Contextual graph grammars characterising Rational Graphs

Christophe Morvan

CONTEXTUAL GRAPH GRAMMARS CHARACTERISING RATIONAL GRAPHS

Christophe Morvan
Universit Paris-Est,
INRIA Rennes Bretagne Atlantique
Campus de Beaulieu, 35042 Rennes, France
Email: christophe.morvan@inria.fr

Abstract
Deterministic graph grammars generate a family of infinite graphs which characterise context-free (word) languages. The present paper introduces a context-sensitive extension of these grammars. We prove that this extension characterises rational graphs (whose traces are context-sensitive languages). We illustrate that this extension is not straightforward: the most obvious context-sensitive graph rewriting systems generate non recursive infinite graphs.

1. Introduction

In 1956, and then in 1959 Noam Chomsky wrote two articles which defined the Chomsky hierarchy. This hierarchy has had a tremendous impact on the development of the theory of formal languages. Since the early sixties, it has served for the classification, and evaluation of the expressive power of formal languages and formal models.

There is a deep connection between this hierarchy and graphs. It is obvious for type 3 languages (regular languages) which are characterised by finite automata. But from the mid-eighties on, there has been an increasing effort to characterise language families by graph families.

One of the first attempts to establish properties on structures characterising type 2 languages has been done by Muller and Schupp. They proved the decidability of the monadic second order theory of the graphs of pushdown automata [MS85]. In their work, the vertices of the graphs are the configurations of the pushdown automaton, and the arcs are transitions between configurations. Such a characterisation is called *internal*, it thoroughly depends on the choice of the machine, and it gives an explicit name to each vertex. Courcelle, in [Cou90], extended the decidability of monadic second order theory to graphs generated by deterministic graph grammars. Indeed these graph grammars were classical graph rewriting systems used to characterise families of graphs. In this paper they were used to define infinite graphs. Indeed these graphs are very close to graphs of pushdown automata ([CK01]) but it is an *external* characterisation: each grammar generates a set of isomorphic graphs. The name of the vertices are not explicitly stated along the generation of the graph. This is really important as it provides a higher level of abstraction. Another slight extension of these two families is formed by the

prefix-recognizable graphs from [Cau96]. In this paper Caucal provides both an external and internal characterisation of these graphs. He also proves that they have a decidable monadic second order theory, and that they characterise context-free languages.

For type 1 languages (context-sensitive languages), there are also several graph characterisations: transition graphs of linear bounded Turing machines [KP99], rational graphs [Mor00], automatic graphs [BG00, Ris03] or linear bounded graphs [CM06a]. All these works provide internal characterisations: the graphs are defined as sets of transitions between vertices which are words on some alphabet.

To our knowledge there is no external graph characterisation of context-sensitive languages. In particular there exists no context-sensitive extension of Courcelle's graph rewriting systems. [Sch97] is a survey presenting several contextual graph rewriting systems but they are used exclusively to characterise families of finite graphs, and no connection is established with the Chomsky hierarchy.

In this paper we propose an external characterisation of rational graphs in terms of contextual rewriting system. It is organised in two main parts: the first recalls the definition of regular and rational graphs and the second presents context-sensitive graph rewriting systems. We first examine a natural, unrestricted, contextual extension of graph grammars. We show that it is too general, as it produces non-recursive graphs. Then, we propose a restriction, which corresponds to rational graphs. Afterwards we examine possible relaxations of some constraints which yield families of non-recursive graphs. We conclude this paper applying our characterisation to reprove some facts on context-sensitive languages.

2. Deterministic graph grammars and rational graphs

In this section we define classical notations and recall fundamental definitions for deterministic graph grammars and rational graphs.

2.1. Mathematical preliminaries

Fundamentals

For any set E, its powerset is denoted by 2^E; if it is finite, its size is denoted by $|E|$. Let the set of non-negative integers be denoted by $\mathbb{N}$, and $\{1, 2, 3, \ldots, n\}$ be denoted by $[n]$. A monoid M is a set equipped with an associative operation (denoted $\cdot$) and a (unique) neutral element (denoted ε). A monoid M is *free* if there exist a subset A of M such that $M = A^* := \bigcup_{n\in\mathbb{N}} A^n$ and for each $u \in M$ there exists a unique finite sequence of elements of A, $(u(i))_{i\in[n]}$, such that $u = u(1)u(2)\cdots u(n)$; if A is finite, then is free, of *finite rank*. Elements of a free monoid of finite rank are called *words*. Let u be a word in M, $|u|$ denotes the length of u and $u(i)$ denotes its ith letter.

