Disambiguation of SGML content models *

Helena Ahonen
Department of Computer Science
University of Helsinki
P.O. Box 26 (Teollisuuskatu 23)
FIN-00014 University of Helsinki, Finland
E-mail: helena.ahonen@helsinki.fi
Tel. +358-0-70844218

Abstract

A Standard Generalized Markup Language (SGML) document has a document type definition (DTD) that specifies the allowed structures for the document. The basic components of a DTD are element declarations that contain for each element a content model, i.e., a regular expression that defines the allowed content for this element. The SGML standard requires that the content models of element declarations are unambiguous in the following sense: a content model is ambiguous if an element or character string occurring in the document instance can satisfy more than one primitive token in the content model without look-ahead. Brüggemann-Klein and Wood have studied the unambiguity of content models, and they have presented an algorithm that decides whether a content model is unambiguous. In this paper we present a disambiguation algorithm that, based on the work of Brüggemann-Klein and Wood, transform an ambiguous content model into an unambiguous one by generalizing the language. We also present some experimental results obtained by our implementation of the algorithm in connection to an automatic DTD generation tool.

1 Introduction

A Standard Generalized Markup Language (SGML) [1] document type definition (DTD) specifies the elements that are allowed in a document of this type, and for each element, its *content model*, i.e., either the structure of its content in terms of the other elements, or, for the unstructured parts of text, the type of data that can occur in its content. Document types in SGML are defined by context-free grammars, while the content models, i.e., the right-hand sides of the productions, are regular expressions. The SGML standard requires that the content models have to be unambiguous in the following sense. A content model is ambiguous if an element or character string occurring in the document instance can satisfy more than one primitive token in the content model without look-ahead.

For instance, if we have the following element declaration and part of an document instance, it is impossible to say whether 12 is an instance of the first Infl_index or the second.

*This work was partially supported by the Academy of Finland and TEKES.

```
<!ELEMENT Inflections  - - (Infl_index?, Infl_index) >

     <Inflections>
     <Infl_index>12</Infl_index>
     </Inflections>
```

There can also be ambiguity in choosing between alternatives. In the following, when seeing

```
      <Headword>tact
```

we cannot say which alternative should be chosen.

```
<!ELEMENT Entry   - - ((Headword, Sense) | (Headword, Example)) >

     <Entry>
     <Headword>tact</Headword>
     <Example>Phil had the tact to leave a moment's
              respectful silence.</Example>
     </Entry>
```

Iterations are still another cause of ambiguity:

```
<!ELEMENT Senses  - - ((Sense, Example)*, Sense) >

     <Senses>
     <Sense>Extremely unpleasant.</Sense>
     <Example>She would never harm an insect, however noxious</Example>
     <Sense>A noxious gas or substance is harmful or poisonous.</Sense>
     </Senses>
```

Brüggemann-Klein and Wood [2, 3] have presented an algorithm that can decide whether a content model is unambiguous. We have developed their ideas further and present in this paper a disambiguation algorithm that transforms an ambiguous content model into an unambiguous one. The resulting content model generalizes the original, i.e., accepts more element structures. We have implemented the disambiguation algorithm as a part of our automatic DTD generation tool [4, 5]. The tool generates DTDs by generalizing tagged document instances. For each element it forms a deterministic finite automaton that first accepts the sample instances only. The automaton is then generalized until it fulfills certain constraints. In the next step the automaton is disambiguated, and finally it is converted into a regular expression, i.e., a content model. Since the tool never creates content models containing &-operators, our disambiguation algorithm cannot disambiguate content models containing them.

The next section outlines the decision algorithm of Brüggemann-Klein and Wood and the basic concepts needed. In Section 3 we present the disambiguation algorithm for automata, and in Section 4 the conversion into a content model. Section 5 gives some experimental results.

