A Learning Algorithm for a Subclass of Tree Rewriting Systems

M. Jayasrirani[1], D.G. Thomas[2],
Atulya K. Nagar[3] and T. Robinson[2]
[1]Arignar Anna Government Arts College, Walajapet, India
[2]Madras Christian College, Chennai - 600 059, India
[2]dgthomasmcc@yahoo.com
[3]Department of Computer Science, Liverpool Hope University
United Kingdom

Abstract

Tree replacement / rewriting systems are an interesting model of computation. They are used in theorem proving, algebraic simplification and language theory. A fundamental property of tree replacement systems is the Church-Rosser property which expresses the fact that interconvertability of two trees can be checked by mere simplification to a common tree. In this paper, we give a learning algorithm for a subclass of the class of Church-Rosser tree replacement systems.

Keywords: Tree manipulating system, Church-Rosser tree rewriting systems, query learning.

1 Introduction

Tree rewriting systems are sets of tree rewriting rules used to compute by repeatedly replacing equal trees in a given formula until the simplest possible form (normal form) is obtained. The Church-Rosser property is certainly one of the most fundamental properties of tree rewriting system. In this system the simplest form of a given tree is unique since the final result does not depend on the order in which the rewritings rules are applied. The Church-Rosser system can offer both flexible computing and effecting reasoning with equations and have been intensively researched and widely applied to automated theorem proving and program verification etc. [10, 15].

On the other hand, grammatical inference is concerned with finding some grammatical description of a language when given only examples of strings from this languages, with some additional information about the structure of the strings, some counter-examples or the possibility of interrogating an oracle. The grammatical inference model of Gold [11] called "identification in the limit from positive data" has inputs of the inference process as just examples of the target language and there is no interaction with the environment. Angluin introduced the grammatical inference based on positive examples and membership questions of computed elements [2]. In [3], Angluin presented another grammatical inference model to identify regular languages based on membership and equivalence queries with a help of a teacher (called minimally adequate teacher).

Sakakibara studied the grammatical inference of languages of unlabeled derivation trees of context-free grammars with the help of structural equivalence queries and structural membership queries [16]. Inference of regular tree languages from positive examples only has been studied in [20]. The method of deducting a finite tree automaton from a finite sample of trees is an adaption of the well-known k-tail method of Biermann and Feldman [5].

An inference algorithm for learning a k-testable tree language was presented in [7]. Besombes and Marion [4] investigated regular tree language exact learning from positive examples and membership queries. Fernau [8] studied the problem of learning regular tree languages from text using the generalised frame work of function distinguisablity. A theoretical approach for the problem of learning multiplicity tree automaton has been done in [1].

In this paper, we investigate Church-Rosser tree rewriting systems as an alternative to describe and manipulate context-sensitive tree languages. Church-Rosser tree rewriting systems have many interesting properties such as decidability of word problem, language description of congruence classes, etc. We present an algorithm for learning a subclass of the class of Church-Rosser tree rewriting systems.

Learning is obtained using membership queries. A teacher or an oracle possesses the knowledge of the Church-Rosser tree rewriting system and hence the knowledge of the congruence classes and answers the membership queries related to the congruence classes made by the learner.

2 Preliminaries

We recall the notions of ranked alphabets, trees, tree replacement, tree composition and substitutions from [10].

Definition 2.1. *A ranked alphabet is a set Σ together with a rank function*

$r : \Sigma \rightarrow N$ where N denotes the set of nonnegative integers. Every symbol f in Σ of rank $r(f) = n$ is said to have arity n. Symbols of arity zero are also called constants.

Definition 2.2. *A tree domain D is a nonempty subset of strings over N_+, the set of positive integers, satisfying the following conditions:*

1. *For each u in D, every prefix v of u is also in D.*
2. *For each u in D, for every positive integer i, if ui is in D then for every j, $1 \leq j \leq i$, uj is also in D.*

Definition 2.3. *A Σ-tree is a function $t : D \rightarrow \Sigma$ such that*

1. *D is a tree domain*
2. *For every u in D, if $n = card(\{i \in N_+ | ui \in D\})$, then $n = r(t(u))$ the arity of the symbol labeling u in t.*

Given a tree t, its domain is denoted as $dom(t)$. The elements of the domain are called nodes or tree addresses. A node u is a leaf if $card(\{i \in N_+ | ui \in D\}) = 0$. The node corresponding to the empty tree is denoted as Λ.

