

# Checkin

## Review - LR Parsers

Describe the difference between an LL(1) and LR(1) parser. Which is more powerful?

University of Kansas | Drew Davidson

*ECS 665*

**COMPILER**

***CONSTRUCTION***

SLR Parser Construction

# Administrivia

## Housekeeping

# Last Time

## LR Parsers

### You Should Know

- How an LR Parser differs from an LL Parser
- Vaguely what the parser automaton does

## LR Parsers

- Concept
- Theory
- Operation

## 2 Amazin' facts:

- All viable prefixes for an LR grammar can be captured by a Finite State Automaton!
- A stack of states can track our position



Parsing

# Review – Bottom-Up Parsing

## LR Parser Construction

May have multiple branches in flight at the same time

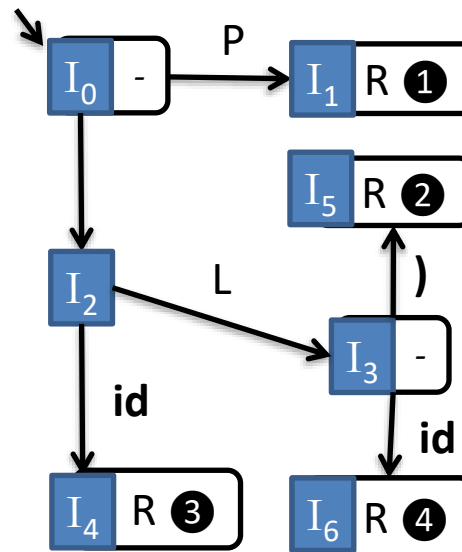
## Operations

- Shift – ingest the lookahead token
- Reduce – digest branches

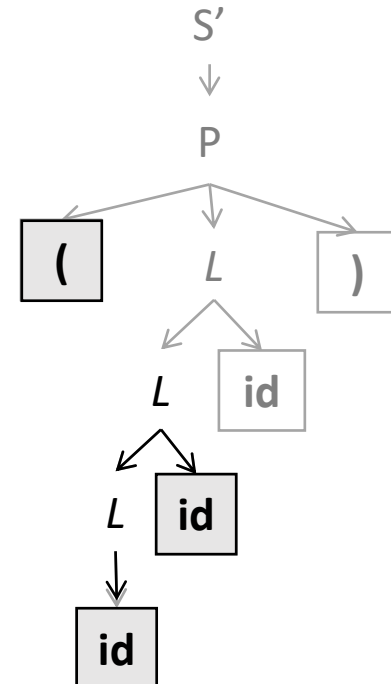
### Grammar G

- ①  $S' ::= P$
- ②  $P ::= ( L )$
- ③  $L ::= id$
- ④  $L ::= L id$

### G Parser Automaton



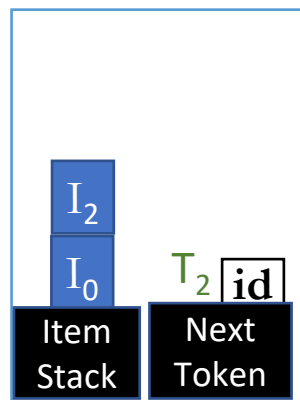
### Correct Parse Tree



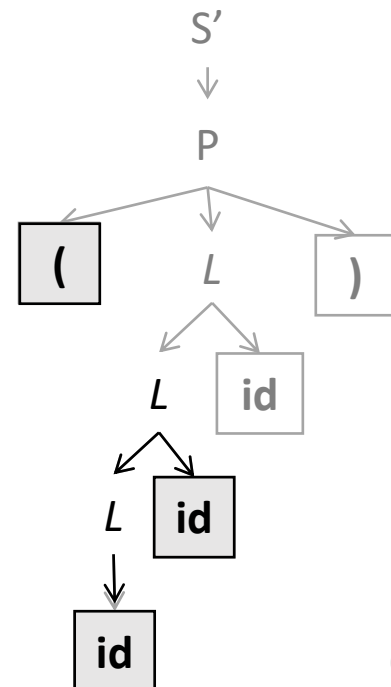
# Recall Intuition: Delayed Commitment

SLR Parsing

- Delay committing to a parent production until all children have been seen
- The stack tracks automaton states... but what do those states really mean?



## Correct Parse Tree



# Today's Outline

## SLR Parsing

### Some LR Context

- Peek inside the automaton

### Build parser automaton

- Closure set
- GoTo set

### Put Parser into Table form

- Action types



Parsing

# A Quick Piece of Vocab: Items

SLR Parsing

## An *item* represents progress through a production

- What constitutes an item depends on LR parser type
- FSM states are sets of items

“Basic” item form

$$Z \rightarrow \alpha \bullet X \beta$$

In a Z production

Already seen symbol(s)  $\alpha$

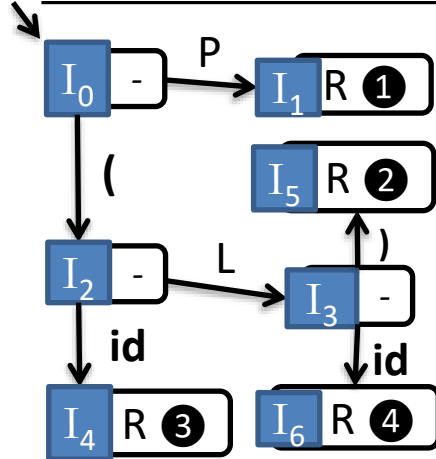
About to produce symbol X

After X, will produce symbols(s)  $\beta$

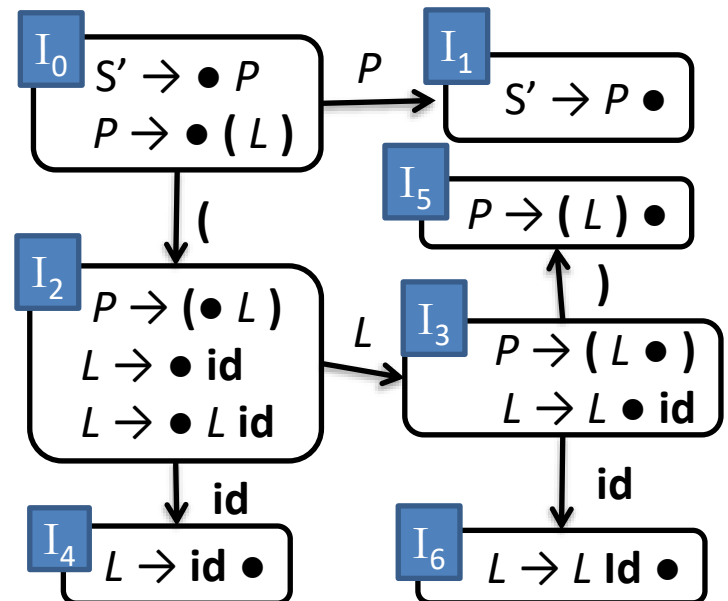
### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$

### G Parser Automaton



### G Parser Automaton (item sets shown)





# A Quick Piece of Vocab: Items

SLR Parsing

## An *item* represents progress through a production

- What constitutes an item depends on LR parser type
- FSM states are sets of items

### “Basic” item form

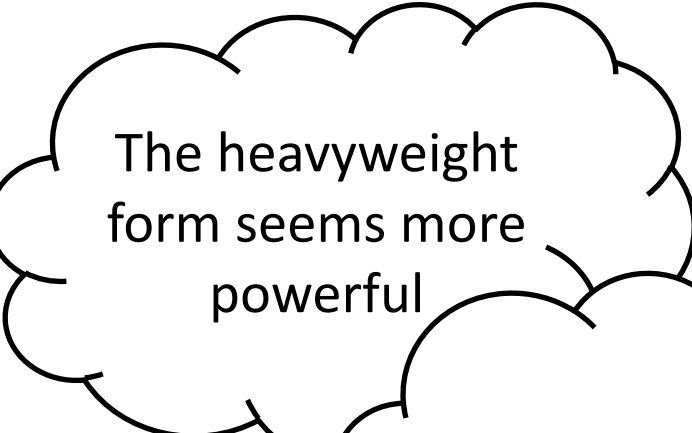
$$Z \rightarrow \alpha \bullet X \beta$$

In a Z production

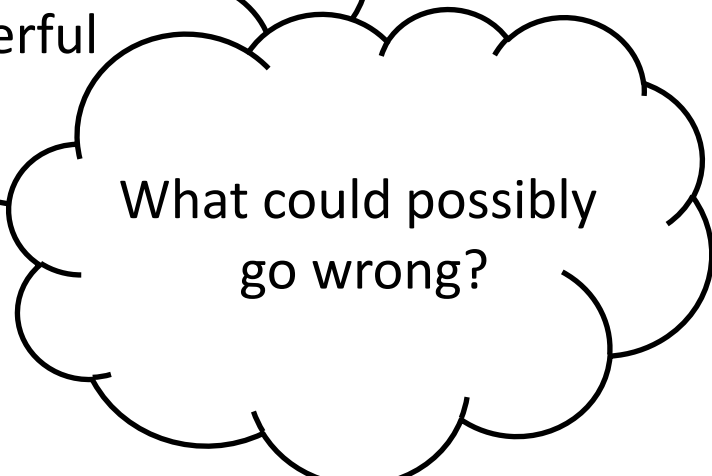
Already seen symbol(s)  $\alpha$

About to produce symbol X

After X, will produce symbols(s)  $\beta$



The heavyweight form seems more powerful



What could possibly go wrong?

