A Chomsky-Like Hierarchy of Infinite Graphs

Didier Caucal[1] and Teodor Knapik[2]

[1] IRISA–CNRS, Campus de Beaulieu,
35042 Rennes, France,
caucal@irisa.fr
[2] ERMIT, Université de la Réunion,
BP 7151, 97715 Saint Denis Messageries Cedex 9, Réunion,
knapik@univ-reunion.fr

Abstract. We consider a strict four–level hierarchy of graphs closely related to the Chomsky hierarchy of formal languages. We provide a uniform presentation of the four families by means of string rewriting.

1 Introduction

Many devices known in computation theory such as automata, machines, transducers or rewrite systems may be used in several ways for encoding infinite graphs. A graph may be associated to a device $\mathcal{D}$ as a transition system that represents its behaviour. In this case we are not interested in any acceptance condition because the device is not used as a recognizer. We simply assimilate it to a reactive system which reads a letter and changes nondeterministically its internal configuration one or more times before reading, in an infinite input stream, the next letter of an *external alphabet* Γ. We call *transition graph* a graph constructed in this way. Yet we have a choice between a graph $\mathfrak{tg}_\Gamma(\mathcal{D})$ that displays all transitions of $\mathcal{D}$, including silent transitions (labelled by τ) or a graph $\mathfrak{tG}_\Gamma(\mathcal{D})$ that reflects only its observable behaviour. In both cases, we may wish the vertices of a graph do not consist of all configurations of $\mathcal{D}$ but belong to a given set C. Indeed, the graphs studied here are *concrete* viz its vertices are named and the words over an *internal alphabet* Σ are used as names. Thus C is a language over Σ which, for decidability related reasons, is assumed to be rational. Transition graphs have been introduced in [15] for pushdown automata, in [11] for linear–bounded automata and in [16, 8] for Turing machines.

Besides transition graphs, infinite graphs may be defined using those machines that are able to accept or generate relations, merely rational transducers and Turing machines. We call these *relation graphs*. In order to obtain edge–labelled graphs, we let each final state of the device correspond to a labelling letter of the external alphabet Γ. For instance, if a single–tape nondeterministic Turing machine $\mathcal{M}$ starts with a word u and stops with a word v in a final state f_a, then the edge $u \xrightarrow{a} v$ belongs to $\mathfrak{rg}_\Gamma(\mathcal{M})$. In a similar way, labelled string–rewriting systems may define relation graphs. Such graphs have an edge $u \xrightarrow{a} v$, whenever a word u may be rewritten into a word v by a rewrite rule which has a label a. Relation graphs have been introduced in [9] for synchronized finite

K. Diks et al. (Eds): MFSC 2002, LNCS 2420, pp. 177–187, 2002.

transducers, in [13] for finite transducers, in [8] for Turing machines and in [5] for labelled string–rewriting systems.

In this paper, we show that several classes of relation and transition graphs have a uniform finite presentation by means of string rewriting. The uniform presentation of this paper has been introduced in [10, 4] and, similarly to the Cayley graph of a group, is based on the following idea: the graph associated to a string–rewriting system $\mathcal{S}$ has an edge $u \xrightarrow{a} v$ whenever au may be rewritten by $\mathcal{S}$ into v, where u and v are irreducible words. However, the vertices of the graph do not consist of all irreducible words but are restricted to a given rational subset of the latter. A slightly different idea introduced in [7] consists in restricting not only the vertices of the graph but also all intermediate rewrite steps to a given rational set. We speak of Cayley–type graphs in the former case and normal–form graphs in the latter case. We show that both provide a uniform presentation of a Chomsky–like hierarchy of infinite graphs.

We are only interested in those graphs which have a rational set of vertices. Since the edges of our graphs are induced by the rewriting relation, we argue that the rationality of this relation is an essential separation criterion for families of graphs presented by string rewriting. This rationality depends on possible overlaps of right–hand sides by left–hand sides of rewrite rules. As shown in [7], only two types of overlaps, namely prefix and right (together with their symmetric versions: suffix and left) lead to rational rewriting relations. We establish that prefix–recognizable (resp. rational) graphs (see survey [18]) correspond to prefix (resp. right) systems. With unrestricted overlaps we get the family of recursively enumerable graphs. We also capture the family of finite graphs located at the bottom of our four–level hierarchy of graphs. The latter is closely related to the Chomsky hierarchy of formal languages via the notion of trace of a graph.

Due to the limitation of the number of pages, proofs are not given in this version but the reader may consult a complete version at

`http://www.univ-reunion.fr/~knapik/publications/mfcs02.ps.gz`

2 Preliminaries

The powerset of a set E is written $\wp(E)$. The set $\{1, \ldots, n\}$ is abbreviated as $[n]$ with $[0] = \varnothing$. We write $\mathcal{D}\mathrm{om}(\mathcal{R})$ (resp. $\mathcal{R}\mathrm{an}(\mathcal{R})$) for the domain (resp. range) of a relation $\mathcal{R}$.

We assume that the reader is familiar with the notions of monoid, word, rational and recognizable subsets of a monoid, regular expression, finite automaton and transducer (see e.g. [3]) as well as pushdown automaton (see e.g. [1]) and Turing machine (see e.g. [12]).

