Rational tree relations

Jean-Claude Raoult

Abstract

We investigate forests and relations on trees generated by grammars in which the non-terminals represent relations. This introduces some synchronization between productions. We show that these sets are also solutions of systems of equations, that they are described by rational expressions involving union, substitution and iterated substitution, and that they are preserved by residuals. We show that they are the images of k-copying descending transducers. Finally, we isolate a subset of these relations which is preserved by composition.

1 Introduction

The rational transductions over a free monoid are relations satisfying several good properties:

1. The identity I, the converse R^{-1} and the (associative) composition of rational transductions are again rational transductions.

2. The image of a rational language is again a rational language. Together with (1), this implies that the domain and range of a rational transduction are rational.

3. The membership relation $(x, y) \in R$ is decidable.

Received by the editors May 95.
Communicated by M. Boffa.
1991 *Mathematics Subject Classification :* 68Q42, 68Q45, 68Q50.
Key words and phrases : Tree automata, rationality, transductions, graph grammars.

4. They accept several equivalent definitions: finite automata with output, grammars of pairs of words, rational subsets of the square of the monoid, bimorphisms.

All these properties have been known long ago, and are used in several areas of theoretical computer science. The definitions by grammars, or by bimorphisms, are adapted to proving further properties. The definition using automata is used in every lexical analyzer.

In the case of trees, the situation is not quite so neat (see Raoult [1992]). Surely rational sets of trees are well defined and well-known (see Gecseg & Steinby [1984]): a typical instance of a rational forest is the set of parse trees of a context-free grammar. Also, plenty of definitions exist for finite state tree transformations. Most of them use finite mechanical devices: finite automata with output. Engelfriet [1975] sorts some of them into top-down or bottom-up tree transformations, which are incomparable in power, and yield by composition an infinite hierarchy. Vogler [1987] allows the automaton to use a pushdown stack, thus extending its power and getting an infinite hierarchy of tree transductions. Engelfriet & Vogler [1991] extend further the definition to "modular tree transducers" which compute all the primitive recursive functions on trees. These same functions had already been obtained by Courcelle & Franchi-Zannettacci [1982] using strongly non circular attribute grammars.

Restricting the power of transformations instead of extending it, Dauchet et alii [1987] define ground tree transducers by the action of two finite automata, one in the domain and one in the range. The relations computed in this way are preserved by a number of operations, composition and iteration included. These relations are essentially the same as the "regular bi-languages" of Pair & Quéré [1968]. They are too particular to coincide with word transductions, when they are restricted to monoids. A nice generalization by Dauchet & Tison [1992] of these ground tree transducers, in which a single finite automaton runs on a superposition of the input and output trees, does extend the word transductions to the case of trees. Nevertheless they leave out cases like the one shown in Fig. 1, given in Arnold & Dauchet [1982], which can be obtained with an automaton with output.

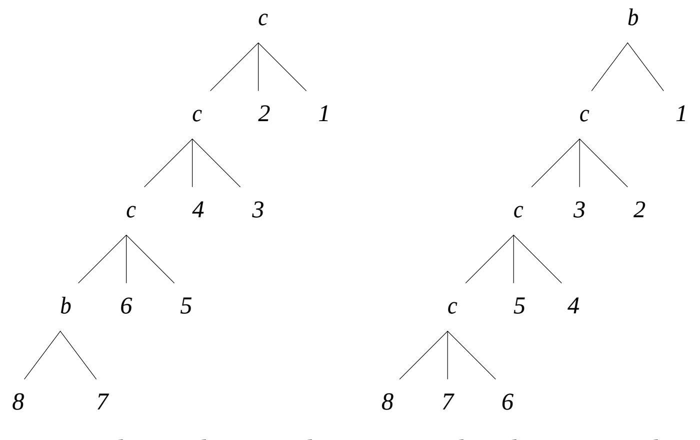

Figure 1: Equal numbers indicate equal subtrees. The relation made of all these couples is not a transduction in the sense of Dauchet and Tison [1992].

Another definition due to Arnold & Dauchet [1982] uses bimorphisms and coincides with non-erasing transductions when restricted to words. For instance, the transformation of Figure 1 is definable in this way. Actually, Dauchet defines in his thesis [1977] a very general form of tree transductions using two bimorphisms (i.e. four morphisms), which coincides with rational word transductions when restricted to words. We propose here yet another definition, the right one of course, which is akin to Schreiber's syntax connected transductions [1975]. Our definition mimics the definition of word transductions, as in Berstel [1979] for instance. The basic idea is to define tree relations recursively by tree grammars, but introducing some synchronization between the productions of several non-terminals.

Example 1.1 For instance, the relation, say $B(x, y)$, depicted in Figure 1 can be defined in a Prolog-like form:

$$\begin{array}{rcl} B(x,y) & \Leftarrow & x = y \vee y = b(u,v) \wedge C(x,u,v) \\ C(x,y,z) & \Leftarrow & x = c(u,v,w) \wedge y = c(u',v',w') \wedge C(u,u',v') \\ & & \wedge v = w' \wedge w = z \\ & \vee & x = b(u,v) \wedge u = y \wedge v = z \end{array}$$

The resulting transductions are the same as in Dauchet's thesis, but we define them more algebraically and, we hope, more intuitively.

Section 2 reviews the notation for trees, tuples of trees and graftings and defines the topic of the paper: relations defined by grammars. In section 3, we show that these relations can also be described by rational (regular) expressions, and derive a pumping lemma. Nevertheless, unary relations (sets of trees) are more general than rational forests. In section 4, we show that these equational relations are also images of k-copying transducers. In section 5 a condition of desynchronization ensures the stability under composition in such a way that the corresponding "rational transductions" extend the rational transductions of words; this condition is shown to be decidable on the grammar.

2 Grammars for relations

The definition in example 1 can be rewritten in order to stress the "tree-like" aspect of the rules:

$$\left\{ \begin{array}{rcl} B_1 B_2 & \longrightarrow & C_1\, b(C_2\, C_3) \ /\ I_1 I_2 \\ C_1 C_2 C_3 & \longrightarrow & c(C_1\, I_1\, I_2')\, c(C_2\, C_3\, I_2)\, I_2' \ /\ b(I_1\, I_1')\, I_2\, I_2' \\ I_1 I_2 & \longrightarrow & a(I_1)\, a(I_2) \ /\ e\, e. \end{array} \right.$$

The definitions below describe this notation.

Definition 2.1 Let F be an alphabet, or as the tradition has it, a set of "function symbols" and X a denumerable set of "variables". The set T^* of tuples of trees (or terms) over F and X is defined recursively by the following equation:

$$T^* = \{\varepsilon\} + X + FT^* + T^*T^*$$

where the concatenation of sets denotes their cartesian product as is customary in formal language theory, and is associative.

When F and X are not clear from the context, they are specified in the notation: $T(F,X)^*$ instead of T^*. The elements of $X + FT^* = T$ are trees: $t = x$ or $t = f(w)$; those of $\{\varepsilon\} + T^*T^*$ are tuples of trees (possibly reduced to a single tree). And T^* is indeed the free monoid generated by T: $T^* = \sum_{n\geqslant 0} T^n$. It is easy to prove (see next proposition), and it is essentially done in Pair & Quéré [1968], that $T(F,X)^*$ is the free monoid over X with operators in F: this structure has two operations: one binary and associative with a unit element (a monoid) and one for "multiplying" an element w of the monoid by an operator f in F, yielding a new element $f(w)$ of the monoid. In the case of the free object T^*, this multiplication consists in adjoining a "root" labelled in F to a tuple $t_1 \ldots t_n$ ($n \geqslant 0$), yielding a tree $f(t_1 \ldots t_n)$.

A graded version is frequently used in conjunction with a representation of composed functions: the set F of operators is graded: $F = \sum_{n\in\mathbb{N}} F_n$. The free monoid T^* is also graded by the length of its elements, the tuple $t_1 \ldots t_n$ being of degree, or length, n. Then all trees $t = f(w)$ are subject to the restriction that the length n of w equals the degree of the operator f: $f(w) \in T \Rightarrow (|w| = n \iff f \in F_n)$. In this context, tuples are sometimes denoted with parentheses and commas when their elements are trees: $(t_1, \ldots, t_n)$ and $f(t_1, \ldots, t_n)$. We denote by $X(w)$ the set of variables of X occurring in the tuple w. This set is ordered by the first occurrence of each variable when read from left to right.

The universal property of T^* can be stated as follows

Proposition 2.2 *Every mapping φ of X into a monoid M with operators in F can be extended uniquely into a morphism of monoids with operators called again $\varphi : T(F,X)^* \to M$.*

Proof. The morphism φ is already defined over X and must satisfy the following constraints, because it is a morphism.

$$\begin{aligned}\varphi(\varepsilon) &= 1_M,\\ \varphi(uv) &= \varphi(u)\varphi(v),\\ \varphi(f(u)) &= f(\varphi(u)).\end{aligned}$$

Conversely $\varphi(w)$ is uniquely defined by structural induction on w by these rules. ■

In particular, let $\alpha : X \to X$ be a bijective mapping. Then the generated mapping $\alpha : T^* \to T^*$ is called a variable renaming. The existence of a variable renaming such that $w' = \alpha(w)$ is an equivalence relation over T^*. In this paper, we shall group the variables of a tuple into sequences and manipulate a sequence globally. Variable renaming will come in a different flavour to reflect this fact.

Definition 2.3 Given a set X of variables, a non empty sequence of distinct variables in X^+ is called a multivariable. If $\mathbf{A} = A_1 \ldots A_n$ ($n > 0$), the multivariable $\mathbf{A}$ has length n. The set of instances of variables (resp. multivariables) is the cartesian product $X\omega$ (resp. $X^+\omega$) where ω is the set of natural numbers. More precisely, (A, j) is the j-th instance of A, which we shall rather denote by A^j.

Note for instance that $\mathbf{I}^3 = I_1^3 I_2^3$ is the third instance of the multivariable $\mathbf{I} = I_1 I_2$, but $I_1^2 I_2^3$ is not an instance of $\mathbf{I}$. Usually, $\mathbf{A}^0$, $\mathbf{A}^1$ and $\mathbf{A}^2$ will be written $\mathbf{A}$, $\mathbf{A}'$ and $\mathbf{A}''$. A bijection $\omega \longrightarrow \omega$ generates a bijection of $T(F, X\omega)^*$ upon itself, called a renaming of the instances. Two tuples will be considered equal if they correspond by a renaming of instances. All tuples considered in this article will be linear: the same instance of the same variable cannot occur twice in a tuple. Instances make it possible to relate variables belonging to the same multivariable. For instance, $b(I_1\,I_1')\,b(I_2\,I_2')$ contains two instances of the multivariable $\mathbf{I}$: I_1I_2 and $I_1'I_2'$; but I_1I_2' is not an instance of $\mathbf{I}$.

A tuple in $T(F, X\omega)^+$ can be drawn as a tuple of trees together with a hyperarc passing through A_1^k, ..., A_n^k in this order and labelled by $\mathbf{A}$. Thus the tuple

$$\alpha = b(b(I_1\,I_1')\,I_1'')\,b(I_2\,b(I_2'\,I_2''))$$

is drawn in Figure 2.

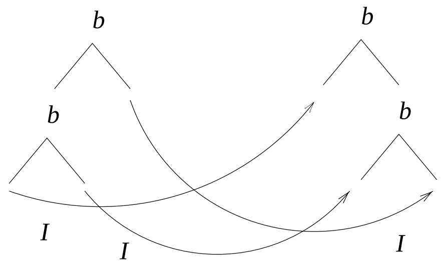

Figure 2: A graphical representation of $b(b(I_1\,I_1')\,I_1'')\,b(I_2\,b(I_2'\,I_2''))$.

