Journal of Automata, Languages and Combinatorics **u** (v) w, x–y

LEARNING REGULAR TREE LANGUAGES FROM CORRECTION AND EQUIVALENCE QUERIES

Cătălin Ionuţ Tîrnăucă
Research Group on Mathematical Linguistics, Rovira i Virgili University
Plaça Imperial Tàrraco 1, 43005, Tarragona, Spain
e-mail: catalinionut.tirnauca@estudiants.urv.cat

and

Cristina Tîrnăucă[1]
Research Group on Mathematical Linguistics, Rovira i Virgili University
Plaça Imperial Tàrraco 1, 43005, Tarragona, Spain
e-mail: cristina.bibire@urv.cat

ABSTRACT

Inspired by the results obtained in the string case, we present in this paper the extension of the correction queries to regular tree languages. Relying on Angluin's and Sakakibara's work, we introduce the algorithm $L_{RTL}C$ and we show that regular tree languages are learnable from equivalence and correction queries when the set of contexts is ordered by a Knuth-Bendix order. Moreover, a subclass of regular tree languages, called injective languages, is learned without equivalence queries. This can be extended for other subclasses and may have some practical relevance in fields like machine translation, pattern and speech recognition, building XML documents.

Keywords: regular tree languages, learning from queries, weight function, correcting context

1. Introduction

The field of grammatical inference was practically introduced by Gold in 1967 [8], when he suggested that learning is an infinite process about making guesses of grammars, and which does not terminate in a finite number of steps but only converges in the limit. Twenty years later, Angluin [1] proposes another popular learning criterion: exact identification using queries. The algorithm, called L^*, allows the Learner to ask questions about a regular language from an oracle (also called MAT - minimally adequate teacher) and halts in polynomial time with a correct description of the language.

[1]This work was possible thanks to a FPU Fellowship (AP2004-6968) from the Spanish Ministry of Education and Science

The first attempt to extend this inference method to context-free grammars (CFGs) belongs to Angluin as well. In the same paper [1] she designs the algorithm L^{cf} and shows that in this framework context-free languages (CFLs) are learnable. But the learnability is proved under powerful restrictions. Among them, we mention two important ones: the grammar should be in Chomsky normal form and the set of nonterminal symbols, along with the start symbol, are assumed to be known by the Learner.

We consider that the problem of inferring CFGs is worth studying from both practical and theoretical points of view. Regarding practical applications, the learnability of a CFG that produces a set of patterns [6] constitutes an important issue for people working in pattern recognition. Also, the ability to infer (in fact to approximate since the problem of natural languages' context-freeness is subject to a long and still unsolved debate) CFGs for natural languages would enable a speech recognizer to adapt its internal grammar according to the particularities of an individual speaker [9]. The problem seems to be interesting also from a theoretical point of view because, although CFLs are well-understood, there are various and serious restrictions in the process of learning them. A very good survey of the problem of learning CFLs can be found in [10].

In 1992 Sakakibara [14] extends the algorithm proposed by Angluin to skeletal regular tree automata which are a variation of finite automata that take skeletons as input. Intuitively, skeletons (introduced in [11]) are derivation trees of strings in a grammar in which internal nodes are unlabeled. Such a tree reveals only the syntactic structure associated with a string with respect to a CFG but not the concrete rules generating it.

The interest in learning regular tree languages is justified by the nice relationship which exists between these languages and CFLs: the yield of any regular tree language is a CFL. In 2003, Drewes and Högberg [4] improve Sakakibara's algorithm, generalizing L^* to regular tree languages. The advantages of this approach consist in avoiding the dead states and minimizing the observation table as much as possible.

Successful language learning algorithms model how humans acquire languages. This view, though questionable, was one of the principal motivations for the early work in grammatical inference. Inspired by how children learn, Becerra-Bonache, Dediu and Tîrnăucă propose in [3] the algorithm LCA with an alternative to membership queries: instead of a yes/no answer, the teacher returns a correcting string. They choose as a correction for the string s, the smallest string s' (in the lex-length order) such that ss' belongs to the target language. Even though the worst case complexity of the two algorithms is the same, LCA runs more efficient on average, mainly because of the embedded information contained in the correction queries.

In this paper we propose an extension of L^* which generalizes the correction queries introduced in [3] from correcting strings to correcting contexts. Though our algorithm, restricted to skeletal tree languages, can be also applied to context-free string languages, we focus on regular tree languages in what follows.

We prove that deterministic bottom-up tree recognizers can be learned from correction and equivalence queries, and that equivalence queries are not needed in the process of learning the class of injective languages (languages for which any two non

equivalent trees have distinct corrections). The last result has relevance in practice since one can imagine the Teacher as a human expert who might not have an automata representation for the language but is still able to return the correction based only on his domain-specific knowledge.

The order we choose for comparing trees (contexts) to obtain the minimal one is a Knuth-Bendix order based on the weights associated to the symbols of the alphabet. The minimal context may correspond to the most probable choice in a real-life setting, like speech recognition or even machine translation if we imagine an extension of our algorithm to transducers.

The paper is organized as follows. In the next section we recall some basic notions regarding trees, tree recognizers and Knuth-Bendix orders. Section 3 presents the observation table as the main data structure of the learning algorithm and introduces correction queries. It contains the description of two algorithms: the Learner and the Teacher, together with the proof of their correctness, the analysis of time complexity and a running example. Section 4 is dedicated to an application: a subclass of regular tree languages, called injective languages, is learned without equivalence queries. We finish with some concluding remarks and future extensions of the algorithms presented here.

2. Preliminaries

In this paper we follow standard definitions and notations in formal language theory. A wealth of further information about this area can be found in [12, 13].

2.1. Trees

The trees considered here are finite, their nodes are labeled by symbols, and the branches leaving any given node have a specified order.

A *ranked alphabet* Σ is a finite set of symbols each of them having a given non-negative integer arity. For any $m \geq 0$, the set of m-ary symbols in Σ is denoted by Σ_m. In examples we may write $\Sigma = \{f_1/m_1, \ldots, f_k/m_k\}$ to indicate that Σ consists of the symbols $f_1, \ldots, f_k$ with the respective ranks $m_1, \ldots, m_k$. In what follows Σ is always a ranked alphabet.

The set T_Σ of Σ-*terms* is the smallest set T such that

- $\Sigma_0 \subseteq T$, and
- $f(t_1, ..., t_m) \in T$ whenever $m > 0$, $f \in \Sigma_m$ and $t_1, ..., t_m \in T$.

Such terms are regarded as representations of trees, and we call them Σ-*trees*. Any $u \in \Sigma_0$ represents a tree with only one node labeled with u. Similarly, $f(t_1, ..., t_m)$ is interpreted as a tree formed by adjoining the m trees represented by $t_1, ..., t_m$ to a new f labeled root. The trees $t_1, ..., t_m$ are said to be the *direct subtrees* of the tree. Subsets of T_Σ are called Σ-*tree languages*. We will generally speak about *trees* and *tree languages* without specifying the alphabets.

Given a set T of Σ-trees, we denote by $T\Sigma$ the set of all trees $f(t_1, ..., t_m)$ such that $f \in \Sigma_m$ for some $m \geq 0$, and $t_1, \ldots, t_m \in T$.

The *height* $hg(t)$ and the set of *subtrees* $sub(t)$ of a Σ-tree t are defined such that

- $hg(t) = 0$, $sub(t) = \{t\}$ for $t \in \Sigma_0$, and
- $hg(t) = \max\{hg(t_1), \ldots, hg(t_m)\} + 1$, $sub(t) = \{t\} \cup sub(t_1) \cup \ldots \cup sub(t_m)$ for $t = f(t_1, \ldots, t_m)$, $m > 0$ and $f \in \Sigma_m$.

Let ξ be a special symbol with arity 0 and not in Σ. A $\Sigma \cup \{\xi\}$-tree in which ξ appears exactly once is called a Σ-*context*, or just a *context*. The set of all Σ-contexts is denoted by C_Σ. If $p, p' \in C_\Sigma$, then $p \cdot p' = p'(p)$ is the Σ-context obtained from p' by replacing the ξ in it with p. Similarly, if $t \in T_\Sigma$ and $p \in C_\Sigma$, then $t \cdot p = p(t)$ is the tree obtained when the ξ in p is replaced with t (we also say that the context p is applied to t).

Let p be a context in C_Σ. Then $depth(p)$ and $trees(p)$ are defined as follows:

- $depth(p) = 0$, $trees(p) = \emptyset$ for $p = \xi$, and
- $depth(p) = depth(p') + 1$, $trees(p) = trees(p') \cup \{t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m\}$ for $p = f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \cdot p'$ with $m > 0$, $f \in \Sigma_m$, $t_1, \ldots, t_{i-1}$, $t_{i+1}, \ldots, t_m \in T_\Sigma$, $1 \leq i \leq m$, and $p' \in C_\Sigma$.

