Chapter 7

Learning Regular Tree Languages

7.1 Introduction

As noted in eg Fernau (2002, 2000), tree language induction is an important subject in the fields of (applied) Formal Learning Theory and Grammar Induction. We have already discussed the motivation from linguistics; one expects a learner to not just act as a characteristic function for a set of strings, but also to assign the right derivations (and thus meaning) to them.\footnote{There are other uses as well, Bernard and de la Higuera (1999) proposes it as a tool in Inductive Logic Programming, and there is also motivation from bioinformatics.}

The work by Buszkowski and Penn and Kanazawa previously discussed has taken this approach, but without explicitly using a tree formalism. In this chapter the benefits of using the tree automata formalism (a generalization of finite state automata) will be shown by discussing results from Angluin (1982), Sakakibara (1992) and Besombes and Marion (2001, 2002a,b). The latter offers the following results:

1. Reset-free context-free grammars are identifiable from parse tree presentations. See Theorem 7.25.

2. Reversible dependency tree grammars are identifiable from positive examples. See Theorem 7.29.

3. Rigid Classical categorial grammars are identifiable from unlabeled derivation trees. See Theorem 7.30 This was already established by Kanazawa in Kanazawa (1998), but the proof is much shorter and conceptually simpler.
Related work: There are several papers on tree grammar identification, Angluin’s and Sakakibara’s results have already been mentioned. Other papers (Gonzalez et al. (1976); Levine (1981); Kamata (1984); Fukuda and Kamata (1984)) are based on the idea of k-tail inference for regular word languages from Biermann and Feldman (1972) and has been expounded in Knuttila and Steinby (1994). More recently, Fernau (2001) has applied this approach to XML grammars. In Carrasco et al. (1998) it was shown that regular tree languages are identifiable with probabilistic samples. The reader interested in context-free language inference may consult the surveys [Lee, Sakakibara1997, Mak97].

7.2 Regular Tree Languages

7.2.1 Trees are Terms

Background on regular tree languages can be found in the survey book Comon et al. (1997). A ranked alphabet is a tuple \((F, \text{arity})\) where \(F\) is a finite set of symbols and \(\text{arity}\) is a function from \(F\) to \(\mathbb{N}\), which indicates the arity of a symbol. Given a set \(X\) of variables, terms are inductively defined: a symbol of arity 0 is a term, a variable of \(X\) is a term, and if \(f\) is a symbol of arity \(n\) and \(t_1, \ldots, t_n\) are terms, then \(f(t_1, \ldots, t_n)\) is a term. The set of all terms is denoted by \(T(F, X)\), the set of ground terms is denoted by \(T(F)\). Throughout, labeled ordered trees are represented by terms.

A context is a term \(C[\Diamond]\) containing a special variable \(\Diamond\) which occurs just once in that term, it marks an empty place. Throughout, the substitution of \(\Diamond\) by a term \(u\) is written \(C[u]\), its states that \(u\) is an occurrence of the term \(C[u]\).

7.2.2 Tree Automata

A (nondeterministic) finite tree automaton (NFTA) is a quadruplet \(A = (F, Q, Q_F, \rightarrow_A)\) such that \(Q\) is a finite set of states, \(Q_F \subseteq Q\) is the set of final states, and \(\rightarrow_A\) is the set of transitions. A state \(q\) of a deterministic tree automaton \(A\) is useful if and only if there exists a tree \(t\) and some node \(x \in \Delta_t\), such that \(\delta(t/x) = q\) and \(\delta(t) \in F\). A deterministic automaton containing only useful states is called stripped. A transition is a ground rewrite rule of the form \(f(q_1, \ldots, q_n) \rightarrow_A q\) where \(q\) and \(q_1, \ldots, q_n\) are states of \(Q\), and \(f\) is a symbol of arity \(n\), in the case that \(n = 0\) the transition is just of the form \(a \rightarrow_A q\).

A finite tree automaton is a deterministic finite tree automaton (DFTA) if it contains no rules sharing the same left hand side. In this case \(\rightarrow_A\) represents a mapping which is not necessarily defined for all entries, so this means that we shall consider incomplete automata. The single derivation relation \(\rightarrow_A\) is
defined so that \( t \rightarrow u \) if and only if there is a transition \( f(q_1, \ldots, q_n) \rightarrow q \) such that \( t = v[f(q_1, \ldots, q_n)] \) and \( u = v[q] \). Note that \( \rightarrow^* \subseteq T(\mathcal{F}, Q) \times T(\mathcal{F}, Q) \) where states of \( Q \) are 0-ary symbols. The derivation relation \( \rightarrow^* \) is the reflexive and transitive closure of \( \rightarrow \). The tree language recognized by \( \mathcal{A} \) is \( L_{\mathcal{A}} = \{ t \in T(\mathcal{F}) \mid t \rightarrow^* q_f \) and \( q_f \in \mathcal{Q}_F \} \).

