Hedge automata: a formal model for XML schemata

MURATA Makoto (FAMILY Given)
Fuji Xerox Information Systems

October 6, 1999

1 Introduction

This note shows preliminaries of the hedge automaton theory. In the XML community, this theory has been recently recognized as a simple but powerful model for XML schemata. In particular, the design of two schema languages for XML, namely RSL(Regular Schema Language) and DSD (Document Structure Description), is directly based on this theory.

2 Hedges

First, we introduce *hedges*. Informally, a hedge is a sequence of trees. In the XML terminology, a hedge is a sequence of elements possibly interevened by character data (or types of character data); in particular, an XML document is a hedge.

A *hedge* over a finite set Σ (of symbols) and a finite set X (of variables) is:

(1) ϵ (the null hedge),

(2) x, where x is a variable in X,

(3) $a\langle u \rangle$, where a is a symbol in Σ and u is a hedge (the addition of a symbol as the root node), or

(4) uv, where u and v are hedges (the concatenation of two hedges).

Figure 1 depicts three hedges: $a\langle \epsilon \rangle$, $a\langle x \rangle$, and $a\langle \epsilon \rangle b\langle b\langle \epsilon \rangle\, x\rangle$. Observe that elements of Σ (i.e., a and b) are used as labels of non-leaf nodes, while elements of X (i.e., x) are used as labels of leaf nodes. We abbreviate $a\langle \epsilon \rangle$ as a. Thus, the third example is denoted by $a\, b\langle b\, x\rangle$.

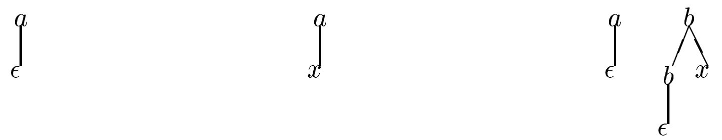

Figure 1: Three hedges: $a\langle \epsilon \rangle$, $a\langle x \rangle$, and $a\langle \epsilon \rangle b\langle b\langle \epsilon \rangle\, x\rangle$

Next, we consider an XML document. Suppose that $\Sigma = \{\texttt{doc}, \texttt{title}, \texttt{image}, \texttt{para}\}$ and $X = \{\texttt{\#PCDATA}\}$. Then, $\texttt{doc}\langle \texttt{title}\langle \texttt{\#PCDATA}\rangle\ \texttt{para}\langle \texttt{\#PCDATA}\rangle\ \texttt{image}\ \texttt{para}\langle \texttt{\#PCDATA}\rangle\rangle$ is a hedge. In the XML syntax, this hedge can be represented as below:

```
<doc>
    <title>#PCDATA</title>
    <para>#PCDATA</para>
    <image/>
    <para>#PCDATA</para>
</doc>
```

3 Regular Hedge Grammar

In this section, we introduce *regular hedge grammars* (RHGs). An RHG is a mechanism for generating hedges. In other words, an RHG describes a set of hedges.

Since the primary role of an XML schema is to describe a set of valid documents, an RHG can be considered as a formal representation of a XML schema.

A *regular hedge grammar* (RHG) is a 5-tuple $\langle \Sigma, X, N, P, r_{\mathrm{f}} \rangle$, where:

(1) Σ is a finite set of symbols,

(2) X is a finite set of variables,

(3) N is a finite set of non-terminals,

(4) P is a finite set of *production rules*, each of which takes one of the two forms as below:

  (a) $n \to x$, where n is a non-terminal in N, and x is a variable in X,

  (b) $n \to a\langle r \rangle$, where n is a non-terminal in N, a is a symbol in Σ, and r is a regular expression comprising non-terminals,

(5) r_{f} is a regular expression comprising non-terminals.

Now, we consider *derivaton* of RHGs. Informally speaking, given a sequence of non-terminals, we repeatedly replace non-terminals with hedges in the right-hand side of corresponding production rules.

Hedge v is *directly derived* from hedge u when:

(1) for some production rule $n \to x$, hedge v is obtained by replacing an occurence of n in u by x, or

(2) for some production rule $n \to a\langle r \rangle$, hedge v is obtained by replacing an occurence of n in u by some $a\langle w \rangle$ such that w is a sequence of non-terminals and matches r.

The *language generated by* G, denoted by $L(G)$, is the set of hedges which are derived from some non-terminal sequence that matches r_{f}.

