Syntax Analysis
2nd Phase of Compiler Construction

Section 2.1: Bottom up parsing

BOTTOM-UP PARSING

- A **bottom-up parser** creates the parse tree of the given input starting from leaves towards the root.
- A bottom-up parser tries to find the right-most derivation of the given input in the reverse order.

$S \Rightarrow \ldots \Rightarrow \omega$ (the right-most derivation of ω)

$\leftarrow$ (the bottom-up parser finds the right-most derivation in the reverse order)

BOTTOM-UP PARSING

- Bottom-up parsing is also known as **shift-reduce parsing** because its two main actions are **shift** and **reduce**.
  - At each **shift** action, the current symbol in the input string is pushed to a stack.
  - At each **reduction** step, the symbols at the top of the stack (this symbol sequence is the right side of a production) is replaced by the non-terminal at the left side of that production.
  - There are two more actions: accept and error.

SHIFT-REDUCE PARSING

- A shift-reduce parser tries to reduce the given input string into the starting symbol.

  a string $\xrightarrow[\text{reduced to}]{}$ the starting symbol

- At each reduction step, a substring of the input matching to the right side of a production rule is replaced by the non-terminal at the left side of that production rule.
- If the substring is chosen correctly, the right most derivation of that string is created in the reverse order.

  Rightmost Derivation: $S \overset{*}{\underset{rm}{\Rightarrow}} \omega$

  Shift-Reduce Parser finds: $\omega \overset{rm}{\Leftarrow} \ldots \overset{rm}{\Leftarrow} S$

SHIFT-REDUCE PARSING -- EXAMPLE

$S \rightarrow aABb$

$A \rightarrow aA \mid a$

$B \rightarrow bB \mid b$

input string: aaabb

aaAbb

aAbb $\Downarrow$ reduction

aABb

S

$$S \underset{rm}{\Rightarrow} aABb \underset{rm}{\Rightarrow} aAbb \underset{rm}{\Rightarrow} aaAbb \underset{rm}{\Rightarrow} aaabb$$

Right Sentential Forms

- How do we know which substring to be replaced at each reduction step?

HANDLE

- Informally, a **handle** of a string is a substring that matches the right side of a production rule.
  - But not every substring matches the right side of a production rule is handle

- A **handle** of a right sentential form γ ($\equiv \alpha\beta\omega$) is
  a production rule $A \to \beta$ and a position of γ
  where the string β may be found and replaced by A to produce
  the previous right-sentential form in a rightmost derivation of γ.

$$S \overset{*}{\underset{rm}{\Rightarrow}} \alpha A\omega \underset{rm}{\Rightarrow} \alpha\beta\omega$$

- If the grammar is unambiguous, then every right-sentential form of the grammar has exactly one handle.

Handle Pruning

- A right-most derivation in reverse can be obtained by **handle-pruning**.

$$S=\gamma_0 \underset{rm}{\Rightarrow} \gamma_1 \underset{rm}{\Rightarrow} \gamma_2 \underset{rm}{\Rightarrow} \dots \underset{rm}{\Rightarrow} \gamma_{n-1} \underset{rm}{\Rightarrow} \gamma_n = \omega$$

input string

- Start from γ_n, find a handle $A_n \rightarrow \beta_n$ in γ_n, and replace β_n by A_n to get γ_{n-1}.
- Then find a handle $A_{n-1} \rightarrow \beta_{n-1}$ in γ_{n-1}, and replace β_{n-1} by A_{n-1} to get γ_{n-2}.
- Repeat this, until we reach S.
- A left-to-right, bottom-up parse works by iteratively searching for a handle, then reducing the handle

A Stack Implementation of A Shift-Reduce Parser

- There are four possible actions of a shift-parser action:

  1. **Shift** : The next input symbol is shifted onto the top of the stack.
  2. **Reduce**: Replace the handle on the top of the stack by the non-terminal.
  3. **Accept**: Successful completion of parsing.
  4. **Error**: Parser discovers a syntax error, and calls an error recovery routine.

- Initial stack contains only the end-marker $.
- The end of the input string is marked by the end-marker $.

A Shift-Reduce Parser

$E \rightarrow E+T \mid T$
$T \rightarrow T*F \mid F$
$F \rightarrow (E) \mid id$

Right-Most Sentential Form	Reducing Production
<u>id</u>+id*id	$F \rightarrow id$
<u>F</u>+id*id	$T \rightarrow F$
<u>T</u>+id*id	$E \rightarrow T$
E+<u>id</u>*id	$F \rightarrow id$
E+<u>F</u>*id	$T \rightarrow F$
E+T*<u>id</u>	$F \rightarrow id$
E+<u>T*F</u>	$T \rightarrow T*F$
<u>E+T</u>	$E \rightarrow E+T$
E	

Right-Most Derivation of **"id+id*id"**

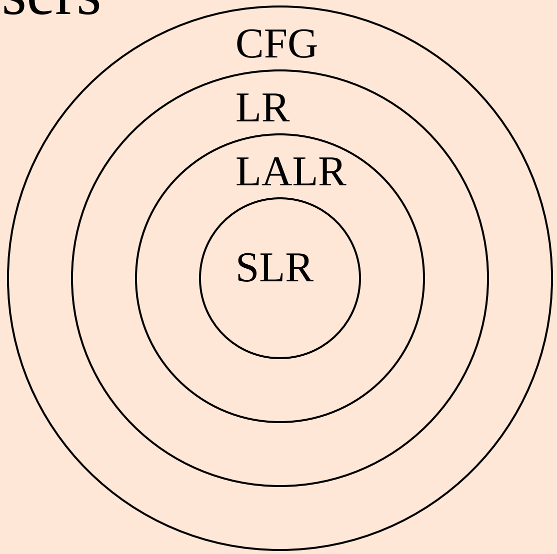
$E \Rightarrow E+T$
$\Rightarrow E+T*F$
$\Rightarrow E+T*id$
$\Rightarrow E+F*id$
$\Rightarrow E+id*id$
$\Rightarrow T+id*id$
$\Rightarrow F+id*id$
$\Rightarrow id+id*id$

<u>Handles</u> are red and underlined in the right-sentential forms.

SHIFT-REDUCE PARSERS

There are two main categories of shift-reduce parsers

1. **Operator-Precedence Parser**
   - simple, but only a small class of grammars.
2. **LR-Parsers**
   - covers wide range of grammars.
     - SLR – simple LR parser
     - LR – most general LR parser
     - LALR – intermediate LR parser (lookahead LR parser)
   - SLR, LR and LALR work in similar manner, only their parsing tables are different.