Hypergraphs

Let F be an *alphabet* (finite set) ranked by a mapping $\varrho : F \to \mathbb{N}$, this mapping associates to each element of F its *arity*. Given a ranked alphabet F, we denote by F_n the set of symbols of arity n. Now given V an arbitrary set, a *hypergraph* G is a subset of $\cup_{n\geqslant 1} F_n V^n$. The vertex set of such a hypergraph is the set $V_G = \{v \in V \mid FV^*vV^* \cap G \neq \emptyset\}$, in our setting, this set is either finite or countable. A hyperarc of arity n is denoted by $f\ v_1\ v_2\ \cdots\ v_n$. Hyper arcs of arity 2 are called *arcs*. Given a hypergraph G, an arc *ast* in G is identified with the labelled transition $s \xrightarrow[G]{a} t$ or simply $s \xrightarrow{a} t$ if G is understood. Furthermore, a is the *label*, s the *source* and t the *target*.

Elements of F_1 (labels of arity 1) are called *colours*.

Graphs

A (simple oriented labelled) *graph* G over V, labelled by P is a hypergraphs having only hyperarcs of arity 1 or 2 ($P = P_1 \cup P_2$). Given a graph G, the set of sources and targets are respectively denoted by $Dom(G)$ and $Im(G)$ (by convention, both sets contain also vertices belonging to arity 1 hyperarcs).

A graph G is *deterministic* if distinct arcs with the same source have distinct label: $r \xrightarrow{a} s \wedge r \xrightarrow{a} t \ \Rightarrow\ s = t$. Let G be a graph, a *path* in between vertices $u \in Dom(G)$ and $v \in Im(G)$, labelled w, corresponds to the existence of vertices $(u_i)_{i\in[n]}$ such that $u = u_1,\ v = u_n$ and $u_i \xrightarrow[G]{w(i)} u_{i+1}$ for $i \in [n-1]$. Such a path is denoted by $\overset{w}{\underset{G}{\Longrightarrow}}$ or simply $\overset{w}{\Longrightarrow}$ if G is understood.

A *graph morphism* g, between graphs G and G' is a mapping from $Dom(G) \cup Im(G)$ to $Dom(G') \cup Im(G')$ such that if there is an arc $u \xrightarrow[G]{a} v$, then there is an arc $g(u) \xrightarrow[G']{a} g(v)$. Such a morphism is an *isomorphism* if g is a bijection, and its inverse is also a morphism.

2.2. Deterministic graph grammars

A finite presentation is an effective tool in order to manipulate an infinite graph. Such presentations allow to conceive algorithms working on these graphs. Deterministic (hyperedge replacement) graph grammars are a classical example of finite (external) characterisations of infinite graphs. These grammars were initially defined to be an extension to graphs of word grammars (see, *e.g.,* [Roz97]). These graph grammars derived, from an axiom, an infinite family of *finite* graphs. Courcelle in [Cou90] considered the deterministic form of these grammars, each grammar, seen as a set of equations, admits a single (up to isomorphism) infinite graph as least solution. In [Cau07], Caucal made an extensive survey on deterministic graphs grammars. In particular, he gave a uniform presentation of several results: he developed a framework of simple techniques, making intensive use of colours and colouring, to provide new proofs of these results.

Definition 2.1 (Hypergraph grammar)**.** A *hypergraph grammar* (HR-grammar for short) G, is a 4-tuple (N, T, R, H_0), where N and T are two ranked alphabets of respectively *non-terminals* and *terminals* symbols; H_0 is the axiom, a finite graph formed by hyperarcs labelled by $N \cup T$, and R is a set of rules of the form $f\ x_1\ \cdots\ x_{\varrho(f)} \to H$ where $f\ x_1\ \cdots\ x_{\varrho(f)}$ is an hyperarc joining disjoint vertices and H is a finite hypergraph.

Observe that usually, the vertices appearing in the left-hand side of a rule appear in the right-hand side (though the definition does not command it).

Remark 2.2. In this paper, we consider graphs, therefore, the terminal symbols will have either rank one, or two. Such a graph is seen as a simple subset of $T_2VV \cup T_1V$.

