MASTER'S DEGREE THESIS

MSc in Applied Mathematics

Title: Too much duplicating patterns represent non-regular languages

Author: Ramos Garrido, Lander

Advisor: Godoy Balil, Guillem

Department: Llenguatges i Sistemes Informàtics

Academic year: 2010/11

Facultat de Matemàtiques i Estadística

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Tesi de màster

Too Much Duplicating Patterns Represent Non-Regular Languages

Lander Ramos Garrido

Advisor: Guillem Godoy Balil

Llenguatges i Sistemes Informàtics

Abstract

A constrained term pattern $s : \varphi$ represents the language of all instances of the term s satisfying the constraint φ. For each variable in s, this constraint specifies the language of its allowed substitutions.

Regularity of languages represented by sets of patterns has been studied for a long time. This problem is known to be co-NP-complete when the constraints allow each variable to be replaced by any term over a fixed signature, and EXPTIME-complete when the constraints restrict each variable to a regular set.

Duplicated variables in the terms of the patterns are often the cause of non-regularity. This is because duplications force to test equalities between subterms. In fact, for the specific classes of constraints mentioned above, if all patterns are linear, then the represented language is necessarily regular. In this paper we focus on the opposite case, that is when there are patterns with "too much" duplicated variables.

We prove that when each pattern of a set has a duplicated variable constrained to an infinite language, then the language represented by the set is necessarily non-regular. We prove this result independently on the kind of constraints used, just assuming that they are mappings from variables to arbitrary languages.

Our result provides an efficient procedure for detecting, in some cases, non-regularity of images of regular languages under tree homomorphisms.

Contents

Introduction

A constrainted term pattern (or just a pattern) is a pair $s : \varphi$, where s is a term and φ maps each variable occurring in s to a term language. The pattern $s : \varphi$ represents the language of all terms obtained from s by replacing each variable x occurring in s by a term in $\varphi(x)$. Thus, φ constraints the instances of s by restricting the possible substitutions of each variable.

Patterns are a widely used formalism in computer science to represent languages. As with many other representation formalisms, one is frequently interested in solving questions like whether a given term belongs to a language, or set inclusion between languages. Another important question is to determine whether the language represented by a set of patterns is (tree) regular, i.e. recognizable by a tree automaton [**CDG$^+$07**]. This question is relevant because regularity allows a more tractable representation formalism with better properties for the represented language.

The expressive power and decidable properties of patterns depend on the mechanism used to describe the constraints. For example, for unrestricted patterns, i.e. when we allow to replace each variable by any term over a fixed signature, regularity has been proved co-NP-complete [**VG92, KR99**]. And it is EXPTIME-complete when the constraints restrict each variable to a regular set [**GGM09**].

Terms with variables occurring at least twice in the term and constrained to an infinite language are often the cause of non-regularity. For example, a pattern of the form $f(x, x) : \varphi$, where φ maps x to an infinite language, represents a non-regular set, since tree automata have a finite number of states and cannot test for equalities between subterms.

In contrast to patterns with duplicated variables, linear patterns, i.e. the ones where each variable occurs at most once in the term, increase the chance of regularity. For example, for unrestricted patterns and for patterns with regular constraints, if all patterns are linear, then the represented language is necessarily regular.

In this paper we focus on proving non-regularity when the patterns of a given set have "too much" duplicated variables. More precisely, we prove that when each pattern of a set has a duplicated variable constrained to an infinite language, then the language represented by the set is necessarily non-regular. This property holds

independently on the kind of constraints used, i.e. we just assume that they are mappings from variables to arbitrary languages.

Finally, we apply our result to show an efficient (linear time) algorithm that detects, in some cases, non-regularity of images of regular languages under tree homomorphisms. The general case of testing regularity of the image of a regular language by a tree homomorphism has been proven to be decidable, but its time complexity has not yet been proved better than a tower of three exponentials [**GGRÀ10**]. In fact, this more general problem is EXPTIME-hard [**GGM09**].

The results presented here have been obtained working with Carles Creus and Guillem Godoy, and have been submitted to the Journal of Applicable Algebra in Engineering Communication and Computing [**CGR**].

The paper is structured as follows. In Chapter 1 we present basic preliminar definitions used in the rest of the paper. In Chapter 2 we define pattern, too much duplicating sets of patterns, and prove that they always represent non-regular languages. In Chapter 3 we apply our result on patterns to detect, in some cases, the non-regularity of the image of a regular language by a tree homomorphism. In Chapter 4 we conclude.

Chapter 1
Preliminaries

We denote by $\mathbb{N}$ the set of positive integers. We denote the set of finite sequences over $\mathbb{N}$ by $\mathbb{N}^*$, and the empty sequence by λ. We denote such sequences as positive integers separated by points, as in the following example:

Example 1.1. *Examples of sequences in* $\mathbb{N}^*$ *are* λ, 1, 2, 10, 1.1, 2.4 *and* $1.2.3.\ldots.n$ *for* $n \in \mathbb{N}$. *Infinite sequences like* $1.1.1.\ldots.1.\ldots$ *are not in* $\mathbb{N}^*$.

The powerset of a set S is denoted by 2^S.

1.1. Terms

A *ranked signature* is a couple $(\Sigma, Arity)$, where Σ is a finite set and *Arity* is a mapping from Σ into $\mathbb{N}$. The *arity* of a symbol $f \in \Sigma$ is $Arity(f)$. The set of symbols of arity m is denoted by $\Sigma^{(m)}$. We sometimes denote Σ explicitly as $\{f_1 : m_1, \ldots, f_n : m_n\}$, where $f_1, \ldots, f_n$ are the function symbols and $m_1, \ldots, m_n$ are the corresponding arities. Symbols in $\Sigma^{(0)}$, called *constants*, are denoted by a, b, c, d, e, with possible subscripts. Elements of arity p are called $p-ary$ symbols. We assume that Σ contains at least one constant. The elements of a set $\mathcal{X}$ of variable symbols are denoted by x, y, z with possible subscripts. We assume that $\mathcal{X}$ and Σ are disjoint.

The set $\mathcal{T}(\Sigma, \mathcal{X})$ of *terms* over Σ and $\mathcal{X}$, is the smallest set satisfying:

- $\Sigma^{(0)} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
- $\mathcal{X} \subseteq \mathcal{T}(\Sigma, \mathcal{X})$
- if $m \geq 1$, $f \in \Sigma^{(m)}$, and $t_1, \ldots, t_m \in \mathcal{T}(\Sigma, \mathcal{X})$ then $f(t_1, \ldots, t_m) \in \mathcal{T}(\Sigma, \mathcal{X})$

The set of variables occurring in a term t is denoted $\texttt{vars}(t)$. A term t is called *linear* if each variable occurs at most once in t. A term t is called *ground* if t contains no variables. The set of all ground terms over Σ is denoted $\mathcal{T}(\Sigma)$.

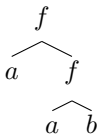
Example 1.2. *Let* $\Sigma = \{f : 2, a : 0, b : 0\}$ *and* $\mathcal{X} = \{x, y\}$. *Examples of terms in* $\mathcal{T}(\Sigma, \mathcal{X})$ *are:*

- $t_1 = a$

- $t_2 = f(a, a)$
- $t_3 = f(b, x)$
- $t_4 = f(a, f(a, b))$
- $t_5 = f(f(a, x), f(b, x))$

Examples like f, $x(a)$ or $g(c)$ are not terms in $\mathcal{T}(\Sigma, \mathcal{X})$, since either the arity does not match, or there are symbols not in Σ nor $\mathcal{X}$.

Terms t_1, t_2 and t_4 are ground terms. The term t_3 is linear, but the term t_5 is non linear. Terms can be represented in a graphical way. For example, the term t_4 is represented by:



1.2. Positions

A *position* is a sequence of natural numbers, i.e. an element of $\mathbb{N}^*$. Given two positions p and p', we denote as $p.p'$ the concatenation of the positions p and p'.

The set of positions of a term t, denoted $\texttt{Pos}(t)$, is the smallest set defined by:

- $\lambda \in \texttt{Pos}(t)$
- If $t = f(t_1, \ldots, t_m)$ and $m \geq 1$ then $\forall i \in \{1, \ldots, m\}$, $\{i.p \mid p \in \texttt{Pos}(t_i)\} \subset \texttt{Pos}(t)$.