2.2. Finite Tree Recognizers and Regular Tree Languages

A *deterministic bottom-up* Σ*-tree recognizer*, or tree recognizer, for short, is a quadruple $A = (Q, \Sigma, \delta, F)$, where Σ is a ranked alphabet, Q is a finite set of *states* such that $Q \cap \Sigma = \emptyset$, $F \subseteq Q$ is a set of *final states*, and the *transition function* δ is a family of maps $\delta_m (m \geq 0, \Sigma_m \neq \emptyset)$ such that:

- $\delta_0 : \Sigma_0 \to Q$, and
- $\delta_m : Q^m \times \Sigma_m \to Q$ for $m > 0$.

Then for $t \in T_\Sigma$, one can inductively define $\delta(t)$ as follows:

- $\delta(t) = \delta_0(t)$ for $t \in \Sigma_0$, and
- $\delta(t) = \delta_m(\delta(t_1), \ldots, \delta(t_m), f)$ for $m > 0$, $f \in \Sigma_m$ and $t = f(t_1, \ldots, t_m)$.

The set $T(A) = \{t \in T_\Sigma \mid \delta(t) \in F\}$ is the tree language *recognized* (*accepted*) by A. Such a tree language is a *regular tree language*. Let $A = (Q, \Sigma, \delta, F)$ and $A' = (Q', \Sigma, \delta', F')$ be deterministic bottom-up Σ-tree recognizers. One can say that A and A' are *equivalent* if they accept the same tree language, and A is *isomorphic* to A' if there exists a bijection $\phi : Q \to Q'$ such that $\phi(F) = F'$ and

- $\phi(\delta_0(c)) = \delta'_0(c)$ for every $c \in \Sigma_0$, and
- $\phi(\delta_m(q_1, \ldots, q_m, f)) = \delta'_m(\phi(q_1), \ldots, \phi(q_m), f)$ for every $m > 0$, $f \in \Sigma_m$ and every $q_1, \ldots, q_m \in Q$.

Let A be a tree recognizer. A state q is called *reachable* if there exists a tree $t \in T_\Sigma$ such that $\delta(t) = q$. A reachable state q is *co-reachable* if there exists a context $p \in C_\Sigma$ such that $\delta(t \cdot p) \in F$, where t is any tree in T_Σ such that $\delta(t) = q$. A state which is reachable but not co-reachable is said to be a *sink state*.

Lemma 1 (Replacement lemma) *Let $A = (Q, \Sigma, \delta, F)$ be a deterministic bottom-up Σ-tree recognizer. For $t, t' \in T_\Sigma$ and $p \in C_\Sigma$, if $\delta(t) = \delta(t')$, then $\delta(t \cdot p) = \delta(t' \cdot p)$.*

For a given tree language $T \subseteq T_\Sigma$, the equivalence relation $\equiv_T$ on T_Σ is defined by: for any $s, t \in T_\Sigma$, $s \equiv_T t$ if for all contexts $p \in C_\Sigma$, $s \cdot p \in T \Leftrightarrow t \cdot p \in T$. One can notice that $\equiv_T$ is a congruence (i.e. an equivalence relation which is preserved by contexts: for $t, t' \in T_\Sigma$, if $t \equiv_T t'$, then $t \cdot p \equiv_T t' \cdot p$ for all contexts $p \in C_\Sigma$).

It is known that each regular tree language T is accepted by a *minimal* tree recognizer, unique up to isomorphism, that can be constructed from any given tree recognizer A of T by first deleting all non-reachable states and then merging all pairs of equivalent states. For details, we refer to [7], pp. 87 - 94. Note that for a minimal tree recognizer A there is at most one sink state and all the states are reachable.

Example 1 Let $\Sigma = \{f/3, g/2, a/0, b/0\}$, and consider the tree language

$$T = \{t \in T_\Sigma \mid t = g(a, b) \cdot [f(a, \xi, b)]^n, n \geq 0\},$$

where $[f(a, \xi, b)]^n = \xi$ and $[f(a, \xi, b)]^n = [f(a, \xi, b)]^{n-1} \cdot f(a, \xi, b)$, $n \geq 1$.

Let us now construct a tree recognizer $A = (Q, \Sigma, \delta, F)$ such that $T(A) = T$. For this, we need a set Q of four states: q_a, q_b, q_s and q_f. The transition function δ is defined as follows:

- $\delta_0(a) = q_a$, $\delta_0(b) = q_b$,
- $\delta_2(q_a, q_b, g) = q_f$,
- $\delta_3(q_a, q_f, q_b, f) = q_f$,
- $\delta_2(q_x, q_y, g) = q_s$ for $x = b$, $y = a$ and $x = y$, $x \in \{a, b\}$, and
- $\delta_3(q_x, q_y, q_z, f) = q_s$ for all remaining cases.

If we take $F = \{q_f\}$, it is clear that $T(A) = T$. The (accepting) computation on the input $t = f(a, f(a, g(a, b), b), b)$ is $\delta(t) = \delta_3(q_a, \delta_3(q_a, \delta_2(q_a, q_b, g), q_b, f), q_b, f) = \delta_3(q_a, \delta_3(q_a, q_f, q_b, f), q_b, f) = \delta_3(q_a, q_f, q_b, f) = q_f$. Also, it is easy to see that q_s is a sink state and q_f, q_a and q_b are co-reachable.

2.3. Knuth-Bendix Orders

Let Σ be a ranked alphabet and $<$ be a strict order on Σ. Let $w : \Sigma \to \mathbb{R}_0^+$ be a *weight function*, where $\mathbb{R}_0^+$ denotes the set of non-negative real numbers. A weight function w is called *admissible* for $<$ if it satisfies the following conditions:

1. There exists $w_0 \in \mathbb{R}_0^+ \backslash \{0\}$ such that $w(c) \geq w_0$ for all $c \in \Sigma_0$.
2. If $g \in \Sigma_1$ and $w(g) = 0$, then g is the greatest element in Σ, i.e. $f \leq g$ for all $f \in \Sigma$.

The weight function w is extended to a function $w : T_\Sigma \to \mathbb{R}_0^+$ as follows:

$$w(t) = \sum_{f \in \Sigma} w(f) \cdot |t|_f,$$

where $|t|_f$ denotes the number of occurrences of the symbol f in t. Thus, $w(t)$ simply adds up the weights of all occurrences of symbols from Σ in t.

The *Knuth-Bendix order* $<_{kbo}$ on T_Σ induced by $<$ and w is defined as follows. For $t, t' \in T_\Sigma$, we have $t <_{kbo} t'$ if $w(t) < w(t')$, or $w(t) = w(t')$ and one of the following two properties holds:

1. There exist symbols $f \in \Sigma_m$ $(m \geq 0)$ and $g \in \Sigma_n$ $(n \geq 0)$ such that $f < g$ and $t = f(s_1, \ldots, s_m)$, $t' = g(t_1, \ldots, t_n)$.
2. There exist a symbol $f \in \Sigma_m$ $(m > 0)$ and an index i, $1 \leq i \leq m$, such that $t = f(t_1, \ldots, t_i, \ldots, t_m)$, $t' = f(s_1, \ldots, s_i, \ldots, s_m)$, $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$ and $t_i <_{kbo} s_i$.

One can see that $(T_\Sigma, <_{kbo})$ is an ordered set, and $<_{kbo}$ is a strict total order. For $t, t' \in T_\Sigma$, $t \leq_{kbo} t'$ if $t <_{kbo} t'$ or $t = t'$.

The Knuth-Bendix order $<_{kbo}$ can be extended from T_Σ to C_Σ in a natural way. Thus, $<$ will be a strict order on $\Sigma \cup \{\xi\}$ such that $\xi < f$ for all $f \in \Sigma$, and we set $w(\xi) = w_0$.

Remark 1 The following properties hold:

1. ξ is the smallest context.
2. For any $p_1, p_2 \in C_\Sigma$, if $p_1 <_{kbo} p_2$, then $p_1 \cdot p <_{kbo} p_2 \cdot p$ and $p \cdot p_1 <_{kbo} p \cdot p_2$ for every $p \in C_\Sigma$.
3. For any $t_1, t_2 \in T_\Sigma$, if $t_1 <_{kbo} t_2$, then $t_1 \cdot p <_{kbo} t_2 \cdot p$ for every $p \in C_\Sigma$.
4. For any $t \in T_\Sigma$ and any $p \in C_\Sigma \backslash \{\xi\}$, $t <_{kbo} t \cdot p$.
5. For any $p, p' \in C_\Sigma$ with $p' \neq \xi$, $p <_{kbo} p' \cdot p$.

Actually, we chose a Knuth-Bendix order for our algorithm just to illustrate that an order with the desired properties does exist. The algorithm will still work with any other order for which all the properties from Remark 1 hold. More details can be found in [2], pp. 111-133.

3. Learning Regular Tree Languages from Corrections

The notion of learning from queries was introduced by Angluin in [1]. The aim was to learn an unknown regular language L (more precisely, to construct the minimal DFA recognizing L). The same approach may be used for trees, as it has been shown in [4, 14]. With the help of MAT, the Learner is supposed to identify the target regular tree language being allowed to use two types of queries: membership and equivalence queries.

