Rational graphs trace context-sensitive languages

Christophe Morvan[1] and Colin Stirling[2]

[1] IRISA, Campus de Beaulieu, 35042 Rennes, France
christophe.morvan@irisa.fr
[2] Division of Informatics, University of Edinburgh
cps@dcs.ed.ac.uk

Abstract. This paper shows that the traces of rational graphs coincide with the context-sensitive languages.

1 Introduction

Infinite transition graphs have become a focus of attention recently. For example, the behaviour of an infinite state system is an infinite transition graph, and researchers have examined when property checking and equivalence checking such graphs is decidable [3]. Muller and Schupp pioneered the study of infinite transition graphs by examining the graphs of pushdown automata [10]. They were interested in extending Cayley graphs to structures that are more general than groups. Courcelle defined a more extensive class, the equational graphs, using deterministic graph grammars [5]. More recently, Caucal constructed a richer class, the prefix-recognisable graphs [4], using transformations of the complete binary tree [4]: such a graph is characterised by an inverse rational substitution followed by a rational restriction of the complete binary tree. Caucal also provided a mechanism for generating the prefix-recognisable graphs, using finite sets of rewrite rules. The first author produced an even richer family, the rational graphs, again using transformations of the complete binary tree [9]: the characterisation involves an inverse linear substitution followed by a rational restriction of the complete binary tree. He also showed that rational graphs are generated by finite-state transducers whose words are the vertices of a graph.

There is a natural relation between transition graphs and languages, namely *the trace*. A trace of a path in a transition graph is its sequence of labels. Relative to designated initial and final sets of vertices, the trace of a transition graph is the set of all path traces which start at an initial vertex and end at a final vertex. For example, the regular languages are the traces of finite transition graphs. For richer families of languages one needs to consider infinite transition graphs. The context-free languages are the traces of the transition graphs of pushdown automata. This remains true for both equational and prefix-recognisable graphs. The next family of languages in the Chomsky hierarchy is the context-sensitive languages. Although they have been studied from a language theoretic point of view, (see [8], for example), only recently have their canonical graphs been examined [7].

In this paper we prove that the traces of rational graphs (relative to regular initial and final vertex sets) coincide exactly with the context-sensitive languages. In Section 2 we describe the rational graphs. In Section 3 we prove that the traces of rational graphs are context-sensitive, using linear bounded Turing machines. Finally, in Section 4 we prove the converse inclusion, using a normal form due to Penttonen for context-sensitive languages [11].

2 Rational graphs

In this section we concentrate on a presentation of rational graphs using finite state transducers that generate them. For a more detailed introduction to rational graphs, their basis using partial semigroups and rational relations, and their characterization in terms of transformations of the complete binary tree, see [9].

Assume a finite alphabet $\mathcal{A}$ and a finite set of symbols X. A vertex of a transition graph is a word $u \in X^*$, and a transition has the form (u, a, v), which we write as $u \xrightarrow{a} v$, where $a \in \mathcal{A}$ and u, v are vertices. A transition graph $G \subseteq X^* \times \mathcal{A} \times X^*$ is a set of transitions.

A transducer is a finite state device that transforms an input word into an output word in stages, see [1, 2]. At each stage, it reads a subword, transforms it, and changes state. A transition of a transducer has the form $p \xrightarrow{u/v} q$ where p and q are states and u is the input subword and v is its transformation. For our purposes, both u and v are elements of X^*, and final states of the transducer are labelled by subsets of $\mathcal{A}$.

Definition 2.1. A *labelled transducer* $T = (Q, I, F, E, L)$ over X and $\mathcal{A}$, is 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 $E \subseteq Q \times X^* \times X^* \times Q$ and a labelling $L : F \to 2^{\mathcal{A}}$.

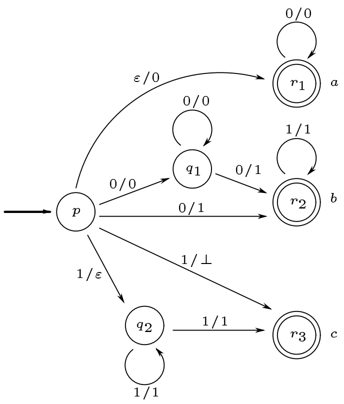
A transition $u \xrightarrow{a} v$ is *recognized* by a labelled transducer T if there is a path in T, $p_0 \xrightarrow{u_1/v_1} p_1 \xrightarrow{u_2/v_2} \dots \xrightarrow{u_{n-1}/v_{n-1}} p_{n-1} \xrightarrow{u_n/v_n} p_n$, with $p_0 \in I$ and $p_n \in F$ and $u = u_1 \dots u_n$ and $v = v_1 \dots v_n$ and $a \in L(p_n)$. Labelled transducers provide a simple characterization of rational graphs, see [9]. Here we treat the characterization as a definition.