Example 1.3. *The positions of the terms in Example 1.2 are:*

- $\texttt{Pos}(t_1) = \{\lambda\}$
- $\texttt{Pos}(t_2) = \{\lambda, 1, 2\}$
- $\texttt{Pos}(t_3) = \{\lambda, 1, 2\}$
- $\texttt{Pos}(t_4) = \{\lambda, 1, 2, 2.1, 2.2\}$
- $\texttt{Pos}(t_5) = \{\lambda, 1, 1.1, 1.2, 2, 2.1, 2.2\}$

The *length* of a position is denoted as $|p|$. It can be defined recursively as $|\lambda| = 0$ and $|i.p| = 1 + |p|$. A position p_1 is a *prefix* of a position p, denoted $p_1 \leq p$, if there is a position p_2 such that $p_1.p_2 = p$ holds. Also, p_1 is a *proper prefix* of p, denoted $p_1 < p$, if $p_1 \leq p$ and $p_1 \neq p$ hold. Two positions p, p' are parallel, denoted by $p||p'$, if $p \not\leq p'$ and $p' \not\leq p$ hold.

1.3. Subterms

The *subterm* of a term t at a position p, denoted $t|_p$, is defined recursively as follows:

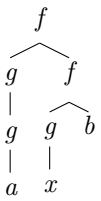
- $t|_\lambda = t$
- $f(t_1, \ldots, t_m)|_{i.p} = t_i|_p$

The *replacement* of a term t at a position p by a term s, denoted $t[s]_p$, is defined recursively as follows:

- $t[s]_\lambda = s$
- $f(t_1, \ldots, t_m)[s]_{i.p} = f(t_1, \ldots, t_i[s]_p, \ldots, t_m)$

The *root* of a term $t = f(t_1, \ldots, t_m)$ is $\texttt{root}(t) = f$. A term t over Σ can be seen as a function from its set of positions into Σ. For this reason, we shall denote by $t(p)$ the symbol labeling t at position p, i.e. $t(p) = \texttt{root}(t|_p)$.

Example 1.4. *Let $\Sigma = \{f : 2, g : 1, a : 0\}$ and $\mathcal{X} = \{x\}$ be signatures. Let t be the following term over $\mathcal{T}(\Sigma, \mathcal{X})$:*



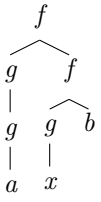
The subterm of t at position 2.1 is $s := t|_{2.1} = g(x)$. The replacement $t[s]_2$ is $f(g(g(a)), g(x))$. The root of t is f and the root of s is g. We can also see s as a function from $\{\lambda, 1\}$ to $\Sigma \cup \mathcal{X}$, defined as $s(\lambda) = g$ and $s(1) = x$.

1.4. Functions on terms

The *height* of a term t, denoted $\texttt{height}(t)$, is defined recursively as follows:

- $\texttt{height}(t) = 0$ if $t \in \Sigma^{(0)}$,
- $\texttt{height}(t) = 0$ if $t \in \mathcal{X}$,
- $\texttt{height}(f(t_1, \ldots, t_m)) = 1 + \texttt{max}(\texttt{height}(t_1), \ldots, \texttt{height}(t_m))$ if $\texttt{root}(t) \in \Sigma^{(i)}$, with $i \geq 1$.

Example 1.5. *Let $\Sigma = \{f : 2, g : 1, a : 0\}$ and $\mathcal{X} = \{x\}$ be signatures. Let t be the following term over $\mathcal{T}(\Sigma, \mathcal{X})$:*



Then $\texttt{height}(t) = 3$. The corresponding heights of each subterm are:

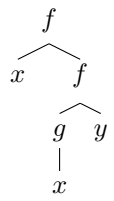
- $\texttt{height}(t|_1) = 2$
- $\texttt{height}(t|_{1.1}) = 1$
- $\texttt{height}(t|_{1.1.1}) = 0$

- $\texttt{height}(t|_2) = 2$
- $\texttt{height}(t|_{2.1}) = 1$
- $\texttt{height}(t|_{2.1.1}) = 0$
- $\texttt{height}(t|_{2.2}) = 0$

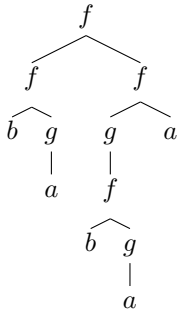
1.5. Substitutions

A *substitution* σ is a mapping from variables in $\mathcal{X}$ to terms in $\mathcal{T}(\Sigma)$. We denote a substitution σ as $\sigma = \{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$, where $x_i \in \mathcal{X}$ and $t_i \in \mathcal{T}(\Sigma)$ for $i \in \{1, \ldots, n\}$, meaning that $\sigma(x_i) = t_i$. A substitution can be extended to terms in $\mathcal{T}(\Sigma, \mathcal{X})$ in such a way that by $\sigma(f(t_1, \ldots, t_m)) = f(\sigma(t_1), \ldots, \sigma(t_m))$ for terms different from variables.

Example 1.6. *Let* $\Sigma = \{f : 2, g : 1, a : 0\}$ *and* $\mathcal{X} = \{x, y\}$ *be signatures. Let* σ *be* $\{x \mapsto f(b, g(a)), y \mapsto a\}$*, and let* t *be the following term:*



Then $\sigma(t)$ *will be the following term:*



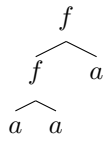
1.6. Automata

A *tree automaton* (TA) is a tuple $A = \langle Q, \Sigma, F, \Delta \rangle$, where Q is a set of states, Σ is a ranked signature, $F \subseteq Q$ is the subset of final states (also called accepting states), and Δ is a set of rules of the form $f(q_1, \ldots, q_m) \to q$, where $q_1, \ldots, q_m, q \in Q$ and $f \in \Sigma^{(m)}$. A Deterministic and Complete Tree Automata (DCTA) satisfies that given a symbol $f \in \Sigma^{(m)}$, and a sequence of m states $q_1, \ldots, q_m$, there exists a unique rule $(f(q_1, \ldots, q_m) \to q)$ in Δ.

The *size* of A is defined by $|A| = |Q|$. Note that the actual size of an automaton (i.e., the number of rules) might be exponential on the number of states, but in this paper we will have enough with this definition.

Tree automata over a signature Σ run on ground terms over Σ. An automaton starts at the leaves and moves upward, associating along a run a state with each subterm inductively. Intuitively, we can apply a rule $r := (f(q_1, \ldots, q_m) \to q) \in \Delta$ to a term t in $\mathcal{T}(\Sigma \cup Q)$ in the following way: if there is a position $p \in \mathtt{Pos}(t)$ such that $t|_p = f(q_1, \ldots, q_m)$, then we can apply the rule r to t, obtaining a term $t' = t[q]_p$.

Example 1.7. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ a TA where $F = Q = \{q\}$, $\Sigma = \{f : 2, a : 0\}$ and $\Delta = \{a \to q, f(q, q) \to q\}$. Then, we can apply rules to the following term t:*



Obtaining the following sequence of terms:

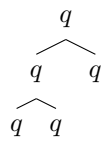
$$f(f(a,a),a) \to f(f(a,a),q)) \to f(f(q,a),q)) \to f(f(q,q),q) \to f(q,q) \to q$$

This intuitive concept can be expressed using runs, defined as follows:

A *run* r of a TA A on a term t is a function $r : \mathtt{Pos}(t) \to Q$ satisfying that, for each position $p \in \mathtt{Pos}(t)$, if $t|_p$ is of the form $f(t_1, \ldots, t_m)$, then there exists a rule of the form $f(q_1, \ldots, q_m) \to q$ in Δ such that $r(p.1) = q_1, \ldots, r(p.m) = q_m$ and $r(p) = q$ hold.

Note that, in the case of Deterministic Tree Automata (DCTA), given a term t there exists a unique run r on t. This fact can be proven inductively, starting on the leaves of the term, where we will only be able to apply one rule. In the case of the term t of the Example 1.7, the run r of A on t is defined by $r(\lambda) = r(1) = r(1.1) = r(1.2) = r(2) = q$.