7.2.3 Reversible Regular Tree Languages

Definition 7.1 A DFTA \( \mathcal{A} \) is reversible if and only if

1. There are no two rules with left hand sides that differ by just one symbol. That is, there are no transitions \( f(p_1, \ldots, p_n, q, p_{n+1}, \ldots, p_m) \rightarrow^* p \) and \( f(p_1, \ldots, p_n, q', p_{n+1}, \ldots, p_m) \rightarrow^* p \) where \( q \neq q' \).
2. \( \mathcal{A} \) has one final state.

A tree language is reversible if it is recognised by a reversible DFTA. Note that a symbol of arity \( n \) and \( n - 1 \) states determine at most one transition.

7.2.4 Reversible Regular Tree Grammars

A regular tree grammar (RTG) is a quadruplet \( \Gamma = (\mathcal{F}, X, \rightarrow^\Gamma, S) \) where \( S \in X \) is the start variable. Each production is of the form \( X \rightarrow t \) where \( X \) is a variable of \( X \) called the head, and \( t \) is a term of \( T(\mathcal{F}, X) \). Throughout no productions of the form \( X \rightarrow Y \), where \( Y \) is a variable of \( X \), are allowed. Define \( t \rightarrow^\Gamma u \) if and only if there is a production \( X \rightarrow^\Gamma Y \) such that \( u = t[X \leftarrow v] \). The derivation relation \( \rightarrow^\Gamma \) is the reflexive and transitive closure of \( \rightarrow^\Gamma \). The language produced by \( \Gamma \) is \( L_{\Gamma} = \{ t \in T(\mathcal{F}) \mid S \rightarrow^\Gamma t \} \).

Definition 7.2 A grammar \( \Gamma \) is reversible if and only if

1. there are no productions \( X \rightarrow^\Gamma C[Y] \) and \( X \rightarrow^\Gamma C[Z] \) starting with the same head \( X \) such that \( Y \neq Z \),
2. there are no productions \( X \rightarrow^\Gamma t \) and \( Y \rightarrow^\Gamma t \) with the same right hand-side such that \( X \neq Y \).

Definition 7.3 An RTG \( \Gamma \) is normal if each production is of the form \( X \rightarrow^\Gamma a \) or of the form \( X \rightarrow^\Gamma f(X_1, \ldots, X_n) \).

Theorem 7.4 A language \( L \) is reversible if and only if the language \( L \) is produced by a reversible and normal RTG.
7.2.5 Identification of Reversible Tree Languages

A positive presentation of a language $L$ is a sequence $t_1, \ldots, t_n, \ldots$ which enumerates all elements of $L$. Let $\Omega$ be a given class of automata (or grammars). An inference algorithm $A$ takes as input a finite segment $t_1, \ldots, t_n$ of a positive presentation of $L$ and guesses an automaton $A(t_1, \ldots, t_n)$. The inference algorithm $A$ converges to $L$ if there is a stage $N$ such that for all $n \geq N$, the language provided by $A(t_1, \ldots, t_n)$ is exactly $L$. A class of languages $\mathcal{L}$ is identifiable if and only if there is an inference algorithm $A$ such that for each positive presentation of a language $L$, $A$ converges to $L$.

Theorem 7.5 The class of reversible tree languages is identifiable.

We shall demonstrate the theorem above in the next section. The main notions are now set, the reader may skip the proofs and follow most discussions about applications in the last section of this chapter.

7.3 Identification of Reversible Tree Languages

7.3.1 An Algebraic Characterization of Reversible Tree Languages

An equivalence relation $\equiv$ is closed under context if for all terms $t$ and $u$ in $T(F)$, $t \equiv u$ if for every context $C[\cdot]$, $C[t] \equiv C[u]$. A congruence $\equiv$ on $T(F)$ is an equivalence relation which is closed under context.

Given a DFTA $A$, the equivalence relation $\equiv_A$ is defined as: $t \equiv_A u$ if $t \xrightarrow{A} q$ and $u \xrightarrow{A} q$. It is easy to see that $\equiv_A$ is closed under context, so it is a congruence.

Lemma 7.6 Let $A$ be a reversible DFTA. For every context $C[\cdot]$ and for every term $t$ and $u$ such that $C[t]$ and $C[u]$ are in $L_A$, $t \equiv_A u$.