Another application of the universal property: a mapping $\sigma : X \to T^*$ extends uniquely into a morphism $\sigma : T^* \to T^*$ and is called a substitution if $\sigma(x) = x$ for all but a finite number of variables. If $\sigma(x_1) = v_1, \ldots, \sigma(x_n) = v_n$ and $\sigma(x) = x$ for all other variables x, the substitution σ is denoted by $\sigma = [v_1/x_1, \ldots, v_n/x_n] = [\mathbf{v}/\mathbf{x}]$ where $\mathbf{v} = v_1 \ldots v_n$ and $\mathbf{x} = x_1 \ldots x_n$. It is usually postfixed: $\mathbf{u}[v_1/x_1, \ldots, v_n/x_n] = \mathbf{u}\sigma$ instead of $\sigma(u)$. In this article, we only consider the graded versions of F and T^* and therefore we shall restrict to the case where $x\sigma$ is a single tree (not a general tuple) for all variables x in X. Since we restrict attention to linear tuples, the substitution will be modified to become an internal operation: when we consider $u[v/\mathbf{x}]$ it will be understood that the representative of v has been chosen such that the instances of variables of v have been chosen greater, say, than all instances of variables in u. It is easy to show that it is unique (up to renaming of its instances of variables):

Proposition 2.4 *Let u and v be two tuples of trees, let $\mathbf{x}$ be a multivariable, and let α be a variable renaming. Suppose $X(u) \cap X(v) = \phi$ and $X(u) \cap X(\alpha(v)) = \phi$. Then $u[\alpha(v)/\mathbf{x}] = \alpha(u[v/\mathbf{x}])$.*

This is a well-known property of term substitution when the variable renaming α restricted to the set $X(u)$ of variables of u is the identity, and it does not even require the linearity of u.

As a first result, if u and v are linear tuples, then $u[v/\mathbf{x}]$ is also linear.

Definition 2.5 Given a set X of variables, a production is a pair $(\mathbf{A}, \alpha)$ in $X^+ \times T(F, X\omega)^+$ in which both sides have same length ($|\mathbf{A}| = |\alpha|$) and in which $\mathbf{A}$ and α are linear. The multivariable $\mathbf{A}$ is the left-hand side and α the right-hand side.

A grammar is a finite set of productions in which the left-hand sides are equal or disjoint and in which the instances of variables occurring in α can be grouped to form instances of left-hand sides of the grammar: if A_i^k occurs in α and $A_1 \dots A_n$ is a left-hand side then $A_1^k, \dots, A_n^k$ occur in α.

Example 2.6 The following grammar is made of three productions for the non-terminal A_1A_2, and represents a right rotation of AVL-trees (the first right-hand side is drawn in Figure 2:

$$\begin{cases} p: & A_1A_2 & \longrightarrow & b(b(I_1\, I_1')\, I_1'')\, b(I_2\, b(I_2'\, I_2'')) \\ q: & I_1I_2 & \longrightarrow & a(I_1)\, a(I_2) \\ r: & I_1I_2 & \longrightarrow & e\, e. \end{cases}$$

Definition 2.7 The step of derivation generated by a grammar G is defined as follows

$$\beta \xrightarrow[G]{} \beta[\alpha/\mathbf{A}^j]$$

if $(\mathbf{A}, \alpha)$ is a production of G and $\mathbf{A}^j$ is an instance $\mathbf{A}$ in β. A derivation is a sequence of steps of derivation, possibly empty.

As a basic case $A_1 \dots A_n \xrightarrow[G]{} \alpha$ for $(A_1 \dots A_n, \alpha) \in G$. This accounts for the fact that productions will be denoted as single steps of derivations: $A_1 \dots A_n \xrightarrow[G]{} \alpha$ or $\mathbf{A} \xrightarrow[G]{} \alpha$. The subscript G will be omitted when the context makes it clear. Intuitively, the variables $A_1, \dots, A_n$ cannot be derived independently, but only synchronously.

Note that if β and α are replaced by equal tuples (i.e. differing only by a renaming of instances), the result $\beta[\alpha/\mathbf{A}^j]$ changes only by a renaming of instances, and thus represents an equal tuple. Finally note also that the derivation is an operation preserving the length, so that all the tuples derived from a given tuple have the same length.

We shall follow the standard terminology of formal language theory and call the left-hand sides of the grammar *non-terminals*. Thus the set $\mathcal{N}$ of non-terminals is a finite subset of X^+. The language generated by a grammar starting from an axiom α is

$$\begin{aligned} L(G,\alpha) \quad = \quad & \{w \in T(F, X\omega)^+;\ \alpha \xrightarrow[G]{}{}^* w \\ & \text{and non-terminals have no instances in } w\}. \end{aligned}$$

Parse trees can be defined as follows.

Definition 2.8 Associate with each grammar G a tree grammar in the following way. With each non-terminal $\mathbf{A}$ in G associate a unary non-terminal $X_{\mathbf{A}}$ and for each production $A \xrightarrow[G]{} \alpha$ order arbitrarily the instances of non-terminals occurring in α intothe sequence $\mathbf{B}_1\ \dots\ \mathbf{B}_m$. Associate with production p the tree production

$$X_{\mathbf{A}} \longrightarrow p(X_{\mathbf{B}}^1\ \dots\ X_{\mathbf{B}}^m).$$

A parse tree for G is a tree generated by the tree grammar associated in this way with G. The yield of a parse tree t is the tuple denoted by $Y(t)$ and defined recursively:

1. if $t = X_{\mathbf{B}}$ then $Y(t) = \mathbf{B} = B_1 \dots B_p$;

2. if $t = p(t_1, \dots, t_m)$ then $Y(t) = \alpha[Y(t_1)/\mathbf{B}_1, \dots, Y(t_m)/\mathbf{B}_m]$ with the above notations for p.

Beware that the ordering of the instances of non-terminals in each right-hand side, albeit arbitrary, in needed to define the yield. For instance, consider the grammar given in example 2.6. The corresponding tree grammar is (for simplicity, we set $X = X_{\mathbf{A}}$ and $Y = X_{\mathbf{I}}$):

$$\begin{cases} X \longrightarrow p(Y\,Y\,Y) \\ Y \longrightarrow q(Y) \\ Y \longrightarrow r \end{cases}$$

where the first, second and third arguments of p correspond to $\mathbf{I}$, $\mathbf{I}'$ and $\mathbf{I}''$ respectively. Then $b(b(a(e)\,e)\,e)\,b(a(e)\,b(e\,e))$ belongs to $L(G, A)$ with parse tree $p(q(r)\,r\,r)$ (see figure 3 below). If the instances of non-terminals in the first production are ordered with the reverse ordering, then the same parse tree has a different yield: $b(b(e\,e)\,a(e))\,b(e\,b(e\,a(e)))$.

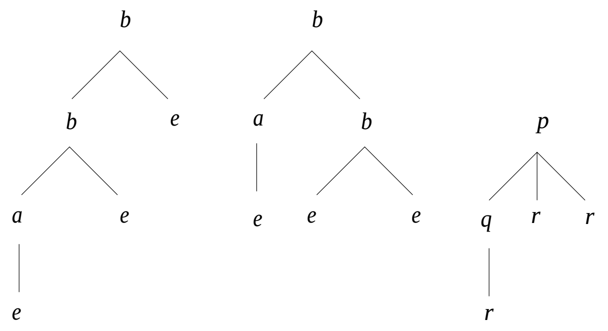

Figure 3: A couple generated by the grammar of example 2.6 and its parse tree.

Of course, the grammar of parse trees is related to the original grammar by the following expected result.

Proposition 2.9 *The language generated by a grammar is equal to the set of yields of all parse trees generated by the associated tree grammar.*

Proof. By induction on the number of derivation steps, left to the reader. ∎

What properties do we expect of these relations? To begin with, grammars may be viewed as a systems of equations, as for context-free languages. The languages generated by the grammar are the least solutions of the system. This is a corollary of Habel's and Kreowski's work [1987] on hyperedge replacement systems (each instance $\mathbf{A}^j$ of the non-terminal $\mathbf{A}$ can be considered as a hyperedge labelled $\mathbf{A}$); it could be rewritten in the framework of tuples of trees. Grammars admit also an equivalent Greibach form: define the size $\|w\|_F$ or simply $\|w\|$ of a tuple of trees w as the number of vertices labelled by function symbols. A grammar is in Greibach normal form when each right-hand member, each production, has size one. Note that this definition is different from Engelfriet's definition [1992]: adapted to our context, Engelfriet's requirement is that all the roots at the right-hand sides are labelled by terminal function symbols. This is a much stronger condition than the existence of one root labelled by a terminal symbol. Our property being simpler, it is not surprising that the proof that every grammar admits an equivalent Greibach form is consequently shorter.

Lemma 2.10 *Given a grammar, one can effectively deduce a grammar with the same non-terminals, generating the same languages from the same non-terminals and in which all right-hand sides have positive sizes.*

Proof. In a first step, determine, for all non-terminals $\mathbf{A}$, all the tuples α of size 0 (and of same length n as $\mathbf{A}$) such that $\mathbf{A} \rightarrow^+ \alpha$. Such tuples belong to $(X\omega)^n$ (modulo variable renaming) and are therefore in finite number. In a second step, for all production $\mathbf{B} \longrightarrow \beta$ of G, for all instance $\mathbf{A}^j$ of $\mathbf{A}$ in β and for all α in $X\omega^n$ such that $\mathbf{A} \rightarrow^+ \alpha$ (this has been determined in the first step), add to the grammar the production $\mathbf{B} \longrightarrow \beta[\alpha/\mathbf{A}]$. Then remove all the productions of size zero. The productions of the resulting grammar are derivations of G and generate the same sets of tuples from the same axioms, as in the case of ordinary context-free grammars. ■

Proposition 2.11 *Given a grammar, one can effectively deduce a Greibach grammar with an extended set of non-terminals, that generates the same languages from the same non-terminals.*

Proof. We may assume from lemma 2.10 that the grammar G contains only productions of positive size. Consider a production of size greater than one:

$$p : \mathbf{A} \underset{G}{\longrightarrow} \alpha$$

with $\alpha = \beta\, f(\gamma)\, \delta$ where β, γ and δ are tuples of lengths a, b and c respectively. Take a new non-terminal $\mathbf{X}$ of length $a+b+c$ and replace production p by the two productions

$$\begin{cases} q : \mathbf{A} \underset{H}{\longrightarrow} X_1 \ldots X_a f(X_{a+1} \,\ldots\, X_{a+b}) X_{a+b+1} \ldots X_{a+b+c} \\ r : \mathbf{X} \underset{H}{\longrightarrow} \beta\gamma\delta \end{cases}$$

of sizes 1 and $\|\alpha\| - 1$. The non-terminals of G are also non-terminals of H and H generates the same sets of terminal tuples from the same axioms as G. Indeed, in one direction it suffices to decompose every step of derivation p into the sequence qr. For the converse, suppose that q appears in a derivation from a non-terminal of G: since the only production of $\mathbf{X}$ is r, this derivation must contain production r. Modulo reordering, one may assume that r follows q (because this is true for the grammar of parse trees). Then the succession of q and r can be replaced by p. This ends the proof by induction on $\sum_{A \longrightarrow \alpha} \|\alpha\| - 1$. ■

The number of productions added in H during the third step is

$$\sum_{(X \longrightarrow \alpha) \in G} \|\alpha\| - 1.$$

Corollary 2.12 *One can decide whether a given tuple is generated by a given grammar starting from a given non-terminal: the languages generated by our grammars are recursive.*

Proof. A tuple of size n is the result of a derivation of length n if the grammar is in Greibach normal form. It is enough to check all derivations of length n. ∎

3 Grammars generate rational languages

Three simple operations preserve the languages generated by our grammars: concatenation, union and projection. But note that the union is restricted: all tuples of a generated language have the same length, that of the chosen axiom. This property must be invariant.