The family of finite (resp. recognizable, rational) subsets of a monoid $\mathcal{M}$ is written $\mathcal{F}\mathrm{in}(\mathcal{M})$ (resp. $\mathcal{R}\mathrm{ec}(\mathcal{M})$, $\mathcal{R}\mathrm{at}(\mathcal{M})$). The free monoid over Σ is written Σ^*, ε stands for the empty word and the length of a word $w \in \Sigma^*$ is denoted by $|w|$.

A *string–rewriting system* (an srs for short) $\mathcal{S}$ is a subset of $\Sigma^* \times \Sigma^*$. An element of $\mathcal{S}$ is called a *rule* and is written $l \to r$; the word l (resp. r) is its

left-hand (resp. right-hand) side. In this paper we consider only finite and recognizable srs. (Recall that a recognizable srs is a finite union of relations of the form $L \times R$, where $L, R \in \mathcal{R}\mathrm{at}(\Sigma^*)$). The *single–step reduction relation* induced by $\mathcal{S}$ on Σ^*, is the binary relation

$$\xrightarrow[\mathcal{S}]{} = \{(xly, xry) \mid x, y \in \Sigma^*,\ l \to r \in \mathcal{S}\} \ .$$

An $\mathcal{S}$*–redex* of a reduction step $u \xrightarrow[\mathcal{S}]{} v$ is a factor l of u such that $l \to r \in \mathcal{S}$, $u = xly$ and $v = xry$ for some $x, y \in \Sigma^*$. A word u *reduces* into a word v (alternatively v is a *descendant* of u), if $u \xrightarrow[\mathcal{S}]{*} v$, where $\xrightarrow[\mathcal{S}]{*}$ is the reflexive–transitive closure of $\xrightarrow[\mathcal{S}]{}$ called the *reduction relation* induced by $\mathcal{S}$ on Σ^*. A word v is *irreducible* w.r.t. $\mathcal{S}$ when v does not belong to $\mathcal{D}\mathrm{om}(\xrightarrow[\mathcal{S}]{})$. The set $\mathcal{I}\mathrm{rr}(\mathcal{S}) := \Sigma^* \setminus \mathcal{D}\mathrm{om}(\xrightarrow[\mathcal{S}]{})$ of all irreducible words w.r.t. $\mathcal{S}$ is rational whenever $\mathcal{D}\mathrm{om}(\mathcal{S})$ is so, since $\mathcal{D}\mathrm{om}(\xrightarrow[\mathcal{S}]{}) = \Sigma^*\big(\mathcal{D}\mathrm{om}(\mathcal{S})\big)\Sigma^*$.

When the reduction w.r.t. $\mathcal{S} \subseteq \Sigma^* \times \Sigma^*$ is restricted to some specific set of configurations $C \subseteq \Sigma^*$ we speak of *configured rewriting* [7]. Only rational sets of configurations are of interest in this paper. Thus, a *configured system* is a pair $(\mathcal{S}, C)$ where $\mathcal{S} \subseteq \Sigma^* \times \Sigma^*$ and $C \in \mathcal{R}\mathrm{at}(\Sigma^*)$. We define $\xrightarrow[(\mathcal{S},C)]{} := \xrightarrow[\mathcal{S}]{} \cap\, C \times C$ and the rewriting relation w.r.t. $(\mathcal{S}, C)$ is the reflexive–transitive closure $\xrightarrow[(\mathcal{S},C)]{*}$ of $\xrightarrow[(\mathcal{S},C)]{}$. We say that $(\mathcal{S}, C)$ is *stable* if for all $u, w \in C$, $v \in \Sigma^*$ such that $u \xrightarrow[\mathcal{S}]{*} v \xrightarrow[\mathcal{S}]{*} w$ we have $v \in C$. We have then the following equality $\xrightarrow[(\mathcal{S},C)]{*} = \xrightarrow[\mathcal{S}]{*} \cap\, C \times C$. Note that the stability is undecidable in general [7] but the following sufficient condition $\xrightarrow[\mathcal{S}]{}(C) \subseteq C$, where $\xrightarrow[\mathcal{S}]{}(C)$ stands for the image of C under $\xrightarrow[\mathcal{S}]{}$, is decidable whenever $\mathcal{S}$ is finite, recognizable or rational. A configured system $(\mathcal{S}, C)$ is finite (resp. recongnizable) when $\mathcal{S}$ is so. We denote by $\mathcal{S}\mathrm{tabCS}$ (resp. $\mathcal{F}\mathrm{inCS}$, $\mathcal{R}\mathrm{ecCS}$) the class of stable (resp. finite, recognizable) configured systems.

An essential feature of an srs is the way that left–hand sides overlap right–hand sides. An *LR–overlap of* $\mathcal{S}$ (resp. of $(\mathcal{S}, C)$) is a tuple $(x_1, l_1 \to r_1, y_1, x_2, l_2 \to r_2, y_2)$, where $x_1, y_1, x_2, y_2 \in \Sigma^*$ and $l_1 \to r_1, l_2 \to r_2 \in \mathcal{S}$ are such that $x_1 r_1 y_1 = x_2 l_2 y_2$ (resp. and $x_1 l_1 y_1, x_1 r_1 y_1, x_2 l_2 y_2, x_2 r_2 y_2 \in C$) and at least one of the following holds: $|x_1| \leq |x_2| < |x_1 r_1|$, $|x_2| < |x_1| \leq |x_2 l_2|$. The former inequality corresponds to the following cases:

and the latter corresponds to the following cases:

Seven kinds of LR–overlaps may be distinguished, including the special case when both$x_1 = x_2$ and $r_1 = l_2$ (see [7] for details).