### “Heavyweight” item form

$$Z \rightarrow \alpha \bullet X \beta, t/u/v$$

As above, but lookahead token could be **t** or **u** or **v**

# FSM State Space Explosion

## LR Parser Construction

**Bad news: The FSM can become impractically large**



# Avoiding State Space Explosion

## Basic LR Parser Construction

### Different Types of LR parsers:

- LR(1) “*canonical LR*”
- LALR
- **SLR** *Our focus*
- LR(0)

### Automaton works the same way

- Size of automaton varies
- Bigger automata are more precise



# Today's Outline

## SLR Parsing

### Some LR Context

### Build parser automaton

- Closure set
- GoTo set

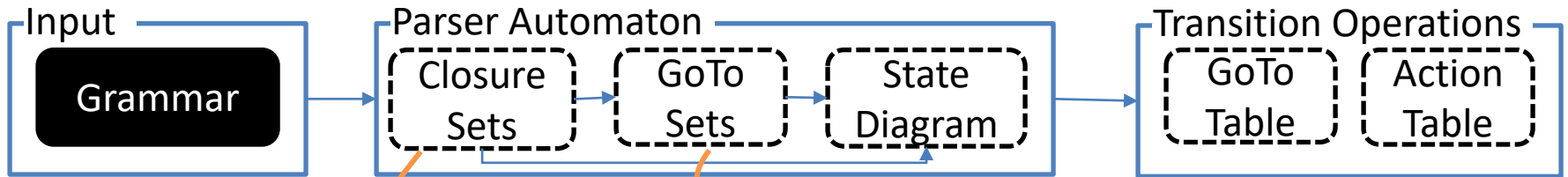
### Put Parser into Table form

- Action types



**Parsing**

# Parser-Building Workflow



*Define the states of the automaton*

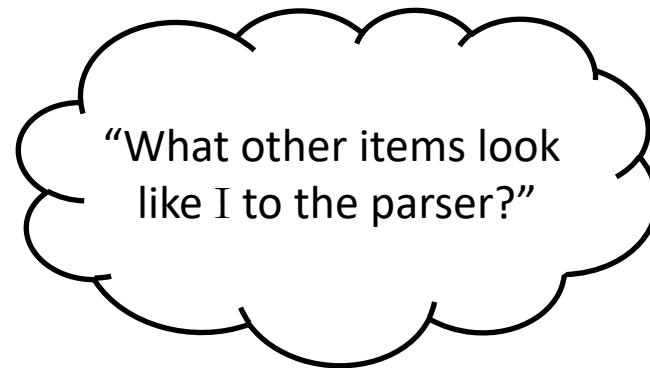
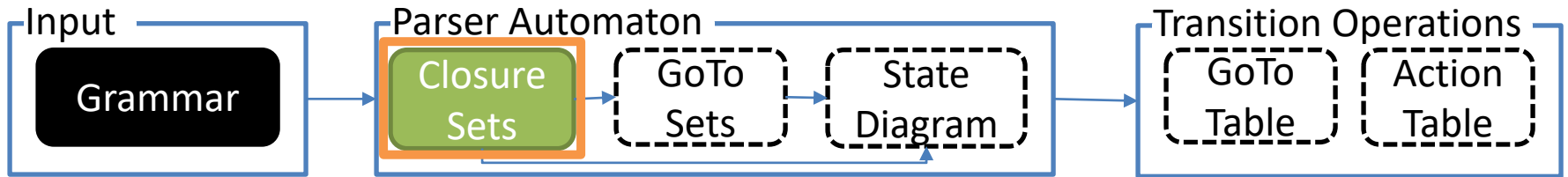
*Define the edges of the automaton*

$$S' \rightarrow \cdot X$$

$$S' ::= X$$

$$X ::= a X$$
$$| b$$

# Parser-Building Workflow: Closure From Grammar



## Building Closure(I)

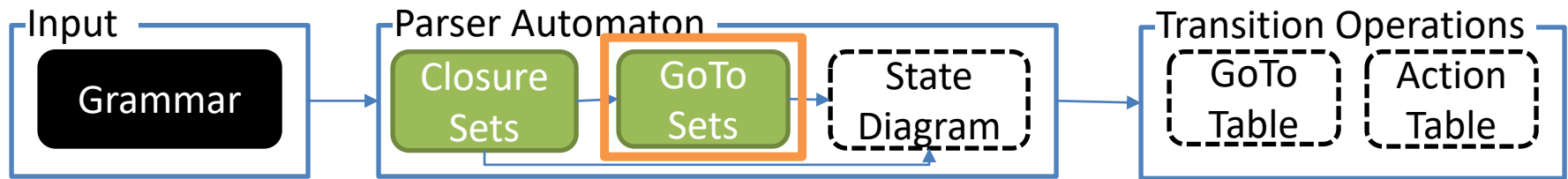
Add I and repeat until saturation:

if  $X \rightarrow \alpha \bullet Z \beta$  is in  $\text{Closure}(I)$ :

for all  $Z ::= \gamma$  productions:

add  $Z \rightarrow \bullet \gamma$

# Parser-Building Workflow: GoTo From Closure



If we were in a state  $I$  item,  
where might we be after parsing  $\hat{\pi}$ ?

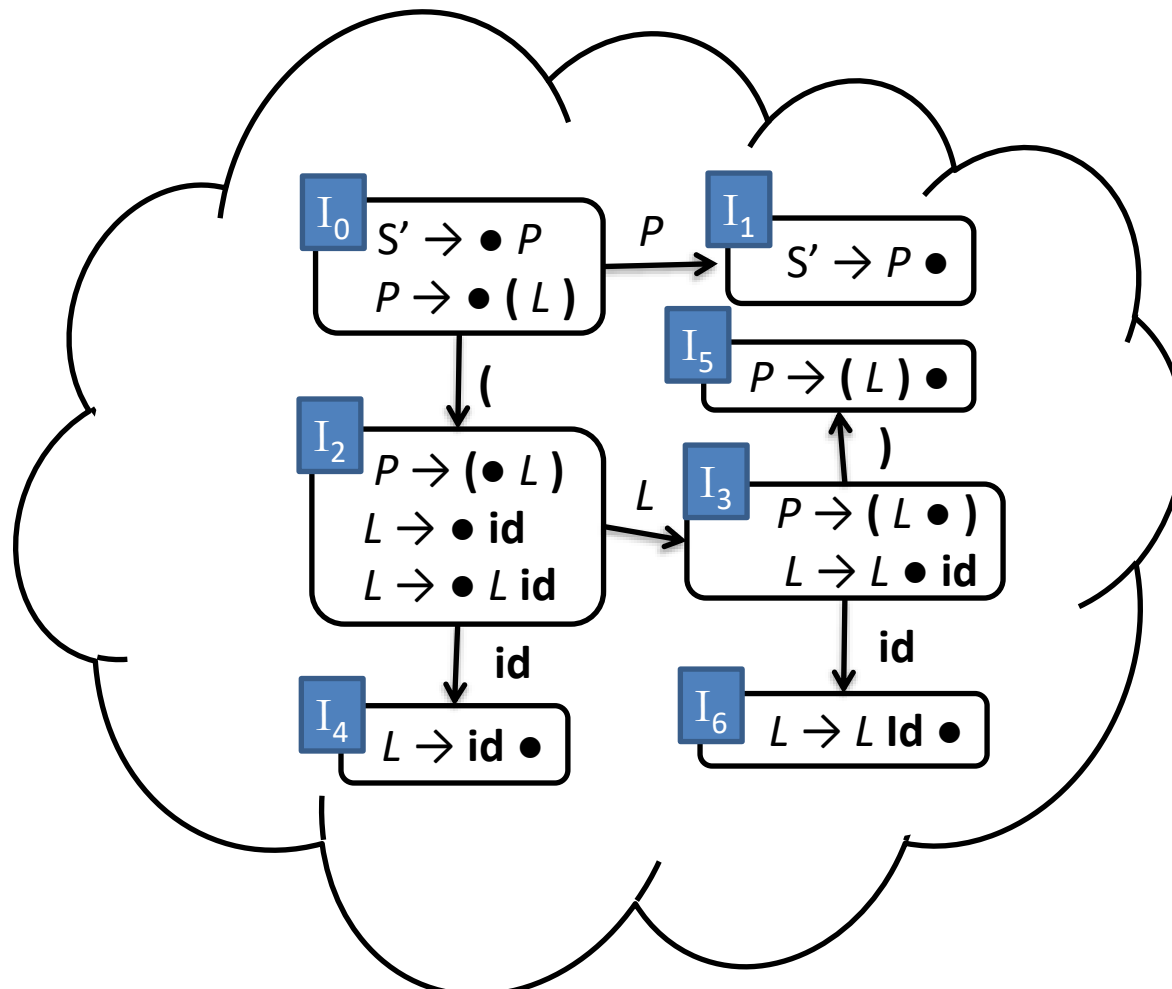
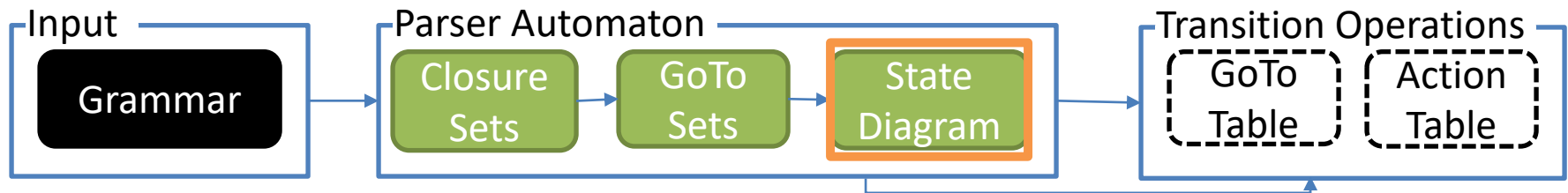
## Building GoTo relation from $I_j$