For the rest of the paper, let us fix an arbitrary ranked alphabet Σ, let $T \subseteq T_\Sigma$ be the regular tree language to be learned, and let A_T be the minimal deterministic bottom-up Σ-tree recognizer which accepts T. The MAT knows the target language and is assumed to answer correctly to the following types of queries:

- *Membership query* (MQ). Given some tree t in T_Σ, the Teacher will check whether or not $t \in T$.

- *Equivalence query* (EQ). Given a tree recognizer A, the Teacher will check whether A is equivalent to A_T. If the answer is "no" a tree *counterexample* $t \in (T(A) \backslash T) \cup (T \backslash T(A))$ (in the symmetric difference of $T(A)$ and T) is returned.

3.1. Correcting Contexts

We introduce a new type of query called *correction query* (CQ). It is an extension of the membership query; the difference consists in the type of the answer that we receive from the Teacher. Instead of a yes/no answer, a context called *correcting context* is returned to the Learner.

The motivation for using correction queries comes from linguistics as one can see in [3]. The main idea is that a child can learn faster if he is helped in a proper manner, not only responding by yes or no to his questions, but also trying to correct him if he makes a mistake. In this way he can add the new information to the previous knowledge in order to infer the correct grammar more easily.

For a tree $t \in T_\Sigma$, the *frontier derivative language* of T with respect to t is the set $t^{-1}T = \{p \in C_\Sigma \mid t \cdot p \in T\} = \{p \in C_\Sigma \mid \delta(t \cdot p) \in F\}$, where $A = (Q, \Sigma, \delta, F)$ is any deterministic bottom-up Σ-tree recognizer accepting the tree language T. One can speak about the frontier derivative language of a state q as being $t_q^{-1}T$, where $t_q \in T_\Sigma$ is such that $\delta(t_q) = q$.

The *correcting context* of a tree $t \in T_\Sigma$ with respect to the tree language T and the Knuth-Bendix order $<_{kbo}$ on C_Σ, denoted $Cor_T(t)$, is the minimal context of the set $t^{-1}T$. In case that no such context exists (i.e., $t^{-1}T = \emptyset$) we say $Cor_T(t) = \theta$, where θ is a symbol which does not belong to $\Sigma \cup \{\xi\}$. Hence, Cor_T is a function from T_Σ to $C_\Sigma \cup \{\theta\}$. Note that $Cor_T(t) = \xi$ if and only if t is in T, and for all $s, t \in T_\Sigma$, $s \equiv_T t$ implies $Cor_T(s) = Cor_T(t)$, but the converse does not hold.

Remark 2 For $t \in T_\Sigma$ and $p_1, p_2 \in C_\Sigma$, if $Cor_T(t) = p_1 \cdot p_2$, then $Cor_T(t \cdot p_1) = p_2$.

Proof. If $Cor_T(t) = p_1 \cdot p_2$, then $p_1 \cdot p_2$ is the smallest context from C_Σ such that $t \cdot p_1 \cdot p_2 \in T$, and hence $p_2 \in (t \cdot p_1)^{-1}T$. But p_2 must be the smallest context with this property since otherwise we reach a contradiction with the minimality of $p_1 \cdot p_2$ (cf. Remark 1). We conclude that $Cor_T(t \cdot p_1) = p_2$. □

Remark 3 If t is a tree in T_Σ such that $t^{-1}T = \emptyset$, then $(t \cdot p)^{-1}T = \emptyset$ for all contexts p in C_Σ.

Proof. Suppose by contradiction that there exists a context $p \in C_\Sigma$ such that $(t \cdot p)^{-1}T \neq \emptyset$. Then for any $p' \in (t \cdot p)^{-1}T$, we have $t \cdot p \cdot p' \in T$, and hence $p \cdot p' \in t^{-1}T$. We reach a contradiction with $t^{-1}T = \emptyset$. □

Remark 4 For any tree t in T_Σ, the following statements hold:

1. If $Cor_T(t) \neq \theta$, then $Cor_T(t \cdot Cor_T(t)) = \xi$.
2. If $Cor_T(t) = \theta$, then $Cor_T(t \cdot p) = \theta$ for all $p \in C_\Sigma$, but the existence of a context $p \in C_\Sigma$ with $Cor_T(t \cdot p) = \theta$ does not imply that $Cor_T(t) = \theta$.

Proof. Let t be an arbitrary tree in T_Σ.

1. If $Cor_T(t) = p \neq \theta$, then $t \cdot p \in T$ which implies $t \cdot p \cdot \xi \in T$, and hence $\xi \in (t \cdot p)^{-1}T$. Because ξ is the smallest possible context, we obtain immediately that $Cor_T(t \cdot Cor_T(t)) = \xi$.
2. If $Cor_T(t) = \theta$, then $t^{-1}T = \emptyset$ which implies, using Remark 3, $(t \cdot p)^{-1}T = \emptyset$ for all $p \in C_\Sigma$, and hence $Cor_T(t \cdot p) = \theta$. Let us now consider the recognizer $A = (Q, \Sigma, \delta, F)$ from Example 1. If we take $t = f(a, g(a, b), b)$ and $p = g(\xi, b)$, it is clear that $Cor_T(t \cdot p) = \theta$, but $Cor_T(t) = \xi \neq \theta$.

□

3.2. Observation Tables

Let U be a set of trees in T_Σ and P a set of contexts in C_Σ. Then, U is called *subtree-closed* if $t \in U$ implies that all subtrees of t are elements of U. The set P is called *ξ-prefix closed* with respect to U if $p \in P \backslash \{\xi\}$ implies that there exists p' in P such that $p = f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \cdot p'$ for $m > 0$, $f \in \Sigma_m$ and $t_1, \ldots, t_{i-1}$, $t_{i+1}, \ldots, t_m \in U$. Let $Composed(U)$ be the set $\{f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \mid m > 0, f \in \Sigma_m, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m \in U\}$.

For the tree language T, an *observation table* (S, E, Cor) consists of a non-empty finite subtree-closed set S of trees, a non-empty finite set E of contexts which is ξ-prefix closed with respect to S, and the restriction Cor of the mapping Cor_T to the set $\{s \cdot e \mid s \in S \cup S\Sigma, e \in E\}$, denoted by $(S \cup S\Sigma)E$. The interpretation of Cor is that, for any $s \in S \cup S\Sigma$ and $e \in E$, $Cor(s \cdot e) = p$ iff $p \in C_\Sigma$ is the minimal context such that $s \cdot e \cdot p$ is accepted by the target tree recognizer A_T.

Observation table		E		
		$\ldots$	e	$\ldots$
S	$\vdots$ s $\vdots$	$\ldots$	$\vdots$ $Cor(s \cdot e)$ $\vdots$	$\ldots$
$S\Sigma \backslash S$	$\vdots$ $f(s_1, s_2, \ldots, s_m)$ $\vdots$	$\ldots$	$\vdots$ $Cor(f(s_1, s_2, \ldots, s_m) \cdot e)$ $\vdots$	$\ldots$

Table 1: The structure of an observation table (S,E,Cor)

Using an approach similar to that of Angluin [1], an observation table can be given as a two-dimensional array with rows labeled by elements of $S \cup S\Sigma$, columns labeled by elements of E, and the entry for row s and column e equal to $Cor(s \cdot e)$. It is a well-known fact that the Myhill-Nerode theorem can be extended to regular tree languages which means that every state of the minimal recognizer can be uniquely

identified by its frontier derivative language. Practically, asking this set as a correction is quite difficult since most of the times it is infinite. The choice that we made (to ask for the minimal context of that set) is justified by practical reasons: there is an efficient algorithm which can compute it. Moreover, the elements in E help us to distinguish between two different states which have the same minimal context (for example, if there are two trees $s_1, s_2 \in S$ belonging to different equivalence classes for which $Cor_T(s_1) = Cor_T(s_2)$, E should, at some point, contain a context e such that $Cor_T(s_1 \cdot e) \neq Cor_T(s_2 \cdot e)$).

The algorithm $L_{RTL}C$ will use the observation table to build a tree recognizer. Rows labeled by the elements of S are candidates for states of the recognizer being constructed, and columns labeled by the elements of E correspond to distinguishing experiments for these states. Rows labeled by elements of $S\Sigma$ are used to construct the transition function. An useful intuitive image is presented in Table 1.

If s is an element of $S \cup S\Sigma$, row_s denotes the finite function from E to $C_\Sigma \cup \{\theta\}$ defined by $row_s(e) = Cor(s \cdot e)$ and represents the observed behavior of the tree s. By $rows(S)$ we understand $\{row_s \mid s \in S\}$.

Definition 1 (closed, consistent and complete observation table) *An observation table (S, E, Cor) is called:*

- closed *if for every s in $S\Sigma$, there exists s' in S such that $row_s = row_{s'}$;*
- consistent *if for any s_1 and s_2 in S such that $row_{s_1} = row_{s_2}$, we have $row_{f(t_1,\ldots,t_{i-1},s_1,t_{i+1},\ldots,t_m)} = row_{f(t_1,\ldots,t_{i-1},s_2,t_{i+1},\ldots,t_m)}$ for all $m > 0$, $f \in \Sigma_m$, $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m \in S$ and $1 \leq i \leq m$;*
- complete *if for any p in $\{Cor(s \cdot e) \mid s \in S, e \in E\}$, we have $trees(p) \subseteq S$.*

In other words a table is closed if the observed behavior of every element of $S\Sigma$ can already be seen from the behaviors of elements of S. It is consistent if, for every two trees in S which have the same observed behavior, the corresponding trees in $S\Sigma$ (the ones obtained by applying a depth one context with subtrees in S) must also have the same behaviors. A table is complete if all the implicit information which arises from the use of the correcting contexts is used (all the subtrees of the correcting contexts should be in the set S since they form a finite set, namely the set of the minimal trees which will be candidates for the states of the recognizer to be constructed).