Proposition 3.1 *The languages generated by grammars are closed by concatenations and by unions when languages are made of tuples of same length.*

Proof. Without loss of generality, we suppose two grammars G and G' having disjoint sets of non-terminals and axioms $\mathbf{S}$ and $\mathbf{S}'$ respectively. Let $\mathbf{U}$ be a new non-terminal of length $|\mathbf{S}|+|\mathbf{S}'|$. Then the grammar $\{\mathbf{U} \to \mathbf{S}\mathbf{S}'\} \cup G \cup G'$ generates from $\mathbf{U}$ the language $L(G,\mathbf{S})L(G',\mathbf{S}')$. And when $|\mathbf{S}| = |\mathbf{S}'|$ let $\mathbf{V}$ be a new non-terminal of length $|\mathbf{V}| = |\mathbf{S}| = |\mathbf{S}'|$. Then the grammar $\{\mathbf{V} \to \mathbf{S}, \mathbf{V} \to \mathbf{S}'\} \cup G \cup G'$ generates from $\mathbf{V}$ the language $L(G,\mathbf{S}) \cup L(G',\mathbf{S}')$. Proof clear.

Proposition 3.2 *The languages generated by grammars are closed by projections. More precisely, given a grammar G one can deduce a grammar G', called the grammar of projections, such that for all tuples α and β of length n and all subset I of $[1,n]$*

$$\alpha \underset{G}{\longrightarrow} \beta \iff (\mathrm{pr}_I\alpha = \mathrm{pr}_I\beta \text{ or } \mathrm{pr}_I\alpha \underset{G'}{\longrightarrow} \mathrm{pr}_I\beta).$$

Proof. Define the grammar G' as follows. For all non-terminals $\mathbf{A}$ of G and all non-empty subsets $I = \{i_1, \dots, i_p\}$ of $[1, |\mathbf{A}|]$, introduce new non-terminals $\mathbf{A}_I$ with a renaming of variables $A_{I,1} \dots A_{I,p} = A_{i_1} \dots A_{i_p}$. Thus the non-terminals $\mathbf{A}_I$ and $\mathbf{A}_J$ are disjoint even if I and J are not disjoint. The productions of $\mathbf{A}_I$ are the projections on I of the productions of $\mathbf{A}$: one gets them from the productions of $\mathbf{A}$ by erasing all trees of the right-hand side the ranks of which are not in I:

$$\mathbf{A} \underset{G}{\longrightarrow} \alpha \Rightarrow \mathbf{A}_I \underset{G'}{\longrightarrow} \mathrm{pr}_I\alpha$$

where $\mathrm{pr}_I\alpha = t_{i_1} \dots t_{i_p}$ if $\alpha = t_1 \dots t_n$ and $I = \{i_1, \dots, i_p\}$ $(i_1 < \dots < i_p)$ and in which all partially erased instances $B^k_{j_1} \dots B^k_{j_q}$ of non-terminals $\mathbf{B}$ have been renamed as instances of $\mathbf{B}_J$: $B^k_{J,1} \dots B^k_{J,q}$ with $J = \{j_1, \dots, j_q\}$.

Consider now one step of derivation for G: $\beta \underset{G}{\longrightarrow} \gamma = \beta[\alpha/\mathbf{A}^j]$ such that $(\mathbf{A} \to \alpha) \in G$ and $|\beta| = m$, and let J be a non empty subset of $[1,m]$. Then let I be the subset of $[1,|\mathbf{A}|]$ such that A^j_i occurs in $\mathrm{pr}_J\beta$ if and only if $i \in I$. There are two cases: if $I = \phi$ there is no variable of $\mathbf{A}^j$ remaining in the projection, and

$\mathrm{pr}_J\beta = \mathrm{pr}_J\gamma$. If $I \neq \phi$ then there is a production in G': $\mathbf{A}_I \longrightarrow \mathrm{pr}_I\alpha$, and therefore there is a step of derivation under G':

$$\mathrm{pr}_J\beta \underset{G'}{\longrightarrow} (\mathrm{pr}_J\beta)[\mathrm{pr}_I\alpha/\mathbf{A}_I^j] = \mathrm{pr}_J\gamma.$$

Conversely, the definition of G' shows that every step of derivation for G' is the projection of a derivation step for G. ■

Consequently, 1) the projection on I of a derivation of a tuple in $L(G,\alpha)$ is a derivation of a tuple in $L(G',\mathrm{pr}_I\alpha)$, for all α; and 2) conversely every derivation of a tuple u in $L(G',\mathrm{pr}_I\alpha)$ is the projection of a derivation of a tuple β in $L(G,\alpha)$. If all non-terminals in G can derive terminal tuples (G is "reduced", and this assumption does not restrict the generated languages), then $\beta \underset{G}{\longrightarrow}{}^* w$ and $u = \mathrm{pr}_I w$.

In free algebras, grammars generate subsets that are solutions of polynomial equations (see Mezei & Wright [1967] for instance, where equationality is called algebraicity). These solutions also have "rational" (regular) descriptions. Rationality (regularity) is well-known: it means stability under the union (denoted by $+$), under some sort of binary product and under its iteration. In the case of words, this binary product is the concatenation. In the case of free algebras, an analogue of concatenation might be any given binary operator (not even associative: see Steinby [1984]).

In the case of trees, though, the operation that corresponds most closely to concatenation is tree substitution. It is also associative, and coincides with concatenation when words are represented by filiform trees. To study this operation, we define an operation $L \cdot_{\mathbf{A}} L'$ extending the substitution as follows.

Notation 3.3 Given a tuple u in $T(F, X\omega)^*$ containing k instances $\mathbf{A}^1, \ldots, \mathbf{A}^k$ of a multivariable $\mathbf{A} = A_1 \ldots A_n$ in X^+ and a language $L \subseteq T(F, X\omega)^n$ we define the language $u \cdot_{\mathbf{A}} L$ as the set

$$u \cdot_{\mathbf{A}} L = \{u[v_1/\mathbf{A}^1, \ldots, v_k/\mathbf{A}^k];\ v_1, \ldots, v_k \in L\}.$$

And given $L' \subseteq T(F, X\omega)^*$, we define by additive extension

$$L' \cdot_{\mathbf{A}} L = \bigcup_{u \in L'} u \cdot_{\mathbf{A}} L.$$

Remark that different instances of the same multivariable may be replaced by possibly different tuples from L.

Example 3.4 For instance if $L = \{a^n(e)\, a^n(e);\ n \geqslant 0\}$ and $I = I_1 I_2$ then

$$b(b(I_1\, I_1')\, b(I_2\, I_2')) \cdot_I L = \{b(b(a^n(e)\, a^m(e))\, b(a^n(e)\, a^m(e)));\ m, n \geqslant 0\}.$$

Note also that $u \cdot_{\mathbf{A}} L$ is the language generated from the axiom u by the (infinite) grammar $\{A \to v;\ v \in L\}$ in the case where $\mathbf{A}$ has no instance in the tuples of L. As a consequence, assuming that $\mathbf{B}$ is a multivariable having no instance in u and L' is a language of tuples of length $|\mathbf{B}|$, the following associativity relation holds

$$(u \cdot_{\mathbf{A}} L) \cdot_{\mathbf{B}} L' = u \cdot_{\mathbf{A}} (L \cdot_{\mathbf{B}} L').$$

Notation 3.5 Given $L \subseteq T(F, X\omega)^n$ and a n-ary multivariable $\mathbf{A} = A_1, \ldots, A_n$, the languages got by iterated substitution of L for $\mathbf{A}$ in $\mathbf{A}$ and in L are the least solutions of

$$\begin{aligned} L^{*\mathbf{A}} &= \{\mathbf{A}\} + L \cdot_{\mathbf{A}} L^{*\mathbf{A}}, \\ L^{+\mathbf{A}} &= L + L \cdot_{\mathbf{A}} L^{+\mathbf{A}}. \end{aligned}$$

The following two equalities are easy to prove, using the explicit construction of the least solution:

$$\begin{aligned} L^{*\mathbf{A}} &= \textstyle\bigcup_n (\mathbf{A} + L) \cdot_{\mathbf{A}} \ \ldots \ \cdot_{\mathbf{A}} (\mathbf{A} + L) \qquad (n \text{ times}), \\ L^{+\mathbf{A}} &= L \cdot_{\mathbf{A}} L^{*\mathbf{A}}. \end{aligned}$$

Definition 3.6 The set Rat_n of rational languages of tuples of length n is the smallest set of languages containing the finite languages of n-tuples and closed by the following operations:

1. $L \in \mathrm{Rat}_n \ \& \ M \in \mathrm{Rat}_n \Rightarrow L \cup M \in \mathrm{Rat}_n$.
2. $L \in \mathrm{Rat}_n \ \& \ |\mathbf{X}| = m \ \& \ M \in \mathrm{Rat}_m \Rightarrow L \cdot_{\mathbf{X}} M \in \mathrm{Rat}_n$.
3. $L \in \mathrm{Rat}_n \ \& \ |\mathbf{X}| = n \Rightarrow L^{*\mathbf{X}} \in \mathrm{Rat}_n$.

The family Rat of rational languages over F and $X\omega$ is the union of all Rat_n $(n \geqslant 0)$.

The following result is expected.

Proposition 3.7 *A language L of n-tuples is rational if and only if it is generated by some grammar G starting from some axiom $A_1 \ldots A_n = \mathbf{A}$*

$$L \in \mathrm{Rat}_n \iff (\exists G, A_1 \ldots A_n)\ L = L(G, A_1 \ldots A_n).$$

Proof. The "only if" part: finite languages can obviously be generated by grammars and proposition 3.1 already ensures preservation of languages generated by grammars under finite unions.

The case of substitution: Consider two grammars G and G' having disjoint non-terminals and let $\mathbf{X} = X_1 \ldots X_n$ be a multivariable of length $n = |\mathbf{S}'|$. Note first that if a multivariable $X_1 \ldots X_n$ has an instance $X_1^j \ldots X_n^j$ in $L(G, \mathbf{S})$ then this instance comes in a whole from an instance of $X_1 \ldots X_n$ in some production $\mathbf{A} \xrightarrow[G]{} \alpha$. Otherwise, the various variables occurring in the multivariable would have different instances, because of the definition of the step of derivation, which introduces new instances of variables. Then the language $L(G, \mathbf{S}) \cdot_{\mathbf{X}} L(G', \mathbf{S}')$ is generated by the grammar $G \cup \{\mathbf{X} \to \mathbf{S}'\} \cup G'$ starting from the axiom $\mathbf{S}$ of G. Or alternatively, one may consider the grammar $G \cup G'$ in which all instances of $\mathbf{X}$ in the productions of G have been replaced by different instances of $\mathbf{S}'$. The proof is the same as the proof of preservation of context-free (word) languages by substitutions (cf. for instance Hopcroft & Ullman [1979], th. 6.2 p. 131), or the proof of the same property for trees (in Gecseg & Steinby [1984], th. 4.6).