if  $(X \rightarrow \alpha \bullet \hat{\pi} \beta$  is in  $I_j$ )

set  $\text{GoTo}(I_j, \hat{\pi}) = I_k$  where

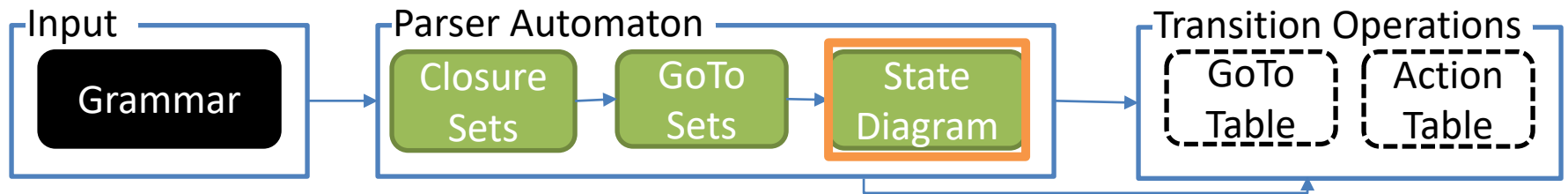
$$I_k = \text{Closure}(X \rightarrow \alpha \hat{\pi} \bullet \beta)$$

# Parser-Building Workflow: FSM from GoTo + Closure





# Parser-Building Workflow: FSM from GoTo + Closure



## LR Automaton Construction

Add new start  $S'$  and  $S' \rightarrow S$

Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$

Saturate FSM:

for each symbol  $\hat{\pi}$  s.t. item in state  $j$   
contains  $A \rightarrow \alpha \bullet \hat{\pi} \beta$ :

add transition from state  $j$  to  
the state for  $\text{GoTo}(j, X)$

# Building an LR Parser: Example

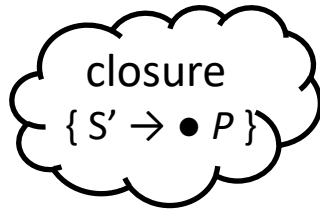
## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to G
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to GoTo
  - Add nodes according to Closure

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= \gamma(L)$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



### GoTo(I,X) =

State that represents  
Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### Closure(I):

Begin with I  
Repeat until saturation:  
if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

Start with the  $I_0$  state

Identify state  
 $\text{Closure}(\{S' \rightarrow \bullet P\})$   
=  
 $S' \rightarrow \bullet P$   
 $(\forall P ::= \bullet \gamma)$   
 $P \rightarrow \bullet (L)$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

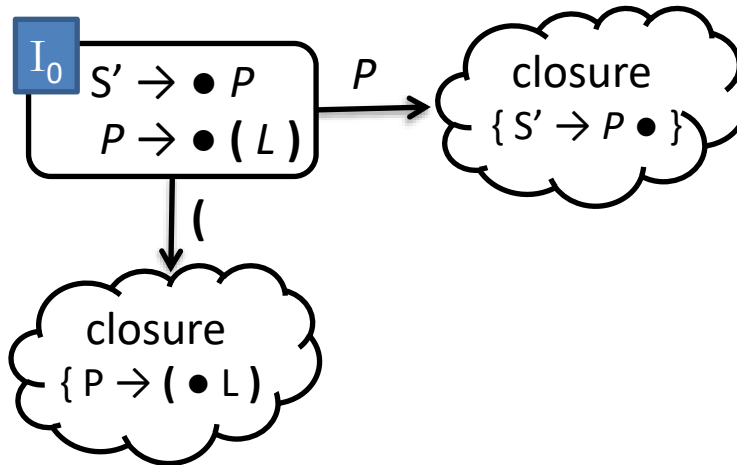
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar $G$

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Start with the  $I_0$  state

Identify state  
 $\text{Closure}(\{S' \rightarrow \bullet P\})$   
 $=$   
 $S' \rightarrow \bullet P$   
 $(\forall P ::= \bullet \gamma)$   
 $P \rightarrow \bullet ( L )$

Add Edges:

$\text{GoTo}(I_0, P) = \text{closure}(\{S' \rightarrow P \bullet\})$   
 $\text{GoTo}(I_0, ( ) = \text{closure}(\{P \rightarrow ( \bullet L )\})$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:  
Add edges according to  $\text{GoTo}$   
Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

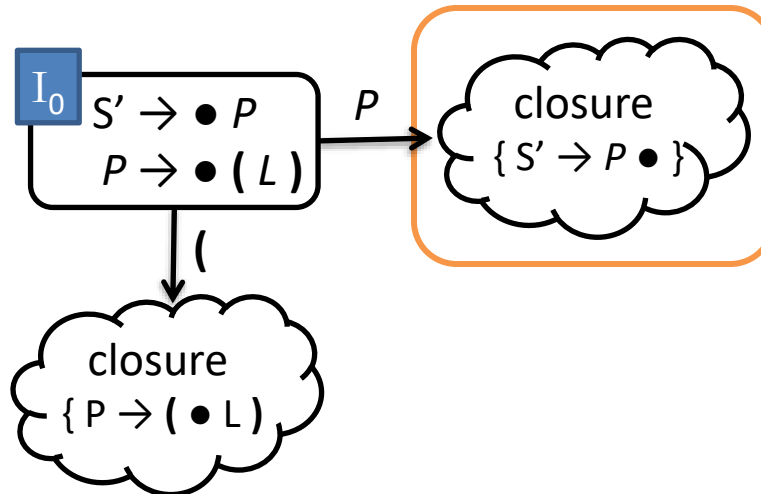
State that represents  
Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
Repeat until saturation:  
if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{S' \rightarrow P \bullet\})$   
 $=$   
 $\{S' \rightarrow P \bullet\}$   
(nothing else)