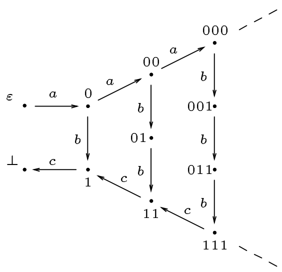
Definition 2.2. A graph $G \subseteq X^* \times \mathcal{A} \times X^*$ is rational if, and only if, G is recognized by labelled transducer.

A transducer is *normalised* if all its transitions have the form, $p \xrightarrow{u/v} q$, where $|u| + |v| = 1$. It is straightforward to show that any rational graph is generable from a normalised transducer.

For each a in $\mathcal{A}$, the subgraph G_a is the restriction of G to transitions labelled a. If G is rational then so is G_a, and we let T_a be the transducer that recognises G_a. If u is a vertex of G then $G_a(u)$ is the set of vertices $\{v \mid u \xrightarrow{a} v\}$.

Example 2.3. The transition graph, below on the right, is generated by the transducer, below on the left.





For example, $001 \xrightarrow{b} 011$ is recognized because of the following path, $p \xrightarrow{0/0} q_1 \xrightarrow{0/1} r_2 \xrightarrow{1/1} r_2$ and b is associated with the final state r_2.

A path in a graph G is a sequence of transitions $u_0 \xrightarrow{a_0} u_1 \xrightarrow{a_1} \ldots \xrightarrow{a_n} u_n$. If $x = a_0 a_1 \ldots a_n$ we write $u_0 \xrightarrow{x} u_n$ and we say that x is the trace of this path. The trace of a graph G, relative to a set I of initial vertices and a set F of final vertices, is the set of traces of all paths between vertices in I and vertices in F.

$$L(G, I, F) := \{x \mid \exists s \in I \; \exists t \in F, s \xrightarrow{x} t\}$$

In other words, the trace of a graph is "the language of its labels". In this paper, we are interested in the trace of a rational graph relative to regular vertex sets I and F.

In Section 4 we shall appeal to rational relations $R \subseteq X^* \times X^*$. A relation R is rational if it is induced by an unlabelled transducer over X: that is, a transducer that does not have a labelling function L, see [1, 2].

3 Traces of rational graphs are context-sensitive

In this section, we prove that the trace of a rational graph G, relative to regular vertex sets I and F, is a context-sensitive language. In fact, we only need to consider the trace of G relative to $I = \{\$\}$ and $F = \{\#\}$, where \$ and # are two new symbols. If G is rational then the following graph G',

$$G \cup \Big(\bigcup_{a \in \mathcal{A}} \{\$\} \times \{a\} \times G_a(I)\Big) \cup \Big(\bigcup_{a \in \mathcal{A}} G_a^{-1}(F) \times \{a\} \times \{\#\}\Big) \cup \Big(\bigcup_{C} \{\$\} \times \{a\} \times \{\#\}\Big)$$

where C is the constraint $a \in \mathcal{A},\ G_a(I) \cap F \neq \emptyset$, is also rational and has the property, $L(G', \{\$\}, \{\#\}) = L(G, I, F)$.

The two most common characterizations of a context-sensitive (cs) language are that it is generated by a cs grammar and that it is recognized by a linear bounded machine, LBM, see for instance [8].

Definition 3.1. A linear bounded machine, LBM, is a Turing machine such that the size of its tape is bounded, linearly, by the length of the input.

At a first glance, one might expect that the trace of a rational graph is recognisable by a LBM because it can store information on its tape (namely the current vertex of the graph), and use it to compute new information (the next vertex). However, the linear bound is a problem as the next example illustrates.

Example 3.2. The very simple transducer, $p \xrightarrow{A/AA} p$ (where p is both an initial and a final state labelled with a), produces the following graph G.