The case of iterated substitution: Let $L(G, \mathbf{S})$ be, as before, a language generated by G starting from $\mathbf{S}$, and $\mathbf{X} = X_1 \ldots X_n$ be a multivariable of length $n = |\mathbf{S}|$. The relation got by iterated substitution of $L(G, \mathbf{S})$ for $\mathbf{X}$ in $\mathbf{X}$ is generated by the following grammar: add a new non-terminal $\mathbf{R}$ of length $|\mathbf{X}|$ to G and replace in G all instances of $\mathbf{X}$ by $\mathbf{R}$. Define G' as the set of all these modified productions together with the productions $\mathbf{R} \longrightarrow \mathbf{S}$ and $\mathbf{R} \longrightarrow \mathbf{X}$. Then $L(G', \mathbf{R}) = L(G, \mathbf{S})^{*\mathbf{X}}$. The proof is the same as for trees (cf. Gecseg & Steinby [1984] th. 4.8, p. 76).

Conversely, the "if" part is easy once one has noticed that the parse trees for a grammar G make a rational set of trees, and that the correspondence between the languages of parse trees and the languages of their yields preserves the rational operations over the sets of parse trees. More precisely, the yield of a tree-language is defined as the set of all yields of the trees of the language:

$$Y(K) = \{Y(t); t \in K\}$$

and the rational operations over the tree-languages are defined as follows:

$$\begin{aligned} K + K' &= K \cup K', \\ K \cdot_{\mathbf{A}} K' &= \bigcup_{t \in K} t \cdot_{\mathbf{A}} K', \\ K^{*\mathbf{A}} &= \{A\} + K \cdot_{\mathbf{A}} K^{*\mathbf{A}}, \\ &= \bigcup_{n \geqslant 0} (\mathbf{A} + K) \cdot_{\mathbf{A}} \cdots \cdot_{\mathbf{A}} (\mathbf{A} + K) \qquad (n \;\; \text{times}). \end{aligned}$$

The correspondence is given by the following lemma.

Lemma 3.8 *Let G be a grammar of tuples in which the non-terminal $\mathbf{X}$ has no production, and K and K' be two sets of parse trees for G. Then*
$Y(K + K') = Y(K) + Y(K')$, where the symbol $+$ denotes the union,
$Y(K \cdot_{\mathbf{X}} K') = Y(K) \cdot_{\mathbf{X}} Y(K')$ and
$Y(K^{*\mathbf{X}}) = Y(K)^{*\mathbf{X}}$.

Proof. The first equality is a direct consequence of the definition of $Y(K)$. It is enough to prove the second by induction on a tree t in K.

Base case: if $t = \mathbf{X}$ then $Y(\mathbf{X} \cdot_{\mathbf{X}} K') = Y(K')$.

General case: if $t = p(t_1, \ldots, t_n)$ where $p : \mathbf{A} \longrightarrow \alpha$ and α contains the instances of non-terminals $B_1, \ldots, B_m$ (possibly $\mathbf{X}$ is among them), then by definition of the operation $\cdot_{\mathbf{X}}$ one has

$$t = p(t_1, \ldots, t_m) \cdot_{\mathbf{X}} K' = p(t_1 \cdot_{\mathbf{X}} K') \ldots (t_m \cdot_{\mathbf{X}} K')$$

and by induction hypothesis

$$Y(t_i \cdot K') = Y(t_i) \cdot_{\mathbf{X}} Y(K').$$

Then by definition of the yield

$$Y(t \cdot_{\mathbf{X}} K') = \alpha[Y(t_1 \cdot_{\mathbf{X}} K')/\mathbf{B}_1, \ldots, Y(t_m) \cdot_{\mathbf{X}} Y(K')/\mathbf{B}_m].$$

By associativity of substitution and from the definition of $Y(t)$ again:

$$\begin{aligned} Y(t \cdot_{\mathbf{X}} K') &= \alpha[Y(t_1)/\mathbf{B}_1, \ldots, Y(t_m)/\mathbf{B}_m] \cdot_{\mathbf{X}} Y(K') \\ &= Y(t) \cdot_{\mathbf{X}} Y(K'). \end{aligned}$$

To prove the third equality, we notice that

$$Y[(\mathbf{X}+K)\cdot_{\mathbf{X}}\cdots\cdot_{\mathbf{X}}(\mathbf{X}+K)] = [\mathbf{X}+Y(K)]\cdot_{\mathbf{X}}\cdots\ \cdot_{\mathbf{X}}[\mathbf{X}+Y(K)] \quad (n \text{ times})$$

by induction on n. Hence:

$$\begin{aligned} Y(K^{*\mathbf{X}}) &= Y(\bigcup_{n\geqslant 0}(\mathbf{X}+K)\cdot_{\mathbf{X}}\cdots\cdot_{\mathbf{X}}(\mathbf{X}+K) \\ &= \bigcup_{n\geqslant 0}[\mathbf{X}+Y(K)]\cdot_{\mathbf{X}}\cdots\ \cdot_{\mathbf{X}}[\mathbf{X}+Y(K)] \\ &= Y(K)^{*\mathbf{X}}. \end{aligned}$$

This ends the proof of the lemma. ∎

Consider now a grammar G. The set of all parse trees for G is generated by the associated grammar defined in section 2. Therefore it is a rational forest, described by a rational expression (cf. Gecseg & Steinby [1984]). The same expression in which the elementary trees are replaced by their yields describes all the lists of trees having a parse tree in the forest: the language generated by the grammar. ∎

This proposition is a slight refinement of theorem 4.3 of Habel & Kreowski [1987] in the particular case of tree relations. Henceforth we shall call "rational" the languages of the form $L(G,\mathbf{A})$ for some grammar G and some axiom $\mathbf{A}$.

This correspondence between the parse trees and their yielded relation also gives an easy pumping lemma: a criterion of non-rationality.

Lemma 3.9 *To every grammar G is associated a number N such that every tuple $u \in L(G,\mathbf{A})$ having at least N occurrences of symbols from F has a decomposition $u = \alpha \cdot_{\mathbf{B}} \beta \cdot_{\mathbf{B}} v$ in which α and β have only one instance of* **B**, *with $\|\beta\| > 0$ and $\alpha \cdot_{\mathbf{B}} (\beta)^{*\mathbf{B}} \cdot_{\mathbf{B}} v \subseteq L(G,\mathbf{A})$.*

Proof. Without loss of generality, we restrict to grammars having no production of size zero (see Proposition 2.11 section 2). Then apply the pumping lemma to the rational set of parse trees (see Gecseg & Steinby [1984] lemma 10.1 p.109): all parse trees t of sufficient depth (at least the number of non-terminals) can be decomposed into

$$t = t_1 \cdot_{\mathbf{B}} t_2 \cdot_{\mathbf{B}} t_3$$

with t_1 and t_2 containing a unique instance of **B** and t_2 of depth at least one, such that $t_1 \cdot_{\mathbf{B}} (t_2)^{\cdot k_{\mathbf{B}}} \cdot_{\mathbf{B}} t_3$ is again a parse tree for G, for all k. Hence the result with $\alpha = Y(t_1)$, $\beta = Y(t_2)$ and $v = Y(t_3)$. To ensure that a parse tree t of u has depth at least m, it is enough to assume that the size $\|u\|$ of u is at least gd^m where g is the maximum size of right-hand sides of G, d is the maximum number of instances of non-terminals occurring in the right-hand sides (the maximum out-degree of the vertices of the parse trees) and m is the number of non-terminals. Indeed, a parse tree of depth p has at most d^p vertices and its yield has size at most gd^p since every production increases the size by g. ∎

From this lemma we can deduce for instance that the set of perfect binary trees (binary trees having all their leaves at the same depth) is not rational, because **B** in β has a fixed length n. Actually, this set is not even algebraic in the classical sense.

Algebraic and rational relations are incomparable: the set of trees of the form $a^n b^n(e)$ for $n \geqslant 0$ is algebraic (generated by the grammar $f(x) \longrightarrow a(f(b(x)))$ / x starting from $f(e)$) but it is not rational, with the same proof as for the non-rationality of the language $a^n b^n; n \geqslant 0$. On the other hand, the forest

$$\{f(a^n(e)\, b^n(e)); n \geqslant 0\} = f\Big((a(x)\, b(y))^{*xy} \cdot_{xy} e\, e\Big)$$

is generated by the following grammar:

$$\left\{ \begin{array}{ccl} A & \to & f(I_1\, I_2) \\ I_1 I_2 & \to & a(I_1)\, b(I_2) \ / \ e\, e \end{array} \right.$$

and described by the expression $f\Big((a(x)\, b(y))^{*xy} \cdot_{xy} e\, e\Big)$: it is rational but not algebraic.

We mention finally that rational languages are preserved under some restricted converse of the substitution: the residual operation, defined as follows.

Definition 3.10 Given a subset L of $T(F, X\omega)^+$ and a linear tuple u in $T(F, X)^+$ containing the sequence of variables $\mathbf{x}$, the residual of L under u is the set

$$u^{-1}L = \{w \in T(F, X\omega)^{|\mathbf{x}|};\ u[w/\mathbf{x}] \in L\}.$$

This definition leads to the following result.

Proposition 3.11 *From a grammar G, a non-terminal $\mathbf{A}$ and a linear tuple $u \in T(F, X)^+$ of same length, one can deduce a grammar G' and a non-terminal $[u^{-1}\mathbf{A}]$ such that*

$$[u^{-1}\mathbf{A}] \underset{G'}{\longrightarrow}{}^{n} v \iff \mathbf{A} \underset{G}{\longrightarrow}{}^{n} u[v/\mathbf{x}]$$

where $\mathbf{x}$ is the sequence of variables occurring in u, for all v containing no instance of a non-terminal.

Proof. It is known that given two terms s and t in $T(F, X)$ which are compatible (there exist two substitutions σ and τ such that $s\sigma = t\tau$), there exist terms $r(x_1, \ldots, x_p, y_1, \ldots, y_q)$ and $s_1, \ldots, s_p$ and $t_1, \ldots, t_q$ satisfying

$$\left\{ \begin{array}{ccl} s & = & r(s_1, \ldots, s_p, y_1, \ldots, y_q) \\ t & = & r(x_1, \ldots, x_p, t_1, \ldots, t_q). \end{array} \right.$$

Its extension to tuples of terms is immediate, and can be proved by induction using proposition 2.2. We shall use it as follows. Define a new grammar G' having the non-terminals $[v^{-1}\mathbf{B}]$ for all non-terminal $\mathbf{B}$ of G and all linear tuples v such that v_i is a variable or a subterm of u (there is only a finite number of such non-terminals). We shall define the productions of $[v^{-1}\mathbf{B}]$ in order to ensure

$$[v^{-1}\mathbf{B}] \underset{G'}{\longrightarrow}{}^{n} w \iff \mathbf{B} \underset{G}{\longrightarrow}{}^{n} v[w/\mathbf{x}] \quad (n > 0).$$

Start from the derivation on the right and isolate the first step:

$$\mathbf{B} \underset{G}{\longrightarrow} \alpha \underset{G}{\longrightarrow}{}^{n-1} v[w/\mathbf{x}]. \qquad (*)$$

Then α and v are compatible: there exist tuples s, t and $r(\mathbf{y}\,\mathbf{z})$ where $\mathbf{y}$ is a subsequence of $\mathbf{x}$ (in v) and $\mathbf{z}$ a sequence of variables of α such that

$$\begin{cases} v &= r(\mathbf{y}\,t) \\ \alpha &= r(s\,\mathbf{z}). \end{cases}$$

Replacing in $\alpha \xrightarrow[G]{n-1} v[w/\mathbf{x}]$ and simplifying by the prefix r we get

$$s\,\mathbf{z} \xrightarrow[G]{n-1} (\mathbf{y}\,t)[w/\mathbf{x}].$$

For all instance $\mathbf{B}^k$ of a non-terminal in α (or equivalently in $s\,\mathbf{z}$), call t^k the $|\mathbf{B}^k|$-tuple such that $t^k_i = x_i$ if B^k_i is a variable in s, and t^k_i is the subtree of t substituted for B^k_i if B^k_i is a variable of $\mathbf{z}$ (these tuples are concatenations of sub-tuples of v hence are in finite number). Define the production of G' associated with v and $\mathbf{B} \xrightarrow[G]{} \alpha$ as

$$[v^{-1}\mathbf{B}] \xrightarrow[G']{} (s\,\mathbf{z})[\ldots, [(t^k)^{-1}\mathbf{B}^k]/\mathbf{B}^k, \ldots].$$

Then derivation $(*)$ exists if and only if $[(t^k)^{-1}\mathbf{B}^k]$ derives w^k, the tuple substituted to the subsequence $\mathbf{x}^k$ of variables of $\mathbf{x}$ occurring in t^k, hence the result by induction on n. ∎

4 Grammars and finite copying transducers

Recall the definition of a tree transducer (actually a top-down, or root-to-frontier tree transducer, see for instance Gecseg & Steinby [1984]).