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

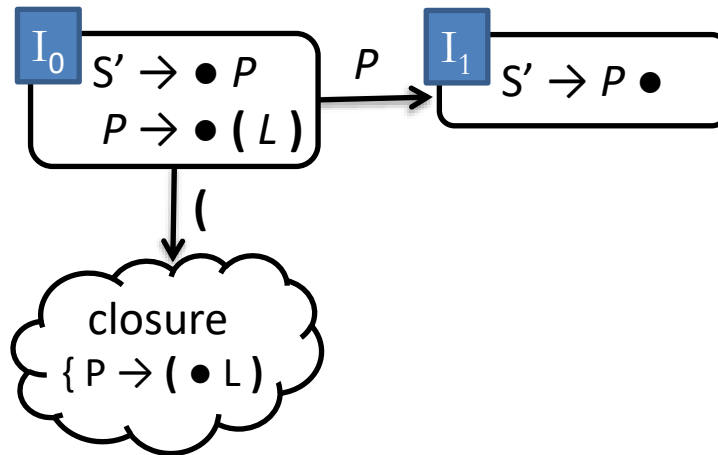
State that represents  
Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
Repeat until saturation:  
if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{ S' \rightarrow P \bullet \})$   
 $=$   
 $\{ S' \rightarrow P \bullet \}$   
(nothing else)

Identify edges from  $I_1$   
(none)

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:  
Add edges according to  $\text{GoTo}$   
Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

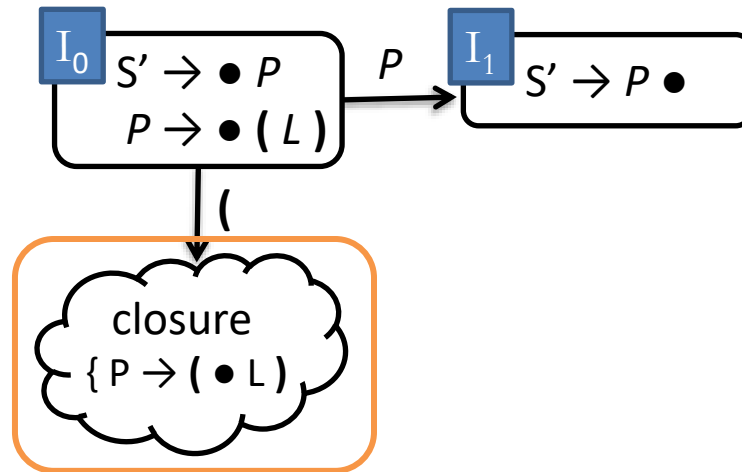
State that represents  
Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
Repeat until saturation:  
if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state

$\text{Closure}(\{P \rightarrow ( \bullet L )\})$   
=  
 $P \rightarrow ( \bullet L )$   
 $L \rightarrow \bullet \text{id}$   
 $L \rightarrow \bullet L \text{id}$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

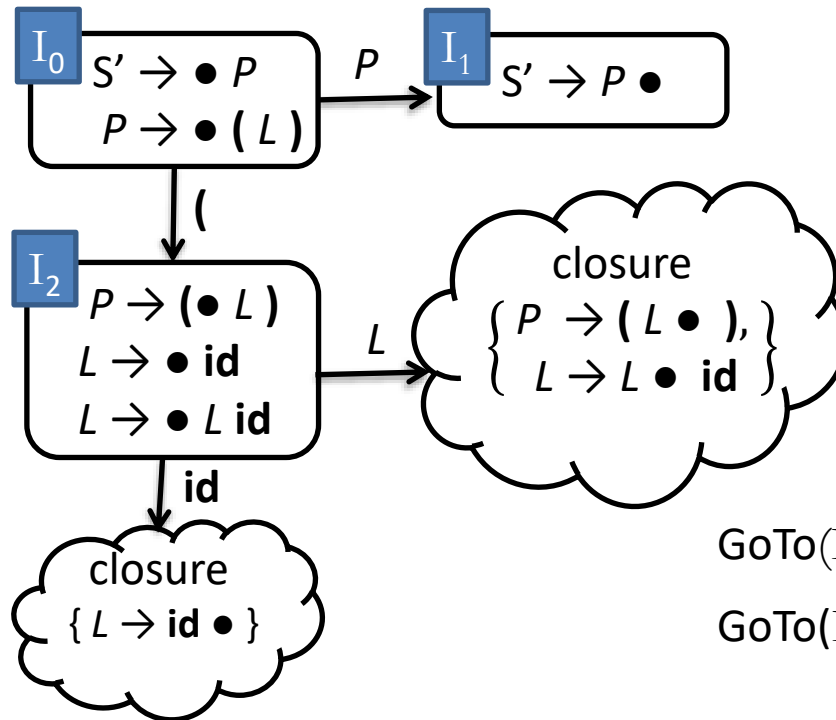
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{ P \rightarrow ( \bullet L ) \})$   
 $=$   
 $P \rightarrow ( \bullet L )$   
 $L \rightarrow \bullet \text{id}$   
 $L \rightarrow \bullet L \text{id}$

Edges out of  $I_2$

$\text{GoTo}(I_2, \text{id}) = \text{closure}(\{L \rightarrow \text{id} \bullet\})$

$\text{GoTo}(I_2, L) = \text{closure}\left(\left\{ \begin{array}{l} P \rightarrow ( L \bullet ), \\ L \rightarrow L \bullet \text{id} \end{array} \right\}\right)$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

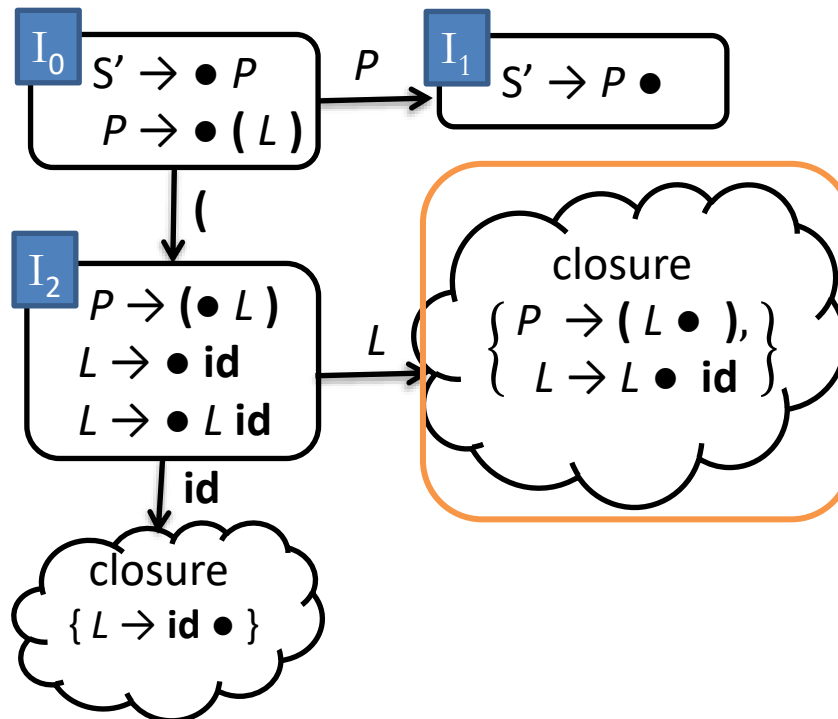
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state

$$\text{closure} \left( \left\{ \begin{array}{l} P \rightarrow ( L \bullet ), \\ L \rightarrow L \bullet \text{id} \end{array} \right\} \right)$$

$$=$$

$$\begin{array}{l} P \rightarrow ( L \bullet ) \\ L \rightarrow L \bullet \text{id} \\ \text{(nothing else)} \end{array}$$



# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

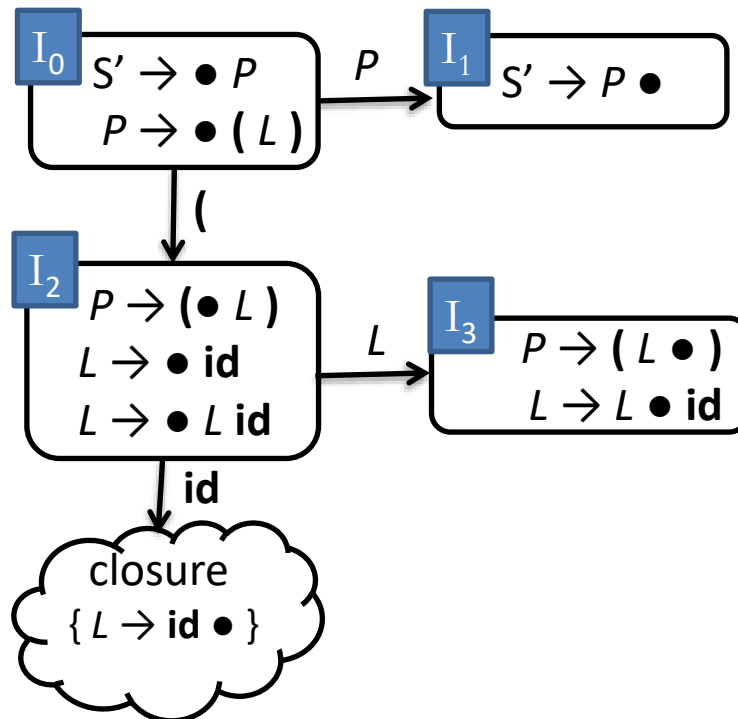
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{closure} \left( \left\{ \begin{array}{l} P \rightarrow ( L \bullet ) \\ L \rightarrow L \bullet \text{id} \end{array} \right\} \right)$   
 =  
 $P \rightarrow ( L \bullet )$   
 $L \rightarrow L \bullet \text{id}$   
 (nothing else)