LR Parsers

- The most powerful shift-reduce parsing (yet efficient) is:

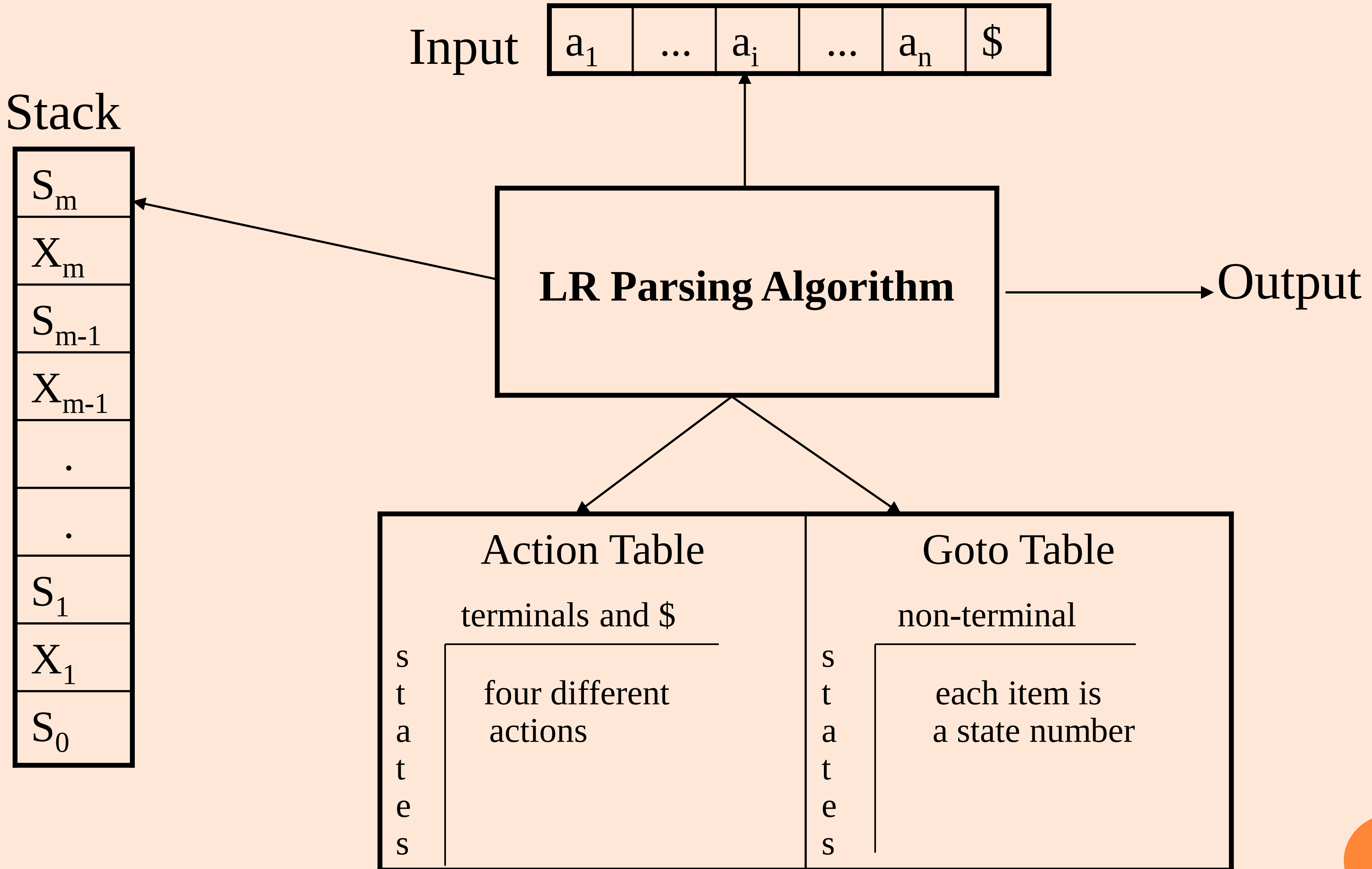
LR(k) parsing.

left to right scanning	right-most derivation	k lookahead (k is omitted ➔ it is 1)

- LR parsing is attractive because:
  - LR parsing is most general non-backtracking shift-reduce parsing.
  - The class of grammars that can be parsed using LR methods is a proper superset of the class of grammars that can be parsed with predictive parsers.

    LL(1)-Grammars $\subset$ LR(1)-Grammars
  - An LR-parser can detect a syntactic error as soon as it is possible to do so.

LR Parsing Algorithm



A Configuration of LR Parsing Algorithm

- A configuration of a LR parsing is:

$$(S_o\ X_1\ S_1 \ldots X_m\ \boxed{S_m,\ a_i}\ a_{i+1} \ldots a_n\ \$\ )$$

Stack — Rest of Input

- Each state symbol summarizes the information contained in the stack below it, and the combination of state symbol on top of the stack and input symbol are used to index the parsing table and determine the shift-reduce action.

- S_m and a_i decides the parser action by consulting the parsing action table.

  (*Initial Stack* contains just S_o)

- A configuration of a LR parsing represents the right sentential form:

$$X_1 \ldots X_m\ a_i\ a_{i+1} \ldots a_n\ \$$$

Actions of a LR-Parser

1. **shift s** -- shifts the next input symbol and the state **s** onto the stack

   $(S_o\ X_1\ S_1 \ldots X_m\ S_m,\ a_i\ a_{i+1} \ldots a_n\ \$) \Rightarrow (S_o\ X_1\ S_1 \ldots X_m\ S_m\ a_i\ S,\ a_{i+1} \ldots a_n\ \$)$

2. **reduce A→β** (or **`rn`** where n is a production number)
   - pop 2*|**β**| (=r) items from the stack; let us assume that **β** = $Y_1Y_2 \ldots Y_r$
   - then push **A** and **s** where **s=goto[s_{m-r},A]**

   $(S_o\ X_1\ S_1 \ldots X_m\ S_m,\ a_i\ a_{i+1} \ldots a_n\ \$) \Rightarrow (S_o\ X_1\ S_1 \ldots X_{m-r}\ S_{m-r}\ A\ s,\ a_i \ldots a_n\ \$)$

   - In fact, $Y_1Y_2 \ldots Y_r$ is a handle.

     $X_1 \ldots X_{m-r}\ A\ a_i \ldots a_n\ \$ \Rightarrow X_1 \ldots X_m\ Y_1 \ldots Y_r\ a_i\ a_{i+1} \ldots a_n\ \$$
   - Output is the reducing production; reduce A→β

3. **Accept** – Parsing successfully completed
4. **Error** -- Parser detected an error (an empty entry in the action table)

The Closure Operation -- Example

$E' \to E$

$E \to E+T$

$E \to T$

$T \to T*F$

$T \to F$

$F \to (E)$

$F \to id$

$closure(\{E' \to \bullet E\}) =$

$\{ \; E' \to \bullet E$ kernel items

$E \to \bullet E+T$

$E \to \bullet T$

$T \to \bullet T*F$

$T \to \bullet F$

$F \to \bullet (E)$

$F \to \bullet id \; \}$

THE CLOSURE OPERATION

- If ***I*** is a set of LR(0) items for a grammar G, then ***closure(I)*** is the set of LR(0) items constructed from I by the two rules:

  1. Initially, every LR(0) item in I is added to closure(I).
  2. If $A \rightarrow \alpha \bullet B\beta$ is in closure(I) and $B \rightarrow \gamma$ is a production rule of G; then $B \rightarrow \bullet \gamma$ will be in the closure(I).
     We will apply this rule until no more new LR(0) items can be added to closure(I).

- Kernel items: which includes the initial item $S' \rightarrow \bullet S$, and all items whose dots are not at the left end
- Nonkernel Items, which have their dots at the left end

CONSTRUCTION OF THE CANONICAL LR(0) COLLECTION

- To create the SLR parsing tables for a grammar G, we will create the canonical LR(0) collection of the grammar G'.

- ***Algorithm***:

C is { closure({S'→• S}) }

repeat the followings until no more set of LR(0) items can be added to ***C***.

  for each I in ***C*** and each grammar symbol X

    if goto(I,X) is not empty and not in ***C***

      add goto(I,X) to ***C***

- goto function is a DFA on the sets in C.

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

Shift • From left of ‘E’ to right of E

I_1
E' → E• **(Complete)**
E → E•+T

Add Productions of E since there is • before E.

Add Productions of T since there is • before T.

Add Productions of F since there is • before F.

