Automata and Logic (CSC2428)
Lecture 8

Prof. Leonid Libkin
notes by Alvin Chin

November 18, 2005

1 Streaming of XML Data

XML data can be encoded into a tree representation. There are two models for encoding of trees: 1) DOM model and 2) SAX model.

1.1 DOM Model

The DOM model stands for Document Object Model. DOM is what is normally used for parsing XML documents and for constructing XML documents. The DOM model uses a pointer structure for traversal of the XML document. There are pointers to *parent*, *firstChild*, *nextSibling*, and *prevSibling* as illustrated in Figure 1 below.

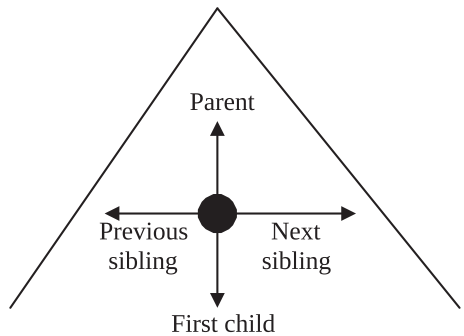

Figure 1: Pointer structure illustrating the DOM model

1.2 SAX Model

The SAX model stands for Simple API for XML. XML documents are parsed as we see them in the tree in a depth-first traversal. The XML document tree becomes a string and it is as if we go through an ASCII file. Figure 2 below shows the tree for an XML document.

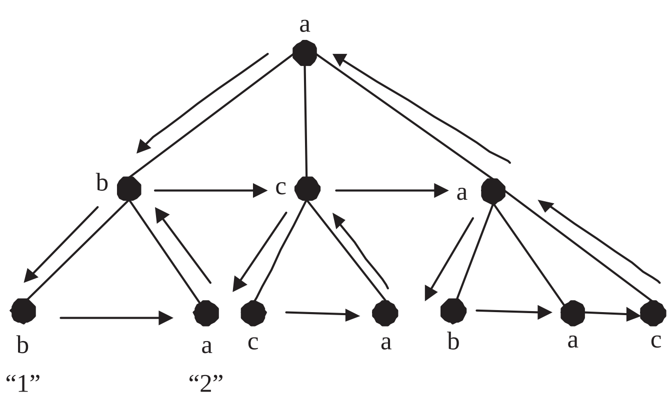

Figure 2: Tree representation of an XML document in the SAX model

Below shows the same XML document as text in a file.

$< a >$

$\quad < b >$

$\qquad < b > .. < /b >$

$\qquad < a > .. < /a >$

$$
$<c>$
$<c> .. </c>$
$<a> .. $
$</c>$
$<a>$
$ .. $
$<a> .. $
$<c> .. </c>$
$$
$<a>$

In SAX, the tree becomes a string, so the above would become (ignoring the indentation):

$<a> .. <a> .. <c><c> .. </c><a> .. </c><a> .. <a> .. <c> .. </c>$

Another way to parse SAX is to parse it as a string. We ignore the data values and just look at the structure. How do we define the structure?

Let *s(T)* be the string representation of tree T (tree representation of the XML document).

1) If T is a single-node tree a then

$$s(T) = <a>$$

2) If T has root a and subtrees T_1, T_2,...T_k as in Figure 3

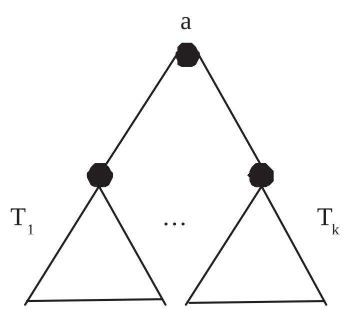

Figure 3: Example of s(T) with subtrees

$$s(T) = <a> s(T_1)...s(T_k) $$

Note that *s(T)* is a concatenation of subtrees T_1, T_2,...T_k with the root a. For our example XML document, the entire string representation is:

$s(T_0) = <a> .. <a> .. <c><c> .. </c><a> .. </c><a> .. <a> .. <c> .. </c>$

For each opening tag, there must be a closing tag. Let

$L_{Tree} = \{s(T) | T$ ranges over all trees over $\Sigma \bigcup \overline{\Sigma}\}$ and $\overline{\Sigma} = \{\overline{a} | \mathrm{a} \in \Sigma\}$ (all the closing tags)

L_{Tree} is not a regular language.

A reasonable model for parsing the string (by looking at the string or part of the XML document) is to use finite automata.

We are interested in classes C of trees such that:

$$\{s(T) | T \in C\} \text{ is regular (in constant space)}$$

This is quite bad because if C is a class of all trees, then this set is not regular.

Question: Does there exist a finite automaton $\mathcal{A}_c$ such that for every tree T,

$$\mathcal{A}_c \text{ accepts } s(T) \Longleftrightarrow T \in C$$

We do not know what C is, we want to find C.