2 1-unambiguity

Brüggemann-Klein and Wood [3] call the unambiguity required in the SGML standard 1-unambiguity and give a definition for it in terms of the pairs of positions that follow each other in a word. Here a 'word' denotes a string of elements, and a 'language' is a set of 'words'. First we have to define the following sets:

Definition 1 Let Σ be a set of elements. For $L \subset \Sigma^*$, let

- $\text{first}(L) = \{a \in \Sigma \mid aw \text{ is in } L \text{ for some word } w\}$,
- $\text{last}(L) = \{a \in \Sigma \mid wa \text{ is in } L \text{ for some word } w\}$,
- $\text{followlast}(L) = \{a \in \Sigma \mid vaw \text{ is in } L, \text{ for some word } v \text{ in } L \setminus \{\epsilon\} \text{ and some word } w\}$.

Furthermore, we extend the definitions of the sets above to expressions E by defining $\text{first}(E) = \text{first}(L(E))$, for each expression E, and similarly for the other sets.

Lemma 2 1-unambiguity of a regular expression E can be defined inductively:

- $E = \emptyset$, $E = \epsilon$, or $E = a$, with $a \in \Sigma$: E is 1-unambiguous.
- $E = F \mid G$: E is 1-unambiguous if and only if F and G are 1-unambiguous, and $\text{first}(F) \cap \text{first}(G) = \emptyset$
- $E = FG$: If $L(E) = \emptyset$, then E is 1-unambiguous. If $L(E) \neq \emptyset$ and $\epsilon \in L(F)$, then E is 1-unambiguous if and only if F and G are 1-unambiguous, $\text{first}(F) \cap \text{first}(G) = \emptyset$, and $\text{followlast}(F) \cap \text{first}(G) = \emptyset$. If $L(E) \neq \emptyset$ and $\epsilon \notin L(F)$, then E is 1-unambiguous if and only if F and G are 1-unambiguous and $\text{followlast}(F) \cap \text{first}(G) = \emptyset$.
- $E = F^*$: E is 1-unambiguous if and only if F is 1-unambiguous and $\text{followlast}(F) \cap \text{first}(F) = \emptyset$.

A regular language is 1-unambiguous if it is denoted by some 1-unambiguous expression. Brüggemann-Klein and Wood characterize the set of 1-unambiguous regular languages in terms of structural properties of the minimal deterministic automata that recognize them and show how a 1-unambiguous expression can be constructed from a 1-unambiguous deterministic automaton. In the following, we consider first the structural properties of 1-unambiguous automata.

Major causes of ambiguities are iterations, i.e., cycles in automata. Therefore, we have to consider the strongly connected components, so-called *orbits*, of an automaton.

Definition 3 Let $M = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton, where Q is the set of *states*, Σ is the set of elements, $\delta : Q \times \Sigma^* \to Q$ is the *transition function*, $q_0 \in Q$ is the *initial state* and $F \subseteq Q$ is the set of *final states*. For $q \in Q$, the strongly connected component of q, i.e., the states of M that can be reached from q and from which q can be reached as well, is called the *orbit* of q and denoted by $O(q)$. We consider the orbit of q to be *trivial* if $O(q) = \{q\}$ and there are no transitions from q to itself in M.

Definition 4 A state q in a finite automaton M is called a *gate* of its orbit if either q is a final state or if there are $q' \in Q \setminus O(q)$ and $a \in \Sigma$ with $\delta(q, a) = q'$. The finite automaton M has the *orbit property* if each orbit of M is homogenous with respect to its gates, i.e., if, for all gates q_1 and q_2 with $O(q_1) = O(q_2)$, we have

- q_1 is a final state if and only if q_2 is a final state.
- $\delta(q_1, a) = q'$ if and only if $\delta(q_2, a) = q'$, for all $q' \in Q \setminus O(q_1) = Q \setminus O(q_2)$, and for all $a \in \Sigma$.