A grammar is *deterministic* if there is a single rewriting rule per non-terminal:

$$(X_1, H_1), (X_2, H_2) \in R \wedge X_1(1) = X_2(1) \Rightarrow (X_1, H_1) = (X_2, H_2)$$

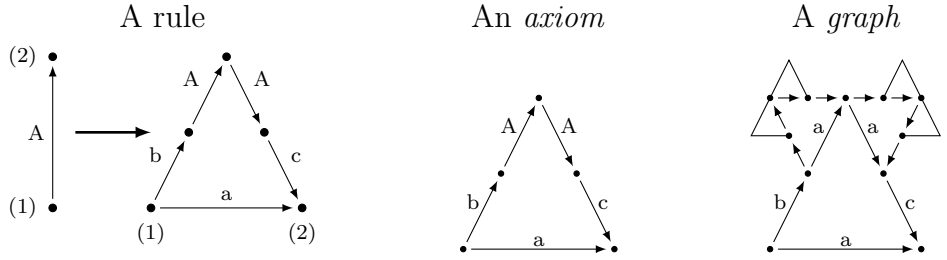
Now, given a set of rules R, the *rewriting* $\underset{R}{\to}$ is the binary relation between hypergraphs defined as follows: M rewrites into N, written $M \underset{R}{\to} N$ if there is a non-terminal hyperarc $X = Av_1v_2 \dots v_p$ in M and a rule $Ax_1x_2 \dots x_p \to H$ in R such that N is obtained by replacing X by H in M: $N = (M - X) \cup h(H)$ for some injection h, mapping v_i to x_i for each i, and every other vertices of H to vertices outside of M. This rewriting is denoted by $M \underset{R,X}{\longrightarrow} N$. Now, this rewriting obviously extends to sets of non-terminal, for E such a set, this rewriting is denoted: $M \underset{R,E}{\longrightarrow} N$. The *complete parallel rewriting* $\underset{R}{\Rightarrow}$ is the rewriting relative to the set of *all* non-terminal hyperarcs of R.

Now given a deterministic graph grammar $G = (N, T, R, H_0)$, and a hypergraph H, we denote by $[H] := H \cap (T\ V_H\ V_H \cup T\ V_H)$ the set of terminal arcs, and colours of H. We define G^ω, the limit set of G, as the set of all countable union of $[H_n]$, derived from H_0 (there are infinitely many H_n for each n):

$$G^\omega = \left\{ \cup_{n \geqslant 0} [H_n] \mid \forall n \geqslant 0, H_n \underset{R}{\Rightarrow} H_{n+1} \right\}$$

Formally, G^ω is a set of graphs, and we say that a graph H is *generated* by G if it belongs to G^ω. But all these graphs are isomorphic, thus, we will often express that G^ω is *the* graph generated by G.

Example 2.3. This is a simple example of deterministic graph grammar and a representation of the resulting graph.



Graph grammars characterise *regular graphs*. This characterisation is very efficient to extend techniques working for finite graphs to these infinite graphs (for example computing the connected components of a regular graph is simply doing it inductively on the right-hand sides of the rules). Furthermore these graphs (restricted to finite degree) correspond to transition graphs of pushdown automata [CK01]. Yet, algorithms defined on pushdown automata often make technical assumptions on the form of the automaton: for example that each *state* reflects that the *configuration* belongs to a certain regular set. In most cases these assumptions only affect the internals of the automaton, it does not affect the structure of its configuration graph. More precisely, these transformations do not affect globally the family of graphs but some individual graph may be transformed switching representation. This is not the case for graph grammars: normal forms preserve the generated graph. Thus graph grammars are more suited when focussing on structural properties.

In the next subsection we will define rational graphs with an internal characterisation, later on, in Section 3 we will extend graph grammars in order to characterise rational graphs. The way we enrich these grammars is similar to what is done for word grammars: contextual rewriting is necessary. However, in order to avoid being too general, we will have to separate context arcs from non-terminal ones, and the axiom will be a regular graph (defined by a HR-grammar).

2.3. Rational graphs

In this subsection we present key elements on rational graphs. More details can be found in [Mor00, MS01].

The family of *rational* subsets of a monoid $(M, \cdot)$ is the least family containing the finite subsets of M and closed under union, concatenation and iteration.