If (S, E, Cor) is a closed, consistent and complete observation table, one can define the corresponding tree recognizer $A(S, E, Cor) = (Q, \Sigma, \delta, F)$ as follows:

- $Q = \{row_s \mid s \in S\}$,
- $F = \{row_s \mid s \in S \text{ and } Cor(s) = \xi\}$,
- $\delta_0(c) = row_c$ for every $c \in \Sigma_0$, and
- $\delta_m(row_{s_1}, \ldots, row_{s_m}, f) = row_{f(s_1,\ldots,s_m)}$ for every $m > 0$, $f \in \Sigma_m$ and $s_1, \ldots, s_m \in S$.

It is clear that $A(S, E, Cor)$ has at most one sink state $q_\theta = row_s$, where $Cor(s) = \theta$. (Note that if $Cor(s) = \theta$, by Remark 4 and the above construction of the recognizer, $Cor(s \cdot e) = \theta$ for all $e \in E$).

To see that this is a well-defined (deterministic) tree recognizer, note that if s_1 and s_2 are elements of S such that row_{s_1}=row_{s_2}, then $Cor(s_1) = Cor(s_1 \cdot \xi)$ and $Cor(s_2) = Cor(s_2 \cdot \xi)$ are defined and equal to each other since E contains ξ. Hence, F is well-defined. Since the observation table (S, E, Cor) is consistent, $row_{f(t_1,\ldots,t_{i-1},s_1,t_{i+1},\ldots,t_m)} = row_{f(t_1,\ldots,t_{i-1},s_2,t_{i+1},\ldots,t_m)}$ for $m > 0$, $f \in \Sigma_m$, $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m \in S$, and because it is closed, this common value is equal to row_s for some s in S. Thus δ is well-defined.

A recognizer $A = (Q, \Sigma, \delta, F)$ is *consistent* with the function Cor if for every s in $S \cup S\Sigma$ and e in E, the following statements hold:

1. $Cor(s \cdot e) = \theta \Leftrightarrow \delta(s \cdot e)$ is a sink state.
2. $Cor(s \cdot e) = p \Leftrightarrow \delta(s \cdot e \cdot p) \in F$ and p is the smallest context with this property.

The important fact about the recognizer $A(S, E, Cor)$ is the following.

Theorem 2 *If (S, E, Cor) is a closed, consistent and complete observation table, then the tree recognizer $A(S, E, Cor)$ is consistent with Cor. Any other tree recognizer consistent with Cor but not isomorphic with $A(S, E, Cor)$ must have more states.*

The theorem is proved by a sequence of straightforward lemmas.

Lemma 3 *Assume that (S, E, Cor) is a closed, consistent and complete observation table. For the recognizer $A(S, E, Cor)$ and for every s in $S \cup S\Sigma$, $\delta(s) = row_s$.*

Proof. It is clear from the definition of $A(S, E, Cor)$. □

Lemma 4 *Assume that (S, E, Cor) is a closed, consistent and complete observation table. For each s in $S \cup S\Sigma$ and e in E, there exists s' in S such that $\delta(s \cdot e) = \delta(s')$ and $Cor(s \cdot e) = Cor(s')$.*

Proof. Let s be an element from $S \cup S\Sigma$. We prove our lemma by induction on the depth of e. When e is ξ, by Lemma 3 we have $\delta(s) = row_s$. Since (S, E, Cor) is a closed table, there exists $s' \in S$ such that $row_{s'} = row_s$ which implies $Cor(s') = Cor(s)$ and $\delta(s') = \delta(s)$.

Next, suppose that the result holds for all e in E of depth at most h, and let e be an element of E which has the depth $h + 1$. Since E is ξ-prefix closed with respect to S, there exists $e' \in E$ of depth h such that $e = f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \cdot e'$ for some $m > 0$, $f \in \Sigma_m$ and $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m \in S$. Because (S, E, Cor) is closed, we can take s' in S such that row_s=$row_{s'}$ (hence, $Cor(s \cdot e) = Cor(s' \cdot e)$ and $\delta(s) = \delta(s')$).

Then, $\delta(s \cdot e) = \delta(s \cdot f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \cdot e')$
$= \delta(s' \cdot f(t_1, \ldots, t_{i-1}, \xi, t_{i+1}, \ldots, t_m) \cdot e')$, by the replacement lemma
$= \delta(f(t_1, \ldots, t_{i-1}, s', t_{i+1}, \ldots, t_m) \cdot e')$.

By the induction hypothesis on e', there exists s'' in S such that $\delta(s'') = \delta(f(t_1, \ldots, t_{i-1}, s', t_{i+1}, \ldots, t_m) \cdot e')$, $Cor(s'') = Cor(f(t_1, ..., t_{i-1}, s', t_{i+1}, ..., t_m) \cdot e')$ which implies $\delta(s \cdot e) = \delta(s'')$ and $Cor(s \cdot e) = Cor(s'')$. □

Lemma 5 *Assume that (S, E, Cor) is a closed, consistent and complete observation table. Then the recognizer $A = A(S, E, Cor)$ is consistent with the function Cor.*

Proof. Let s be in $S \cup S\Sigma$ and e in E. From Lemma 4 we know that there exists s' in S such that $\delta(s \cdot e) = \delta(s')$ and $Cor(s \cdot e) = Cor(s')$. So, it is enough to prove that for all s' in S, we have:

1. $Cor(s') = \theta \Leftrightarrow \delta(s')$ is a sink state.
2. $Cor(s') = p \Leftrightarrow \delta(s' \cdot p) \in F$ and p is the smallest context with this property.

For the first statement, it is enough to see that $Cor(s') = \theta$ if and only if $row_{s'}$ is a sink state, and by Lemma 3, if and only if $\delta(s')$ is a sink state. For the second one, we will first show that $Cor(s') = p$ implies $\delta(s' \cdot p) \in F$. If $p = \xi$, the result is immediate. Suppose p equals $p_1 \cdot p_2 \cdot \ldots \cdot p_n$ with $p_i \in C_\Sigma$ having the node labeled ξ at depth exactly 1 for all $i \in \{1, 2, \ldots, n\}$ and $n \geq 1$.

From the definition of $trees(p)$ it is clear that $trees(p_i) \subseteq trees(p)$ for every i in $\{1, 2, \ldots, n\}$. But (S, E, Cor) is complete, so $trees(p) \subseteq S$, and hence $trees(p_i) \subseteq S$. Using the fact that $depth(p_i) = 1$, we obtain that p_i belongs to $Composed(S)$. Then, because the table (S, E, Cor) is closed, we can inductively find the subset of trees $\{s_1, \ldots, s_n\} \subseteq S$, as it is shown in the sequel:

$s' \in S$, $p_1 \in Composed(S) \Rightarrow \exists\ s_1 \in S$ such that $\delta(s' \cdot p_1) = \delta(s_1)$, $Cor(s' \cdot p_1) = Cor(s_1)$

$s_1 \in S$, $p_2 \in Composed(S) \Rightarrow \exists\ s_2 \in S$ such that $\delta(s_1 \cdot p_2) = \delta(s_2)$, $Cor(s_1 \cdot p_2) = Cor(s_2)$

$\ldots$

$s_{n-1} \in S$, $p_n \in Composed(S) \Rightarrow \exists\ s_n \in S$ such that $\delta(s_{n-1} \cdot p_n) = \delta(s_n)$ and $Cor(s_{n-1} \cdot p_n) = Cor(s_n)$.

Clearly, $\delta(s' \cdot p) = \delta(s_n)$ (using the replacement lemma). Starting from $Cor(s') = p$ and applying several times Remark 2, we obtain $Cor(s_n) = \xi$ which is equivalent to $\delta(s_n) \in F$, and furthermore with $\delta(s' \cdot p) \in F$.

Next, we will show that if we take $p = p_1 \cdot p_2 \cdot \ldots \cdot p_n$ to be the smallest context in Knuth-Bendix order such that $\delta(s' \cdot p) \in F$, then $Cor(s') = p$. We know that the set $\{p \mid \delta(s' \cdot p) \in F\}$ is not empty because we proved that it contains the context $Cor(s')$.