Definition 4.1 Given a ranked alphabet of function symbols F and a set X of variables, a top-down tree transducer on $T(F, X)$ (we shall omit 'top-down') is a finite set Q of unary states together with a finite set of rules of the form

$$qf(x_1\,\ldots\,x_n) \vdash t(q_1x_{i_1}, \ldots, q_px_{i_p})$$

where $q \in Q$, $f \in F$, n is the degree of f, $x_1, \ldots, x_n$ are distinct variables in X, $t \in T(F, Q\{x_1, \ldots x_n\})$ and $q_1x_{i_1}, \ldots q_px_{i_p}$ are the elements of $Q\{x_1, \ldots, x_n\}$ occurring in t from left to right.

Note that instead of the strict notation $q(x)$ for a unary q, we allow the shorter qx. The rule above generates a relation on $T(F \cup Q, X)$ by substitution of t_1 for x_1,… t_n for x_n (to apply the rule at the root):

$$qt = qf(t_1\,\ldots\,t_n) \vdash_\tau t(q_1t_{i_1}, \ldots, q_pt_{i_p})$$

and by addition of a context (to apply the rule below the root):

$$t \vdash_\tau t' \Rightarrow f(t_1\,\ldots\,t\,\ldots t_n) \vdash_\tau f(t_1\,\ldots\,t'\,\ldots t_n)$$

for all function symbol $f \in F$ an all trees $t_1, \ldots, t_n$. The name τ of the transducer may be omitted if no ambiguity results. Once τ is given, a state q defines the relation

$L(\tau, q) \subseteq T(F) \times T(F)$ which contains all pairs (s, t) such that $qs \vdash^+ t$ (there is no state left in t):

$$L(\tau, q) = \{(s, t) \in T(F, X) \times T(F, X);\ qs \vdash_\tau^+ t\}.$$

Transducers τ come usually equipped with a distinguished initial state q_0 and in this case the relation defined by τ is $L(\tau, q_0)$.

If in the definition of the transducer all the rules satisfy $t = f(x_1 \ldots x_n)$ then the transducer is a F-automaton and its domain is the language recognized by the automaton. Any transducer τ induces an automaton over $\bigcup_n \{(f, i);\ f \in F_n \text{ and } 1 \leqslant i \leqslant n\}$, and which has states in the set Q^* of sequences of states of τ by $q_1 \ldots q_d \vdash_{(f,j)} q_{(j,1)} \ldots q_{(j,h_j)}$ ($h_j \geqslant 0$ and $q_{(j,i)} \in Q$ for $i = 1, \ldots, h_j$) if there exist d rules $q_i f(x_1, \ldots, x_n) \vdash_\tau r_i$ (for $i = 1, \ldots, d$) such that $q_{(j,1)}x_j \ldots q_{(j,h_j)}x_j$ is the sequence of all occurrences of elements of Qx_j in the sequence $r_1 \ldots r_d$ (cf. Engelfriet et al. [1980], def. 3.1.8). This automaton is restricted to the states accessible from the sequence q_0.

Definition 4.2 A tree transducer is k-copying if and only if its induced automaton restricted to the states accessible from the initial state of the transducer, has set of states included in $1 + Q + \cdots + Q^k$.

This is essentially definition 3.1.9 of Engelfriet et al. [1980].

Example 4.3 Consider the transducer having set $Q = \{A, B_1, B_2, C_1, C_2\}$ of states and the following rules:

$$\begin{array}{rcl}
Ap(x) & \vdash & a(B_1x, B_2x) \\
B_1q(x, y) & \vdash & b(B_1x, C_2y) \\
B_2q(x, y) & \vdash & b(B_2x, C_1y) \\
B_1r & \vdash & d \\
B_2r & \vdash & d \\
C_1s(x) & \vdash & c(d, B_1x) \\
C_2s(x) & \vdash & B_2x \\
C_1t & \vdash & d \\
C_2t & \vdash & d
\end{array}$$

This transducer induces the following automaton:

$$\begin{array}{rcl}
(A)p(x) & \vdash & p((B_1B_2)x) \\
(B_1B_2)q(x, y) & \vdash & q((B_1B_2)x\,(C_2C_1)y) \\
(B_2B_2)q(x, y) & \vdash & q((B_2B_1)x\,(C_1C_2)y) \\
(B_1B_2)r & \vdash & r \\
(B_2B_1)r & \vdash & r \\
(C_1C_2)s(x) & \vdash & s((B_1B_2)x) \\
(C_2C_1)s(x) & \vdash & s((B_2B_1)x) \\
(C_1C_2)t & \vdash & t \\
(C_2C_1)t & \vdash & t
\end{array}$$

in which we have only computed the transitions of states accessible from A. This automaton has states in $1 + Q + Q^2$: the transducer is 2-copying.

When the states of the induced automaton belong to $1 + Q$ the rules of the transducer do not duplicate any variable: no subtree is duplicated during the transductions.

Proposition 4.4 *With every pair of a k-copying tree transducer and an initial state q_0 can be associated a grammar G in which non-terminals have length at most $k+1$, and a non-terminal $\mathbf{X}_{q_0}$ such that $L(G, \mathbf{X}_{q_0}) = L(\tau, q_0)$. Hence, this relation has a rational range.*

Proof. Given a transducer τ over an alphabet P, with set Q of states, one defines the following grammar G: for all state sequence $u = q_1 \dots q_d$ $(0 \leqslant d \leqslant k)$ introduce a non-terminal $\mathbf{X}_u = X_{u,0}X_{u,1} \dots X_{u,d}$ of length $d+1$ with productions

$$\mathbf{X}_u \underset{G}{\longrightarrow} p(X_{v_1,0} \, \dots \, X_{v_m,0})\, r'_1 \, \dots \, r'_d$$

if there exist d rules:

$$q_i p(x_1 \, \dots \, x_m) \vdash_\tau r_i \quad (i = 1, \dots, d)$$

where r'_i is deduced from r_i as follows. In the tuple $r_1 \dots r_d$, denote by $q_{(j,1)}x_j \dots q_{(j,h_j)}x_j$ the sequence of subtrees of depth one having leaves equal to x_j (it is the sequence used in the definition of the induced automaton). To shorten notations we denote by v_j the sequence of states $q_{(j,1)}, \dots, q_{(j,h_j)}$. Then the tuple $r'_1 \dots r'_d$ is deduced from the tuple $r_1 \dots r_d$ by replacing the sequence of subtrees $q_{(j,1)}x_j, \dots, q_{(j,h_j)}x_j$ by the non-terminal variables $X_{v_j,1}, \dots, X_{v_j,h_j}$, for $j = 1, \dots, m$ (the first such variable is $X_{v_j,1}$ because $X_{v_j,0}$ is the j-th argument of p).

We shall prove the following equivalence for all trees $t, t_1, \dots, t_d$ in $T(F)$ and all sequences $u = q_1 \dots q_d$ in $1 + Q + \dots + Q^k$:

$$\mathbf{X}_u \underset{G}{\longrightarrow}{}^+ t t_1 \dots t_d \iff q_i t \vdash_\tau^+ t_i \quad (i = 1, \dots, d).$$

The proof is by induction on the size $\|t\|$ of t. If $t = p(s_1 \, \dots \, s_m)$ for $m \geqslant 0$ ($m = 0$ is the base case), then for having $q_1 t \vdash^+ t_1, \dots, q_d t \vdash^+ t_d$ it is necessary and sufficient that the following three conditions be satisfied.

1. There exist d rules $q_i p(x_1 \, \dots \, x_m) \vdash_\tau r_i(q_1 x_{i_1}, \dots, q_p x_{i_p})$ for $i = 1, \dots, d$ which are the first steps of each transduction;
2. the subtrees s_i transduce to terms in $T(F, X)$: $q_{i_j} s_{i_j} \vdash_\tau {}^* a_{i_j}$
3. which are subterms of the t_i: $t_i = r_i(a_{i_1}, \dots, a_{i_{p_i}})$ (for $i = 1, \dots, d$).

Then by definition of the associated grammar, the first condition is equivalent to the next one:

1′. There exists a production in G:

$$\mathbf{X}_{q_1 \dots q_d} \underset{G}{\longrightarrow} p(X_{v_1,0} \, \dots \, X_{v_m,0})\, r'_1 \, \dots \, r'_d.$$

The second condition is now equivalent (by induction hypothesis) to

2′. $\mathbf{X}_{v_1} \underset{G}{\longrightarrow}{}^+ s_1 a_{(1,1)} \dots a_{(1,h_1)}$, etc. down to $\mathbf{X}_{v_m} \underset{G}{\longrightarrow}{}^+ s_m a_{(m,1)} \dots a_{(m,h_m)}$.

The conjunction of conditions 1′, 2′ and 3 is in turn equivalent to

$$\begin{array}{rcl} \mathbf{X}_{q_1\dots q_d} & \underset{G}{\longrightarrow} & p(X_{v_1,0}\ \dots\ X_{v_m,0})\, r'_1\ \dots\ r'_d \\ & \underset{G}{\longrightarrow}^{*} & p(s_1\ \dots\ s_m)\, r_1(a_1,\dots,a_{p_1})\dots r_d(a_{1+p_{d-1}},\dots a_{p_d}) \\ & = & tt_1\dots t_d \end{array}$$

As a particular case one gets the following equivalence for all state q and all trees s and t:

$$qs \vdash_{\tau}^{+} t \iff \mathbf{X}_q \underset{G}{\longrightarrow}^{+} s\,t.$$

There remains to be checked that we introduce only a finite number of productions. But we have proved that if the left-hand side of the grammar is the non-terminal $\mathbf{X}_u$ then the non-terminals in the right-hand sides are $\mathbf{X}_{v_1},\dots,\mathbf{X}_{v_m}$ where $u \vdash v_j$ is a rule of the automaton. Therefore the grammar is finite if and only if there is only a finite number of non-terminals accessible from the axiom $\mathbf{X}_{q_0}$ if and only if the transducer is k-copying for some k. The range of the relation is the projection of $L(G,\mathbf{X}_{q_0})$ got by erasing its first components, hence the result. ■

We shall now prove the following partial converse.