Definition 2.4. *Given a tree t and a tree address u in $dom(t)$, the subtree of t at u, denoted as t/u, is the tree whose domain is the set $dom(t/u) = \{v \in N_+^* | uv \in dom(t)\}$ and such that $t/u(v) = t(uv)$ for every v in $dom(t/u)$.*

A tree is finite if its domain is finite. The set of finite Σ-trees is denoted as T_Σ.

Let $X = \{x_1, x_2, \ldots\}$ be a countable set of variables, and let $X_n = \{x_1, x_2, \ldots, x_n\}$ (with $X_0 = \phi$). Adjoining the set X to the constants in S (each variable is of arity zero), we get the set of trees with variables as $T_\Sigma(X)$. Similarly adjoining X_n, we obtain $T_\Sigma(X_n)$. The set $(T_\Sigma(X_m))^n$ of n tuples of trees with variables from X_m is denoted as $T_\Sigma(m, n)$ and $T_\Sigma(X_m) = T_\Sigma(m, 1)$.

Given $t = (t_1, \ldots, t_n)$ in $T_\Sigma(m, n)$ and t' in $T_\Sigma(n, 1)$, their composition or catenation is the tree denoted by
$t' \cdot t = t'(t_1, \ldots, t_n)$ and defined by the sets of pairs:
$\{(v, t'(v))/v \in dom(t'), t'(v) \notin X_n\}$,
$\{(uv, t_i(v))/u \in dom(t'), v \in dom(t_i), t'(u) = x_i, 1 \leq i \leq n\}$
If $t = (t_1, \ldots, t_n)$ and $t' = (t'_1, \ldots, t'_p) \in T_\Sigma(n, p)$,
then $t' \cdot t = (t'_1(t_1, \ldots, t_n), \ldots, t'_p(t_1, \ldots, t_n))$.

Definition 2.5. *Given a tree t_1, an address u in $dom(t_1)$ and another tree t_2, the tree obtained by replacement of t_2 for u in t_1 is the tree denoted as*

$t_1[u \leftarrow t_2]$ *defined by the sets of pairs:*
$\{(v, t_1(v))/v \in dom(t_1), u$ *is not a prefix of* $v\}$
$\{(uv, t_2(v))/v \in dom(t_2)\}$

Given a tree, an independent set of addresses $\{u_1, \ldots, u_n\}$ in $dom(t)$, and n trees $t_1, \ldots, t_n$, the tree $t[u_1 \leftarrow t_1, \ldots, u_n \leftarrow t_n]$ is obtained by simultaneous replacement of t_i at u_i.

The height $|t|$ of a finite tree t is defined as $hg(t) = 0$, $root(t) = t$ for $t \in X \cup \Sigma_0$;
$hg(t) = 1 + max\{hg(t_i)/1 \leq i \leq m, t = (t_1, t_2, \ldots, t_m)\}$.

Given a finite tree 't', the set of variables $var(t)$ occurring in t is the finite set

$$Var(t) = \{x_i \in X / \exists u \in dom(t), t(u) = x_i\}.$$

A substitution is any function $h : X \rightarrow T_\Sigma(X)$. Since $T_\Sigma(X)$ is the free algebra over the set X, every substitution extends uniquely to a unique homomorphism:
$\bar{h} : T_\Sigma(X) \rightarrow T_\Sigma(X)$, that is, to a function $\bar{h}$ such that
$\bar{h}(x) = h(x)$ for every x in X
$\bar{h}(f(t_1, \ldots, t_n)) = f(\bar{h}(t_1), \ldots, \bar{h}(t_n))$ if the rank of f, $r(f) \geq 1$ and
$\bar{h}(a) = h(a)$ for a constant 'a'.

3 Tree Replacement Systems

In this section, we recall the necessary definitions and notations related to tree rewriting system [10].

Definition 3.1. *A set of rules S over Σ is a subset of $T_\Sigma(X) \times T_\Sigma(X)$. Each pair (s, t) in S is called a rule and is also denoted as $s \rightarrow t$.*

The congruence generated by S is the reflexive transitive closure $\Leftrightarrow_S^*$ of the relation $\Leftrightarrow_S$ defined as follows: For any two trees t_1 and t_2 in $T_\Sigma(X)$, if there is some tree T in $T_\Sigma(X)$, some tree address u both in $dom(t_1)$ and $dom(t_2)$, some pair (s, t) such that either $s \rightarrow t$ or $t \rightarrow s$ is a rule in S, some substitution $h : Var(s) \cup Var(t) \rightarrow T_\Sigma(X)$ and $t_1 = T[u \leftarrow \bar{h}(s)]$, $t_2 = T[u \leftarrow \bar{h}(t)]$, then we write $t_1 \Leftrightarrow t_2$.

