Foundations of XML (CSC2538)
Lecture 3

Prof. Leonid Libkin
notes by Anthony Widjaja To

November 4, 2005

1 Introduction

In the last lecture, we study the notion of binary trees and automata over them. In this lecture, we study the notion of (labeled) unranked trees, which is a realistic model of XML documents. We then define a class of automata over unranked trees, which define precisely all regular unranked tree languages. It turns out that this more general notion of trees can be studied by restricting ourselves to binary trees. That is, given an unranked tree, it is possible to construct a binary tree in such a way that (ir)regularity, FO-(in)expressibility, and MSO-(in)expressibility are preserved. From this, we immediately get a result that connects regularity and MSO-definability for unranked tree languages. Following this discussion, we move to a different topic: formal model of XML schema languages. *Extended context-free grammars* serve as our formal model for DTDs. We also note a particular weakness of DTDs, that it is impossible to associate two different contexts for one tag label. So, we introduce the definition of *extended DTDs*, which captures schema languages that are more expressive than DTDs such as *XSD*. It turns out that extended DTDs capture precisely MSO. In the sequel, unless stated otherwise, "trees" always mean "(labeled) unranked trees".

2 Formal model of XML documents

In previous lectures, we saw that XML documents can be viewed as trees each of whose nodes may have an arbitrarily large, but finite, number of children. These types of trees are what we call labeled unranked trees.

2.1 Unranked Trees

An *unranked tree domain* is a finite prefix-closed subset of $\mathbb{N}^*$ (i.e. that $s \in D$ implies that any prefix of s belongs to D) such that whenever $s.i \in D$, we have $s.j \in D$ for all $j \leq i$. We now define three ordering relations over D:

1. Child relation $\prec_{\text{ch}}^*$. This relation is the transitive closure of $\prec_{\text{ch}}$ which is defined as: $s \prec_{\text{ch}} s'$ iff $s' = s.i$ for some $i \in \mathbb{N}$. When $s \prec_{\text{ch}}^* s'$, we say that s is an *ancestor* of s' (and s' is a *descendant* of s).

2. Next sibling relation $\prec_{\text{sib}}^*$. This relation is the transitive closure of $\prec_{\text{sib}}$ which is defined as: $s \prec_{\text{sib}} s'$ iff $s = s_0.i$ and $s' = s_0.(i+1)$ where $s_0 \in D$ and $i \in \mathbb{N}$. When $s \prec_{\text{sib}}^* s'$, we say that s is *older* than s' (and s' *younger* than s).

3. First-child relation: $s \prec_{\text{fc}} s.0$.

Next we fix a finite non-empty alphabet Σ. We then partition D into several disjoint (not necessarily non-empty) sets $\{P_a\}_{a \in \Sigma}$. The intended interpretation for $s \in P_a$ is that the label for s is a. Intuitively, a label corresponds to an XML tag.

Definition 2.1 An *unranked tree* is a structure of the form

$$\langle D, \prec_{\text{ch}}^*, \prec_{\text{sib}}^*, (P_a)_{a \in \Sigma} \rangle.$$

A *(unranked) tree language* over Σ is a set of trees over Σ. For our purposes, we always assume that, for every tree language L, there exists an element $a \in \Sigma$ that is used to label the root of every tree in L. For a tree T, we will also define a labeling function $\lambda : D \to \Sigma$ with respect to the labeling defined by $\{P_a\}_{a \in \Sigma}$.

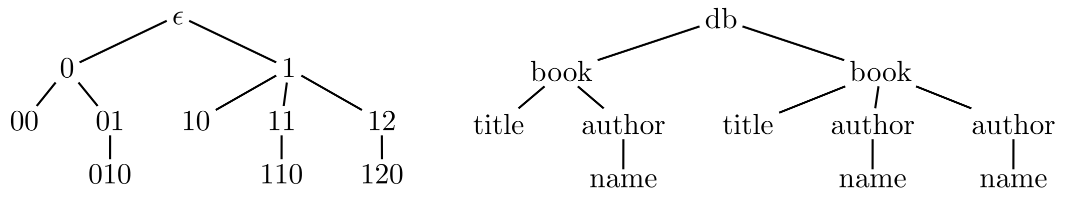

Figure 1: An unranked tree skeleton and its labeled version

See figure 1 for an example. When we talk about trees with respect to MSO, we will replace $\prec_{\text{ch}}^*$ by $\prec_{\text{ch}}$ and $\prec_{\text{sib}}^*$ by $\prec_{\text{sib}}$ in our structures and vocabularies, as the former is MSO-definable using the latter.

