How to Create a Programming Language

L9: SYNTAX-DIRECTED TRANSLATION

1

OVERVIEW

- ASSOCIATE INFORMATION WITH PROGRAMMING LANGUAGE CONSTRUCT
  - ATTACHING ATTRIBUTES TO THE GRAMMAR SYMBOLS
  - SEMANTIC RULES FOR THE PRODUCTION COMPUTES THE ATTRIBUTES
- S-ATTRIBUTED DEFINITIONS
- L-ATTRIBUTED DEFINITIONS

2

SYNTAX-DIRECTED DEFINITION (SDD)

- GENERALIZATION OF CONTEXT FREE GRAMMAR
  - EACH GRAMMAR SYMBOL HAS A SET OF ATTRIBUTES
- ATTRIBUTES
  - THEIR VALUES ARE COMPUTED BY SEMANTIC RULES (ANNOTATING, DECORATING)
  - **SYNTHESIZED ATTRIBUTES** OF A NODE: VALUES ARE COMPUTED FROM THE ATTRIBUTES OF CHILDREN NODE
  - **INHERITED ATTRIBUTES** OF A NODE: VALUES ARE COMPUTED FROM THE SIBLING AND PARENT NODES
- DEPENDENCIES BETWEEN ATTRIBUTES
  - REPRESENTED BY DEPENDENCY GRAPH
  - DERIVE EVALUATION ORDER FROM THE DEPENDENCY GRAPH

3

SYNTAX-DIRECTED DEFINITION (SDD)

- EXAMPLE

	PRODUCTION	SEMANTIC RULES
1)	$L \rightarrow E\ \mathbf{n}$	$L.val = E.val$
2)	$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$
3)	$E \rightarrow T$	$E.val = T.val$
4)	$T \rightarrow T_1 * F$	$T.val = T_1.val \times F.val$
5)	$T \rightarrow F$	$T.val = F.val$
6)	$F \rightarrow (\ E\ )$	$F.val = E.val$
7)	$F \rightarrow \mathbf{digit}$	$F.val = \mathbf{digit}.\mathrm{lexval}$

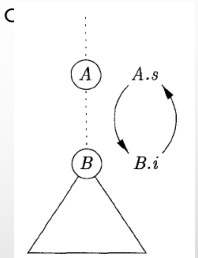
- TERMINALS HAVE SYNTHESIZED ATTRIBUTES ONLY
- START SYMBOL DOES NOT HAVE INHERITED ATTRIBUTE
- EXERCISE: DRAW THE PARSE TREE FOR 3 * 5 + 4 N

4

EVALUATING SDDS

- WHEN INHERITED AND SYNTHESIZED ATTRIBUTES ARE MIXED, THERE ARE NO GUARANTEE THAT THESE ATTRIBUTES C

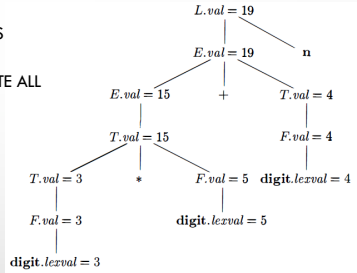
PRODUCTION	SEMANTIC RULES
$A \rightarrow B$	$A.s = B.i;$ $B.i = A.s + 1$



5

BOTTOM-UP EVALUATION

- S-ATTRIBUTED DEFINITION
  - SYNTAX-DIRECTED DEFINITION THAT USES SYNTHESIZED ATTRIBUTES EXCLUSIVELY.
  - BOTTOM-UP EVALUATION CAN ANNOTATE ALL ATTRIBUTES



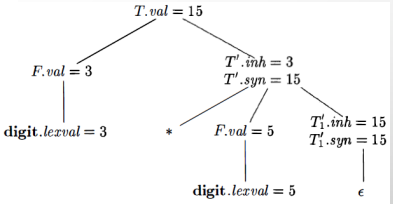
Annotated parse tree for $3 * 5 + 4$ **n**

6

TOP-DOWN EVALUATION

- INHERITED ATTRIBUTES CAN GIVE CONTEXT TO LANGUAGE CONSTRUCT
  - E.G. WHETHER AN ID APPEARS ON THE LHS OR THE RHS OF =
  - EXAMPLE BELOW PARSES 1 * 2, 1 * 2 * 3, …

	PRODUCTION	SEMANTIC RULES
1)	$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.syn$
2)	$T' \rightarrow * F\ T_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
3)	$T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4)	$F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$



Annotated parse tree for $3 * 5$

7

DEPENDENCY GRAPH

- IT CAN DEPICT THE INTERDEPENDENCIES AMONG THE INHERITED AND SYNTHESIZED ATTRIBUTES AT THE NODE.
- DETERMINING THE EVALUATION ORDER OF THE ATTRIBUTES.

