EPiC Series in Computing

Volume 46, 2017, Pages 300–318

LPAR-21. 21st International Conference on Logic for Programming, Artificial Intelligence and Reasoning

Parallel Graph Rewriting with Overlapping Rules

Rachid Echahed[1] and Aude Maignan[2]

[1] LIG - CNRS and Université de Grenoble Alpes, France
rachid.echahed@imag.fr
[2] LJK - Université de Grenoble Alpes and CNRS, France
aude.maignan@imag.fr

Abstract

We tackle the problem of simultaneous transformations of networks represented as graphs. Roughly speaking, one may distinguish two kinds of simultaneous or parallel rewrite relations over complex structures such as graphs: (i) those which transform disjoint subgraphs in parallel and hence can be simulated by successive mere sequential and local transformations and (ii) those which transform overlapping subgraphs simultaneously. In the latter situations, parallel transformations cannot be simulated in general by means of successive local rewrite steps. We investigate this last problem in the framework of overlapping graph transformation systems. As parallel transformation of a graph does not produce a graph in general, we propose first some sufficient conditions that ensure the closure of graphs by parallel rewrite relations. Then we mainly introduce and discuss two parallel rewrite relations over graphs. One relation is functional and thus deterministic, the other one is not functional for which we propose sufficient conditions which ensure its confluence.

1 Introduction

Graph structures are fundamental tools that help modeling complex systems. In this paper, we are interested in the evolution of such structures whenever the dynamics is described by means of systems of rewrite rules. Roughly speaking, a rewrite rule can be defined as a pair $l \to r$ where the left-hand and the right-hand sides are of the same structure. A rewrite system, consisting of a set of rewrite rules, induces a rewrite relation $(\to)$ over the considered structures. The rewrite relation corresponds to a sequential application of the rules, that is to say, a structure G rewrites into a structure G' if there exits a rule $l \to r$ such that l occurs in G. Then G' is obtained from G by replacing l by r.

Besides this classical rewrite relation, one may think of a parallel rewrite relation which rewrites a structure G into a structure G' by firing, *simultaneously*, some rules whose left-hand sides occur in G. Simultaneous or parallel rewriting of a structure G into G' can be used as a means to speed up the computations performed by rewrite systems and, in such a case, parallel rewriting can be simulated by successive sequential rewrite steps. However, there are situations where parallel rewrite steps cannot be simulated by sequential steps as in formal grammars [8], cellular automata (CA) [18] or L-systems [14]. This latter problem is of interest in this paper in the case where structures are graphs.

Graph rewriting is a very active area where one may distinguish two main stream approaches, namely (i) the algorithmic approaches where transformations are defined by means of the actual actions one has to perform in order to transform a graph, and (ii) the algebraic approaches where graph transformations are defined in an abstract level using tools borrowed from category theory such as pushouts, pullbacks etc. [15]. In this paper, we introduce a new class of graph rewrite systems following an algorithmic approach where rewrite rules may overlap. That is to say, in the process of graph transformation, it may happen that some occurrences of left-hand sides of different rules can share parts of the graph to be rewritten. This overlapping of the left-hand sides, which can be very appealing in some cases, turns out to be a source of difficulty to define rigorously the notion of parallel rewrite steps. In order to deal with such a difficulty we follow the rewriting modulo approach (see, e.g. [13]) where a rewrite step can be composed with an equivalence relation. Another complication comes from the fact that a graph can be reduced in parallel in a structure which is not always a graph but rather a structure we call *pregraph*. Thus, we propose sufficient conditions under which graphs are closed under parallel rewriting. The rewrite systems we obtain generalize some known models of computation such as CA, L-systems and more generally substitution systems [18].

The paper is organized as follows. The next section introduces the notions of pregraphs and graphs in addition to some preliminary definitions. In Section 3, a class of rewrite systems, called *environment sensitive rewrite systems* is introduced together with a parallel rewrite relation. We show that graphs are not closed under such rewrite relation and propose sufficient conditions under which the outcome of a rewrite step is always a graph. Then, in Section 4, we define two particular parallel rewrite relations, one performs full parallel rewrite steps whereas the second one uses the possible symmetries that may occur in the rules and considers only matches up to automorphisms of the left-hand sides. Section 5 illustrates our framework through some examples. Concluding remarks and related work are given in Section 6. Due to lack of space, the missing proofs are provided in [4].

2 Pregraphs and Graphs

In this section we first fix some notations and give preliminary definitions and properties. 2^A denotes the power set of A. $A \uplus B$ stands for the disjoint union of two sets A and B. In the following, we introduce the notion of (attributed) *pregraphs*, which denotes a class of structures we use to define parallel graph transformations. Elements of a pregraph may be attributed via a function λ which assigns, to elements of a pregraph, attributes in some sets which underly a considered attributes' structure $\mathcal{A}$. For instance $\mathcal{A}$ may be a Σ-algebra [16] or merely a set.

Definition 1 (Pregraph)**.**
A pregraph H is a tuple $H = (\mathcal{N}_H, \mathcal{P}_H, \mathcal{PN}_H, \mathcal{PP}_H, \mathcal{A}_H, \lambda_H)$ such that :

- *$\mathcal{N}_H$ is a finite set of nodes and $\mathcal{P}_H$ is a finite set of ports,*
- *$\mathcal{PN}_H$ is a relation $\mathcal{PN}_H \subseteq \mathcal{P}_H \times \mathcal{N}_H$,*
- *$\mathcal{PP}_H$ is a symmetric binary relation on ports, $\mathcal{PP}_H \subseteq \mathcal{P}_H \times \mathcal{P}_H$,*
- *$\mathcal{A}_H$ is a structure of attributes,*
- *λ_H is a function $\lambda_H : \mathcal{P}_H \uplus \mathcal{N}_H \to 2^{\mathcal{A}_H}$ such that $\forall x \in \mathcal{N}_H \uplus \mathcal{P}_H, card(\lambda_H(x))$ is finite.*

An element (p, n) in $\mathcal{PN}_H$ means that port p is *associated* to node n. An element (p_1, p_2) in $\mathcal{PP}_H$ means that port p is *linked* to port p_2. In a pregraph, a port can be associated (resp. linked) to several nodes (resp. ports).

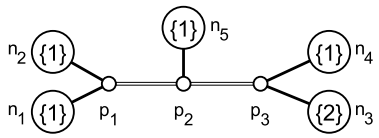


Figure 1: Example of a pregraph H such that: $\mathcal{A}_H = \mathbb{N}$, $\mathcal{N}_H = \{n_1, n_2, n_3, n_4, n_5\}$, $\mathcal{P}_H = \{p_1, p_2, p_3\}$, $\mathcal{PN}_H = \{(p_1, n_1), (p_1, n_2), (p_2, n_5), (p_3, n_3), (p_3, n_4)\}$, $\mathcal{PP}_H = \{(p_1, p_2), (p_2, p_3), (p_2, p_1), (p_3, p_2)\}$. $\mathcal{PP}_H$ could be reduced to its non symmetric port-port connection $\{(p_1, p_2), (p_2, p_3)\}$. $\lambda_H(n_i) = \{1\}$ for $i \in \{1, 2, 4, 5\}$, $\lambda_H(n_3) = \{2\}$. $\lambda_H(p_j) = \emptyset$, for $j \in \{1, 2, 3\}$. Port attributes ($\emptyset$) have not been displayed on the figure.

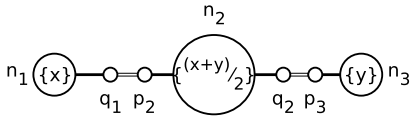


Figure 2: Example of a pregraph H such that: $\mathcal{A}_H = (\mathbb{Q}[x, y]; +, /)$, $\mathcal{N}_H = \{n_1, n_2, n_3\}$, $\mathcal{P}_H = \{p_2, p_3, q_1, q_2\}$, $\mathcal{PN}_H = \{(q_1, n_1), (p_2, n_2), (q_2, n_2), (p_3, n_3)\}$, $\mathcal{PP}_H$ reduced to its non symmetric port-port connection is $\mathcal{PP}_H = \{(p_2, q_1), (p_3, q_2)\}$. $\lambda_H(n_1) = \{x\}$, $\lambda_H(n_2) = \{(x + y)/2\}$, $\lambda_H(n_3) = \{y\}$, $\lambda_H(p_2) = \lambda_H(p_3) = \emptyset$, $\lambda_H(q_1) = \lambda_H(q_2) = \emptyset$.

Example 1. *Figure 1 shows an example of a pregraph where the node attributes are natural numbers and Figure 2 shows an example where attributes could be expressions such as* $\frac{x+y}{2}$.

Below we introduce the definition of graphs used in this paper. In order to encode classical graph edges between nodes, restrictions over port associations are introduced. Intuitively, an edge e between two nodes n_1 and n_2 will be encoded as two semi-edges (n_1, p_1) and (n_2, p_2) with p_1 and p_2 being ports which are linked via an association (p_1, p_2).

Definition 2 (Graph)**.** *A graph, G, is a pregraph $G = (\mathcal{N}, \mathcal{P}, \mathcal{PN}, \mathcal{PP}, \mathcal{A}, \lambda)$ such that :*

(i) $\mathcal{PN}$ is a relation $\subseteq \mathcal{P} \times \mathcal{N}$ which associates at most one node to every port[1]*. That is to say, $\forall p \in \mathcal{P}, \forall n_1, n_2 \in \mathcal{N}, ((p, n_1) \in \mathcal{PN}$ and $(p, n_2) \in \mathcal{PN}) \implies n_1 = n_2$.*

(ii) $\mathcal{PP}$ is a symmetric binary relation[2] *on ports, $\mathcal{PP} \subseteq \mathcal{P} \times \mathcal{P}$, such that $\forall p_1, p_2, p_3 \in \mathcal{P}, ((p_1, p_2) \in \mathcal{PP}$ and $(p_1, p_3) \in \mathcal{PP}) \implies p_2 = p_3$ and $\forall p \in \mathcal{P}, (p, p) \notin \mathcal{PP}$.*

The main idea of our proposal is based on the use of equivalence relations over nodes and ports (merging certain nodes and ports under some conditions) in order to perform parallel graph rewriting in presence of overlapping rules. Thus, to a given pregraph H, we associate two equivalence relations on ports, $\equiv^P$, and on nodes, $\equiv^N$, as defined below.

Definition 3 ($\equiv^P, \equiv^N$)**.** *Let $H = (\mathcal{N}_H, \mathcal{P}_H, \mathcal{PN}_H, \mathcal{PP}_H, \mathcal{A}_H, \lambda_H)$ be a pregraph. We define two relations $\equiv^P$ and $\equiv^N$ respectively on ports ($\mathcal{P}_H$) and nodes ($\mathcal{N}_H$) of H as follows:*

[1] The relation $\mathcal{PN}$ could be seen as a partial function $\mathcal{PN} : \mathcal{P} \to \mathcal{N}$ which associates to a given port p, a node n, $\mathcal{PN}(p) = n$; thus building a semi-edge "port-node".