Graphs

By *graph* we always understand here a countable oriented edge–labelled simple graph, viz., a subset of $D \times \Gamma \times D$ where D is a countable set and Γ is a finite alphabet. The family of all such graphs with labels in Γ is written $\mathrm{G}(\Gamma)$ (independently of D). Given a graph $G \subseteq D \times \Gamma \times D$ where D, we write $d \xrightarrow[G]{a} d'$ for an a–labelled edge of G from a vertex $d \in D$ to a vertex $d' \in D$. A *path* of G from d to d' is a sequence of edges $(d_{i-1} \xrightarrow[G]{a_i} d_i)_{i \in [n]}$ such that $d_0 = d$ and $d_n = d'$. The word $a_1 \dots a_n$ is the *label of the path*. We write $d \xrightarrow[G]{w} d'$ if there exists a path labelled by $w \in \Gamma^*$ from d to d'. We use the same notation in a more general case when the labels belong to an arbitrary monoid. We also write $d \xrightarrow[G]{L} d'$, where $L \subseteq \Gamma^*$, to mean that there exists $w \in L$ such that $d \xrightarrow[G]{w} d'$.

An *inverse* of G is the graph $\overline{G} := \{d' \xrightarrow{\overline{a}} d \mid d \xrightarrow[G]{a} d'\}$, $\overline{G} \subseteq D \times \overline{\Gamma} \times D$ where $\overline{\Gamma}$ is the alphabet of *formal inverses* of Γ. A map $h \colon \Gamma_2 \to \mathcal{R}\mathrm{at}((\Gamma_1 \cup \overline{\Gamma_1})^*)$, yields the map $h^{-1} \colon \mathrm{G}(\Gamma_1) \to \mathrm{G}(\Gamma_2)$ called *inverse rational mapping*[1] defined as follows $h^{-1}(G) := \{d \xrightarrow{a} d' \mid d \xrightarrow[G \cup \overline{G}]{h(a)} d'\}$. The image of a family of graphs $\mathcal{G}$ by all inverse rational mappings is written $\mathrm{RatM}^{-1}(\mathcal{G})$.

The *restriction* of $G \subseteq D \times \Gamma \times D$ to $C \subseteq D$, written $G|_C$ is the subgraph of G induced by C, namely $G|_C := G \cap (C \times \Gamma \times C)$. When C is a rational set, we speak of a *rational restriction*. This, of course, makes sense only when the vertices of the graph belong to a monoid $\mathcal{M}$. In this case we also define a multiplication on the left (resp. right) of a graph by an element of the monoid $m \in \mathcal{M}$, viz., $mG := \{mu \xrightarrow{a} mv \mid u \xrightarrow[G]{a} v\}$ (resp. $Gm := \{um \xrightarrow{a} vm \mid u \xrightarrow[G]{a} v\}$).

We denote by $G_1 \equiv G_2$ the existence of an isomorphism between two graphs G_1 and G_2. Given two families of graphs $\mathcal{G}_1$ and $\mathcal{G}_2$, we write $\mathcal{G}_1 \subseteqq \mathcal{G}_2$ if, for every $G_1 \in \mathcal{G}_1$, there exists $G_2 \in \mathcal{G}_2$ such that $G_1 \equiv G_2$. We write $\mathcal{G}_1 \equiv \mathcal{G}_2$ when both $\mathcal{G}_1 \subseteqq \mathcal{G}_2$ and $\mathcal{G}_2 \subseteqq \mathcal{G}_1$.

Given a graph G and two vertices d_1, d_2 of G, we define $\mathrm{L}(G, d_1, d_2) := \{w \mid d_1 \xrightarrow[G]{w} d_2\}$. The *trace* of a family of graphs $\mathcal{G}$, written $\mathrm{L}(\mathcal{G})$ is defined as follows: $\mathrm{L}(\mathcal{G}) := \{\mathrm{L}(G, d_1, d_2) \mid G \in \mathcal{G};\ d_1, d_2 \text{ are vertices of } G\}$.

3 Graphs Defined by String-Rewriting

Just like a Cayley graph is associated to a group presentation, we define a *Cayley–type graph* $\mathfrak{CG}_\Gamma(\mathcal{S})$ associated to an srs $\mathcal{S} \subseteq \Sigma^* \times \Sigma^*$ as follows:

$$\mathfrak{CG}_\Gamma(\mathcal{S}) := \{u \xrightarrow{a} v \mid a \in \Gamma,\ u, v \in \mathcal{I}\mathrm{rr}(\mathcal{S}),\ au \xrightarrow[\mathcal{S}]{*} v\}$$

[1] An inverse rational mapping $h^{-1} \colon \mathrm{G}(\Gamma_1) \to \mathrm{G}(\Gamma_2)$ is not an inverse of $h \colon \Gamma_2 \to \mathcal{R}\mathrm{at}((\Gamma_1 \cup \overline{\Gamma_1})^*)$.