Example 5 The automaton in Figure 1 has four orbits, $\{1\}$, $\{2\}$, $\{3, 4, 5\}$, and $\{6\}$. The automaton does not have the orbit property, since the gates of the third orbit, states 3 and 4, are not homogenous. The state 3 is a final state, while the state 4 is not, and there is a R-transition from the state 4 but none from the state 3. □

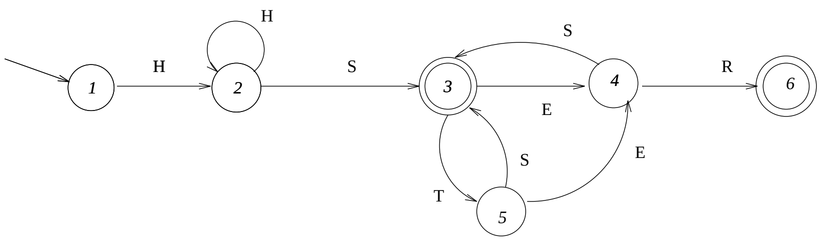

Figure 1: Automaton with four orbits.

Definition 6 For a state q of a finite automaton M, let the *orbit automaton* M_q of q be the automaton obtained by restricting the state set of M to $O(q)$ with initial state q and with the gates of $O(q)$ as the final states of M_q. The language of M_q is called the *orbit language* of q. The languages $L(M_q)$ are also called the orbit languages of M. We also consider a larger subautomaton of M related to q: the finite automaton M^q is M with its state set restricted to the states reachable from q and with q as the initial state.

Example 7 In the automaton M in Figure 1, the orbit automaton M_5 contains the states 3, 4, and 5, with the state 5 as the initial state and the states 3 and 4 as the final states, whereas the automaton M^5 contains the states 3, 4, 5, and 6, with the state 5 as the initial state and the states 3 and 6 as the final states. □

Definition 8 For a deterministic finite automaton M, a symbol a in Σ is *M-consistent* if there is a state $f(a)$ in M such that all final states of M have an a-transition to $f(a)$. A set S of symbols is M-consistent if each symbol in S is M-consistent.

Definition 9 Let M be a finite automaton and S be a set of symbols. The *S-cut* M_S of M is constructed from M by removing, for each $a \in S$, all a-transitions that leave a final state of M.

Now we can define the structural properties of a 1-unambiguous deterministic automaton.

Theorem 10 Let M be a minimal deterministic finite automaton and S be an M-consistent set of symbols. Then, $L(M)$ is 1-unambiguous if and only if

1. M_S satisfies the orbit property and
2. All orbit languages of M_S are 1-unambiguous.

Furthermore, if M consists of a single, nontrivial orbit, and $L(M)$ is 1-unambiguous, M has at least one M-consistent symbol.

Algorithm 11 1-unambiguous; the decision algorithm for 1-unambiguity of the language denoted by a minimal deterministic finite automaton M.

1. Compute $S := \{a \in \Sigma \mid a \text{ is } M\text{-consistent}\}$
2. **if** M has a single, trivial orbit **then** return true
3. **if** M has a single, nontrivial orbit **and** $S = \emptyset$
   then return false
4. Compute the orbits of M_S
5. **if not** OrbitProperty(M_S) **then** return false
6. **for** each orbit K of M_S **do**
   Choose $x \in K$
   if not 1-unambiguous($(M_S)_x$) **then** return false
7. Return true

Example 12 When deciding the unambiguity of the automaton M in Figure 2 the set of M-consistent symbols is in the beginning empty. Since there are several orbits, we compute the orbits of M. Note that $M_S = M$, if S is empty. The automaton fulfills the orbit property. Thus, we check recursively the unambiguity of each of the orbit automata.

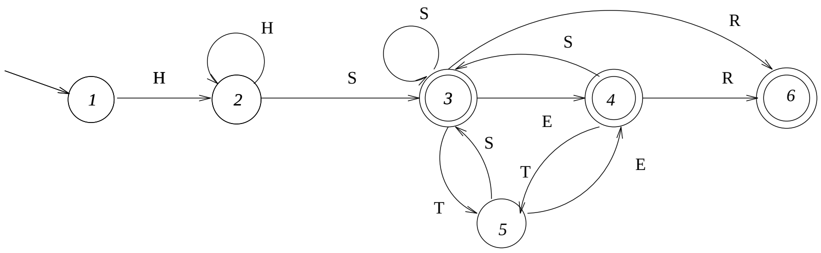

Figure 2: Unambiguous automaton.

