

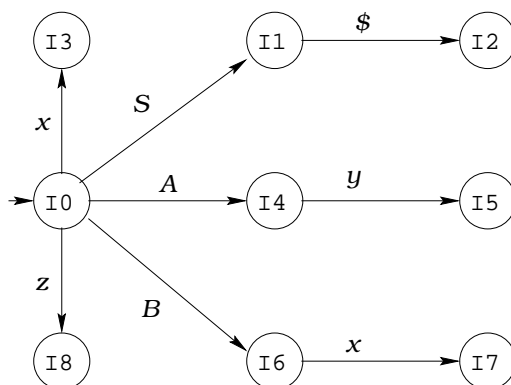
Tutorial 2: Solutions

CSC 467 Compilers and Interpreters
Fall Semester, 2005

1. (a) The $LR(1)$ items are:

$I_0: S' \rightarrow \cdot S \$, \$ \epsilon$	$I_2: S' \rightarrow S \$ \cdot, \$ \epsilon$
$S \rightarrow \cdot x, \$$	$I_3: S \rightarrow x \cdot, \$$
$S \rightarrow \cdot A y, \$$	$I_4: S \rightarrow A \cdot y, \$$
$A \rightarrow \cdot B x, y$	$I_5: S \rightarrow A y \cdot, \$$
$B \rightarrow \cdot, x$	$I_6: A \rightarrow B \cdot x, y$
$B \rightarrow \cdot z, x$	$I_7: A \rightarrow B x \cdot, y$
$I_1: S' \rightarrow S \cdot \$, \$ \epsilon$	$I_8: B \rightarrow z \cdot, x$

and the *goto* graph:

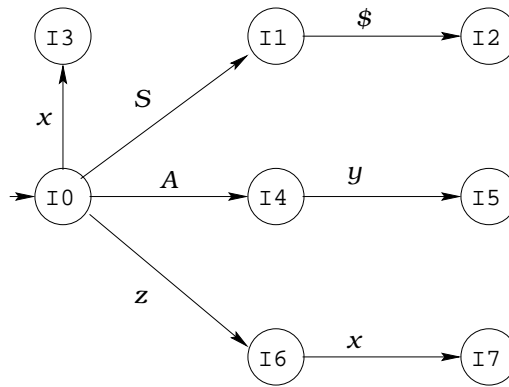


(b) There is a shift/reduce conflict in I_0 between items $S \rightarrow \cdot x, \$$ and $B \rightarrow \cdot, x$ so G_1 in neither $LR(0)$ nor $LR(1)$.

(c) The $LR(1)$ items are:

$I_0: S' \rightarrow \cdot S \$, \$ \epsilon$	$I_3: S \rightarrow x \cdot, \$$
$S \rightarrow \cdot x, \$$	$A \rightarrow x \cdot, y$
$S \rightarrow \cdot A y, \$$	$I_4: S \rightarrow A \cdot y, \$$
$A \rightarrow \cdot x, y$	$I_5: S \rightarrow A y \cdot, \$$
$A \rightarrow \cdot z x, y$	$I_6: A \rightarrow z \cdot x, y$
$I_1: S' \rightarrow S \cdot \$, \$ \epsilon$	$I_7: A \rightarrow z x \cdot, y$
$I_2: S' \rightarrow S \$ \cdot, \$ \epsilon$	

and the *goto* graph:



- (d) There are neither $LR(1)$ nor $LR(0)$ conflicts. Looking at I_3 , $LR(0)$ items $S \rightarrow x \cdot$ and $A \rightarrow x \cdot$ correspond to the actions $action[3, \$] = reduce$ “ $S \rightarrow x$ ” and $action[3, y] = reduce$ “ $A \rightarrow x$ ” respectively, since $FOLLOW(S) = \{\$ \}$ and $FOLLOW(A) = \{y \}$ (this is not a conflict). So, G_2 is both $LR(0)$ and $LR(1)$.
- (e) Note that $L(G_1)$ only consists of the strings $x\$, xy\$,$ and $zxy\$$. Any finite language is trivially regular, therefore $L(G_1)$ is regular. We also observe that an “easy to express” language, such as $L(G_1)$, which can be recognized by a finite automaton is written in a “hard to parse” grammar, such as G_1 .

2. A possible translation scheme is:

```

ident_list → ident { ident_list.l = CreateList(ident.x) }
             | ident_list_1 TCOMMA ident
             { if (Find(ident.x, ident_list_1.l)
                 then yyerror("Duplicate identifier")
                 else Insert(ident.x, ident_list_1.l) }
  
```

3. (a) No markers are needed now. The semantic rules are:

Production	Semantic Rules
$S \rightarrow L$	$L.ps = 10$ $S.ht = L.ht$
$L \rightarrow L_1 B$	$L_1.ps = L.ps$ $B.ps = L.ps$ $L.ht = \max(L_1.ht, B.ht)$
$L \rightarrow B$	$B.ps = L.ps$ $L.ht = B.ht$
$B \rightarrow B_1 \text{ sub } F$	$B_1.ps = B.ps$ $F.ps = \text{shrink}(B_1.ps)$ $B.ht = \text{disp}(B_1.ht, F.ht)$
$B \rightarrow F$	$F.ps = B.ps$ $B.ht = F.ht$
$F \rightarrow \{ L \}$	$L.ps = F.ps$ $F.ht = L.ht$
$F \rightarrow \text{text}$	$F.ht = \text{text.ht} * F.ps$

(b) and the translation scheme constructed from the table in part (a) is:

S	→		{ L.ps = 10 }
		L	{ S.ht = L.ht }
L	→		{ L ₁ .ps = L.ps }
		L ₁	{ B.ps = L.ps }
		B	{ L.ht = max(L ₁ .ht, B.ht) }
L	→		{ B.ps = L.ps }
		B	{ L.ht = B.ht }
B	→		{ B ₁ .ps = B.ps }
		B ₁	
		sub	{ F.ps = shrink(B ₁ .ps) }
		F	{ B.ht = disp(B ₁ .ht, F.ht) }
B	→		{ F.ps = B.ps }
		F	{ B.ht = F.ht }
F	→	{	{ L.ps = F.ps }
		L }	{ F.ht = L.ht }
F	→	text	{ F.ht = text.ht * F.ps }