[2] The relation $\mathcal{PP}$ could also be seen as an injective (partial) function from ports to ports such that $\forall p \in \mathcal{P}, \mathcal{PP}(p) \neq p$ and $\forall p_1, p_2 \in \mathcal{P}, \mathcal{PP}(p_1) = p_2$ iff $\mathcal{PP}(p_2) = p_1$.

- $\equiv^P$ *is defined as* $(\mathcal{PP}_H \bullet \mathcal{PP}_H)^*$
- $\equiv^N$ *is defined as* $(\mathcal{PN}_H^- \bullet \equiv^P \bullet \mathcal{PN})^*$

where $\bullet$ *denotes relation composition,* $^-$ *the converse of a relation and* * *the reflexive-transitive closure of a relation. We write* $[n]$ *(respectively,* $[p]$*) the equivalence class of node* n *(respectively, port* p*).*

Roughly speaking, relation $\equiv^P$ is the closure of the first part of condition (ii) in Definition 2. The base case says that if two ports p_1 and p_2 are linked to a same port p, then p_1 and p_2 are considered to be equivalent. $\equiv^N$ is almost the closure of condition (i) in Definition 2. That is, two nodes n_1 and n_2, which are associated to a same port (or two equivalent ports), are considered as equivalent nodes.

Proposition 1. *The relations* $\equiv^P$ *and* $\equiv^N$ *are equivalence relations.*

Remark 1. *The relations* $\equiv^P$ *and* $\equiv^N$ *can be computed incrementally as follows:*
Base cases: $\equiv_0^P = \{(x,x) \mid x \in \mathcal{P}_H\}$ *and* $\equiv_0^N = \{(x,x) \mid x \in \mathcal{N}_H\}$
Inductive steps:
Rule I*: if* $q, q' \in \mathcal{P}_H$ *such that,* $q \equiv_i^P q'$*,* $(q,p_1) \in \mathcal{PP}_H$ *and* $(q',p_2) \in \mathcal{PP}_H$ *then* $p_1 \equiv_{i+1}^P p_2$*.*
Rule II*: if* $p_1 \in \mathcal{P}_H$*,* $p_2 \in \mathcal{P}_H$*,* $(p_1,n_1) \in \mathcal{PN}_H$*,* $(p_2,n_2) \in \mathcal{PN}_H$ *and* $p_1 \equiv_i^P p_2$ *then* $n_1 \equiv_{i+1}^N n_2$*.*
Rule III*: If* $n_1 \equiv_i^N n'$ *and* $n' \equiv_i^N n_2$ *then* $n_1 \equiv_{i+1}^N n_2$*.*

Proposition 2. *The limits of the series* $(\equiv_i^P)_{i\geq 0}$ *and* $(\equiv_i^N)_{i\geq 0}$ *are respectively* $\equiv^P$ *and* $\equiv^N$*.*

The equivalence relations $\equiv^P$ and $\equiv^N$ are used to introduce the notion of quotient pregraph as defined below.

Definition 4 (Quotient Pregraph)**.** *Let* $H = (\mathcal{N}_H, \mathcal{P}_H, \mathcal{PN}_H, \mathcal{PP}_H, \mathcal{A}_H, \lambda_H)$ *be a pregraph and* $\equiv^P$ *and* $\equiv^N$ *be respectively the two equivalence relations over ports and nodes as introduced in Definition 3. We write* $\overline{H}$ *the pregraph* $\overline{H} = (\mathcal{N}_{\overline{H}}, \mathcal{P}_{\overline{H}}, \mathcal{PN}_{\overline{H}}, \mathcal{PP}_{\overline{H}}, \mathcal{A}_{\overline{H}}, \lambda_{\overline{H}})$ *where* $\mathcal{N}_{\overline{H}} = \{[n] \mid n \in \mathcal{N}_H\}$*,* $\mathcal{P}_{\overline{H}} = \{[p] \mid p \in \mathcal{P}_H\}$*,* $\mathcal{PN}_{\overline{H}} = \{([p],[n]) \mid (p,n) \in \mathcal{PN}_H\}$*,* $\mathcal{PP}_{\overline{H}} = \{([p],[q]) \mid (p,q) \in \mathcal{PP}_H\}$*,* $\mathcal{A}_{\overline{H}} = \mathcal{A}_H$ *and* $\lambda_{\overline{H}}([x]) = \cup_{x' \in [x]} \lambda_H(x')$ *where* $[x] \in \mathcal{N}_{\overline{H}} \uplus \mathcal{P}_{\overline{H}}$

Example 2. *Figure 3 illustrates two computations of quotient pregraphs.*

Remark 2. *If* H *is a graph,* $\overline{H}$ *and* H *are isomorphic. Indeed, in a graph, a port can be associated (resp. linked) to at most one node (resp. one port).*

Below, we define the notion of homomorphisms of pregraphs and graphs. This notion assumes the existence of homomorphisms over attributes [3].

Definition 5 (Pregraph and Graph Homomorphism)**.** *Let* $L = (\mathcal{N}_L, \mathcal{P}_L, \mathcal{PN}_L, \mathcal{PP}_L, \mathcal{A}_L, \lambda_L)$ *and* $G = (\mathcal{N}_G, \mathcal{P}_G, \mathcal{PN}_G, \mathcal{PP}_G, \mathcal{A}_G, \lambda_G)$ *be two pregraphs. Let* $a : \mathcal{A}_L \to \mathcal{A}_G$ *be a homomorphism over attributes. A* pregraph homomorphism*,* $h^a : L \to G$*, between* L *and* G*, built over attribute homomorphism* a*, is defined by two functions* $h_N^a : \mathcal{N}_L \to \mathcal{N}_G$ *and* $h_P^a : \mathcal{P}_L \to \mathcal{P}_G$ *such that (i)* $\forall (p_1,n_1) \in \mathcal{PN}_L$*,* $(h_P^a(p_1), h_N^a(n_1)) \in \mathcal{PN}_G$*, (ii)* $\forall (p_1,p_2) \in \mathcal{PP}_L$*,* $(h_P^a(p_1), h_P^a(p_2)) \in \mathcal{PP}_G$*, (iii)* $\forall n_1 \in \mathcal{N}_L, a(\lambda_L(n_1)) \subseteq \lambda_G(h_N^a(n_1))$ *and (iv)* $\forall p_1 \in \mathcal{P}_L, a(\lambda_L(p_1)) \subseteq \lambda_G(h_P^a(p_1))$*.*
A graph homomorphism is a pregraph homomorphism between two graphs.

Notation: Let E be a set of attributes, we denote by $a(E)$ the set $a(E) = \{a(e) \mid e \in E\}$.

Proposition 3. *Let* H *and* H' *be two isomorphic pregraphs. Then* $\overline{H}$ *and* $\overline{H'}$ *are isomorphic.*

We end this section by defining an equivalence relation over pregraphs.

Definition 6 (Pregraph equivalence)**.** *Let* G_1 *and* G_2 *be two pregraphs. We say that* G_1 *and* G_2 *are equivalent and write* $G_1 \equiv G_2$ *iff the quotient pregraphs* $\overline{G_1}$ *and* $\overline{G_2}$ *are isomorphic.*

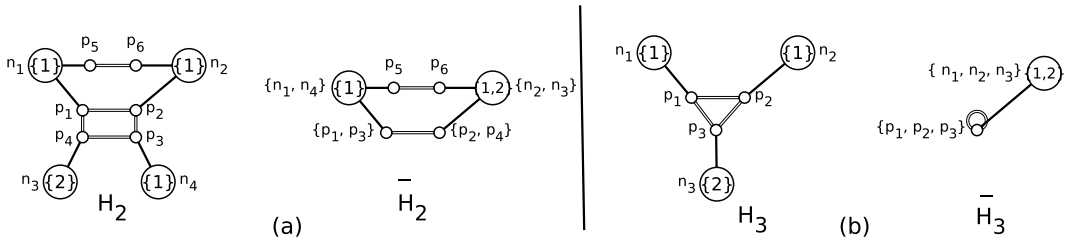


Figure 3: (a) A pregraph H_2 and its corresponding quotient pregraph $\overline{H_2}$ which is a graph. (b) A pregraph H_3 and its corresponding quotient pregraph $\overline{H_3}$ which is not a graph.

3 Graph Rewrite Systems

In this section, we define the considered rewrite systems and provide sufficient conditions ensuring the closure of graph structures under the defined rewriting process.

Definition 7 (Rewrite Rule, Rewrite System, Variant)**.** *A rewrite rule is a pair $l \to r$ where l and r are graphs over the same sets of attributes. A rewrite system $\mathcal{R}$ is a set of rules. A* variant *of a rule $l \to r$ is a rule $l' \to r'$ where nodes, ports as well as the variables of the attributes are renamed with* fresh *names.*

Let $l' \to r'$ be a variant of a rule $l \to r$. Then there is a *renaming mapping* h^a, built over an attribute renaming $a : \mathcal{A}_l \to \mathcal{A}_{l'}$, and consisting of two maps h^a_N and h^a_P over nodes and ports respectively : $h^a_N : \mathcal{N}_l \cup \mathcal{N}_r \to \mathcal{N}_{l'} \cup \mathcal{N}_{r'}$ and $h^a_P : \mathcal{P}_l \cup \mathcal{P}_r \to \mathcal{P}_{l'} \cup \mathcal{P}_{r'}$ such that, the elements in $\mathcal{N}_{l'}$ and $\mathcal{P}_{r'}$ are new and the restrictions of h^a to $l \to l'$ (respectively $r \to r'$) are graph isomorphisms.

In general, parts of a left-hand side of a rule remain unchanged in the rewriting process. This feature is taken into account in the definition below which refines the above notion of rules by decomposing the left-hand sides into an *environmental* part, intended to stay unchanged, and a *cut* part which is intended to be removed. As for the right-hand sides, they are partitioned into a *new* part consisting of added items and an environmental part (a subpart of the left-hand side) which is used to specify how the new part is connected to the environment.

Definition 8 (Environment Sensitive Rewrite Rule, Environment Sensitive Rewrite System)**.** *An environment sensitive rewrite rule is a rewrite rule (ESRR for short) $l \to r$ where l and r are graphs over the same attributes $\mathcal{A}$ such that:*

- $l = (\mathcal{N}_l, \mathcal{P}_l, \not\mathcal{P}\mathcal{N}_l, \not\mathcal{P}\mathcal{P}_l, \mathcal{A}, \lambda_l)$ *where*
$\mathcal{N}_l = \mathcal{N}_l^{cut} \uplus \mathcal{N}_l^{env}, \mathcal{P}_l = \mathcal{P}_l^{cut} \uplus \mathcal{P}_l^{env}, \not\mathcal{P}\mathcal{N}_l = \not\mathcal{P}\mathcal{N}_l^{cut} \uplus \not\mathcal{P}\mathcal{N}_l^{env}, \not\mathcal{P}\mathcal{P}_l = \not\mathcal{P}\mathcal{P}_l^{cut} \uplus \not\mathcal{P}\mathcal{P}_l^{env}$ *and* $\lambda_l = \lambda_l^{cut} \uplus$[3] λ_l^{env} *with some additional constraints :*

  (1) **on** $\not\mathcal{P}\mathcal{N}_l$ **:** $\forall (p, n) \in \not\mathcal{P}\mathcal{N}_l, (n \in \mathcal{N}_l^{cut}$ *or* $p \in \mathcal{P}_l^{cut}) \Rightarrow (p, n) \in \not\mathcal{P}\mathcal{N}_l^{cut}$.