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

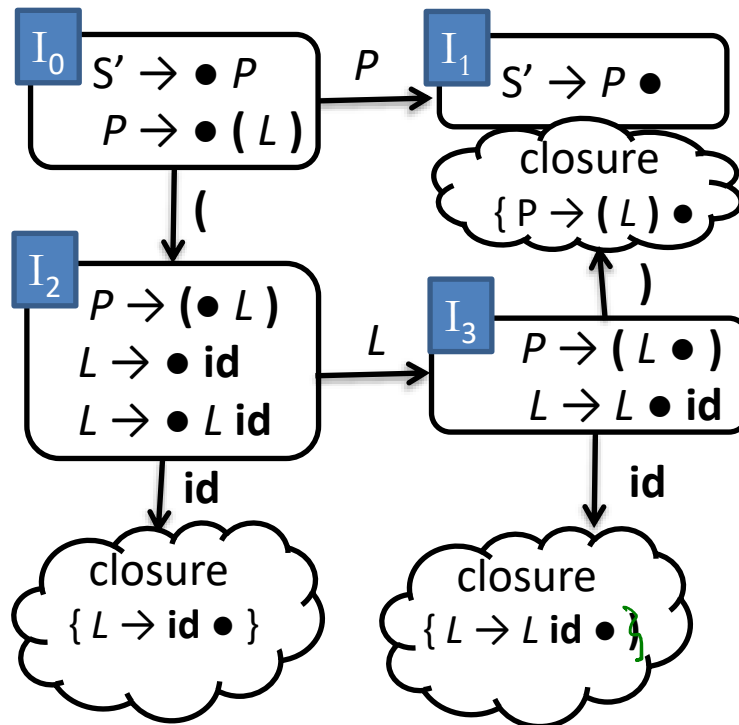
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{closure}\left(\left\{\begin{array}{l} P \rightarrow ( L \bullet ) \\ L \rightarrow L \bullet \text{id} \end{array}\right\}\right)$   
 $=$   
 $P \rightarrow ( L \bullet )$   
 $L \rightarrow L \bullet \text{id}$   
 (nothing else)

Edges out of  $I_3$   
 $\text{GoTo}(I_3, ) = \text{closure}(\{P \rightarrow ( L ) \bullet\})$   
 $\text{GoTo}(I_3, \text{id}) = \text{closure}(\{L \rightarrow L \text{id} \bullet\})$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

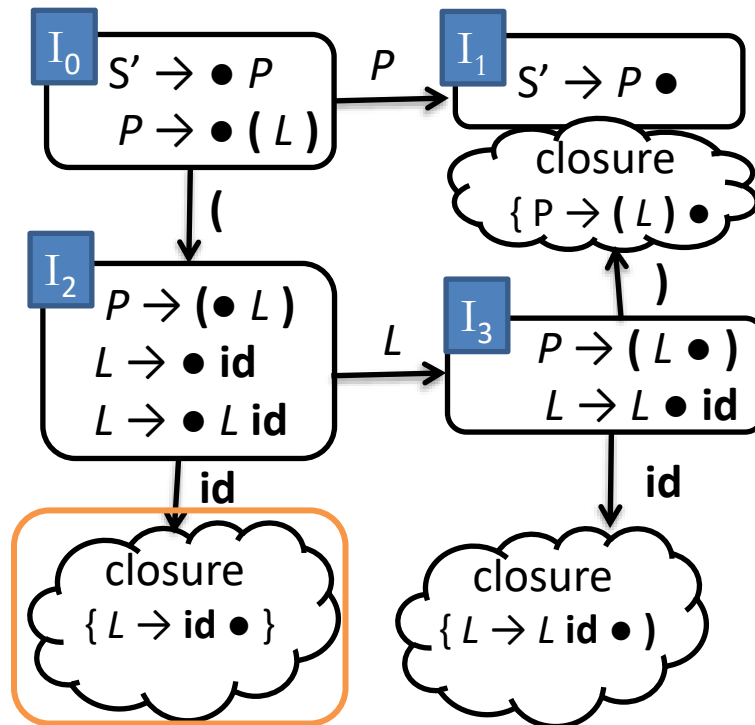
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{L \rightarrow \text{id} \bullet\})$   
 $=$   
 $L \rightarrow \text{id} \bullet$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

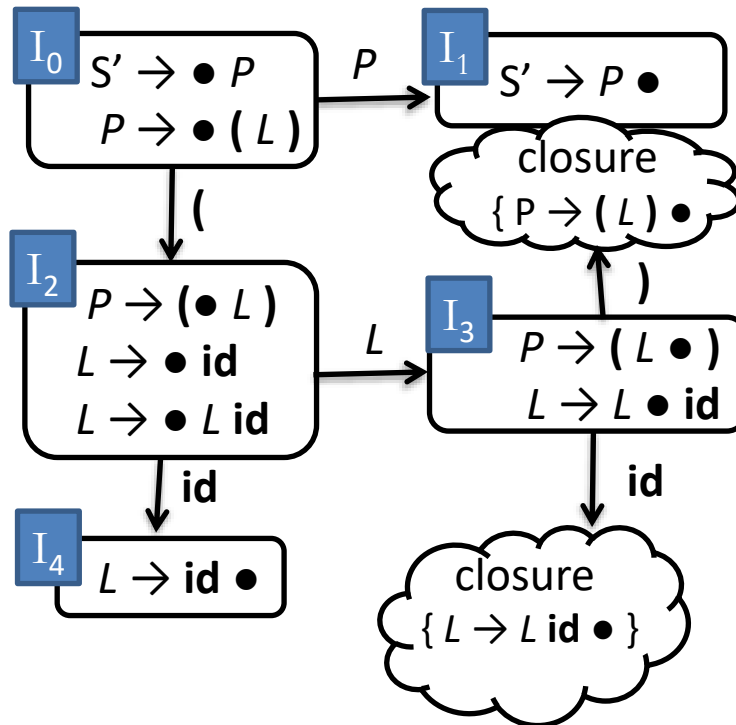
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{L \rightarrow \text{id} \bullet\})$

$$= L \rightarrow \text{id} \bullet$$

No edges out of  $I_4$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

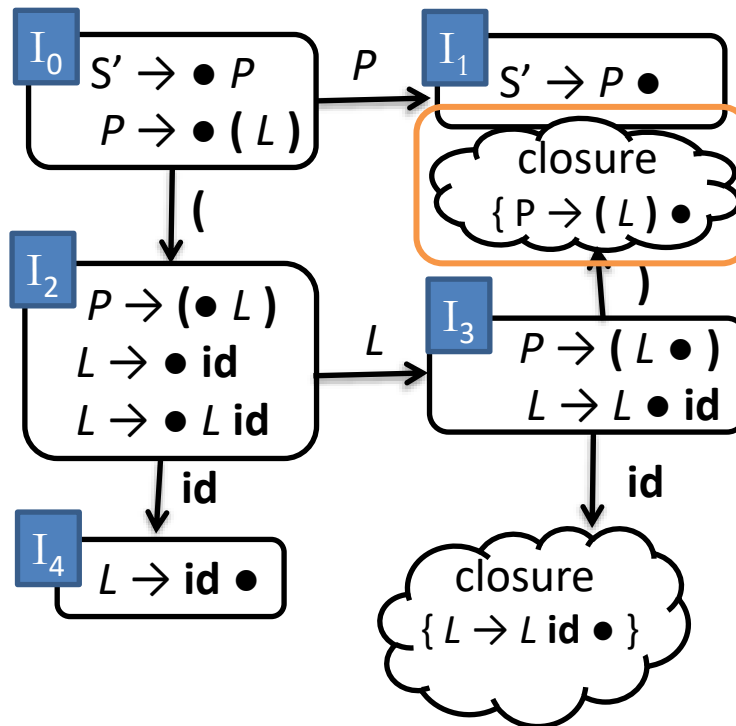
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{P \rightarrow ( L ) \bullet\})$   
 =  
 $P \rightarrow ( L ) \bullet$   
 (nothing else)