Consider an RHG $G = \langle \{a\}, \{x\}, \{n_1, n_2\}, P, n_1? \rangle$, where:

$$P = \{n_1 \to a\langle n_2^+ \rangle,\ n_2 \to x\}.$$

Then,

$$L(G) = \{\epsilon, a\langle x \rangle, a\langle xx \rangle, a\langle xxx \rangle, \ldots\}$$

Next, we construct an RHG that mimicks a DTD. As an example, consider a DTD as follows:

```
<!ELEMENT  doc    (title, (para|image)*)>
<!ELEMENT  title  (#PCDATA)>
<!ELEMENT  para   (#PCDATA)>
<!ELEMENT  image  EMPTY>
```

This DTD can be captured by an RHG $G = \langle \Sigma, X, N, P, n_d \rangle$ where

$$\begin{array}{rcl}
\Sigma & = & \{\texttt{doc}, \texttt{title}, \texttt{image}, \texttt{para}\}, \\
X & = & \{\texttt{\#PCDATA}\}, \\
N & = & \{n_d, n_t, n_p, n_i, n_\#\} \\
P & = & \{n_d \rightarrow \texttt{doc}\langle n_t(n_p|n_i)^* \rangle, \\
 & & n_t \rightarrow \texttt{title}\langle n_\# \rangle, \\
 & & n_p \rightarrow \texttt{para}\langle n_\# \rangle, \\
 & & n_i \rightarrow \texttt{image}\langle \epsilon \rangle, \\
 & & n_\# \rightarrow \texttt{\#PCDATA}, \}
\end{array}$$

Next, consider an RHG $G = \langle \Sigma, X, N, P, n_1 \rangle$ where

$$\begin{array}{rcl}
\Sigma & = & \{\texttt{segment}, \texttt{para}\}, \\
X & = & \{\texttt{\#PCDATA}\}, \\
N & = & \{n_1, n_2, n_p, n_\#\} \\
P & = & \{n_1 \rightarrow \texttt{segment}\langle n_p^* n_2^* \rangle, \\
 & & n_2 \rightarrow \texttt{segment}\langle n_p^* \rangle, \\
 & & n_p \rightarrow \texttt{para}\langle n_\# \rangle, \\
 & & n_\# \rightarrow \texttt{\#PCDATA}, \}
\end{array}$$

Both the rule for non-terminal n_1 and that for n_2 have s*egment* in the right-hand side. However, the former has content model $n_p^* n_2^*$, and the latter has content model n_p^*. This impiles that top-level segments can have subordinate segments, but these subordinate segments cannot have subordinate segments.

The DTD syntax cannot exactly capture this RHG, since every occurrence of segments is forced to have the same content model. The smallest DTD that covers this RHG is as below:

```
<!ELEMENT segment (para*, segment*)>
<!ELEMENT para    (#PCDATA)>
```

Observe that this DTD allows unlimited nesting of segments. Since the DTD syntax does not allow two content models for segments, this DTD uses one loose content model.

4 Hedge Automaton

In this section, we introduce deterministic hedge automata and non-deterministic hedge automata.

A *deterministic hedge automaton* (DHA) is $\langle \Sigma, X, Q, \alpha, \iota, F \rangle$, where:

(1) Σ is a finite set of symbols,

(2) X is a finite set of variables,

(3) Q is a finite set of states,

(4) α is a function from $\Sigma \times Q^*$ to Q such that for every $q \in Q$ and $x \in \Sigma$, set $\{q_1 q_2 \ldots q_k \mid k \geq 0, \alpha(x, q_1 q_2 \ldots q_k) = q\}$ is a regular set,

(5) ι is a function from X to Q, and

(6) F is a regular set over Q.

Figure2 shows the execution of a DHA for a hedge shown in Figure 1.

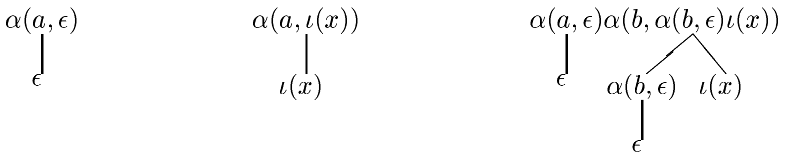

Figure 2: Execution of a deterministic hedge automaton

Next, we show a DHA that accepts the first example in Section 3. Let $M = \langle\{a\}, \{x\}, \{q_0, q_1, q_2\}, \alpha, \iota, q_1?\rangle$, where:

$$
\begin{aligned}
\alpha(a, u) &= \begin{cases} q_1 & (u \in L(q_2^+)), \\ q_0 & (otherwise), \end{cases} \\
\iota(x) &= q_2.
\end{aligned}
$$

Then,

$$L(G) = \{\epsilon, a\langle x\rangle, a\langle xx\rangle, a\langle xxx\rangle, \ldots\}$$

Next, we introduce non-deterministic hedge automata. A *non-deterministic hedge automaton* (NDHA) is $\langle\Sigma, X, Q, \alpha, \iota, F\rangle$, where:

(1) Q, Σ, and F are as specified in the definition of DFA,

(2) α is a relation (called *transition relation*) from $\Sigma \times Q^*$ to Q (or a function from $\Sigma \times Q^*$ to 2^Q) such that for every $q \in Q$ and $x \in \Sigma$, $\{q_1 q_2 \ldots q_k \mid k \geq 0, \alpha(x, q_1 q_2 \ldots q_k, q)\}$ is a regular string language, and

(3) ι is a relation from X to Q (or a function from X to 2^Q).

By definition, a DHA is also a NDHA. We only have to confuse a state and a singleton set containing that state. Thus, the above DHA is also an example of NDHAs.

