Lecture 15

Semantics Analysis

Awanish Pandey

Department of Computer Science and Engineering
Indian Institute of Technology
Roorkee

February 21, 2025

Take aways from the last class

- Dependency Graph

Take aways from the last class

- Dependency Graph
- S-attriuted grammar

Take aways from the last class

- Dependency Graph
- S-attriuted grammar
- Abstract Syntax Tree and its construction

Take aways from the last class

- Dependency Graph
- S-attriuted grammar
- Abstract Syntax Tree and its construction
- Bottom-up evaluation of S-attributed definition

Take aways from the last class

- Dependency Graph
- S-attriuted grammar
- Abstract Syntax Tree and its construction
- Bottom-up evaluation of S-attributed definition
- L-attributed Grammar

Take aways from the last class

- Dependency Graph
- S-attriuted grammar
- Abstract Syntax Tree and its construction
- Bottom-up evaluation of S-attributed definition
- L-attributed Grammar
- Translation Scheme

Translation Scheme

- A CFG where semantic actions occur within the RHS of production

Translation Scheme

- A CFG where semantic actions occur within the RHS of production
- A translation scheme to map infix to postfix

Translation Scheme

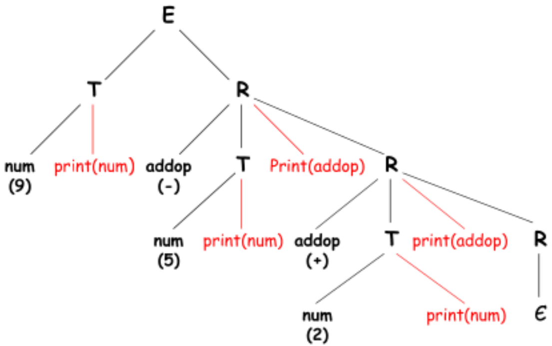
- A CFG where semantic actions occur within the RHS of production
- A translation scheme to map infix to postfix

  $E \rightarrow TR$

  $R \rightarrow addop\ T \quad \{print(addop)\} \quad R|\epsilon$

  $T \rightarrow num \quad print(num)$

Parse tree for 9-5+2



- Assume actions are terminal symbols

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain 9 5 – 2 $+$

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain 9 5 – 2 +
- When designing translation scheme, ensure attribute value is available when referred to

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain 9 5 – 2 +
- When designing translation scheme, ensure attribute value is available when referred to
- In case of synthesized attribute it is trivial.

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain $9\ 5 - 2\ +$
- When designing translation scheme, ensure attribute value is available when referred to
- In case of synthesized attribute it is trivial.
- In case of both inherited and synthesized attributes?

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain 9 5 – 2 $+$
- When designing translation scheme, ensure attribute value is available when referred to
- In case of synthesized attribute it is trivial.
- In case of both inherited and synthesized attributes?
  - $S \rightarrow$ __ A __ B __

- Assume actions are terminal symbols
- Perform depth first order traversal to obtain 9 5 – 2 $+$
- When designing translation scheme, ensure attribute value is available when referred to
- In case of synthesized attribute it is trivial.
- In case of both inherited and synthesized attributes?
  - $S \rightarrow$ __ A __ B __
  - An inherited attribute for a symbol on rhs of a production must be computed in an action before that symbol
  - $S \rightarrow A_i A B_i B S_s$

Example: Translation scheme for EQN

$S \rightarrow B$
$B \rightarrow B_1 B_2$
$B \rightarrow B_1 \text{ sub } B_2$
$B \rightarrow \textit{text}$

Example: Translation scheme for EQN

$S \rightarrow B \qquad\qquad B.pts = 10$

Example: Translation scheme for EQN

$S \rightarrow B$	$B.pts = 10$
$B \rightarrow B_1 B_2$	$B_1.pts = B.pts$

Example: Translation scheme for EQN

$$
\begin{array}{ll}
S \rightarrow B & B.pts = 10 \\
B \rightarrow B_1 B_2 & B_1.pts = B.pts \\
 & B_2.pts = B.pts
\end{array}
$$

Example: Translation scheme for EQN

$S \rightarrow B$	$B.pts = 10$
$B \rightarrow B_1 B_2$	$B_1.pts = B.pts$
	$B_2.pts = B.pts$
$B \rightarrow B_1$ sub B_2	$B_1.pts = B.pts$