1.3 Proposition (Segoufin/Vianu, 2002)

If C_d is a class of trees given by a non-recursive DTD d (dependency graph but no cycles) then $\mathcal{A}_{C_d}$ exists

$$d \longrightarrow \mathcal{A}_{C_d}$$

if $\mathcal{A}_{C_d}$ is an NFA it can be constructed in exponential time. We just need to validate the string against the NFA.

Definition: A property of trees (eg. DTD) is streamable if there exists a finite automaton $\mathcal{A}$ such that $\mathcal{A}_{C_d}$ accepts $s(T) \Longleftrightarrow T$ has that property that satisfies the DTD.

Observation: There are very simple recursive DTDs which are not streamable. Figure 4 shows an example of a recursive DTD.

$$r \longrightarrow aa$$
$$a \longrightarrow a \mid \epsilon$$

$$r\ a...a\ \overline{a}...\overline{a}\ a...a\ \overline{a}...\overline{a}\ \overline{r} = r\ a^n\ \overline{a}^n\ a^m\ \overline{a}^m\ \overline{r}$$

where $r = <r>,\ a = <a>,\ \overline{a} = ,\ \overline{r} = </r>$ for simplicity and

$$s(T') = r\ a^{n+k}\ \overline{a}^n\ a^m\ \overline{a}^{m+k}\ \overline{r} \text{ (need to balance)}$$

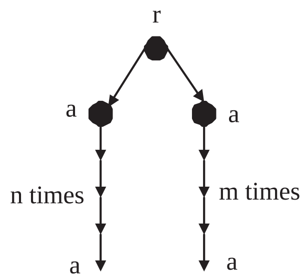

Figure 4: Example of a simple recursive DTD

Example: For every finite automaton $\mathcal{A}$, one can find n, m, k such that $\mathcal{A}$ cannot distinguish

$$r\ a^n\ \overline{a}^n\ a^m\ \overline{a}^m\ \overline{r} \text{ and } r\ a^{n+k}\ \overline{a}^n\ a^m\ \overline{a}^{m+k}\ \overline{r}$$

What is T'? Figure 5 illustrates this.

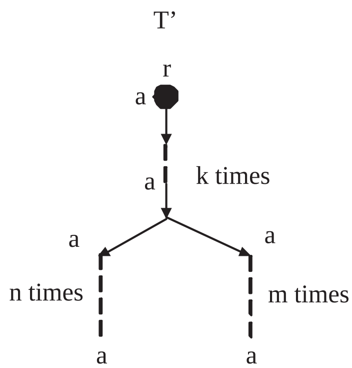

Figure 5: Tree T'

Open problems:

1) Characterize streamable DTDs, extended DTDs.
2) Is being streamable decidable?

1.4 Streamable recursive DTD

$$r \longrightarrow a$$
$$a \longrightarrow a \mid \epsilon$$

$$\{s(T) \mid T \models D\} = \{r\ a^n\ \overline{a}^n\ \overline{r} \mid n \geq 0\}$$

The automaton that realizes this is illustrated in Figure 6 below.

$$r\ a^*\ \overline{a}^*\ \overline{r} \bigcap \{s(T) \mid T \text{ is a tree}\} = \{s(T) \mid T \models D\}$$

Property C is streamable if there exists an automaton $\mathcal{A}$ such that

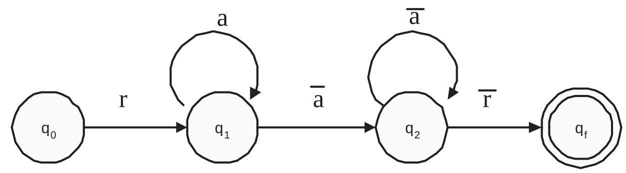

Figure 6: Automaton for example streamable recursive DTD

$$L(\mathcal{A}) \bigcap \{\ s(T) \mid T \text{ is a tree }\} = \{\ s(T) \mid T \text{ has } C\ \}$$

The automaton is not balanced but when $\bigcap$ with $s(T)$ then it becomes balanced.

P_a - labelling predicates
$\prec_{fc}$ - first child

For each regular language L,

$$U_L(x) \text{ is true if } w_x \in L$$

This is shown in Figure 7.

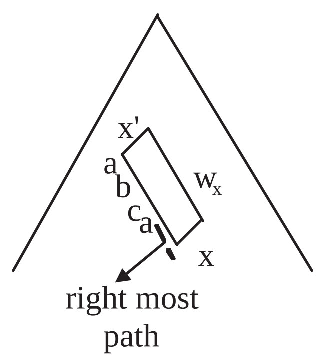

Figure 7: Showing w_x on the right most path of the tree

<u>Theorem</u> An extended DTD is streamable $\Longleftrightarrow$ it is definable in MSO in vocabulary.

$$(P_a)_{a \in \sum}, \prec_{fc}, (U_L)_{L\ regular}$$

<u>Problems:</u>

1) What to do with unary queries? (which node to select because there is no lookahead)
2) Data values? What happens when add data values - how does this affect streaming