A *transducer* is a finite automaton labelled by pairs of words over a finite alphabet X, see for example [Ber79]. A transducer accepts a relation in $X^* \times X^*$; these relations are called rational relations as they are rational subsets of the product monoid $(X^* \times X^*, \cdot)$.

Now, let us consider the graphs of $\Sigma \times X^* \times X^*$ (here, in order to separate explicitly first and second vertices, we use Cartesian product instead of concatenation). Rational graphs are extensions of rational relations, they are defined by *labelled transducers*.

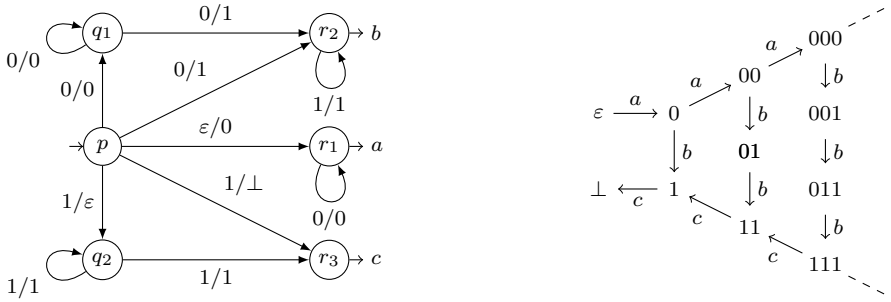
Definition 2.4. A *labelled transducer* $T = (Q, I, F, E, L)$ over X and Σ, is composed of a finite

set of states Q, a set of initial states $I \subseteq Q$, a set of final states $F \subseteq Q$, a finite set of transitions (or edges) $E \subseteq Q \times X^* \times X^* \times Q$ and a mapping L from F into 2^Σ.

An arc $u \xrightarrow{a} v$ is *accepted* by a labelled transducer T if there is a path from a state in I to a state f in F labelled by (u, v) and such that $a \in L(f)$.

Definition 2.5. The set of rational graphs in $\Sigma \times X^* \times X^*$, denoted by $Rat(\Sigma \times X^* \times X^*)$, is formed by the graphs accepted by labelled transducers.

Example 2.6. The graph depicted below, on the right-hand side, is generated by the labelled transducer on the left-hand side.



The path $p \xrightarrow{0/0} q_1 \xrightarrow{0/1} r_2 \xrightarrow{1/1} r_2$ accepts the pair $(001, 011)$, the final state r_2 is labelled by b thus there is an arc $001 \xrightarrow{b} 011$ in the graph.

Rational graphs have been introduced in order to define a general family of infinite graphs generalising previous families. They have few decidable properties, but they characterise context-sensitive languages (CSL) [MS01]. The rational trees (rooted connected acyclic *rational graphs* such that each vertex has at most one predecessor) have a decidable first order theory [CM06b].

Employing transducers to define a family of graphs is an internal characterisation. A side-effect of such a definition is that the set of vertices of a rational graph is a regular set of words. This implies that some graphs are not rational simply because of their vertex set, but it is not related to their structure. In contrast, for finite graphs, every algorithm, every characterisation, is given *up to renaming of the vertices*. Such external characterisation is difficult to obtain for infinite graphs. Still, characterisations like HR-grammars overcome this problem. The next section defines contextual graph grammars characterising rational graphs.

3. Contextual graph grammars

There are several contextual graph rewriting systems (see Section 4 in [Sch97]). But like for HR-grammars, these systems have not been used to characterise families of infinite graphs. Especially in the sense of deterministic graph grammars. We will see that it is difficult to define such systems generating exclusively recursive sets of arcs.

3.1. Contextual graph rewriting systems

Recursivity is understood differently for a set of finite graphs, and a single infinite graph: in the first case, it corresponds to checking that a given graph belongs to the family, whereas for systems that produce a single graph, recursivity corresponds to deciding, from the grammar, the existence of an arc between two given vertices. In the case of sets of graphs, as soon as the grammar does not remove *terminal arcs*, it is recursive. In the case of a single graph, it is not sufficient: the rules may use non-terminal arcs to perform complex computations, which may eventually produce an arc depending on the result of this computation, in such a situation, the graph itself may be non-recursive.

In this first subsection we present a naive contextual rewriting system which generates non-recursive graphs.