Proof: By induction on the size of the context. Basis: Suppose that $C[\cdot] = \cdot$. By Lemma assumption, $t$ and $u$ are in $L_A$. Since $A$ is reversible, there is only one final state $q_f$, therefore we have $t \xrightarrow{A} q_f$ and $u \xrightarrow{A} q_f$. We conclude that $t \equiv_A u$.

Inductive step: Suppose that $C[\cdot] = C'[f(t_1, \ldots, t_n, \cdot, t_{n+1}, \ldots, t_m)]$. By the lemma assumption, $t_i \xrightarrow{A} q_i$ for some state $q_i$. Since $A$ is reversible the states $t_1, \ldots, t_m$ determine a unique transition whose left hand side is $C'[f(q_1, \ldots, q_n, q, q_{n+1}, \ldots, q_m)]$. This implies that $t \xrightarrow{A} q$ and $u \xrightarrow{A} q$. Therefore $t \equiv_A u$. \qed
7.3. IDENTIFICATION OF REVERSIBLE TREE LANGUAGES

Given a tree language $L$, the congruence $\equiv_L$ is defined as: $t \equiv_L u$ if for every context $C[\emptyset], C[t] \in L$ if and only if $C[u] \in L$. Following the Myhill-Nerode Theorem, the index of $\equiv_L$ is lower or equal than the index of $\equiv_A$ for any automaton $A$ which recognises $L$.

As a consequence, the minimal DFTA (up to a renaming of states) $A_L = \langle F, Q, Q_F, \rightarrow_L \rangle$ which recognises $L$ is defined as follows: let $[t]$ be the equivalence class of $t$ with respect to $\equiv_L$ and $\text{sub}(L)$ be the set of subterms of $L$. Let $Q = \{[t]| t \in \text{sub}(L)\}$ and $Q_F = \{[t]| t \in L\}$. For every state $[t_1], \ldots, [t_n]$ and for each $n$-ary symbol $f \in F$, there is a transition $f([t_1], \ldots, [t_n]) \rightarrow_L [f(t_1, \ldots, t_n)]$.

**Theorem 7.7** A tree language $L$ is reversible if and only if for every context $C[\emptyset]$ and for every term $t$ and every term $u$, if $C[t]$ and $C[u]$ are in $L$, then $t \equiv_L u$.

**Proof:** There is a reversible DFTA $A$ which recognises $L$. Following the Myhill-Nerode Theorem, $\equiv_A$ refines $\equiv_L$, that is if $t \equiv_A u$ then $t \equiv_L u$. Therefore, we conclude by Lemma 7.6.

Conversely, consider the minimal DFTA $A_L$ recognising $L$. This DFTA has only one final state because the lemma assumption says that each tree of $L$ belongs to the same equivalence class. Next, suppose that we have two transitions of the form $f(p_1, \ldots, p_n, q, p_{n+1}, \ldots, p_m) \rightarrow_L q''$ and $f(p_1, \ldots, p_n, q', p_{n+1}, \ldots, p_m) \rightarrow_L q'$. Since $A_L$ is minimal, there are terms $t_1, \ldots, t_m$ such that $t_i \rightarrow_L p_i$. This leads us to consider a context $C[\emptyset] = C'[f(t_1, \ldots, t_n, \emptyset, t_{n+1}, \ldots, t_m)]$ such that $C[q] \rightarrow_L q_f$ and $C[q'] \rightarrow_L q_f$ where $q_f$ is the final state of $A_L$. However, the lemma assumption implies that $q = q'$. We conclude that $A_L$ is reversible, and so $L$ is reversible.

This proof has the following consequence:

**Corollary 7.8** A tree language $L$ is reversible if and only if the minimal DFTA $A_L$ is reversible.

**Example 7.9** Let $L = \{f(g^n(a)) \mid n \geq 0\} \cup \{g(f^n(a)) \mid n \geq 0\}$. The tree language $L$ is recognised by the DFTA $A$ whose transitions are

$\begin{align*}
 a & \rightarrow_A A & f(A) & \rightarrow_A C \\
 g(A) & \rightarrow_A B & f(C) & \rightarrow_A S \\
 g(B) & \rightarrow_A G & f(F) & \rightarrow_A F \\
 g(G) & \rightarrow_A G & f(G) & \rightarrow_A S \\
 g(F) & \rightarrow_A S
\end{align*}$

The final states are $S, B$ and $C$. 
The DFTA $\mathcal{A}$ is minimal and is therefore isomorphic to $\mathcal{A}_L$. As a consequence of Corollary 7.8 the language $L$ is not a reversible tree language, although it is the union of two reversible tree languages.