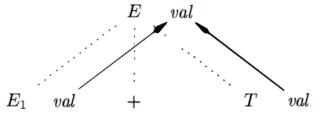
```
for each node n in the parse tree do
   for each attribute a of the grammar symbol at n do
      construct a node in the dependency graph for a

for each node n in the parse tree do
   for each semantic rule b := f(c1, c2, …, ck)
                  associated with the production at n do
      for i := 1 to k do
         construct an edge
         from the node for ci to the node for b
```


8

DEPENDENCY GRAPH

- EXAMPLE 1.

PRODUCTION	SEMANTIC RULE
$E \rightarrow E_1 + T$	$E.val = E_1.val + T.val$



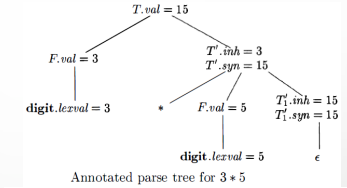
9

DEPENDENCY GRAPH

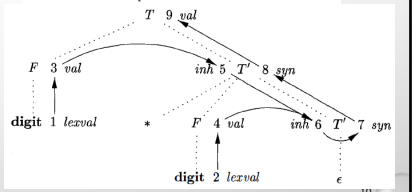
- EXAMPLE 2

	PRODUCTION	SEMANTIC RULES
1)	$T \rightarrow F\ T'$	$T'.inh = F.val$ $T.val = T'.syn$
2)	$T' \rightarrow *\ F\ T_1'$	$T_1'.inh = T'.inh \times F.val$ $T'.syn = T_1'.syn$
3)	$T' \rightarrow \epsilon$	$T'.syn = T'.inh$
4)	$F \rightarrow$ **digit**	$F.val =$ **digit**$.lexval$

Exercise: Draw a dependency graph for 2 * 3 * 4



Annotated parse tree for 3 * 5



10

EVALUATION ORDER

- TOPOLOGICAL SORT OF A DIRECTED ACYCLIC GRAPH
  - ANY ORDERING $M_1, M_2, \ldots, M_K$ OF THE NODES OF THE GRAPH SUCH THAT IF THERE IS AN EDGE M_I -> M_J, THEN M_I **APPEARS BEFORE** M_J IN THE ORDERING.
- ANY TOPOLOGICAL SORT OF A DEPENDENCY GRAPH GIVES A VALID ORDER TO EVALUATE ATTRIBUTES.
- EVALUATION OF SEMANTIC RULES IN THIS ORDER YIELDS THE TRANSLATION.

12

L-ATTRIBUTED DEFINITIONS

- AN SDD IS **L-ATTRIBUTED**, IF EACH **INHERITED** ATTRIBUTE OF X_J IN **A** -> $X_1\ X_2\ \ldots\ X_N$ DEPENDS ONLY ON
  - THE ATTRIBUTES OF THE SYMBOLS $X_1, X_2, \ldots X_{J-1}$
  - THE INHERITED ATTRIBUTES OF **A**
- EVERY S-ATTRIBUTED DEFINITION IS L-ATTRIBUTED, BECAUSE IT DOESN'T HAVE INHERITED ATTRIBUTES.
- L-ATTRIBUTED DEFINITIONS CAN BE EVALUATED IN DEPTH-FIRST ORDER.

13

L-ATTRIBUTED DEFINITIONS

- EXAMPLE

PRODUCTION	SEMANTIC RULE
$T \to F\ T'$	$T'.inh = F.val$
$T' \to *\ F\ T'_1$	$T'_1.inh = T'.inh \times F.val$

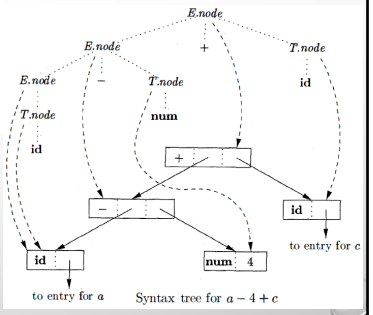
- EXERCISE: IS THIS AN L-ATTRIBUTED DEFINITION?

