# CS480
# Translators

What is Bottom Up Parsing?

Chap. 4

Oregon State University

# Quiz #6 (Answers)

$S \underset{rm}{\Rightarrow} 0S1 \Rightarrow 00S11 \Rightarrow 000111$

- S->0S1 | 01  Indicate the handle for:
  - 000111
  - 00S11

right sentential form

- S->SS+ | SS* | a  Indicate the handle for:
  - SSS+a*+
  - SS+a*a+
  - aaa*a++

$S \underset{rm}{\Rightarrow} SS+ \Rightarrow SSS*+ \Rightarrow SSa*+$
$\Rightarrow S\boxed{SS+}a*+$

- Give a bottom up parse for 000111 and aaa*a++

# LR(0) Conditions

on all symbols

1. For any configurating set containing the item A –> u•xv there is no complete item B –> w• in that set. In the tables, this translates to no shift-reduce conflict on any state. This means the successor function from that set either shifts to a new state or reduces, but not both.

2. There is at most one complete item A –> u• in each configurating set. This translates to no reduce-reduce conflict on any state. The successor function has at most one reduction.

$E \to E + T$
$E \to T$
$T \to (E)$
$T \to id$

# Create a LR(0) Parse Table

| State on Stack | id | + | ( | ) | $ | E | T |
|---|---|---|---|---|---|---|---|
| 0 | s4 | | s3 | | | 1 | 2 |
| 1 | | s5 | | | accept | | |
| 2 | r2 | r2 | r2 | r2 | r2 | | |
| 3 | s4 | | s3 | | | 6 | 2 |
| 4 | r4 | r4 | r4 | r4 | r4 | | |
| 5 | s4 | | s3 | | | | 8 |
| 6 | | s5 | | s7 | | | |
| 7 | r3 | r3 | r3 | r3 | r3 | | |
| 8 | r1 | r1 | r1 | r1 | r1 | | |

- Let’s parse: id + (id)

# What if we change the grammar?

E'->E

1 E -> E**+**T

2 E -> T

3 T -> **(**E**)**

4 T -> **id**

5 T -> **id[**E**]**

LR(0) grammar

why not LR(0)?

added a production

$I_0$ $E' \to .E$ $I_1$
$E \to .E+T$ $I_1$
$E \to .T$ $I_2$
$T \to .(E)$ $I_3$
$T \to .id$ $I_4$
$T \to .id[E]$ $I_4$

$I_1$ $E' \to E.$ Accept
$E \to E.+T$ $I_5$

$I_2$ $E \to T.$ R2

# Collection of Configurating Sets

| *Configurating set* | *Successor* |
|---|---|
| I0: E' –> •E | I1 |
| 1 E –> •E+T | I1 |
| 2 E –> •T | I2 |
| 3 T –> •(E) | I3 |
| 4 T –> •id | I4 |
| 5 T –> •id[E] | I4 |
| I1: E' –> E• | Accept |
| E –> E•+T | I5 |
| I2: E –> T• | Reduce 2 |
| I3: T –> (•E) | I6 |
| E –> •E+T | I6 |
| E –> •T | I2 |
| T –> •(E) | I3 |
| T –> •id | I4 |
| T –> •id[E] | I4 |
| I4: T –> id• | Reduce 4 |
| T –> id•[E] | I9 |

+, ), ], $

| *Configurating set* | *Successor* |
|---|---|
| I5: E –> E+•T | I8 |
| T –> •(E) | I3 |
| T –> •id | I4 |
| T –> •id[E] | I4 |
| I6: T –> (E•) | I7 |
| E –> E•+T | I5 |
| I7: T –> (E)• | Reduce 3 |
| I8: E –> E+T• | Reduce 1 |
| I9: T –> id[•E] | I10 |
| E –> •E+T | I10 |
| E –> •T | I2 |
| T –> •(E) | I3 |
| T –> •id | I4 |
| T –> •id[E] | I4 |
| I10: T –> id[E•] | I11 |
| E –> E•+T | I5 |
| I11: T –> id[E]• | Reduce 5 |

Oregon State University

# Construct SLR(1) Table

1. Construct F = $\{I_0, I_1, \ldots I_n\}$, the collection of configurating sets for G'.

2. State i is determined from $I_i$. The parsing actions for the state are determined as follows:

   a) If A –> $\underline{u}$• is in $I_i$ then set Action[i,a] to reduce A –> $\underline{u}$ for all a in Follow(A) (A not equal to S').

   b) If S' –> S• is in $I_i$ then set Action[i,$] to accept.

   c) If A –> $\underline{u}$•a$\underline{v}$ is in $I_i$ and successor($I_i$, a) = $I_j$, then set Action[i,a] to shift j (a is a terminal).

3. The goto transitions for state i are constructed for all nonterminals A using the rule: If successor($I_i$, A) = $I_j$, then Goto [i, A] = j.

4. All entries not defined by rules 2 and 3 are errors.

5. The initial state is the one constructed from the configurating set containing S' –>S.

# What are the Follows?

Is it SLR(1)?

