Parsing (Part 1)

Compiler Construction

2024-10-29

Prof. Dr. Michael Kuhn

michael.kuhn@ovgu.de

Parallel Computing and I/O
Institute for Intelligent Cooperating Systems
Faculty of Computer Science
Otto von Guericke University Magdeburg
https://parcio.ovgu.de

Outline

Parsing (Part 1)

- Context-Free Grammars
- LL Grammars
- Summary

Context-Free Grammars

- Terminal: Final symbol of the language (lower case letters)
- Non-terminal: Structure that can be transformed (upper case letters)
- Sentence: Sequence of terminals of the language
    - Sentential form: Sequence of terminals and non-terminals (Greek letters)
- Context-free grammar: List of rules making up the language
    1. $P \longrightarrow E$ (*P* is the start symbol)
    2. $E \longrightarrow X + Y$
    3. $X \longrightarrow \text{int}$
    4. $Y \longrightarrow \text{float}$

Group Exercise

- Perform a top-down derivation of $4 * \text{foo} + 1$ using grammar G_4.

Group Exercise

- Perform a top-down derivation of $4 * \text{foo} + 1$ using grammar G_4.

	Sentential form	Rule
	P	$P \longrightarrow E$
	E	$E \longrightarrow E + T$
	$E + T$	$E \longrightarrow T$
	$T + T$	$T \longrightarrow T * F$
•	$T * F + T$	$T \longrightarrow F$
	$F * F + T$	$F \longrightarrow \text{int}$
	$\text{int} * F + T$	$F \longrightarrow \text{ident}$
	$\text{int} * \text{ident} + T$	$T \longrightarrow F$
	$\text{int} * \text{ident} + F$	$F \longrightarrow \text{int}$
	$\text{int} * \text{ident} + \text{int}$	

Group Exercise

- Perform a bottom-up derivation of $4 * \text{foo} + 1$ using grammar G_4.

Group Exercise

- Perform a bottom-up derivation of $4 * \text{foo} + 1$ using grammar G_4.

Sentential form	Rule
$\text{int} * \text{ident} + \text{int}$	$F \longrightarrow \text{int}$
$\text{int} * \text{ident} + F$	$T \longrightarrow F$
$\text{int} * \text{ident} + T$	$F \longrightarrow \text{ident}$
$\text{int} * F + T$	$F \longrightarrow \text{int}$
• $F * F + T$	$T \longrightarrow F$
$T * F + T$	$T \longrightarrow T * F$
$T + T$	$E \longrightarrow T$
$E + T$	$E \longrightarrow E + T$
E	$P \longrightarrow E$
P	

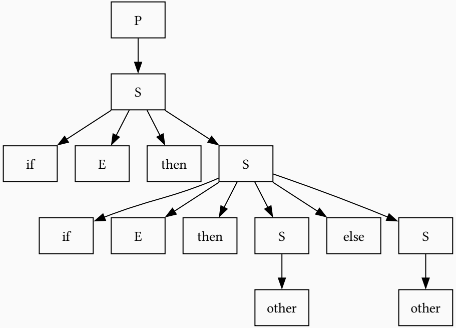
Group Exercise

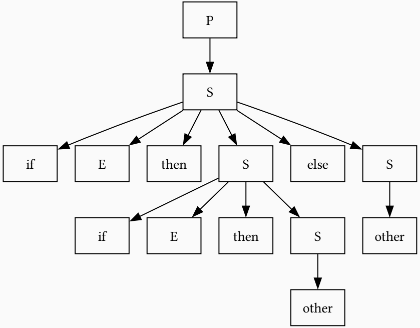
- Perform a bottom-up derivation of $4 \ast \mathrm{foo} + 1$ using grammar G_4.