PRODUCTION	SEMANTIC RULES
$A \to B\ C$	$A.s = B.b;$ $B.i = f(C.c, A.s)$

14

APPLICATION: CONSTRUCTING A SYNTAX TREE

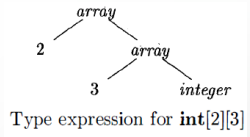
	PRODUCTION	SEMANTIC RULES
1)	$E \to E_1 + T$	$E.node$ = **new** $Node('+', E_1.node, T.node)$
2)	$E \to E_1 - T$	$E.node$ = **new** $Node('-', E_1.node, T.node)$
3)	$E \to T$	$E.node = T.node$
4)	$T \to (\ E\ )$	$T.node = E.node$
5)	$T \to$ **id**	$T.node$ = **new** $Leaf($**id**$, $**id**$.entry)$
6)	$T \to$ **num**	$T.node$ = **new** $Leaf($**num**$, $**num**$.val)$



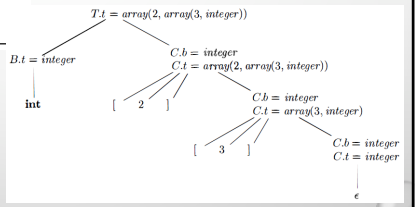
15

APPLICATION: TYPE EXPRESSION

PRODUCTION	SEMANTIC RULES
$T \to B\ C$	$T.t = C.t$
	$C.b = B.t$
$B \to$ **int**	$B.t = integer$
$B \to$ **float**	$B.t = float$
$C \to [\ \textbf{num}\ ]\ C_1$	$C.t = array(\textbf{num}.val,\ C_1.t)$
	$C_1.b = C.b$
$C \to \epsilon$	$C.t = C.b$



Type expression for **int**[2][3]



16

TOP-DOWN TRANSLATION

- L-ATTRIBUTED DEFINITIONS WILL BE IMPLEMENTED DURING PREDICTIVE PARSING.
- ELIMINATING LEFT RECURSION FROM TRANSLATION SCHEME
  - EVALUATE INHERITED ATTRIBUTES (R.I) BEFORE A USE OF R
  - EVALUATE SYNTHESIZED ATTRIBUTES (A.A, R.S) AT THE END OF THE PRODUCTION

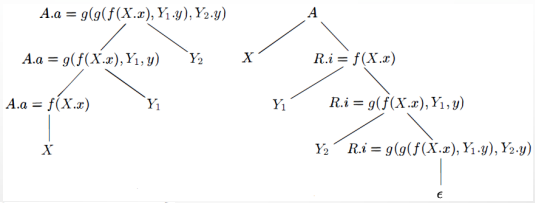
$$A \to A_1\ Y\ \{A.a = g(A_1.a, Y.y)\}$$
$$A \to X\ \{A.a = f(X.x)\}$$

$$A \to X\ R$$
$$R \to Y\ R \mid \epsilon$$

$$A \to X\ \{R.i = f(X.x)\}\ R\ \{A.a = R.s\}$$
$$R \to Y\ \{R_1.i = g(R.i, Y.y)\}\ R_1\ \{R.s = R_1.s\}$$
$$R \to \epsilon\ \{R.s = R.i\}$$

17

ELIMINATING LEFT RECURSION FROM TRANSLATION SCHEME



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

$$A \rightarrow X\ \{R.i = f(X.x)\}\ R\ \{A.a = R.s\}$$
$$R \rightarrow Y\ \{R_1.i = g(R.i, Y.y)\}\ R_1\ \{R.s = R_1.s\}$$
$$R \rightarrow \epsilon\ \{R.s = R.i\}$$

Exercise: Eliminate Left Recursion from

```
A -> A1 Y { A.a = g(A1.a, Y.y) }
A -> A1 Z { A.a = h(A1.a, Z.z) }
A -> X { A.a = f(X.x) }
```