where $\Gamma \subseteq \Sigma$. The expressive power is increased when the set of the vertices is restricted to a rational set $C \in \mathcal{R}\mathrm{at}(\Sigma^*)$:

$$\mathfrak{CG}_\Gamma(\mathcal{S}, C) := \mathfrak{CG}_\Gamma(\mathcal{S})|_C = \{u \xrightarrow{a} v \mid a \in \Gamma,\ u, v \in \mathcal{I}\mathrm{rr}(\mathcal{S}) \cap C,\ au \xrightarrow[\mathcal{S}]{*} v\} \ .$$

In the case of a configured system $(\mathcal{S}, C)$, we speak of *normal–form graph* $\mathfrak{NG}_\Gamma(\mathcal{S}, C)$ defined as follows:

$$\mathfrak{NG}_\Gamma(\mathcal{S}, C) := \{u \xrightarrow{a} v \mid a \in \Gamma,\ u, v \in \mathcal{I}\mathrm{rr}(\mathcal{S}) \cap C,\ au \xrightarrow[(\mathcal{S},C)]{*} v\} \ .$$

Note that, by definition, $au \in C$ and every reduction step from au into v is in C which is not the case for Cayley–type graphs (see examples on Fig. 1). In addition $\mathfrak{CG}_\Gamma(\mathcal{S}, C) = \mathfrak{NG}_\Gamma(\mathcal{S}, \Sigma^*)|_C$.

Let us denote by $\mathcal{F}\mathrm{in}\mathfrak{g}_\Gamma$ the class of finite graphs with edge labels in Γ. As expected, every finite graph may be encoded as both Cayley–type and normal–form graph.

Proposition 3.1. *The following classes of graphs coincide:*

(1) $\mathcal{F}\mathrm{in}\mathfrak{g}_\Gamma$,
(2) $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \subseteq \Sigma^* \times \Sigma^*,\ C \in \mathcal{F}\mathrm{in}(\Sigma^*)\}$,
(3) $\{\mathfrak{NG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \subseteq \Sigma^* \times \Sigma^*,\ C \in \mathcal{F}\mathrm{in}(\Sigma^*)\}$.

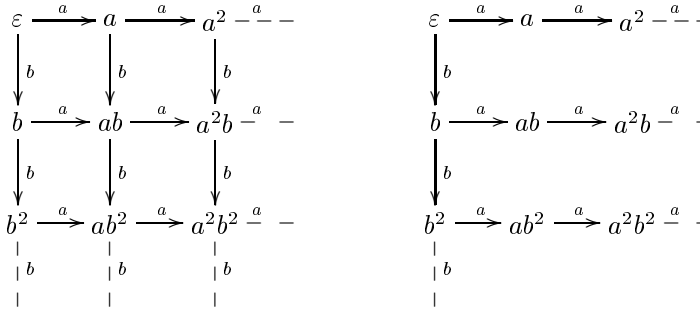


Fig. 1. $\mathfrak{CG}_\Gamma(\mathcal{S}, C)$ and $\mathfrak{NG}_\Gamma(\mathcal{S}, C)$ for $\mathcal{S} = \{ba \to ab\}$ and $C = a^*b^*$

4 Prefix-Recognizable Graphs

A well known restriction in string rewriting is the prefix rewriting [2, 5]. Given an srs $\mathcal{S}$, the *single–step prefix–reduction relation* w.r.t. $\mathcal{S}$, written $\xrightarrow[\mathcal{S}]{}\!\!\!\!\blacktriangleright$ is defined as the right closure of $\mathcal{S}$ by Σ^*: $\xrightarrow[\mathcal{S}]{}\!\!\!\!\blacktriangleright := \mathcal{S}\Sigma^* = \{(lx, rx) \mid l \to r \in \mathcal{S},\ x \in \Sigma^*\}$ and the transitive closure of $\xrightarrow[\mathcal{S}]{}\!\!\!\!\blacktriangleright$ is the *prefix–reduction relation* $\xrightarrow[\mathcal{S}]{*}\!\!\!\!\blacktriangleright$. For finite string rewriting systems, the prefix rewriting is equivalent to pushdown automata

[5] provided that the rules of an srs are labelled by Γ. A single–step reduction (resp. prefix–reduction) relation w.r.t. a labelled srs $\mathcal{L} \subseteq \Sigma^* \times \Gamma \times \Sigma^*$ is a graph

$$\mathfrak{g}_\Gamma(\mathcal{L}) := \xrightarrow[\mathcal{L}]{} = \{xly \xrightarrow{a} xry \mid l \xrightarrow{a} r \in \mathcal{L},\ x, y \in \Sigma^*\},$$
$$(\text{resp. } \mathfrak{pg}_\Gamma(\mathcal{L}) := \xrightarrow[\mathcal{L}]{\blacktriangleright} = \{ly \xrightarrow{a} ry \mid l \xrightarrow{a} r \in \mathcal{L},\ y \in \Sigma^*\}).$$