  (2) **on** $\not\mathcal{P}\mathcal{P}_l$ **:** $\forall (p, p') \in \not\mathcal{P}\mathcal{P}_l, p \in \mathcal{P}_l^{cut} \Rightarrow (p, p') \in \not\mathcal{P}\mathcal{P}_l^{cut}$ *and*
$\forall (p, p') \in \not\mathcal{P}\mathcal{P}_l, (p, p') \in \not\mathcal{P}\mathcal{P}_l^{cut} \Leftrightarrow (p', p) \in \not\mathcal{P}\mathcal{P}_l^{cut}$

  (3) **on** λ_l **:** $\forall n \in \mathcal{N}_l^{cut}, (n, \lambda_l(n)) \in \lambda_l^{cut}$ *and* $\forall p \in \mathcal{P}_l^{cut}, (p, \lambda_l(p)) \in \lambda_l^{cut}$.

[3]Here, the function λ_l is considered as a set of pairs $(x, \lambda_l(x))$, i.e. the graph of λ_l.

- $r = (\mathcal{N}_r, \mathcal{P}_r, \mathcal{PN}_r, \mathcal{PP}_r, \mathcal{A}, \lambda_r)$ *where*
$\mathcal{N}_r = \mathcal{N}_r^{new} \uplus \mathcal{N}_r^{env}, \mathcal{P}_r = \mathcal{P}_r^{new} \uplus \mathcal{P}_r^{env}, \mathcal{PN}_r = \mathcal{PN}_r^{new} \uplus \mathcal{PN}_r^{env}, \mathcal{PP}_r = \mathcal{PP}_r^{new} \uplus \mathcal{PP}_r^{env}, \lambda_r = \lambda_r^{new} \uplus \lambda_r^{env}$ *such that* $\mathcal{N}_r^{env} \subseteq \mathcal{N}_l^{env}$, $\mathcal{P}_r^{env} \subseteq \mathcal{P}_l^{env}$, $\mathcal{N}_r^{new} \cap \mathcal{N}_l^{env} = \emptyset$ *and* $\mathcal{P}_r^{new} \cap \mathcal{P}_l^{env} = \emptyset$ *with some additional constraints :*

(4) **on** $\mathcal{PN}_r$ **:** $\forall (p,n) \in \mathcal{PN}_r, (p,n) \in \mathcal{PN}_r^{env}$ *iff* $(p \in \mathcal{P}_r^{env}$ *and* $n \in \mathcal{N}_r^{env}$ *and* $(p,n) \in \mathcal{PN}_l^{env})$.

(5) **on** $\mathcal{PP}_r$ **:** $\forall (p,p') \in \mathcal{PP}_r, (p,p') \in \mathcal{PP}_r^{env}$ *iff* $(p \in \mathcal{P}_r^{env}$ *and* $p' \in \mathcal{P}_r^{env}$ *and* $(p,p') \in \mathcal{PP}_l^{env})$.

(6) **on** λ_r **:** $\forall n \in \mathcal{N}_r^{env}, (\exists y, (n,y) \in \lambda_r^{env})$ *iff* $(\lambda_r^{env}(n) = \lambda_l^{env}(n))$;
$\forall p \in \mathcal{P}_r^{env}, (\exists y, (p,y) \in \lambda_r^{env})$ *iff* $(\lambda_r^{env}(p) = \lambda_l^{env}(p))$.

An environment sensitive rewrite system (ESRS for short) is a set of environment sensitive rewrite rules.

Roughly speaking, constraints (1) and (2) ensure that if an item (node or port) is to be removed (belonging to a "cut" component), links involving that item should be removed too along with its attributes (constraint (3)). This is compatible with the rewriting process defined in the forthcoming Definition 12, which removes all links involving cut items, and thus preventing dangling items. Constraints (4) and (5) ensure that links, considered as new (belonging to "new" components), of a given right-hand side of a rule, should not appear in the left-hand side. Constraint (6) ensures that an item (node or port) is newly attributed in the right-hand side iff it is a new item or it was assigned by λ_l^{cut} in the left-hand side.

Proposition 4. *Let* $l \to r$ *be a an ESRR such that* $l = (\mathcal{N}_l = \mathcal{N}_l^{cut} \uplus \mathcal{N}_l^{env}, \mathcal{P}_l = \mathcal{P}_l^{cut} \uplus \mathcal{P}_l^{env}, \mathcal{PN}_l = \mathcal{PN}_l^{cut} \uplus \mathcal{PN}_l^{env}, \mathcal{PP}_l = \mathcal{PP}_l^{cut} \uplus \mathcal{PP}_l^{env}, \mathcal{A}, \lambda_l = \lambda_l^{cut} \uplus \lambda_l^{env})$ *and* $r = (\mathcal{N}_r = \mathcal{N}_r^{new} \uplus \mathcal{N}_r^{env}, \mathcal{P}_r = \mathcal{P}_r^{new} \uplus \mathcal{P}_r^{env}, \mathcal{PN}_r = \mathcal{PN}_r^{new} \uplus \mathcal{PN}_r^{env}, \mathcal{PP}_r = \mathcal{PP}_r^{new} \uplus \mathcal{PP}_r^{env}, \mathcal{A}, \lambda_r = \lambda_r^{new} \uplus \lambda_r^{env})$.
Then the following properties hold:

- *For all* $(p,n) \in \mathcal{PN}_r$, $(p,n) \in \mathcal{PN}_r^{new}$ *iff* $p \in \mathcal{P}_r^{new}$ *or* $n \in \mathcal{N}_r^{new}$ *or* $(p \in \mathcal{P}_r^{env}$ *and* $n \in \mathcal{N}_r^{env}$ *and* $(p,n) \notin \mathcal{PN}_l^{env})$
- *For all* $(p,p') \in \mathcal{PP}_r$, $(p,p') \in \mathcal{PP}_r^{new}$ *iff* $p \in \mathcal{P}_r^{new}$ *or* $p' \in \mathcal{P}_r^{new}$ *or* $(p \in \mathcal{P}_r^{env}$ *and* $p' \in \mathcal{P}_r^{env}$ *and* $(p,p') \notin \mathcal{PP}_l^{env}(p))$
- *For all* $x \in \mathcal{N}_r \cup \mathcal{P}_r$, $(x, \lambda_r(x)) \in \lambda_r^{new}$ *iff* $x \in \mathcal{N}_r^{new} \cup \mathcal{P}_r^{new}$ *or* $(x, \lambda_l(x)) \in \lambda_l^{cut}$

Example 3. *Let us consider a rule* $R_T : l \to r$ *which specifies a way to transform a triangle into four triangle graphs. Figure 4 depicts the rule. Black parts should be understood as members of the* cut *component of the left-hand side, yellow items are in the* environment *parts. The red items are* new *in the right-hand side. More precisely,* l^{env} *consists of* $\mathcal{N}_l^{env} = \{\alpha, \beta, \gamma\}$, $\mathcal{P}_l^{env} = \{\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2\}$, $\mathcal{PN}_l^{env} = \{(\alpha_1, \alpha), (\alpha_2, \alpha), (\beta_1, \beta), (\beta_2, \beta), (\gamma_1, \gamma), (\gamma_2, \gamma)\}$, *and* $\mathcal{PP}_l^{env} = \emptyset$. *The cut component of the left-hand side consists of three port-port connections and their corresponding symmetric connections which will not be written :* $\mathcal{PP}_l^{cut} = \{(\alpha_2, \beta_1), (\beta_2, \gamma_1), (\gamma_2, \alpha_1)\}$. *The environment component in the right-hand side allows to reconnect the newly introduced items.* r^{env} *consists of the ports* $\mathcal{P}_r^{env} = \{\alpha_1, \alpha_2, \beta_1, \beta_2, \gamma_1, \gamma_2\}$. r^{new} *consists of* $\mathcal{N}_r^{new} = \{U, V, W\}$, $\mathcal{P}_r^{new} = \{u_1, u_2, u_3, u_4, v_1, v_2, v_3, v_4, w_1, w_2, w_3, w_4\}$, $\mathcal{PN}_r^{new} = \{(u_1, U), (u_2, U), (u_3, U), (u_4, U), (v_1, V), (v_2, V), (v_3, V), (v_4, V), (w_1, W), (w_2, W), (w_3, W), (w_4, W)\}$ *and* $\mathcal{PP}_r^{new} = \{(\alpha_1, w_2), (\alpha_2, u_1), (\beta_1, u_2), (\beta_2, v_1), (\gamma_1, v_2), (\gamma_2, w_1), (u_3, w_3), (u_4, v_4), (w_4, v_3)\}$. *The sets of attributes are empty in this example.*

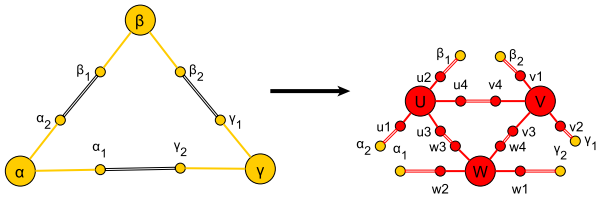


Figure 4: Rule R_T

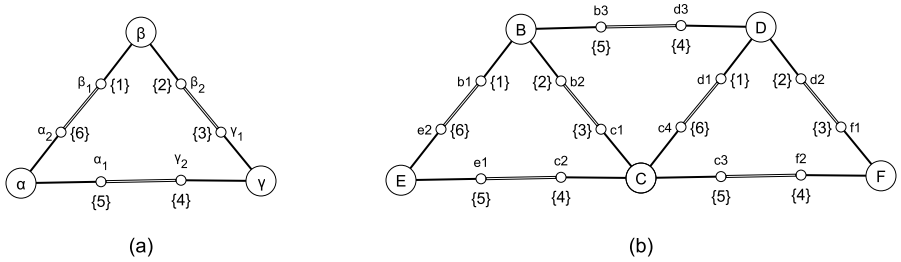


Figure 5: (a) A graph L. (b) A graph G.