### 7.3.2 Characteristic Samples

Let $L$ be a reversible language recognized by the minimal automaton $\mathcal{A}_L$. Each state $q$ has a term $rt(q)$ of $T(\mathcal{F})$ associated with it such that $rt(q) \xrightarrow{L} q$ and a minimal context $C_q[\emptyset]$, with respect to the size, such that $C_q[q] \xrightarrow{L} q_f$.

The set of characteristic samples $CS(L)$ is the smallest set which contains

1. the term $C_q[rt(q)]$ for each state $q$,

2. the term $C_q[f(rt(q_1), \ldots, rt(q_n))]$ for each transition $f(q_1, \ldots, q_n) \xrightarrow{L} q$.

It follows immediately that $CS(L) \subseteq L$. Also note that $C_{q_f}[\emptyset] = \emptyset$ where $q_f$ is a final state.

**Theorem 7.10** Let $L_1$ and $L_2$ be two reversible languages over $T(\mathcal{F})$, and assume $CS(L_1) \subseteq L_2$. Then $L_1 \subseteq L_2$.

**Proof:** Suppose that $\mathcal{A}_1 = (\mathcal{F}, Q_1, \{q_{f_1}\}, \rightarrow_1)$ and $\mathcal{A}_2 = (\mathcal{F}, Q_2, \{q_{f_2}\}, \rightarrow_2)$ are the minimal DFTAs which recognize $L_1$ and $L_2$ respectively.

For each state $q \in Q_1$, $C_q[rt(q)] \in CS(L_1) \subseteq L_2$. Thus $rt(q) \in sub(L_2)$. It follows that there is a unique function $\theta : Q_1 \rightarrow Q_2$ such that $rt(q) \xrightarrow{L_2} \theta(q)$, because $\mathcal{A}_2$ is deterministic.

We now state an important observation which is a consequence of Theorem 7.7. For each term $t \in T(\mathcal{F})$, if $C_q[t]$ is in $L_2$, then $t \xrightarrow{L_2} \theta(q)$.

By induction on the size of the term $t \in T(\mathcal{F})$ it can be shown that for each state $q \in Q_1$, if $C_q[t] \in L_1$ then $C_q[t] \in L_2$. It follows that if $t \in L_1$ then $t \in L_2$, because $C_{q_{t_1}}[t] = t$.

**Basis:** Suppose that $t$ is a symbol of arity 0. There is a transition $t \xrightarrow{L} q$ and so $C_q[t] \in CS(L_1) \subseteq L_2$.

**Inductive step:** Suppose that $t = f(t_1, \ldots, t_n)$. By the lemma hypothesis, there is a state $q_i$ in $Q_1$ such that $t_i \xrightarrow{L_1} q_i$. Because $C_{q_i}[t_i] \in L_1$, the induction hypothesis is applied to obtain $C_{q_i}[t_i] \in L_2$. Following the observation above, $t_i \xrightarrow{L_2} \theta(q_i)$. We have

$$C_q[f(t_1, \ldots, t_n)] \xrightarrow{L_2} C_q[f(\theta(q_1), \ldots, \theta(q_n))] \tag{7.1}$$

On the other hand, since $C_q[f(rt(q_1), \ldots, rt(q_n))] \in L_2$, we have

$$C_q[f(rt(q_1), \ldots, rt(q_n))] \xrightarrow{L_2} C_q[f(\theta(q_1), \ldots, \theta(q_n))] \tag{7.2}$$
7.3. IDENTIFICATION OF REVERSIBLE TREE LANGUAGES

By combining 7.1 and 7.3,

\[
C_q[f(t_1, \ldots, t_n)] \overset{*}{\rightarrow} q_{f_2}
\]

(7.3)

so \(C_q[t] \in L_2\).

Based on Angluin (1982), the characteristic samples of reversible tree languages are telltale sets. Consequently, the class of reversible tree languages is identifiable.

7.3.3 An Efficient Learning Algorithm

The learning algorithm works as follows: The input is a finite set \(S\) of positive examples, that is terms of a (regular) target language. Let us first define the prefix tree automaton \(PTA(S) = (F, Q_0, Q_{F_0}, \rightarrow)\) as follows. For each subterm \(t\) of \(\text{sub}(S)\), there is a state written \([t]\) in \(Q_0\). The set of final states \(Q_{F_0}\) contains each state \([t]\) where \(t \in S\). For each subterm \(f(t_1, \ldots, t_n)\) of \(\text{sub}(S)\), we have a transition \(f([t_1], \ldots, [t_n]) \rightarrow_0 [f(t_1, \ldots, t_n)]\). It follows directly that \(L_{PTA(S)} = S\).