The orbits $\{1\}$ and $\{6\}$ are trivial, and hence 1-unambiguous. The orbit automaton M_2 has one consistent symbol, namely H, and the $\{H\}$-cut of the orbit automaton is trivial. Hence, the orbit automaton M_2 is 1-unambiguous.

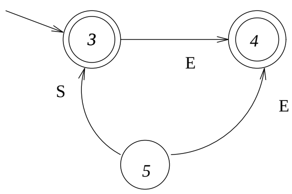

Figure 3: $M_{\{T,S\}}$, consistent symbols have been removed.

The orbit automaton $M_3 = M_4 = M_5$ has two consistent symbols, S and T. In Figure 3 we can see the orbit automaton when the S- and T-transitions have been removed, i.e., the $\{T, S\}$-cut. Now we have three trivial orbits, and hence, the whole automaton is 1-unambiguous. □

3 Disambiguation

If the language of a content model is not 1-unambiguous, we have to disambiguate the corresponding automaton. Disambiguation generalizes the language, i.e., the resulting content model accepts more element structures than the original one. Hence, all the existing documents are still valid.

Algorithm 11 can be modified so that the automaton is transformed into a 1-unambiguous automaton if it is not 1-unambiguous originally. There are two reasons why a 1-unambiguity test fails: first, we have a single, nontrivial orbit automaton M, but there is no M-consistent symbols, and second, the automaton does not have the orbit property. The modified algorithm is the following. Contrary to the previous algorithm we allow the automaton to be temporarily non-deterministic, i.e., the transition function δ is defined as $\delta : Q \times \Sigma \rightarrow Q^*$.

Algorithm 13 Disambiguate; the transformation algorithm for 1-unambiguity of the language denoted by a minimal deterministic finite automaton M.

1. Compute $S := \{a \in \Sigma \mid a \text{ is } M\text{-consistent}\}$
2. **if** M has a single, trivial orbit **then** exit
3. **if** M has a single, nontrivial orbit **and** $S = \emptyset$

then
 Choose some symbol a such that $f(a) \in \delta(q, a)$,
 with some final state q, and some state $f(a)$ of M.
 for each final state q' of M
  if $q'' \in \delta(q', a)$, with $q'' \neq f(a)$ and $q'' \in M$
  then merge states $f(a)$ and q''
  else $\delta(q', a) := \delta(q', a) \cup \{f(a)\}$
 $S := \{a\}$

4. Compute the orbits of M_S
5. ForceOrbitProperty(M_S)
6. **for** each orbit K of M_S **do**
 Choose $x \in K$
 Disambiguate($(M_S)_x$)

How to choose a consistent symbol? We have preferred a symbol a such that there are a maximum amount of final states q' such that $f(a) \in \delta(q', a)$, but any symbol leaving a final state is a possible candidate. A good choice is also a symbol that leaves all the final states but not to the same state. The choice of the consistent symbol affects the result of the disambiguation.

Algorithm 14 ForceOrbitProperty.

1. **for** each orbit K of M
2. Let $g_1, \ldots g_k$ be the gates of K.
3. **if** there exists a gate $g_i \in F$
  then for each g_i
   $F := F \cup \{g_i\}$
4. **for** each ordered pair of gates (g_i, g_j)
5.   **if** there exists a symbol a such that $q \in \delta(g_i, a)$,
    with $q \in Q \backslash O(g_i)$, and $q \notin \delta(g_j, a)$
   then
    if $q' \in \delta(q_j, a)$, s.t. $q' \neq q$
    then $\delta(g_j, a) := \delta(g_j, a) \cup \{q\}$

Example 15 The automaton M in Figure 4 does not have the orbit property, since in the orbit automaton M_3 the gate 4 is non-final while the gate 3 is final, and additionally, both states have a transition labelled R, but not to the same state. Hence, the algorithm first makes the gate 4 final and adds new R-transitions from the state 3 to the state 6 and from the state 4 to the state 7.