The last example RHG in Section 3 can be readily captured by a NDHA $\langle\Sigma, X, Q, \alpha, \iota, F\rangle$, where

$$
\begin{aligned}
\Sigma &= \{\texttt{segment}, \texttt{para}\}, \\
X &= \{\texttt{\#PCDATA}\}, \\
Q &= \{q_1, q_2, q_p, q_\#\} \\
\alpha(a, u) &\ni q_1 \; (a = \texttt{segment}, u \in L(q_p^* q_2^*)), \\
\alpha(a, u) &\ni q_2 \; (a = \texttt{segment}, u \in L(q_p^*)), \\
\alpha(a, u) &\ni q_p \; (a = \texttt{para}, u \in L(q_\#)), \\
\iota(x) &= q_\# \; (x = \texttt{\#PCDATA}), \\
F &= \{q_1\}
\end{aligned}
$$

5 Properties of Regular Hedge Languges

5.1 Equivalence

The following conditions are equivalent.

(1) L is generated by a RHG,

(2) L is accepted by a DHA, and

(3) L is accepted by a NDHA.

The proof that (3) implies (2) is done by the subset construction. The rest of the proof is straightforward.

5.2 Boolean closure

Suppose that set L_1 and L_2 are accepted by (N)DHA M_1 and M_2, respectively. We can effectively construct (N)DHAs that accept the following languages.

(1) the intersection of L_1 and L_2,

(2) the union of L_1 and L_2,

(3) the complement of L_1 (the set of all hedges not contained by L_1)

5.3 Parse trees of extended context-free grammars

The set of parse trees of an extended context-free grammar is said to be a *local* tree language. A lot is known about the relationships between local tree languages and regular hedge languagess. We mention two observations which are directly relevant to XML.

(1) A local tree language is a regular hedge language (in other words, for any extenced context-free grammar, we can construct a DHA.), and

(2) For any regular hedge language that contains trees only, there exists a unique minimal local tree language that includes that regular hedge language.

Observation (1) implies that RHGs are more powerful than DTDs, while (2) ensures that given any RHG, we can construct a reasonable DTD.

BIBLIOGRAHICS NOTES

In the theoretical computer science community, regular hedge languages were first studied by Pair et al[PQ68] and Takahashi[Tak75]. Regular hedge language can also be considered as extensions of regular tree languages [Tha87]. We borrow some concepts from these papers but adopt definitions more similar to those for regular string languages.

We define RHG's similarily to [PQ68, Tak75], but we avoid projections. Alternatively, our definition can be considered as a hedge-version of Brainerd's tree regular grammars (called "tree generating regular systems") [Bra69].

Our definitions of NDHAs and DHAs are derived from (non-)deterministic tree automata of [Tha67] except that we have extended them to hedges.

It was Kil-Ho Shin (Fuji Xerox) who first proposed to use regular hedge languages as a formal model for schemata of structured documents. His proposal dates back to November, 1991, but he never published any papers. In search of a formalism for document schemata, HIYAMA Masayuki (FAMILY Given) reached a similar formalism in 1996. Since 1993, the present author has applied regular hedge languages (and hedge monoids, which are outside the scope of this note) for schema transformation [Mur97b, Mur98, Mur97a].

The word "hedge" was originally proposed by Bruno Courcelle [Cou89]. Derick Wood recommended the use of this word, and it has become the standard word in the XML community after a tutorial by Paul Prescod in 1999. For more information, see the special section on hedge automata in the he SGML/XML Web Page (http://www.oasis-open.org/cover/topics.html#forestAutomata).

References

[Bra69] Walter S. Brainerd. Tree generating regular systems. *Information and Control*, 14:217–231, 1969.

[Cou89] Bruno Courcelle. On recognizable sets and tree automata. In Maurice Nivat and Hassan Aït-Kaci, editors, *Resolution of Equations in Algebraic Structures*. Academic Press, 1989.

[Mur97a] Makoto Murata. DTD transformation by patterns and contextual conditions. In *Proceedings of SGML/XML'97 (Also available at http://www.fxis.co.jp/DMS/sgml/xml/sgmlxml97.html and http://www.fxis.co.jp/DMS/sgml/xml/dtd_transformation/index.html)*. GCA, 1997.

[Mur97b] Makoto Murata. Transformation of documents and schemas by patterns and contextual conditions. In *Principles of Document Processing '96*, volume 1293 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[Mur98] Makoto Murata. Data model for document transformation and assembly (extended abstract). In *Principles of Digital Document Processing '98*, volume 1481 of *LNCS*. Springer-Verlag Inc., 1998.

[PQ68] C. Pair and A. Quere. Définition et etude des bilangages réguliers. *Information and Control*, 13(6):565–593, December 1968.

[Tak75] Masako Takahashi. Generalizations of regular sets and their application to a study of context-free languages. *Information and Control*, 27:1–36, 1975.

[Tha67] James W. Thatcher. Characterizing derivation trees of context-free grammars through a generalization of finite automata theory. *Journal of Computer and System Sciences*, 1:317–322, 1967.

[Tha87] James W. Thatcher. Tree automata: An informal survey. In Alfred V. Aho, editor, *Currents in the theory of computing*, pages 143–172. Prentice-Hall, 1987.