Proposition 4.5 *From any grammar G one can deduce a transducer τ having as state set the set of all variables of the non-terminals of G, having as domain the set of all the parse trees of the grammar, and such that $t_1\dots t_n \in L(G, A_1\dots A_n)$ if and only if for some parse tree s, one has $A_i s \vdash_\tau t_i$ for $i = 1,\dots,n$.*

Proof. In order to prove this proposition we shall first show an example then prove a technical lemma. Given a grammar G, the associated transducer τ is defined as follows. Its input alphabet is the set P of names of productions in G. Its output alphabet is the terminal alphabet of G. Its set of states is the set of variables occurring in the non-terminals: $Q = \{A_i; A_1\dots A_n \in \mathcal{N}\}$. Its rules are

$$A_i p(x_1\ \dots\ x_m) \vdash r_i[\dots, B^j_\ell x_j / B^j_\ell, \dots]$$

for $\ell = 1,\dots,|\mathbf{B}^j|$ and $j = 1,\dots,m$ whenever there exists a production

$$p : A_1\dots A_n \underset{G}{\longrightarrow} r_1\dots r_n$$

in which occur the instances $\mathbf{B}^1,\dots,\mathbf{B}^m$ of non-terminals.

Example 4.6 Consider the following grammar G:

$$\left\{\begin{array}{lrcl} p: & A & \longrightarrow & a(B_1\,B_2) \\ q: & B_1B_2 & \longrightarrow & b(B_1\,C_2)\,b(B_2\,C_1) \\ r: & B_1B_2 & \longrightarrow & d\,d \\ s: & C_1C_2 & \longrightarrow & c(d\,B_1)\,B_2 \\ t: & C_1C_2 & \longrightarrow & d\,d. \end{array}\right.$$

The associated tranducer is given in example 4.3. For instance, the tree

$$a(b(b(d\,d)\,d)\,b(b(d\,c(d\,d))\,d))$$

has $p(q(q(r\,s(r))\,t))$ as parse tree and has the following derivation:

$$\begin{array}{rcl}
Ap(q(q(r\,s(r))\,t)) & \longrightarrow & a(B_1q(q(r\,s(r))\,t)\,B_2q(q(r\,s(r))\,t)) \\
& \longrightarrow^2 & a(b(B_1q(r\,s(r))\,C_2t)\,b(B_2q(r\,s(r))\,C_1t)) \\
& \longrightarrow^4 & a(b(b(B_1r\,C_2s(r))\,d)\,b(b(B_2r\,C_1s(r))\,d)) \\
& \longrightarrow^4 & a(b(b(d\,B_2r)\,d)\,b(b(d\,c(d\,B_1r))\,d)) \\
& \longrightarrow^2 & a(b(b(d\,d)\,d)\,b(b(d\,c(d\,d))\,d)).
\end{array}$$

When the construction of proposition 4.4 is applied to the resulting tree transducer, the grammar which is built is easy to get directly from the original grammar: for each non-terminal $\mathbf{A} = A_1 \ldots A_n$ of G define a non-terminal $\mathbf{X_A} = X_{\mathbf{A},0}X_{\mathbf{A},1}\ldots X_{\mathbf{A},n}$ of G' (with $|\mathbf{X_A}| = |\mathbf{A}| + 1$) and for each production

$$p : \mathbf{A} \underset{G}{\longrightarrow} (r_1, \ldots, r_d)$$

of G define a production of G'

$$p' : \mathbf{X_A} \underset{G'}{\longrightarrow} p(X_{\mathbf{B}^1,0}\, \ldots\, X_{\mathbf{B}^m,0})\, r'_1\, \ldots\, r'_n$$

in which r'_i is deduced from r_i by replacing every variable B^j_ℓ of a non-terminal $\mathbf{B}^j$ by the variable $\mathbf{X}_{\mathbf{B}^j,\ell}$ of the corresponding non-terminal in G'. The construction of this associated grammar makes it clear that the projection of the language generated from any non-terminal $\mathbf{X_A}$ got by erasing the first tree is $L(G, \mathbf{A})$. The next lemma states the correspondence.

Lemma 4.7 *Given a grammar G, define the corresponding transducer τ as above; then construct the grammar G' deduced from τ as in proposition 4.4. Then G' is the grammar directly deduced from G as above.*

Proof. Production p and its n corresponding rules in τ are shown below

$$p : \mathbf{A} \longrightarrow r_1\, \ldots\, r_n \iff A_ip(x_1\, \ldots\, x_m) \vdash r'_i \quad (i = 1, \ldots, n)$$

in which $r'_1\, \ldots\, r'_n$ is deduced from $r_1\, \ldots\, r_n$ by replacing B^j_ℓ by $B^j_\ell x_j$. Then the grammar G' corresponding to the transducer τ by proposition 4.4 has non-terminals $\mathbf{X}_u$ of length $n+1$ for all sequences of states $u = q_1 \ldots q_n$, and productions

$$\mathbf{X}_u \underset{G''}{\longrightarrow} p(X_{v_1,0}\, \ldots\, X_{v_m,0})\, r''_1\, \ldots\, r''_n$$

if there exist n rules of the transducer

$$q_ip(x_1\, \ldots\, x_m) \vdash_\tau r'_i \quad (i = 1, \ldots, n)$$

and the tuple $r''_1\, \ldots\, r''_n$ is deduced from $r'_1\, \ldots\, r'_n$ by replacing the sequence of subterms $q_{(1,1)}x_1, \ldots, q_{(1,h_1)}x_1$ by the non-terminal variables $X_{v_1,1}, \ldots, X_{v_1,h_1}$, etc, the sequence $q_{(m,1)}x_m, \ldots,\ q_{(m,h_m)}x_m$ by the non-terminal variables $X_{v_m,1},\ \ldots, X_{v_m,h_m}$. By definition of τ, the sequence of subterms $q_{(j,1)}x_j,\ \ldots,\ q_{(j,h_j)}x_j$ is equal to $B^j_1x_j$, $\ldots$, $B^j_{m_j}x_j$, so that the corresponding non-terminal in G' is $\mathbf{X}_{v_j}$ of length $h_j + 1$. This also proves that the transducer τ is k-copying, where k is the maximum length of the non-terminals in G. ∎

Proposition 4.5 above is an easy consequence of this lemma. Propositions 4.4 and 4.5 together with an inspection of the lengths of the non-terminals of the implied grammars yield the following connection, which was the aim of this section.

Proposition 4.8 *Rational relations generated by grammars in which non-terminals have length bounded by k are exactly the ranges of k-copying top-down tree transducers.*

5 Composition of relations

We consider now languages in $T(F)^{p+q}$ as binary relations in $T(F)^p \times T(F)^q$ in order to investigate their behavior with respect to composition. The first (resp. the second) projection will be understood to be $\mathrm{pr}_1 : T(F)^p \times T(F)^q \to T(F)^p$ (resp. $\mathrm{pr}_2 : T(F)^p \times T(F)^q \to T(F)^q$). To explicit these two projections, attach to each non-terminal $\mathbf{X}$ a subset I of $[1, |\mathbf{X}|]$ indicating which of its variables X_i are in the first projection: those satisfying $i \in I$; those that are in the second projection satisfy $i \notin I$. Without loss of generality we shall assume, modulo renaming the variables of each non-terminal, that $I = [1, p]$ and $\overline{I} = [p+1, p+q]$. This assumption allows to write $\mathbf{X} = \mathbf{X}_s\mathbf{X}_d$ where $\mathbf{X}_s$ is the sequence of variables in the first projection (*sinistra*) and $\mathbf{X}_d$ is the sequence of variables in the second projection (*dextra*).

The following operation on grammars, using complete derivations for productions, will be used in the sequel.

Definition 5.1 Given two grammars G and H over the same non-terminals, the grammar GH has same non-terminals and productions

$$A \longrightarrow \alpha[\beta_1/\mathbf{B}^1, \ldots, \beta_m/\mathbf{B}^m]$$

where $A \underset{G}{\longrightarrow} \alpha$, where $\mathbf{B}^1, \ldots, \mathbf{B}^m$ are all the instances of non-terminals in α, and where $\mathbf{B}^j \underset{H}{\longrightarrow} \beta_j$ for $j = 1, \ldots, m$.

In particular, terminal productions of G ($m = 0$) belong to GH. As usual we denote $G^1 = G$ and $G^k = G^{k-1}G$ for $k > 1$. It is straightforward to show $L(G^k, A) = L(G, A)$ for all $k > 0$ and all non-terminal $\mathbf{A}$, as with words.

Begin with two rational relations having recognizable projections in the sense of Pair & Quéré [1968], or of Gecseg & Steinby [1984] when the length of the projections is one. Their composition need not have the same property, and need not even be rational, as in the following example.

Example 5.2 Consider the relations generated by $\mathbf{A}$ and $\mathbf{X}$. Here a and b have arity one and zero, so that these relations are actually relations on words.

$$\begin{cases} A_1A_2A_3 &= a(A_1)\, A_2\, a(A_3) + a\, B_1\, a(B_2) \\ B_1B_2 &= b(B_1)\, b(B_2) + b\, b, \end{cases}$$

$$\begin{cases} X_1X_2X_3 &= b(X_1)\, X_2\, b(X_3) + b(Y_1)\, Y_2\, b \\ Y_1Y_2 &= a(Y_1)\, a(Y_2) + a\, a. \end{cases}$$

The relations $L(G, \mathbf{X}) \subseteq T \times T^2$ and $L(G, \mathbf{A}) \subseteq T^2 \times T$ have both projections recognizable: $L(G, \mathbf{X}) = \{(b^m a^n, a^n, b^m); n, m > 0\}$ and $L(G, \mathbf{A}) = \{(a^n, b^m, a^n b^m); m, n >$

$0\}$. The composition is $\{(b^m a^n, a^n b^m); m, n > 0\}$ which is not rational. What happens here is that two variables of A (and of X) are allowed to occur at unbounded depths while the third variable remains at a fixed depth, remembering in some way a synchronization to come between variables at arbitrarily distant depths. The definition below will rule out this situation.

Definition 5.3 A relation in $T^p \times T^q$ is called a transduction when it is generated by a grammar G such that in some power G^k of G, all the non-terminals occurring in the right-hand sides have all their variables in two trees at most, one in each projection of the relation. Such a grammar is called a transduction grammar.

This definition implies that each non terminal $\mathbf{X}$ of the grammar has been decomposed into $\mathbf{X}_s\mathbf{X}_d$ indicating which variables are in which projections. The first and second projections of a transduction grammar generate recognizable languages because each projection is equivalent to a grammar in which all non-terminals have length one. But the grammars defined in example 5.2 above are not transduction grammars. Yet, they have recognizable projections.

Transduction grammars are a recursive subset of all the grammars:

Proposition 5.4 *It is decidable whether a grammar G is a transduction grammar.*

Proof. If a grammar G is not a transduction grammar, then the first projection of G (or the second) satisfies the following assertion: for all k, there is a right-hand side $t_1 \ldots t_p$ of $\mathrm{pr}_1 G^k$ and a non-terminal $\mathbf{X}$ having two variables in two different trees t_{k_1} and t_{k_2} of this right-hand side. These two variables X_i and X_j occur in two trees produced by two variables Y_p and Y_q of some non-terminal $\mathbf{Y}$ in some right-hand side of G^{k-1}. And so forth, up to two variables in two trees of a right-hand side of G. We have therefore $k-1$ pairs of variables belonging to the non-terminals of G. Choose $k = 2 + \sum_{\mathbf{X}\in\mathcal{N}} |\mathbf{X}|(|\mathbf{X}|-1)$, computable from G. Then there cannot exist $k-1$ different pairs of variables; so there must exist a non-terminal $\mathbf{X}$ and two variables X_i and X_j of $\mathbf{X}$ such that

$$\mathbf{X} \underset{G}{\longrightarrow}^{\ell} t_1 \ldots t_{|\mathbf{X}|}$$

for some $\ell \leqslant k$ and X_i occurs in t_i and X_j occurs in t_j. Conversely, if such a derivation exists for a projection of G then G clearly cannot be a transduction grammar. All such derivations can be checked for this condition on G, and therefore it is decidable whether G is not a transduction grammar. ■

We introduce now a notation which will be used throughout the rest of the section.