Example: Translation scheme for EQN

$S \rightarrow B$	$B.pts = 10$
$B \rightarrow B_1 B_2$	$B_1.pts = B.pts$
	$B_2.pts = B.pts$
$B \rightarrow B_1 \text{ sub } B_2$	$B_1.pts = B.pts$
	$B_2.pts = shrink(B.pts)$
$B \rightarrow text$	

Example: Translation scheme for EQN

$S \rightarrow B$	$B.pts = 10$
	$S.ht = B.ht$
$B \rightarrow B_1 B_2$	$B_1.pts = B.pts$
	$B_2.pts = B.pts$
	$B.ht = max(B_1.ht, B_2.ht)$
$B \rightarrow B_1 \text{ sub } B_2$	$B_1.pts = B.pts$
	$B_2.pts = shrink(B.pts)$
	$B.ht = disp(B_1.ht, B_2.ht)$
$B \rightarrow text$	$B.ht = text.h * B.pts$

Example: SDT Form

$S \rightarrow \{B.pts = 10\}$ B
$\{S.ht = B.ht\}$

Example: SDT Form

$S \rightarrow \{B.pts = 10\}$ B

$\{S.ht = B.ht\}$

$B \rightarrow \{B_1.pts = B.pts\} \quad B_1$

$\{B_2.pts = B.pts\} \quad B_2$

$\{B.ht = max(B_1.ht, B_2.ht)\}$

Example: SDT Form

$S \rightarrow \{B.pts = 10\}$ B
$\{S.ht = B.ht\}$
$B \rightarrow \{B_1.pts = B.pts\} \quad B_1$
$\{B_2.pts = B.pts\} \quad B_2$
$\{B.ht = max(B_1.ht, B_2.ht)\}$
$B \rightarrow \{B_1.pts = B.pts\} \quad B_1$ sub
$\{B_2.pts = shrink(B.pts)\} \quad B_2$
$\{B.ht = disp(B_1.ht, B_2.ht)\}$

Example: SDT Form

$$
\begin{aligned}
S &\rightarrow \{B.pts = 10\}\ \mathrm{B} \\
&\quad \{S.ht = B.ht\} \\
B &\rightarrow \{B_1.pts = B.pts\} \quad B_1 \\
&\quad \{B_2.pts = B.pts\} \quad B_2 \\
&\quad \{B.ht = max(B_1.ht, B_2.ht)\} \\
B &\rightarrow \{B_1.pts = B.pts\} \quad B_1\ \mathrm{sub} \\
&\quad \{B_2.pts = shrink(B.pts)\} \quad B_2 \\
&\quad \{B.ht = disp(B_1.ht, B_2.ht)\} \\
B &\rightarrow text \qquad \{B.ht = text.h * B.pts\}
\end{aligned}
$$

Top Down Translation

- Use predictive parsing to implement L- attributed definitions

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

$$E \to E + T \quad print(+)$$

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

$E \rightarrow TR$

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

$E \rightarrow TR$
$R \rightarrow +Tprint(+)R$

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

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

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

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

  - Tricky case:

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

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

  - Tricky case:
    $A \rightarrow A_1 Y \quad A.a = g(A_1.a.Y.y)$

Top Down Translation

- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

$$E \rightarrow TR$$
$$R \rightarrow +Tprint(+)R$$
$$R \rightarrow \epsilon$$

  - Tricky case:

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

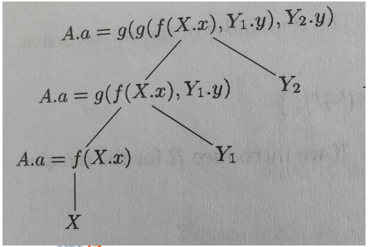
Top Down Translation

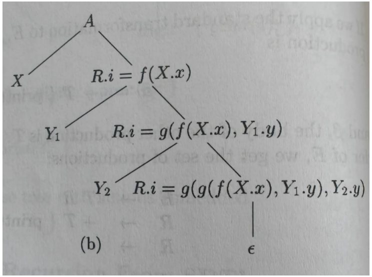
- Use predictive parsing to implement L- attributed definitions
- Eliminate Left Recursion
  - Easy Case:

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

$E \rightarrow TR$
$R \rightarrow +Tprint(+)R$
$R \rightarrow \epsilon$

  - Tricky case:
    $A \rightarrow A_1 Y \quad A.a = g(A_1.a.Y.y)$
    $A \rightarrow X \quad A.a = f(X.x)$





Eliminating Left Recursion

$$A \to X \quad \{R.i = f(X.x)\}R\{A.a = R.s\}$$

Eliminating Left Recursion

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

Eliminating Left Recursion

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