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to G
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to GoTo
  - Add nodes according to Closure

### GoTo(I,X) =

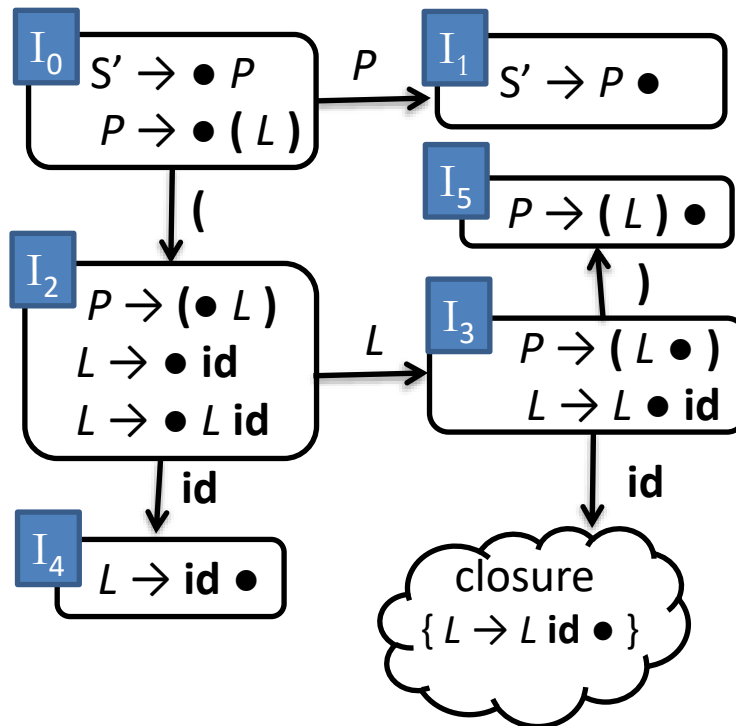
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### Closure(I):

Begin with I  
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{ P \rightarrow ( L ) \bullet \})$   
 =  
 $P \rightarrow ( L ) \bullet$   
 (nothing else)

No edges out of  $I_5$

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

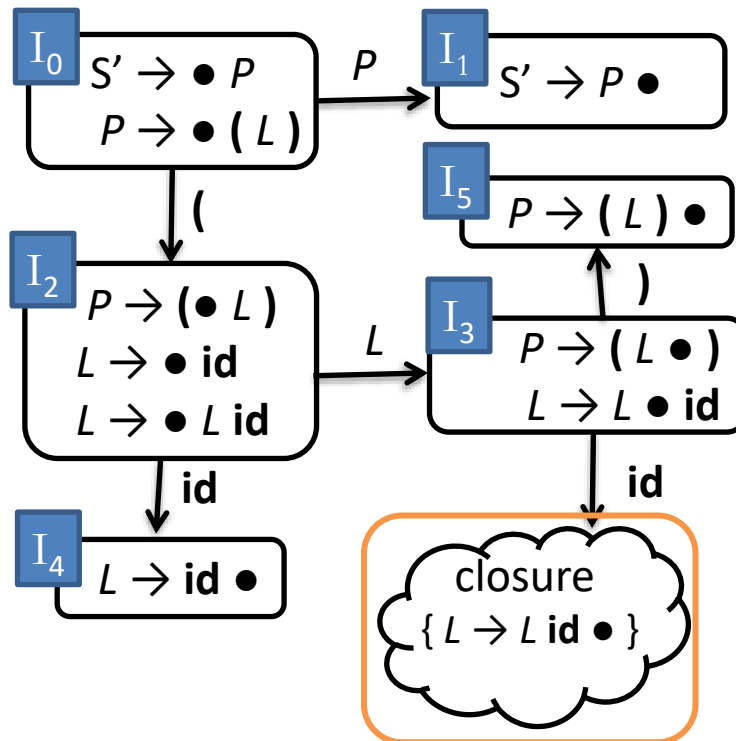
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{L \rightarrow \text{id} \bullet\})$   
 $=$   
 $L \rightarrow \text{id} \bullet$   
 (nothing else)

# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to G
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to GoTo
  - Add nodes according to Closure

### GoTo(I,X) =

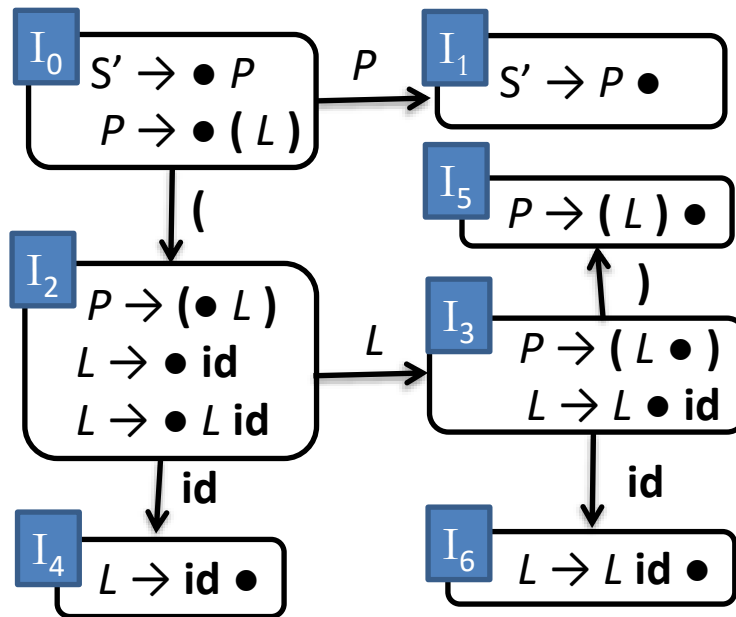
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### Closure(I):

Begin with I  
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I)$ :  
 $\forall Z ::= \gamma$ , add  $Z \rightarrow \bullet \gamma$

### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Identify state  
 $\text{Closure}(\{L \rightarrow \text{id} \bullet\})$   
 =  
 $L \rightarrow \text{id} \bullet$   
 (nothing else)

No edges out of  $I_6$



# Building an LR Parser: Example

## SLR Parsing

### Parse Table Construction

- 1: Add new 1<sup>st</sup> production  $S' ::= S$  to  $G$
- 2: Build State  $I_0$  for  $\text{Closure}(\{S' \rightarrow \bullet S\})$
- 3: Saturate FSM:
  - Add edges according to  $\text{GoTo}$
  - Add nodes according to  $\text{Closure}$

### $\text{GoTo}(I, X) =$

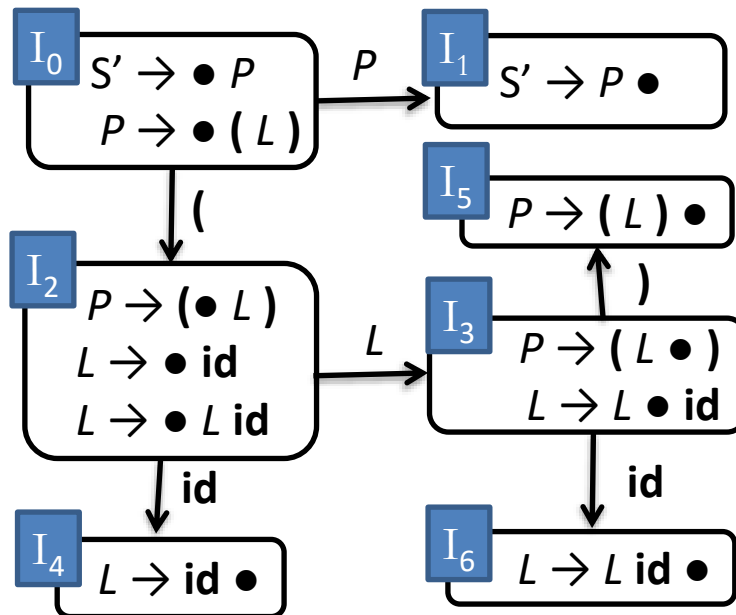
State that represents  
 Closure of all items  
 $A \rightarrow \alpha \hat{\pi} \bullet \beta$  where  
 $A \rightarrow \alpha \bullet \hat{\pi} \beta \in I$

### $\text{Closure}(I):$

Begin with  $I$   
 Repeat until saturation:  
 if  $X \rightarrow \alpha \bullet Z \beta \in \text{Closure}(I):$   
 $\forall Z ::= \gamma, \text{ add } Z \rightarrow \bullet \gamma$

### Grammar G

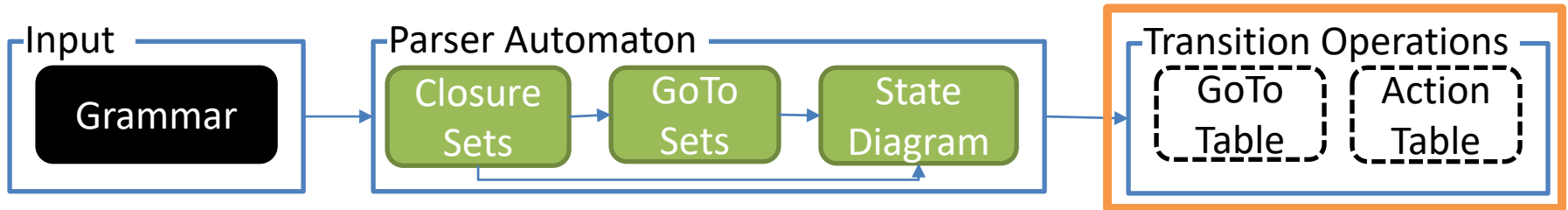
- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Automaton Complete!