Initial Set
E' → •E
1) E → •E+T
2) E → •T
3) T → •T*F
4) T → •F
5) F → • (E)
6) F → •id

The Canonical LR(0) Collection -- Example

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

Shift • From left of 'T' to right of T

I_2

$E \rightarrow T\bullet$ **(Complete)**
$T \rightarrow T\bullet *F$

Initial Set

$E' \rightarrow \bullet E$
1) $E \rightarrow \bullet E+T$
2) $E \rightarrow \bullet T$
3) $T \rightarrow \bullet T*F$
4) $T \rightarrow \bullet F$
5) $F \rightarrow \bullet (E)$
6) $F \rightarrow \bullet id$

The Canonical LR(0) Collection -- Example

I_0
E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

Shift • From left of 'F' to right of F

I_3
T → F• **(Complete)**

Initial Set
E' → •E
1) E → •E+T
2) E → •T
3) T → •T*F
4) T → •F
5) F → • (E)
6) F → • id

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

Shift • From left of ' (' to right of '('

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

Initial Set

$E' \rightarrow \bullet E$
1) $E \rightarrow \bullet E+T$
2) $E \rightarrow \bullet T$
3) $T \rightarrow \bullet T*F$
4) $T \rightarrow \bullet F$
5) $F \rightarrow \bullet (E)$
6) $F \rightarrow \bullet id$

Add Productions of E since there is • before E.

Add Productions of T since there is • before T.

Add Productions of F since there is • before F.

The Canonical LR(0) Collection -- Example

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

Shift • From left of 'id' to right of 'id'

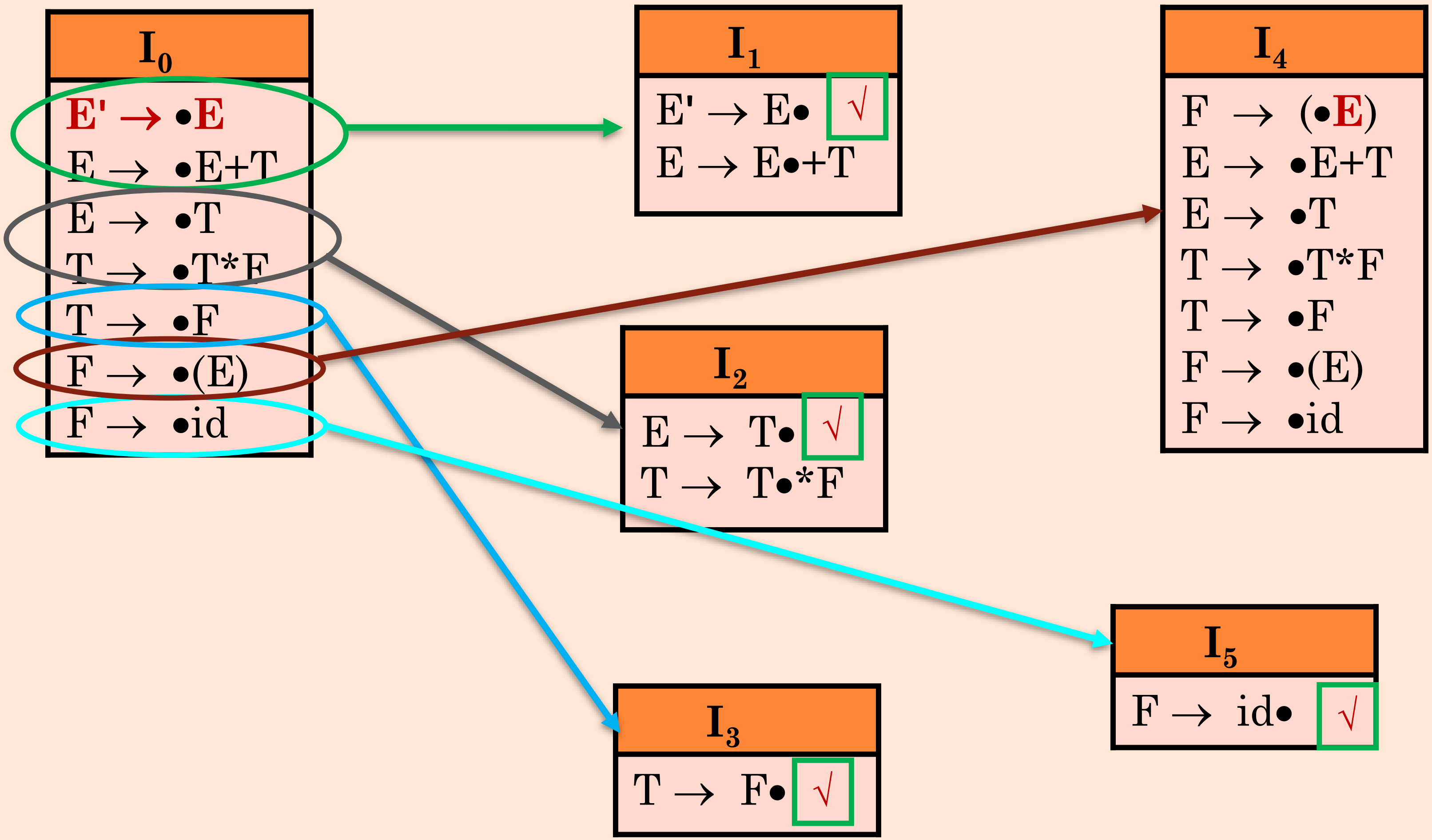
I_5

F → id• **(Complete)**

Initial Set

E' → •E
1) E → •E+T
2) E → •T
3) T → •T*F
4) T → •F
5) F → • (E)
6) F → •id

THE CANONICAL LR(0) COLLECTION -- EXAMPLE



THE CANONICAL LR(0) COLLECTION

I_0

$E' \to \bullet E$
$E \to \bullet E+T$
$E \to \bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet(E)$
$F \to \bullet id$

I_4

$F \to (\bullet E)$
$E \to \bullet E+T$
$E \to \bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet(E)$
$F \to \bullet id$

I_1

$E' \to E\bullet$ √
$E \to E\bullet+T$

Shift • From left of '+' to right of '+'

I_6

$E \to E+\bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet (E)$
$F \to \bullet id$

Initial Set

$E' \to \bullet E$
1) $E \to \bullet E+T$
2) $E \to \bullet T$
3) $T \to \bullet T*F$
4) $T \to \bullet F$
5) $F \to \bullet (E)$
6) $F \to \bullet id$

I_2

$E \to T\bullet$ √
$T \to T\bullet*F$

I_3

$T \to F\bullet$ √

I_5

$F \to id\bullet$ √

Add Productions of T since there is • before T.

Add Productions of F since there is • before F.

The Canonical LR(0) Collection

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1

E' → E• √
E → E•+T

I_2

E → T• √
T → T•*F

I_3

T → F• √

I_4

F → (•E)
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_5

F → id• √

I_6

E → E+•T
T → •T*F
T → •F
F → • (E)
F → •id

I_7

T → T*•F
F → •(E)
F → •id

Shift • From left of '*' to right of '*'

Initial Set

E' → •E
1) E → •E+T
2) E → •T
3) T → •T*F
4) T → •F
5) F → • (E)
6) F → •id

Add Productions of F since there is • before F.