In other words, t_2 is obtained by making a subtree of $t_1(\bar{h}(s))$ which is a substitution instance of one side of a rule in $S(s \rightarrow t, t \rightarrow s)$ and replacing it.

Definition 3.2. *Two trees t_1 and t_2 are congruent (mod S) if $t_1 \Leftrightarrow^* t_2$. The class of trees congruent to the tree 't' is $[t]_S = \{t'/t' \Leftrightarrow_S^* t\}$.*

The set of congruence classes $\{[t]_S / t \in T_\Sigma\}$ forms a monoid under multiplication, $[t]_S \cdot [r]_S = [tr]_S$ with identity $[\Lambda]_S$. This monoid is the quotient monoid $T_\Sigma / \Leftrightarrow_S^*$ denoted by M_S.

Definition 3.3. *Given a set of rules S over Σ, the relation $\Rightarrow_S$ is defined as $t \Rightarrow_S s$, if $t \Leftrightarrow s$ and $hg(t) > hg(s)$, $\forall\, t, s \in T_\Sigma(X)$. $\Rightarrow_S^*$ is the reflexive transitive closure of $\Rightarrow_S$ and $(S, \Rightarrow_S)$ is called a tree replacement (rewriting) system on Σ.*

Given a tree replacement system $(S, \Rightarrow_S)$, a tree t is irreducible (*mod S*) if there is no tree t' such that $t \Rightarrow_S t'$. Let $IRR(S)$ be the set of all irreducible trees (mod S).

Definition 3.4. *A tree replacement system $(S, \Rightarrow_S)$ is Church-Rosser if for all trees t_1, t_2 with $t_1 \Leftrightarrow_S^* t_2$, there exists a tree t_3 such that $t_1 \Rightarrow_S^* t_3$ and $t_2 \Rightarrow_S^* t_3$.*

The word problem for a tree replacement system $(S, \Rightarrow_S)$ is that given any two trees s, t in $T_\Sigma(X)$, deciding whether s and t are congruent to each other or not.

The word problem is undecidable in general for any tree replacement system [10] but it has been proved that the word problem for any Church-Rosser tree replacement system is decidable.

Example 3.1. *Let the trees q, s, t, s', t', t_1 and t_2 be in $T_\Sigma(X)$ where $\Sigma = \{a, b, c, d, x, y\}$ and $X = \{x, y\}$. Let (s, t) or (t, s) be a rule in S. Let $q = a(b(a(c,d),c), a(b(d,c),d))$ be a tree in $T_\Sigma(X)$ as shown in the Figure 1.*

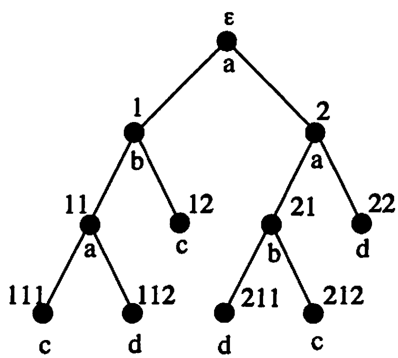

Figure 1: Tree q

Let $s = a(x, y)$ and $t = a(c, b(y, x))$ be two trees as shown in the Figure 2.

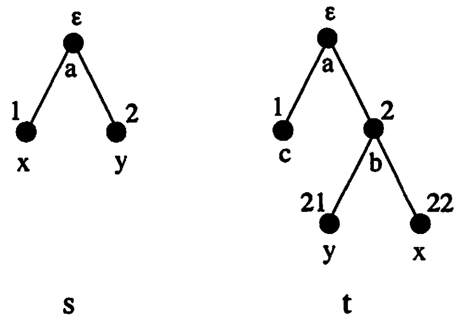

Figure 2: Trees s and t

Let tree $s' = \bar{h}(s) = a(c,d)$ be the substitution instance of s and tree $t' = \bar{h}(t) = a(c,b(d,c))$, the substitution instance of t as shown in the Figure 3.