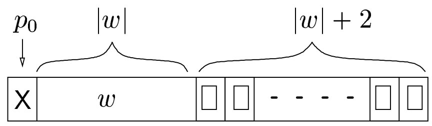
$$\begin{array}{lllll}
A \xrightarrow{a} AA & \xrightarrow{a} A^4 & \xrightarrow{a} \cdots & \xrightarrow{a} & A^{2^n} \cdots \\
A^3 \xrightarrow{a} A^6 & \xrightarrow{a} A^{12} & \xrightarrow{a} \cdots & \xrightarrow{a} & A^{3.2^n} \cdots \\
A^5 \xrightarrow{a} A^{10} & \xrightarrow{a} A^{20} & \xrightarrow{a} \cdots & \xrightarrow{a} & A^{5.2^n} \cdots \\
\vdots & \vdots & \vdots & &
\end{array}$$

The trace of G relative to $I = \{A\}$ and $F = A^*$ is the language a^*. But, the length of vertices is exponential in the length of the word that is recognized. For example, the path recognizing a^3 is $A \xrightarrow{a} AA \xrightarrow{a} A^4 \xrightarrow{a} A^8$.

Our solution[1] is to work on transitions in parallel. If $u \xrightarrow{a} v \xrightarrow{b} w$ and the first "output" in the transducer T_a involves a transition $q \xrightarrow{\epsilon/X} q'$ then X will be the first element of v and therefore we can also activate T_b (and remember that q' is then the current state of T_a). In this way, as we shall see, we only ever need to be working with current head elements which may activate subsequent transitions.

Proposition 3.3. *Traces of rational graphs are cs-languages.*

Proof (Sketch). Assume a rational graph G that is generated by a normalised transducer T. We now show that $L(G, \{\$\}, \{\#\})$ is recognised by a LBM, M. Let $w \in \mathcal{A}^*$, and assume that w_k is the kth letter of w. The tape of M has length $2|w| + 3$, and it has a left-end marker X. The initial configuration of M is,



where the tape symbols are the left-end marker, the letters of $\mathcal{A}$, and then blank symbols. M then transforms this configuration into

[1] One might believe that it is possible to encode vertices in such a way that their lengths are linear in the length of the recognized word. But, the problem is then how to deduce the "next vertex function". In particular, if some branches of the transducer have a linear growth and others have an exponential growth, we are unable to construct the machine.

$$
\begin{array}{ccccccccccccc}
 & p_1 & & & & & & & & & & & \\
 & \downarrow & & & & & & & & & & & \\
\end{array}
$$

$$
\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
\mathtt{X} & i_{w_1} & \$ & i_{w_2} & \varepsilon & --- & i_{w_k} & \varepsilon & --- & i_{w_n} & \varepsilon & \% & \varepsilon \\
\hline
\end{array}
$$

where ε and % are new tape-symbols, and i_{w_k} are symbols corresponding to an initial state of T_{w_k}. We assume that there are tape symbols for each state of the transducer T.

Now, we consider transitions where the head moves to the right. Assume that the current configuration is,

$$
\begin{array}{c|c|c|c|c|c}
 & p_1 & & & & \\
 & \downarrow & & & & \\
\hline
\text{- - -} & p_a & A & p_b & C & \text{- - - -} \\
\hline
\end{array}
$$

where A and C both belong to $\mathcal{A} \cup \{\$, \#, \varepsilon\}$, p_a is any state of T_a and p_b is either a state of T_b or %. There are two possible moves. First, if there is a transition $p_a \xrightarrow{A/\varepsilon} q_a$ in T_a then M can make the following step.

$$
\begin{array}{c|c|c|c|c|c}
 & & & p_1 & & \\
 & & & \downarrow & & \\
\hline
\text{- - -} & q_a & \varepsilon & p_b & C & \text{- - - -} \\
\hline
\end{array}
\quad \text{Move (a)}
$$

A is removed because the transition $p_a \xrightarrow{A/\varepsilon} q_a$ consumes this input. C is unchanged because ε is the output. We also assume that M remains in state p_1. Second, if there is a transition $p_a \xrightarrow{\varepsilon/B} q'_a$ in T_a and C is ε then M can make the following step.

$$
\begin{array}{c|c|c|c|c|c}
 & & & p_1 & & \\
 & & & \downarrow & & \\
\hline
\text{- - -} & q'_a & A & p_b & B & \text{- - - -} \\
\hline
\end{array}
\quad \text{Move (b)}
$$

A is not removed because the transition $p_a \xrightarrow{\varepsilon/B} q'_a$ consumes nothing. $C = \varepsilon$ is changed to B. Again, we assume M remains in state p_1.