THE CANONICAL LR(0) COLLECTION

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ $\boxed{\surd}$
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ $\boxed{\surd}$
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ $\boxed{\surd}$

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ $\boxed{\surd}$

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

Shift • From left of 'E' to right of 'E'

THE CANONICAL LR(0) COLLECTION

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1

E' → E• √
E → E•+T

I_2

E → T• √
T → T•*F

Shift • From left of 'T' to right of 'T'

I_3

T → F• √

I_4

F → (•**E**)
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

I_5

F → id• √

I_6

E → E+•**T**
T → •T*F
T → •F
F → • (E)
F → •id

I_7

T → T*•**F**
F → •(E)
F → •id

I_8

F → (E•)
E → E•+T

THE CANONICAL LR(0) COLLECTION

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

Shift • From left of 'F' to right of 'F'

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

THE CANONICAL LR(0) COLLECTION

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1

E' → E• √
E → E•+T

I_2

E → T• √
T → T•*F

I_3

T → F• √

Shift • From left of '(' to right of '('

I_4

F → (.E)
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

I_5

F → id• √

I_6

E → E+•T
T → •T*F
T → •F
F → • (E)
F → •id

I_7

T → T*•F
F → •(E)
F → •id

I_8

F → (E•)
E → E•+T

The Canonical LR(0) Collection

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

Shift • From left of 'id' to right of 'id'

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

THE CANONICAL LR(0) COLLECTION

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1

E' → E• √
E → E•+T

I_2

E → T• √
T → T•*F

I_3

T → F• √

I_4

F → (•**E**)
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

I_5

F → id• √

I_6

E → E+•**T**
T → •T*F
T → •F
F → • (E)
F → •id

I_7

T → T*•**F**
F → •(E)
F → •id

I_8

F → (E•)
E → E•+T

Shift • From left of 'T' to right of 'T'

I_9

E → E+T• √
T → T•*F

THE CANONICAL LR(0) COLLECTION

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

Shift • From left of 'F' to right of 'F'

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

I_9

$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

The Canonical LR(0) Collection

I_0
E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1
E' → E• √
E → E•+T

I_2
E → T• √
T → T•*F

I_3
T → F• √

I_4
F → (•**E**)
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

I_5
F → id• √

Shift • From left of '(' to right of '('

I_6
E → E+•**T**
T → •T*F
T → •F
F → • (E)
F → •id

I_7
T → T*•**F**
F → •(E)
F → •id

I_8
F → (E•)
E → E•+T

I_9
E → E+T• √
T → T•*F

The Canonical LR(0) Collection

I_0
$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1
$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2
$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3
$T \rightarrow F\bullet$ √

I_4
$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5
$F \rightarrow id\bullet$ √

Shift • From left of ‘id’ to right of ‘id’

I_6
$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7
$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8
$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

I_9
$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

The Canonical LR(0) Collection

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

Shift • From left of ‘F’ to right of ‘F’

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

I_9

$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

I_{10}

$T \rightarrow T*F\bullet$ √

THE CANONICAL LR(0) COLLECTION

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

Shift • From left of '(' to right of '('

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

I_9

$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

I_{10}

$T \rightarrow T*F\bullet$ √

The Canonical LR(0) Collection

I_0

$E' \to \bullet E$
$E \to \bullet E+T$
$E \to \bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet(E)$
$F \to \bullet id$

I_1

$E' \to E\bullet$ √
$E \to E\bullet+T$

I_2

$E \to T\bullet$ √
$T \to T\bullet*F$

I_3

$T \to F\bullet$ √

I_4

$F \to (\bullet E)$
$E \to \bullet E+T$
$E \to \bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet (E)$
$F \to \bullet id$

I_5

$F \to id\bullet$ √

Shift • From left of 'id' to right of 'id'

I_6

$E \to E+\bullet T$
$T \to \bullet T*F$
$T \to \bullet F$
$F \to \bullet (E)$
$F \to \bullet id$

I_7

$T \to T*\bullet F$
$F \to \bullet(E)$
$F \to \bullet id$

I_8

$F \to (E\bullet)$
$E \to E\bullet+T$

I_9

$E \to E+T\bullet$ √
$T \to T\bullet*F$

I_{10}

$T \to T*F\bullet$ √

The Canonical LR(0) Collection

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

Shift • From left of ')' to right of ')'

I_9

$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

I_{10}

$T \rightarrow T*F\bullet$ √

I_{11}

$F \rightarrow (E)\bullet$ √

THE CANONICAL LR(0) COLLECTION

I_0

$E' \rightarrow \bullet E$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_1

$E' \rightarrow E\bullet$ √
$E \rightarrow E\bullet+T$

I_2

$E \rightarrow T\bullet$ √
$T \rightarrow T\bullet*F$

I_3

$T \rightarrow F\bullet$ √

I_4

$F \rightarrow (\bullet E)$
$E \rightarrow \bullet E+T$
$E \rightarrow \bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_5

$F \rightarrow id\bullet$ √

Shift • From left of '+' to right of '+'

I_6

$E \rightarrow E+\bullet T$
$T \rightarrow \bullet T*F$
$T \rightarrow \bullet F$
$F \rightarrow \bullet (E)$
$F \rightarrow \bullet id$

I_7

$T \rightarrow T*\bullet F$
$F \rightarrow \bullet(E)$
$F \rightarrow \bullet id$

I_8

$F \rightarrow (E\bullet)$
$E \rightarrow E\bullet+T$

I_9

$E \rightarrow E+T\bullet$ √
$T \rightarrow T\bullet*F$

I_{10}

$T \rightarrow T*F\bullet$ √

I_{11}

$F \rightarrow (E)\bullet$ √

THE CANONICAL LR(0) COLLECTION

I_0

- E' → •E
- E → •E+T
- E → •T
- T → •T*F
- T → •F
- F → •(E)
- F → •id

I_1

- E' → E• √
- E → E•+T

I_2

- E → T• √
- T → T•*F

I_3

- T → F• √

I_4

- F → (•E)
- E → •E+T
- E → •T
- T → •T*F
- T → •F
- F → • (E)
- F → •id

I_5

- F → id• √

I_6

- E → E+•T
- T → •T*F
- T → •F
- F → • (E)
- F → •id

I_7

- T → T*•F
- F → •(E)
- F → •id

I_8

- F → (E•)
- E → E•+T

I_9

- E → E+T• √
- T → T•*F

Shift • From left of '*' to right of '*'

I_{10}

- T → T*F• √

I_{11}

- F → (E) • √

The Canonical LR(0) Collection

Initial Set

E' → •E
1) E → •E+T
2) E → •T
3) T → •T*F
4) T → •F
5) F → • (E)
6) F → •id

I_0

E' → •E
E → •E+T
E → •T
T → •T*F
T → •F
F → •(E)
F → •id

I_1

E' → E• √
E → E•+T

I_2

E → T• √
T → T•*F

I_3

T → F• √

I_4

F → (•E)
E → •E+T
E → •T
T → •T*F
T → •F
F → • (E)
F → •id

I_5

F → id• √

I_6

E → E+•T
T → •T*F
T → •F
F → • (E)
F → •id

I_7

T → T*•F
F → •(E)
F → •id

I_8

F → (E•)
E → E•+T

I_9

E → E+T• √
T → T•*F

I_{10}

T → T*F• √

I_{11}

F → (E) • √

CONSTRUCTING SLR PARSING TABLE

(OF AN AUGUMENTED GRAMMAR G')

1. Construct the canonical collection of sets of LR(0) items for G'.
   $C \leftarrow \{I_0, ..., I_n\}$

2. Create the parsing action table as follows
   - If a is a terminal, $A \rightarrow \alpha.a\beta$ in I_i and $goto(I_i,a)=I_j$ then action[i,a] is ***shift j.***
   - If $A \rightarrow \alpha.$ is in I_i, then action[i,a] is ***reduce A→α*** for all a in FOLLOW(A) where $A \neq S'$.
   - If $S' \rightarrow S.$ is in I_i, then action[i,$] is ***accept***.
   - If any conflicting actions generated by these rules, the grammar is not SLR(1).