Let N_R be a finite ranked set of *non-terminals*, and T_R a finite ranked set of *terminals*.

We propose here a natural definition of contextual graph rewriting system.

Definition 3.1 (Contextual graph rewriting system)**.** A *contextual graph rewriting system* S, is a set of rules of the form $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$ where $f\ x_1\ \cdots\ x_{\varrho(f)}$ is a non-terminal hyperarc, H_c is a finite *context* graph, and H is a finite hypergraph, that can share some vertices with H_c and $\{x_1,\ \cdots,\ x_{\varrho(f)}\}$. Furthermore, H_c is composed only of terminal hyperarcs, and $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ forms a connected hypergraph.

The sketch of proof for Proposition 3.2 provides an example of contextual rewriting rules (rules r_0, r_1, r_Δ are explicitely depicted).

Now, given a rewriting rule $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)} \to H_c \cup H$, a rewriting step in a graph G consists in finding the non-terminal f in G, such that there is a morphism h of $H_c \cup f\ x_1\ \cdots\ x_{\varrho(f)}$ into G then removing f from G, and adding H to G according to the rule, and to the morphism h. Given a contextual graph rewriting system S, and a finite graph H, we define $S^\omega(H)$ in the same way as G^ω for a HR-grammar G. Recall that this graph is a restriction to terminal symbols, in particular in our setting it only has hyperarcs of arity 1 or 2.

We say that such a contextual rewriting system is *deterministic*, if there is, at most, one rule for each non-terminal. This restriction only *limits* non-determinism, as there might be some situations where the context of a non-terminal can be found more than once. In such situation, it means that the rewriting system may generate at least two non-isomorphic graphs.

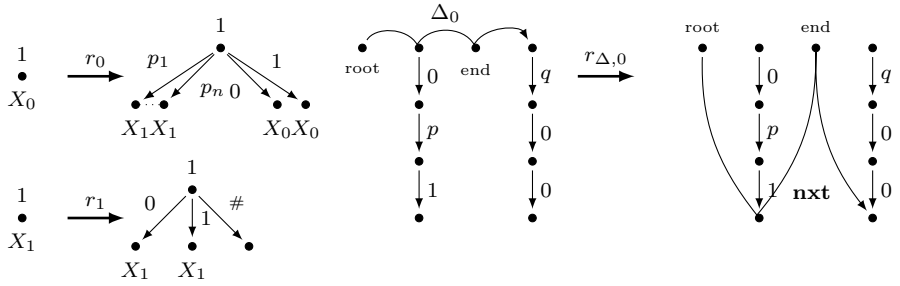
Unfortunately this natural generalization of HR-grammars is much too general (even restricted to systems where there is always a single morphism to match the context): the graph rewriting systems generate non recursive graphs.

The first observation toward this result is the following: the transition graph of a Turing Machine is recursive: given two configurations of the machine it is trivial to check that they

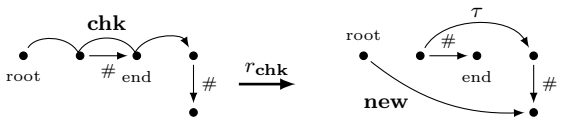
are connected in the transition graph. The second observation is that *the accessible restriction* (restriction to reachable configurations) of this graph is non-recursive; if it were recursive it would imply the decidability of accessibility for Turing machines.

Proposition 3.2. *Let M be a Turing machine; u_0, and u_1 two configurations of M, there exists effectively a contextual graph rewriting system S which generates, from a finite axiom A, a set of graphs isomorphic to the accessible component of the transition graph of M, from its initial configuration. Furthermore each vertex in bijection with u_0 and u_1 is marked by a colour c_0 and c_1 respectively (each graph in S^ω contains a single pair of such vertices).*

Proof (sketch). Let M be a Turing machine, with tape alphabet $\Sigma = \{0, 1\}$,and states in the finite set Q. For simplification we assume the tape is filled initially by zeros, and that M operates on such a tape, we also assume that the tape is only infinite on the right, finally a configuration ends with a 1 or a state symbol. A transition of M is a tuple (p, A, q, B, δ) where $p, q \in Q$ are respectively the current and next state of M, $A \in \Sigma$ is the tape letter immediately on the right of the reading head, $\delta \in \{-1, +1\}$ represent the movement of the head to the left or right, and B is the letter replacing A after the move. Let $\Delta = (p, q, 1, 0, -1)$ be a transition of M.