A run can also be expressed as a labeled tree, i.e. a term with unfixed rank. The set of positions of a run r on a term t will be $\mathtt{Pos}(t)$, and the label of each position $p \in \mathtt{Pos}(t)$ will be $r(p)$. In the previous example, we can express r as the following tree:



A run r is accepting if $r(\lambda)$ is accepting. A term t is accepted or recognized by A if there exists an accepting run of A on t. In the case of DTA, a term t is accepted by A if the unique run r of A on t is such that $r(\lambda) \in F$

Example 1.8. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TA where $Q = \{q_0, q_1\}$, $F = \{q_0\}$, $\Sigma = \{f : 2, a : 0, b : 0\}$, and Δ is the following set of rules:*

- $a \to q_1$
- $b \to q_0$
- $f(q_0, q_0) \to q_0$
- $f(q_0, q_1) \to q_1$
- $f(q_1, q_0) \to q_1$
- $f(q_1, q_1) \to q_0$

Note that A only accepts the terms over Σ having an even number of a's.

Example 1.9. *Let $A = \langle Q, \Sigma, F, \Delta \rangle$ be a TA where $Q = \{q_a, q_f, q_g\}$, $F = \{q_g\}$, $\Sigma = \{a : 0, g : 1, f : 2\}$ and Δ is the following set of rules:*

- $a \to q_a$
- $g(q_a) \to q_g$
- $g(q_f) \to q_g$
- $g(q_g) \to q_g$
- $f(q_a, q_a) \to q_f$
- $f(q_a, q_f) \to q_f$
- $f(q_a, q_g) \to q_f$
- $f(q_f, q_a) \to q_f$
- $f(q_f, q_f) \to q_f$
- $f(q_f, q_g) \to q_f$
- $f(q_g, q_a) \to q_f$
- $f(q_g, q_f) \to q_f$
- $f(q_g, q_g) \to q_f$

Some examples of terms recognized by A are $g(a)$, $g(g(a))$, $g(f(a, a))$ and $g(f(a, g(a)))$, with runs $q_g(q_a)$, $q_g(q_g(q_a))$, $q_g(q_f(q_a, q_a))$ and $q_g(q_f(q_a, q_g(q_a)))$ respectively. On the other hand, it is not difficult to see that terms like $f(a, a)$, a, $f(g(a), a)$ or $f(a, f(a, a))$ are not recognized by A, since there does not exist a run r on any of those terms holding $r(\lambda) = q_g$.

A *language* over Σ is a set of ground terms. The language recognized by A, denoted $\mathcal{L}(A)$, is the set of terms accepted by A. By $\mathcal{L}(A, q)$ we denote the set of terms for which there exists a run r of A holding $r(\lambda) = q$.

Example 1.10. *In the Example 1.9 it is straightforward to see that $\mathcal{L}(A, q) = \{a\}$, $\mathcal{L}(A, q_f)$ is the set of terms whose root is f, and $\mathcal{L}(A, q_g)$ is the set of terms whose root is g. Therefore the language recognized by A is the language $\{g(t) \mid t \in \mathcal{T}(\Sigma)\}$.*

We say that a language L is *regular* if there exists a TA A such that $\mathcal{L}(A) = L$.

Example 1.11. *Some examples of regular languages are $\emptyset$, recognized by an automaton with no accepting states, $\mathcal{T}(\Sigma)$, recognized by an automaton in which every state is accepting, all the finite languages, and the set of terms whose root belongs to a subset of Σ, like the one of the Example 1.9.*

A classical example of non-regular language is the set of complete terms over $\Sigma = \{f : 2, a : 0\}$, i.e., the smallest language L such that:

- *$a \in L$*
- *If $t \in L$, then $f(t, t) \in L$*

This language is well known to be non-regular, as stated in [**GGJ09**]. *In order to prove this fact, we proceed by contradiction and supose that there exists a DCTA A recognizing L. The size of A must be finite. Since there are infinite complete terms, by the pigeonhole principle there exist two different terms,* t_1 *and* t_2*, having accepting runs reaching the same state. Since* $f(t_1, t_1)$ *is also complete, it must be accepted by A. Hence, the uncomplete tree* $f(t_1, t_2)$ *must also be accepted, which is a contradiction.*

Given a TA A, a term t and a run r of A on t, we define $r|_p$ as the run of A on $t|_p$ described by $r|_p(p') = r(p.p')$. In addition, given a run r' of A on a term t' and satisfying $r'(\lambda) = r(p)$, we define $r[r']_p$ as the run of A on $t[t']_p$ described by $r[r']_p(p') = r(p')$ for positions p' holding $p \not\leq p'$, and by $r[r']_p(p.p') = r'(p')$.

1.7. Tree Homomorphisms

Tree homomorphisms are a generalization of homomorphisms for words to the case of arbitrary ranked alphabets. In the word case, it is known that the class of regular sets is closed under application of homomorphisms and inverse homomorphisms. The situation is different in the tree case because whereas recognizable tree languages are closed under inverse homomorphisms, they are closed only under particular homomorphisms, like for example linear homomorphisms (duplication of terms does not take place). First, we define tree homomorphisms.

Let Σ, Σ' be two signatures. A *tree homomorphism* is a function $\phi : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ which can be defined as follows.

Let $\mathcal{X}_m$ represent the set of variables $\{x_1, \ldots, x_m\}$ for each natural number m. The definition of a tree homomorphism $\phi : \mathcal{T}(\Sigma) \to \mathcal{T}(\Sigma')$ requires to define another function ϕ_Σ, which given a symbol $f \in \Sigma^{(m)}$, it maps f into a term $t_f \in \mathcal{T}(\Sigma', \mathcal{X}_m)$

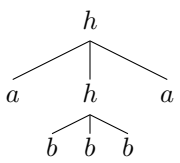
After that, we can define recursively $\phi(t)$, where $t \in \mathcal{T}(\Sigma)$, as follows:

- $\phi(a) = t_a$ if $a \in \Sigma^{(0)}$
- $\phi(f(t_1, \ldots, t_m)) = \{x_1 \mapsto \phi(t_1), \ldots, x_m \mapsto \phi(t_m)\}(t_f)$ otherwise

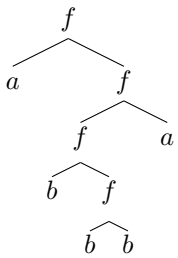
Example 1.12. *Let* $\Sigma = \{h : 3, a : 0, b : 0\}$ *and* $\Sigma' = \{f : 2, a : 0, b : 0\}$. *Let us consider the tree homomorphism* ϕ *determined by* ϕ_Σ *defined by:*

- $\phi_\Sigma(h) = f(x_1, f(x_2, x_3))$
- $\phi_\Sigma(a) = a$
- $\phi_\Sigma(b) = b$

For instance, let t *be the following term over* Σ*:*



Then, $\phi(t)$ will be the following term over Σ':



As we can see, the homomorphism ϕ defines a transformation from ternary trees into binary trees.

We can extend the definition of a homomorphism to languages in the natural way. Given a language $L \subseteq \mathcal{T}(\Sigma)$, we define $\phi(L)$ as $\{\phi(t) \mid t \in L\}$. The following example shows that tree homomorphisms do not always preserve recognizability.

Example 1.13. *Let $\Sigma = \{g : 1, a : 0\}$ and $\Sigma' = \{f : 2, a : 0\}$. Let L the language $g^n(a)$, i.e., $\mathcal{T}(\Sigma)$. It is straightforward to see that L is regular, beause it is the language recognized by $A = \langle\{q\}, \Sigma, \{q\}, \Delta\rangle$, where $\Delta = \{a \to q, g(q) \to q\}$.*

Now let ϕ be the homomorphism determined by ϕ_Σ defined by:

- $\phi_\Sigma(a) = a$
- $\phi_\Sigma(g) = f(x_1, x_1)$

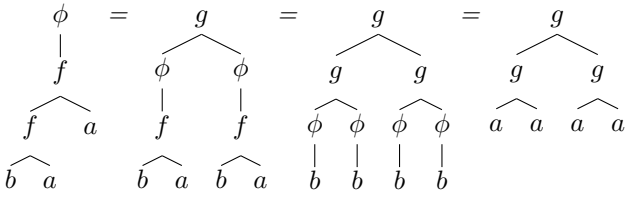
It is easy to see that $\phi(L)$ is the set of complete terms over $\mathcal{T}(\Sigma')$. In Example 1.11 we saw that this language is not regular. Therefore homomorphisms do not always preserve regularity.

Example 1.14. *Let $A = \langle Q, \Sigma, F, \Delta\rangle$ be the automaton of Example 1.8. Consider the tree homomorphism $\phi : \mathcal{T}(\Sigma) \to \mathcal{T}(\{g : 2, a : 0\})$ determined by ϕ_Σ defined by:*

- $\phi_\Sigma(a) = a$
- $\phi_\Sigma(b) = a$
- $\phi_\Sigma(f) = g(x_1, x_1)$

Note that $\phi(\mathcal{L}(A))$ is again the set of complete trees over binary g and nullary a, which is not regular, according to Example 1.11.