Notation 5.5 Given a tree t in $T(F, X\omega)$ and a grammar, we consider the set of non-terminals having a variable in t. This set is ordered by left to right ordering in t of their first variable (in our case, each non-terminal will have a single variable in t): $\mathbf{X}^1, \ldots, \mathbf{X}^m$. Then we set

$$\nu(t) = \pi_t(X_1^1 \ldots X_{p_1}^1 \ldots X_1^m \ldots X_{p_m}^m)$$

where

$$\pi_t(X_i^j) = \begin{cases} \varepsilon & \text{if } X_i^j \text{ occurs in } t \\ X_i^j & \text{otherwise} \end{cases}$$

We call $\nu(t)$ the sequence of variables attached to t.

Intuitively, $\nu(t)$ is the sequence of non-terminals having a variable in t in which this variable has been erased (see figure 4).

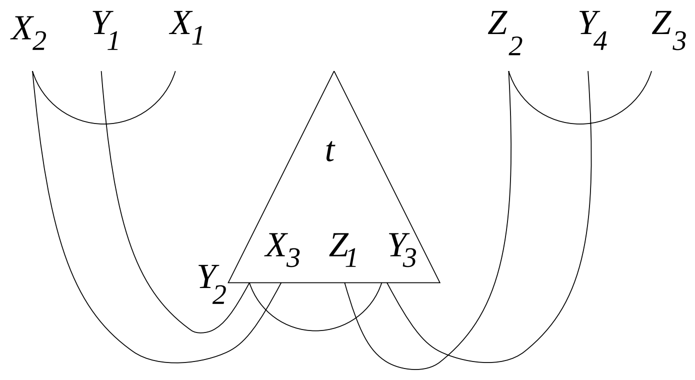

Figure 4: An example of variables attached to a tree. Here $\nu(t) = Y_1Y_4X_1X_2Z_2Z_3$

Proposition 5.6 *Every transduction can be generated by a transduction grammar in which all non-terminals have only one variable in the first projection (resp. in the second projection).*

Proof. Provided that we replace G by some power G^k, we may assume that all non-terminals have their variables in at most two trees, one in each projection. Make the projections visible in the style indicated at the beginning of the section: all tuples α are factorized into $\alpha = \alpha_s\alpha_d$.

Next, split each production $p : \mathbf{A}_s\mathbf{A}_d \longrightarrow \alpha_s\alpha_d = t_1 \dots t_k\alpha_d$ in two: in a first step, the second projection α_d is produced alone; in a second step, the first projection α_s is generated: for all $i = 1, \dots, k$ introduce a new non-terminal ${}_i\mathbf{P}$ of length $1 + |\nu(t_i)|$ and define β as α_d in which all the variables in $\nu(t_i)$ have been renamed ${}_iP_2, {}_iP_3$, etc. for all i. Now, introduce the following production in a new grammar H:

$$p' : \mathbf{A} \underset{H}{\longrightarrow} {}_1P_1 \dots {}_kP_1\, \beta,$$

and the unique productions of the new terminals ${}_i\mathbf{P}$ in another grammar K:

$${}_i\mathbf{P} \underset{K}{\longrightarrow} t_i\, \nu(t_i).$$

The composition of p' followed by these k new productions is equal to p. Or again, we have $HK = G$. But now, the non-terminals ${}_i\mathbf{P}$ have a unique variable in the first projection. Replacing the non-terminals of G by their productions in H consists in considering KH. The resulting grammar is the union of KH and of the productions in H of the non-terminals of G. It generates the same languages from the same non-terminals as G and satisfies the required condition. ■

Notice that $(KH)(KH) = KGH$ so that the resulting grammar is again a transduction grammar. This proposition has the following corollary.

Corollary 5.7 *The first (and the second) projection of a transduction grammar generates from any non-terminal a finite union of concatenations of recognizable tree-languages.*

Proof. The first projection of a transduction grammar generates the same languages from the same non-terminals as a grammar in which all non-terminals have only one variable. The languages generated starting from each of these non-terminals are recognizable tree-languages in the sense of Gecseg & Steinby [1984]. ■

We are now in a position to prove our main result.

Theorem 5.8 *The transductions contain the identity and are preserved by converse and composition.*

Proof. The identity is defined by a grammar of transduction, and definition 5.3 is symmetrical with respect to the first and the second projections: transductions are closed by taking the converse. There remains to show the stability under composition. Suppose two transductions grammars G and H. Using proposition 5.6 we assume that all non-terminals occurring in the right-hand sides of G have at most one variable in the second projection, and that all non-terminals occurring in the right-hand sides of H have at most one variable in the first projection. To be short, we denote by $\alpha \circ \beta$ what is in fact $L(G, \alpha) \circ L(H, \beta)$. The basic situation will be the following: with every pair $(\mathbf{A}, t)$ of a non-terminal $\mathbf{A} = A_1 \ldots A_n$ of G and a subtree t of ξ_s, where $\mathbf{X} \underset{H}{\longrightarrow} \xi_s \xi_d$, we associate a non-terminal $\mathbf{L}$ of length $\ell = n - 1 + \sum_{i=1}^{i=m} |\mathbf{X}^i| - 1$ where $\mathbf{X}^1, \ldots, \mathbf{X}^m$ are the non-terminals having variables in t. This non-terminal $\mathbf{L}$ is actually a new name for the variables of $\mathbf{A}_s \nu(t)$:

$$A_1 \ldots A_{n-1}\, \nu(t) \mapsto L_1 \ldots L_\ell$$

and it will generate $\mathbf{A} \circ t\nu(t)$. Symmetrically, for all non-terminal $\mathbf{X}$ of H and all subtree t' of α_d, where $\mathbf{A} \underset{G}{\longrightarrow} \alpha_s \alpha_d$, we associate with the pair $(t', \mathbf{X})$ the non-terminal $\mathbf{L}'$ which must generate $\nu(t')\, t' \circ \mathbf{X}$. There is only a finite number of such non-terminals. The method consists in remarking that for $\mathbf{L}$ to generate $\mathbf{A} \circ t\, \nu(t)$ using productions of G or of H, the non-terminal $\mathbf{A}$ is "late" and should generate at its occurrence $\mathbf{A}_d = A_n$ some tree having t as a prefix. Proceed by necessary conditions. There must exist a production $\mathbf{A} \underset{G}{\longrightarrow} \alpha = \alpha_s\, \alpha_d$ such that t and α_d are compatible: in this case there exist unique trees $r(x_1, \ldots, x_p, y_1, \ldots y_q), t_1, \ldots, t_p, t'_1, \ldots, t'_q$ such that

$$\begin{cases} t &= r(t_1, \ldots, t_p, y_1, \ldots, y_q) \\ \alpha_d &= r(x_1, \ldots, x_p, t'_1, \ldots, t'_q). \end{cases}$$

Note that t and α_d are linear, that t_i is a subterm of t, hence of ξ, and that t'_j is a subterm of α_d, hence of α. Introduce the following production associated with $\mathbf{A} \underset{G}{\longrightarrow} \alpha = \alpha_s\, \alpha_d$:

$$\mathbf{L} \underset{K}{\longrightarrow} \alpha'\, \mathbf{z}$$

obtained from $\alpha_s\,\nu(r)$ by renaming the variables in the following way. At each variable x_i of r, there is a variable B_m belonging to an instance of a non-terminal $\mathbf{B}$ in $\alpha_s\,\alpha_d$. This variable B_m must still produce t_i. As above define ${}_i\mathbf{L}$ to be the the non-terminal "associated with the pair $(\mathbf{B}, t_i)$, of length $|\mathbf{B}| - 1 + |\nu(t_i)|$. Symmetrically at each variable y_j of r, there is a variable Y_1 belonging to an instance of a non-terminal $\mathbf{Y}$ in ξ. This variable Y_1 must produce t'_j. As above, define a new non-terminal ${}_j\mathbf{M}$ associated with the pair $(t'_j, \mathbf{Y})$ which must generate $\nu(t'_j)\,t'_j \circ \mathbf{Y}$.

Note that variables that were synchronized in $\nu(t)$ may be desynchronized in $\mathbf{z}$ as showed in figure 5.

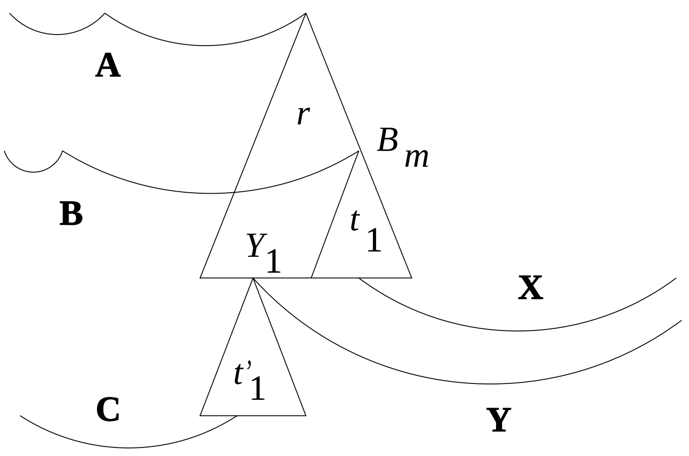

Figure 5: Originally all variables of $\mathbf{X}$ and $\mathbf{Y}$ are synchronized with the variables of $\mathbf{A}$. When $\mathbf{A}$ has produced $\alpha_s r(x, t_1)$, the variables of $\mathbf{X}$ and $\mathbf{Y}$ will be desynchronized.

Then for $\mathbf{L}$ to generate $A_1 \ldots A_n \circ t\,\nu(t)$ it is necessary and sufficient that inductively

$$\begin{array}{rcl} {}_i\mathbf{L} & \to^+ & \mathbf{B} \circ t_i\,\nu(t_i) \\ {}_j\mathbf{M} & \to^+ & \nu(t'_j)\,t'_j \circ \mathbf{Y} \end{array}$$

for all i and all j.

There remains to check that the grammar $G \cup H \cup K$ constructed is indeed a transduction grammar. This is the hypothesis for G and H. Regarding K, its first projection is a subgrammar of G (up to productions of size zero) and its second projection is a sub grammar of H (up to productions of size zero). Therefore they both satisfy the condition. ■

On the way, we have given a construction of the composition of two transductions available for words as well, but described nowhere — to our knowledge.

In the following examples, we advise the reader to draw graphical representations of the productions, in the style of figure 5.

Example 5.9 Give $\mathbf{I} = e\,e \;+\; b(I_1\,I'_1)\,b(I_2\,I'_2)$ and let us compute first $\mathbf{I} \circ \mathbf{I}$. Start arbitrarily by letting $I_1 I_2$, i.e. "the left $\mathbf{I}$" produce (here we let $J_1 J_2$ denote $I_1 I_2 \circ I'_1 I'_2$, or $\mathbf{J} = \mathbf{I} \circ \mathbf{I}$):

$$\begin{array}{rcl} 1 : \mathbf{J} & \longrightarrow & e\,\mathbf{K} \\ 2 : \mathbf{J} & \longrightarrow & b(L_1\,L_2)\,L_3 \end{array}$$

where $\mathbf{K}$ has length one and denotes $e \circ I_1 I_2$; and $L_1 L_2 L_3$ denotes $I_1\, I'_1\, b(I_2\, I'_2) \circ I''_1 I''_2$. These two non-terminals have productions:

$$\begin{array}{l} 3 : \mathbf{K} \longrightarrow e \\ 4 : \mathbf{L} \longrightarrow J_1\, J'_1\, b(J_2\, J'_2). \end{array}$$

Since production 3 is the only production of K we may apply it immediately after production 1, and similarly for production 4 which will follow immediately production 2. We get the grammar of I again, up to the name of variables. Thus $\mathbf{I} \circ \mathbf{I} = \mathbf{I}$.