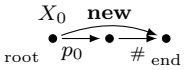
Rules r_0 and r_1 generate a complete tree corresponding to configurations of M, each path leading from the root to an arc labelled # corresponds to a configuration of the machine. For each transition Δ of M, there is one or two rules[2]similar to $r_{\Delta,0}$ simulating the rewriting step from a configuration to the next one. The context of such a rule is formed by the arcs $0, p, 1$ and $q, 0, 1$ the rewriting consist simply in removing the non-terminal Δ_0 and producing **nxt**. Before applying these rules the transition is simply moving along a single path (recalling the root and the end). The rule corresponding to **nxt** performs the same path along the second and forth vertices. The rule that ensures termination and thus the restriction to a single connected component is the following:



The left hand-side of this rule witnesses that the computation has reached the end of the original configuration, the non-terminal **new** records the position of the root and the position of the extremity of the *new* configuration, the right hand-side from **new** starts over a new

[2]There are two rules when $\delta = -1$ depending on the symbol preceding the reading head.

computation. There is a τ-labelled arc between the two vertices corresponding to the respective configurations. Ultimately the graph is formed simply by τ labelled arcs. The axiom is this finite graph:



Colours c_0 and c_1 are simply added by a single rule having for context a tree corresponding to both configurations. If it were decidable whether there is an τ-arc between c_0 and c_1, it would enable to decide the existence of an arc in the (non-recursive) accessible component reachable in the transition graph of M from the initial configuration. □

Proposition 3.2 may be reformulated into following corollary.

Corollary 3.3. *Deterministic contextual graph rewriting systems generate non recursive graphs.*

Obviously from a Turing machine, M, and two configurations, the existence of an arc between these configurations in the accessible component of the transition graph of M might be checked if it could be checked in the graph generated by the system derived from Proposition 3.2.

3.2. Contextual hyper-edge-replacement graph grammars

We present, now, a more restrictive contextual rewriting system which will be used to characterise rational graphs.

Definition 3.4. A *contextual hyper-edge-replacement hypergraph grammar* (CHR-grammar for short) is a tuple (C, N, T, R_c, H_0), where C, N and T are finite ranked alphabets of respectively contextual, non-terminal and terminal symbols; R_c is a *deterministic contextual graph rewriting system* where, for each rule $H_c \cup fx_1 \dots x_{\varrho(f)} \to H_c \cup H$, the graph H_c is formed only by arcs labelled in C, and H by arcs labelled in $T \cup N$, and H_0 is the axiom: a *deterministic* regular graph formed by arcs with labels in C, and a single non-terminal hyperarc.

Given a CHR-grammar $G = (C, N, T, R_c, H_0)$, the set G^ω is defined similarly as for HR-grammar as $R_c^\omega(H_0)$.

Observe that, for each rule in R_c, the context graph (H_c) is not modified: no arc is removed, and no arc is added, thus successive rewritings preserve the axiom H_0. Furthermore, this definition imposes that the axiom is a deterministic regular graph. This ensures that for each rule $r \in R_c$ (with non-terminal A) and each occurrence of A in the graph, there is at most a single morphism which maps the context of the left-hand side of r to the neighbourhood of A. This restriction may be checked, since verifying that a given graph grammar generates a deterministic graph is decidable (direct consequence of Proposition 3.13 in [Cau07]).

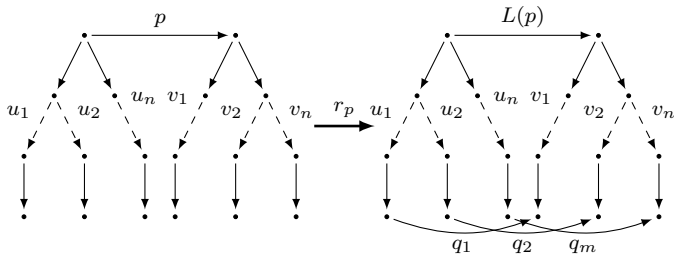
Later in this paper we will discuss other structural restrictions (for the axiom), and see that it is difficult to allow graphs that are not trees. We will also have to impose stronger restrictions on the rules R_c in order achieve a characterisation.

We first show that an $|X|$-ary tree as axiom is sufficient to obtain each graph of $Rat(\Sigma \times X^* \times X^*)$.