We know that $\delta(s' \cdot p) = \delta(s_n)$ and $\delta(s_n) \in F$ implies $Cor(s_n) = \xi$. But $Cor(s_n) = Cor(s_{n-1} \cdot p_n)$, so $Cor(s_{n-1} \cdot p_n) = \xi$, and hence $p_n \in s_{n-1}^{-1} T$. We will show that this implies $Cor(s_{n-1}) = p_n$. Suppose that there exists a context $r_n <_{kbo} p_n$ such that $Cor(s_{n-1}) = r_n$. Then, we show that there exists a context p' strictly smaller than p such that $\delta(s' \cdot p') \in F$ which contradicts the choice we have made for p. We can take p' to be $p_1 \cdot p_2 \cdot \ldots \cdot p_{n-1} \cdot r_n$. Therefore, $\delta(s' \cdot p') = \delta(s' \cdot p_1 \cdot \ldots \cdot p_{n-1} \cdot r_n) = \delta(s_{n-1} \cdot r_n)$, and because $Cor(s_{n-1}) = r_n$, it follows that $\delta(s_{n-1} \cdot r_n) \in F$.

But $Cor(s_{n-2} \cdot p_{n-1}) = Cor(s_{n-1}) = p_n$. In an analogous way as above we obtain $Cor(s_{n-2}) = p_{n-1} \cdot p_n$ (otherwise $p'' = p_1 \cdot \ldots \cdot p_{n-2} \cdot r$, where $r = Cor(s_{n-2})$, $r <_{kbo} p_{n-1} \cdot p_n$ is a context strictly smaller than p such that $\delta(s' \cdot p'')$ is in F).

Reasoning in the same manner we obtain $Cor(s') = p_1 \cdot p_2 \cdot \ldots \cdot p_n$ which implies $Cor(s') = p$.

We showed that:

1. $Cor(s') = p$ implies $\delta(s' \cdot p) \in F$.
2. If p is the smallest context such that $\delta(s' \cdot p) \in F$, then $Cor(s') = p$.

If $Cor(s') = p$, then $\delta(s' \cdot p) \in F$ (cf. 1). Now, assume p is not the smallest context such that $\delta(s' \cdot p) \in F$. Let us take $p' <_{kbo} p$ to be the smallest context with this property. Then, cf. 2, $Cor(s') = p'$, and hence $p = p'$. This concludes the proof of the lemma.

□

Lemma 6 *Assume that (S, E, Cor) is a closed, consistent and complete observation table. Suppose that the recognizer $A(S, E, Cor)$ has n states. If $A' = (Q', \Sigma, \delta', F')$ is any recognizer consistent with Cor that has n or fewer states, then A' is isomorphic with $A(S, E, Cor)$.*

Proof. We define the relation $\phi \subseteq Q \times Q'$ as follows: for each $s \in S$, $row_s \ \phi \ q' \Leftrightarrow \delta'(s) = q'$.

Let us take $s_1, s_2 \in S$ such that there exists $q' \in Q'$, $row_{s_1} \ \phi \ q'$ and $row_{s_2} \ \phi \ q'$ (clearly, $\delta'(s_1) = q' = \delta'(s_2)$). We will show that this implies $row_{s_1} = row_{s_2}$. Suppose by contrary that $row_{s_1} \neq row_{s_2}$. Then, there exists $e \in E$ such that $row_{s_1}(e) \neq row_{s_2}(e)$, and so $Cor(s_1 \cdot e) \neq Cor(s_2 \cdot e)$. We distinguish two cases: $Cor(s_1 \cdot e) = \theta$, $Cor(s_2 \cdot e) \neq \theta$, and $Cor(s_1 \cdot e), Cor(s_2 \cdot e) \neq \theta$.

Case I) $Cor(s_1 \cdot e) = \theta$, $Cor(s_2 \cdot e) = p \neq \theta$.

Because A' is consistent with Cor, $\delta'(s_1 \cdot e)$ is a sink state and $\delta'(s_2 \cdot e \cdot p) \in F'$. But $\delta'(s_1) = \delta'(s_2)$, and so $\delta'(s_1 \cdot e) = \delta'(s_2 \cdot e)$. Hence, $\delta'(s_2 \cdot e)$ is a sink state. This means that $\delta'(s_2 \cdot e \cdot p)$ is a sink state which contradicts $\delta'(s_2 \cdot e \cdot p) \in F'$.

Case II) $Cor(s_1 \cdot e) = p_1$, $Cor(s_2 \cdot e) = p_2$, $p_1 \neq p_2$ and $p_1, p_2 \neq \theta$.

Because A' is consistent with Cor, we have $\delta'(s_1 \cdot e \cdot p_1) \in F'$, $\delta'(s_2 \cdot e \cdot p_2) \in F'$ and p_1, p_2 are the smallest contexts with this property. But $\delta'(s_1) = \delta'(s_2)$, and so $\delta'(s_1 \cdot e \cdot p_1) = \delta'(s_2 \cdot e \cdot p_1)$. Hence, $\delta'(s_2 \cdot e \cdot p_1) \in F'$ which implies $p_2 \leq_{kbo} p_1$. In a similar way it can be shown that $p_1 \leq_{kbo} p_2$. We draw the conclusion that $p_1 = p_2$ which leads to a contradiction.

We have shown that the relation ϕ is an injection. This implies that $|Q| \leq |\phi(Q)|$. From our hypothesis we know that $|Q'| \leq |Q|$. So, $|Q| \leq |\phi(Q)| \leq |Q'| \leq |Q|$ implies $|Q| = |\phi(Q)| = |Q'|$ which makes our relation ϕ to be a function.

Because the function ϕ is injective and has the domain and range finite and of the same cardinality, it follows immediately that ϕ is surjective, and hence bijective.

We will show that ϕ is an isomorphism from A to A', that is, it preserves the transition function and $\phi(F) = F'$:

1. We have $q' \in \phi(F) \Leftrightarrow \exists s \in S$ such that $row_s \in F$ and $\phi(row_s) = q' \Leftrightarrow \exists s \in S$ such that $Cor(s) = \xi$ and $\delta'(s) = q'$. Because A' is consistent with Cor, this is equivalent to $\exists s \in S$ such that $\delta'(s) \in F'$ and $\delta'(s) = q'$. Hence, $q' \in F'$.
2. Let $c \in \Sigma_0 \cap S$. We have $\phi(\delta_0(c)) = \phi(row_c) = \delta'_0(c)$.

3. Let us take $m > 0$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in S$. We want to show that $\phi(\delta_m(row_{t_1}, \ldots, row_{t_m}, f)) = \delta'_m(\phi(row_{t_1}), \ldots, \phi(row_{t_m}), f)$. Indeed, $\phi(\delta_m(row_{t_1}, \ldots, row_{t_m}, f)) = \phi(row_{f(t_1,\ldots,t_m)}) = \phi(row_{s'}) = \delta'(s')$ with $s' \in S$ such that $row_{s'} = row_{f(t_1,\ldots,t_m)}$. Moreover, $\delta'_m(\phi(row_{t_1}), \ldots, \phi(row_{t_m}), f) = \delta'_m(\delta'(t_1), \ldots, \delta'(t_m), f) = \delta'(f(t_1, ..., t_m))$. Since $\delta'(s')$ and $\delta'(f(t_1, \ldots, t_m))$ have identical row values, namely $row_{s'}$ and $row_{f(t_1,\ldots,t_m)}$, they must be the same state of A', and hence we obtain that $\phi(\delta_m(row_{t_1}, ..., row_{t_m}, f))$ is equal with $\delta'_m(\phi(row_{t_1}), \ldots, \phi(row_{t_m}), f)$.

This concludes the proof of Lemma 6. □

Now the proof of Theorem 2 follows since from Lemma 5 we know that $A(S, E, Cor)$ is consistent with Cor and Lemma 6 shows that any other recognizer consistent with Cor is either isomorphic to $A(S, E, Cor)$ or contains at least one more state. Thus, $A(S, E, Cor)$ is the unique minimal tree recognizer consistent with Cor.

3.3. The Learner $L_{RTL}C$

The algorithm can be seen as an interaction between two actors: the *Learner* who must identify the target language being allowed to use specific kinds of questions (CQs and EQs), and the *Teacher* who knows this language and is assumed to answer correctly to the questions.

Algorithm 1 Learning Regular Tree Languages from Corrections Algorithm $L_{RTL}C$

```
1: Initialize S as {a} for an arbitrarily fixed a ∈ Σ_0, and E as {ξ};
2: UPDATE(Tab);
3: repeat
4:   repeat
5:     while Tab is not closed do
6:       Tab:=CLOSURE(Tab);
7:     end while
8:     while Tab is not consistent do
9:       Tab:=CONSISTENCY(Tab);
10:    end while
11:    if Tab is not complete then
12:      Tab:=COMPLETENESS(Tab);
13:    end if
14:  until Tab is closed and consistent
15:  eq:=EQUIV(Tab);
16: until eq="yes"
17: Return A(S, E, Cor);
```

In what follows we explain the steps performed by the *Learner* in order to identify the target language. The Learner algorithm uses as its main data structure the observation table that we described in the previous section. We denote an arbitrary observation table (S, E, Cor) by Tab.