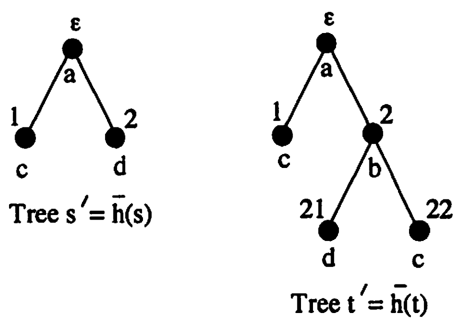

Figure 3: Trees s' and t'

Tree t_1 is obtained from tree q by replacing the subtree at node 2 of the tree q by the tree s'. Similarly tree t_2 is obtained by replacing the subtree at the node 2 of the tree q by the tree t'. These two trees t_1 and t_2 are represented in Figure 4.

It can also be seen that t_2 is obtained if the subtree $\bar{h}(s)$ of t_1 at node 2 is replaced by $\bar{h}(t)$. Similarly t_1 is obtained if the subtree $\bar{h}(t)$ of t_2 at node 2 is replaced by $\bar{h}(s)$.

Hence $t_1 \Leftrightarrow_S t_2$.

Let S be a tree rewriting system on Σ.

Let $RED(S)$ be the set of all reducible trees with respect to S. That is

$$RED(S) = T_\Sigma(X) - IRR(S).$$

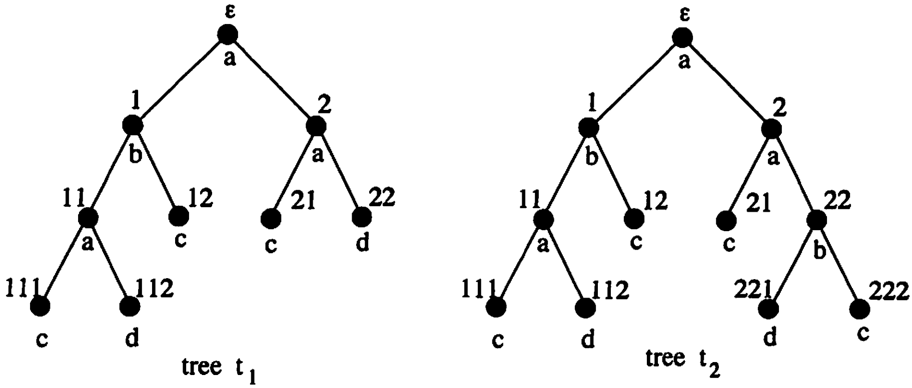

Figure 4: Trees t_1 and t_2

Church-Rosser tree rewriting systems T have many interesting properties in connection with decision problems and formal language theory.

Definition 3.5. *A tree rewriting system T on Σ is called reduced if for every rewriting rule $(s,t) \in T$, t is an irreducible tree with respect to T and s is an irreducible tree with respect to $T - \{(s,t)\}$.*

Definition 3.6. *Let T be a tree rewriting system on Σ. For a tree $t \in T_\Sigma(X)$, s is called a normal form of t, if $s \in [t]_T$ and s is an irreducible tree with respect to T.*

We now explain that given a tree replacement system R there is an algorithm to reduce every tree to an irreducible tree. This can be used to show that there is an algorithm for deciding whether two trees are congruent modulo a finite Church-Rosser tree replacement system S. So it is necessary to generalize the notion of a left most reduction to trees. For trees, there are two ways of reduction namely top most reduction and bottom most reduction. For our purpose, the notion of bottom most reduction seems to be adequate.

Let R be a tree replacement system such that for every tree s, there is atmost one tree t with $s \to t$ is a rule in R.

It is assumed that there is a total ordering on each subset of rules $\{s' \to t' \in R |$ for some $s \to t$ in R, $hg(s') = hg(s)\}$.

Given a tree t_0, let us imagine that we scan t_0 from bottom up, trying to match a subtree r of t_0 of height one with a substitution instance of the left hand side of a reduction $s \to t$ in R. Then if there are at least two rules $s \to t$ and $s' \to t'$ such that t_0 can be written as $t_0 = s_0[u \leftarrow r]$ where $r = s(t_1, \ldots, t_m) = s'(t'_1, \ldots, t'_n)$. We will use the greatest rule $s \to t$ in the total ordering of all rules such that $r = s(t_1, \ldots, t_m)$ for which s has largest height. After having reduced r to $r' = t(t_1, \ldots, t_m)$, we repeat this process with $s_0[u \leftarrow r']$. This process forces the rules to be applied in a certain order called bottom up normal order. Thus we obtain an irreducible tree from t_0.