Next a succession of NFTAs \(A_0 = PTA(S), A_1, \ldots, A_n\) is computed by repeatedly applying one of the reduction rules described in Figure 7.1 until we find two identical NFTAs. That is, \(A_{i+1}\) is obtained from \(A_i\) by merging two states following one of the rules. The process terminates after \(n\) reduction steps since each step decreases the number of states, so \(nf(S) = A_n\).

Lemma 7.11 The DFTA \(nf(S)\) is a reversible DFTA.

Proof: Since no rules are applicable, \(nf(S)\) is necessarily reversible.

An automaton homomorphism between the NFTA \(A_1 = (F, Q_1, Q_{F_1}, \rightarrow)\) and \(A_2 = (F, Q_2, Q_{F_2}, \rightarrow)\) is a function \(\theta\) which maps \(Q_1\) to \(Q_2\) and which has the property that if \(f(q_1, \ldots, q_n) \rightarrow q\) then \(f(\theta(q_1), \ldots, \theta(q_n)) \rightarrow_2 \theta(q)\), for each transition of \(A_1\). We say that \(A_1 \subseteq A_2\) if there is some automaton homomorphism between them. Note that if \(A_1 \subseteq A_2\) then \(L_{A_1} \subseteq L_{A_2}\).

Lemma 7.12 If \(S \subseteq L\) then \(PTA(S) = A_0 \subseteq A_L\).

Proof: Define \(\theta\) as \(\theta([t])\) is the equivalence class of \(t\).

Lemma 7.13 Let \(A\) be a reversible DFTA. Assume that \(A_{i+1}\) is obtained by applying either \(R1, R2\) or \(R3\) to \(A_i\). If \(A_i \subseteq A\) then \(A_{i+1} \subseteq A\).
1. **R1** If \( f(p_1, \ldots, p_n, q, p_{n+1}, \ldots, p_m) \rightarrow p \) and \( f(p_1, \ldots, p_n, q', p_{n+1}, \ldots, p_m) \rightarrow p \) then \( q = q' \).

2. **R2** if \( f(q_1, \ldots, q_n) \rightarrow q \) and \( f(q_1, \ldots, q_n) \rightarrow q' \) then \( q = q' \).

3. **R3** if \( q_{f_1} \) and \( q_{f_2} \) are both final states then \( q_{f_1} = q_{f_2} \).

Figure 7.1: Reduction rules

**Proof:** Let \( \theta \) be the automaton homomorphism from \( A_i \) to \( A \). We prove the lemma by inspecting the three rules which can be used to construct \( A_{i+1} \). Rule **R1** is applied. Since \( A \) is reversible, \( \theta(q) = \theta(q') \). Rule **R2** is applied. Since \( A \) is deterministic, \( \theta(q) = \theta(q') \). Rule **R3** is applied. Since \( A \) has only one final state, \( \theta(q_{f_1}) = \theta(q_{f_2}) \). Thus \( \theta \) is an automaton homomorphism from \( A_{i+1} \) to \( A \).

\[ \square \]

**Lemma 7.14** For each reversible language \( L \), if \( S \subseteq L \) then \( L_{u(S)} \subseteq L \).

**Proof:** Follows directly from Lemma 7.12 and Lemma 7.13.

\[ \square \]

**Example 7.15** Suppose we have the entry \( (f(a, b), f(g(a), b), f(g(a)), b)) \). The algorithm first calculates the automaton \( PTA(S) = (F, \mathcal{Q}, \mathcal{Q}_F, \rightarrow_0) \), where

- \( F \) is the set \( \{f, g, a, b\} \),
- \( \mathcal{Q}_0 \) is the set \( \{[a], [b], [f(a, b)], [g(a)], [f(g(a), b)], [g(g(a))], [f(f(g(a), b))]\} \),
- \( \mathcal{Q}_F \) is the set \( \{[f(g(a)), b]\} \),
- \( \rightarrow \) is the set \( \{a \rightarrow _0 [a], b \rightarrow _0 [b], f([a], [b]) \rightarrow _0 [f(a, b)], g(a) \rightarrow _0 [g(a)], f([g(a)], [b]) \rightarrow _0 [f(g(a)), b], g([g(a))] \rightarrow _0 [g(g(a))], f([g(g(a))], [b]) \rightarrow _0 [f(a, b)]\} \).

Applying the rule **R3** three times we get \( [f(a, b)] = [f(g(a), b)] = [f(f(g(a), b))] \) and the new set \( \{a \rightarrow [a], b \rightarrow [b], f([a], [b]) \rightarrow [f(a, b)], g([a]) \rightarrow [g(a)], f([g(a)], [b]) \rightarrow [f(a, b)], g([g(a))] \rightarrow [g(g(a))], f([g(g(a))], [b]) \rightarrow [f(a, b)]\} \).

