Information Processing Letters 114 (2014) 85–93

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

Excessively duplicating patterns represent non-regular languages

Carles Creus [1,2], Guillem Godoy [*,1], Lander Ramos [1]

Universitat Politècnica de Catalunya, Omega building, Jordi Girona 1-3, Barcelona 08034, Spain

ARTICLE INFO

Article history:
Received 5 March 2013
Received in revised form 10 October 2013
Accepted 11 November 2013
Available online 13 November 2013
Communicated by A. Muscholl

Keywords:
Theory of computation
Pattern
Regular tree language
Tree automaton
Tree homomorphism

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. In both cases, duplication of variables in the terms of the patterns is a necessary condition for non-regularity. This is because duplications force the recognizer to test equality between subterms. Hence, 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 "excessively duplicating" variables. We prove that when each pattern of a non-empty 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 of 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.

1. Introduction

A constrained 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 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 constrains the instances of s by restricting the possible substitutions of each variable. A set of patterns S represents the union of the languages represented by each pattern in S.

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 whether a language is a subset of another language. 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 [1]. This question is relevant because regular tree languages have a more tractable representation by means of tree automata, and have better properties like closure by several set operations, and existence of efficient procedures for some of such operations and for several decision problems.

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 [2,3].

* Corresponding author. Tel.: +34 93 4137815; fax: +34 93 4137833.
E-mail addresses: ccreuslopez@gmail.com (C. Creus), ggodoy@lsi.upc.edu (G. Godoy), landertxu@gmail.com (L. Ramos).

[1] The three authors were supported by Spanish Ministry of Education and Science by the FORMALISM project (TIN2007-66523).
[2] Supported by an FPU grant from the Spanish Ministry of Education.

0020-0190/$ – see front matter
http://dx.doi.org/10.1016/j.ipl.2013.11.010

The problem is EXPTIME-complete when the constraints restrict each variable to a regular set [4].

Patterns 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 only have a finite number of states and cannot test equality 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. However, when considering more expressive constraints, linearity of the patterns does not necessarily imply regularity of the represented language.

In this paper we focus on proving non-regularity when the patterns of a given set have "excessively duplicating" 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 of 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. Note that the general case has recently been proved EXPTIME-complete [5].

The paper is structured as follows. In Section 2 we present basic preliminary definitions used in the rest of the paper. In Section 3 we define patterns, the particular case of excessively duplicating patterns, and prove that sets of excessively duplicating patterns always represent non-regular languages. In Section 4 we apply our result on patterns to detect, in some cases, the non-regularity of the image of a regular language under a tree homomorphism. We conclude in Section 5.

2. Preliminaries

2.1. Terms

We use the standard notation from the term rewriting literature [6]. The powerset of a set S is denoted by 2^S, and the cardinal of S is denoted by $\#S$. A signature Σ is a (finite) set of function symbols with arity, which is partitioned as $\bigcup_i \Sigma^{(i)}$ such that $f \in \Sigma^{(m)}$ if the arity of f is 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, with possible subscripts. The elements of a set $\mathcal{X}$ of variable symbols are denoted by x, y, z with possible subscripts. The set $\mathcal{T}(\Sigma,\mathcal{X})$ of *terms* over Σ and $\mathcal{X}$, is the smallest set containing $\mathcal{X}$ such that $f(t_1,\ldots,t_m)$ is in $\mathcal{T}(\Sigma,\mathcal{X})$ whenever $f \in \Sigma^{(m)}$ and $t_1,\ldots,t_m \in \mathcal{T}(\Sigma,\mathcal{X})$. The set of variables occurring in a term t is denoted $\mathtt{vars}(t)$. A term t is called *linear* if each variable occurs at most once in t. A term t is called *ground* if it contains no variables. The set of all ground terms over Σ is denoted $\mathcal{T}(\Sigma)$. A *language* over Σ is a set of ground terms.

A *position* is a sequence of natural numbers. The symbol λ denotes the empty sequence, and $p.p'$ denotes the concatenation of the positions p and p'. The set of positions of a term t, denoted $\mathtt{Pos}(t)$, is defined recursively as $\mathtt{Pos}(f(t_1,\ldots,t_m)) = \{\lambda\} \cup \{i.p \mid i \in \{1,\ldots,m\} \wedge p \in \mathtt{Pos}(t_i)\}$. The *length* of a position is denoted as $|p|$. Note that $|\lambda| = 0$ and $|i.p| = 1 + |p|$. A position p_1 is a *prefix* of a position p, denoted $p_1 \leqslant p$, if there is a position p_2 such that $p_1.p_2 = p$. Also, p_1 is a *proper prefix* of p, denoted $p_1 < p$, if $p_1 \leqslant p$ and $p_1 \neq p$. Two positions p, p' are parallel, denoted by $p \| p'$, if $p \not\leqslant p'$ and $p' \not\leqslant p$.

The *subterm* of a term t at a position $p \in \mathtt{Pos}(t)$, denoted $t|_p$, is defined recursively as $t|_\lambda = t$ and $f(t_1,\ldots,t_m)|_{i.p} = t_i|_p$. The *replacement* of the subterm at position p in a term t by a term s, denoted $t[s]_p$, is defined recursively as $t[s]_\lambda = s$ and $f(t_1,\ldots,t_m)[s]_{i.p} = f(t_1,\ldots,t_i[s]_p,\ldots,t_m)$.

The *root* of a term $t = f(t_1,\ldots,t_m)$ is $\mathtt{root}(t) = f$. 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., $t(p) = \mathtt{root}(t|_p)$.

The *height* of a term t, denoted $\mathtt{height}(t)$, is defined recursively as $\mathtt{height}(t) = 0$ if t is a variable or a constant, and as $\mathtt{height}(f(t_1,\ldots,t_m)) = 1 + \max(\mathtt{height}(t_1),\ldots,\mathtt{height}(t_m))$ otherwise.

A *substitution* σ is a mapping from variables to terms. It can also be used as a function from terms to terms recursively defined by $\sigma(f(t_1,\ldots,t_m)) = f(\sigma(t_1),\ldots,\sigma(t_m))$ for terms different from variables. For example, if σ is $\{x \mapsto f(b,y), y \mapsto a\}$, then $\sigma(g(x,y))$ is $g(f(b,y),a)$.

2.2. Automata

A *tree automaton* (TA) is a tuple $A = \langle Q, \Sigma, F, \Delta\rangle$, where Q is a finite set of states, Σ is a 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 $f \in \Sigma^{(m)}$. The *size* of A is defined by $\#A = \#Q$.

A *run* r of a TA A on a term $t \in \mathcal{T}(\Sigma)$ 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 $r(p) = q$. 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. 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 such that $r(\lambda) = q$.

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

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' 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 $p \not\leqslant p'$, and otherwise $r[r']_p(p.p') = r'(p')$.