18

PREDICTIVE TRANSLATOR

- FOR EACH NONTERMINAL A, CONSTRUCT A FUNCTION A WITH
  - FORMAL PARAMETERS FOR THE INHERITED ATTRIBUTES OF A
  - RETURNS A COLLECTION OF THE SYNTHESIZED ATTRIBUTES OF A
- DECIDE WHAT PRODUCTION TO USE BASED ON THE LOOKAHEAD
- CODE FOR THE PRODUCTION
  - FOR A TOKEN X, SAVE THE VALUE OF X AT X.X AND MATCH THE TOKEN
  - FOR A NONTERMINAL B, DO THE ASSIGNMENT C := B(B1, … BK), WHERE B1, …, BK ARE THE VARIABLES FOR THE INHERITED ATTRIBUTES OF B, AND C IS A VARIABLE FOR THE SYNTHESIZED ATTRIBUTE OF B
  - FOR AN ACTION, COPY THE CODE INTO THE PARSER, REPLACE REFERENCE TO ATTRIBUTES BY THEIR CORRESPONDING VARIABLES.

19

EXAMPLE: PREDICTIVE TRANSLATION

```
E -> E1 + T { E.np = mknode('+', E1.np, T.np) }
E -> E1 - T { E.np = mknode('-', E1.np, T.np) }
E -> T      { E.np = T.np }
```

```
E -> T    { R.i = T.np }
     R    { E.np = R.s }
R -> +
     T    { R1.i = mknode('+', R.i, T.np) }
     R1   { R.s = R1.s}
R -> -
     T    { R1.i = mknode('-', R.i, T.np) }
     R1   { R.s = R1.s}
R -> eps  { R.s =  R.i }
```

```
SN* R(SN* i) {
  char op;
  SN *Ts, *R1i, *R1s, *s;

  if (la == '+') {
    op  = la;
    match('+');
    Ts  = T();
    R1i = mknode('+', i, Ts);
    R1s = R(R1i);
    s   = R1s;
  }
  else if (la == '-')
    ...
  else
    s = i;

  return s;
}
```


20

PREDICTIVE TRANSLATION

- QUIZ: SKETCH A PREDICTIVE TRANSLATION FOR THE FOLLOWING GRAMMAR

```
E -> E1 + T { E.np = mknode('+', E1.np, T.np) }
E -> T      { E.np = T.np }
T -> T1 * F { T.np = mknode('*', T1.np, F.np) }
T -> F      { T.np = F.np }
```


21

BOTTOM-UP TRANSLATION

- L-ATTRIBUTED DEFINITIONS WILL BE IMPLEMENTED DURING LR-PARSING
- LR PARSERS USE A STACK TO HOLD INFORMATION ABOUT PARSED SUBTREES
  - ADD EXTRA FIELDS VAL IN THE STACK TO HOLD THE VALUES OF THE **SYNTHESIZED ATTRIBUTES**.
  - IF THE I[TH] STATE SYMBOL IS A, THEN VAL[I] HOLDS THE ATTRIBUTES ASSOCIATED WITH A.
  - E.G.

IF `A -> X Y Z` IS A PRODUCTION AND
`A.A = F(X.X, Y.Y, Z.Z)` IS A SEMANTIC RULE

```
Z.Z = VAL[TOP], Y.Y = VAL[TOP-1], X.X =
VAL[TOP-2]
A.A = F(VAL[TOP-2], VAL[TOP-1], VAL[TOP])
```


22

INHERITED ATTRIBUTES IN YACC

```
DECLARATION
    : CLASS TYPE IDLIST;
CLASS
    : GLOBAL {$$ = 1;}
    | LOCAL  {$$ = 2;}
    ;
TYPE
    : REAL    {$$ = 1;}
    | INTEGER {$$=2;}
    ;
IDLIST
    : ID        {MKSYMBOL($0,$-1, $1)}
    | IDLIST ID {MKSYMBOL($0,$-1, $2)}
    ;
```


23

EXAMPLE: EVALUATION BY PARSER STACK

L	→	E **n**	{ print($E.val$); }
E	→	$E_1 + T$	{ $E.val = E_1.val + T.val$; }
E	→	T	{ $E.val = T.val$; }
T	→	$T_1 * F$	{ $T.val = T_1.val \times F.val$; }
T	→	F	{ $T.val = F.val$; }
F	→	(E)	{ $F.val = E.val$; }
F	→	**digit**	{ $F.val =$ **digit**$.lexval$; }