E'->E

E -> E**+**T

E -> T

T -> **(**E**)**

T -> **id**

T -> **id[**E**]**

- Follow(E)? = { +, ), ], $ }
- Follow(T)?

start

# SLR(1) Parse Table

| State on Stack | id | + | ( | ) | [ | ] | $ | E | T |
|---|---|---|---|---|---|---|---|---|---|
| 0 | s4 | | s3 | | | | | 1 | 2 |
| 1 | | s5 | | | | | accept | | |
| 2 | | r2 | | r2 | | r2 | r2 | | |
| 3 | s4 | | s3 | | | | | 6 | 2 |
| 4 | | r4 | | r4 | s9 | r4 | r4 | | |
| 5 | s4 | | s3 | | | | | | 8 |
| 6 | | s5 | | s7 | | | | | |
| 7 | | r3 | | r3 | | r3 | r3 | | |
| 8 | | r1 | | r1 | | r1 | r1 | | |
| 9 | s4 | | s3 | | | | | 10 | 2 |
| 10 | | s5 | | | | s11 | | | |
| 11 | | r5 | | r5 | | r5 | r5 | | |

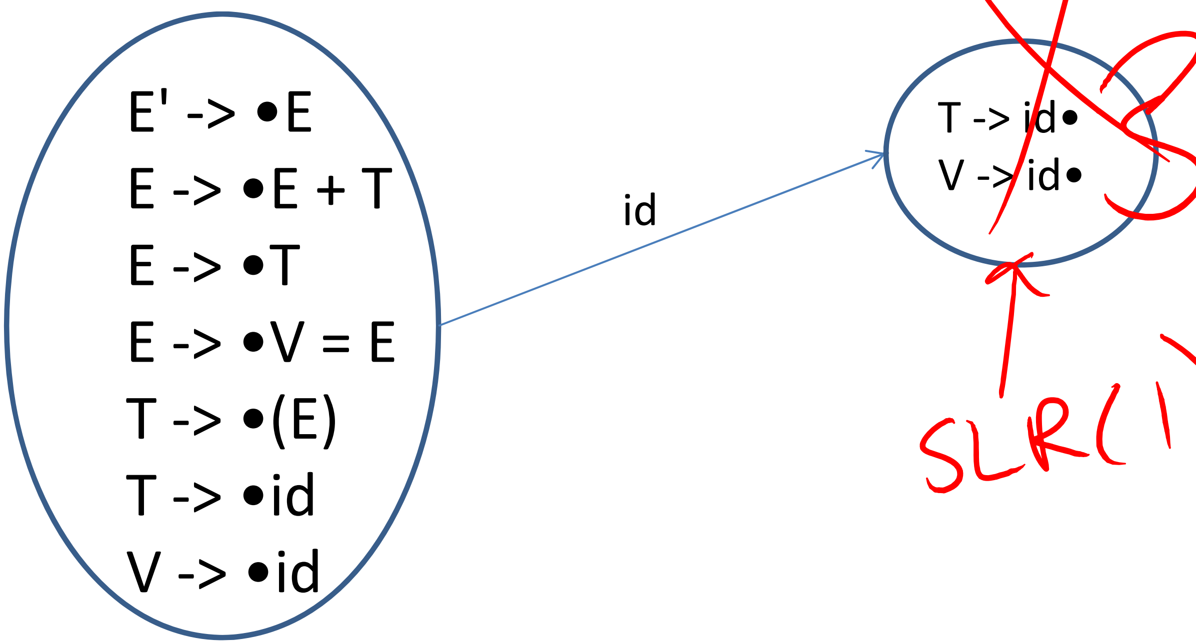
# Let’s consider another example

E' –> E

E –> E + T | T | V = E

T –> (E) | id

V –> id




~~LR(0)~~

Follow(E) = {+, ), $}

Follow(T) 〃

Follow(V) = {=}

SLR(1)

# SLR(1) Conditions

1. For any item A –> u•xv in the set, with terminal x, there is no complete item B –>w• in that set with x in Follow(B). In the tables, this translates no shift-reduce conflict on any state. This means the successor function for x from that set either shifts to a new state or reduces, but not both.

2. For any two complete items A –> u• and B –> v• in the set, the follow sets must be disjoint, e.g. Follow(A) ∩ Follow(B) is empty. This translates to no reduce-reduce conflict on any state. If more than one nonterminal could be reduced from this set, it must be possible to uniquely determine which using only one token of lookahead.

OSU Oregon State University

# Quiz #7

Follow(S) = {$}

- Determine if the grammar is LR(0) or SLR(1)

S' -> S

1 *S*->**real** *IDLIST*

2 *IDLIST*->*IDLIST*, *ID*

3 *IDLIST*->*ID*

4 *ID*->**A**|**B**|**C**|**D**

$I_0$ S' -> .S $I_1$
S -> .real IDLIST $I_2$

$I_1$ S' -> S. Accept

$I_2$ S -> real .IDLIST $I_3$
IDLIST -> .IDLIST , ID $I_3$
-> .ID $I_4$
ID -> .A|B|C|D $I_5$

- Construct the corresponding parse table for the grammar.
- Show how you would parse **real A, B, C**

$I_3$ S -> real IDLIST . R1
IDLIST -> IDLIST . , ID $I_6$