Lecture 16

Semantics Analysis

Awanish Pandey

Department of Computer Science and Engineering
Indian Institute of Technology
Roorkee

February 25, 2025

Take aways from the last class

- Translation Scheme for Equation

Take aways from the last class

- Translation Scheme for Equation
- Top-down parsing of translation scheme

Take aways from the last class

- Translation Scheme for Equation
- Top-down parsing of translation scheme
- Eliminate Left recursion

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \to \epsilon$

$E \to TR$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$E \rightarrow TR$
$R \rightarrow +T \quad print(+) \quad R$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$E \rightarrow TR$$
$$R \rightarrow +T \quad print(+) \quad R$$
$$R \rightarrow -T \quad print(-) \quad R$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$E \rightarrow TR$
$R \rightarrow +T \quad print(+) \quad R$
$R \rightarrow -T \quad print(-) \quad R$
$R \rightarrow \epsilon$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$
\begin{aligned}
&E \rightarrow TR\\
&R \rightarrow +T \quad print(+) \quad R\\
&R \rightarrow -T \quad print(-) \quad R\\
&R \rightarrow \epsilon\\
&T \rightarrow num \quad print(num.val)
\end{aligned}
$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$E \rightarrow TR$$

$$E \rightarrow TR$$
$$R \rightarrow +T \quad print(+) \quad R$$
$$R \rightarrow -T \quad print(-) \quad R$$
$$R \rightarrow \epsilon$$
$$T \rightarrow num \quad print(num.val)$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$E \rightarrow TR$
$R \rightarrow +T \quad print(+) \quad R$
$R \rightarrow -T \quad print(-) \quad R$
$R \rightarrow \epsilon$
$T \rightarrow num \quad print(num.val)$

$E \rightarrow TR$
$R \rightarrow +TMR$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$E \rightarrow TR$
$R \rightarrow +T \quad print(+) \quad R$
$R \rightarrow -T \quad print(-) \quad R$
$R \rightarrow \epsilon$
$T \rightarrow num \quad print(num.val)$

$E \rightarrow TR$
$R \rightarrow +TMR$
$R \rightarrow -TNR$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$
\begin{aligned}
&E \rightarrow TR \\
&R \rightarrow +T \quad print(+) \quad R \\
&R \rightarrow -T \quad print(-) \quad R \\
&R \rightarrow \epsilon \\
&T \rightarrow num \quad print(num.val)
\end{aligned}
$$

$$
\begin{aligned}
&E \rightarrow TR \\
&R \rightarrow +TMR \\
&R \rightarrow -TNR \\
&R \rightarrow \epsilon
\end{aligned}
$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +T \quad print(+) \quad R \\
R &\rightarrow -T \quad print(-) \quad R \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad print(num.val)
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +TMR \\
R &\rightarrow -TNR \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad \{print(num.val)\}
\end{aligned}
$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +T \quad print(+) \quad R \\
R &\rightarrow -T \quad print(-) \quad R \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad print(num.val)
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +TMR \\
R &\rightarrow -TNR \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad \{print(num.val)\} \\
M &\rightarrow \epsilon \quad \{print(+)\}
\end{aligned}
$$

Bottom up evaluation of inherited attributes

- Remove embedded actions from translation scheme
- Make transformation so that embedded actions occur only at the ends of their productions
- Replace each action by a distinct marker non terminal M and attach action at end of $M \rightarrow \epsilon$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +T \quad print(+) \quad R \\
R &\rightarrow -T \quad print(-) \quad R \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad print(num.val)
\end{aligned}
$$

$$
\begin{aligned}
E &\rightarrow TR \\
R &\rightarrow +TMR \\
R &\rightarrow -TNR \\
R &\rightarrow \epsilon \\
T &\rightarrow num \quad \{print(num.val)\} \\
M &\rightarrow \epsilon \quad \{print(+)\} \\
N &\rightarrow \epsilon \quad \{print(-)\}
\end{aligned}
$$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \to XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \to XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example $real \quad p, q, r$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example $real \quad p, q, r$

$D \rightarrow T \quad \{L.in = T.type\} \quad L$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \to XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example $real \quad p, q, r$

$D \to T \quad \{L.in = T.type\} \quad L$
$T \to int \quad \{T.type = integer\}$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example $real \quad p, q, r$