We show graphically how this tree homomorphism transforms a term in $\mathcal{T}(\Sigma)$ to a term in $\mathcal{T}(\{g : 2, a : 0\})$:



A tree homomorphism ϕ is linear if for each $f \in \Sigma$ of arity m, $\phi_\Sigma(f) = t_f$ is a linear term in $\mathcal{T}(\Sigma', \mathcal{X}_m)$, i.e., each variable appears at most once in t_f. If ϕ is a linear tree homomorphism and L is a regular language, then $\phi(L)$ is a regular language. The proof of this fact can be found in [**CDG$^+$07**].

Chapter 2
Non-regular sets of patterns

In this chapter we will prove the non-regularity of a set of patterns when they hold some properties, namely that they have duplicated variables constrained to infinite languages. We first give a formal definition of our problem in Section 2.1. In Section 2.2 we define some properties on ground terms that will easen the proof of non-regularity, which is given in Section 2.3.

2.1. Too much duplicating sets of patterns

We define a pattern as a term plus constraints for the variables occurring in the term. The constraint of a variable is an arbitrary language of ground terms. An instance of the pattern is obtained by applying a substitution, where each variable must be replaced by a term belonging to the corresponding constraint to the variable.

Definition 2.1. A *pattern* is a pair $s : \varphi$ where s is a term in $\mathcal{T}(\Sigma, \mathcal{X})$ and φ is a function $\varphi : \mathtt{vars}(s) \to 2^{\mathcal{T}(\Sigma)}$, i.e. a function mapping each variable occurring in s to an arbitrary language of ground terms.

The language defined by $s : \varphi$, denoted $\mathcal{L}(s : \varphi)$, is $\{t \mid \exists \sigma : (t = \sigma(s) \wedge \forall x \in \mathtt{vars}(s) : (\sigma(x) \in \varphi(x)))\}$. A term t is an instance of $s : \varphi$ if t is in $\mathcal{L}(s : \varphi)$. The language defined by a set of patterns S is $\bigcup_{s:\varphi \in S} \mathcal{L}(s : \varphi)$.

Example 2.2. *In all the examples of this section, we will use the signature* $\Sigma = \{a : 0,\ f : 2\}$.

Consider the pattern $s : \varphi = x : \{x \mapsto \{f(t, t) \mid t \in \mathcal{T}(\Sigma)\}\}$. *Note that* $\mathcal{L}(s : \varphi) = \{f(t, t) \mid t \in \mathcal{T}(\Sigma)\}$ *is not regular althought* s *is linear.*

Consider the set of patterns $S = \{f(x, x) : \{x \mapsto \mathcal{T}(\Sigma)\},\ y : \{y \mapsto \mathcal{T}(\Sigma)\}\}$ *Note that* $\mathcal{L}(S) = \mathcal{T}(\Sigma)$ *is regular although* S *contains a pattern with a duplicated variable constrained to an infinite language. This is because the language of the other pattern is regular and contains the language of the first pattern.*

Consider the set of patterns $S = \{f(x, x) : \{x \mapsto \mathcal{T}(\Sigma)\},\ y : \{y \mapsto \{f(t_1, t_2) \mid t_1, t_2 \in \mathcal{T}(\Sigma) \wedge t_1 \neq t_2\}\}\}$. *Note that* $\mathcal{L}(S) = \mathcal{T}(\Sigma) \setminus \{a\}$ *is regular although* S

contains two patterns with disjoint associated languages, and one of them has a duplicated variable constrained to an infinite language.

In this paper we focus on patterns that have duplicated variables constrained to an infinite language, which are defined as follows.

Definition 2.3. A pattern $s : \varphi$ is *too much duplicating* if $\mathcal{L}(s : \varphi)$ is not empty and there exists a variable x occurring in at least two different positions of s such that $|\varphi(x)|$ is infinite.

A non-empty finite set of patterns S is *strongly too much duplicating* if each $s : \varphi$ in S is too much duplicating.

A non-empty finite set of patterns S is *weakly too much duplicating* if for each $s : \varphi$ in S, either $s : \varphi$ is too much duplicating or $\mathcal{L}(s : \varphi)$ is finite, and at least one of the patterns in S is too much duplicating.

2.2. Uneven terms

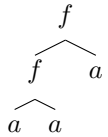
In this Subsection we give some properties on ground terms. These properties are later used in the proof of non-regularity of strongly too much duplicating set of patterns. We start by defining a set of positions of a term having "tall enough" subterms.

Definition 2.4. Let t be a term in $\mathcal{T}(\Sigma)$. Let h, H be natural numbers. We define $\mathtt{Pos}^h_H(t)$ as the set of positions p in $\mathtt{Pos}(t)$ satisfying $|p| \leq h$ and $\mathtt{height}(t|_p) \geq H$.

Note that $\mathtt{Pos}^h_H(t)$ is closed by prefix and $(\mathtt{height}(t) < H \Leftrightarrow \mathtt{Pos}^h_H(t) = \emptyset)$ holds. The following statements are also straightforward.

Example 2.5.

- *If* $t = f(a, a)$, $h = 2$ *and* $H = 3$, *then* $\mathtt{Pos}^h_H(t) = \emptyset$, *because* $\mathtt{height}(t) = 1 < H = 3$.
- *Let* t *be the following term:*



  Let $h = 1$ *and* $H = 2$, *then* $\mathtt{Pos}^h_H(t) = \{\lambda\}$ *because* $|\lambda| = 0 \leq h = 1$ *and* $\mathtt{height}(t|_\lambda) = 2 \geq H = 2$. *No other position* p *fullfils the conditions.*
- *Let* t *be as in the previous case,* $h = 1$ *and* $H = 1$, *then* $\mathtt{Pos}^h_H(t) = \{\lambda, 1\}$ *follows similarly to the previous case. But, if we take* $h = 0$ *and* $H = 1$, *then* $\mathtt{Pos}^h_H(t) = \{\lambda\}$ *since* $|1| = 1 > h = 0$.

Lemma 2.6. *Let* h *be a natural number. Then, there exists a natural number* K *such that, for all term* t *in* $\mathcal{T}(\Sigma)$ *and all natural number* H, $|\mathtt{Pos}^h_H(t)| < K$ *holds.*

PROOF. Since the signature Σ is finite and the arity of each function symbol is finite, then the number of different positions p satisfying $|p| \leq h$ in terms $t \in \mathcal{T}(\Sigma)$ is bounded. □

Lemma 2.7. *Let h, H be natural numbers. Let t, s be terms of $\mathcal{T}(\Sigma)$. Let p be a position of t satisfying $\mathtt{height}(t|_p) < \mathtt{height}(s)$ and $p \notin \mathtt{Pos}^h_H(t)$. Then $\mathtt{Pos}^h_H(t) \subseteq \mathtt{Pos}^h_H(t[s]_p)$.*

PROOF. We consider any position $\bar{p}$ in $\mathtt{Pos}^h_H(t)$ and prove $\bar{p} \in \mathtt{Pos}^h_H(t[s]_p)$. Note that $\mathtt{height}(t|_{\bar{p}}) \geq H$ and $|\bar{p}| \leq h$. Moreover, since $p \notin \mathtt{Pos}^h_H(t)$ holds, either $|p| > h$ or $\mathtt{height}(t|_p) < H$ hold. Thus, p cannot be a prefix of $\bar{p}$. Moreover, since $\mathtt{height}(t|_p) < \mathtt{height}(s)$ holds, $\mathtt{height}(t|_{\bar{p}}) \leq \mathtt{height}(t[s]_p|_{\bar{p}})$ also holds. Thus, $H \leq \mathtt{height}(t[s]_p|_{\bar{p}})$ holds, and hence $\bar{p} \in \mathtt{Pos}^h_H(t[s]_p)$ follows. □

Now we define the set of uneven terms. An uneven term t is a ground term such that it is guaranteed that it has subterms with different height at the positions of $\mathtt{Pos}^h_H(t)$. This property is later used to show that a specific term cannot be an instance of some patterns with duplicated variables.

Definition 2.8. Let h, H be natural numbers. We define $\mathcal{U}^h_H$ as the set of terms t satisfying that for each pair of different positions $p_1, p_2 \in \mathtt{Pos}^h_H(t)$, $\mathtt{height}(t|_{p_1}) \neq \mathtt{height}(t|_{p_2})$ holds.