Next, we consider transitions where the head moves to the left. If the current configuration is

$$
\begin{array}{c|c|c|c|c|c}
 & & & p_1 & & \\
 & & & \downarrow & & \\
\hline
\text{- - -} & p_b & B & p_a & C & \text{- - - -} \\
\hline
\end{array}
$$

and either $C = \varepsilon$ or ($C = \#$ and $p_a = \%$) then M can make the following step.

$$
\begin{array}{c|c|c|c|c|c}
 & p_1 & & & & \\
 & \downarrow & & & & \\
\hline
\text{- - -} & p_b & B & p_a & C & \text{- - - -} \\
\hline
\end{array}
\quad \text{Move (c)}
$$

Finally, there is the case when M checks for success. This happens either when the head is scanning % and the next symbol is $\#$ or the head is scanning f_{w_i} and the next symbol is ε.

						p_1 $\downarrow$					p_1 $\downarrow$	
X	p_{w_1}	A_1	p_{w_2}	A_2	- - - - -	f_{w_i}	ε	- - - - -	p_{w_n}	A_n	%	#

M now checks that each p_{a_i} is a final state f_{a_i} of T_{a_i} and that A_i is ε, and if this is true, then M enters a final accepting state. It is now not difficult to show that M recognizes the language $L(G, \{\$\}, \{\#\})$. $\square$

4 Cs-languages are traces of rational graphs

In this section we prove the converse inclusion that every cs-language is the trace of a rational graph relative to regular initial and final sets of vertices. The proof is subtle, for reasons that we shall explain, and uses Penttonen's characterization of a cs-language as the image of a linear language under a length preserving left cs transformation [11].

A cs-language is generated by a cs grammar. A cs grammar Γ consists of a finite set of nonterminals N, a finite set of terminals T and a finite set of productions P, each of which has the form $UAW \to UVW$, where U and W are in $(N \cup T)^*$, A in N and V in $(N \cup T)^+$. An application of the production $UAW \to UVW$ in Γ to U_1UAWW_1 produces U_1UVWW_1, which we write as $U_1UAWW_1 \underset{\Gamma}{\to} U_1UVWW_1$. Moreover, we assume that $\overset{*}{\underset{\Gamma}{\to}}$ is the reflexive and transitive closure of $\underset{\Gamma}{\to}$. Sometimes, we shall use the notation $\overset{*}{\underset{\Gamma}{\to}}(U)$ to be the set $\{V \mid U \overset{*}{\underset{\Gamma}{\to}} V\}$. The language of $U \in (N \cup T)^+$, denoted by $\mathsf{L}(U)$, is the set of words $\left\{u \in T^* \mid U \underset{\Gamma}{\to}^* u\right\}$. Usually, there is a start symbol $S \in N$ of Γ, in which case the language generated by the grammar, $\mathsf{L}(\Gamma)$, is $\mathsf{L}(S)$. There are various normal forms for cs grammars, including the "left form" due to Penttonen [11].

Theorem 4.1 (Penttonen 74). *Every cs-language can be generated by a grammar, whose productions have the form*

$$A \to BC, AB \to AC, A \to a$$

where A, B, C are nonterminals and a is a terminal.

If ε-transitions are allowed in rational graphs and Γ is a cs grammar, then it is easy to construct a rational graph whose trace is $\mathsf{L}(\Gamma)$. One merely simulates the derivation $S \underset{\Gamma}{\to}^* v$ in the graph. Each production can be simulated by a transducer whose final state is labelled ϵ (for instance, $AB \to AC$ is captured by the transducer $p \xrightarrow{D/D} p$, $p \xrightarrow{AB/AC} q$ and $q \xrightarrow{D/D} q$ for each $D \in N$) and the graph outputs v at the end, $v \xrightarrow{v} \varepsilon$. However, the traces of rational graphs with ε-transitions coincide with the recursively enumerable languages. Indeed, following Knapik and Payet's exposition in [7], it is straightforward to prove the next result. It appeals to the standard notion of projection π with respect to a subalphabet: if $J \subseteq \mathcal{B}^*$ and $\mathcal{C} \subseteq \mathcal{B}$ then $\pi_{\mathcal{C}}(J)$ is the language that is the result of erasing all occurrences of letters in $\mathcal{B} - \mathcal{C}$ from words in J.