$D \rightarrow T \quad \{L.in = T.type\} \quad L$
$T \rightarrow int \quad \{T.type = integer\}$
$T \rightarrow real \quad \{T.type = real\}$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example *real* $\quad p, q, r$

$D \rightarrow T \quad \{L.in = T.type\} \quad L$
$T \rightarrow int \quad \{T.type = integer\}$
$T \rightarrow real \quad \{T.type = real\}$
$L \rightarrow \{L_1.in = L.in\} \quad L_1, id \quad \{addtype(id.entry, L.in)\}$

Inheriting attribute on parser stacks

- Bottom up parser reduces rhs of $A \rightarrow XY$ by removing XY from stack and putting A on the stack
- Synthesized attributes of X_s can be inherited by Y by using the copy rule $Y.i = X.s$
- Example $real \quad p, q, r$

$D \rightarrow T \quad \{L.in = T.type\} \quad L$
$T \rightarrow int \quad \{T.type = integer\}$
$T \rightarrow real \quad \{T.type = real\}$
$L \rightarrow \{L_1.in = L.in\} \quad L_1, id \quad \{addtype(id.entry, L.in)\}$
$L \rightarrow id \quad \{addtype(id.entry, L.in)\}$

Analysis

State stack	INPUT	PRODUCTION
	real p,q,r	
real	p,q,r	
T	p,q,r	T → real
Tp	,q,r	
TL	,q,r	L → id
TL,	q,r	
TL,q	,r	
TL	,r	L → L,id
TL,	r	
TL,r	-	
TL	-	L → L,id
D	-	D → TL

Analysis

- Every time a reduction to L is made value of T type is just below it

Analysis

- Every time a reduction to L is made value of T type is just below it
- Use the fact that $T.val$ (type information) is at a known place in the stack

Analysis

- Every time a reduction to L is made value of T type is just below it
- Use the fact that $T.val$ (type information) is at a known place in the stack
- When production $L \rightarrow id$ is applied, $id.entry$ is at the top of the stack and $T.type$ is just below it, therefore,

Analysis

- Every time a reduction to L is made value of T type is just below it
- Use the fact that $T.val$ (type information) is at a known place in the stack
- When production $L \rightarrow id$ is applied, $id.entry$ is at the top of the stack and $T.type$ is just below it, therefore,
  $addtype(id.entry, L.in) \longleftrightarrow addtype(val[top], val[top-1])$

Analysis

- Every time a reduction to L is made value of T type is just below it
- Use the fact that $T.val$ (type information) is at a known place in the stack
- When production $L \rightarrow id$ is applied, $id.entry$ is at the top of the stack and $T.type$ is just below it, therefore,
  $addtype(id.entry, L.in) \longleftrightarrow addtype(val[top], val[top-1])$
- Similarly when production $L \rightarrow L_1, id$ is applied $id.entry$ is at the top of the stack and $T.type$ is three places below it, therefore,

Analysis

- Every time a reduction to L is made value of T type is just below it
- Use the fact that $T.val$ (type information) is at a known place in the stack
- When production $L \rightarrow id$ is applied, $id.entry$ is at the top of the stack and $T.type$ is just below it, therefore,
  $addtype(id.entry, L.in) \longleftrightarrow addtype(val[top], val[top-1])$
- Similarly when production $L \rightarrow L_1, id$ is applied $id.entry$ is at the top of the stack and $T.type$ is three places below it, therefore,
  $addtype(id.entry, L.in) \longleftrightarrow addtype(val[top], val[top-3])$

Analysis

$D \rightarrow TL$
$T \rightarrow int \quad val[top] = integer$
$T \rightarrow real \quad val[top] = real$
$L \rightarrow L, id \quad addtype(val[top], val[top-3])$
$L \rightarrow id \quad addtype(val[top], val[top-1])$

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.
- Example
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABC \quad C_i = A_s$

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.
- Example

  $S \rightarrow aAC \quad C_i = A_s$

  $S \rightarrow bABC \quad C_i = A_s$

  $C \rightarrow c \quad C_s = g(C_i)$

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.
- Example
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABC \quad C_i = A_s$
  $C \rightarrow c \quad C_s = g(C_i)$
- C inherits A_s

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.
- Example
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABC \quad C_i = A_s$
  $C \rightarrow c \quad C_s = g(C_i)$
- C inherits A_s
- There may or may not be a B between A and C on the stack when reduction by rule $C \rightarrow c$ takes place