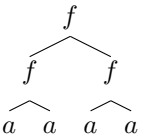
The following property and example are straightforward by the definition of $\mathcal{U}^h_H$.

Proposition 2.9. *Let h, H be natural numbers. Let t be a term satisfying $\mathtt{height}(t) < H$. Then $t \in \mathcal{U}^h_H$.*

PROOF. Since $\mathtt{height}(t) < H$, $\mathtt{Pos}^h_H(t)$ is empty, and hence, $t \in \mathcal{U}^h_H$ trivially holds. □

Example 2.10.

- *Let t be the following term:*



  Then $t \in \mathcal{U}^1_2$ holds because $\mathtt{Pos}^1_2(t) = \{\lambda\}$ holds and hence there are no two different positions in $\mathtt{Pos}^1_2(t)$. But $t \notin \mathcal{U}^1_1$ because $\mathtt{Pos}^1_1(t) = \{\lambda, 1, 2\}$ and $\mathtt{height}(t|_1) = \mathtt{height}(t|_2)$ holds.
- *If $t = f(a, a)$ then $t \in \mathcal{U}^h_2$, for all $h \geq 0$, as stated in Proposition 2.9.*

We now give two lemmas with important properties on uneven terms. First we show that the positions of a term having subterms with equal height cannot be in $\mathtt{Pos}^h_H$, if the term is uneven. Next, we show which conditions must hold in a replacement of a subterm in order to preserve the unevenness.

Lemma 2.11. *Let h, H be natural numbers. Let t be a term in $\mathcal{U}_H^h$. Let $p_1, \ldots, p_m$ be different positions in $\mathtt{Pos}(t)$, with $m \geq 2$, and satisfying $|p_1|, \ldots, |p_m| \leq h$ and $t|_{p_1} = \ldots = t|_{p_m}$. Then, $p_1, \ldots, p_m \notin \mathtt{Pos}_H^h(t)$ holds.*

PROOF. Since $t|_{p_1} = \ldots = t|_{p_m}$ holds, $\mathtt{height}(t|_{p_1}) = \ldots = \mathtt{height}(t|_{p_m})$ also holds. Hence, since $|p_1|, \ldots, |p_m| \leq h$, either all $p_1, \ldots, p_m$ are in $\mathtt{Pos}_H^h(t)$ or none of them is. The first case is not possible, since otherwise, all $\mathtt{height}(t|_{p_1}), \ldots, \mathtt{height}(t|_{p_m})$ should be different because t is in $\mathcal{U}_H^h$. □

Proposition 2.12. *Let t, s be terms in $\mathcal{T}(\Sigma)$, and let p be a position of t satisfying $\mathtt{height}(t) > |p| + \mathtt{height}(t|_p)$ and $\mathtt{height}(t) > |p| + \mathtt{height}(s)$. Then, $\mathtt{height}(t) = \mathtt{height}(t[s]_p)$ holds.*

PROOF. Note that $\mathtt{height}(t) = \mathtt{Max}_{\bar{p} \in \mathtt{Pos}(t)} |\bar{p}|$ holds. By the assumption $\mathtt{height}(t) > |p| + \mathtt{height}(t|_p)$ the position with maximum length in $\mathtt{Pos}(t)$ is also a position in $\mathtt{Pos}(t[s]_p)$. By the assumption $\mathtt{height}(t) > |p| + \mathtt{height}(s)$, there are no larger positions in $\mathtt{Pos}(t[s]_p)$. Thus, $\mathtt{height}(t) = \mathtt{height}(t[s]_p)$ follows. □

Lemma 2.13. *Let h, H be natural numbers. Let t be a term in $\mathcal{U}_H^h$. Let s be a term in $\mathcal{T}(\Sigma)$, and let p be a position of t satisfying $|p| \leq h$ and $\mathtt{height}(t|_p), \mathtt{height}(s) < H - h$. Then $t[s]_p \in \mathcal{U}_H^h$.*

PROOF. In order to conclude, we must prove that, for any two different positions p_1, p_2 in $\mathtt{Pos}_H^h(t[s]_p)$, $\mathtt{height}(t[s]_p|_{p_1}) \neq \mathtt{height}(t[s]_p|_{p_2})$ holds. To this end, it suffices to see that, for each position $\bar{p}$ in $\mathtt{Pos}_H^h(t[s]_p)$, $\bar{p}$ is also in $\mathtt{Pos}_H^h(t)$ and $\mathtt{height}(t[s]_p|_{\bar{p}}) = \mathtt{height}(t|_{\bar{p}})$ holds. This is vacuously true for positions $\bar{p}$ parallel with p. The case where p is a prefix of $\bar{p}$ is not possible because, since $\bar{p}$ is in $\mathtt{Pos}_H^h(t[s]_p)$, $\mathtt{height}(t[s]_p|_{\bar{p}}) \geq H \geq H - h > \mathtt{height}(s)$ holds. In the case where $\bar{p}$ is a proper prefix of p, $\mathtt{height}(t[s]_p|_{\bar{p}}) = \mathtt{height}(t|_{\bar{p}})$ follows from Proposition 2.12. □

2.3. Non-regularity proof

In order to prove non-regularity of a strongly too much duplicating set of patterns $S = \{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$, we proceed by contradiction by assuming that a tree automaton A recognizes $\mathcal{L}(S)$. We construct a specific term of the language, and thus recognized by A, and modify it in such a way that its set $\mathtt{Pos}_H^h$ increases until it is no longer an instance of any $s_i : \varphi_i$, but A still recognizes it. To this end, we add a new position p to $\mathtt{Pos}_H^h$ by replacing the subterm at position p by a big term. This big term is obtained by using the traditional concept of pumping, for which we introduce the following notation:

Definition 2.14. Let t be a term in $\mathcal{T}(\Sigma)$. Let A be a TA. Let r be a run of A on t. Let p_1, p_2 be positions in $\mathtt{Pos}(t)$ such that $p_1 < p_2$ and $r(p_1) = r(p_2)$. Then we recursively define $\mathtt{Pumping}(t, p_1, p_2, n) = \mathtt{Pumping}(t[t|_{p_1}]_{p_2}, p_1, p_2, n-1)$ for $n > 0$ and $\mathtt{Pumping}(t, p_1, p_2, 0) = t$. Similarly, we recursively define $\mathtt{Pumping}(r, p_1, p_2, n) = \mathtt{Pumping}(r[r|_{p_1}]_{p_2}, p_1, p_2, n-1)$ for $n > 0$ and $\mathtt{Pumping}(r, p_1, p_2, 0) = r$.

Note that $\mathtt{Pumping}(r, p_1, p_2, n)$ is a run of A on $\mathtt{Pumping}(t, p_1, p_2, n)$, and that when r is accepting, so is $\mathtt{Pumping}(r, p_1, p_2, n)$.

Lemma 2.15. *Let A be a TA. Let h, H be natural numbers. Let t be a term in $\mathcal{U}^h_H \cap \mathcal{L}(A)$. Let p be a position satisfying $|p| \leq h$ and $h + |A| \leq \mathtt{height}(t|_p) < H$.*

Then, there exists a term $\tilde{t}$ in $\mathcal{U}^h_H \cap \mathcal{L}(A)$ such that $\mathtt{Pos}^h_H(t) \subsetneq \mathtt{Pos}^h_H(\tilde{t})$.

Proof. Let r be an accepting run of A on t. Since $|p| \leq h$ and $h+|A| \leq \mathtt{height}(t|_p)$ hold, by the pigeonhole principle there exist positions p_0, p_1, p_2 satisfying $p \leq p_0 \leq p_1 < p_2$, $|p_0| = h$ and $r(p_1) = r(p_2)$.

We define $\tilde{t} = \mathtt{Pumping}(t, p_1, p_2, \mathtt{max}(\mathtt{height}(t), H))$. Since $\mathtt{Pumping}(r, p_1, p_2, \mathtt{max}(\mathtt{height}(t), H))$ is an accepting run of A on $\tilde{t}$, $\tilde{t}$ is in $\mathcal{L}(A)$.