3. Create the parsing goto table
   - for all non-terminals A, if $goto(I_i,A)=I_j$ then goto[i,A]=j

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

5. Initial state of the parser contains $S' \rightarrow .S$

Constructing SLR Parsing Tables

- An **LR(0) item** of a grammar G is a production of G a dot at the some position of the right side.
- Ex: $A \rightarrow aBb$ Possible LR(0) Items: $A \rightarrow \bullet aBb$

  (four different possibility) $A \rightarrow a \bullet Bb$

  $A \rightarrow aB \bullet b$

  $A \rightarrow aBb \bullet$
- Sets of LR(0) items will be the states of action and goto table of the SLR parser.
- A collection of sets of LR(0) items (**the canonical LR(0) collection**) is the basis for constructing SLR parsers.
- *Augmented Grammar*:

  G' is G with a new production rule $S' \rightarrow S$ where S' is the new starting symbol.

GOTO OPERATION

- If I is a set of LR(0) items and X is a grammar symbol (terminal or non-terminal), then goto(I,X) is defined as follows:
  - If $A \to \alpha \bullet X\beta$ in I then every item in **closure({$A \to \alpha X \bullet \beta$})** will be in goto(I,X).

Example:

$I = \{ \; E' \to \bullet E, \; E \to \bullet E{+}T, \; E \to \bullet T,$
$\quad T \to \bullet T{*}F, \; T \to \bullet F,$
$\quad F \to \bullet (E), \; F \to \bullet id \; \}$

$goto(I,E) = \{ \; E' \to E \bullet, \; E \to E \bullet {+}T \; \}$

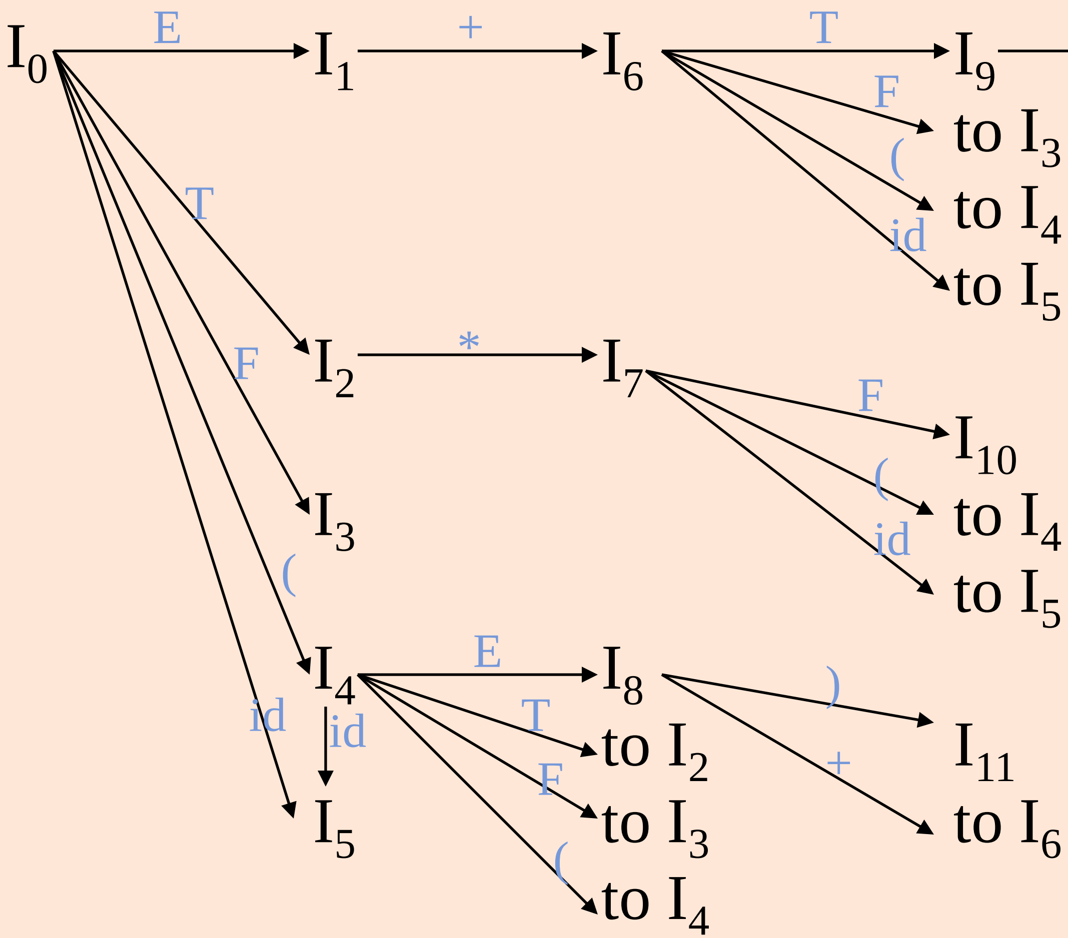
$goto(I,T) = \{ \; E \to T \bullet, \; T \to T \bullet {*}F \; \}$

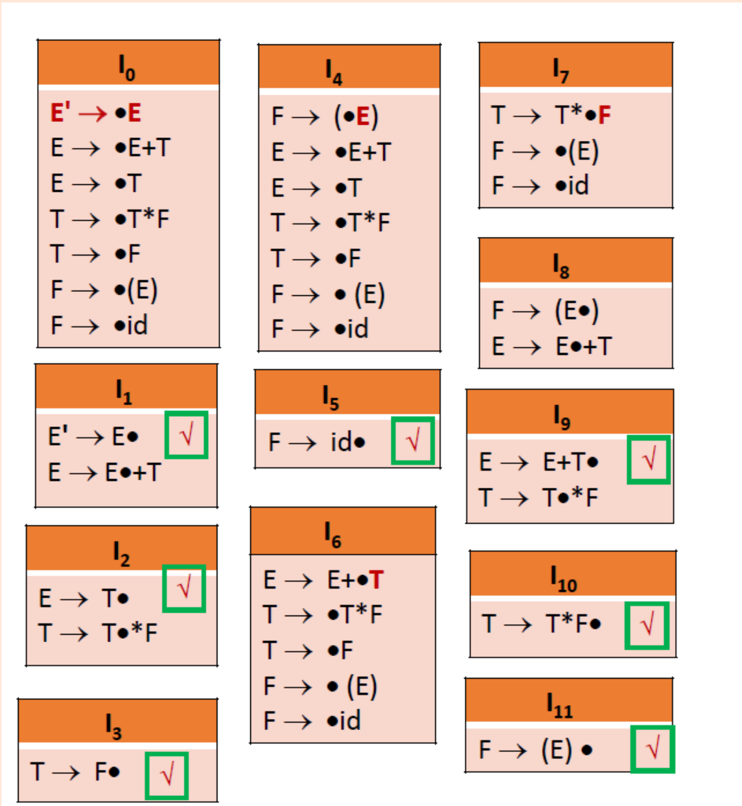
$goto(I,F) = \{ T \to F \bullet \; \}$

$goto(I,() = \{ \; F \to (\bullet E), \; E \to \bullet E{+}T, \; E \to \bullet T, \; T \to \bullet T{*}F, \; T \to \bullet F,$
$\quad F \to \bullet (E), \; F \to \bullet id \; \}$