Remark 3. *From the definition of an environment sensitive rule, the environment components* $r^{env} = (\mathcal{N}_r^{env}, \mathcal{P}_r^{env}, \mathcal{PN}_r^{env}, \mathcal{PP}_r^{env}, \mathcal{A}, \lambda_r^{env})$ *and* $l^{env} = (\mathcal{N}_l^{env}, \mathcal{P}_l^{env}, \mathcal{PN}_l^{env}, \mathcal{PP}_l^{env}, \mathcal{A}, \lambda_l^{env})$ *are graphs. However, since* $\mathcal{PP}_l^{cut}$ *may include ports in* $\mathcal{P}_l^{env}$ *and* $\mathcal{PN}_l^{cut}$ *may include nodes in* $\mathcal{N}_l^{env}$ *or ports in* $\mathcal{P}_l^{env}$*, the cut component* $l^{cut} = (\mathcal{N}_l^{cut}, \mathcal{P}_l^{cut}, \mathcal{PN}_l^{cut}, \mathcal{PP}_l^{cut}, \mathcal{A}, \lambda_l^{cut})$ *is in general neither a graph nor a pregraph. For the same reasons* $r^{new} = (\mathcal{N}_r^{new}, \mathcal{P}_r^{new}, \mathcal{PN}_r^{new}, \mathcal{PP}_r^{new}, \mathcal{A}, \lambda_r^{new})$ *is in general neither a graph nor a pregraph.*

Finding an occurrence of a left-hand side of a rule within a graph to be transformed consists in finding a *match*. This notion is defined below.

Definition 9 (Match)**.** *Let* L *and* G *be two graphs. A* match $m^a : L \to G$ *is defined as an injective graph homomorphism with* $a : \mathcal{A}_L \to \mathcal{A}_G$ *being an injective homomorphism over attributes.*

Example 4. *Figure 5 gives a graph* L *and a graph* G*. Because of ports attributes, only two matches,* m_1^{id} *and* m_2^{id} *can be defined from* L *to* G*:*

- m_1^{id} *:* $m_1^{id}(\alpha) = E; m_1^{id}(\beta) = B;\ m_1^{id}(\gamma) = C;\ m_1^{id}(\alpha_1) = e_1; m_1^{id}(\alpha_2) = e_2; m_1^{id}(\beta_1) = b_1; m_1^{id}(\beta_2) = b_2;\ m_1^{id}(\gamma_1) = c_1;\ m_1^{id}(\gamma_2) = c_2.$
- m_2^{id} *:* $m_2^{id}(\alpha) = C; m_2^{id}(\beta) = D;\ m_2^{id}(\gamma) = F;\ m_2^{id}(\alpha_1) = c_3; m_2^{id}(\alpha_2) = c_4; m_2^{id}(\beta_1) = d_1; m_2^{id}(\beta_2) = d_2;\ m_2^{id}(\gamma_1) = f_1;\ m_2^{id}(\gamma_2) = f_2.$

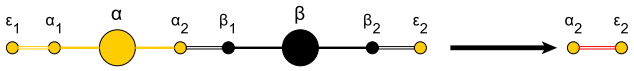
Notice that the occurrences in G *of* $m_1^{id}(L)$ *and* $m_2^{id}(L)$ *overlap on node* C*.*

In order to define the notion of parallel rewrite step, we have to restrict a bit the class of the considered rewrite systems. Indeed, let $l_1 \to r_1$ and $l_2 \to r_2$ be two ESRR. Applying these

two rules in parallel on a graph G is possible only if there is "no conflict" while firing the two rules simultaneously. A conflict may occur if some element of the environment of r_1^{env} is part of l_2^{cut} and vice versa. To ensure conflict free rewriting, we introduce the notion of *conflict free* ESRS. Let us first define the notion of compatible rules.

Definition 10 (compatible rules)**.** *Two ESRR's $l_1 \to r_1$ and $l_2 \to r_2$ are said to be compatible iff for all graphs G and matches $m_1^{a_1} : l_1 \to G$ and $m_2^{a_2} : l_2 \to G$, (i) no element of $m_1^{a_1}(r_1^{env})$ is in $m_2^{a_2}(l_2^{cut})$ and (ii) no element of $m_2^{a_2}(r_2^{env})$ is in $m_1^{a_1}(l_1^{cut})$.*

Conditions (i) and (ii) ensure that the constructions defined by $m_1^{a_1}(r_1)$ (respectively by $m_2^{a_2}(r_2)$) can actually be performed ; i.e, no element used in $m_1^{a_1}(r_1)$ (respectively by $m_2^{a_2}(r_2)$) is missing because of its inclusion in $m_2^{a_2}(l_2^{cut})$ (respectively in $m_1^{a_1}(l_1^{cut})$). For instance, the reader can easily verify that two variants of the rule



are not compatible. Verifying that two given rules are compatible is decidable and can be checked on a finite number, less than $max(size(l_1), size(l_2))$, of graphs where the *size* of a graph stands for its number of nodes and ports.

Proposition 5. *The problem of the verification of compatibility of two rules is decidable.*

Definition 11. *A conflict free ESRS is an ESRS consisting of pairwise compatible rules.*

Definition 12 (parallel rewrite step)**.** *Let $\mathcal{R}$ be a conflict free ESRS $\mathcal{R} = \{L_i \to R_i \mid i = 1 \dots n\}$. Let G be a graph. Let I be a set of variants of rules in $\mathcal{R}$, $I = \{l_i \to r_i \mid i = 1 \dots k\}$ and M a set of matches $M = \{m_i^{a_i} : l_i \to G \mid i = 1 \dots k\}$. We say that graph G rewrites into a pregraph G' using the rules in I and matches in M, written $G \Rightarrow_{I,M} G'$, $G \Rightarrow_M G'$ or simply $G \Rightarrow G'$ if G' is obtained following the two steps below:*
First step: *A pregraph $H = (\mathcal{N}_H, \mathcal{P}_H, \mathcal{PN}_H, \mathcal{PP}_H, \mathcal{A}_H, \lambda_H)$ is computed using the different matches and rules as follows.*

- $\mathcal{N}_H = (\mathcal{N}_G - \cup_{i=1}^k \mathcal{N}^{cut}_{m_i^{a_i}(l_i)}) \uplus \cup_{i=1}^k \mathcal{N}^{new}_{r_i}$
- $\mathcal{P}_H = (\mathcal{P}_G - \cup_{i=1}^k \mathcal{P}^{cut}_{m_i^{a_i}(l_i)}) \uplus \cup_{i=1}^k \mathcal{P}^{new}_{r_i}$
- $\mathcal{PN}_H = ((\mathcal{PN}_G \cap (\mathcal{P}_H \times \mathcal{N}_H)) - \cup_{i=1}^k \mathcal{PN}^{cut}_{m_i^{a_i}(l_i)}) \uplus \cup_{i=1}^k \mathcal{PN}^{new}_{m_i^{a_i}(r_i)}$
- $\mathcal{PP}_H = ((\mathcal{PP}_G \cap (\mathcal{P}_H \times \mathcal{P}_H)) - \cup_{i=1}^k \mathcal{PP}^{cut}_{m_i^{a_i}(l_i)}) \uplus \cup_{i=1}^k \mathcal{PP}^{new}_{m_i^{a_i}(r_i)}$
- $\mathcal{A}_H = \mathcal{A}_G$ *and* $\lambda_H = (\lambda_G - \cup_{i=1}^k \lambda^{cut}_{m_i^{a_i}(l_i)}) \cup \cup_{i=1}^n \lambda^{new}_{m_i^{a_i}(r_i)}$

Second step*:* $G' = \overline{H}$

Notation: Let p, p' be ports and n a node, in notation $m^a(r)$ above, $m^a(p, p') = (m^a(p), m^a(p'))$, $m^a(p, n) = (m^a(p), m^a(n))$, $m^a(p) = p$ if $p \in \mathcal{P}_r^{new}$ and $m^a(n) = n$ if $n \in \mathcal{N}_r^{new}$.

Notice that a parallel rewrite step, as defined above, cannot be achieved by a sequence of single rewrites since, in general, the application of one rule can destroy other rule's matching.

Example 5. *Consider the graph G below and matches m_1 and m_2 of the rule R_T (cf. Figure 4).*

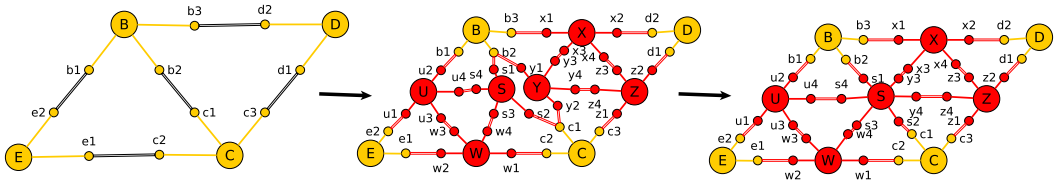
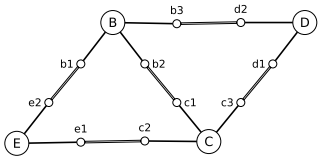


Figure 6: A parallel rewrite step with overlapping between two triangles. Notice that two variants of R_T with fresh new variables have been provided in order to produce the pregraph H. In the quotient graph $\overline{H} = G'$, $[S] = \{S, Y\}$, $[s_1] = \{s_1, y_1\}$, $[s_2] = \{s_2, y_2\}$.



- m_1 : $m_1(\alpha) = E; m_1(\beta) = B$; $m_1(\gamma) = C$; $m_1(\alpha_1) = e_1; m_1(\alpha_2) = e_2; m_1(\beta_1) = b_1; m_1(\beta_2) = b_2$; $m_1(\gamma_1) = c_1$; $m_1(\gamma_2) = c_2$. *The isomorphism of the port-node and port-port connections are easily deduced.*

- m_2 : $m_2(\alpha) = B; m_2(\beta) = D$; $m_2(\gamma) = C$; $m_2(\alpha_1) = b_2; m_2(\alpha_2) = b_3; m_2(\beta_1) = d_2; m_2(\beta_2) = d_1$; $m_2(\gamma_1) = c_3$; $m_2(\gamma_2) = c_1$. *The two matches overlap.*

Figure 6 shows the different steps of the application of two matches of the rule defined in Figure 4. The pregraph, H, in the middle is obtained after the first step of Definition 12. Its quotient pregraph, G', is the graph on the right. G' has been obtained by merging the nodes S and Y and the ports s_1 and y_1 as well as ports s_2 and y_2. These mergings are depicted by the quotient sets $[S], [s_1]$ and $[s_2]$. For sake of readability, the brackets have been omitted for quotient sets reduced to one element.

As a quotient pregraph is not necessarily a graph (see Figure 3), the above definition of parallel rewrite step does not guarantee, in general, the production of graphs only. Hence, we propose hereafter a sufficient condition, which could be verified syntactically, that ensures that the outcome of a parallel rewrite step is still a graph.

