ΕΠΛ323 - Θεωρία και Πρακτική Μεταγλωττιστών

Lecture 9a

Syntax-directed Translation

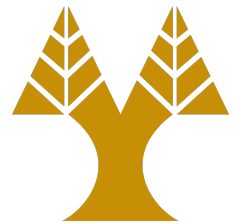
Elias Athanasopoulos
eliasathan@cs.ucy.ac.cy

S-attributed Definition

PRODUCTION	SEMANTIC RULES
$L \rightarrow E$ **n**	*print*(*E.val*)
$E \rightarrow E_1 + T$	*E.val*:=E_1*.val*+*T.val*
$E \rightarrow T$	*E.val*:=*T.val*
$E \rightarrow E_1 * T$	*E.val*:=E_1*.val***T.val*
$T \rightarrow F$	*T.val*:=*F.val*
$F \rightarrow (E)$	*F.val*:=*E.val*
$F \rightarrow$ **digit**	*F.val*:=**digit**.*lexval*

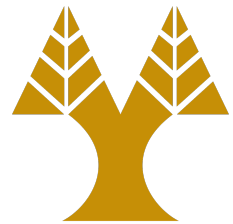
Only synthesized attributes.

Translation Scheme (Bottom-Up)



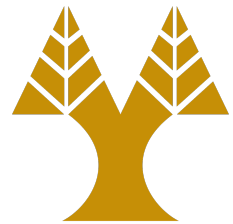
PRODUCTION	CODE FRAGMENT
$L \rightarrow E$ **n**	*print*(*val*[*top*])
$E \rightarrow E_1 + T$	*val*[*ntop*]:=*val*[*top*−2]+*val*[*top*]
$E \rightarrow T$	
$E \rightarrow E_1 * T$	*val*[*ntop*]:=*val*[*top*−2]*val*[*top*]
$T \rightarrow F$	
$F \rightarrow (E)$	*val*[*ntop*]:=*val*[*top*−1]
$F \rightarrow$ **digit**	

Recall
(SLR Parsing Table)



STATE	*action*						*goto*		
	id	+	*	(	)	$	*E*	*T*	*F*
0	s5			s4			1	2	3
1		s6				**acc**			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

INPUTE	*state*	*value*	ACTION
3*5+4**n**	(1) 0	_	
*5+4**n**	(2) 0 **3** 5	3	shift 5
*5+4**n**	(3) 0 *F* 3	3	*F*→**digit**
*5+4**n**	(4) 0 *T* 2	3	*T*→*F*
5+4**n**	(5) 0 *T* 2 * 7	3 _	shift 7
+4**n**	(6) 0 *T* 2 * 7 **5** 5	3 _ 5	shift 5
+4**n**	(7) 0 *T* 2 * 7 *F* 10	3 _ 5	*F*→**digit**
+4**n**	(8) 0 *T* 2	15	*T*→*T***F*
+4**n**	(9) 0 *E* 1	15	*E*→*T*
4**n**	(10) 0 *E* 1 + 6	15_	shift 6
n	(11) 0 *E* 1 + 6 **4** 5	15 _ 4	shift 5
n	(12) 0 *E* 1 + 6 *F* 3	15 _ 4	*F*→**digit**
n	(13) 0 *E* 1 + 6 *T* 9	15 _ 4	*T*→*F*
n	(14) 0 *E* 1	19	*E*→*E*+*T*
	(15) 0 *E* 1 **n**	19 _	shift
	(16) 0 *L*	19	*L*→*E* **n**



Productions

(1) *E* →*E*+*T*
(2) *E* →*T*
(3) *T* →*T***F*
(4) *T* →*F*
(5) *F* →(E)
(6) *F* →**digit**

When a production with *r* symbols on the right side is reduced, the value of *ntop* is set to *top* – *r* + *1*.

L-attributed Definition

- An inherited attribute for a symbol on the right side of a production must be computed in an action before that symbol.
- An action must not refer to a synthesized attribute of a symbol to the right of the action.
- A synthesized attribute for the nonterminal on the left can only be computed after all attributes it references have been computed. The action computing such attributes can usually be placed at the end of the right side of the production.