Lemma 3.1. *For any tree rewriting system T that is Church-Rosser, there is a unique reduced tree rewriting system T' that is Church-Rosser and equivalent to T. Furthermore, one can effectively construct T' from T.*

Lemma 3.2. *Let T and T' be two equivalent tree rewriting systems. If T is Church-Rosser and $IRR(T) = IRR(T')$, then T' is also Church-Rosser.*

4 Procedure for Learning Church-Rosser Tree Rewriting System R

Let Σ be a given ranked alphabet. We consider Church-Rosser tree rewriting system T on Σ. Let $M_T = \{L_1, L_2, \ldots, L_n\}$ be the quotient monoid where each L_i is a congruence class of a tree with respect to T. Then, the congruence relation $\Leftrightarrow_T^*$ is of finite index and so each congruence class L_i $(1 \leq i \leq n)$ is a regular tree language [6]. Algebraic properties of a Church-Rosser tree rewriting system T for which M_T is finite enable us to present an efficient learning procedure for congruence classes with only membership queries. Since the congruence of T partitions the set $T_\Sigma(X)$ into disjoint congruence classes, any tree in $T_\Sigma(X)$ is in only one congruence class with respect to T. So, the membership query for congruence classes is meaningful and reasonable.

The unique reduced Church-Rosser tree rewriting system R equivalent to T is then obtained. The learning procedure to obtain R consists of two parts, one for $IRR(R)$ and the other for the tree rewriting system R.

For any tree $t \in T_\Sigma(X)$ given as input, the oracle answers membership query by producing an n-tuple that contains $n-1$ zeros and one 1 since $M_T = M_R = \{L_1, L_2, \ldots, L_n\}$. The learner gets the value of n when the empty tree Λ is given as input for membership query. The input is a tree $t \in T_\Sigma$ and the output is an n tuple $q(t) = (k_1, k_2, \ldots, k_n)$ where $k_i = 1$ if $t \in L_i$ and $t_i = 0$ if $t \notin L_i$ $(1 \leq i \leq n)$. Let p_i be the projection defined by $p_i(x) = x_i$ for any n-tuple $x = (x_1, x_2, \ldots, x_n)$, $1 \leq i \leq n$.

Learning Procedure

Learning is obtained using membership queries. A teacher or an oracle possesses the knowledge of the Church-Rosser tree rewriting system and hence the knowledge of the congruence classes and answers the membership queries related to the congruence classes made by the learner.

Membership queries are made to the oracle for the input trees, starting with the empty tree Λ, which is an irreducible tree with respect to R and continued with the trees in T_Σ^0. Let $t_1 = \Lambda$ and suppose $t_2, t_3, \ldots, t_s$ are the lexicographically ordered trees in T_Σ^0 where $s-1$ is the number of constants in Σ.

A tree t_i $(2 \leq i \leq s)$ belonging to L_j for some j $(1 \leq j \leq n)$ is an irreducible tree with respect to R whenever $t_i \in L_j$ but $t_p \notin L_j$ for $p = 1, 2, \ldots, i-1$. Hence by membership queries all the irreducible trees in T_Σ^0 with respect to R are obtained.

The process is continued by making membership queries for trees in T_Σ^1 $(T_\Sigma^0 \cap IRR(R))$, the set of all trees of height one with subtrees in $T_\Sigma^0 \cap IRR(R)$, which can be lexicographically ordered.

Thus the process gives irreducible trees with respect to R in T_Σ^0 and T_Σ^1. In general the process is continued recursively by making membership queries for trees in T_Σ^1 $(T_\Sigma^{r-1} \cap IRR(R))$, the set of all trees of height r, with subtrees in $T_\Sigma^{r-1} \cap IRR(R)$, $r \geq 1$. This process terminates when each L_j receives an irreducible tree with respect to R.

The algorithm for forming irreducible trees with respect to R, terminates when the process for finding trees with respect to R in T_Σ^k ends, when $k = max\{hg(t)/t \in IRR(R)\}$ since (a) $IRR(R)$ is finite (b) each L_j $(1 \leq j \leq n)$ contains exactly one irreducible tree with respect to R and (c) irreducible trees with respect to R are shortest trees in their respective classes $L_1, L_2, \ldots, L_n$.

To identify the unique, reduced Church-Rosser tree rewriting system R equivalent to the unknown tree rewriting system T, the learner performs again the membership queries as in the procedure for the lexicographically ordered trees in the set $T_\Sigma^1(IRR(R)) - IRR(R)$, where $T_\Sigma^1(IRR(R))$ is the set of all trees with subtrees in $IRR(R)$ in the next level.