Theorem 1. *Let $\mathcal{R}$ be a conflict free ESRS $\mathcal{R} = \{L_i \to R_i \mid i = 1 \dots n\}$. Let G be a graph. Let I be a set of variants of rules in $\mathcal{R}$, $I = \{l_i \to r_i \mid i = 1 \dots k\}$ and M a set of matches $M = \{m_i^{a_i} : l_i \to G \mid i = 1 \dots k\}$. Let G' be the pregraph such that $G \Rightarrow_{I,M} G'$. If $\forall p, p' \in \mathcal{P}_{l_i}^{env}$, $(p, p') \notin \mathcal{PP}_{r_i}^{new}$, then G' is a graph.*

4 Two Parallel Rewrite Relations

The set of matches, M, in Definition 12 is not constrained and thus the induced parallel rewrite relation is nondeterministic since at each step one may choose several sets of matches leading to different rewrite outcomes. In this section, we are rather interested in two confluent parallel rewrite relations which are realistic and can be good candidates for implementations. The first one performs all possible reductions (up to node and port renaming) whereas the second relation is more involved and performs reductions up to left-hand sides' automorphisms.

4.1 Full Parallel Rewrite Relation

We start by a technical definition of an equivalence relation, $\approx$, over matches.

Definition 13 ($\approx$). *Let $L \to R$ be a rule and G a graph. Let $l_1 \to r_1$ and $l_2 \to r_2$ be two variants of the rule $L \to R$. We denote by $h_1^{a_1}$ (respect. $h_2^{a_2}$) the (node, port and attribute) renaming mapping such that the restriction of $h_1^{a_1}$ (respectively, $h_2^{a_2}$) to $L \to l_1$ (respectively $L \to l_2$) is a graph isomorphism. Let $m_1^{b_1} : l_1 \to G$ and $m_2^{b_2} : l_2 \to G$ be two matches. We say that $m_1^{b_1}$ and $m_2^{b_2}$ are equivalent and write $m_1^{b_1} \approx m_2^{b_2}$ iff for all elements x (in $\mathcal{P}_L$, $\mathcal{N}_L$, $\mathcal{PP}_L$ or $\mathcal{PN}_L$) of L, $m_1^{b_1}(h_1^{a_1}(x)) = m_2^{b_2}(h_2^{a_2}(x))$ and for all x in $\mathcal{A}_L$, $b_1(a_1(x)) = b_2(a_2(x))$.*

The relation $\approx$ is clearly an equivalence relation. Intuitively, two matches $m_1^{b_1} : l_1 \to G$ and $m_2^{b_2} : l_2 \to G$ are equivalent, $m_1^{b_1} \approx m_2^{b_2}$, whenever (i) l_1 and l_2 are left-hand sides of two variants of a same rule, say $L \to R$, and (ii) $m_1^{b_1}$ and $m_2^{b_2}$ coincide on each element x of L.

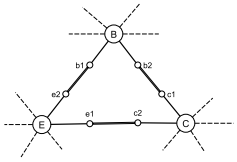
Definition 14 (full parallel matches). *Let $\mathcal{R}$ be a graph rewrite system and G a graph. Let $\mathcal{M}_{\mathcal{R}}(G) = \{m_i^{a_i} : l_i \to G \mid m_i^{a_i}$ is a match and $l_i \to r_i$ is a variant of a rule in $\mathcal{R}\}$. A set, M, of* full parallel matches*, with respect to a graph rewrite system $\mathcal{R}$ and a graph G, is a maximal set such that (i) $M \subset \mathcal{M}_{\mathcal{R}}(G)$ and (ii) $\forall m_1^{a_1}, m_2^{a_2} \in M, m_1^{a_1} \not\approx m_2^{a_2}$.*

A set of full parallel matches M is not unique because any rule in $\mathcal{R}$ may have infinitely many variants. However the number of non equivalent matches could be easily proven to be finite.

Definition 15 (full parallel rewriting). *Let $\mathcal{R}$ be a conflict free ESRS and G a graph. Let M be a set of full parallel matches with respect to $\mathcal{R}$ and G. We define the* full parallel *rewrite relation and write $G \rightrightarrows_M G'$ or simply $G \rightrightarrows G'$, as the parallel rewrite step $G \Rightarrow_M G'$.*

Proposition 6. *Let $\mathcal{R}$ be a conflict free ESRS. The rewrite relation $\rightrightarrows$ is deterministic. That is to say, for all graphs G, ($G \rightrightarrows G_1$ and $G \rightrightarrows G_2$) implies that G_1 and G_2 are isomorphic.*

Example 6.



Let us consider the rule R_T defined in Figure 4 and the subgraph S depicted on the side. The reader can verify that there are six different matches, $m_1 \dots m_6$, between the left-hand side of R_T and graph S.

These matches are sketched below. Variants of R_T have been omitted for sake of readability.

- m_1 : $m_1(\alpha) = E; m_1(\beta) = B;\ m_1(\gamma) = C;\ m_1(\alpha_1) = e_1; m_1(\alpha_2) = e_2; m_1(\beta_1) = b_1; m_1(\beta_2) = b_2;\ m_1(\gamma_1) = c_1;\ m_1(\gamma_2) = c_2$.
- m_2 : $m_2(\alpha) = E; m_2(\beta) = C;\ m_2(\gamma) = B;\ m_2(\alpha_1) = e_2; m_2(\alpha_2) = e_1; m_2(\beta_1) = c_2; m_2(\beta_2) = c_1;\ m_2(\gamma_1) = b_2;\ m_2(\gamma_2) = b_1$.
- m_3 : $m_3(\alpha) = B; m_3(\beta) = E;\ m_3(\gamma) = C;\ m_3(\alpha_1) = b_2; m_3(\alpha_2) = b_1; m_3(\beta_1) = e_2; m_3(\beta_2) = e_1;\ m_3(\gamma_1) = c_2;\ m_3(\gamma_2) = c_1$.
- m_4 : $m_4(\alpha) = B; m_4(\beta) = C;\ m_4(\gamma) = E;\ m_4(\alpha_1) = b_1; m_4(\alpha_2) = b_2; m_4(\beta_1) = c_1; m_4(\beta_2) = c_2;\ m_3(\gamma_1) = e_1;\ m_4(\gamma_2) = e_2$.

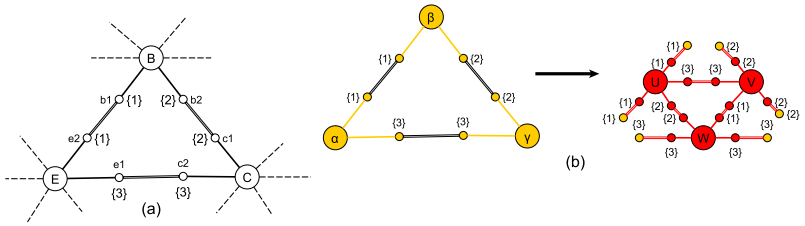


Figure 7: (a) Subgraph S with distinguishing attributes on ports. The attributes are $\{1, 2, 3\}$. (b) Rule R_T with distinguishing attributes.

- m_5 : $m_5(\alpha) = C; m_5(\beta) = B$; $m_5(\gamma) = E$; $m_5(\alpha_1) = c_2; m_5(\alpha_2) = c_1; m_5(\beta_1) = b_2; m_5(\beta_2) = b_1$; $m_5(\gamma_1) = e_2$; $m_5(\gamma_2) = e_1$.
- m_6 : $m_5(\alpha) = C; m_5(\beta) = E$; $m_5(\gamma) = B$; $m_5(\alpha_1) = c_1; m_5(\alpha_2) = c_2; m_5(\beta_1) = e_1; m_5(\beta_2) = e_2$; $m_5(\gamma_1) = b_1$; $m_5(\gamma_2) = b_2$.

Here, the homomorphisms over attributes are always the identity, that is why they have been omitted. Thanks to the six matches and the rule R_T, the reader may check that the subgraph S can be rewritten, by using six different variants of rule R_T, into a pregraph containing 3×6 new nodes and 12×6 new ports. The quotient pregraph has only 3 new nodes but has 42 new ports. Each pair of new nodes has 6 connections.

This example shows that the full parallel rewriting has to be used carefully since it may produce non intended results due to overmatching the same subgraphs. To overcome this issue, one may use attributes in order to lower the possible matches. We call such attributes *distinguishing attributes*. In order to consider only one match of the subgraph S considered in Example 6 by the rule R_T, one option is to apply full parallel rewrite relation with distinguishing attributes on the subgraph depicted in Figure 7 (a) and rule R_T with distinguishing attributes given in Figure 7 (b), leading to a pregraph whose quotient is a graph with 3 new nodes and 12 new ports. This graph is the expected one.

Another way to mitigate the problems of overmatching subgraphs, in addition to the use of distinguishing attributes, consists in taking advantage of the symmetries that appear in the graphs of rewrite rules. This leads us to define a new rewrite relation which gets rid of multiple matches of the same left-hand-side of a fixed rule. We call this relation *parallel up to automorphisms* and is defined below.

4.2 Parallel Rewrite Relation up to Automorphisms

Let us consider a graph G which rewrites into G_1 and G_2 using an ESRR $l \to r$. This means that there exist two matches $\mu_i^{\beta_i} : l \to G$ with $i \in \{1, 2\}$ such that $G \Rightarrow_{l \to r, \mu_i^{\beta_i}} G_i$. One may wonder whether G_1 and G_2 are the same (up to isomorphism) whenever matches $\mu_1^{\beta_1}$ and $\mu_2^{\beta_2}$ are linked by means of an automorphism of l. That is to say, when there exists an automorphism $h^a : l \to l$ with $\mu_1^{\beta_1} = \mu_2^{\beta_2} \circ h^a$. Intuitively, matches $\mu_1^{\beta_1}$ and $\mu_2^{\beta_2}$ could be considered as the same up to a permutation of nodes. We show below that G_1 and G_2 are actually isomorphic but under some syntactic condition we call *symmetry condition*.

Notation: Let G be a graph with attributes in $\mathcal{A}$. We write $H(G)$ to denote the set of automorphisms of G, i.e. $H(G)$ is the set of isomorphisms $h^a : G \to G$, with a being an isomorphism on the attributes of G, $a : \mathcal{A} \to \mathcal{A}$.