In the case of recognizable labelled string rewriting systems, the prefix rewriting together with rational restrictions leads to the family of *prefix–recognizable graphs* introduced in [6]:

$$\mathcal{P}\text{ref}\mathcal{R}\text{ec}\mathfrak{g}_\Gamma := \{ \xrightarrow[\mathcal{L}]{\blacktriangleright} |_D \mid \mathcal{L} = \bigcup_{i=1}^{k} L_i \times \{a_i\} \times R_i,\ k \in \mathbb{N},\ a_i \in \Gamma,$$
$$L_i, R_i, D \in \mathcal{R}\text{at}(\Sigma^*)\}$$
$$\equiv \{\bigcup_{i=1}^{n} (U_i \xrightarrow{a_i} V_i)W_i \mid n \in \mathbb{N},\ a_i \in \Gamma,\ U_i, V_i, W_i \in \mathcal{R}\text{at}(\Sigma^*)\},$$

where $(U_i \xrightarrow{a_i} V_i)W_i := \{u_i w_i \xrightarrow{a_i} v_i w_i \mid u_i \in U_i,\ v_i \in V_i,\ w_i \in W_i\}$. We shall use both expressions as alternative definitions of this family of graphs (see [6] for the proof of the above). We recall from [6] that $\mathrm{L}(\mathcal{P}\text{ref}\mathcal{R}\text{ec}\mathfrak{g}_\Gamma)$ is exactly the family of context–free languages, $\mathcal{P}\text{ref}\mathcal{R}\text{ec}\mathfrak{g}_\Gamma$ is an effective boolean algebra, the transitive closure of a prefix–recognizable relation is prefix–recognizable and every graph of $\mathcal{P}\text{ref}\mathcal{R}\text{ec}\mathfrak{g}_\Gamma$ has a decidable monadic theory.

In order to characterize prefix–recognizable graphs as normal form graphs, we use prefix configured systems.

Definition 4.1. *A configured system $(\mathcal{S}, C)$ is* prefix*, if every LR–overlap $(x_1, l_1 \to r_1, y_1, x_2, l_2 \to r_2, y_2)$ of $(\mathcal{S}, C)$ is such that $x_1 = x_2$. The class of prefix configured system is written* $\mathcal{P}$refCS*.*

The fact that in some situations the rewriting w.r.t. a configured system which is prefix acts as prefix rewriting is essential for the proof of the following.

Theorem 4.2. *The following classes of graphs coincide:*

(1) $\mathcal{P}\text{ref}\mathcal{R}\text{ec}\mathfrak{g}_\Gamma$,
(2) $\{\mathfrak{MG}_\Gamma(\mathcal{S}, C) \mid (\mathcal{S}, C) \in \mathcal{F}\text{inCS} \cap \mathcal{P}\text{refCS} \cap \mathcal{S}\text{tabCS}\}$,
(3) $\{\mathfrak{MG}_\Gamma(\mathcal{S}, C) \mid (\mathcal{S}, C) \in \mathcal{R}\text{ecCS} \cap \mathcal{P}\text{refCS} \cap \mathcal{S}\text{tabCS}\}$.

We now turn our attention to Cayley–type graphs. In order to characterize prefix–recognizable graphs, it is sufficient to check for overlaps only for descendants of some specific sets.

Definition 4.3. *An srs $\mathcal{S}$ is* prefix from $D \subseteq \Sigma^*$*, if every LR–overlap $(x_1, l_1 \to r_1, y_1, x_2, l_2 \to r_2, y_2)$ of $\mathcal{S}$ such that $x_1 l_1 y_1 \in \xrightarrow[\mathcal{S}]{*}(D)$ satisfies $x_1 = x_2$.*

For an srs $\mathcal{S}$, the property of being prefix from D guarantees that

$$\xrightarrow[\mathcal{S}]{*} \cap \left((D \cap \Gamma\mathcal{I}\text{rr}(\mathcal{S})) \times \Sigma^*\right) = \xrightarrow[\mathcal{S}]{*\blacktriangleright} \cap \left((D \cap \Gamma\mathcal{I}\text{rr}(\mathcal{S})) \times \Sigma^*\right) .$$

Proposition 4.4. *The following problem is decidable:*
instance: $\mathcal{S} \in \mathrm{Rec}(\Sigma^* \times \Sigma^*)$, $D \in \mathrm{Rat}(\Sigma^*)$,
question: is $\mathcal{S}$ *prefix from* D *?*

For the proof of the above proposition, the following lemma is useful.

Lemma 4.5 ([7]). *For every* $(\mathcal{S}, C) \in \mathrm{RecCS} \cap \mathrm{PrefCS} \cap \mathrm{StabCS}$, $\xrightarrow[(\mathcal{S},C)]{*}$ *is a rational relation.*

The characterization of prefix–recognizable graphs in terms of Cayley–type graphs is given in the following.

Theorem 4.6. *The following classes of graphs coincide:*

(1) $\mathrm{Pref}\mathfrak{Rec}_\Gamma$,
(2) $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C) \mid C \in \mathrm{Rat}(\Sigma^*),\ \mathcal{S}$ *is finite and prefix from* $\Gamma C\}$,
(3) $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C) \mid C \in \mathrm{Rat}(\Sigma^*),\ \mathcal{S}$ *is recognizable and prefix from* $\Gamma C\}$.

5 Rational Graphs

A graph $G \subseteq \Sigma^* \times \Gamma \times \Sigma^*$ is *rational* when, for each $a \in \Gamma$, the edge relation $\xrightarrow[G]{a}$ is a rational subset of $\Sigma^* \times \Sigma^*$. Rational graphs and have been introduced in [13]. We denote by $\mathrm{Rat}\mathfrak{g}_\Gamma$ the class of rational graphs with edge labels in Γ. As established in [14], $\mathrm{L}(\mathrm{Rat}\mathfrak{g}_\Gamma)$ is exactly the family of context sensitive languages.

