Monash University
Faculty of Information Technology

程序代写代做 CS编程辅导

FIT2014 Theory of Computation

Lecture 15

Parsing

WeChat: cstutorcs

Assignment Project Exam Help

slides by Graham Farr

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

COMMONWEALTH OF AUSTRALIA
Copyright Regulations 1969
WARNING
This material has been reproduced and communicated to you by or on behalf of Monash University
in accordance with s113P of the Copyright Act 1968 (the Act).
The material in this communication may be subject to copyright under the Act.
Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.
Do not remove this notice.

Overview

- Concepts and definitions
- Examples
- Shift-reduce parser
- Lex & Yacc

程序代写代做 CS编程辅导

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Parsing

程序代写代做 CS编程辅导

Suppose you have a Context Free Grammar, and a string of letters.

Parsing: determining whether

- is a word in the language,

  and if it is,
- finding a parse tree, or a derivation, for it.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Parser: a program that does this.

- Two main types:
  - Top-down parsers
  - Bottom-up parsers
    - reduce the string to the Start symbol
    - repeatedly apply production rules in reverse

a^*ba^*b

$$
\begin{aligned}
S &\longrightarrow BB \\
B &\longrightarrow \mathtt{a}B \\
B &\longrightarrow \mathtt{b}
\end{aligned}
$$

程序代写代做 CS编程辅导

Input string: aabab

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

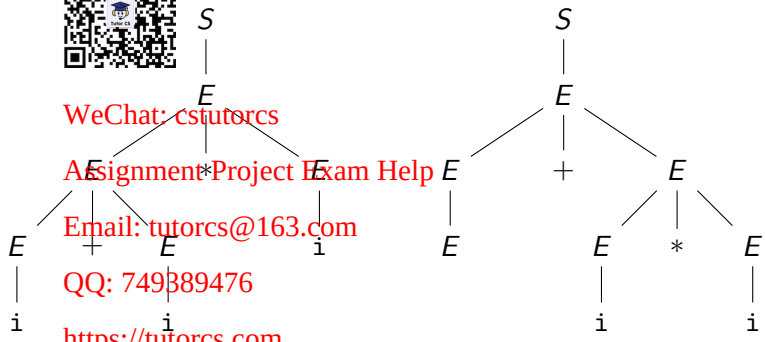
Plus-Times-A

1. $S \longrightarrow E$
2. $E \longrightarrow E + E$
3. $E \longrightarrow E * E$
4. $E \longrightarrow \texttt{i}$

This grammar is *ambiguous*.

程序代写代做CS编程辅导

Input string: $\texttt{i} + \texttt{i} * \texttt{i}$



WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Two parse trees

Plus-Times-B

$$
\begin{array}{lcl}
S & \longrightarrow & E \\
E & \longrightarrow & T + E \\
E & \longrightarrow & T \\
T & \longrightarrow & F * T \\
T & \longrightarrow & F \\
F & \longrightarrow & \texttt{i}
\end{array}
$$

程序代写代做CS编程辅导

Input string: `i + i * i`

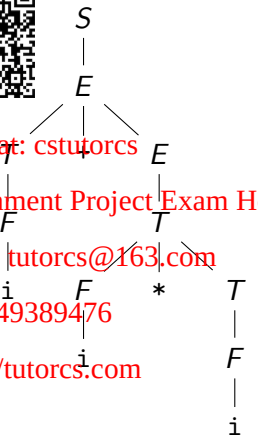
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com



LR Parser

程序代写代做 CS编程辅导

- bottom-up parser
- scans input **L**eft to Right
- constructs a **R**ightmost derivation in reverse
- implemented using a Deterministic Pushdown Automaton (DPDA)
- Not all CFGs have such a parser: DCFL $\neq$ CFL.
- We'll look at one type of LR parser: **shift-reduce parser**

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Shift-reduce Parser

This is a particular type of LR Parser.

程序代写代做 CS编程辅导

It has:

- a **stack**: terminals and non-terminals processed so far.
  - Initially: empty.
- a **buffer**: the rest of the input string (yet to be processed).
  - Initially: contains the entire input string.

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Repeatedly . . .

- **Shift** input letters onto the **stack**, OR
- When a string of top-most **stack** symbols equal the right-hand side of a production rule:
  - **Reduce** that string, i.e., use production rule in reverse

. . . until **Stack** only has Start symbol, and **buffer** is empty.

a^*ba^*b

Input: abb

1. $S \rightarrow BB$
2. $B \rightarrow aB$
3. $B \rightarrow b$

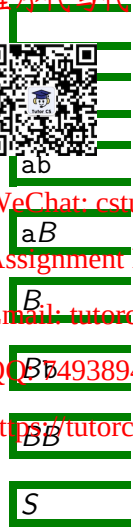


abb		shift
bb		shift
ab	b	reduce, (3)
aB	b	reduce, (2)
B	b	shift
BB		reduce, (3)
S		reduce, (1)
		ACCEPT

程序代写代做 CS编程辅导
WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Plus-Times-A

1. $S \longrightarrow E$
2. $E \longrightarrow E + E$
3. $E \longrightarrow E * E$
4. $E \longrightarrow \texttt{i}$

Input: i+i*i

程序代写代做 CS编程辅导

	i+i*i	shift
	+i*i	reduce, (4)
	+i*i	shift
E+	i*i	shift
E+i	*i	reduce, (4)
E+E	*i	

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Plus-Times-A

1. $S \longrightarrow E$
2. $E \longrightarrow E + E$
3. $E \longrightarrow E * E$
4. $E \longrightarrow \texttt{i}$