Now the rule **R1** can be applied twice to obtain \( [a] = [g(a)] = [g(g(a))] \), so the output is the automaton \( (F, \mathcal{Q}, \mathcal{Q}_F, \rightarrow_0) \), where

- \( F \) is the set \( \{f, g, a, b\} \),
- \( \mathcal{Q} \) is the set \( \{[a], [b], [f(a, b)]\} \),
- \( \mathcal{Q}_F \) is the set \( \{[[f(a, b)]\} \),
Theorem 7.16 (Proof of Theorem 7.5). Assume that \( L \) is a reversible tree language. For each positive presentation \( t_1, \ldots, t_n, \ldots \) of \( L \), there is a step \( N \) such that for all \( n > N \), \( \text{nf}(t_1, \ldots, t_n) \) is the minimal reversible DFTA which recognizes \( L \). In other words, the class of reversible tree languages is identifiable.

Proof: There is a step \( N \) such that for all \( n > N \), \( t_1, \ldots, t_n \) contains \( \text{CS}(L) \). By Lemma 7.14, \( L_{\text{nf}(t_1, \ldots, t_n)} \subseteq L \). Now, \( L_{\text{nf}(t_1, \ldots, t_n)} \subseteq L \) is a reversible tree language. We apply Theorem 7.10, and we conclude that \( L \) is a regular tree language. Therefore, we have \( L = L_{\text{nf}(t_1, \ldots, t_n)} \).

The learning algorithm is incremental and runs in quadratic time in the size of the examples.

7.4 Learning with Structural Examples

We now discuss the identification in the limit of word languages and we give some applications as a way of illustrating the learning method developed in the previous sections.

7.4.1 Context-Free Word Languages

Let us recall briefly the definition of context-free grammars (CFG). A context-free grammar \( G \) is a quadruple \( (\Sigma, \mathcal{N}, \rightarrow, S) \) where \( \Sigma \) is the alphabet, \( \mathcal{N} \) is the set of non-terminal symbols, and \( S \in \mathcal{N} \) is the start symbol. The production rules are defined by \( G \) and are of the form \( X \rightarrow w \) where \( X \in \mathcal{N} \) and \( w \in (\Sigma \cup \mathcal{N})^* \). The language \( L_G \) defined by \( G \) is \( L_G = \{ w \mid S \xrightarrow{G} w \} \). Define \( \mathcal{D}(G) \) as the set of derivation trees of \( G \).

There is a close relationship between context-free word languages and regular tree grammars. To see this, let \( \Sigma \) be the set of symbol of arity 0 of \( \mathcal{F} \). We define the yield function yield from \( T(\mathcal{F}) \) to \( \Sigma \) as follows:

\[
\text{yield}(a) = a \\
\text{yield}(f(t_1, \ldots, t_n)) = \text{yield}(t_1) \ldots \text{yield}(t_n)
\]

If \( L \subseteq T(\mathcal{F}) \) is a tree language, then \( \text{yield}(L) = \{ \text{yield}(t) \mid t \in L \} \).

Theorem 7.17 If \( L \) is a regular tree language then \( \text{yield}(L) \) is a context-free language.

The second thing is that the set of derivation trees of context-free grammars is a regular tree language.
Theorem 7.18 Assume that $G$ is a context-free grammar. Then, the set of derivation trees $D(G)$ is a regular tree language.

Let us look at the construction of a reversible and normal RTG $\Gamma = \langle F, X, \rightarrow, S \rangle$ from a grammar $G = \langle \Sigma, N, \rightarrow, S \rangle$. Every letter in $\Sigma$ is a 0-ary symbol in $F$. For each rule $X \rightarrow w$ where $w$ is a word of length $n$, there is a symbol $X_n$ of arity $n$ in $F$. Every non-terminal of $N$ is a variable of $X$. For each letter $a \in \Sigma$, there is a variable $[a]$. For each production $X \rightarrow \prod w_i$ where $w_i$ is a word of length $n$, there is a production $X \rightarrow X_n(\prod X_i)$ where $X_i = w_i$ if $w_i$ is in $N$, otherwise $X_i = [w_i]$. Lastly, for each letter $a \in \Sigma$, we have $[a] \rightarrow a$.

7.4.2 Structural Examples

In the case of context-free languages it is in general hard to prove learnability of string languages (or coming up with sufficient conditions for such classes to be learnable). Consequently several authors have suggested to learn classes of grammars, like context-free grammars Sakakibara (1992) or categorial grammars Kanazawa (1998), where positive examples are annotated by additional information and are called structural examples. We present two cases to illustrate this idea.