$goto(I,id) = \{ \; F \to id \bullet \; \}$

Transition Diagram (DFA) of Goto Function





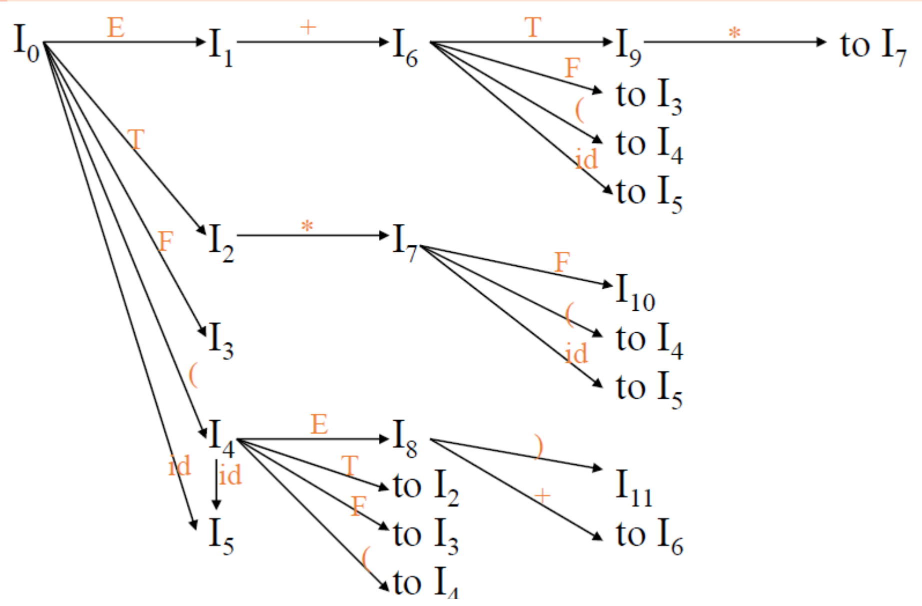
FIRST AND FOLLOW SET

Consider the following Grammar

$E \rightarrow E+T$
$E \rightarrow T$
$T \rightarrow T*F$
$T \rightarrow F$
$F \rightarrow (E)$
$F \rightarrow id$

FIRST(E) = {(,id}
FIRST (T)= {(,id}
FIRST (F)= {(,id}

FOLLOW (E) = {$,),+}
FOLLOW(T) ={$,),+,*}
FOLLOW (F)= {$,),+,*}



I_0
- E' → •E
- E → •E+T
- E → •T
- T → •T*F
- T → •F
- F → •(E)
- F → •id

I_1
- E' → E• √
- E → E•+T

I_2
- E → T• √
- T → T•*F

I_3
- T → F• √

I_4
- F → (•E)
- E → •E+T
- E → •T
- T → •T*F
- T → •F
- F → • (E)
- F → •id

I_5
- F → id• √

I_6
- E → E+•T
- T → •T*F
- T → •F
- F → • (E)
- F → •id

I_7
- T → T*•F
- F → •(E)
- F → •id

I_8
- F → (E•)
- E → E•+T

I_9
- E → E+T• √
- T → T•*F

I_{10}
- T → T*F• √

I_{11}
- F → (E) • √

	ACTION						GOTO		
state	**id**	**+**	*****	**(**	**)**	**$**	**E**	**T**	**F**
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

E' → .E
1) **E → E+T**
2) **E → T**
3) **T → T*F**
4) **T → F**
5) **F → (E)**
6) **F → id**

FOLLOW (E) = {$,),+}
FOLLOW(T) ={$,),+,*}
FOLLOW (F)= {$,),+,*}

- S*i* means shift and stack state *i*
- r*j* means reduce by production numbered *j*
- *acc* means accept state
- blank mean error

(SLR) Parsing Tables for Expression Grammar

1) $E \rightarrow E+T$
2) $E \rightarrow T$
3) $T \rightarrow T*F$
4) $T \rightarrow F$
5) $F \rightarrow (E)$
6) $F \rightarrow id$

- S*i* means shift and stack state *i*
- r*j* means reduce by production numbered *j*
- *acc* means accept state
- blank mean error

Action Table / Goto Table

State	id	+	*	(	)	$		E	T	F
0	s5			s4				1	2	3
1		s6				acc				
2		r2	s7		r2	r2				
3		r4	r4		r4	r4				
4	s5			s4				8	2	3
5		r6	r6		r6	r6				
6	s5			s4					9	3
7	s5			s4						10
8		s6			s11					
9		r1	s7		r1	r1				
10		r3	r3		r3	r3				
11		r5	r5		r5	r5				

Actions of a (S)LR-Parser -- Example

Stack	Input	Action	Output
0	id*id+id$	shift 5	
0id5	*id+id$	reduce by F→id	F→id
0F3 (GOTO)	*id+id$	reduce by T→F	T→F
0T2(GOTO)	*id+id$	shift 7	
0T2*7	id+id$	shift 5	
0T2*7id5	+id$	reduce by F→id	F→id
0T2*7F10 (GOTO)	+id$	reduce by T→T*F	T→T*F
0T2 (GOTO)	+id$	reduce by E→T	E→T
0E1(GOTO)	+id$	shift 6	
0E1+6	id$	shift 5	
0E1+6id5	$	reduce by F→id	F→id
0E1+6F3 (GOTO)	$	reduce by T→F	T→F
0E1+6T9 (GOTO)	$	reduce by E→E+T	E→E+T
0E1	$	accept	

	ACTION							GOTO		
State	**id**	**+**	*****	**(**	**)**	**$**		**E**	**T**	**F**
0	s5			s4				1	2	3
1		s6				**acc**				
2		r2	s7		r2	r2				
3		r4	r4		r4	r4				
4	s5			s4				8	2	3
5		r6	r6		r6	r6				
6	s5			s4					9	3
7	s5			s4						10
8		s6			s11					
9		r1	s7		r1	r1				
10		r3	r3		r3	r3				
11		r5	r5		r5	r5				

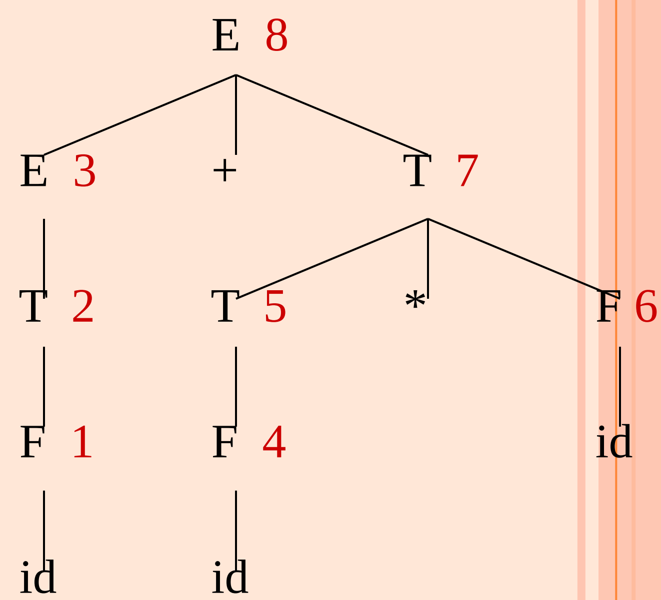
E' → .E
1) E → E+T
2) E → T
3) T → T*F
4) T → F
5) F → (E)
6) F → id

•S*i* means shift and stack state *i*
•r*j* means reduce by production numbered *j*
acc means accept state
•blank mean error

A Stack Implementation of A Shift-Reduce Parser

Stack	**Input**	**Action**
$	id+id*id$	shift
$id	+id*id$	reduce by F $\rightarrow$ id
$F	+id*id$	reduce by T $\rightarrow$ F
$T	+id*id$	reduce by E $\rightarrow$ T
$E	+id*id$	shift
$E+	id*id$	shift
$E+id	*id$	reduce by F $\rightarrow$ id
$E+F	*id$	reduce by T $\rightarrow$ F
$E+T	*id$	shift
$E+T*	id$	shift
$E+T*id	$	reduce by F $\rightarrow$ id
$E+T*F	$	reduce by T $\rightarrow$ T*F
$E+T	$	reduce by E $\rightarrow$ E+T
$E	$	accept