Now, we prove $\mathtt{Pos}^h_H(t) \subsetneq \mathtt{Pos}^h_H(\tilde{t})$. First, note that all positions p' parallel with p_0 satisfy that $t|_{p'} = \tilde{t}|_{p'}$, thus $p' \in \mathtt{Pos}^h_H(t)$ if and only if $p' \in \mathtt{Pos}^h_H(\tilde{t})$, for such p'. Second, since $|p_0| = h$ holds, any position p' having p_0 as a proper prefix is neither in $\mathtt{Pos}^h_H(t)$ nor $\mathtt{Pos}^h_H(\tilde{t})$. Third, since $\mathtt{height}(\tilde{t}|_{p_0}) > H$ holds, any prefix $\bar{p}$ of p_0 is in $\mathtt{Pos}^h_H(\tilde{t})$. In summary, since p is a prefix of p_0 and not in $\mathtt{Pos}^h_H(t)$, it follows $\mathtt{Pos}^h_H(t) \subsetneq \mathtt{Pos}^h_H(\tilde{t})$.

It rests to see $\tilde{t} \in \mathcal{U}^h_H$, i.e. to prove that for two different positions $\bar{p}_1, \bar{p}_2$ in $\mathtt{Pos}^h_H(\tilde{t})$, $\mathtt{height}(\tilde{t}|_{\bar{p}_1}) \neq \mathtt{height}(\tilde{t}|_{\bar{p}_2})$ holds. This fact is vacuously true when either $\bar{p}_1 < \bar{p}_2$ or $\bar{p}_2 < \bar{p}_1$ or both $\bar{p}_1$ and $\bar{p}_2$ are parallel with p_0. Since $|p_0| = h$, the case where p_0 is a proper prefix of either $\bar{p}_1$ or $\bar{p}_2$ is not possible by the definition of $\mathtt{Pos}^h_H(\tilde{t})$. In the case where either $\bar{p}_1$ or $\bar{p}_2$, say $\bar{p}_1$, is a prefix of p_0, and $\bar{p}_2$ is not, this fact follows from $\mathtt{height}(\tilde{t}|_{\bar{p}_2}) \leq \mathtt{height}(t) < \mathtt{height}(\tilde{t}|_{p_0}) \leq \mathtt{height}(\tilde{t}|_{\bar{p}_1})$. □

Theorem 2.16. *Let $\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ be a strongly too much duplicating set of patterns over $\Sigma \cup \mathcal{V}$.*

Then, $\mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$ is not regular.

Proof. We prove it by contradiction by assuming that $\mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$ is regular. Let A be a TA satisfying $\mathcal{L}(A) = \mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$. Let h be $\mathtt{max}(\{\mathtt{height}(s_1), \ldots, \mathtt{height}(s_n)\})$, and let $x_1, \ldots, x_n$ be variables occurring at least twice in $s_1, \ldots, s_n$, respectively, and satisfying $|\varphi_1(x_1)| = |\varphi_2(x_2)| = \ldots = |\varphi_n(x_n)| = \infty$. Let $t_1, \ldots, t_n$ be terms in $\varphi_1(x_1), \ldots, \varphi_n(x_n)$, respectively, satisfying $\mathtt{height}(t_1), \ldots, \mathtt{height}(t_n) \geq h + |A|$. Let $h_{\mathtt{min}}$ be the height of a term with minimum height in $\mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$. Let H be $1 + \mathtt{max}(h_{\mathtt{min}}, h + \mathtt{max}(\mathtt{height}(t_1), \ldots, \mathtt{height}(t_n)))$.

Note that $\mathcal{U}^h_H \cap \mathcal{L}(A)$ is not empty, since a term $t_{\mathtt{min}}$ in $\mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$ with minimal height satisfies $\mathtt{height}(t_{\mathtt{min}}) = h_{\mathtt{min}} < H$, and hence, by Proposition 2.9, $t_{\mathtt{min}} \in \mathcal{U}^h_H$ holds. Let $t_{\mathtt{max}}$ be a term in $\mathcal{U}^h_H \cap \mathcal{L}(A)$ with maximal $|\mathtt{Pos}^h_H(t_{\mathtt{max}})|$. Note that this term exists since by Lemma 2.6, the set $\{|\mathtt{Pos}^h_H(t)| \mid t \in \mathcal{U}^h_H \cap \mathcal{L}(A)\}$ is bounded. Since $t_{\mathtt{max}}$ is in $\mathcal{L}(A)$, there exists an i in $\{1, \ldots, n\}$, such that $t_{\mathtt{max}} \in \mathcal{L}(s_i : \varphi_i)$ holds. Let $p_1, \ldots, p_m$ be the positions where x_i occurs in s_i. Note that

$m \geq 2$ and $t_{\mathtt{max}}|_{p_1} = t_{\mathtt{max}}|_{p_2} = \ldots = t_{\mathtt{max}}|_{p_m}$ hold. Thus, since $t_{\mathtt{max}} \in \mathcal{U}_H^h$ holds, by Lemma 2.11 $p_1, \ldots, p_m \notin \mathtt{Pos}_H^h(t_{\mathtt{max}})$ holds.

We define a new term t as $t_{\mathtt{max}}$ if $\mathtt{height}(t_{\mathtt{max}}|_{p_1}) \geq h + |A|$ holds, and as $t_{\mathtt{max}}[t_i]_{p_1} \ldots [t_i]_{p_m}$ otherwise. We show that, in any case, $p_1, \ldots, p_m \notin \mathtt{Pos}_H^h(t)$, $\mathtt{Pos}_H^h(t_{\mathtt{max}}) \subseteq \mathtt{Pos}_H^h(t)$ and $t \in \mathcal{U}_H^h \cap \mathcal{L}(A)$ hold. In the first case, these facts are vacuously true. In the second case, the fact $p_1, \ldots, p_m \notin \mathtt{Pos}_H^h(t)$ follows from $\mathtt{height}(t_i) < H$. The fact $\mathtt{Pos}_H^h(t_{\mathtt{max}}) \subseteq \mathtt{Pos}_H^h(t)$ follows from the repeated application of Lemma 2.7, which can be applied because $\mathtt{height}(t_{\mathtt{max}}|_{p_1}) < h + |A| \leq \mathtt{height}(t_i)$ and $p_1, \ldots, p_m \notin \mathtt{Pos}_H^h(t_{\mathtt{max}})$ hold. The fact $t \in \mathcal{U}_H^h$ follows by repeated application of Lemma 2.13, which can be applied because $|p_1|, \ldots, |p_m| \leq h$ and $\mathtt{height}(t_{\mathtt{max}}|_{p_1}) < h + |A| \leq \mathtt{height}(t_i) < H - h$. The fact $t \in \mathcal{L}(A)$ follows from $t \in \mathcal{L}(s_i : \varphi_i)$.

By Lemma 2.15 applied to t at position p_1, there exists a term $\tilde{t}$ in $\mathcal{U}_H^h \cap \mathcal{L}(A)$ such that $\mathtt{Pos}_H^h(t) \subsetneq \mathtt{Pos}_H^h(\tilde{t})$. Note that $\mathtt{Pos}_H^h(t_{\mathtt{max}}) \subseteq \mathtt{Pos}_H^h(t) \subsetneq \mathtt{Pos}_H^h(\tilde{t})$ holds. This is in contradiction with the definition of $t_{\mathtt{max}}$. □

Corollary 2.17. *Let $\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ be a weakly too much duplicating set of patterns over $\Sigma \cup \mathcal{V}$.*

Then, $\mathcal{L}(\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\})$ is not regular.

Proof. Since $S := \{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ is a weakly too much duplicating set of patterns, it can be expressed as the union of two sets, $S_1 \cup S_2$, where S_1 contains all the too much duplicating patterns of S, and S_2 all the patterns generating a finite language. By Theorem 2.16, $\mathcal{L}(S_1)$ is not regular. Moreover, $\mathcal{L}(S_2)$ is finite, and the union of a finite language with a non-regular one is not regular. Therefore, $\mathcal{L}(S)$ is not regular. □

Chapter 3
Non-regularity detection for tree homomorphisms

In this chapter we e show how to use the result of the previous chapter to conclude non-regularity, in some cases, of images of regular languages by tree homomorphisms.

3.1. Recursive pattern descriptions

We start by defining another kind of patterns which define non-regular languages.

Definition 3.1. A pattern $s : \varphi$ is a *quasi too much duplicating* pattern if there exists a variable x appearing in some positions of s such that there exists a weakly too much duplicating set of patterns S holding $\mathcal{L}(S) = \varphi(x)$.