Corollary 4.2. *If $K \subseteq \mathcal{A}^*$ is a recursively enumerable language and $c \notin \mathcal{A}$ is a new letter, then there is a language $K' \subseteq (\mathcal{A} \cup \{c\})^*$ and a rational graph G whose trace is K' and $K = \pi_{\mathcal{A}}(K')$.*

This result is also a well known property of cs-languages (see, for example, [6]). However, the trace of a rational graph without ε-transitions is a recursive language, [9], and therefore the rational graphs with ε-transitions are more expressive than rational graphs.

What is the real problem of being able to generate a cs-language from a rational graph? As we can see simulating an application of a single production does not create a problem. And therefore simulating sequences of productions is also not problematic. The main issue is generating each word of length n in "real time", that is, in precisely n steps. This is in stark contrast with derivations in cs grammars. The number of applications of productions in a derivation $S \underset{\Gamma}{\rightarrow}^* u$ can be exponential in the length of the word u, as the next example illustrates.

Example 4.3.

$$\Gamma \begin{cases} S \rightarrow AT & AB \rightarrow AC \quad AC \rightarrow AB \\ T \rightarrow RT \mid BD & BE \rightarrow BF \quad BG \rightarrow BD \\ R \rightarrow BD & CD \rightarrow CE \quad CF \rightarrow CG \\ DC \rightarrow DB & EB \rightarrow EC \quad GC \rightarrow GB \end{cases}$$

There is the derivation $S \underset{\Gamma}{\rightarrow}^* A(BD)^n$ just by applying the S, T and R rules. However, consider the number of steps it takes to replace an occurrence of D with G in $A(BD)^n$. For instance, when $n = 2$, $\boxed{AB}DBD \rightarrow A\boxed{CD}BD \rightarrow AC\boxed{EB}D \rightarrow ACE\boxed{CD} \rightarrow \boxed{AC}ECE \rightarrow A\boxed{BE}CE \rightarrow \boxed{AB}FCE \rightarrow A\boxed{CF}CE \rightarrow ACGCE$. To replace D at position $2n+1$ with G requires 2^n applications of productions involving the initial letter A in the first position, as it requires 2^{n-1} applications of both $AB \rightarrow AC$ and $AC \rightarrow AB$. This also means that the initial A cannot be removed before the end of the computation.

Consequently, we use a more refined characterization of the cs-languages than that they are generable by cs grammars, that is due to Penttonen (Theorem 3 in [11]).

Theorem 4.4 (Penttonen 74). *There is a linear language L_{lin} such that every cs-language K can be represented in the form,*
$K = \left\{ u \in \mathcal{A}^* \mid \exists v \in L_{lin} \ \wedge \ \exists n \in \mathbb{N} \ \wedge \ v \underset{\tau}{\rightarrow}^n u \right\}$, *where τ is a length preserving left cs transformation.*

A *linear language* is generated from a context-free grammar where each production contains at most one nonterminal on its right hand side: for example, if $S \rightarrow aSb, S \rightarrow ab$ then $\mathsf{L}(S)$ is linear. In Theorem 4.4, a length preserving left cs transformation is a set of rewrite rules of the form $ua \rightarrow ub$ where $u \in \mathcal{A}^*$ and $a, b \in \mathcal{A}$. However, we shall work with linear languages whose alphabet is a set of nonterminals that, for us, will be vertex symbols. Clearly, Theorem 4.4 can be easily recast with respect to such an alphabet just by assuming each element of $\mathcal{A}$ is coded as a nonterminal A. However, we also need rules to transform nonterminals into terminals, which will just be of the form $A \to a$. A length preserving left cs transformation τ therefore is a set of productions of the form $UA \to UB$. The relation $\overset{*}{\underset{\tau}{\to}}$, therefore, has the important property that if $U \overset{*}{\underset{\tau}{\to}} V$ then the length of U is the same as the length of V. In fact, the proof of Theorem 4.4 in [11] shows that it is only necessary to consider productions of length at most 2.

Definition 4.5. A *2-left-length-preserving transformation* is a set of productions of the form, $AB \to AC$ and $A \to a$, where $A, B, C \in N$ and $a \in \mathcal{A}$.

It is clear how to define a rational graph whose trace is a linear language, L_{lin}. However, we then need to "filter" it through a 2-left-length-preserving transformation τ. But, there is no obvious transducer associated with τ. In particular, the relation $\underset{\tau}{\to}^*$ may not be rational (that is, definable by a finite transducer), as the following example illustrates.