1. The full presentation of a CFG $G$ is a sequence $t_1, t_2, \ldots$ of all parse trees of $G$.

2. A delabeling $sk$ is function defined by:

$$sk(a) = a$$
$$sk(f(t_1, \ldots, t_n)) = \sigma_n(sk(t_1), \ldots, sk(t_n))$$

The skeleton presentation of a grammar $G$ is a sequence $sk(t_1), sk(t_2), \ldots$ of all delabeled parse trees of $G$.

The available data consists of a regular tree language which is a homomorphic image of derivation trees of some CFG. These observations lead to the consideration of $h$-presentations of a CFG $G$ defined as follows: assume that $h$ is a tree homomorphism, then an $h$-presentation of a CFG $G$ is a sequence $h(t_1), \ldots, h(t_n), \ldots$ where $t_1, \ldots, t_n, \ldots$ is an enumeration of all parse trees of $G$.

Note that $h$ being a linear homomorphism is a sufficient condition for an $h$-presentation to be a regular tree language.

7.4.3 Grammar Identification

Let $h(D(G)) = \{h(t) | t \in D(G)\}$. Let $\Omega$ be a given class of grammars and $h$ be a tree homomorphism. Given a $h$-presentation $h(t_1), \ldots, h(t_n), \ldots$ the inference
algorithm $A$ converges to $G \in \Omega$ if there is a stage $N$ such that for all $n > N$, the language provided by $A(h(t_1), \ldots, h(t_n))$ is exactly $h(D(G))$.

A class of grammars $\Omega$ is identifiable from $h$-presentations if and only if there is an inference algorithm $A$ such that for each $h$-presentation of a grammar $G$, $A$ converges to $G$.

7.4.4 Sakakibara’s Approach
Sakakibara demonstrated that the class of reversible context-free grammars is identifiable from skeleton presentations.

Definition 7.19 A CFG is reversible if and only if

1. (Reset-free) there are no productions $X \xrightarrow{G} wYv$ and $X \xrightarrow{G} wZv$ such that $Y$ and $Z$ are non-terminals and $Y \neq Z$,

2. (Deterministic) there are no productions $X \xrightarrow{G} t$ and $Y \xrightarrow{G} t$ with the same right hand-side and such that $X \neq Y$.

A reversible context-free language is a language produced by a reversible CFG.

Theorem 7.20 (Sakakibara (1992)). The class of reversible context-free grammars is identifiable from skeleton presentations.

Proof: The skeleton presentation of a reversible CFG is a reversible tree language, so Theorem 7.5 implies that the class of skeleton presentations is identifiable. □

7.4.5 Identification of Reversible Context-Free Word Languages
Definition 7.21 The class $\mathcal{R}$ of reversible context-free word languages is the smallest class such that for every reversible tree language $L_1$ there is a language $L$ in $\mathcal{R}$ such that $L = \text{yield}(L_1)$. We call $L_1$ a reversible tree presentation of $L$.

Again from Theorem 7.5 the following result is obtained:

Theorem 7.22 The class $\mathcal{R}$ is identifiable from reversible tree presentations.

In order to make this result clear and to compare it with Sakakibara’s work, we shall illustrate it by discussing some of its consequences.
Example 7.23 The grammar $\Gamma$ defined below is reversible:

\[
\begin{align*}
S & \rightarrow f(X, Y) & X & \rightarrow c \\
X & \rightarrow g(A, X, B) & Y & \rightarrow d \\
Y & \rightarrow g(A, Y, B) & B & \rightarrow b \\
A & \rightarrow a
\end{align*}
\]

By Theorem 7.5, $L_\Gamma$ is identifiable. Theorem 7.22 states that the context-free language $\text{yield}(L_\Gamma)$ is identifiable from any enumeration of $L_\Gamma$ which constitutes a reversible tree presentation. However, $L_\Gamma$ is not the set of parse trees of a context-free language.

Example 7.24 This example shows that learning is facilitated by a full presentation of parse trees. The following CFG $G$ is not reversible because it is not deterministic:

\[
\begin{align*}
S & \rightarrow ab \\
S & \rightarrow aXb \\
X & \rightarrow ab
\end{align*}
\]

The set $D(G)$ of parse trees is a reversible tree language and is generated by the following (reversible and minimal) tree grammar:

\[
\begin{align*}
S & \rightarrow_G S_2([a], [b]) \\
S & \rightarrow_G S_3([a], X, [b]) \\
X & \rightarrow_G X_2([a], [b]) \\
[a] & \rightarrow_G a \\
[b] & \rightarrow_G b
\end{align*}
\]

Theorem 7.25 The class of reset-free context-free languages is identifiable from full presentations.