The learning algorithm then forms the tree rewriting system

$$S = \Big\{(s,t)/s \in T_\Sigma^1(IRR(T)) - IRR(T), t \in IRR(T), s \text{ and } t \text{ both belong to } L_j \text{ for some } j(1 \leq j \leq n) \text{ on } \Sigma\Big\}$$

From S a reduced tree rewriting system S' equivalent to S on Σ is obtained and thus the learner obtains R which is same as S' on Σ.

An example run

We illustrate the procedure for learning the reduced Church-Rosser tree rewriting system
$R = \{(b(c), c), (b(d), d), (a(c, c), c), (a(d, d), d), (a(c, d), c), (a(d, c), d)\}$ on $\Sigma = \{a, b, c, d\}$ with arities of a, b, c, d as 2, 1, 0, 0 respectively. That is $\rho(a) = 2$, $\rho(b) = 1$, $\rho(c) = \rho(d) = 0$.

$M_R = \{[\Lambda]_R, [c]_R, [d]_R\}$ where $L_1 = [\Lambda]_R$, $L_2 = [c]_R$ and $L_3 = [d]_R$.

Membership queries are made for the trees $\wedge, c, d$ belonging to T^0_Σ and the oracle produces the answers $q(\Lambda) = (1, 0, 0)$, $q(c) = (0, 1, 0)$, $q(d) = (0, 0, 1)$ for which the learner obtains $IRR(R)$ as $\{\wedge, c, d\}$. Again membership queries are made for the trees in the set $T^1_\Sigma = \{b(c), b(d), a(c, c), a(d, d), a(c, d), a(d, c)\}$ and the oracle produces the answers:

$q(b(c)) = (0, 1, 0)$, $q(b(d)) = (0, 0, 1)$
$q(a(c, c)) = (0, 1, 0)$, $q(a(d, d)) = (0, 0, 1)$
$q(a(c, d)) = (0, 0, 1)$, $q(a(d, c)) = (0, 0, 1)$

From which the learner obtains
$S = \{(b(c), c), (b(d), d), (a(c, c), c), (a(d, d), d), (a(c, d), c), (a(d, c), d)\}$. The reduced tree rewriting system S' equivalent to S is then obtained as $S' = S = R$.

Learning Algorithms

Algorithm for Learning $IRR(R)$

begin
  $IRR(R) = \phi$
  Input the empty tree $t_1 = \Lambda$
  n = number of entries in $q(\Lambda)$
  $L_1 = \{\Lambda\}$
  $IRR(R) = \{\Lambda\}$
  $N_1 = 1$
  For $j = 2$ to n,
  Initialize: $L_j = \phi$; $N_j = 0$
  Input trees t_i $(i = 2, 3, \dots)$ ordered according to height (trees of same height are lexicographically ordered) such that
  $t_i \in T^0_\Sigma \cup T^1_\Sigma(T^{r-1}_\Sigma \cap IRR(R))$, $(r \geq 1)$
  while $N_j = 0$ for some j do
  begin
    For $j = 1$ to n do
    begin
      If $p_j(q(t_i)) = 1$ do
      begin
        $L_j = L_j \cup \{t_i\}$

```
                If N_j = 0 do
                begin
                    N_j = 1
                    IRR(R) = IRR(R) ∪ {t_i}
                end
            end
        end
    end
    output IRR(R)
end.
```

Algorithm for Learning R

```
begin
    Input trees t_i (i = 1, 2, 3, ...) ordered according to height (trees of
        same height are lexicographically ordered) such that
    t_i ∈ T_Σ^1(IRR(T)) − IRR(T)
    Initialize: S = φ
    For s ∈ T_Σ^1(IRR(T)) − IRR(T) do
    begin
        For t ∈ IRR(R) do
        begin
            If p_j(q(s)) = p_j(q(t)) = 1 for some j (1 ≤ j ≤ n), then
            S = S ∪ {(s, t)}
        end
    end
    Initialize: S' = S
    begin
        For (s, t) ∈ S', do
        begin
            If (s_1, t_1) ∈ S' − {(s, t)} such that s has s_1 as a subtree, then
            S' = S' − {(s, t)}
        end
    end
    output: R = S'
end.
```

Correctness of the Learning Algorithm

We establish the correctness of the learning algorithm by showing that S and R are equivalent and S is Church-Rosser. Also, if S' is a reduced tree rewriting system equivalent to S, then $S' = R$.

Lemma 4.1.
$IRR(S) = IRR(R)$.