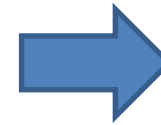
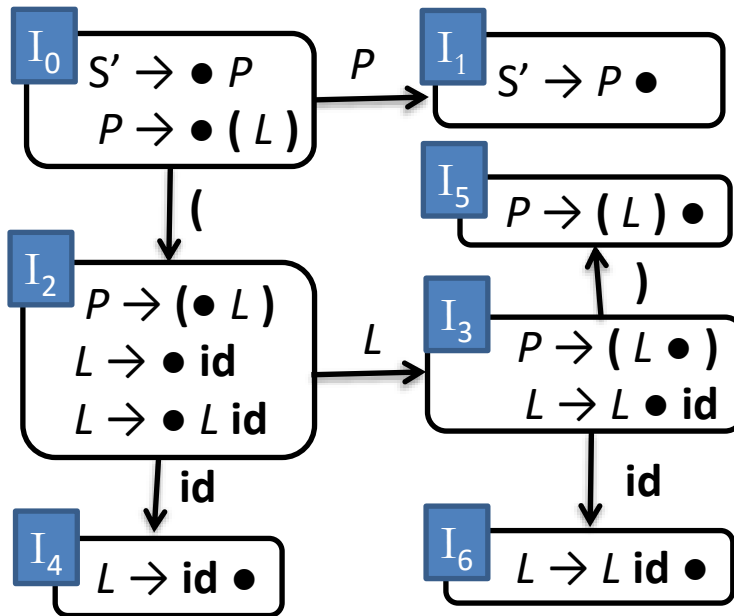
# Time to Convert FSM to a Table

## LR Parser Construction



### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$

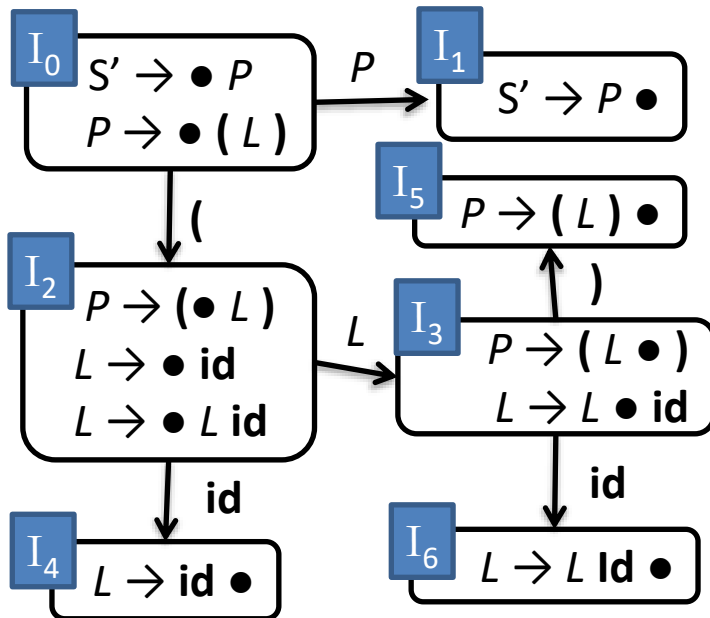


# Basic Table Structure

## LR Parser Construction

**Row:** Item

**Column:** Symbol



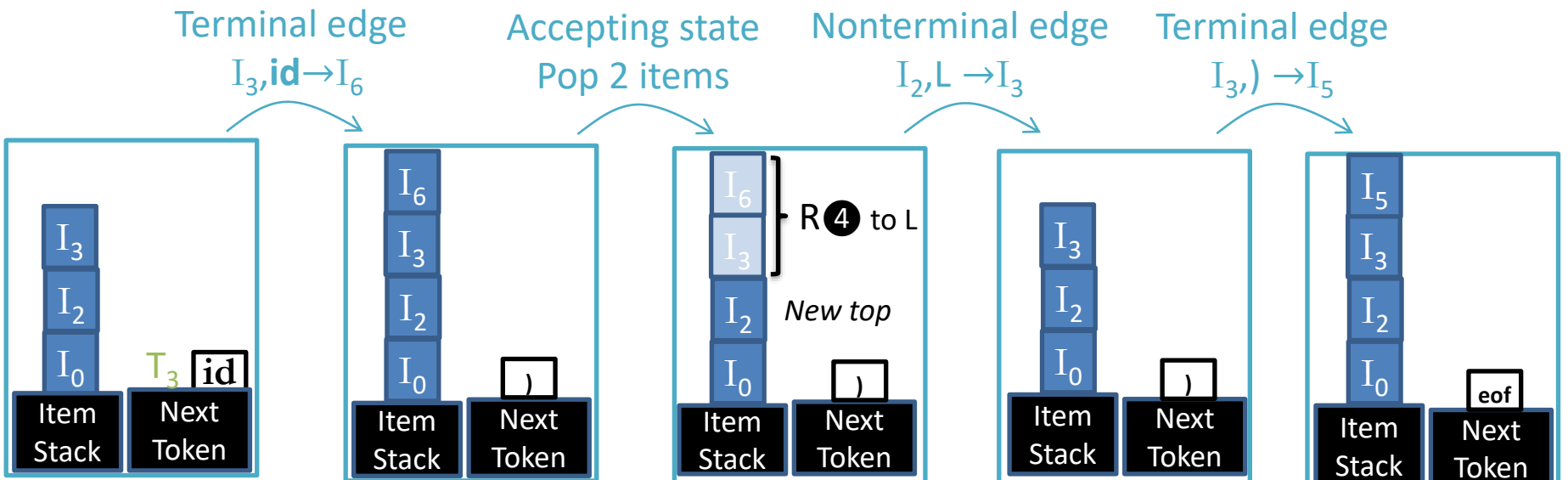
	Action Table				GoTo Table	
	(	)	id	eof	$P$	$L$
$I_0$						
$I_1$						
$I_2$						
$I_3$						
$I_4$						
$I_5$						
$I_6$						

# What Should the Table Look Like?

## LR Parser Construction

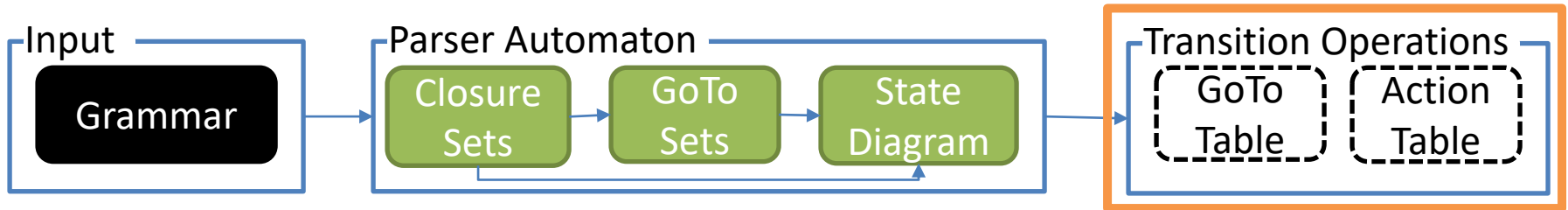
### Types of Actions (hence, table cell entries):

- Shift (terminal edge), push 1 state, advance lookahead
- Reduce (reduce state), pop RHS stack items
- GoTo (nonterminal edge), push LHS item following reduce
- (Special actions: accept / reject)