Example

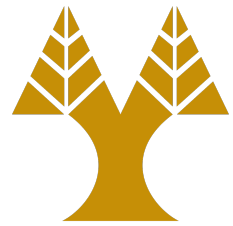
```
S→A1A2 { A1.in := 1; A2.in := 2 }
A→a    { print(A.in) }
```

X

```
S→{ A1.in := 1 } A1 {A2.in := 2 } A2
A→a { print(A.in) }
```

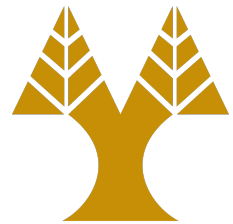
✓

Emitting Translation while Traversing a Parse Tree



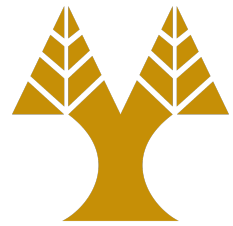
```
procedure dfvisit(n: node);
begin
   for each child m of n (left to right)
      do begin
         evaluate inherited attributes of m;
         dfvisit(m)
      end
      evaluate synthesized attributes of n
end
```

Translation Scheme (Top Down)



```
E → T R
R → addop T  { print(addop.lexeme) } R1 | ε
T → num { print(num.val) }
```

Left-recursion Elimination (with semantic rules)



$A \rightarrow Aa \mid b$

$A \rightarrow A_1 Y \ \{\ A.a := g(A_1.a,\ Y.y)\ \}$

$A \rightarrow X \ \{\ A.a := f(X.x)\ \}$

$A \rightarrow bR,\ R \rightarrow aR \mid \varepsilon$

$A \rightarrow X \ \{\ R.i := f(X.x)\}$

$R\ \{\ A.a := R.s\ \}$

$R \rightarrow Y \ \{\ R_1.i := g(R.i,\ Y.y)\}$

$R_1\{\ R.s := R_1.s\ \}$

$R \rightarrow \varepsilon\{\ R.s := R.i\ \}$

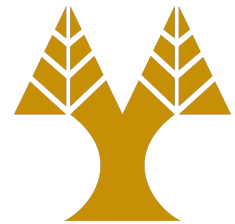
Example (Left Recursion)

$E \rightarrow E_1$ + T	{ $E.val := E_1.val + T.val$ }
$E \rightarrow E_1$ - T	{ $E.val := E_1.val - T.val$ }
$E \rightarrow T$	{ $E.val := T.val$ }
$T \rightarrow$ (E)	{ $T.val := E.val$ }
$T \rightarrow$ **num**	{ $T.val$:=**num**.val }

Example (No Left Recursion)

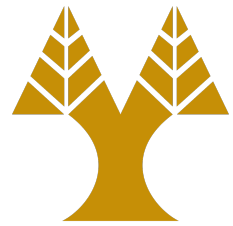
$E \rightarrow$	T	{ $R.i := T.val$ }
	R	{ $E.val := R.s$ }
$R \rightarrow$	+	
	T	{ $R_1.i := R_1.i + T.val$ }
	R_1	{ $R.s := R_1.s$ }
$R \rightarrow$	-	
	T	{ $R_1.i := R_1.i - T.val$ }
	R_1	{ $R.s := R_1.s$ }
$R \rightarrow$	ε	{ $R.s := R.i$ }
$T \rightarrow$	(	
	E	
	)	{ $T.val := E.val$ }
$T \rightarrow$	**num**	{ $T.val :=$ **num**.val }

Translation Scheme (Top-down)



PRODUCTION	SEMANTIC RULES
$E \rightarrow E_1 + T$	*E.nptr:=mknode('+', E_1.nptr, T.nptr)*
$E \rightarrow E_1 - T$	*E.nptr:=mknode('-', E_1.nptr, T.nptr)*
$E \rightarrow T$	*E.nptr:=T.nptr*
$T \rightarrow (E)$	*T.nptr:=E.nptr*
$T \rightarrow$ **id**	*T.nptr:=mkleaf(***id***,* **id***.entry)*
$T \rightarrow$ **num**	*T.nptr*:=mkleaf(**num**, **num**.val)

Translation Scheme (Left-recursion Eliminated)



$E \rightarrow$	T	{ $R.i := T.nptr$ }
	R	{ $E.nptr := R.s$ }
$R \rightarrow$	+	
	T	{ $R_1.i := mknode('+', R.i, T.nptr)$ }
	R_1	{ $R.s := R_1.s$ }
$R \rightarrow$	-	
	T	{ $R_1.i := mknode('-', R.i, T.nptr)$ }
	R_1	{ $R.s := R_1.s$ }
$R \rightarrow$	ε	{ $R.s := R.i$ }
$T \rightarrow$	(	
	E	
	)	{ $T.nptr := E.nptr$ }
$T \rightarrow$	**id**	{ $T.nptr := mkleaf($**id**, **id**.entry) }
$T \rightarrow$	**num**	{ $T.nptr := mkleaf($**num**, **num**.val) }