7.5 Lexical Dependency Tree Languages

Following Dikovsky and Modina (2000),\footnote{Also see Dikovsky (2001), for details on a dependency tree grammar formalism that can generate non-semilinear context-sensitive languages.} we present a class of projective dependency grammars that was introduced in Hays (1961) and Gaifman (1965).
A lexical dependency grammar (LDG) is a quadruplet \((\Sigma, N, \rightarrow, S)\), where \(\Sigma\) is the alphabet, \(N\) is the set of non-terminal symbols, and \(S \in N\) is the start symbol. Each production is of the form

\[ X \rightarrow U_1, \ldots, U_p a V_1, \ldots, V_q, \]

where \(X \in N\) and all \(U_i\) and \(V_j\) are in \(\Sigma \cup N\). The right-hand side can be interpreted either as a labeled ordered directed tree of depth 1 whose head is \(a\), or as the word \(U_1, \ldots, U_p a V_1, \ldots, V_q\). Thus there is a total order on the tree nodes.

**Example 7.26** The grammar \(\Gamma_0 = \langle \{a, b\}, \{S\}, P, S \rangle\) where \(P\) consists of:

\[
S \rightarrow a \ S \ b \mid a \ b
\]

We define partial dependency trees recursively as follows:

1. \(S\) is a partial dependency tree generated by \(\Gamma\).

2. If

\[
\ldots X \ldots b \ldots
\]

is a partial dependency tree generated by \(\Gamma\), and if

\[
X \rightarrow U_1 \ldots U_p a V_1 \ldots V_q
\]

is a production of \(\Gamma\), then

\[
\ldots U_1 \ldots U_p a V_1 \ldots V_p \ldots b \ldots
\]

is a partial dependency tree generated by \(\Gamma\).

A dependency tree generated by an LDG \(\Gamma\) is a partial dependency tree of \(\Gamma\) in which all nodes are terminal symbols. The language \(D(\Gamma)\) is the set of all dependency trees generated by \(\Gamma\).

**Example 7.27** The language generated by \(\Gamma_0\) is

\[
D(\Gamma_0) = \{ab, aabb, aaabbb, \ldots\}
\]
Without dependencies, we recognize the context-free language \( \{ a^n b^n \mid n > 0 \} \).
Without the linear order on letters, the regular tree language is
\[
\{ a(b), (a(a(b), b)), a(a(b), b), b, \ldots \}.
\]

Note that the arity of \( a \) is 1 or 2, but this is not problematic. We write \( \alpha \alpha \beta \)
for \( \alpha_1 \alpha_2 \alpha_3 \beta_1 \beta_2 \).

**Definition 7.28** An LDG grammar is reversible if and only if the following three conditions are satisfied:

1. if \( X \rightarrow UaV \) and if \( Y \rightarrow UaV \), then \( X = Y \).

2. If \( X \rightarrow \alpha Y \beta \alpha \gamma \) and if \( X \rightarrow \alpha Z \beta \alpha \gamma \), then \( Y = Z \), where \( Y, Z \in N \).

3. If \( X \rightarrow \alpha \alpha \beta Y \gamma \) and if \( X \rightarrow \alpha \alpha \beta Z \gamma \), then \( Y = Z \), where \( Y, Z \in N \).

The class of reversible dependency tree languages is the class of languages generated by reversible LDG grammars.

**Theorem 7.29** The class of reversible dependency trees is identifiable.

**Proof:** The algorithm described in Section 7.3.3 is easily adapted to take the node ordering into account and to handle symbols with variable arities. It is not difficult to see that a presentation is a reversible tree language. \( \square \)

### 7.6 Classical Categorial Grammar

We are now ready to present Besombes and Marion’s alternative proof of Kanzawa’s theorem on learning \( \mathcal{L}_{\text{rigid}} \) from structures. We feel that its conciseness demonstrates the power of the tree automata-approach to learning:
Theorem 7.30 (Kanazawa). The class of rigid grammars is identifiable from unlabeled derivation tree presentations.

Proof: The set of unlabeled derivation trees is a reversible regular language. To see this, construct a normal RTG $\Gamma$ such that for each subtype $A$ of a type assigned to a symbol there is a state $[A]$. For each symbol $u$ of $\Sigma$ there is a corresponding production $[\text{Lex}(u)] \rightarrow u$. For each possible subtype $A$ of such a type we have the following productions (known as functional productions):

\[
[B] \rightarrow \setminus([A], [A \backslash B]) \\
[A] \rightarrow /([A / B], [B])
\]

Note that functional productions are reversible. Since the grammar is rigid, $\Gamma$ is reversible. Thus, by Theorem 7.5, this class is learnable. \qed