Parse Tree



SLR EXAMPLE -II

Consider the following Grammar
$S \rightarrow (L)$ $S \rightarrow x$ $L \rightarrow S$ $L \rightarrow L,S$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

Shift • From left of 'S' to right of S

I_1
$S' \rightarrow S\bullet$ **(Complete)**

Initial Set
$S' \rightarrow \bullet S$
1) $S \rightarrow \bullet(L)$
2) $S \rightarrow \bullet x$
3) $L \rightarrow \bullet S$
4) $L \rightarrow \bullet L,S$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

Shift • From left of '(' to right of '('

I_2
$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

Initial Set
$S' \rightarrow \bullet S$
1) $S \rightarrow \bullet(L)$
2) $S \rightarrow \bullet x$
3) $L \rightarrow \bullet S$
4) $L \rightarrow \bullet L,S$

I_1	
$S' \rightarrow S\bullet$	$\boxed{\surd}$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

Shift • From left of ‘x’ to right of ‘x’

I_3	
$S \rightarrow x\bullet$	√

I_1	
$S' \rightarrow S\bullet$	√

I_2
$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0

S' → •S
S → •(L)
S → •x

I_1

S' → S• √

I_2

S → (•L)
L → •S
L → •L,S
S → •(L)
S → •x

Shift • From left of 'L' to right of 'L'

I_4

S → (L•)
L → L•,S

I_3

S → x• √

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0

S' → •S
S → •(L)
S → •x

I_4

S → (L•)
L → L•,S

Initial Set

S' → •S
1) S → •(L)
2) S → •x
3) L → •S
4) L → •L,S

I_1

S' → S• √

I_2

S → (•L)
L → •S
L → •L,S
S → •(L)
S → •x

Shift • From left of 'S' to right of 'S'

I_5

L → S• √

I_3

S → x• √

The Canonical LR(0) Collection -- Example

I_0

$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_1

$S' \rightarrow S\bullet$ √

I_2

$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_3

$S \rightarrow x\bullet$ √

Shift • From left of 'L' to right of 'L'

I_4

$S \rightarrow (L\bullet)$
$L \rightarrow L\bullet,S$

I_5

$L \rightarrow S\bullet$ √

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_1
$S' \rightarrow S\bullet$ √

I_2
$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

Shift • From left of '(' to right of '('

I_3
$S \rightarrow x\bullet$ √

I_4
$S \rightarrow (L\bullet)$
$L \rightarrow L\bullet,S$

I_5
$L \rightarrow S\bullet$ √

The Canonical LR(0) Collection -- Example

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_1	
$S' \rightarrow S\bullet$	√

I_2
$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_3	
$S \rightarrow x\bullet$	√

Shift • From left of 'x' to right of 'x'

I_4
$S \rightarrow (L\bullet)$
$L \rightarrow L\bullet,S$

I_5	
$L \rightarrow S\bullet$	√

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0
$S' \rightarrow \bullet S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_1	
$S' \rightarrow S\bullet$	√

I_2
$S \rightarrow (\bullet L)$
$L \rightarrow \bullet S$
$L \rightarrow \bullet L,S$
$S \rightarrow \bullet(L)$
$S \rightarrow \bullet x$

I_3	
$S \rightarrow x\bullet$	√

I_4
$S \rightarrow (L\bullet)$
$L \rightarrow L\bullet,S$

Shift • From left of ')' to right of ')'

I_5	
$L \rightarrow S\bullet$	√

I_6	
$S \rightarrow (L)\bullet$	√

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0

$S' \to \bullet S$
$S \to \bullet(L)$
$S \to \bullet x$

I_1

$S' \to S\bullet$ √

I_2

$S \to (\bullet L)$
$L \to \bullet S$
$L \to \bullet L,S$
$S \to \bullet(L)$
$S \to \bullet x$

I_3

$S \to x\bullet$ √

I_4

$S \to (L\bullet)$
$L \to L\bullet,S$

I_5

$L \to S\bullet$ √

I_6

$S \to (L)\bullet$ √

Shift • From left of ‘,’ to right of ‘,’

I_7

$L \to L,\bullet S$
$S \to \bullet(L)$
$S \to \bullet x$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE

I_0

S' → •S
S → •(L)
S → •x

I_1

S' → S• √

I_2

S → (•L)
L → •S
L → •L,S
S → •(L)
S → •x

I_3

S → x• √

I_4

S → (L•)
L → L•,S

I_5

L → S• √

I_6

S → (L)• √

I_7

L → L,•S
S → •(L)
S → •x

Shift • From left of 'S' to right of 'S'

