *Theoretical Computer Science*, vol. 176, no. 1, pp. 39 – 65, 1997.

[11] L. Capra, M. D. Pierro, and G. Franceschinis, "A High Level Language for Structural Relations in Well-Formed Nets," in *Proc. of the 26th Int. Conf. ATPN 2005.* Springer, 2005, vol. LNCS 3536, pp. 168–187. [Online]. Available: $http://dx.doi.org/$10.1007/11494744$_1$1

[12] L. Capra, M. De Pierro, and G. Franceschinis, *Computing structural properties of symmetric nets.* Springer International Publishing, 2015, vol. 9259, pp. 125–140.

[13] L. Capra, M. D. Pierro, and G. Franceschinis, "A Tool for Symbolic Manipulation of Arc Functions in Symmetric Net Models," in *Proceedings of the 7th International Conference on Performance Evaluation Methodologies and Tools*, ser. ValueTools '13. Torino, Italy: ICST, 2013, pp. 320–323. [Online]. Available: http://dx.doi.org/10.4108/icst.valuetools.2013.254407