Example 4.6. If τ is $\{AB \to AC, AC \to AA, CA \to CB\}$, then its rewriting relation is not rational. $\underset{\tau}{\to}^*((AB)^*) \ \cap \ A^*B^* = \{A^nB^m \mid n \geqslant m\}$. $(AB)^*$ is a regular language and, therefore, if $\underset{\tau}{\to}^*$ were rational then $\underset{\tau}{\to}^*((AB)^*)$ and $\{A^nB^m \mid n \geqslant m\}$ would also be regular sets.

A more subtle construction is needed to rationally capture τ. Consider a derivation $U(1)\dots U(n) \underset{\tau}{\to}^* V(1)\dots V(n)$ where $n > 1$ and each $U(i)$ and $V(i)$ is a nonterminal. We can represent such a derivation in the following 2-dimensional form

$$\begin{matrix} U_0(1) & \cdots & U_0(n) \\ U_1(1) & \cdots & U_1(n) \\ \vdots & & \vdots \\ U_m(1) & \cdots & U_m(n) \end{matrix}$$

where $U_0(i) = U(i)$ and $U_m(i) = V(i)$ and $U_i(1)\dots U_i(n) \underset{\tau}{\to} U_{i+1}(1)\dots U_{i+1}(n)$. As noted previously, m can be exponential in n. We wish to capture this derivation in $(n-1)$ steps with a rational relation R_τ in one left to right swoop. First, notice that in the first column there is just one element because $U(1) = V(1)$. In the second column there may be a number of different, and repeating, elements. For instance, if $U_i(2) \neq U_{i+1}(2)$ then $U(1)U_i(2) \underset{\tau}{\to} U(1)U_{i+1}(2)$. Consider the subsequence of this column with initial element $U_0(2)$ and subsequent elements when it changes, $[U_0(2), U_{i_1}(2), \dots, U_{i_{k_2}}(2)]$, so the final element $U_{i_{k_2}}(2)$ is $V(2)$. Next, consider the third column starting with $U_0(3)$. Changes to this element may depend on elements in the second column and not just on $U(2)$. For example $U_{i_j}(2)U_\ell(3) \underset{\tau}{\to} U_{i_j}(2)U_{\ell+1}(3)$. And so on. What we now define is a rational relation R_τ which will include

$$[U(1)]U(2)\dots U(n) \ R_\tau \ [U_1(2)U_{i_1}(2)\dots U_{i_{k_2}}(2)]U(3)\dots U(n)$$

and by composition, it will have the property

$$[U(1)]U(2)\dots U(n)\ R_\tau^{j-1}\ [U_1(j)U_{i_1}(j)\dots U_{i_{k_j}(j)}]U(j+1)\dots U(n)$$

where R_τ^k is the $k$fold composition of R_τ. Consequently, this relation transforms a word of the form $[U]AV$ into a word $[AU']V$ where $U \in N^+$ and $V, U' \in N^*$ and $A \in N$ which says that A may be rewritten to elements of U' in turn, depending on the changing context represented by U. We now define R_τ.

Definition 4.7. If τ is a 2-left-length-preserving transformation and $X = N \cup \{[,]\}$, then $R_\tau \subseteq X^* \times X^*$ is the rational relation recognized by the following transducer T_τ.

$$T_\tau \begin{cases} I \xrightarrow{[X/[A} (A,X,A) & \forall A, X \in N & \text{Type 1} \\ (A,X,Y) \xrightarrow{\varepsilon/Z} (A,X,Z) & \forall A,X,Y,Z \in N \text{ and } XY \underset{\tau}{\rightarrow} XZ & \text{Type 2} \\ (A,X,Y) \xrightarrow{Z/\varepsilon} (A,Z,Y) & \forall A,X,Y,Z \in N & \text{Type 3} \\ (A,X,Y) \xrightarrow{]A/]} F & \forall A,X,Y \in N & \text{Type 4} \\ F \xrightarrow{X/X} F & \forall X \in N & \text{Type 5} \end{cases}$$

States of this transducer are: (X,Y,Z) for all $X,Y,Z \in N$, the initial state I and the final state F.

Example 4.8. Assume $\tau = \{AB \to AC, AC \to AB, CB \to CE, BE \to BF\}$.

$$[A]B\ R_\tau\,[BCB]\colon \begin{array}{r|c|} A & B \\ A & C \\ A & B \end{array} \qquad [BCB]BAB\ R_\tau\,[BEF]AB\colon \begin{array}{r|c|l} AB & B & AB \\ AC & B & AB \\ AC & E & AB \\ AB & E & AB \\ AB & F & AB \end{array}$$

The relation R_τ underlies a rational graph whose trace is the corresponding language. Before proving this, we need a technical lemma.