Proof.

$$\begin{aligned} IRR(S) &= T_\Sigma(X) - RED(S) \\ &= T_\Sigma(X) - (T_\Sigma(X) - IRR(R)) \\ &= IRR(R) \end{aligned}$$

□

Lemma 4.2.
$t_1 \overset{*}{\Leftrightarrow}_S t_2$ implies $t_1 \overset{*}{\Leftrightarrow}_R t_2$ for $t_1, t_2 \in T_\Sigma(X)$.

Proof.
It is enough to prove that $t_1 \Leftrightarrow_S t_2$ implies $t_1 \overset{*}{\Leftrightarrow}_R t_2$ for $t_1, t_2 \in T_\Sigma(X)$.

Suppose $t_1 \Leftrightarrow_S t_2$ holds. Then $t_1 = t'st''$ and $t_2 = t'tt''$ where either $(s,t) \in S$ or $(t,s) \in S$ and $t', t'' \in T_\Sigma(X)$. By the definition of S, s and t both belong to L_i for some i $(1 \leq i \leq n)$. That is $s \overset{*}{\Leftrightarrow}_R t \Rightarrow t_1 \overset{*}{\Leftrightarrow}_R t_2$. □

Lemma 4.3.
Cardinality of M_S is n. That is $card(M_S) = n$.

Proof.
$IRR(S) = \{t_1, t_2, \ldots, t_n\}$. Here no two t_i's are congruent with respect to S. The reason is as follows. Suppose $t_i \overset{*}{\Leftrightarrow}_S t_j$ for some i and j, $1 \leq i \leq n$, $1 \leq j \leq n$, $i \neq j$. Then by Lemma 4.2, $t_i \overset{*}{\Leftrightarrow}_R t_j$. This is not possible since R is Church-Rosser and t_i and t_j are irreducible trees with respect to R. Thus, every congruence class with respect to S has exactly one t_i. Since $card(IRR(S)) = n$, we have $card(M_S) = n$. □

Lemma 4.4.
For $s, t \in T_\Sigma(X)$, $s \overset{*}{\Leftrightarrow}_R t$ implies $s \overset{*}{\Leftrightarrow}_S t$.

Proof.
Suppose for $s, t \in T_\Sigma(X)$, $s \overset{*}{\Leftrightarrow}_R t$ holds. Since R is Church-Rosser, there exists exactly one $t_i \in IRR(R)$ such that $s \overset{*}{\rightarrow}_R t_i$ and $t \overset{*}{\rightarrow}_R t_i$. That is $s \overset{*}{\Leftrightarrow}_R t_i$ and $t_i \overset{*}{\Leftrightarrow}_R t$. Since every congruence class with respect to S contains exactly one irreducible tree with respect to S and $IRR(S) = IRR(R)$, let $t_j \in [s]_S$. This implies that $t_j \overset{*}{\Leftrightarrow}_S s$ and hence $t_j \overset{*}{\Leftrightarrow}_R s$ which means $t_j \overset{*}{\Leftrightarrow}_R t_i$. This is impossible since R is Church-Rosser. Hence $j = i$. That is $t_i \in [s]_S$. Similarly we can show that $t_i \in [t]_S$. Thus $t_i \overset{*}{\Leftrightarrow}_S s$ and $t_i \overset{*}{\Leftrightarrow}_S t$ together imply that $s \overset{*}{\Leftrightarrow}_S t$. □

Theorem 4.1.
R and S are equivalent.

Proof.
The equivalence of R and S follows from Lemma 4.2 and Lemma 4.4. □

Theorem 4.2.
S is Church-Rosser.

Proof.
The Church-Rosserness of S follows from Lemma 3.2, Lemma 4.1 and Theorem 4.1. □

Theorem 4.3.
$S' = R$ where S' is a reduced tree rewriting system equivalent to S.

Proof.
By Lemma 3.1 and Theorem 4.1, S' is unique and Church-Rosser. Since S' is equivalent to S, which in turn is equivalent to R and R is reduced. By applying Lemma 3.1 once again, we obtain $S' = R$. □

Time Analysis

Here, the number of trees to be processed through membership query for learning $IRR(R)$ can be found.

We can show that the time taken by the learning algorithm to learn $IRR(R)$ is polynomial in the number of congruence classes, the arities of members of Σ and the number of elements in Σ. We assume that the oracle requires a single unit of time to answer each membership query.