Proposition 3.5. *Each rational graph is generated by a* CHR*-grammar.*

Like for Proposition 3.2, the proof is in the full paper. We sketch this straightforward construction.

Proof (sketch). Let G be a rational graph in $\Sigma \times X^* \times X^*$ (and T a transducer representing it), let H_0 be the complete $|X|$-ary tree labelled on X (with a non-terminal p_0 on the root). For each state p of T, we have the following rule r_p.



Here, we suppose that there are transitions $p \xrightarrow{u_i/v_i} q_i$ for some states $(q_i)_{i\in[m]}$, and also $L(p)$ represent all labels produced at state p (if p is a terminal state). Now each pair of path in H_0 correspond to a path of pairs in T. Thus both graphs are the same. □

3.3. Tree-separated contextual grammars generate rational graphs

This subsection identifies restrictions on CHR-grammar which enable a converse inclusion for Proposition 3.5.

A CHR-grammar (C, N, T, R_c, H_0) is called a *tree-*CHR*-grammar* if the axiom H_0 is a tree, and the left-hand side of each rule of R_c is formed by trees *rooted* in the vertices of the non-terminal (some vertices of this non-terminal may be non-root vertices of theses trees). Furthermore, if each of these trees possesses a single vertex belonging to the non-terminal (its root) this grammar is called a *tree-separated-*CHR*-grammar*. Graphs generated by the latter are captured by rational graphs:

Proposition 3.6. *Any graph generated from a tree-separated-*CHR*-grammar, is isomorphic to a graph in* $Rat(\Sigma \times X^* \times X^*)$.

There are several difficulties to establish this result: the axiom is not a complete tree, so we need to ensure that an arc is never set on a vertex that does not exist. Hyperarcs of arity greater than 2 need to be taken into account. Finally, there may be twists: in the right-hand side of a rule, the path from the goal of the original non-terminal may lead to the source of some other non-terminal.

In order to prove Proposition 3.6 we first state two technical lemmas (whose proofs are postponed to the appendix).

Lemma 3.7. *The set of path labels from the root of a regular tree leading to some colour (arc of arity* 1*) is a regular set of words.*

Lemma 3.8. *Any tree-separated-*CHR*-grammar G can be effectively transformed into a tree-separated-*CHR*-grammar G' generating the same set of graphs and such that, for each rule r of G', each non-terminal appearing in the right-hand side of r has at most one vertex in each tree of the context graph.*

We are now able to sketch the proof of Proposition 3.6.

Proof (sketch). We first suppose the axiom is some complete n-ary tree. Lemma 3.8 enables that each non-terminal has at most a single vertex in each subtree of the right-hand side of each rule.

Then the states of the transducer are defined by ordered pairs of vertices of the non-terminals, and the transitions from the paths in the right-hand sides. Terminal states are defined by pairs of vertices of terminal arcs (source, target).

Second, if the regular tree is not a complete tree, we use Lemma 3.7 to compute a regular set of paths reaching leaves: P. Then, it is not sufficient to make an intersection with the set $P \times P$, since some rewriting may not occur from the absence of some vertex *along* the rewritings. Thus, we synchronize the transducer with the set P: each state $A_{(u_1,u_2)}$ checks that the complete context for the rule of A is available. □

Combining Proposition 3.6 and 3.5 we state the main result of this paper (observe that the grammar which proves Proposition 3.5 is tree-separated-CHR-grammar).

Theorem 3.9. *The family of rational graphs, and the graphs generated by tree-separated-*CHR*-grammars coincide.*

Now, slightly modifying the rules presented in the proof Proposition 3.2 enables to extend it. The rules r_0 and r_1 generate the axiom, and the only rule that does not satisfy tree-separated-CHR-grammar restrictions is $r_{\mathbf{chk}}$. From this observation we obtain the following result.

Proposition 3.10. *Tree-*CHR*-grammars generate non-recursive graphs.*

This result proves that in some sense tree-separated-CHR-grammars are the most general context-sensitive graph rewriting system capturing rational graphs and nothing more.

4. Applications and conclusion

Applications

We first reformulate Theorem 4.12 of [MS01] in the context of tree-separated-CHR-graphs.