Example 5.10 Consider grammar $G = H$ below

$$\left\{ \begin{array}{lcl} A_1 A_2 & = & c(A_1\, I_1\, I'_1)\, b(b(A_2\, I_2)\, I'_2) \;+\; I_1\, I_2 \\ R_1 R_2 & = & b(A_2\, I_1)\, b(A_1\, I_2) \end{array} \right.$$

depicted in figure 6 (together with the definition of $\mathbf{I}$ but this will be understood). We shall construct the grammar of the relation $\mathbf{A} \circ \mathbf{R}$ having axiom $\mathbf{S}$ (denoting $A_1 A_2 \circ R_1 R_2$). Since $\mathbf{R}$ has a unique production, we let it produce first:

$$\mathbf{S} \longrightarrow L_1\, b(L_2\, L_3)$$

where $L_1 L_2 L_3$ denotes $A_1 A_2 \circ b(A'_2\, I_2)\, A'_1 I_1$ Then $\mathbf{A}$ is late and we make it produce in $\mathbf{L}$:

$$\mathbf{L} \longrightarrow b(V_1\, J_1)\, V_2\, J_2$$

where $V_1 V_2$ denotes $I_1 I_2 \circ A_2 A_1$ and $J_1 J_2$ denotes $I_1 I_2 \circ I'_1 I'_2$. Knowing $\mathbf{I} \circ \mathbf{I} = \mathbf{I}$, the second relation is $\mathbf{J} = \mathbf{I}$; the first one is $\mathbf{I} \circ \mathbf{A}^{-1} = \mathbf{A}^{-1}$, i.e. $V_1 V_2 = A_2 A_1$ (knowing $\mathbf{I} \circ \mathbf{X} = \mathbf{X}$ for all relation $\mathbf{X}$).

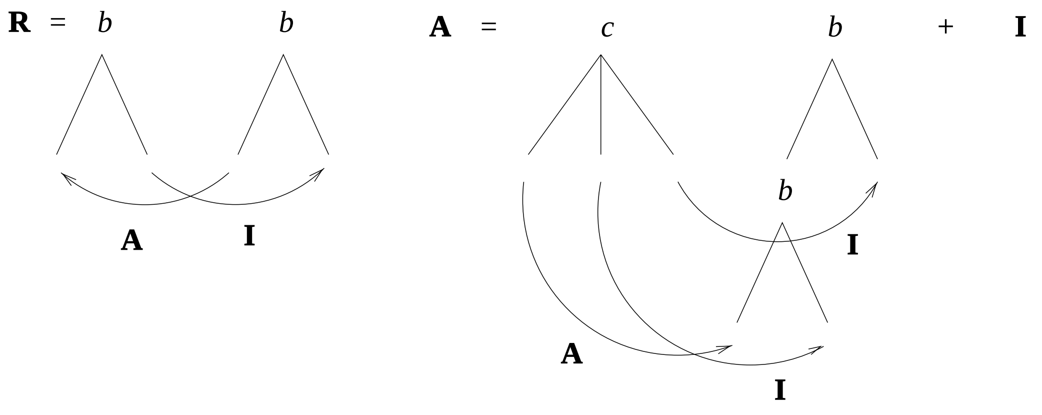

Figure 6: The grammar G.

Coming from the other production of $\mathbf{A}$, we get

$$\mathbf{L} \longrightarrow c(U_1\, U_2\, J_1)\, U_3\, J_2$$

where $U_1 U_2 U_3$ denotes $A_1\, I_1\, b(A_2\, I_2) \circ A'_2 A'_1$ and $J_1 J_2$ denotes $I_1 I_2 \circ I'_1 I'_2$. Knowing $\mathbf{I} \circ \mathbf{I} = \mathbf{I}$, the second relation is simply $\mathbf{I}$. In the first relation, the variables can be

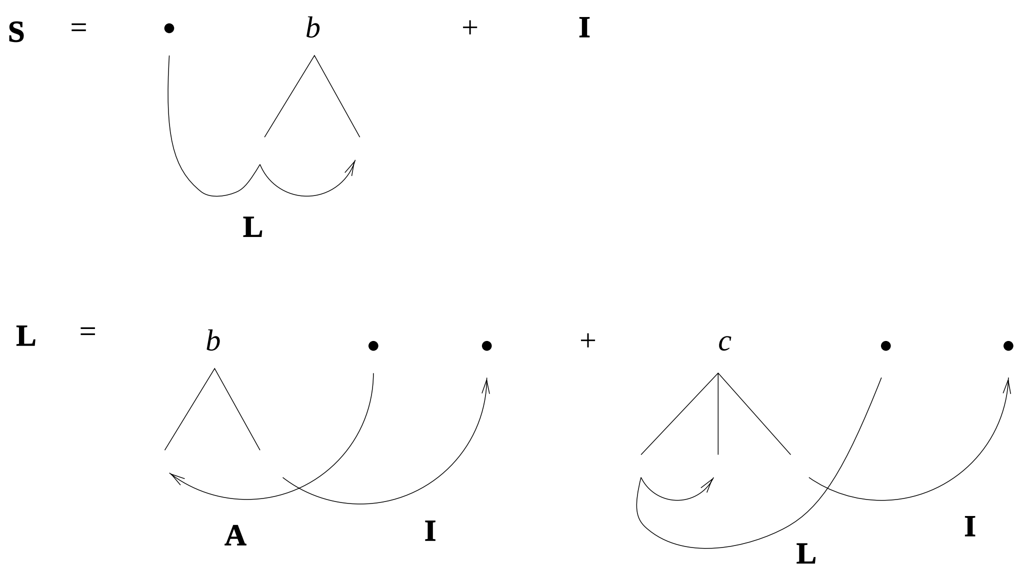

Figure 7: The result of composing $\mathbf{A} \circ \mathbf{R}$.

renamed by permuting $\mathbf{A}$ and $\mathbf{A}'$; we find, knowing $\mathbf{I} \circ \mathbf{I} = \mathbf{I}$ that this new non-terminal is in fact a permutation of $\mathbf{L}$: more precisely, $U_1U_2U_3 = L_2L_3L_1$. Thus the composition is complete. We get:

$$\begin{cases} \mathbf{S} &= L_1\, b(L_2\, L_3) \\ \mathbf{L} &= b(A_2\, I_1)\, A_1\, I_2 \;+\; c(L_2\, L_3\, I_1)\, L_1\, I_2 \end{cases}$$

depicted in figure 7.
The resulting grammar generates, among others, the pair of trees in figure 1.

Corollary 5.11 *The image and the inverse image of a recognizable language by a transduction is a recognizable language.*

Proof. Since the converse of a transduction is a transduction, it is enough to prove that the image is recognizable. The image of a rational language K by a transduction $\mathbf{R}$ is the range (i.e. the second projection) of the transduction $\mathrm{id}_K \circ \mathbf{R}$, which is a recognizable language. ■

6 Conclusion

Starting from the ordinary theory of rational trees, we extended it to relations over trees. Some results go through easily, like the correspondence between grammars generating relation, rational expressions describing them and equations of which they are solutions. We have also been able to derive a pumping lemma, to test non-rationality. This is not surprising inasmuch as key concepts like derivation trees carry over without difficulty.

A few pitfalls should nevertheless be advertised. The main one concerns the restriction of our rational relations to the case of unary relations: an equational unary relation is not a rational (or in this case, recognizable) forest.

The stability under residuals is not straightforward either. It is a case where, starting from a set of trees, the removal of a prefix tree leaves a set of tuples. An expected result is that residuals of rational languages by a finite set of tuples are again rational languages.

Finally, composition is also fairly different from the case of free monoids. In general, the composition of two rational relations, even with recognizable projections, is not a rational relation. Nevertheless, we have been able to isolate a subset of these relations encompassing all known transductions preserved by composition and giving a recognizable image of a recognizable set of trees. This subset, when restricted to trees of arity zero or one, coincides with the rational transductions of words; and it contains also the transductions of Arnold and Dauchet [1982]. It contains also the logically defined transductions of Courcelle [1994] when they are restricted to trees. The question remains whether this subset is recursive. The probable answer is no, but it remains to be proved.

References

[1] A. Arnold, M. Dauchet, Morphismes et bimorphismes d'arbres, *Theoret. Comput. Sci.* **20** (1982) 33–93.

[2] J. Berstel, *Transduction of context-free languages,* Taubner Studienbücher (1979).

[3] B. Courcelle, Monadic second-order definable graph transductions: a survey, *in Theoret. Comput. Sci.* **126** (1994) 53–75.

[4] M. Dauchet, *Transductions de forêts, bimorphismes de magmoïdes,* Thèse, Université de Lille 1 (1977).

[5] M. Dauchet, S. Tison, Algebraic complexity of tree languages, *in Tree automata and languages,* M. Nivat & A. Podelski ed., Elsevier (1992).

[6] M. Dauchet, S. Tison, T. Heuillard, P. Lescanne, Decidability or the confluence of ground term rewriting systems, *INRIA report* **675** (1987).

[7] N. Dershowitz, J.-P. Jouannaud, Rewriting systems, *in Handbook of Theoretical Computer Science,* J. van Leeuwen ed., Elsevier (1990).

[8] J. Engelfriet, Bottom-up and top-down tree transformations — a comparison, *Math. Systems Theory* **9** (1975) 198–231.

[9] J.Engelfriet, A Greibach normal form for context-free graph grammars, *Lecture Notes Comput. Sci.* **623** (1992) 138–149.

[10] J. Engelfriet, G. Rozenberg, G Slutzki, Tree transducers, L-systems, and two-way machines, *J. Comput. Syst. Sci.* **20** (1980) 150–202.

[11] J. Engelfriet, H. Vogler, Modular tree transducers, *Theoret. Comput. Sci.* **78** (1991) 267–303.

[12] F. Gecseg, M. Steinby, *Tree automata,* Akademiai Kiado, Budapest (1984).

[13] A. Grazon, J.-C. Raoult, Equational sets of tree-vectors, *IRISA report* **563**, Rennes (1990).

[14] A. Habel, H.J. Kreowski, Characteristics of graph languages generated by edge replacement, *Theoret. Comput. Sci.* **51** (1987) 81–115.

[15] J.E. Hopcroft, J.D. Ullman, *Introduction to automata theory, languages and computation,* Addison-Wesley (1979).

[16] J. Mezei, J.B. Wright, Algebraic automata and context-free sets, *Inform. Control* **11** (1967) 3–29.

[17] C. Pair, A. Quéré, Définition et étude des bilangages réguliers, *Inform. Control* **13** (1968) 565–593.

[18] J.-C. Raoult, A survey of tree-transductions, *in Tree automata and languages,* M. Nivat & A. Podelski ed., Elsevier (1992) 311–326 (and *report* **1410** INRIA-Rennes (1991)).

[19] P. P. Schreiber, Tree-transducers and syntax-connected transductions, *Lecture Notes Comput. Sci.* **33** (1975) 202–208.

[20] M. Steinby, On certain algebraically defined tree transformations, *in Proc. Coll. Math. Janos Bolyai on Algebra, Combinatorics and Logic in Computer Science*, Györ, Hungary (1983) 745–764.

[21] H. Vogler, Basic tree transducers, *J. Comput. Syst. Sci.* **34** (1987) 87–128.

Jean-Claude Raoult
IRISA, Campus de Beaulieu
Université de Rennes 1
F-35042 RENNES (France)
Jean-Claude.Raoult@univ-Rennes1.fr