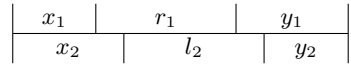
In a more concrete way, rational graphs may be seen as relation graphs of rational transducers with labelled accepting states. To a rational transducer $\mathcal{T} = (Q, \xrightarrow[\mathcal{T}]{}, \iota, (F_a)_{a\in\Gamma})$ where Q is a finite set of states, $\xrightarrow[\mathcal{T}]{} \subseteq Q \times \Sigma^* \times \Sigma^* \times Q$ is a finite transition relation, $\iota \in Q$ is an initial state and $F_a \subseteq Q$ is a set of final states labelled by $a \in \Gamma$, we associate its relation graph:

$$\mathfrak{rg}_\Gamma(\mathcal{T}) := \{u \xrightarrow{a} v \mid a \in \Gamma,\ \exists f \in F_a : \iota \xrightarrow[\mathcal{T}]{(u,v)} f\}$$

We denote by $\mathfrak{rg}_\Gamma\mathrm{RT}$ the family of relation graphs of rational transducers. Of course $\mathfrak{rg}_\Gamma\mathrm{RT} = \mathrm{Rat}\mathfrak{g}_\Gamma$.

In order to characterize rational graphs as normal form graphs, we use right configured systems.

Definition 5.1. *A configured system* $(\mathcal{S}, C)$ *is* right*, if every LR–overlap* $(x_1, l_1 \to r_1, y_1, x_2, l_2 \to r_2, y_2)$ *of* $(\mathcal{S}, C)$ *is such that* $|x_1| \leq |x_2|$ *and* $|x_1 r_1| \leq |x_2 l_2|$*:*



Right configured systems have been studied in [7] and linked to rational relations as follows.

Lemma 5.2 ([7]). *For every* $(\mathcal{S}, C) \in \mathrm{RecCS} \cap \mathrm{RightCS} \cap \mathrm{StabCS}$, $\xrightarrow[(\mathcal{S},C)]{*}$ *is a rational relation.*

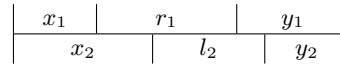
The above lemma is useful in proving the following characterization of rational graphs in terms of normal form graphs.

Theorem 5.3. *The following classes of graphs coincide:*

(1) $\mathcal{R}\mathrm{at}\mathfrak{g}_\Gamma$,
(2) $\{\mathfrak{NG}_\Gamma(\mathcal{S},C) \mid (\mathcal{S},C) \in \mathcal{F}\mathrm{in}\mathcal{CS} \cap \mathcal{S}\mathrm{tab}\mathcal{CS} \cap \mathcal{R}\mathrm{ight}\mathcal{CS}\}$,
(3) $\{\mathfrak{NG}_\Gamma(\mathcal{S},C) \mid (\mathcal{S},C) \in \mathcal{R}\mathrm{ec}\mathcal{CS} \cap \mathcal{S}\mathrm{tab}\mathcal{CS} \cap \mathcal{R}\mathrm{ight}\mathcal{CS}\}$,

The second characterization of rational graphs is given in terms of Cayley–type graphs. Again, we have to forbid some overlaps but the restriction needs only to be limited to the context of words that encode the vertices of a graph.

Definition 5.4. *An srs $\mathcal{S}$ is* right from D *if every LR–overlap $(x_1, l_1 \to r_1, y_1, x_2, l_2 \to r_2, y_2)$ of $\mathcal{S}$ such that $x_1 l_1 y_1 \in \overset{*}{\underset{\mathcal{S}}{\to}}(D)$ satisfies $|x_1| \leq |x_2|$ and $|x_1 r_1| \leq |x_2 l_2|$:*



Proposition 5.5. *The following problem is decidable:*
instance: $\mathcal{S} \in \mathcal{R}\mathrm{ec}(\Sigma^* \times \Sigma^*)$, $D \in \mathcal{R}\mathrm{at}(\Sigma^*)$,
question: is $\mathcal{S}$ right from D ?

We have the following characterization of rational graphs in terms of Cayley–type graphs.

Theorem 5.6. *The following classes of graphs coincide:*

(1) $\mathcal{R}\mathrm{at}\mathfrak{g}_\Gamma$
(2) $\{\mathfrak{CG}_\Gamma(\mathcal{S},C) \mid \mathcal{S}$ *is finite and right from* $\Gamma C\}$,
(3) $\{\mathfrak{CG}_\Gamma(\mathcal{S},C) \mid \mathcal{S}$ *is recognizable and right from* $\Gamma C\}$.

6 Recursively Enumerable Graphs

There are at least two possibilities for associating graphs to Turing machines. We may use an off–line Turing machine with a read–only input tape and a work tape. The head of the input tape cannot move backwards. The vertices of the graph are the configurations of the work tape. There is an a–labelled edge from a configuration c_1 to a configuration c_2 if there is a computation of the machine that leads from c_1 to c_2 whereas only one letter a is read on the input tape. We speak of observable transition graph in this case. Of course, as in former sections, it is useful to restrict the vertices of a graph to a rational set C. This definition is introduced in [11] and further discussed in [8].