Theorem 4.1. [**MS01**] *The set of paths between two colours in tree-separated-*CHR*-graphs coincide with context-sensitive languages.*

A major interest of having a graph characterisation of languages is that it enables simple proofs for language properties. In this subsection we provide a direct applications of tree-separated-CHR-grammars to reprove simple results for CSL.

Proposition 4.2. *Let G_1 and G_2 be two tree-separated-*CHR*-graphs: the iteration of G_1, the concatenation and synchronisation product of G_1 and G_2 are tree-separated-*CHR*-graphs.*

These results are proved constructing a compound axiom and compound rules. Using Theorem 4.1, Proposition 4.2 proves the following results:

Corollary 4.3. *The* CSL *are closed under concatenation and Kleene star; the intersection of two* CSL *is a* CSL.

One of the most stunning result for CSL is closure under complementation [Sze88, Imm88]. It would be interesting to reprove this result using a graph approach. Unfortunately, proving closure under complementation with graphs is related to closure under determinisation. But here, languages recognised by deterministic tree-separated-CHR-graphs are deterministic CSL. If they contained all CSL it would prove Kurdoda's conjecture [Kur64] on the equivalence between deterministic and non-deterministic CSL. Our conjecture is that deterministic rational graphs do not recognise all CSL. Thus, using this graph characterisation to prove closure under complement would be similar to proving the closure under complement of Bchi automata: it would not use the graph structure.

Conclusion

In this paper we have presented a family of infinite graphs generated by contextual graph grammars. This family characterises rational graphs. We also have examined several variants which generate non-recursive graphs.

Elaborating on techniques from HR-grammar we hope to provide a toolbox for manipulating these graphs and then being able to improve the knowledge of CSL.

An interesting question is the characterisation of the graphs of higher order pushdown automata in terms of graph rewriting systems. Such a characterisation could enable a deeper understanding of the structure of these objects.

References

[Ber79] J. Berstel. *Transductions and context-free languages*. Teubner, 1979.

[BG00] A. Blumensath and E. Grädel. Automatic Structures. In *15th IEEE Symposium on Logic in Computer Science LICS 2000*, pages 51–62, 2000.

[Cau96] D. Caucal. On transition graphs having a decidable monadic theory. In Icalp 96, volume 1099 of *LNCS*, pages 194–205, 1996.

[Cau07] D. Caucal. Deterministic graph grammars. In *Texts in logics and games 2*, pages 169–250. Amsterdam University Press, 2007.

[CK01] D. Caucal and T. Knapik. An internal presentation of regular graphs by prefix-recognizable ones. *Theory of Computing Systems*, 34(4), 2001.

[CM06a] A. Carayol and A. Meyer. Context-sensitive languages, rational graphs and determinism. *Logical Methods in Computer Science*, 2(2), 2006.

[CM06b] A. Carayol and C. Morvan. On rational trees. In Zoltán sik, editor, CSL 06, volume 4207 of *LNCS*, pages 225–239, 2006.

[Cou90] B. Courcelle. *Handbook of Theoretical Computer Science*, chapter Graph rewriting: an algebraic and logic approach. Elsevier, 1990.

[Imm88] N. Immerman. Nondeterministic space is closed under complementation. *SIAM Journal on computing*, 17(5):935–938, October 1988.

[KP99] T. Knapik and E. Payet. Synchronization product of linear bounded machines. In *FCT*, volume 1684 of *LNCS*, pages 362–373, 1999.

[Kur64] S.-Y. Kuroda. Classes of languages and linear-bounded automata. *Information and Control*, 7(2):207–223, June 1964.

[Mor00] C. Morvan. On rational graphs. In J. Tiuryn, editor, Fossacs 00, volume 1784 of *LNCS*, pages 252–266, 2000.

[MS85] D. Muller and P. Schupp. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.

[MS01] C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In *MFCS*, volume 2136 of *LNCS*, pages 548–559, 2001.

[Ris03] C. Rispal. The synchronized graphs trace the context-sensitive languages. In *ENTCS*, volume 68, pages 55–70, 2003.

[Roz97] G. Rozenberg, editor. *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*. World Scientific, 1997.

[Sch97] A. Schürr. Programmed graph replacement systems. In Rozenberg [Roz97], pages 479–546.

[Sze88] R. Szelepcsényi. The method of forced enumeration for nondeterministic automata. *Acta Informatica*, 26(3):279–284, November 1988.