Example 3.2. *Using the signature of the previous examples, we define the following pattern $s : \varphi$ as $s = f(x, y)$, $\varphi(x) = \{f(t, t) \mid t \in \mathcal{T}(\Sigma)\}$ and $\varphi(y) = \mathcal{T}(\Sigma)$. Note that $\mathcal{L}(s : \varphi) = \{f(f(t_1, t_1), t_2) \mid t_1, t_2 \in \mathcal{T}(\Sigma)\}$. It is clear that $s : \varphi$ is not a too much duplicating pattern since s is linear. But note that the language of the allowed substitutions for x is the set of terms of the form $f(t, t)$, which can be expressed with the weakly (in fact, strongly) too much duplicating set of patterns $\{f(z, z) : \{z \mapsto \mathcal{T}(\Sigma)\}\}$. Therefore, $s : \varphi$ is a quasi too much duplicating pattern.*

The expressive power of sets of quasi too much duplicating patterns is the same as for weakly too much duplicating sets of patterns, as stated in the following lemma and corollary.

Lemma 3.3. *Let $s : \varphi$ be a quasi too much duplicating pattern. Then, there exists a weakly too much duplicating set of patterns $\bar{S}$ holding $\mathcal{L}(\bar{S}) = \mathcal{L}(s : \varphi)$.*

Proof. Let x be a variable appearing in some positions of s and such that there exists a weakly too much duplicating set of patterns $S = \{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ holding $\mathcal{L}(S) = \varphi(x)$. Without loss of generality, we assume that $\texttt{vars}(s_i)$ and $\texttt{vars}(s)$ are disjoint for $i \in \{1, \ldots, n\}$. Let $p_1, \ldots, p_k$ be all the positions of s where x occurs. Let $\bar{s}_i$ be $s[s_i]_{p_1} \ldots [s_i]_{p_k}$ for $i \in \{1, \ldots, n\}$. Let $\bar{\varphi}_i$ be such that $\bar{\varphi}_i(y) =$

$\varphi_i(y)$ if $y \in \mathtt{vars}(s_i)$, for $i \in \{1, \ldots, n\}$, and $\bar{\varphi}_i(y) = \varphi(y)$ if $y \in \mathtt{vars}(s) \setminus \{x\}$. We define $\bar{S}$ as $\{\bar{s}_1 : \bar{\varphi}_1, \ldots, \bar{s}_n : \bar{\varphi}_n\}$. It is obvious that $\mathcal{L}(\bar{S}) = \mathcal{L}(s : \varphi)$ holds and, by construction, $\bar{S}$ is a weakly too much duplicating set of patterns. □

Example 3.4. *Using the signature of the previous examples, we define the following quasi too much duplicating pattern* $s : \varphi$ *as* $s = f(x, y)$ *and* $\varphi(x) = \varphi(y) = \{f(a, f(t, t)) \mid t \in \mathcal{T}(\Sigma)\} \cup \{f(f(t, t), a) \mid t \in \mathcal{T}(\Sigma)\}$. *We now show that we can replace the* x *appearing in* s *in order to get a weakly too much duplicating set of patterns* $\bar{S}$ *generating the same language.*

In our case, the weakly too much duplicating set of patterns $S = \{s_1 : \varphi_1,\ s_2 : \varphi_2\}$, *with* $s_1 = f(a, f(z_1, z_1))$, $s_2 = f(f(z_2, z_2), a)$ *and* $\varphi_1(z_1) = \varphi_2(z_2) = \mathcal{T}(\Sigma)$, *satifies that* $\mathcal{L}(S) = \varphi(x)$. *We can define* $\bar{S}$ *replacing the occurrences of* x *in* s *as follows:* $\{s[s_1]_1 : \bar{\varphi}_1,\ s[s_2]_1 : \bar{\varphi}_2\}$, *with* $\bar{\varphi}_1(y) = \bar{\varphi}_2(y) = \varphi(y)$ *and* $\bar{\varphi}_1(z_1) = \bar{\varphi}_2(z_2) = \mathcal{T}(\Sigma)$. *Note that* $s[s_1]_1 = f(f(a, f(z_1, z_1)), y)$ *and* $s[s_2]_1 = f(f(f(z_2, z_2), a), y)$. *It is obvious that* $\mathcal{L}(\bar{S}) = \mathcal{L}(s : \varphi)$.

The following corollary follows directly from Lemma 3.3 and the definition of weakly too much duplicating set of patterns.

Corollary 3.5. *Let* S *be a set of patterns* $\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ *such that for each* $i \in \{1, \ldots, n\}$, $s_i : \varphi_i$ *is either a too much duplicating pattern, or a quasi too much duplicating pattern, or a pattern holding that* $\mathcal{L}(s_i : \varphi_i)$ *is finite. Moreover, we assume that there exists a* $j \in \{1, \ldots, n\}$, *such that* $\mathcal{L}(s_i : \varphi_i)$ *is not finite.*

Then, there exists a weakly too much duplicating set of patterns $\bar{S}$ *such that* $\mathcal{L}(S) = \mathcal{L}(\bar{S})$.

We use the new pattern-based formalism of the following Definition 3.6 to express the input of the problem, i.e. images of regular languages under tree homomorphisms. The reason to introduce this new formalism is that it is closer to our notion of weakly too much duplicating set of patterns than the original problem. In Lemma 3.8, we show that the original problem can actually be expressed in terms of this new formalism.

Definition 3.6. An *abstract pattern* is a pair $s : \varphi$ where s is a term and φ maps each variable occurring in s to a variable ranging over languages.

A *recursive pattern-description* (RPD) is a set of equations $\mathfrak{L} = \{L_1 = S_1, \ldots, L_m = S_m\}$ over variables $L_1, \ldots, L_m$ (the L_i are not languages, but syntactic variables ranging over languages), where each S_i is a set of abstract patterns $\{s_1 : \varphi_1, \ldots, s_n : \varphi_n\}$ satisfying that, for each $k \in \{1, \ldots, n\}$ and for each $x \in \mathtt{vars}(s_k)$, there exists a $j \in \{1, \ldots, m\}$ such that $\varphi_k(x) = L_j$ holds.

Note that descriptions like $L = \{x : \{x \mapsto L\}\}$ do not determine a language. Nevertheless, over RPD we are able to detect some cases of non-regularity for the minimal solution.

Example 3.7. *Using the signature of the previous examples, we define the following* RPD $\mathfrak{L} = \{L_1 = \{a : \{\},\ f(x_1, x_1) : \{x_1 \mapsto L_1\}\}\}$. *It is easy to see that the only solution for* L_1 *is the language of complete trees.*

Now let $\mathfrak{L}'$ be $\{L_2 = S_2, L_3 = S_3\}$, with $S_2 = \{a : \{\},\ f(x_2, x_2) : \{x_2 \mapsto L_3\}\}$ and $S_3 = \{f(x_3, x_3) : \{x_3 \mapsto L_2\}\}$. As before, $\mathfrak{L}'$ has only one solution, by assigning the language of complete trees with even height to L_2, and the language of complete trees with odd height to L_3.

Lemma 3.8. *Let A be a TA and ϕ a tree homomorphism. Let $\{q_1, \ldots, q_m\}$ be the set of states of A. Then, there exists an* RPD *$\mathfrak{L} = \{L_{q_1} = S_{q_1}, \ldots, L_{q_m} = S_{q_m}\}$ such that $L_{q_1} := \phi(\mathcal{L}(A, q_1)), \ldots, L_{q_m} := \phi(\mathcal{L}(A, q_m))$ is the minimal solution of $\mathfrak{L}$.*

Proof. Let A be $\langle Q, \Sigma, F, \Delta\rangle$ more explicitly written. For each q in Q, we define S_q as the following set of abstract patterns:

$$\left\{\phi(f(x_1, \ldots, x_k)) : \varphi \;\middle|\; f(q_1, \ldots, q_k) \to q \in \Delta \ \wedge\ \varphi(x_1) = L_{q_1}, \ldots, \varphi(x_k) = L_{q_k}\right\}$$

By construction, it is straightforward that $L_{q_1} := \phi(\mathcal{L}(A, q_1)), \ldots, L_{q_m} := \phi(\mathcal{L}(A, q_m))$ is the minimal solution of $\mathfrak{L}$. □