The Learner starts with an initial observation table $Tab = (S, E, Cor)$, where for an arbitrarily fixed a in Σ_0, $S = \{a\}$, $E = \{\xi\}$, and the value $Cor(a \cdot \xi)$ is obtained by asking a correction query. The procedure UPDATE receives as a parameter an observation table and asks correction queries for all the entries in the table where there is no information available. The goal of the inner loop is to construct a closed, consistent and complete observation table.

```
Procedure CLOSURE(Tab)
    find s in SΣ such that row_s ∉ rows(S);
    S := S ∪ {s};
    UPDATE(Tab);
    return Tab;
```

The procedure CLOSURE is very simple. It just searches for a tree $s \in S\Sigma$ such that $row_s \notin rows(S)$ and adds it to S. After that it updates the table received as a parameter.

```
Procedure CONSISTENCY(Tab)
    find s_1, s_2 ∈ S, m > 0, f ∈ Σ_m, 1 ≤ i ≤ m, t_1, ..., t_{i-1}, t_{i+1}, ..., t_m ∈ S
    and e ∈ E such that
     row_{s_1} = row_{s_2} and
     Cor(f(t_1, ..., t_{i-1}, s_1, t_{i+1}, ..., t_m)·e) ≠ Cor(f(t_1, ..., t_{i-1}, s_2, t_{i+1}, ..., t_m)·e);
    E := E ∪ {f(t_1, ..., t_{i-1}, ξ, t_{i+1}, ..., t_m) · e};
    UPDATE(Tab);
    return Tab;
```

The procedure CONSISTENCY searches for trees in S which have the same row values, and hence seem to represent the same state of the automaton, but which have a different behavior once we apply a context p in $Composed(S)$.The procedure adds a new experiment to E in order to distinguish between these two states.

```
Procedure COMPLETENESS(Tab)
    while there exist s ∈ S and e ∈ E such that trees(Cor(s · e)) ⊈ S do
       for all t ∈ trees(Cor(s · e))\S
          S := S ∪ {t};
       end for
       UPDATE(Tab);
       COMPLETENESS(Tab);
    end while
    return Tab;
```

The procedure COMPLETENESS is recursive, but terminating. It is enough to see that the contexts returned by the Teacher have a special feature. Each subtree of those contexts is a minimal tree in its equivalence class. Because T is a regular tree language, there are a finite number of equivalence classes, and hence the table can be found not complete at most n times, where n represents the number of states of the minimal recognizer A_T.

The procedure EQUIV just asks the Teacher if the conjectured recognizer is equivalent to the target tree recognizer. If the answer is no, it takes the counterexample returned by the Teacher and adds it to S, along with all its subtrees. After that it updates the observation table.

```
Procedure EQUIV(Tab)
    construct A = A(S, E, Cor);
    if A and A_T are equivalent then return "yes";
    else get a counterexample t;
         for all t' ∈ sub(t)
            S := S ∪ {t'};
         end for
         UPDATE(Tab);
         return "no";
    end if
```

If the algorithm terminates, it obviously returns the correct recognizer. It is also clear that $L_{RTL}C$ halts in finitely many steps since:

- whenever the table is found not closed or not consistent the number of distinct rows in S increases by at least one;
- the procedure COMPLETENESS is performed at most n times;
- for any closed, consistent and complete observation table (S, E, Cor), if d denotes the number of different values of row_s for s in S, then any recognizer consistent with Cor must have at least d states (from the injectivity of function ϕ defined in Lemma 6);
- $L_{RTL}C$ can make at most $n - 1$ incorrect conjectures since the size of the conjectured recognizer is initially at least one and may not exceed n (whenever the Teacher answers by a counterexample the number of distinct values of row_s for s in S increases by at least 1).

Hence, $L_{RTL}C$ always eventually finds a closed, consistent and complete observation table (S, E, Cor) and makes a conjecture $A(S, E, Cor)$. Since $L_{RTL}C$ has to make another conjecture as long as it is running, it must terminate by making a correct conjecture.

Let us now discuss the time complexity of the algorithm $L_{RTL}C$. In what follows we denote by $|M|$ the cardinality of the set M. The total running time of $L_{RTL}C$ could be bounded by a polynomial in n (n is the number of states in A_T) if the Teacher always returns the minimal possible counterexample. However, there is no restriction on the size of the counterexample. Hence, we have to use both n and the maximum size of the counterexamples (denoted by m in the sequel) as parameters when describing the complexity of the algorithm.

We will first show that all the procedures run in time polynomial in the size of the observation table and that the final observation table is polynomial in n and m (although it is exponential in the maximum rank of symbols).

One can notice that the procedures CLOSURE(Tab) and CONSISTENCY(Tab) cannot be called more than $n-1$ times since each of them increases the number of distinct rows from S, and this number is initially one and cannot exceed n. The same statement also holds for the procedure COMPLETENESS(Tab): it cannot be called more than n times since the total number of minimal tree representatives for the states coincides with the number of states. When the observation table is closed, consistent and complete, the algorithm runs the procedure EQUIV(Tab), and this can happen no more than n times since any counterexample adds at least one distinct row to the set S.

To compute CLOSURE(Tab), we will have to consider each pair in $S \times (S\Sigma \backslash S)$ and compare the observed behaviors of the two trees (in the worst case). This task can be done using $O(|S| \cdot |S\Sigma \backslash S| \cdot |E|)$ comparisons.

To compute CONSISTENCY(Tab), we first find the trees s, s' in S such that $row_s = row_{s'}$. This requires, in the worst case, $O(|S|^2 \cdot |E|)$ operations. Then, we try to find $m > 0$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in S$ such that $row_{f(t_1,\ldots,t_{i-1},s,t_{i+1},\ldots,t_m)} \neq row_{f(t_1,\ldots,t_{i-1},s',t_{i+1},\ldots,t_m)}$. In the worst case the algorithm would have to perform $O(k \cdot |S|^{r-1} \cdot |E|)$ comparisons, where k is the size of the alphabet and r the maximum arity of the symbols in Σ.

To compute COMPLETENESS(Tab), for all contexts in $\{Cor(s \cdot e) \mid s \in S, e \in E\}$, we need to see if their direct subtrees are also in S. This can be done in $O(n(r-1) \cdot |S|^2 \cdot |E|)$ steps.

To compute EQUIV(Tab), we first construct $A(S, E, Cor)$ in time polynomial in the size of the observation table. A counterexample requires the addition of at most m trees of size at most m to S, and this can happen at most $n-1$ times.

For the procedure UPDATE(Tab) it is clear that the number of correction queries asked coincides with the size of the final observation table. Regarding the dimensions of S, $S\Sigma\backslash S$ and E, it is easy to see that the number of trees in S cannot exceed $2n+m(n-1)$ (it starts with one tree, the procedures CLOSURE and CONSISTENCY together can add at most $n-1$ trees, EQUIV at most $m(n-1)$ trees, and COMPLETENESS at most n trees), $S\Sigma\backslash S$ has at most $k|S|^r$ elements, and the number of contexts in E cannot be greater than n.

Hence, the total running time of $L_{RTL}C$ can be bounded by a polynomial function of m and n.

3.4. The Teacher

In this section we show that there exists an algorithm which can compute the answers to correction queries in polynomial time in the size of the finite tree recognizer to be learned. Actually, the Teacher can give the answer in linear time if some precomputation is done (see Algorithm 2, lines 1-6). We just have to precompute for each state q its minimal context c^q (this computation is polynomial in the size of the target recognizer). The algorithm presented below first determines for each q the minimal tree t^q such that $\delta(t^q) = q$, and then computes the minimal context c^q such that $\delta(t \cdot c^q) \in F$ for all $t \in T_\Sigma$ with $\delta(t) = q$, where $A_T = (Q, \Sigma, \delta, F)$ is the minimal tree recognizer for the target tree language T. When the Teacher receives a tree t as

an input, it is enough to compute $\delta(t)$ (which can be done in linear time) and return $c^{\delta(t)}$.

In the process of computing the minimal trees we have to consider only the non-recursive rules, that is rules of the form $\delta_m(q_1, \ldots, q_m, f) = q$ such that $q \notin \{q_1, \ldots, q_m\}$. We show that recursive rules do not help in the process of finding the minimal tree.

Algorithm 2 Computing Minimal Context Algorithm

1: **for** all $q \in Q$ **do**
2: t^q:=MinT$(q, \emptyset)$;
3: **end for**
4: **for** all $q \in Q$ **do**
5: c^q:=MinC$(q, \emptyset)$;
6: **end for**
7: **for** any input tree t submitted by the Learner **do**
8: $q := \delta(t)$;
9: return c^q;
10: **end for**

Indeed, let us suppose we know that the minimal trees for the states $q_1, \ldots, q_{i-1}$, $q_{i+1}, \ldots, q_m$ are $t_1, \ldots, t_{i-1}$, $t_{i+1}, \ldots, t_m$, respectively, and we want to compute the minimal tree for q using the rule $\delta_m(q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_m, f) = q$. It is clear that the tree $f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_m)$ is always greater than the tree t, no matter what are the weights of $f, t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_m, t$ (this is true even for the case in which $m = 1$, $w(f) = 0$, $t <_{kbo} f(t)$).

```
Procedure MinT(q, P)
    MinTree:=θ;
    for all rules δ_m(q_1, ..., q_m, f) = q in R_t^P do
       if MinT(q_i, P ∪ {q}) ≠ θ for all i ∈ {1, ..., m} then
          t := f(MinT(q_1, P ∪ {q}), ..., MinT(q_m, P ∪ {q}));
          if MinTree = θ then MinTree:=t
          else
             if t <_kbo MinTree then MinTree:=t
             end if
          end if
       end if
    end for
    return MinTree;
```

Also, we have to eliminate the loops from our computation. For example, if we want to compute the minimal tree for a state q using the rule $\delta_m(q_1, \ldots, q_m, f) = q$, then in the recursive process of computing minimal trees for all states q_i we have to eliminate all rules which contain the state q. Otherwise, we will find ourselves in a similar situation to the one described above. That is why we need to introduce

the set P of "not allowed" states. We construct the set R_t^P by eliminating from the set of all rules first the recursive ones and after that all the rules of the form $\delta_m(q_1, \ldots, q_m, f) = q$, where at least one q_i is in P.