Proposition 7. *Let $l \to r$ be an ESRR. let $l_1 \to r_1$ and $l_2 \to r_2$ be two variants of the rule $l \to r$. Let $v_1^{c_1}$, $v_1'^{c_1}$, $v_2^{c_2}$, $v_2'^{c_2}$ be the isomorphisms reflecting the variant status of these two rules with $v_1^{c_1} : l \to l_1$, $v_1'^{c_1} : r \to r_1$, $v_2^{c_2} : l \to l_2$, $v_2'^{c_2} : r \to r_2$, $l_i = v_i^{c_i}(l)$, $r_i = v_i'^{c_i}(r)$ and $v_i^{c_i}(r_i^{env}) = v_i'^{c_i}(r_i^{env})$ for $i \in \{1,2\}$. Let G be a graph and G_1' and G_2' be two pregraphs. Let $G \Rightarrow_{l_1 \to r_1, m_1^{b_1}} G_1'$ and $G \Rightarrow_{l_2 \to r_2, m_2^{b_2}} G_2'$ be two rewrite steps such that there exist two automorphisms $h^a : l \to l$ and $h'^a : r \to r$ such that (i) with $m_1^{b_1} = m_2^{b_2} \circ v_2^{c_2} \circ h^a \circ (v_1^{c_1})^{-1}$ and (ii) for all elements x of r^{env}, $h'^a(x) = h^a(x)$. Then, G_1' and G_2' are isomorphic.*

Sketch. The sketch of the proof is depicted in Figure 8. The attributes structures used in the rule $l \to r$ (respectively, $l_1 \to r_1$ and $l_2 \to r_2$) are denoted A (respectively, A_1 and A_2) whereas the attibutes structure of the transfomed graph G is denoted B. From the hypotheses, we can easily infer the exitence of two isomorphisms $h_v^{c_v} : l_1 \to l_2$ and $h_v'^{c_v} : r_1 \to r_2$ such that $h_v^{c_v} = v_2^{c_2} \circ h^a \circ (v_1^{c_1})^{-1}$ and $h_v'^{c_v} = v_2'^{c_v} \circ h'^a \circ (v_1'^{c_1})^{-1}$. And we have $c_v = c_2 \circ a \circ c_1^{-1}$.

Let $G \Rightarrow_{l_1 \to r_1, m_1^{b_1}} G_1'$ and $G \Rightarrow_{l_2 \to r_2, m_2^{b_2}} G_2'$ such that $m_1^{b_1}(l_1) = m_2^{b_2}(l_2)$. By definition of a rewrite step, there exist a pregraph G_1 (respect. a pregraph G_2) and an injective homomorphism $m_1'^{b_1} : r_1 \to G_1$ (respect. $m_2'^{b_2} : r_2 \to G_2$) such that $G_1' = \overline{G}_1$ (respect. $G_2' = \overline{G}_2$). Moreover, since, by definition, r^{env} is included in l^{env} for any ESRR $l \to r$, we have $m_1'^{b_1}(r_1^{env}) = m_1^{b_1}(r_1^{env})$ (respect. $m_2'^{b_2}(r_2^{env}) = m_2^{b_2}(r_2^{env})$), where $m_i'^{b_i}$, for $i \in \{1,2\}$, are defined as follows: for $n \in \mathcal{N}_{r_i}, m'^{b_i}_i(n) = \begin{cases} m_i^{b_i}(n) \; if \; n \in \mathcal{N}_{r_i}^{env} \\ n \; otherwise \end{cases}$ for $p \in \mathcal{P}_{r_i}, m'^{b_i}_i(p) = \begin{cases} m_i^{b_i}(p) \; if \; p \in \mathcal{P}_{r_i}^{env} \\ p \; otherwise \end{cases}$

Now, let us define the isomorphism $h''^d : G_1 \to G_2$ with $d(x) =$ if $x \in b_1(A_1)$ then $b_2 \circ c_v \circ b_1^{-1}(x)$ else x. Let us consider x such that x is an element of r^{env} (port or node). We have $m_1^{b_1}(v_1^{c_1}(x)) = m_2^{b_2}(v_2^{c_2}(h^a(x)))$ is an element of G. Moreover $m_1'^{b_1}(v_1'^{c_1}(x)) \in G_1$ and $m_2'^{b_2}(v_2'^{c_2}(h'^a(x)) \in G_2$. Let us denote $y = m_1^{b_1}(v_1^{c_1}(x))$. By construction $m_1'^{b_1}(v_1'^{c_1}(x)) = m_1^{b_1}(v_1^{c_1}(x)) = y$ because $x \in r^{env}$. From the hypothesis we have $h^a(x) = h'^a(x)$. Thus $m_2'^{b_2}(v_2'^{c_2}(h'^a(x)) = m_2'^{b_2}(v_2'^{c_2}(h^a(x))$ and then we have $m_2'^{b_2}(v_2'^{c_2}(h'^a(x)) = m_2^{b_2}(v_2^{c_2}(h^a(x)) = y$. Then, for all elements z of the non-modified part of G which is $G - m_1^{b_1}(v_1^{c_1}(l))$ (z can be a port or a node if y is not a node) such that $(z,y) \in G$, we have that $(z,y) \in G_1$ and $(z,y) \in G_2$ and $h''^d = Id^d$ on $G_1 - m_1'^{b_1}(v_1'^{c_1}(r))$. Finally the definition of h''^d is :

For $y \in \mathcal{N}_{G_1} \cup \mathcal{P}_{G_1}, h''^d(y) = \begin{cases} m_2'^{b_2}(h_v'^{c_v}((m_1'^{b_1})^{-1}(y))) \; if \; y \in m_1'^{b_1}(r_1) \\ y \; otherwise \end{cases}$

For all types of existing connections (y,z) of G_1 where y and z in $\mathcal{N}_{G_1} \cup \mathcal{P}_{G_1}$, $h''^d(y,z) = (h''^d(y), h''^d(z))$ is in G_2. By construction, the homomorphism conditions on attributes are fulfilled by h''^d. Thus, $h''^d : G_1 \to G_2$ is a pregraph homomorphism. In addition, h''^d is bijective by construction. From h''^d and Proposition 3, we infer the isomorphism $h^{(3)d} : G_1' \to G_2'$.

□

Definition 16 (Symmetry Condition)**.** *An ESRR $l \to r$ verifies the* symmetry condition *iff* $\forall h^a \in H(l), \exists\, h'^a \in H(r),$ *such that* $\forall x \in r^{env}, h^a(x) = h'^a(x)$

The reader can check that the rule R_T verifies the symmetry condition.

Definition 17 (Matches up to automorphism, $\sim^l$)**.** *Let $l \to r$ be an ESRR satisfying the symmetry condition. Let $l_1 \to r_1$ and $l_2 \to r_2$ be two different variants of the rule $l \to r$. Let $v_1^{c_1} : l \to l_1$ and $v_2^{c_2} : l \to l_2$ be the isomorphisms that reflect the variant status of l_1 and l_2 of*

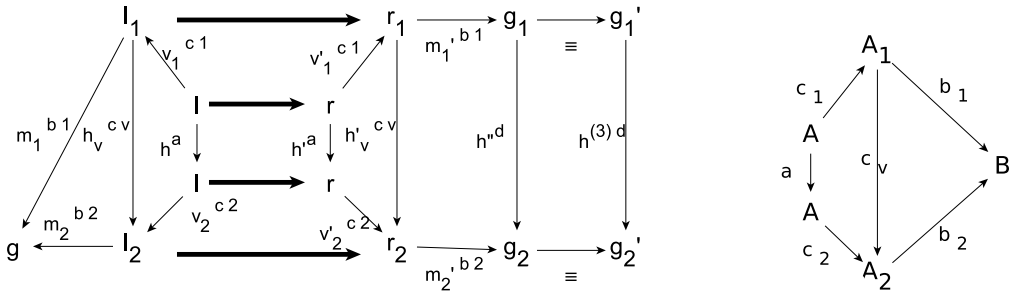


Figure 8: Sketch of the proof of Proposition 7

l. Let $m_1^{b_1} : l_1 \to g$ and $m_2^{b_2} : l_2 \to g$ be two matches such that $m_1^{b_1}(l_1) = m_2^{b_2}(l_2)$. We say that matches $m_1^{b_1}$ and $m_2^{b_2}$ are equal up to (l-)automorphism and write $m_1^{b_1} \sim^l m_2^{b_2}$ iff there exists an automorphism $h^a : l \to l$ such that $m_1^{b_1} = m_2^{b_2} \circ v_2^{c_2} \circ h^a \circ v_1^{c_1 -1}$.

Definition 18 (Rewriting up to automorphisms)**.** *Let $\mathcal{R}$ be a conflict free ESRS whose rules satisfy the symmetry condition and G a graph. Let $M(\mathcal{R}, G)^{auto} = \{m_i^{a_i} : l_i \to G \mid l_i$ is the lhs side of a variant of a rule $l \to r$ in $\mathcal{R}$ and $m_i^{a_i}$ is a match up to automorphism$\}$. We define the rewrite relation $\rightrightarrows_{auto}$ which rewrites graph G by considering only matches up to automorphisms. I.e., the set of matches M of Definition 12 is $M(\mathcal{R}, G)^{auto}$.*

Theorem 2. *Let $\mathcal{R}$ be a conflict free ESRS whose rules satisfy the symmetry condition. Then $\rightrightarrows_{auto}$ is deterministic. That is, for all graphs G, ($G \rightrightarrows_{auto} G'_1$ and $G \rightrightarrows_{auto} G'_2$) implies that G'_1 and G'_2 are isomorphic.*

5 Examples

We illustrate the proposed framework through three examples borrowed from different fields. We particularly provide simple confluent rewrite systems encoding cellular automata, the koch snowflake and the mesh refinement.

5.1 Cellular automata (CA)

A cellular automaton is based on a fixed grid composed of cells. Each cell computes its new state synchronously. At instant $t+1$, the value of a state k, denoted $x_k(t+1)$ may depend on the valuations at instant t of the state k itself, $x_k(t)$, and the states $x_n(t)$ such that n is a neighbor of k. Such a formula is of the following shape, where f is a given function and $\nu(k)$ is the set of the neighbors of cell k: $x_k(t+1) = f(x_k(t), x_n(t), n \in \nu(k))$ In the case of a graph G, the neighbors of a cell (node) k, $\nu(k)$, is defined by : $l \in \nu(k)$ iff $\exists p_1, \exists p_2, (p_1, k) \in \mathcal{PN}_G \wedge (p_2, l) \in \mathcal{PN}_G \wedge (p_1, p_2) \in \mathcal{PP}_G$. Usually, the grid is oriented such that any cell of $\nu(k)$ has a unique relative position with respect to the cell k. This orientation is easily modeled by distinguishing attributes on ports. For instance, one can consider Moore's neighborhood [6] on a 2-dimensional grid. This neighborhood of radius 1 is composed of 8 neighbors. The distinguishing attributes on ports belong to the set $\{e, w, n, s, ne, se, nw, sw\}$ which defines the 8 directions where e = east, w = west, n = north, s = south etc.