PRODUCTION	CODE FRAGMENT
$L \rightarrow E$ **n**	*print* (*val* [*top*])
$E \rightarrow E_1 + T$	*val* [*ntop*] := *val* [*top* − 2] + *val* [*top*]
$E \rightarrow T$	
$T \rightarrow T_1 * F$	*val* [*ntop*] := *val* [*top* − 2] × *val* [*top*]
$T \rightarrow F$	
$F \rightarrow$ (E)	*val* [*ntop*] := *val* [*top* − 1]
$F \rightarrow$ **digit**	

INPUT	*state*	*val*	PRODUCTION USED
3*5+4 **n**	_	_	
*5+4 **n**	3	3	
*5+4 **n**	F	3	$F \rightarrow$ **digit**
*5+4 **n**	T	3	$T \rightarrow F$
5+4 **n**	T *	3 _	
+4 **n**	T * 5	3 _ 5	
+4 **n**	T * F	3 _ 5	$F \rightarrow$ **digit**
+4 **n**	T	15	$T \rightarrow T * F$
+4 **n**	E	15	$E \rightarrow T$
4 **n**	E +	15 _	
n	E + 4	15 _ 4	
n	E + F	15 _ 4	$F \rightarrow$ **digit**
n	E + T	15 _ 4	$T \rightarrow F$
	E	19	$E \rightarrow E + T$
	E **n**	19 _	
	L	19	$L \rightarrow E$ **n**

24

MARKER NONTERMINALS

- NONTERMINALS WITH THE EPSILON PRODUCTION.
- MOVE EMBEDDED ACTIONS TO THE RIGHT ENDS OF THEIR PRODUCTIONS.

```
E -> T R
E -> + T { print('+') } R
   | - T { print('-') } R
   | eps
T -> num { print(num.val) }
```

```
E -> T R
R -> + T M R | - T N R | eps
T ->  num { print(num.val) }
M ->  eps { print('+') }
N ->  eps { print('-') }
```


25

MARKER NONTERMINALS

- SIMULATING THE EVALUATION OF INHERITED ATTRIBUTES
  - E.G. WHEN REDUCING C->C,

Production	Semantic Rules
S -> aAC	C.i = A.s
S -> bABC	C.i = A.s
C -> c	C.s = g(C.i)

C.i = val[top - 1] or C.i = val[top - 2]

Production	Semantic Rules
S -> aAC	C.i = A.s
S -> bABMC	M.i = A.s, C.i = M.s
C -> c	C.s = g(C.i)
M -> eps	M.s = M.i

C.i = val[top - 1]

26

MARKER NONTERMINALS

- WHEN INHERITED ATTRIBUTES ARE NOT UPDATED BY COPY, ITS VALUE IS NOT IN THE VAL STACK.

Production	Semantic Rules
S -> aAC	C.i = f(A.s)

f(A.s) is not in val stack

Production	Semantic Rules
S -> aANC	N.i = A.s, C.i = N.s
N -> eps	N.s = f(N.i)

C.i = val[top -1]

27

PARSER STACK FOR INHERITED ATTRIBUTES

- ASSUME THAT EVERY NONTERMINAL A HAS ONE INHERITED ATTRIBUTE A.I AND EVERY GRAMMAR SYMBOL X HAS A SYNTHESIZED ATTRIBUTE X.S
- FOR EVERY PRODUCTION $A \rightarrow X_1 \ldots X_N$, REPLACE IT WITH $A \rightarrow M_1 X_1 \ldots M_N X_N$ WHERE $M_1 \ldots M_N$ ARE NEW MARKERS.
  - SYNTHESIZED ATTRIBUTES $X_J.S$ WILL BE IN VAL STACK ASSOCIATED WITH X_J
  - INHERITED ATTRIBUTES $X_J.I$ APPEARS IN VAL STACK BUT ASSOCIATED WITH M_J

28

MARKER NONTERMINALS

- ADDING MARKER NONTERMINALS DOESN'T INTRODUCE CONFLICTS TO LL(1) GRAMMARS
- FOR LR(1) GRAMMARS, MARKER NONTERMINALS CAN INTRODUCE PARSING CONFLICTS.

29


QUESTIONS?
Sejong - How to Create a Programming Language - Tony Mione / 2020
30

30