Example 3.9. *Let A be a TA defined as $\langle\{q, q_f\}, \Sigma, \{q_f\}, \Delta\rangle$, where $\Sigma = \{a : 0,\ g : 1,\ f : 2\}$ and $\Delta = \{a \to q,\ g(q) \to q,\ f(q, q) \to q_f\}$. The language recognized by A is $\{f(g^n(a), g^m(a)) \mid n, m \geq 0\}$. Let ϕ be a tree homomorphism defined as $\phi(a) = a$, $\phi(g(x)) = f(\phi(x), \phi(x))$ and $\phi(f(x_1, x_2)) = \phi(x_1)$. Then, $\phi(\mathcal{L}(A))$ is the language of complete trees over the signature $\{a : 0,\ f : 2\}$.*

Using the construction detailed in the proof of Lemma 3.8, we define the RPD *$\mathfrak{L} = \{L_q = \{a : \{\},\ f(x, x) : \{x \mapsto L_q\}\},\ L_{q_f} = \{x_1 : \{x_1 \mapsto L_q\}\}\}$. Note that the only solution for $\mathfrak{L}$ maps L_q and L_{q_f} to the language of complete trees over the signature $\{a : 0,\ f : 2\}$, as in Example 3.7. Recall that $\phi(\mathcal{L}(A))$ is precisely this language.*

3.2. The algorithm

We now show an algorithm that, using the previous results, allows us to conclude in some cases the non-regularity of the images of regular languages under tree homomorphisms, by using the RPD representation. In the description of the algorithm we deliberately confuse the variables $L_{q_1}, \ldots, L_{q_m}$ with the list of languages which are the minimal solution of $\mathfrak{L}$. The algorithm proceeds as follows:

(1) Input: A TA $A = \langle Q, \Sigma, F, \Delta\rangle$ and a tree homomorphism ϕ.
(2) Construct $\mathfrak{L} = \{L_{q_1} = S_{q_1}, \ldots, L_{q_m} = S_{q_m}\}$ from A and ϕ according to Lemma 3.8.
(3) Detect which languages L_q are empty.
(4) Detect which languages L_q are infinite.
(5) For each $L_q = S_q$ in $\mathfrak{L}$ detect which patterns in S_q are too much duplicating. It suffices to test for each pattern $s : \varphi$ whether φ maps a variable occurring twice in s to an infinite $L_{q'}$, and each variable occurring in s to a non-empty language.
(6) Iteratively:
    - Mark as weakly too much duplicating each infinite L_q holding that each of the patterns in S_q has been detected as either too much duplicating, or quasi too much duplicating, or generating a finite language.

- Detect which patterns of each S_q are quasi too much duplicating. It suffices to test for each pattern $s : \varphi$ whether φ maps a variable occurring in s to an $L_{q'}$ which has been marked as weakly too much duplicating, and each variable occurring in s to a non-empty language.

(7) Output ``Non-regular'' if for all $q_f$ in $F$, either $L_{q_f}$ has been marked as weakly too much duplicating or finite, and at least one of them has been marked as weakly too much duplicating. Otherwise output ``Do not know''.

The algorithm can be implemented in linear time using proper data structures and well known algorithms like Strongly Connected Components, in order to detect cycles and infinite languages, and Topological Sort, in order to mark weakly too much duplicating languages.

Example 3.10. *We show an execution of the algorithm with the* RPD $\mathfrak{L} = \{L_q = S_q, L_{q_f} = S_{q_f}\}$ *resulting from the TA and tree homomorphism of Example 3.9.*

- *The algorithm detects L_q and L_{q_f} as non-empty.*
- *Next, it detects L_q as infinite since it has a self-loop, and L_{q_f} is also detected as infinite since S_{q_f} contains a pattern with a variable constrained to L_q.*
- *Next, it detects the pattern $f(x, x) : \{x \mapsto L_q\}$ in S_q as a too much duplicating pattern.*
- *At the first iteration step, L_q is marked as weakly too much duplicating since the pattern $a : \{\}$ generates a finite language and the pattern $f(x, x) : \{x \mapsto L_q\}$ has been detected as too much duplicating. In the same iteration, the pattern $x_1 : \{x_1 \mapsto L_q\}$ of S_{q_f} is detected as quasi too much duplicating.*
- *At the second iteration step, L_{q_f} is marked as weakly too much duplicating since the pattern $x_1 : \{x_1 \mapsto L_q\}$ of S_{q_f} has been detected as quasi too much duplicating. The algorithm outputs* ``Non-regular'' *since L_{q_f} has been marked as weakly too much duplicating.*

Non-regularity may not always be detected, since neither all non-regular languages are weakly too much duplicating nor the algorithm detects all the languages which can be represented by a weakly too much duplicating set of patterns. This is shown in the following example.

Example 3.11. *Let A be a TA defined as $\langle\{q, q_f\}, \Sigma, \{q_f\}, \Delta\rangle$, where $\Sigma = \{a : 0,\ g : 1,\ f_1 : 1,\ f_2 : 2\}$ and $\Delta = \{a \to q,\ g(q) \to q,\ f_1(q) \to q_f,\ f_2(q, q) \to q_f\}$. The language recognized by A is $\{f_1(g^n(a)) \mid n \geq 0\} \cup \{f_2(g^n(a), g^m(a)) \mid n, m \geq 0\}$. Let ϕ be a tree homomorphism defined as $\phi(a) = a$, $\phi(g(x)) = g(\phi(x))$, $\phi(f_1(x)) = f(\phi(x), f(a, a))$ and $\phi(f_2(x, y)) = f(\phi(x), f(\phi(y), \phi(y)))$, using a new function symbol f with arity 2.*

It is easy to see that $\phi(\mathcal{L}(A))$ can be described with the weakly too much duplicating set of patterns $\{f(x, f(y, y)) : \varphi\}$, where $\varphi(x) = \varphi(y) = \{g^n(a) \mid n \geq 0\}$, which is therefore non-regular by Corollary 2.17. Yet, using the construction detailed in the proof of Lemma 3.8, the algorithm would not be able to conclude the non-regularity. First, we would express the input as the RPD $\mathfrak{L} = \{L_q = \{a : \{\},\ g(x) : \{x \mapsto L_q\}\},\ L_{q_f} = \{f(x, f(a, a)) : \{x \mapsto L_q\},\ f(x, f(y, y)) : \{x \mapsto L_q, y \mapsto L_q\}\}\}$. *The algorithm would correctly detect L_q and L_{q_f} as infinite, and the pattern $f(x, f(y, y)) : \{x \mapsto L_q, y \mapsto L_q\}$ as too much duplicating, but neither L_q nor L_{q_f}*

could be marked as weakly too much duplicating. In the case of L_{q_f} *this is because the pattern* $f(x, f(a, a)) : \{x \mapsto L_q\}$ *represents an infinite language but it is not detected as too much duplicating neither quasi too much duplicating.*

Chapter 4
Conclusions

In spite of the high complexity for deciding regularity of languages represented by patterns and images of regular languages by tree homomorphisms, we have provided a very efficient procedure (linear time) for detecting non-regularity in some cases. It would be interesting to study alternative procedures for detecting regularity in some cases in an efficient way.

Our result of non-regularity of weakly too much duplicating sets of patterns is independent from the mechanism used for representing the constraints. Thus, it might be possible to apply it to other formalisms than images of tree homomorphisms, and this deserves further study.

References

[CDG+07] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, *Tree automata techniques and applications*, Available at http://www.grappa.univ-lille3.fr/tata, 2007.

[CGR] C. Creus, G. Godoy, and L. Ramos, *Too much duplicating patterns represent non-regular languages*, Applicable Algebra in Engineering Communication and Computing, Submitted.

[GGJ09] A. Gascón, G. Godoy, and F. Jacquemard, *Closure of tree automata languages under innermost rewriting*, Electron. Notes Theor. Comput. Sci. **237** (2009), 23–38.

[GGM09] O. Giménez, G. Godoy, and S. Maneth, *Deciding regularity of the set of instances of a set of terms with regular constraints is exptime-complete*, CoRR **abs/0911.3674** (2009).

[GGRÀ10] G. Godoy, O. Giménez, L. Ramos, and C. Àlvarez, *The hom problem is decidable*, STOC, 2010, pp. 485–494.

[KR99] G. Kucherov and M. Rusinowitch, *Patterns in words versus patterns in trees: A brief survey and new results*, Ershov Memorial Conference, Springer, 1999, pp. 283–296.

[VG92] S. Vágvölgyi and R. Gilleron, *For a rewrite system it is decidable whether the set of irreducible, ground terms is regognizable*, Bulletin of the EATCS **48** (1992), 197–209.