Input: `i+i*i`

程序代写代做 CS编程辅导

	`i+i*i`	shift
	`+i*i`	reduce, (4)
	`+i*i`	shift
$E+$	`i*i`	shift
$E+\texttt{i}$	`*i`	reduce, (4)
$E+E$	`*i`	shift
$E+E*$	`i`	shift
$E+E*\texttt{i}$		reduce, (4)
$E+E*E$		reduce, (3)
$E+E$		reduce, (2)
E		reduce, (1)
S		ACCEPT

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

Plus-Times-A

1. $S \longrightarrow E$
2. $E \longrightarrow E + E$
3. $E \longrightarrow E * E$
4. $E \longrightarrow \texttt{i}$

Input: `i+i*i`

程序代写代做 CS编程辅导

	`i+i*i`	shift
	`+i*i`	reduce, (4)
	`+i*i`	shift
	`i*i`	shift
E`+i`	`*i`	reduce, (4)
E`+`E	`*i`	

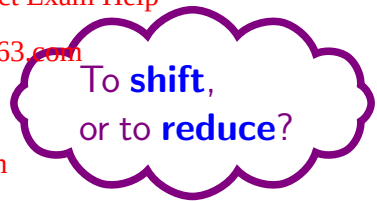
WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com



Plus-Times-A

1. $S \longrightarrow E$
2. $E \longrightarrow E + E$
3. $E \longrightarrow E * E$
4. $E \longrightarrow \texttt{i}$

Input: `i+i*i`

程序代写代做 CS编程辅导

	i+i*i	shift
	+i*i	reduce, (4)
	+i*i	shift
	i*i	shift
E+i	*i	reduce, (4)
E+E	*i	**reduce, (2)**
E	*i	shift
E*	i	shift
E*i		reduce, (4)
E*E		reduce, (3)
E		reduce, (1)
S		ACCEPT

WeChat: cstutorcs
Assignment Project Exam Help
Email: tutorcs@163.com
QQ: 749389476
https://tutorcs.com

⟶ **shift-reduce conflict**

Also:

reduce-reduce conflict: letters on top of stack correspond to $>$ one production rule.

Unix/Linux tools

Yacc

- **Y**et **A**nother **C**ompiler-**C**ompiler
- a parser-generator
- Input: a Context-Free Grammar
- Output: parser, in file `y.tab.c`
- Typically used with a lexical analyser, e.g., Lex.

Lex

- **Lex**ical Analyser
- Input: regular expression for each token ...
- Output: lexical analyser, in file `lex.yy.c`

Both are widely available in Unix/Linux

程序代写代做 CS编程辅导

WeChat: cstutorcs

Assignment Project Exam Help

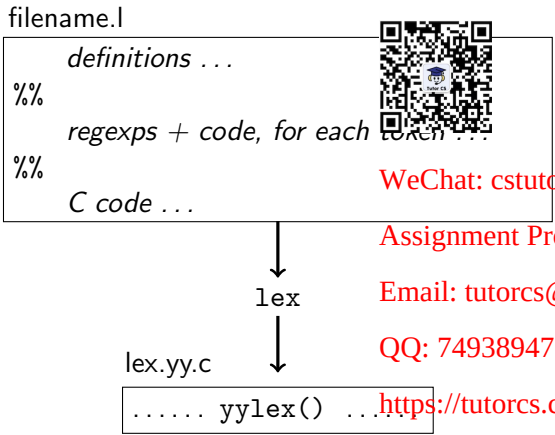
Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Lex: lexical analysis

程序代写代做 CS编程辅导



WeChat: cstutorcs

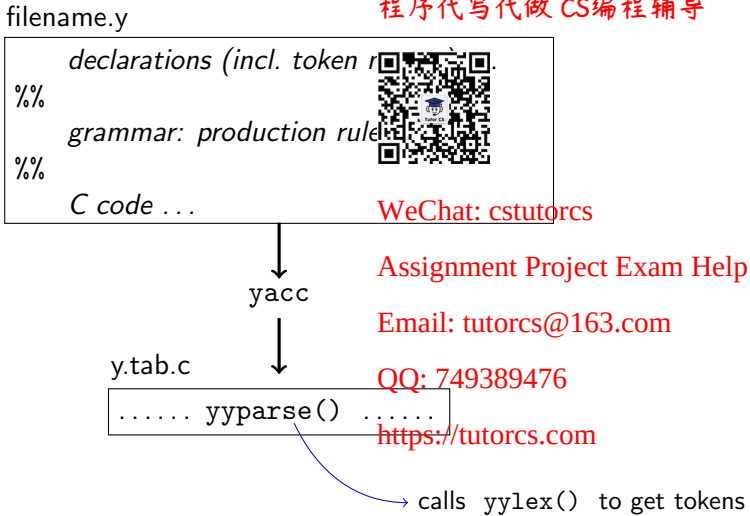
Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Yacc: parser generation



程序代写代做 CS编程辅导

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Lex & Yacc

程序代写代做 CS编程辅导

- Compile `y.tab.c` and `l` using, say, `cc`.
- yields an executable parser.
- It can evaluate as it parses.
- See Assignment 2.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com

Conflict resolution in Yacc:

- Shift-reduce: shift
- Reduce-reduce: use the rule listed first.

Revision

程序代写代做 CS编程辅导

- Construct a parse tree for given string and grammar.
- Understand how a Shift-reduce Parser works.
- Start using Lex and Yacc.

WeChat: cstutorcs

Assignment Project Exam Help

Email: tutorcs@163.com

QQ: 749389476

https://tutorcs.com