Lemma 4.9. *If U, $V \in N^*$ and $|U| = |V| = n$, then the following statements are equivalent.*

(i) $U \underset{\tau}{\rightarrow}^* V$, *using only productions of the form $AB \to AC \in \tau$*
(ii) $[U(1)]U(2)\cdots U(n)\ {R_\tau}^{n-1}[U(n)WV(n)]$, *for some W in N^**

Proof. (*Sketch*) Let U and V belong to N^*, and assume that they have equal length n. Moreover, assume that $\{r_1, r_2, \cdots, r_k\}$ are all the productions in τ of the form $AB \to AC$. First, we show $(i) \Rightarrow (ii)$. If $U \underset{\tau}{\rightarrow}^* V$, then there exists $m \geqslant 1$ such that $U = U_1 \underset{r_{\ell_1}}{\rightarrow} U_2 \cdots \underset{r_{\ell_{m-1}}}{\rightarrow} U_m = V$. For each j between 1 and $m-1$ assume that I_j is the set $\{i \mid U_i(j) \neq U_{i+1}(j)\}$: that is, it is the set of indices i such that the rule r_{ℓ_i} of τ changes the jth letter of U_i. Notice that $I_1 = \emptyset$, and if $i \neq j$ then $I_i \cap I_j = \emptyset$ and, moreover, $\bigcup_{j=1}^n I_j = [m]$. The following holds, for all $j \geq 2$.

$$[U_1(j-1) \prod_{i \in I_{j-1}} U_{i+1}(j-1)]U_1(j)\ R_\tau\,[U_1(j) \prod_{i \in I_j} U_{i+1}(j)]$$

Therefore, there is a path in T_τ which starts in I and ends in F that is labelled $[U_1(j-1)\prod_{i\in I_{j-1}} U_{i+1}(j-1)]U_1(j)$ on the left and $[U_1(j)\prod_{i\in I_j} U_{i+1}(j)]$ on the right. By Definition 4.7, therefore, there is the following transition in T_τ.

$$I \xrightarrow{[U_1(j-1)/[U_1(j)} (U_1(j), U_1(j-1), U_1(j))$$

Using a type 4 transition, there is a transition to the final state.

$$(U_m(j), U_m(j-1), U_m(j)) \xrightarrow{]U_m(j)/]} F$$

Thus, for all $j \geqslant 2$

$$[U_1(j-1)\textstyle\prod_{i\in I_{j-1}} U_{i+1}(j-1)]U_1(j)\ R_\tau\ [U_1(j)\prod_{i\in I_j} U_{i+1}(j)]$$

By induction on j it follows that

$$[U(1)]U(2)\cdots U(n)\ {R_\tau}^{j-1}\ [U_1(j)\prod_{i\in I_j} U_{i+1}(j)]U_1(j+1)\cdots U_1(n)$$

The proof $(ii) \Rightarrow (i)$ is easier. Associated with any $k$fold composition of R_τ is a sequence of applications of productions of τ. $\square$

This lemma is used in the proof of the next result.

Proposition 4.10. *Cs-languages are traces of rational graphs.*

Proof. Assume K is a cs-language, and let τ be a 2-left-length-preserving transformation, by Theorem 4.4, such that $\underset{\tau}{\rightarrow}^*(L_{lin}) = K$. We use the relation R_τ of Definition 4.7 to construct a rational graph. For each letter a in $\mathcal{A}$, N_a is the set of nonterminals A such that $A \underset{\tau}{\rightarrow} a$. Let R_a be the relation $R_\tau \cap [N^*N_a]N^+ \times [N^*]N^*$ which is rational, see [2]. Therefore, the following graph G_0 is rational, $G_0 = \bigcup_{a\in\mathcal{A}} \{(x,a,y) \mid (x,y) \in R_a\}$. Therefore, the following graph G_1 is also rational.

$$G_1 = G_0 \cup \bigcup_{a\in\mathcal{A}} [N^*N_a] \times \{a\} \times \{\varepsilon\}$$

Let $[L_{lin}]$ be the language $\{[A]U \mid A \in N \wedge U \in N^* \wedge AU \in L_{lin}\}$. Therefore, $L(G_1, [L_{lin}], \{\varepsilon\}) = K$.