The existence of a minimal tree for each state is guaranteed since the target recognizer is minimal, and hence all its states are reachable. The symbol θ is used to indicate that the minimal tree has not been found yet or that there is no minimal tree for that state in the given circumstances (we can think of the set P as a supplementary restriction imposed on the search space).

In order to construct the minimal context for each state we also need to remove from the search space the rules which would generate cycles. More precisely, suppose we know that the state q appears in the left hand side only in the rule $\delta_m(q_1, \ldots, q_{i-1}, q, q_{i+1}, \ldots, q_m, f) = q'$ and q' appears in the left hand side of two rules and one of them is $\delta_m(q'_1, \ldots, q', \ldots, q'_l, g) = q$. It is clear that we should not use this rule to compute the minimal context, first of all because we get into a loop, and secondly because $c <_{kbo} g(t^{q'_1}, \ldots, f(t^{q_1}, \ldots, t^{q_{i-1}}, \xi, t^{q_{i+1}}, \ldots, t^{q_{m-1}}), \ldots, t^{q'_l}) \cdot c$ for any context $c \in C_\Sigma$. In order to avoid this kind of loops we introduce the set R_c^P which contains only rules of the form $\delta_m(q_1, \ldots, q_m, f) = q$ in which $q \notin P$.

```
Procedure MinC(q, P)
    if q ∈ F then MinContext:=ξ;
    else
       MinContext:=θ;
       for all rules δ_m(q_1, ..., q_{i-1}, q, q_{i+1}, ..., q_m, f) = q' in R_c^{P∪{q}} do
          if MinC(q', P ∪ {q}) ≠ θ then
             c := f(t^{q_1}, ..., t^{q_{i-1}}, ξ, t^{q_{i+1}}, ..., t^{q_m})·MinC(q', P ∪ {q});
             if MinContext= θ then MinContext:=c
             else
                if c <_kbo MinContext then MinContext:=c;
                end if
             end if
          end if
       end for
    end if
    return MinContext;
```

It is also obvious that choosing to replace each state q by the minimal tree t^q is going to give us the smallest correcting context because if we take any other tree t, then $t^q \cdot c <_{kbo} t \cdot c$ for any context $c \in C_\Sigma$. Choosing to stop the search once we found a final state is justified by the properties of the Knuth-Bendix order. Continuing the search on the same path would give us only bigger and bigger contexts.

The existence of a minimal context for all states except for the sink state is also guaranteed since the target recognizer is minimal, and hence there is a most one state which is not co-reachable (the sink state). The symbol θ is used here for similar reasons: to indicate that either the minimal context has not been found yet, or that, given the set of rules which can be applied, there is no minimal context for that state.

3.5. Running Example

In what follows we show how our algorithm works on an example. Let $\Sigma = \{f/3, g/2, a/0, b/0\}$ be a ranked alphabet and the following order $<$ among the symbols of the alphabet: $a < b < g < f$. The weight function $w : \Sigma \cup \{\xi\} \rightarrow \mathbb{R}_0^+$ is defined by: $w(\xi) = w(a) = w(b) = 1$, $w(g) = 2$ and $w(f) = 3$. Suppose the language to be learned is $T = \{g(a,b) \cdot [f(a,\xi,b)]^n \mid n \geq 0\} \cup \{g(b,b) \cdot [f(a,\xi,b)]^n \mid n \geq 0\}$, where $[f(a,\xi,b) = \xi]^0$ and $[f(a,\xi,b)]^n = [f(a,\xi,b)]^{n-1} \cdot f(a,\xi,b)$, $n \geq 1$.

Clearly, T is a regular tree language and $A_T = (Q, \Sigma, \delta, F)$ the minimal tree recognizer for T, where $Q = \{q_a, q_b, q_s, q_f\}$, $F = \{q_f\}$, and the transition function δ is defined as follows:

- $\delta_0(a) = q_a$, $\delta_0(b) = q_b$,
- $\delta_2(q_a, q_b, g) = q_f$, $\delta_2(q_b, q_b, g) = q_f$, $\delta_2(q_x, q_a, g) = q_s$ for $x \in \{a, b\}$,
- $\delta_3(q_a, q_f, q_b, f) = q_f$, and $\delta_3(q_x, q_y, q_z, f) = q_s$ for all remaining cases.

The algorithm starts by constructing the following observation table, where $S = \{a\}$ and $E = \{\xi\}$ (actually S can contain any of the two symbols with arity 0, i.e., a or b).

Observation table		E
		ξ
S	a	$g(\xi, b)$
$S\Sigma\backslash S$	b	$g(\xi, b)$
	$g(a,a)$	θ
	$f(a,a,a)$	θ

Table 2: The observation table for $S = \{a\}$ and $E = \{\xi\}$

The observation table from Table 2 is not closed because $row_{g(a,a)} \notin rows(S)$, and the algorithm proceeds by adding the tree $g(a,a)$ to S. Updating the current table, we get the observation table from Table 3.

Observation table		E
		ξ
S	a	$g(\xi, b)$
	$g(a,a)$	θ
$S\Sigma\backslash S$	b	$g(\xi, b)$
	$g(a, g(a,a))$	θ
	$\vdots$	$\vdots$
	$f(g(a,a), g(a,a), g(a,a))$	θ

Table 3: The observation table for $S = \{a, g(a,a)\}$ and $E = \{\xi\}$

This table is closed and consistent but not complete (we have $g(\xi, b) \in Cor(S \cdot E)$, and $trees(g(\xi, b)) = \{b\} \not\subseteq S$). Hence, $L_{RTL}C$ adds the tree b to S and updates the table (see Table 4).

Observation table		E
		ξ
S	a	$g(\xi, b)$
	b	$g(\xi, b)$
	$g(a, a)$	θ
$S\Sigma \backslash S$	$g(a, b)$	ξ
	$g(b, a)$	θ
	$g(b, b)$	ξ
	$\vdots$	$\vdots$
	$f(g(a, a), g(a, a), g(a, a))$	θ

Table 4: The observation table for $S = \{a, b, g(a, a)\}$ and $E = \{\xi\}$

This table is not closed because $row_{g(a,b)} \notin rows(S)$ and not consistent: $row_a = row_b$ but $row_{g(a,a)} = \theta \neq \xi = row_{g(a,b)}$. The algorithm continues by adding the tree $g(a, b)$ to S and context $g(a, \xi)$ to E. We get the observation table from Table 5.

Observation table		E		States
		ξ	$g(a, \xi)$	
S	a	$g(\xi, b)$	θ	q_a
	b	$g(\xi, b)$	ξ	q_b
	$g(a, a)$	θ	θ	q_s
	$g(a, b)$	ξ	θ	q_f
$S\Sigma \backslash S$	$g(b, a)$	θ	θ	q_s
	$g(b, b)$	ξ	θ	q_f
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$f(a, g(a, a), b)$	θ	θ	q_s
	$f(a, g(a, b), b)$	ξ	θ	q_f
	$\vdots$	$\vdots$	$\vdots$	$\vdots$
	$f(g(a, b), g(a, b), g(a, b))$	θ	θ	q_s

Table 5: The observation table for $S = \{a, b, g(a, a), g(a, b)\}$ and $E = \{\xi, g(a, \xi)\}$

The above observation table is closed, consistent and complete, and the conjectured recognizer $A(S, E, Cor)$ is isomorphic to A_T. Hence, the Teacher's answer to the EQ will be "yes". The algorithm $L_{RTL}C$ outputs $A(S, E, Cor)$ and halts.

4. The Class of Injective Languages

In this section we show that there exists an infinite class of tree languages learnable from correction queries only, and we present a restricted version of $L_{RTL}C$, namely $L_{RTL}C^{inj}$.

Definition 2 (injective languages) *A tree language T is* injective *(or T has the* injectivity property*) if for any two trees t_1, t_2 in T_Σ, $t_1 \equiv_T t_2 \Leftrightarrow Cor_T(t_1) = Cor_T(t_2)$. The subclass of regular tree languages which have the injectivity property is denoted by $\mathcal{I}nj$.*

Note that one implication always holds: $t_1 \equiv_T t_2 \Rightarrow Cor_T(t_1) = Cor_T(t_2)$. It is an easy exercise to see that the tree language from Example 1 belongs to the class $\mathcal{I}nj$ and the one from Section 3.5 does not.