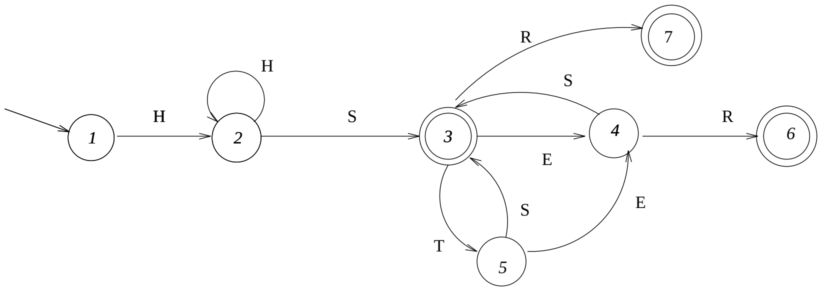

Figure 4: Ambiguous automaton.

All the other orbit automata are 1-unambiguous except the orbit automaton M_3, since there are no M_3-consistent symbols. We choose T to be consistent, add a transition from the state 4 to the state 5, and construct a {T}-cut by removing T-transitions (Figure 5).

Figure 5: a) T is consistent. b) $\{T\}$-cut.

The automaton still has one non-trivial orbit that is not 1-unambiguous. We choose S to be consistent, and now the {S}-cut has only trivial orbits.

Figure 6: a) S is consistent. b) $\{S\}$-cut.

Disambiguation of an orbit automaton may cause non-determinism in the containing automaton. Hence, after disambiguation we have to merge the states causing non-determinism and repeat the disambiguation of the whole automaton until the automaton is deterministic. In Example 15 we merge the states 6 and 7, after which the next disambiguation step does not alter the automaton. The resulting 1-unambiguous automaton can be seen in Figure 2. The algorithm converges: if the automaton is not 1-unambiguous, the algorithm either adds new arcs or merges states. Since both an automaton with one state only and a totally connected automaton are 1-unambiguous, the algorithm stops in these situations at last.

4 Conversion into a content model

If an automaton accepts a 1-unambiguous language, it is possible to construct a corresponding 1-unambiguous regular expression.

We can represent the interconnections among the orbits by constructing a reduced automaton for M. The states of the reduced automaton are the orbits of M. There is a transition from state C to a different state C' of the reduced automaton if there is a transition in M from some state in the orbit C to some state in the orbit C'. The reduced automaton cannot have any cycles, because if there were a cycle, then all the orbits in the cycle would be one orbit, meaning that the orbits were not properly computed.

Now we can construct the expression for the automaton M by constructing the expression for the reduced automaton. Brüggemann-Klein and Wood [3] give the following method. First we assume that the 1-unambiguous regular expressions for the orbits can be constructed.

We assume that M has more than one orbit and consider the orbit $O(q_0)$ of the initial state q_0. Let $b_1, \ldots, b_n$ be the distinct symbols of the transitions that leave $O(q_0)$. Since M satisfies the orbit property, there are states $q_1, \ldots, q_n$ outside $O(q_0)$ such that all gates of $O(q_0)$ have a b_i-transition to q_i, and there are no other outgoing transitions from $O(q_0)$ to the outside. Since M is deterministic, M_{q_0} has no b_i-transition from a final state. If E_0 and E^i are 1-unambiguous expressions that denote the languages $L(M_{q_0})$ and $L(M^{q_i})$, respectively, then the 1-unambiguous expression

$$E = E_0(b_i E^i \mid \ldots \mid b_n E^n)$$

denotes $L(M)$.

Now we will show how the regular expressions for the orbits can be constructed. Clearly a regular expression for a trivial orbit is an empty string ϵ. For a single, nontrivial orbit automaton M, Brüggemann-Klein and Wood give the following construction method. If E_S and $E_S^{f(a_i)}$ are 1-unambiguous expressions denoting $L(M_S)$ and $L(M_S^{f(a_i)})$, where $S = \{a_1 \ldots a_k\}$ are the consistent symbols, then the 1-unambiguous expression $E_S\ (a_1 E_S^{f(a_1)} \mid \ldots \mid a_k E_S^{f(a_k)})^*$ denotes $L(M)$.