The grid is defined by a graph $G = (\mathcal{N}_G, \mathcal{P}_G, \mathcal{PN}_G, \mathcal{PP}_G, \mathcal{A}_G, \lambda_G)$ such that :

- $\mathcal{N}_G = \{m_{i,j}\}_{i\in I, j\in J}$, where intervals I and J are defined as $I = [-N, N] \cap \mathbb{Z}$ and $J = [-N', N'] \cap \mathbb{Z}$ for some natural numbers N and N'.
- $\mathcal{P}_G = \{e_{i,j}, w_{i,j}, s_{i,j}, n_{i,j}, ne_{i,j}, nw_{i,j}, se_{i,j}, sw_{i,j} |\ i \in I, j \in J\}$,
- $\mathcal{PN}_G = \{(e_{i,j}, m_{i,j}), (w_{i,j}, m_{i,j}), (s_{i,j}, m_{i,j}), (n_{i,j}, m_{i,j}), (ne_{i,j}, m_{i,j}), (nw_{i,j}, m_{i,j}), (se_{i,j}, m_{i,j}), (sw_{i,j}, m_{i,j}) | i \in I, j \in J\}$,
- $\mathcal{PP}_G = \{(e_{i,j}, w_{i,j+1}), (w_{i,j}, e_{i,j-1}), (n_{i,j}, s_{i-1,j}), (s_{i,j}, n_{i+1,j}), (ne_{i,j}, sw_{i-1,j+1}), (se_{i,j}, nw_{i+1,j+1}), (nw_{i,j}, se_{i-1,j-1}), (sw_{i,j}, ne_{i+1,j-1}) |\ i \in I, j \in J\}$,
- $\forall i \in I, \forall j \in J,\ \lambda_G(m_{i,j}) \subseteq \mathcal{A}_G$,
- $\forall i \in I, \forall j \in J,\ \lambda_G(e_{i,j}) = \{e\},\ \lambda_G(w_{i,j}) = \{w\}, \lambda_G(s_{i,j}) = \{s\},\ \lambda_G(n_{i,j}) = \{n\},\ \lambda_G(ne_{i,j}) = \{ne\},\ \lambda_G(nw_{i,j}) = \{nw\},\ \lambda_G(se_{i,j}) = \{se\},\ \lambda_G(sw_{i,j}) = \{sw\}$.

The attributes of the nodes correspond to states of the cells. They belong to a set $\mathcal{A}_G$. To implement the dynamics of the automaton one needs only one rewrite rule $\{\rho = l \to r\}$ which corresponds to the function f. The rule does not modify the structure of the grid but modifies the attributes of nodes. Thus a left-hand side has a structure of a star with one central node (see Figure 9), for which the rule at hand expresses its dynamics, surrounded by its neighbors. Nodes, ports and edges of the left-hand side belong to the environment part of the rule. Only the attribute of the central node belongs to the cut part since this attribute is modified by the rule. In the left-hand-side, the attributes of nodes are variables to which values are assigned during the matches. The right-hand-side is reduced to a single node named i. Its attribute corresponds to the new part of the right-hand side.

Figure 9 illustrates such rules by implementing the well known *game of life*. It is defined using Moore's neighborhood and the dynamics of the game is defined on a graph G such that attributes of nodes are in $\{0, 1\}$ and

$x_i(t+1) = ((\sum_{l\in\nu(i)} x_l(t)) =? = 3) + ((x_i(t) =? = 1) \times (\sum_{l\in\sigma(i)} x_l(t)) =? = 2))$

where $(x =? = y) \Leftrightarrow \begin{cases} 1\ if\ x = y \\ 0\ otherwise \end{cases}$

The neighborhood of a node i and its dynamics verify the symmetry condition, thus there is no need to define attributes on ports. The rewriting relation $\rightrightarrows_{auto}$ is applied on the rewrite system $\mathcal{R} = \{\rho = l \to r\}$ reduced to one rule depicted in Figure 9. More precisely the graphs of the rule as defined as follows:

$l = (\mathcal{N}_l, \mathcal{P}_l, \mathcal{PN}_l, \mathcal{PP}_l, \mathcal{A}_l, \lambda_l)$ with

- $\mathcal{N}_l = \mathcal{N}_l^{env} = \{i, a, b, c, d, e, f, g, h\}$,
- $\mathcal{P}_l = \mathcal{P}_l^{env} = \{i_1, i_2, i_3, i_4, i_5, i_6, i_7, i_8, a_1, b_1, c_1, d_1, e_1, f_1, g_1, h_1\}$,
- $\mathcal{PN}_l = \mathcal{PN}_l^{env} = \{(a_1, a), (b_1, b), (c_1, c), (d_1, d), (e_1, e), (f_1, f), (g_1, g), (h_1, h), (i_1, i)(i_2, i), (i_3, i), (i_4, i), (i_5, i), (i_6, i), (i_7, i), (i_8, i)\}$,
- $\mathcal{PP}_l = \mathcal{PP}_l^{env} = \{(i_1, a_1)(i_2, b_1), (i_3, c_1), (i_4, d_1), (i_5, e_1), (i_6, f_1), (i_7, g_1), (i_8, h_1)\}$.
- $\mathcal{A}_l = \{0, 1, x_i\} \cup \{y_q \mid q \in \{a, b, c, d, e, f, g, h\}\}$ and $\lambda_l = \lambda_l^{env} \cup \lambda_l^{cut}$ with $\lambda_l^{cut} : \{i\} \to \mathcal{A}_l$ such that $\lambda_l^{cut}(i) = \{x_i\}$; and $\lambda_l^{env} : \{a, b, c, d, e, f, g, h\} \to \mathcal{A}_l$ such that $\lambda_l^{env}(q) = \{y_q\}$

$r = (\mathcal{N}_r, \mathcal{P}_r, \mathcal{PN}_r, \mathcal{PP}_r, \mathcal{A}_r, \lambda_r)$ with

- $\mathcal{N}_r = \mathcal{N}_r^{env} = \{i\}$,

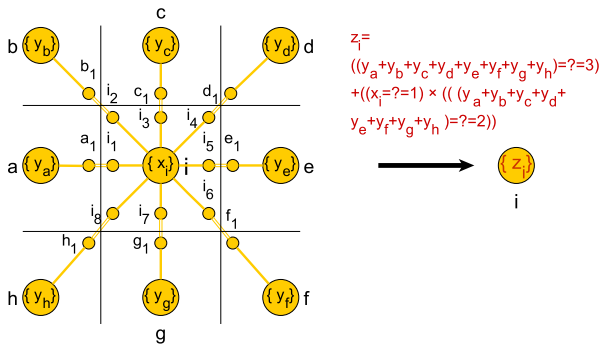


Figure 9: game of life rule

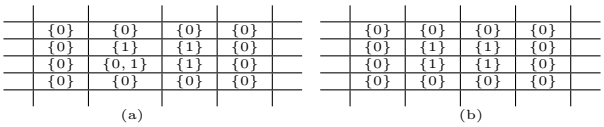


Figure 10: (a) initial grid; (b) fixed point

- $\mathcal{P}_r = \emptyset$, $\mathcal{PN}_r = \emptyset$, $\mathcal{PP}_r = \emptyset$.
- $\mathcal{A}_r = \mathcal{A}_l$
- Moreover, on nodes, $\lambda_r = \lambda_r^{new}$ (λ_r^{env} being empty) with $\lambda_r^{new} : \{i\} \rightarrow Att_r$ and $\lambda_r^{new}(i) = \{((y_a + y_b + y_c + y_d + y_e + y_f + y_g + y_h) =? = 3) + ((x_i =? = 1) \times ((y_a + y_b + y_c + y_d + y_e + y_f + y_g + y_h) =? = 2))\}$.

In the classical formulation of cellular automata, a cell contains one and only one value. The model we propose can deal with cells with one or several values. For instance, the initial state of the game of life can be a grid containing $\{0\}$'s except for 4 cells describing a square (see Figure 10(a)).

In this configuration one cell have 2 values which means, on the example, that the cell is dead or alive or we don't have any information on the state of the cell. The behavior of all possible trajectories is computed in parallel and the fixed point is reached. The initial state Figure 11(a) yields Figure 11(b) as a fixed point. Here we observe that the indeterminacy concerns at most 4 cells over time.

5.2 The Koch snowflake

The well-known Koch snowflake is based on segment divisions (variants exist on surfaces, both can be modeled by our formalism). Each segment is recursively divided into three segments of equ

Let us consider the following triangle G as an initial state.

$G = (\mathcal{N}_G, \mathcal{P}_G, \mathcal{PN}_G, \mathcal{PP}_G, \mathcal{A}_G, \lambda_G)$ with $\mathcal{N}_G = \{1, 2, 3\}$, $\mathcal{P}_G = \{p_1, q_1, p_2, q_2, p_3, q_3\}$, $\mathcal{PN}_G = \{(p_1, 1), (q_1, 1), (p_2, 2), (q_2, 2), (p_3, 3), (q_3, 3)\}$, $\mathcal{PP}_G = \{(p_1, q_2), (p_2, q_3), (p_3, q_1)\}$.

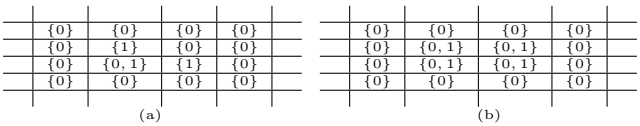


Figure 11: (a) initial grid; (b) fixed point

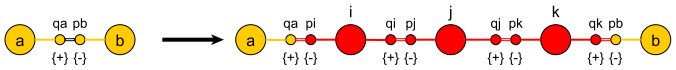


Figure 12: Koch Snowflake rule $l \to r$ with the node attribute computation $\lambda_r(i) = \frac{2}{3}\lambda_l(a) + \frac{1}{3}\lambda_l(b)$, $\lambda_r(j) = \frac{1}{2}(\lambda_l(a) + \lambda_l(b)) + \frac{\sqrt{3}}{6}(-\lambda_l(a)^T + \lambda_l(b)^T)$, $\lambda_r(k) = \frac{1}{3}\lambda_l(a) + \frac{2}{3}\lambda_l(b)$

$\lambda_G(1) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$, $\lambda_G(2) = \begin{pmatrix} 0 \\ \sqrt{2} \end{pmatrix}$, $\lambda_G(3) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, $\lambda_G(p_1) = \lambda_G(p_2) = \lambda_G(p_3) = \{-\}$, $\lambda_G(q_1) = \lambda_G(q_2) = \lambda_G(q_3) = \{+\}$.

The attributes of ports are distinguishing attributes. The attributes of nodes are the $\mathbb{R}^2$ positions of the nodes. Every node got one attribute in $\mathbb{R}^2$, thus by abuse of notation, we get rid of the set notation of attributes and use a functional one. The implementation of both relations $\rightrightarrows$ and $\rightrightarrows_{auto}$ using the rule depicted in Figure 12 provide the expected pictures of flakes as in Figures 13.