Simulating the evaluation of inherited attributes

- The scheme works only if grammar allows position of attribute to be predicted.
- Example
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABC \quad C_i = A_s$
  $C \rightarrow c \quad C_s = g(C_i)$
- C inherits A_s
- There may or may not be a B between A and C on the stack when reduction by rule $C \rightarrow c$ takes place
- When reduction by $C \rightarrow c$ is performed the value of C_i is either in [top-1] or [top-2]

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \to aAC \quad C_i = A_s$

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABMC \quad M_i = A_s; C_i = M_s$

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABMC \quad M_i = A_s; C_i = M_s$
  $C \rightarrow c \quad C_s = g(C_i)$

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABMC \quad M_i = A_s; C_i = M_s$
  $C \rightarrow c \quad C_s = g(C_i)$
  $M \rightarrow epsilon \quad M_s = M_i$

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABMC \quad M_i = A_s; C_i = M_s$
  $C \rightarrow c \quad C_s = g(C_i)$
  $M \rightarrow epsilon \quad M_s = M_i$
- When production $M \rightarrow \epsilon$ is applied we have $M_s = M_i = A_s$

Simulating the evaluation

- Insert a marker M just before C in the second rule and change rules to
  $S \rightarrow aAC \quad C_i = A_s$
  $S \rightarrow bABMC \quad M_i = A_s; C_i = M_s$
  $C \rightarrow c \quad C_s = g(C_i)$
  $M \rightarrow epsilon \quad M_s = M_i$
- When production $M \rightarrow \epsilon$ is applied we have $M_s = M_i = A_s$
- Therefore value of C_i is always at [top-1]

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \rightarrow X_1 \cdots X_n$ introduce n markers $M_1 \cdots M_n$ and replace the production by

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \rightarrow X_1 \cdots X_n$ introduce n markers $M_1 \cdots M_n$ and replace the production by$A \rightarrow M_1 X_1 \cdots M_n X_n$

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \rightarrow X_1 \cdots X_n$ introduce n markers $M_1 \cdots M_n$ and replace the production by$A \rightarrow M_1 X_1 \cdots M_n X_n$
  $M_1 \rightarrow M_n \rightarrow \epsilon$

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \rightarrow X_1 \cdots X_n$ introduce n markers $M_1 \cdots M_n$ and replace the production by$A \rightarrow M_1 X_1 \cdots M_n X_n$
  $M_1 \rightarrow M_n \rightarrow \epsilon$
- Synthesized attribute $X_{j,s}$ goes into the value entry of X_j

General algorithm

- Algorithm: Bottom up parsing and translation with inherited attributes
- Input: L attributed definitions
- Output: A bottom up parser
- Assume every non terminal has one inherited attribute and every grammar symbol has a synthesized attribute
- For every production $A \to X_1 \cdots X_n$ introduce n markers $M_1 \cdots M_n$ and replace the production by$A \to M_1X_1 \cdots M_nX_n$
  $M_1 \to M_n \to \epsilon$
- Synthesized attribute $X_{j,s}$ goes into the value entry of X_j
- Inherited attribute $X_{j,i}$ goes into the value entry of M_j

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production $A \rightarrow M_1X_1 \cdots M_nX_n$ *then*

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production $A \rightarrow M_1X_1 \cdots M_nX_n$ *then* A_i is in position $top - 2j + 2$

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production $A \to M_1X_1 \cdots M_nX_n$ *then*
  A_i is in position $top - 2j + 2$
  $X_{1.i}$ is in position $top - 2j + 3$

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production $A \rightarrow M_1 X_1 \cdots M_n X_n$ *then*
  A_i is in position $top - 2j + 2$
  $X_{1.i}$ is in position $top - 2j + 3$
  $X_{1.s}$ is in position $top - 2j + 4$

Algorithm

- If the reduction is to a marker M_j and the marker belongs to a production $A \rightarrow M_1 X_1 \cdots M_n X_n$ *then*
  A_i is in position $top - 2j + 2$
  $X_{1.i}$ is in position $top - 2j + 3$
  $X_{1.s}$ is in position $top - 2j + 4$
- If reduction is to a non terminal A by production, $A \rightarrow M_1 X_1 \cdots M_n X_n$ then compute A_s and push on the stack