A second possibility, considered in [8] is based on a relation graph. The accepting states of a single–tape Turing machine $\mathcal{M}$ are labelled by Γ. Formally, $\mathcal{M} = (Q, T, \iota, (F_a)_{a\in\Gamma})$ where Q is a finite set of states, $\iota \in Q$ is an initial state, $F_a \subseteq Q$ is the set of accepting states labelled by $a \in \Gamma$ and T is a finite set of transition rules. The tape alphabet $\Sigma_\square := \Sigma \cup \{\square\}$ is extended with the blank $\square \notin \Sigma$ and transition rules are of the form $pA \to qB\delta$ with $p, q \in Q$, $A, B \in \Sigma_\square$

and $\delta \in \{+,-\}$. A *single–step computation relation* $\xrightarrow[\mathcal{M}]{} \subseteq \Sigma_{\square}^* Q \Sigma_{\square}^* \times \Sigma_{\square}^* Q \Sigma_{\square}^*$ of $\mathcal{M}$ is defined as follows

$$\begin{aligned}\xrightarrow[\mathcal{M}]{} := {} & \{(upAv, uBqv) \mid pA \to qB+ \in T,\ u,v \in \Sigma_{\square}^*\} \cup \\ & \{(up, uBq) \mid p\square \to qB+ \in T,\ u \in \Sigma_{\square}^*\} \cup \\ & \{(uCpAv, uqCBv) \mid pA \to qB- \in T,\ C \in \Sigma_{\square},\ u,v \in \Sigma_{\square}^*\} \cup \\ & \{(uCp, uqCB) \mid p\square \to qB- \in T,\ C \in \Sigma_{\square},\ u \in \Sigma_{\square}^*\} \cup \\ & \{(pAv, q\square Bv) \mid pA \to qB- \in T,\ v \in \Sigma_{\square}^*\} \cup \\ & \{(p, q\square B) \mid p\square \to qB- \in T\}\end{aligned}$$

The *relation graph* $\mathfrak{tg}_\Gamma(\mathcal{M})$ of $\mathcal{M}$ is defined as follows:

$$\mathfrak{rg}_\Gamma\mathrm{TM} := \{u \xrightarrow{a} \overleftarrow{v}\,\overrightarrow{w} \mid u \in \Sigma^*,\ v,w \in \Sigma_{\square}^*,\ a \in \Gamma,\ \exists f \in F_a : \iota u \xrightarrow[\mathcal{M}]{*} vfw\}$$

where $\overleftarrow{v}$ (resp. $\overrightarrow{w}$) stands for the longest suffix (resp. prefix) of v (resp. w) which does not contain $\square$. The class of relation (resp. transition) graphs of Turing machines is written $\mathfrak{rg}_\Gamma\mathrm{TM}$ (resp. $\mathfrak{tG}_\Gamma\mathrm{TM}$). In [8], it is established that $\mathfrak{rg}_\Gamma\mathrm{TM} \equiv \mathfrak{tG}_\Gamma\mathrm{TM}$. Thus, the class of *recursively enumerable graphs*, written $\mathrm{RE}\mathfrak{g}_\Gamma$, is the class of graphs of Turing machines: $\mathrm{RE}\mathfrak{g}_\Gamma := \mathfrak{rg}_\Gamma\mathrm{TM} \equiv \mathfrak{tG}_\Gamma\mathrm{TM}$. Several other characterizations of this class are given in [8], among which we need the following for the main theorem of this section (i.e. Theorem 6.2).

Theorem 6.1 ([8]). $\mathfrak{rg}_\Gamma\mathrm{TM} \equiv \mathrm{Rat}\mathrm{M}^{-1}(\mathfrak{rg}_\Gamma\mathrm{RT})$.

Of course, $\mathrm{L}(\mathrm{RE}\mathfrak{g}_\Gamma)$ is the family of recursively enumerable languages.

The following theorem gives characterizations of the class of recursively enumerable graphs in terms of both normal–form graphs and Cayley–type graphs.

Theorem 6.2. *The following families of graphs coincide:*

(1) $\mathrm{RE}\mathfrak{g}_\Gamma$.
(2) $\{\mathfrak{NG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \in \mathcal{F}\mathrm{in}(\Sigma^* \times \Sigma^*),\ C \in \mathcal{R}\mathrm{at}(\Sigma^*)\}$,
(3) $\{\mathfrak{NG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \in \mathcal{R}\mathrm{ec}(\Sigma^* \times \Sigma^*),\ C \in \mathcal{R}\mathrm{at}(\Sigma^*)\}$,
(4) $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \in \mathcal{F}\mathrm{in}(\Sigma^* \times \Sigma^*),\ C \in \mathcal{R}\mathrm{at}(\Sigma^*)\}$,
(5) $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C) \mid \mathcal{S} \in \mathcal{R}\mathrm{ec}(\Sigma^* \times \Sigma^*),\ C \in \mathcal{R}\mathrm{at}(\Sigma^*)\}$,

To close this section, we point out a similar result of [16, 17] establishing that there is an observational equivalence (up to τ–transitions) between the class of rooted Cayley–type graphs of finite srs $\{\mathfrak{CG}_\Gamma(\mathcal{S}, C)_{\mathrm{acc}} \mid \mathcal{S} \in \mathcal{F}\mathrm{in},\ C \in \mathcal{R}\mathrm{at}(\Sigma^*)\}$ and the class of rooted transition graphs of Turing machines (with nonobservable transitions τ) $\mathfrak{tg}_\Gamma\mathrm{TM}$.