I_8

L → L,S• √

The Canonical LR(0) Collection -- Example

I_0

$S' \rightarrow \bullet S$

$S \rightarrow \bullet(L)$

$S \rightarrow \bullet x$

I_1

$S' \rightarrow S\bullet$ $\boxed{\checkmark}$

I_2

$S \rightarrow (\bullet L)$

$L \rightarrow \bullet S$

$L \rightarrow \bullet L,S$

$S \rightarrow \bullet(L)$

$S \rightarrow \bullet x$

I_3

$S \rightarrow x\bullet$ $\boxed{\checkmark}$

I_4

$S \rightarrow (L\bullet)$

$L \rightarrow L\bullet,S$

I_5

$L \rightarrow S\bullet$ $\boxed{\checkmark}$

I_6

$S \rightarrow (L)\bullet$ $\boxed{\checkmark}$

I_7

$L \rightarrow L,\bullet S$

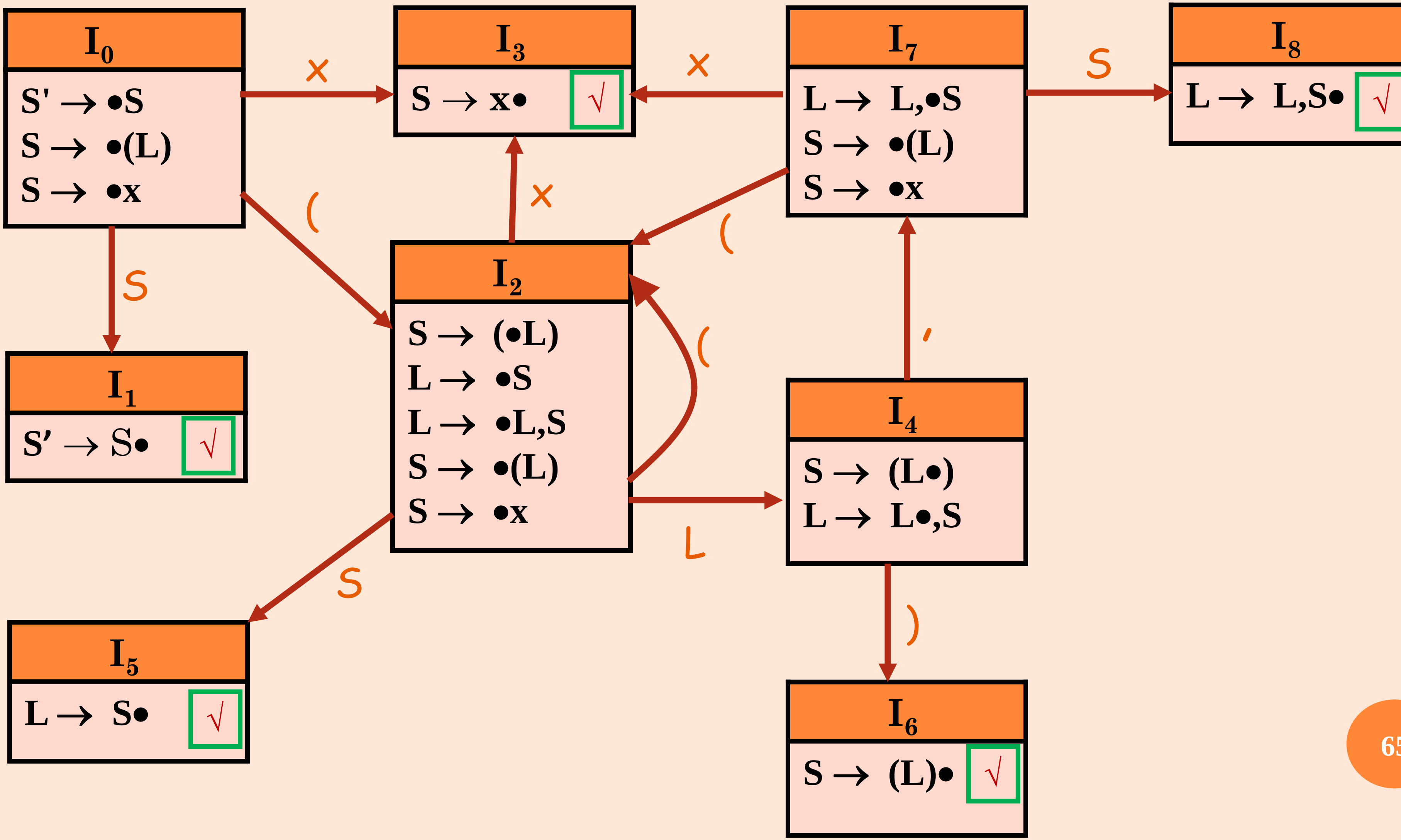
$S \rightarrow \bullet(L)$

$S \rightarrow \bullet x$

I_8

$L \rightarrow L,S\bullet$ $\boxed{\checkmark}$

THE CANONICAL LR(0) COLLECTION -- EXAMPLE



SHIFT/REDUCE AND REDUCE/REDUCE CONFLICTS

- If a state does not know whether it will make a shift operation or reduction for a terminal, we say that there is a **shift/reduce conflict**.
- If a state does not know whether it will make a reduction operation using the production rule `i` or `j` for a terminal, we say that there is a **reduce/reduce conflict**.

- If the SLR parsing table of a grammar G has a conflict, we say that grammar is not SLR grammar.

First and Follow Set

Consider the following Grammar
$S \rightarrow L=R$
$S \rightarrow R$
$L \rightarrow *R$
$L \rightarrow id$
$R \rightarrow L$

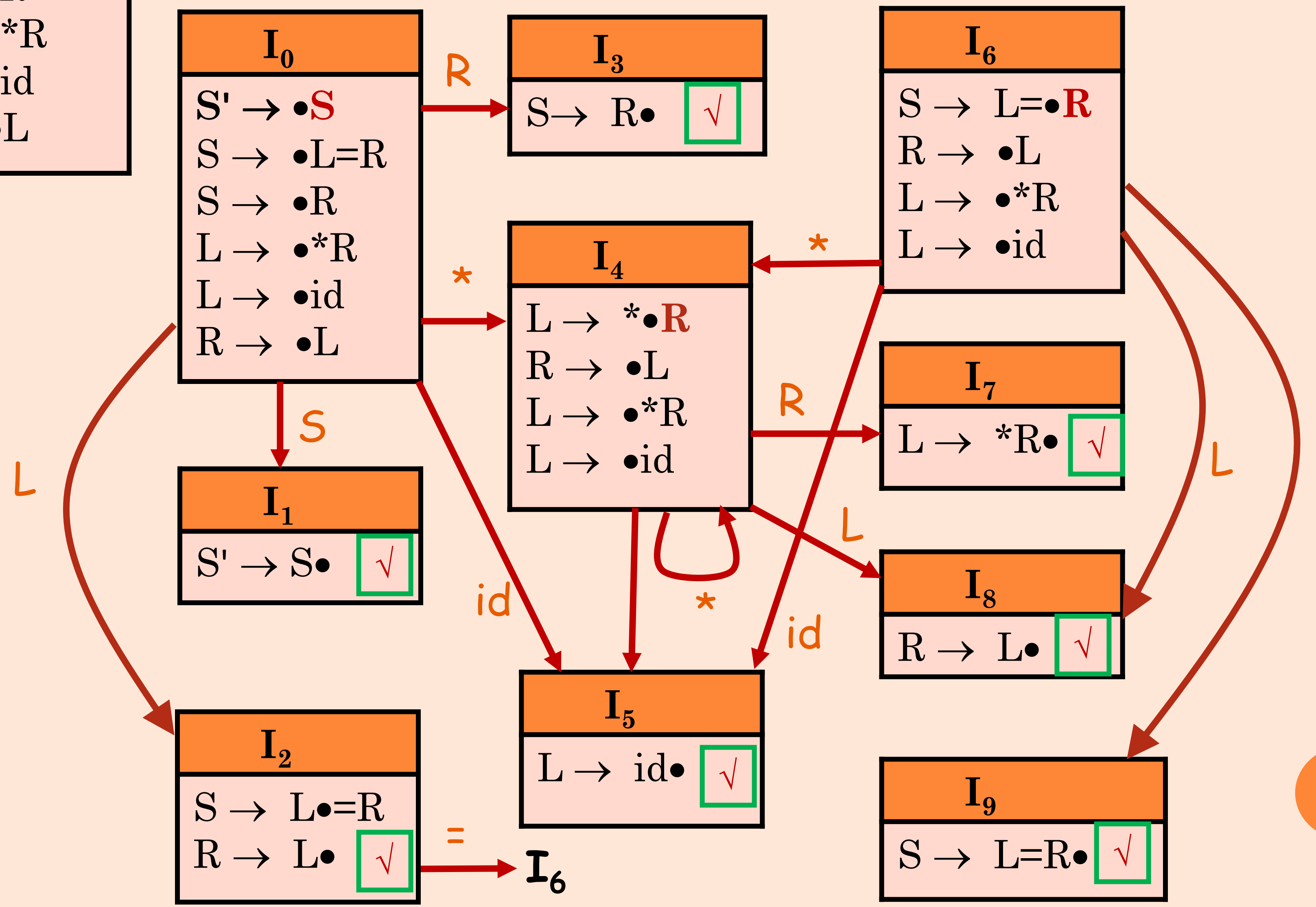
FIRST(S) = {*,id}
FIRST (L)= {*,id}
FIRST (R)= {*,id}

FOLLOW (S) = {$}
FOLLOW(L) ={$,=}
FOLLOW (R)= {$,=}

CONFLICT EXAMPLE

Initial Grammar

S' → •**S**
1) S → •L=R
2) S → •R
3) L → •*R
4) L → •id
5) R → •L



(SLR) PARSING TABLES FOR EXPRESSION GRAMMAR

Initial Grammar

S' → .S
1) S → .L=R
2) S → .R
3) L → .*R
4) L → .id
5) R → .L

	ACTION					GOTO		
State	**id**	**=**	*****	**$**		**S**	**L**	**R**
0	s_5		s_4			1	2	3
1				acc				
2		s_6 r_5		r_5				
3		r_2		r_2				
4	s_5		s_4				8	7
5		r_4		r_4				
6	s_5		s_4				8	9
7		r_3		r_3				
8		r_5		r_5				
9		r_1		r_1				

Shift/ Reduce Conflict

I_2

S → L•=R
R → L• √

=

I_6

S → L=•R
R → •L
L → •*R
L → •id

FOLLOW (S) = {$}
FOLLOW(L) ={$,=}
FOLLOW (R)= {$,=}

Part II will include LR(1) parsers and error recovery