$$\begin{array}{rl}
& u \in L(G_1, [L_{lin}], \{\varepsilon\}), \text{and the length of } u \text{ is } n \\
\Leftrightarrow & [V(1)]V(2)\cdots V(n) \xrightarrow[G_1]{u} \varepsilon \\
\Leftrightarrow & [V(1)]V(2)\cdots V(n)\ R_{u(1)}\,[V(2)\cdots V'(2)]V(3)\cdots V(n)\ R_{u(2)}\cdots \\
& \cdots\ R_{u(n-1)}[V(n)\cdots V'(n)]\ R_{u(n)}\,\varepsilon \\
\overset{\text{Lemma 4.9}}{\Leftrightarrow} & V(1)V(2)\cdots V(n) \underset{\tau}{\rightarrow}^* V(1)V'(2)\cdots V'(n) \\
& \text{and } V(1) \underset{\tau}{\rightarrow} u(1) \wedge \forall i \in [2\ldots n],\ V'(i) \underset{\tau}{\rightarrow} u(i) \\
\Leftrightarrow & u \in K
\end{array}$$

The problem is now that $[L_{lin}]$ is not a regular language. To get a proper converse of Proposition 3.3 we need to find a rational graph whose trace between two regular sets is K. However, $[L_{lin}]$ is generable by a context-free grammar Γ in Greibach normal form, where every production has a single nonterminal (belonging to N) on its right hand side. The relation R_τ transforms a word in L_{lin} from left to right, and therefore G_1 can be transformed into G in such a way that it starts with extended words in Γ that begin with precisely two terminal letters. Let I be this finite set of words. The transducer generating G will also apply productions of Γ to obtain more letters of N so that the computation continues. Consequently, it now follows that $K = L(G, I, \{\varepsilon\})$. □

The proof of Proposition 4.10 constructs a graph that may have infinite degree. However, there can only be infinite degree if there are cycles in the transformation τ. It is possible to remove such cycles: for any 2-left-length-preserving transformation τ, there is an equivalent 2-left-length-preserving transformation τ' which is acyclic. Therefore, it follows that for any cs-language there is a rational graph of finite, but not necessarily bounded, degree whose trace coincides with the language.

Proposition 4.11. *If $K \subseteq \mathcal{A}^*$ is a cs-language, then there is a rational graph G of finite degree such that $K = L(G, I, F)$ where I and F are finite sets of vertices.*

Combining Propositions 3.3 and 4.10, we thereby obtain the main result of the paper.

Theorem 4.12. *Traces of rational graphs are all the cs-languages.*

5 Conclusion

We have shown that the traces of rational graphs relative to regular initial and final sets of vertices coincide with the cs-languages. This new characterization of the cs-languages may offer new insights into these languages. For instance, an interesting question is: is it possible to define a subfamily of rational graphs whose traces are the deterministic cs-languages?

Acknowledgements

The authors are grateful to Thierry Cachat, Didier Caucal and Jean-Claude Raoult for their helpful comments on this paper.

References

1. Autebert, J.-M. and Boasson, L. *Transductions rationelles.* Masson, 1988.
2. Berstel, J. *Transductions and context-free languages.* Teubner, 1979.
3. Moller, F. Burkart, O, Caucal, D. and Steffen B. *Handbook of Process Algebra*, chapter Verification on infinite structures, pages 545–623. Elsevier, 2001.

4. Caucal, D. On transition graphs having a decidable monadic theory. In Icalp 96, volume 1099 of *LNCS*, pages 194–205, 1996.
5. Courcelle, B. *Handbook of Theoretical Computer Science*, chapter Graph rewriting: an algebraic and logic approach. Elsevier, 1990.
6. Ginsburg, S. and Rose, G.F. Preservation of languages by transducers. *Information and control*, 9:153–176, 1966.
7. Knapik, T. and Payet, E. Synchronization product of linear bounded machines. In *FCT*, volume 1684 of *LNCS*, pages 362–373, 1999.
8. Mateescu, A, Salomaa, A. *Handbook of Formal Languages*, volume 1, chapter Aspects of classical language theory, pages 175–252. Springer-Verlag, 1997.
9. Morvan, C. On rational graphs. In Fossacs 00, volume 1784 of *LNCS*, pages 252–266, 2000.
10. Muller, D. and Schupp, P. The theory of ends, pushdown automata, and second-order logic. *Theoretical Computer Science*, 37:51–75, 1985.
11. Penttonen, M. One-sided and two-sided context in formal grammars. *Information and Control*, 25:371–392, 1974.