7 Conclusion

We have considered four families of graphs: finite graphs $\mathcal{F}\mathrm{in}\mathfrak{g}_\Gamma$, prefix–recognizable graphs $\mathcal{P}\mathrm{ref}\mathcal{R}\mathrm{ec}\mathfrak{g}_\Gamma$, rational graphs $\mathcal{R}\mathrm{at}\mathfrak{g}_\Gamma$, and recursively enumerable

graphs $\mathrm{RE}\mathfrak{g}_\Gamma$. These families form a strict hierarchy analogous to the Chomsky hierarchy of formal languages. Indeed the notion of trace relates each level of the hierarchy of graphs to the Chomsky hierarchy: $\mathrm{L}(\mathcal{F}\mathrm{in}\mathfrak{g}_\Gamma)$ is the family of rational languages, $\mathrm{L}(\mathcal{P}\mathrm{ref}\mathcal{R}\mathrm{ec}\mathfrak{g}_\Gamma)$ is the family of context–free languages, $\mathrm{L}(\mathcal{R}\mathrm{at}\mathfrak{g}_\Gamma)$ is the family of context–sensitive languages and $\mathrm{L}(\mathrm{RE}\mathfrak{g}_\Gamma)$ is the family of recursively enumerable languages.

Whereas the Chomsky hierarchy has a uniform presentation in terms of grammars (type 0, 1, 2 and 3), we have shown that string rewriting provides a uniform presentation of a Chomsky–like hierarchy of graphs. We have established that both notions of Cayley–type graph and normal form graph lead to the same Chomsky–like hierarchy. In addition, we have established that in both cases finite string–rewriting systems are as expressive as recognizable ones.

We did not discussed sub-families of the Chomsky–like hierarchy. At least one of them has a known characterization in terms of string rewriting, namely the family of pushdown transition graphs [4].

References

1. J.-M. Autebert, J. Berstel, and L. Boasson. Context–free languages and push–down automata. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, pages 111–174. Springer–Verlag, 1997.
2. J. R. Bchi. Regular canonical systems. *Archiv fr Math. Logik und Grundlagenforschung*, 6:91–111, 1964.
3. J. Berstel. *Transductions and Context–Free Languages*. B. G. Teubner, Stuttgart, 1979.
4. H. Calbrix and T. Knapik. A string–rewriting characterization of Muller and Schupp's context–free graphs. In V. Arvind and R. Ramanujam, editors, *18th International Conference on Foundations of Software Technology and Theoretical Computer Science*, LNCS 1530, pages 331–342, Chennai, Dec. 1998.
5. D. Caucal. On the regular structure of prefix rewriting. *Theoretical Comput. Sci.*, 106:61–86, 1992.
6. D. Caucal. On infinite transition graphs having a decidable monadic second–order theory. In F. M. auf der Heide and B. Monien, editors, *23th International Colloquium on Automata Languages and Programming*, LNCS 1099, pages 194–205, Paderborn, July 1996. A long version will appear in TCS.
7. D. Caucal. On word–rewriting systems having a rational derivation. In J. Tiuryn, editor, *FoSSaCS '2000*, LNCS 1784, pages 48–62, Berlin, Mar. 2000.
8. D. Caucal. On the transition graphs of Turing machines. In M. Margenstern and Y. Rogozhin, editors, *MCU '2001*, LNCS 2055, pages 177–189, Chisinau, May 2001.
9. B. Khoussainov and A. Nerode. Automatic presentations of structures. In D. Leivant, editor, *Logic and Computational Complexity*, LNCS 960, pages 367–392, Indianapolis, Oct. 1994. Selected papers from the LCC'94 Intl. Workshop.
10. T. Knapik. Spécifications de Thue. Technical Report INF/95/12/02/a, Institut de Recherche en Mathematiques et Informatique Appliquées, 1996.
11. T. Knapik and É. Payet. Synchronized product of linear bounded machines. In G. Ciobanu and G. Paun, editors, *12 th International Symposium on Fundamentals of Computation Theory*, LNCS 1684, pages 362–373, Iaşi, Aug. 1999.

12. A. Mateescu and A. Salomaa. Aspects of classical language theory. In G. Rozenberg and A. Salomaa, editors, *Word, Language, Grammar*, volume 1 of *Handbook of Formal Languages*, pages 175–251. Springer–Verlag, 1997.
13. C. Morvan. On rational graphs. In J. Tiuryn, editor, *FoSSaCS '2000*, LNCS 1784, pages 252–266, Berlin, Mar. 2000.
14. C. Morvan and C. Stirling. Rational graphs trace context-sensitive languages. In A. Pultr, J. Sgall, and P. Kolman, editors, *MFCS '2001*, LNCS 2136, Marianske Lazne, Aug. 2001.
15. D. E. Muller and P. E. Schupp. Pushdown automata, graphs, ends, second–order logic, and reachability problems. In *Proceedings of the 13th Annual ACM Symposium on Theory of Computing*, pages 46–54, Milwaukee, May 1981.
16. É. Payet. *Produit synchronisé pour quelques classes de graphes infinis*. PhD thesis, Université de la Réunion, 2000.
17. É. Payet. Thue specifications, infinite graphs and synchronized product. *Fundamenta Informaticae*, 44(3):265–290, 2000.
18. W. Thomas. A short introduction to infinite automata. In W. Kuich, G. Rozenberg, and A. Salomaa, editors, *Developments in Language Theory, 5th International Conference, DLT 2001*, LNCS 2295, pages 130–144, Vienna, July 2001.