	Sentential form	Rule
	$\mathrm{int} \ast \mathrm{ident} + \mathrm{int}$	$F \longrightarrow \mathrm{int}$
	$\mathrm{int} \ast \mathrm{ident} + F$	$T \longrightarrow F$
	$\mathrm{int} \ast \mathrm{ident} + T$	$F \longrightarrow \mathrm{ident}$
•	$\mathrm{int} \ast F + T$	$T \longrightarrow F$
	$\mathrm{int} \ast T + T$	$E \longrightarrow T$
	$\mathrm{int} \ast E + T$	$E \longrightarrow E + T$
	$\mathrm{int} \ast E$	$P \longrightarrow E$
	$\mathrm{int} \ast P$	↯

Group Exercise

- Write out two possible parse trees using grammar G_5 for the sentence:
  `if E then if E then other else other`





Group Exercise

- *Modify* grammar G_5 to prevent the dangling-else problem.
  (Hint: Prevent the inner S from containing an if without an else.)

Group Exercise

- Modify grammar G_5 to prevent the dangling-else problem.
  (Hint: Prevent the inner S from containing an if without an else.)
  - $P \longrightarrow S$
  - $S \longrightarrow \text{if } E \text{ then } S$
  - $S \longrightarrow S'$
  - $S' \longrightarrow \text{if } E \text{ then } S' \text{ else } S$
  - $S' \longrightarrow \text{other}$

Outline

Parsing (Part 1)

- Context-Free Grammars
- LL Grammars
- Summary

LL Grammars

- LL(1) grammars are a subset of context-free grammars
- They can be parsed considering only one non-terminal and the next token
  - Remove ambiguity
  - Eliminate left recursion
  - Eliminate common left prefixes

Group Exercise

- Eliminate left recursion for grammar G_{12}.

Group Exercise

- Eliminate left recursion for grammar G_{12}.
    - $E \longrightarrow \text{id } E'$
    - $E \longrightarrow \text{integer } E'$
    - $E' \longrightarrow +EE'$
    - $E' \longrightarrow \epsilon$

Group Exercise

- Eliminate common left prefixes for grammar G_{12}.

Group Exercise

- Eliminate common left prefixes for grammar G_{12}.
  - $S \longrightarrow \text{if } E \text{ then } SS'$
  - $S' \longrightarrow \epsilon$
  - $S' \longrightarrow \text{else } S$

Group Exercise

- $P \longrightarrow S$
- $P \longrightarrow SP$
- $S \longrightarrow \text{if } E \text{ then } SS'$
- $S \longrightarrow \text{while } ES$
- $S \longrightarrow \text{begin } P \text{ end}$
- $S \longrightarrow \text{print } E$
- $S \longrightarrow E$
- $S' \longrightarrow \epsilon$
- $S' \longrightarrow \text{else } S$
- $E \longrightarrow \text{id } E'$
- $E \longrightarrow \text{integer } E'$
- $E' \longrightarrow +EE'$
- $E' \longrightarrow \epsilon$

Group Exercise

- Determine the FIRST sets of the previous grammar.

Group Exercise

- Determine the FIRST sets of the previous grammar.
  - $P = \{\text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}\}$
  - $S = \{\text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}\}$
  - $S' = \{\text{else}, \epsilon\}$
  - $E = \{\text{id}, \text{integer}\}$
  - $E' = \{+, \epsilon\}$

Group Exercise

- Determine the FOLLOW sets of the previous grammar.

Group Exercise

- Determine the FOLLOW sets of the previous grammar.
  - $P = \{\$\}$
  - $S = \{\$, \text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}, \text{else}\}$
  - $S' = \{\$, \text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}, \text{else}\}$
  - $E = \{\text{then}, \text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}, \$, \text{else}, +\}$
  - $E' = \{\text{then}, \text{if}, \text{while}, \text{begin}, \text{print}, \text{id}, \text{integer}, \$, \text{else}, +\}$

Outline

Parsing (Part 1)

Context-Free Grammars

LL Grammars

Summary

Summary

- Context-free grammars are more powerful than regular expressions
  - They are described using multiple rules that replace non-terminals
- LL(1) are a subset of context-free grammars without ambiguities
  - They also eliminate left recursion and common left prefixes

References

[Thain, 2020] Thain, D. (2020). ***Introduction to Compilers and Language Design: Second Edition.*** `http://compilerbook.org/`.