Example 16 Let E be a regular expression denoting the language $L(M)$ of M, and let E_i, E^i, and E_S^i be regular expressions denoting the languages $L(M_{q_i})$, $L(M^{q_i})$, and $L(M_S^{q_i})$, respectively, with $q_i \in Q$, and $S \subset \Sigma$. Now we can construct a 1-unambiguous expression for the automaton in Figure 2 in the following way.

$$
\begin{aligned}
E &= E_1,\ h,\ E^2 \\
E_1 &= \epsilon \\
E^2 &= E_2,\ s,\ E^3 \\
E_2 &= \epsilon,\ (h,\ \epsilon)^* = h^* \\
E^3 &= E_3\ (\epsilon \mid r,\ E^6) \\
E_3 &= e?\ (s,\ E^3_{\{s,t\}} \mid t,\ E^5_{\{s,t\}})^* \\
E^3_{\{s,t\}} &= e? \\
E^5_{\{s,t\}} &= (s,\ (e \mid \epsilon)\ \mid e) = (s,\ e? \mid e) \\
E^6 &= \epsilon
\end{aligned}
$$

and finally

$$
\begin{aligned}
E_3 &= e?,\ (s,\ e?\ \mid t,\ (s,\ e? \mid e))^* \\
E &= h,\ h^*,\ s,\ (e?,\ (s,\ e?\ \mid t,\ (s,\ e? \mid e))^*),\ r?
\end{aligned}
$$

□

5 Experimental results

We have implemented the presented disambiguation algorithm as a part of our automatic DTD generation program, which generates a DTD from tagged instances. We have experimented with several document types, e.g., a Finnish dictionary and an engineering textbook. The next element declaration demonstrates an unambiguous content model for a dictionary entry:

```
<!ELEMENT EN  - -  (HP,  ( S, ( ( TF, ( EX | S,  EX?  )?  | EX )
                              (TF, (S, EX? | EX )? )*, (R | S )? )?
                         | EX, (TF, ( EX | S, EX? )?
                              (TF, (EX | S, EX? )? )*, (S | R)? )?
                         | TF, (( EX | S, EX?  | TF, (S, EX? | EX )? )
                              (TF, (S, EX? | EX )? )*, (S | R)? )?
                         | R, EX?  | PrF | BW, EX
                         | PaF, S, (( TF, (EX | S, EX? )? | EX ),
                              (TF, (S, EX? | EX )? )*, (R | S )? )? )? ) >
```

6 Conclusions

We have presented a disambiguation algorithm for SGML content models, based on the work of Brüggemann-Klein and Wood. We have used the algorithm as a part of an automatic DTD generating tool that forms content models from tagged documents, but the algorithm could be used to transform existing DTDs as well.

The idea of the algorithm is the following. First, minimal deterministic finite automata corresponding the content models of each element are constructed. The strongly connected components, orbits, of each automaton are then generalized to conform the 1-unambiguity constraints given by Brüggemann-Klein and Wood, e.g., to obtain the orbit property. Finally, the automata are converted into 1-unambiguous content models.

References

[1] Information Processing – Text and Office Systems – Standard Generalized Markup Language (SGML). Technical Report ISO/IEC 8879, International Organization for Standardization ISO/IEC, Geneva/New York, 1986.

[2] Anne Brüggemann-Klein and Derick Wood. Deterministic regular languages. In A. Finkel and M. Jantzen, editors, *STACS '92, Proceedings of the 9th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science 577, pages 173–184. Springer–Verlag, 1992.

[3] Anne Brüggemann-Klein and Derick Wood. One-unambiguous regular languages. Technical report, Institut für Informatik, Universität Freiburg, May 1994. Accessible at URL: http://www.informatik.uni-freiburg.de/Personalia/Brueggemann-Klein.html.

[4] Helena Ahonen, Heikki Mannila, and Erja Nikunen. Generating grammars for SGML tagged texts lacking DTD. In M. Murata and H. Gallaire, editors, *Proceedings of the Workshop on Principles of Document Processing '94. Also to appear in Mathematical and Computer Modelling.*, 1994.

[5] Helena Ahonen. Generating grammars for structured documents using grammatical inference methods. PhD Thesis, Department of Computer Science, University of Helsinki, 1996. In preparation.