Let us denote $\lambda_l(a) = \begin{pmatrix} x_a \\ y_a \end{pmatrix}$ and $\lambda_l(b) = \begin{pmatrix} x_b \\ y_b \end{pmatrix}$. In this example, the attributes of nodes i, j and k are defined as follows: $\lambda_r(i) = \frac{2}{3}\lambda_l(a) + \frac{1}{3}$ $\lambda_l(b) = \begin{pmatrix} \frac{2}{3}x_a + \frac{1}{3}x_b \\ \frac{2}{3}y_a + \frac{1}{3}y_b \end{pmatrix}$,

$\lambda_r(j) = \frac{1}{2}(\lambda_l(a) + \lambda_l(b)) + \frac{\sqrt{3}}{6}(\lambda_l(a)^T + \lambda_l(b)^T) = \begin{pmatrix} \frac{1}{2}(x_a + x_b) + \frac{\sqrt{3}}{6}(y_a - y_b) \\ \frac{1}{2}(y_a + y_b) + \frac{\sqrt{3}}{6}(-x_a + x_b) \end{pmatrix}$, and

$\lambda_r(k) = \frac{1}{3}\lambda_l(a) + \frac{2}{3}\lambda_l(b) = \begin{pmatrix} \frac{1}{3}x_a + \frac{2}{3}x_b \\ \frac{1}{3}y_a + \frac{2}{3}y_b \end{pmatrix}$

5.3 Mesh refinement

Mesh refinement consists in creating iteratively new partitions of the considered space. The initial mesh G we consider is depicted Figure 15. Distinguishing attributes are given on ports. Attributes on nodes are omitted but we can easily consider coordinates. Triangle refinements are given in Figure 14. The three rules verify the symmetry condition and we apply the $\rightrightarrows_{auto}$ relation on G to obtain the graph G' described in Figure 15. Iteratively, the rewrite system can be applied again on G' and so forth.

6 Conclusion and Related Work

Parallel rewriting technique is a tough issue when it has to deal with overlapping reducible expressions. In this paper, we have proposed a framework, based on the notion of rewriting modulo, to deal with graph transformation where parallel reductions may overlap some parts of the transformed graph. In general, these transformations do no lead to graphs but to a

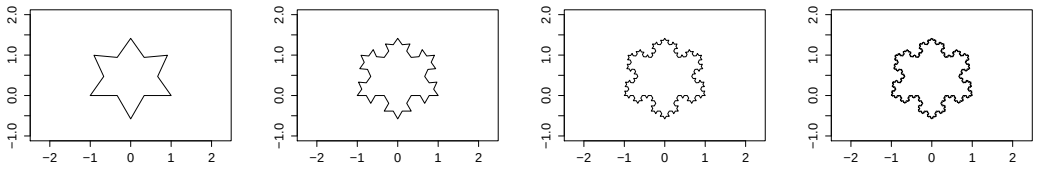


Figure 13: Flake results : flake at the different time steps 1,2,3 and 4

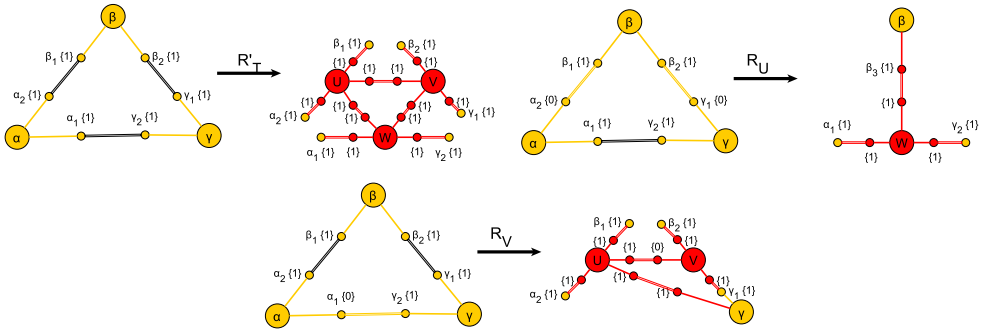


Figure 14: The rules R'_T, R_U, R_V are refinement rules defined e.g. in [1]

structure we call pregraphs. We proposed sufficient conditions which ensure that graphs are closed under parallel transformations. We also defined two parallel transformations: (i) one that fires all possible rules in parallel (full parallel) and (ii) a second rewrite relation which takes advantage of the possible symmetries that may occur in the rules by reducing the possible number of matches that one has to consider. The two proposed parallel rewrite relations are confluent (up to isomorphisms).

Our proposal subsumes some existing formalisms where simultaneous transformations are required such as cellular automata [18] or (extensions of) L-systems [14]. Indeed, one can easily write graph rewriting systems which define classical cellular automata, with possibly evolving structures (grids) and where the content of a cell, say C, may depend on cells not necessary adjacent to C. As for L-systems, they could be seen as formal (context sensitive) grammars which fire their productions in parallel over a string. Our approach here generalizes L-systems at least in two directions: first by considering graphs instead of strings and second by considering overlapping graph rewrite rules instead of context sensitive (or often context free) rewrite rules. Some graph transformation approaches could also be considered as extension of L-systems such as star-grammars [12] or hyperedge replacement [7]. These approaches do not consider overlapping matches but act as context free grammars. Other parallel graph transformations have been proposed in the literature where parallel transformations can be achieved on *independent* graphs gathered in multi-sets such as in [9] or by using a pullback approach [2]. However, in [5] parallel graph grammars with overlapping matches have been considered. In that work, overlapping subgraphs remain unchanged after reductions, contrary to our framework which does not require such restrictions. The idea behind parallel graph grammars has been lifted to general replacement systems in [17]. Amalgamation, see e.g.[10],

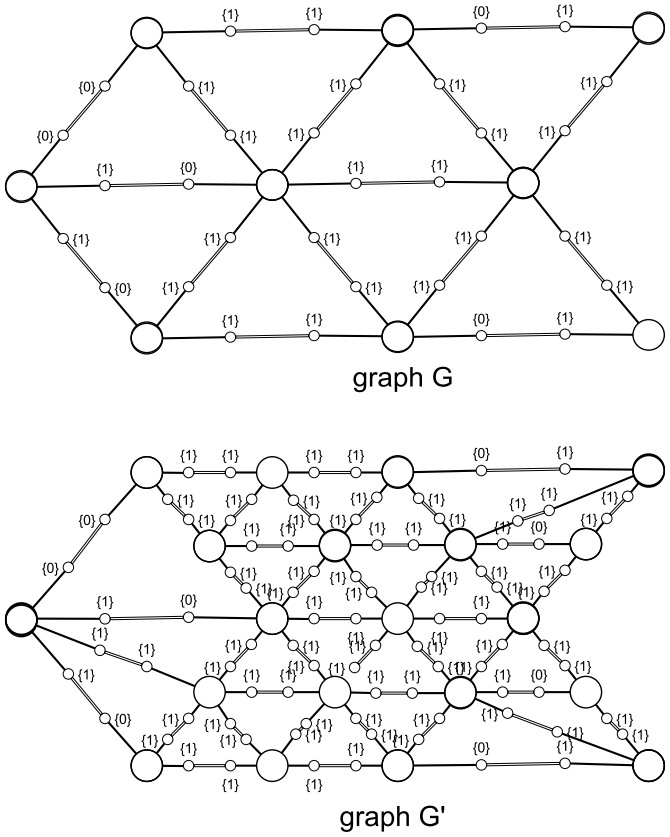


Figure 15: $G \rightrightarrows_{auto} G'$

aims at investigating how the parallel application of two rules can be decomposed into a common part followed by the remainder of the two considered parallel rules. Amalgamation does not consider full parallel rewriting as investigated in this paper. Another approach based on complex transformation has been introduced in [11]. This approach can handle overlapping matches but requires from the user to specify the transformation of these common parts. This requires to provide detailed rules. For instance, the two first cases of the triangle mesh refinement example requires about sixteen rules including local transformations and inclusions, instead of two rules in our framework.

The strength of our approach lies in using an equivalence relation on the resulting pregraph. This equivalence plays an important role in making graphs closed under rewriting. Other relations may also be candidate to equate pregraphs into graphs. we plan to investigate such kind of relations in order to widen the class of rewrite systems that may be applied in parallel on graph structures in presence of overlaps. We also plan to investigate other issues such as stochastic rewriting and conditional rewriting which would be a plus in modeling some natural phenomena. Analysis of the proposed systems remains to be investigated further.

Acknowledgment

We are grateful to the anonymous referees for their valuable comments. This work has been partially supported by the project CLIMT (ANR-11-BS02-016) and by the LabEx PERSYVAL-Lab (ANR-11-LABX-0025-01) funded by the French program Investissement d'avenir.

References

[1] R. E. Bank, A. H. Sherman, and A. Weiser. Refinement algorithms and data structures for regular local mesh refinement. In R. Stepleman et al., editor, *Scientific Computing*, pages 3–17. IMACS/North-Holland, 1983.

[2] Michel Bauderon. Parallel rewriting of graphs through the pullback approach. *Electr. Notes Theor. Comput. Sci.*, 2:19–26, 1995.

[3] Dominique Duval, Rachid Echahed, Frédéric Prost, and Leila Ribeiro. Transformation of attributed structures with cloning. In Stefania Gnesi and Arend Rensink, editors, *Fundamental Approaches to Software Engineering, FASE 2014*, volume 8411 of *LNCS*, pages 310–324. Springer, 2014.

[4] Rachid Echahed and Aude Maignan. Parallel graph rewriting with overlapping rules. *CoRR*, abs/1701.06790, 2017.

[5] Hartmut Ehrig and Hans-Jörg Kreowski. Parallel graph grammars. In A. Lindenmayer and G. Rozenberg, editors, *Automata, Languages, Development*, pages 425–447. Amsterdam: North Holland, 1976.

[6] Moore G.A. Automatic scanning and computer processes for the quantitative analysis of micrographs and equivalent subjects. *Pictorial Pattern Recognition*, 1969.

[7] Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer, 1992.

[8] H. C. M. Kleijn and Grzegorz Rozenberg. A study in parallel rewriting systems. *Information and Control*, 44(2):134–163, 1980.

[9] Hans-Jörg Kreowski and Sabine Kuske. Graph multiset transformation as a framework for massively parallel computation. In *4th International Conference on Graph Transformations, ICGT, Leicester, United Kingdom, September 7-13, 2008.*, volume 5214 of *Lecture Notes in Computer Science*, pages 351–365. Springer, 2008.

[10] Michael Löwe. Algebraic approach to single-pushout graph transformation. *Theor. Comput. Sci.*, 109(1&2):181–224, 1993.

[11] Luidnel Maignan and Antoine Spicher. Global graph transformations. In Detlef Plump, editor, *Proceedings of the 6th International Workshop on Graph Computation Models*, volume 1403, pages 34–49. CEUR-WS.org, 2015.

[12] Manfred Nagl. *Graph-Grammatiken: Theorie, Anwendungen, Implementierung*. Vieweg, 1979.

[13] Gerald E. Peterson and Mark E. Stickel. Complete sets of reductions for some equational theories. *J. ACM*, 28(2):233–264, 1981.

[14] Przemyslaw Prusinkiewicz and Aristid Lindenmayer. *The algorithmic beauty of plants*. Springer, 1996.

[15] Grzegorz Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[16] Donald Sannella and Andrzej Tarlecki. *Foundations of Algebraic Specification and Formal Software Development*. EATCS Monographs on theoretical computer science. Springer, 2012.

[17] Gabriele Taentzer. *Parallel and distributed graph transformation - formal description and application to communication-based systems*. Berichte aus der Informatik. Shaker, 1996.

[18] Stephen Wolfram. *A new kind of science*. Wolfram-Media, 2002.