A few remarks are in order. Notice that we did not include $\prec_{\text{fc}}$ as it is definable using $\prec_{\text{ch}}^*$ and $\prec_{\text{sib}}^*$ in any reasonable logic. Observe also that this model is an approximation of real XML documents as it does not incorporate attributes. [This is a reasonable design choice because, firstly, most XML documents do not use this feature, and, secondly, it is wise to minimize the use of attributes in an XML document as they should only be used to represent *extra* information.]

2.2 Unranked Tree Automata

A *nondeterministic unranked tree automaton (NUTA)* $\mathcal{A}$ is a triplet (Q, Q_0, δ) where $Q_0 \subseteq Q$, and $\delta : Q \times \Sigma \to 2^{Q^*}$ such that the output of δ is always a regular language represented by a regular expression over Q. In this definition, Q is a set of states and Q_0 the set of all initial states. A *run* of $\mathcal{A}$ on a tree T is a function $\rho : D \to Q$ such that, for every $s \in D$, if $\lambda(s) = a$ and s has children $s.0, \ldots, s.(m-1)$, then

$$\rho(s.0) \cdots \rho(s.(m-1)) \in \delta(\rho(s), a).$$

Notice that $\epsilon \in \delta(\rho(s), \lambda(s))$ for every leaf s of T, which is the reason why we need not include "final states" in $\mathcal{A}$. [One may easily switch between the view of NUTA as bottom-up and top-down parallel automata by identifying Q_0 appropriately (either as initial states, or final states).] A run is said to be *accepting* if $\rho(\epsilon) \in Q_0$. A Σ-tree T is *accepted* by $\mathcal{A}$ is there exists an accepting run of $\mathcal{A}$ on T. The set of trees accepted by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. Finally, a tree language L (i.e. a set of unranked trees) over Σ is *regular* if there exists a NUTA $\mathcal{A}$ over Σ such that $L = L(\mathcal{A})$. The following theorem is folklore:

Theorem 2.1 *An unranked tree language is regular iff it is definable in MSO*

2.3 Translation from unranked trees to binary trees

We first state the main theorem.

Theorem 2.2 (Rabin 1971 & 1972) *There exists a recursive function r_Σ from the set of all unranked trees over Σ to the set of all binary trees over Σ such that for any unranked tree language L over Σ:*

1. *L is regular iff $r(L)$ is regular,*
2. *L is MSO-definable iff $r(L)$ is MSO-definable, and*
3. *L is FO-definable iff $r(L)$ is FO-definable.*

Fact 2.3 *r in the above theorem is MSO-definable.*

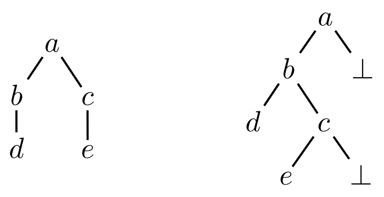

Figure 2: Translating unranked trees to binary trees

So, theorem 2.1 is a corollary of the above theorem and the famous Doner-Thatcher-Wright theorem, which was discussed in the previous lecture. We now sketch how r is defined. We inductively define a map $\mathcal{R} : \mathbb{N}^* \to \{0,1\}^*$ as follows:

1. $\mathcal{R}(\epsilon) \stackrel{\text{def}}{=} \epsilon$ and $\mathcal{R}(0) \stackrel{\text{def}}{=} 0$.
2. If $\mathcal{R}(s) = s'$ where $s = s_0.i$ and $i \in \mathbb{N}$, then $\mathcal{R}(s.0) \stackrel{\text{def}}{=} s'.0$ and $\mathcal{R}(s_0.(i+1)) \stackrel{\text{def}}{=} s'.1$.

When D is a tree domain, we define

$$\mathcal{R}(D) \stackrel{\text{def}}{=} \{\mathcal{R}(s) : s \in D\}.$$

Furthermore, we define

$$r(D) \stackrel{\text{def}}{=} \mathcal{R}(D) \cup \{\mathcal{R}(s).i : s \in D,\ \mathcal{R}(s).(i-1) \in \mathcal{R}(D), \text{ but } \mathcal{R}(s).i \notin \mathcal{R}(D)\}.$$

The label of the nodes in $\mathcal{R}(D)$ follow those of D, whereas the new extra nodes are labeled by the symbol $\perp \notin \Sigma$. See figure 2 for an example.

3 Formal Model of DTDs

There are two kinds of XML documents: those that are "valid", and those that are "invalid". In order to distinguish them, we have to impose an XML *schema* on these documents. XML Document Type Definition (DTDs) is one possible (and the most commonly used) such schema language. In this section, we focus on a formal model of DTDs. As we do not incorporate attributes in our formal model of XML documents, we shall restrict ourselves only to DTDs *without* attribute declarations.