We find that the number of trees to be processed through membership query for learning $IRR(R)$ is less than or equal to $1 + m(n + 2)$ where $m = card\, T^1_\Sigma$ which is fixed and n is the total number of congruence classes with respect to R. The trees to be processed are in the set $F = \{\Lambda\} \cup T^0_\Sigma \cup T^1_\Sigma(T^0_\Sigma \cap IRR(R)) \cup T^1_\Sigma(T^1_\Sigma \cap IRR(R)) \cup T^1_\Sigma(T^2_\Sigma \cap IRR(R)) \cup \cdots \cup T^1_\Sigma(T^{r-1}_\Sigma \cap IRR(R))$ where $r = max\{hg(t) | t \in IRR(R)\}$ and if $k = card\, T^0_\Sigma$ then

$$\begin{aligned}
card\, F &= 1 + k + m + ms_1 + \cdots + ms_{r-1} \text{ where } s_j = card\, T^1_\Sigma(T^j_\Sigma \cap IRR(R)) \\
&= 1 + k + m + m(s_1 + \cdots + s_{r-1}) \\
&\leq 1 + m + m + m(s_1 + \cdots + s_{r-1}) \quad (\text{where } k \leq m) \\
&= 1 + 2m + m(n - s_0 - s_r) \\
&\leq 1 + 2m + mn \\
&= 1 + m(n + 2).
\end{aligned}$$

Conclusion

In this paper we propose a new method to learn a class of tree rewriting systems which yield many decidable properties in the area of tree rewriting systems. It is worth examining to find a learning algorithm to learn the whole class of all Church-Rosser tree rewriting systems.

References

[1] Amaury Habrard and Jose Oncina, Learning multiplicity tree automata, *ICGI 2006*, (2006), 268–280.

[2] D. Angluin, On the complexity of minimum inference of regular sets, *Information and Control*, 39 (1978) 337–350.

[3] D. Angluin, Learning regular sets from queries and counter examples, *Inform. Comput.*, 75 (1987), 87–106.

[4] J. Besombes and J.Y. Marion, Learning tree languages from positive examples and membership queries, *Theoretical Computer Science*, 382 (2007), 183–197.

[5] A.W. Biermann and J.A. Feldman, On the synthesis of finite state machines from samples of behavior, *IEEE Trans. Comput.* C-21 (1972), 592–597.

[6] H. Comon, M. Dauchet, R. Gulleron, F. Jacquemard, D. Lugiez, S. Tison and M. Tomasi, Tree automata techniques and applications, Available on : http://www.grappa.univ-lille3.fr/tata, (2002).

[7] Damian Lopez, Jose M. Sempere and Pedro Garcia, Inference of reversible tree languages, *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 34(4) (2004), 1658–1665.

[8] H. Fernau, Learning tree language from text, *ITA*, 4 (2007), 351–374.

[9] Z. Fulop, J. Attila and S. Vagvolgyi, Minimal equational representations of recognizable tree languages, *Acta Infomatica*, 34 (1997), 59–84.

[10] J.H. Gallier and R.V. Book, Reductions in tree replacement systems, *Theoretical Computer Science*, 37 (1985), 123–150.

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

[12] G. Heut, Confluent reductions: abstract properties and applications to term rewriting systems, *Journal of the Association for Computing Machinery*, 27(4) (1980), 797–821.

[13] M.R.K. Krishna Rao, Some classes of term rewriting systems inferable from positive data, *Theoretical Computer Science*, 397 (2008), 129–149.

[14] R. McNaughton, P. Narendran and F. Otto, Church-Rosser Thue systems and formal languages, *Journal of the Association for Computing Machinery*, 35(2) (1988), 324–244.

[15] B.K. Rosen, Tree-manipulating systems and Church-Rosser theorems, *Journal of the Association for Computing Machinery*, 20(1) (1973), 160–187.

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

[17] K. Salomaa, Deterministic tree pushdown automata and monadic tree rewriting systems, *Journal of Computer and Systems Sciences*, 37 (1988), 367–394.

[18] D.G. Thomas, S.C. Samuel, P.J. Abisha and K.G. Subramanian, Tree replacement and public key cryptosystem, *Lecture Notes in Computer Science*, 2551 (2002), 71–78.

[19] D.G. Thomas, R. Siromoney, K.G. Subramanian and V.R. Dare, Thue systems and DNA - A learning algorithm for a subclass, *Lecture Notes in Artificial Intelligence*, 744 (1993), 314–327.

[20] Timo Knuutila and Magnes Steinby, The inference of tree languages from finite samples - an algebraic approach, *Theoretical Computer Science*, 129 (1994), 337–367, Elsevier.