

# Exercises for Language Processors (WS2001-2002) \*

Prof. Dr. Peter Thiemann  
Simon Helsen

## 5 Model Solution

### 5.1 Validation of SGML

We need the following attributes:

- Synthesised attributes:  $\{value, valid\}$ . The attribute *value* says in what element we currently reside and propagates this upwards. The attribute *valid* contains the information whether we have a valid parse or not.
- Inherited attributes:  $\{regexp, drink\}$ . The attribute *regexp* is propagated down the parse tree and contains the regular expression required to check if the sequence of elements is an element of the language. We need attribute *drink* to keep track of whether we can have drinks or not.

---

\*©2002: it is not permitted to lend, distribute or copy any of these exercises and their model solutions either by paper or in electronic form other than for personal use. The authors cannot be held responsible for mistakes or inaccuracies in either formulation or solution of the exercises.

The attribute grammar for validating SGML-documents makes use of the function  $\mathcal{E}$  and  $\mathcal{D}$  from the lecture:

$E \rightarrow \langle \text{lunch} \rangle L \langle / \text{lunch} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{lunch}}$ $L.regexp = \underline{\text{meal}} \text{ meal}^*$ $L.drink = \text{false}$
$E \rightarrow \langle \text{meal} \rangle L \langle / \text{meal} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{meal}}$ $L.drink = \text{true}$ $L.regexp = (\underline{\text{appetiz}} \mid \epsilon) \underline{\text{steak}} \underline{\text{custname}}$
$E \rightarrow \langle \text{appetiz} \rangle L \langle / \text{appetiz} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{appetiz}}$ $L.regexp = \underline{\text{soup}} \mid \underline{\text{salad}}$ $L.drink = E.drink$
$E \rightarrow \langle \text{soup} \rangle L \langle / \text{soup} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{soup}}$ $L.regexp = \epsilon$ $L.drink = E.drink$
$E \rightarrow \langle \text{salad} \rangle L \langle / \text{salad} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{salad}}$ $L.regexp = \epsilon$ $L.drink = E.drink$
$E \rightarrow \langle \text{steak} \rangle L \langle / \text{steak} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{steak}}$ $L.regexp = \epsilon$ $L.drink = E.drink$
$E \rightarrow \langle \text{drink} \rangle L \langle / \text{drink} \rangle$	$E.valid = L.valid \wedge E.drink$ $E.value = \epsilon$ $L.regexp = (\underline{\text{beer}} \mid \underline{\text{cola}})$ $L.drink = E.drink$
$E \rightarrow \langle \text{cola} \rangle L \langle / \text{cola} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{cola}}$ $L.regexp = \epsilon$ $L.drink = E.drink$
$E \rightarrow \langle \text{beer} \rangle L \langle / \text{beer} \rangle$	$E.valid = L.valid$ $E.value = \underline{\text{beer}}$ $L.regexp = \epsilon$ $L.drink = E.drink$
$E \rightarrow \langle \text{custname} \rangle n \langle / \text{custname} \rangle$	$E.\underline{\text{custname}}$ $E.valid = \text{true}$
$L \rightarrow \epsilon$	$L.valid = \epsilon \in \text{Lan}(\mathcal{E}(L.regexp))$
$L \rightarrow EL$	$L_1.regexp = \mathcal{D}(L.regexp, E.value)$ $L.valid = L_1.valid \wedge E.valid$ $E.drink = L.drink$

## 5.2 LAG(k) Attributed Grammars

1. Consider the following attribute grammar with  $\mathcal{A} = \{a, b, c, d\}$ :

$$\begin{array}{llll}
 \pi_1 : Z \rightarrow X & X.b = 1 & \pi_3 : X \rightarrow s & X.a = X.b \\
 \pi_2 : X \rightarrow WX & X.a = W.c & \pi_4 : W \rightarrow t & W.c = W.d \\
 & X_1.b = X.b & & \\
 & W.d = X_1.a & & 
 \end{array}$$

Does there exist a  $k$  which makes the grammar LAG(k)? No:

- First, we check for LAG(1):

$$\begin{array}{ll}
 \pi_1 : & \pi_2 : j = \epsilon; j = 2 > i = \epsilon; \mathbf{j} = \mathbf{1} \leq \mathbf{i} = \mathbf{2} \\
 \pi_3 : j = \epsilon & \pi_4 : j = \epsilon
 \end{array}$$

So, only one attribution for  $\pi_2$  is causing problems: the attribute  $a$  is calculated too *late*.

- Hence, we have to calculate  $a$  in a pass before  $d$ . Unfortunately, we have to calculate  $c$  before  $a$  and  $d$  before  $c$ . This makes a cycle, ie. in an LAG(k) grammar, both  $a$  and  $d$  have to be calculated in the same pass, but as we saw in the previous point, this is not possible!

2. Consider the following attribute grammar with  $A = \{a, b, c, d, e, f, g\}$ :

$$\begin{array}{llll}
 \pi_1 : Z \rightarrow X & X.b = 1 & \pi_2 : X \rightarrow WXY & X.a = W.c + X_1.a \\
 \pi_3 : X \rightarrow s & X.a = X.b & & X.e = Y.g \\
 & X.e = X.b & & X_1.b = X.b \\
 \pi_4 : W \rightarrow t & W.c = W.d & & W.d = X_1.e \\
 \pi_5 : Y \rightarrow u & Y.g = Y.f & & Y.f = X_1.e
 \end{array}$$

Does there exist a  $k$  which makes the grammar LAG(k)? Yes, we can take  $k = 2$ .

- First, we check for LAG(1):

$$\begin{array}{ll}
 \pi_1 : & \pi_2 : j = \epsilon; j = \epsilon; j = 2 > i = \epsilon; \\
 & \mathbf{j} = \mathbf{1} \leq \mathbf{i} = \mathbf{2}; j = 3 > i = 2 \\
 \pi_3 : j = \epsilon; j = \epsilon & \pi_4 : j = \epsilon \\
 \pi_5 = \epsilon & 
 \end{array}$$

So, only one attribution for  $\pi_2$  is causing problems: the attribute  $e$  is calculated too late.

- The following partition solves the above dependency:  $A_1 = \{e, g, f, b\}$

and  $A_2 = \{a, c, d\}$ :

$\pi_1 :$

$\pi_2 :$   $u = 2 = v = 2$  and  $j = \epsilon$ ;  
 $u = 1 = v = 1$  and  $j = \epsilon$ ;  
 $u = 1 = v = 1$  and  $j = 2 > i = \epsilon$ ;  
 $u = 1 < v = 2$ ;  
 $u = 2 = v = 2$  and  $j = 3 > i = 2$

$\pi_4 : u = 2 = v = 2$  and  $j = \epsilon$

$\pi_3 :$   $u = 1 < v = 2$  and  $j = \epsilon$ ;  
 $u = 2 = v = 2$  and  $j = \epsilon$

$\pi_5 : u = 1 = v = 1$  and  $j = \epsilon$