Definition 3.1 Suppose Σ is a fixed finite alphabet (of "element types"). A *DTD* over Σ is $d = (r, P)$ where r is a distinguished element of Σ, which we call *root*, and $P : \Sigma \to \text{regex}(\Sigma)$ where $\text{regex}(\Sigma)$ is the set of all regular expressions over Σ.

Notice the similarity between the definition of context-free grammars (CFGs) and that of DTDs. The only differences are that the body of each production rule in P can contain regular expressions (like $a*$) and that there is exactly one production rule in P for every element of Σ, whereas both of these constraints are not imposed for CFGs. In this sense, we can think of DTDs as *extended context-free grammars (ECFGs).*

Example 3.1 Suppose we have the following DTD:

```
<!DOCTYPE db [
    <!ELEMENT db    (book)*>
    <!ELEMENT book  (title,author+)>
    <!ELEMENT author (name)>
    <!ELEMENT name  (#PCDATA)>
]>
```

This can be modelled as follows. Set $\Sigma \stackrel{\text{def}}{=} \{\text{db}, \text{book}, \text{title}, \text{author}, \text{name}\}$, and $d = (\text{db}, P)$, where P is defined as follows:

$$\begin{aligned} \text{db} &\mapsto \text{book}^* \\ \text{book} &\mapsto \text{title.author}^+ \\ \text{author} &\mapsto \text{name} \\ \text{name} &\mapsto \epsilon \end{aligned}$$

Notice that we map name to ϵ. This is because we are interested only in the structure of XML documents. For an example of a tree that satisfies this DTD (we will define this notion shortly), see figure 1. $\dashv$

Definition 3.2 Suppose that T is a tree over Σ and $d = (r, P)$ is a DTD over Σ. Then, we say that T *satisfies* d (or T is *valid* with respect to d), written $T \models d$, if:

1. The root of T is labeled r.
2. For each node s in T labeled a with children labeled $a_0, \ldots, a_{m-1}$, we have $a_0. \ldots .a_{m-1} \in P(a)$.

Furthermore, we define $SAT(d) = \{T : T \models d\}$, i.e., the set of trees that satisfy d (or more intuitively, the set of trees that are "generated" by d).

It is not hard to see that there is a NUTA that recognizes exactly $SAT(d)$ for any given d. However, in general the set

$$\{SAT(d) : \text{d is a DTD over } \Sigma\}$$

does not capture all the regular tree languages. This is because this set is in general not closed under union and complementation (exercise!). Fortunately, we can find a natural extension of the notion of DTDs in order to capture all the regular tree languages.

Definition 3.3 An *extended DTD* over Σ is a triplet (Σ', μ, d') where Σ' is any finite alphabet with $\Sigma \subseteq \Sigma'$, $\mu : \Sigma' \to \Sigma$ is a "specialization" function, and d' is a DTD over Σ'.

We say that a tree T (over Σ) satisfies d, denoted $T \models d$, if there exists a tree T' over Σ' that satisfies d' such that $\mu(T') = T$, i.e., replacing each label a in T' by $g(a)$ gives us T. The set of Σ-labeled trees that satisfy d is denoted $SAT(d)$.

Theorem 3.2 (Thatcher 1968) *The set*

$$\{SAT(d) : d \textit{ is an extended DTD over } \Sigma\}$$

is precisely the set of all regular tree languages over Σ.

This theorem was reproved by Vianu in 1999.

Proof Idea We will sketch how to prove one direction: L is a regular tree language over Σ only if there exists an extended DTD d that satisfies precisely the unranked trees in L. So, assume that L is a regular tree language with a distinguished root label $a \in \Sigma$. Then, there must exists an automaton $\mathcal{A} = (Q, Q_0, \delta)$ for L. Without loss of generality, one may assume that $Q_0 = \{q_0\}$ (exercise!). Let us now define $d = (\Sigma', \mu, d')$:

1. $\Sigma' \stackrel{\text{def}}{=} \Sigma \times Q$,
2. for each $b \in \Sigma$ and $q \in Q$, $\mu((a, q)) = a$,

3. set the root element of d' to be a, and

4. for each $b \in \Sigma$ and $q \in Q$, set $d'(a, q) = e$ where e is the output of $\delta(q, a)$ in which each occurrence of $q' \in Q$ is replaced by the disjunction of (q', b) where b ranges over all Σ.

We leave it as an easy exercise for the reader to check that this construction works. $\square$