Theorem 7 *For any tree language T in $\mathcal{I}nj$, the algorithm asks only one EQ when learning.*

Proof. For the two trivial languages $T = \emptyset$ and $T = T_\Sigma$ the proof is immediate. Hence, let T be a non-trivial tree language in $\mathcal{I}nj$ and $A_T = (Q, \Sigma, \delta, F)$ the minimal tree recognizer accepting T with $|Q| = n > 1$.

We show by induction that, at any step $k < n$ of the algorithm, the table (S, E, Cor) has the following properties: $E = \{\xi\}$, $|S| = k$, and (S, E, Cor) is not closed but consistent.

For $k = 1$, $E = \{\xi\}$ and $S = \{a\}$, where $a \in \Sigma_0$, it is clear that the table is not closed since this would imply that the target recognizer has only one state which contradicts the non-triviality of T.

Suppose that the result holds for all steps strictly smaller than k, and we want to prove that the above three conditions are satisfied at step k ($k < n$). Since at step $k-1$ the set S had $k-1$ trees and the table was not closed, it means that at step k after procedure CLOSURE the set S has one more element (and hence k elements), E continues to be $\{\xi\}$, and clearly the table is consistent. The only fact that needs to be shown is that (S, E, Cor) is not closed.

Assume by contradiction that (S, E, Cor) is closed. Then, we can construct a recognizer $A' = A(S, E, Cor) = (Q', \Sigma, \delta', F')$ with $|Q'| = k$, as in Section 3.2 (note that the observation table does not need to be complete in order to construct such a recognizer).

We define $\phi : Q' \to Q$ by $\phi(row_s) := \delta(s)$. We prove that the following statements hold: ϕ is well-defined and injective, $\phi(F') \subseteq F$, $\phi(\delta'_m(row_{s_1}, \ldots, row_{s_m}, f)) = \delta_m(\phi(row_{s_1}), \ldots, \phi(row_{s_m}), f)$ for $m \geq 0$, $f \in \Sigma_m$, $s_1, \ldots, s_m \in S$.

Clearly, ϕ is well-defined since there are no two trees $s_1 \neq s_2$ in S such that $row_{s_1} = row_{s_2}$. To see that ϕ is injective, let us take two distinct states in Q', namely row_{s_1} and row_{s_2}. Because $E = \{\xi\}$, $row_{s_1} \neq row_{s_2}$ is equivalent to $Cor_T(s_1) \neq Cor_T(s_2)$, and since T has the injectivity property, this is equivalent to $s_1 \not\equiv_T s_2 \Leftrightarrow \delta(s_1) \neq \delta(s_2) \Leftrightarrow \phi(row_{s_1}) \neq \phi(row_{s_2})$. Thus, ϕ is injective.

For any q in $\phi(F')$, there exists s in S such that $row_s \in F'$ and $\phi(row_s) = q$ which is equivalent to $Cor_T(s) = \xi$ and $\delta(s) = q$. Hence, q is in F.

We have $\delta'_m(row_{s_1}, \ldots, row_{s_m}, f) = row_{f(s_1,\ldots,s_m)} = row_s$, where $s \in S$, and hence $\phi(\delta'_m(row_{s_1}, \ldots, row_{s_m}, f)) = \phi(row_s) = \delta(s)$. But $\delta(s) = \delta(f(s_1, \ldots, s_m))$ because $f(s_1, \ldots, s_m) \equiv_T s$ since $Cor_T(f(s_1, \ldots, s_m)) = Cor_T(s)$ and the tree language T is injective. So, $\phi(\delta'_m(row_{s_1}, \ldots, row_{s_m}, f)) = \delta(f(s_1, \ldots, s_m)) = \delta_m(\delta(s_1), \ldots, \delta(s_m), f) = \delta_m(\phi(row_{s_1}), \ldots, \phi(row_{s_m}), f)$.

Clearly, A' is a complete recognizer (i.e., for all $m \geq 0$, δ'_m is a total function). Now it is enough to see that we have constructed an injective morphism from A' to A_T such that $|Q'| = k < n = |Q|$ which leads us to a contradiction. Hence, (S, E, Cor) is not closed.

We proved that the procedure CLOSURE(Tab) is called at least $n - 1$ times. Obviously, this means that by that time S already has n different rows, and since it cannot have more (the total number of states in the target recognizer is n), in step n the table will be closed and consistent. After running the procedure COMPLETNESS (which cannot add any new row because the table already contains n different rows), the output recognizer will be the target one, and the answer to the EQ will be yes. □

Corollary 8 *The class $\mathcal{I}nj$ is learnable from correction queries only.*

Proof. One can modify $L_{RTL}C$ to output the conjectured automaton $A(S, E, Cor)$ and halt when the table is closed, consistent and complete. From the previous theorem it is clear that the algorithm will return the target automaton. □

The restricted version of $L_{RTL}C$ for the subclass of injective languages is presented in Algorithm 3.

Algorithm 3 Learning Injective Languages from Corrections Algorithm $L_{RTL}C^{inj}$

```
1: Initialize S as {a} for an arbitrarily fixed a ∈ Σ_0, and E as {ξ};
2: UPDATE(Tab);
3: while Tab is not closed do
4:    Tab:=CLOSURE(Tab);
5: end while
6: Return A(S, E, Cor);
```

5. Concluding Remarks

We introduced the notions of correcting contexts and complete observation tables, and we showed that regular tree languages can be learned from correction and equivalence queries. We considered that in the case of regular tree languages it was worth describing how the Teacher reacts and returns counterexamples because this is not trivial. Also, we proved that injective tree languages do not need equivalence queries in order to be learned in this framework.

This new approach seems to be interesting not only from a theoretical point of view, but also for the idea which is behind the algorithm, namely learning from corrections. Let us imagine that the oracle is the world wide web. Clearly, we can think of ways to implement corrections (not necessarily the one defined in this article). But for the web to answer an equivalence query will be practically impossible. Another practical reason for which eliminating the equivalence queries seems relevant is that it might happen that we have access to language elements but not to the device which produces the language.

Our algorithm, as well as the ones from [4, 14], has the disadvantage that it runs in exponential time in the maximum rank of symbols from the alphabet. A possible improvement could be obtained by adapting our idea of correcting contexts to the algorithm presented in [5].

In the future we would like to study other types of correcting contexts (which would allow different subclasses of regular tree languages to be learnable without equivalence queries), and to extend $L_{RTL}C$ for inferring weighted tree automata which are very useful tools for practitioners.

Finally, regular tree languages provide a versatile toolkit for building mathematical models of various subjects like, for example, XML documents. In particular we are interested in linguistic applications. The leading idea is that the input in such applications is formed by annotated trees. The annotations consist in semantic information which facilitates the learning process. Such data are widely available and come from many computational linguistic formal grammars such as tree adjoining grammars or categorial grammars.

Acknowledgments

We are grateful to professors Magnus Steinby, Victor Mitrana and Zoltán Ésik for valuable advices and a careful review. Also, many thanks to the anonymous reviewers who helped us improving the present paper.

References

[1] D. Angluin, Learning regular sets from queries and counterexamples. *Information and Computation* **75** (1987), 87–106.

[2] F. Baader, T. Nipkow, *Term Rewriting and All That*. Cambridge University Press, New York, 1998.

[3] L. Beccera-Bonache, A. H. Dediu, C. Tîrnăucă, Learning DFA from correction and equivalence queries. In: Y. Sakaibara, S. Kobayashi, K. Sato, T. Nishino, E. Tomita (eds.), *Proc., 8th Int. Colloq. on Grammatical Inference (ICGI)*. LNAI 4201, Springer-Verlag, Berlin Heidelberg, 2006, 281–292.

[4] F. Drewes, J. Högberg, Learning a regular tree language from a teacher. In: Z. Ésik, Z. Fülöp (eds.), *Proc., 7th Int. Conf. on Developments in Language Theory (DLT)*. LNCS 2710, Springer-Verlag, Berlin Heidelberg, 2003, 279–291.

[5] F. Drewes, J. Högberg, Query learning of regular tree languages: how to avoid dead states. *Theory of Computing Systems* (2006). To appear.

[6] K. Fu, *Syntactic Methods in Pattern Recognition*. Academic Press, New York, 1974.

[7] F. Gécseg, M. Steinby, *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[8] E. M. Gold, Language identification in the limit. *Information and Control* **10** (1967), 447–474.

[9] J. J. Horning, A study of grammatical inference. Techn. Rep. 139, Univ. of Stanford, Dept. of Computer Science, 1969.

[10] L. Lee, Learning of context-free languages: a survey of the literature. Techn. Rep. TR-12-96, Harvard University, 1996.

[11] L. S. Levy, A. K. Joshi, Skeletal structural descriptions. *Information and Control* **39** (1978), 192–211.

[12] C. Martín-Vide, V. Mitrana, G. Păun (eds.), *Formal Languages and Applications*. Studies in Fuzzyness and Soft Computing 148, Springer, Berlin, 2004.

[13] G. Rozenberg, A. Salomaa (eds.), *Handbook of Formal Languages, vol. 3: Beyond Words*. Springer-Verlag New York, Inc., New York, 1997.

[14] Y. Sakakibara, Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science* **76** (1990), 223–242.