# Time to Convert FSM to a Table

## LR Parser Construction



### SLR Building algorithm

For each edge  $I_j, \tau = I_k$  in the FSM:

shift

if  $\tau$  is a terminal: set  $\text{Action}[I_j, \tau] = \text{shift } I_k$

go to

if  $\tau$  is a nonterminal: set  $\text{GoTo}[I_j, \tau] = I_k$

accept

If state  $I_j$  includes item  $S' \rightarrow S \bullet$

set  $\text{Action}[I_j, \text{eof}] = \text{accept}$

reduce

If state  $I_j$  includes item  $A \rightarrow \alpha \bullet$  where  $A$  is not  $S'$

for each  $t$  in  $\text{FOLLOW}(A)$ :

set  $\text{Action}[I_j, t] = \text{reduce by } A \rightarrow \alpha$

All other entries are error actions

# Time to Convert FSM to a Table

## LR Parser Construction

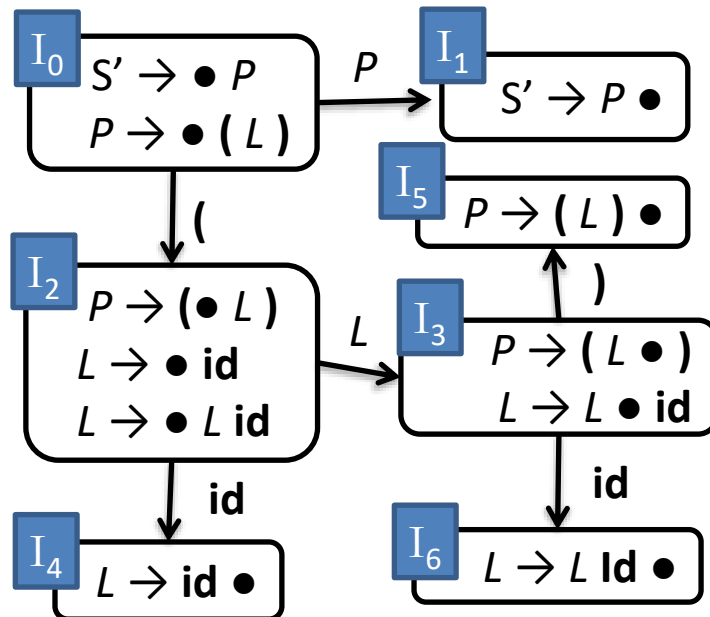
### 4 Types of Table entries

Shift, GoTo, Accept, Reduce

*Standard DFA  
Translation*

#### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$



Action Table

GoTo Table

	(	)	id	eof	P	L
I <sub>0</sub>	I <sub>2</sub>				I <sub>1</sub>	
I <sub>1</sub>						
I <sub>2</sub>			I <sub>4</sub>			I <sub>3</sub>
I <sub>3</sub>		I <sub>5</sub>	I <sub>6</sub>			
I <sub>4</sub>						
I <sub>5</sub>						
I <sub>6</sub>						

# Time to Convert FSM to a Table

## LR Parser Construction

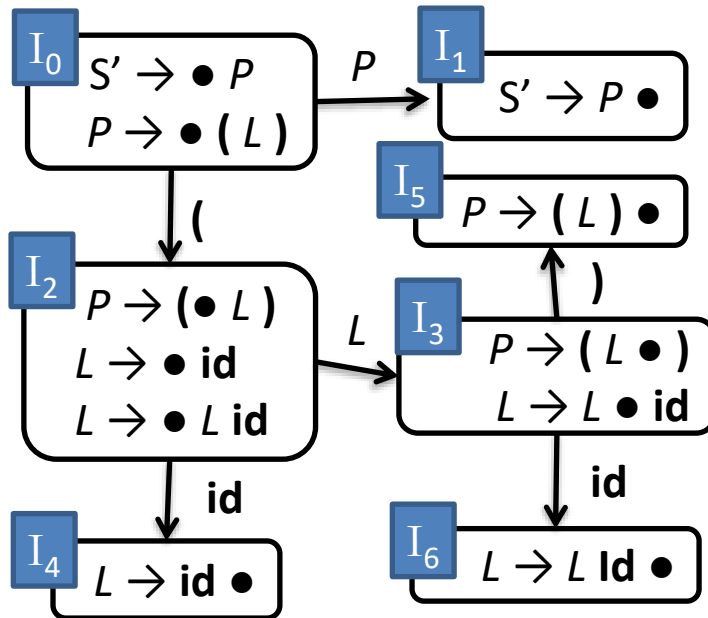
### 4 Types of Table entries

Shift, GoTo, **Accept**, Reduce

*Mark state(s)  
with  $S' \rightarrow S \bullet$   
as accept*

#### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$



Action Table

GoTo Table

	(	)	id	eof	P	L
$I_0$	$I_2$				$I_1$	
$I_1$				☺		
$I_2$			$I_4$			$I_3$
$I_3$		$I_5$	$I_6$			
$I_4$						
$I_5$						
$I_6$						

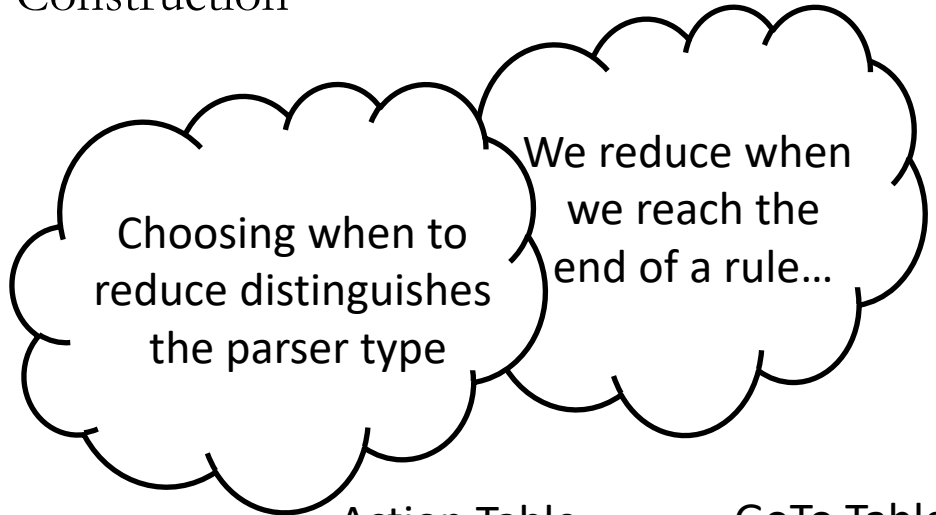
# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

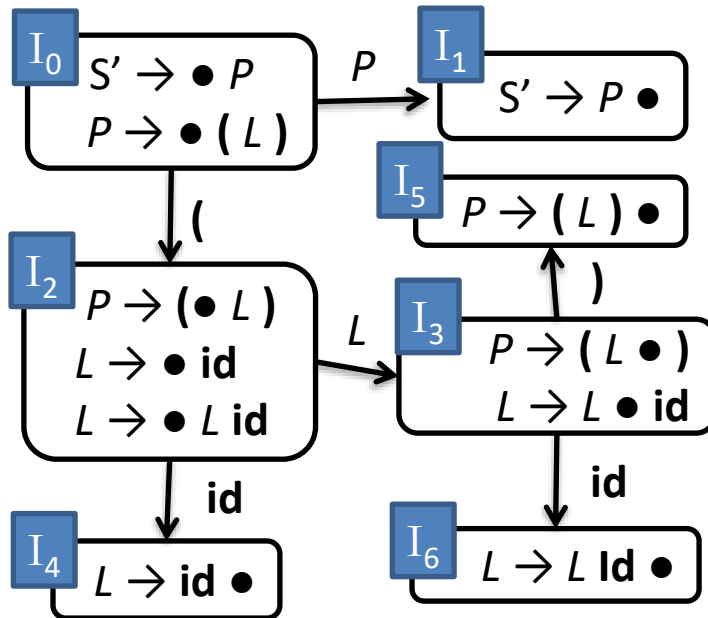
Shift, GoTo, Accept, **Reduce**

Simplest Version – LR(0): reduce whenever the state is reached



#### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$



Action Table

GoTo Table

	(	)	id	eof	P	L
I <sub>0</sub>	I <sub>2</sub>				I <sub>1</sub>	
I <sub>1</sub>				☺		
I <sub>2</sub>			I <sub>4</sub>			I <sub>3</sub>
I <sub>3</sub>		I <sub>5</sub>	I <sub>6</sub>			
I <sub>4</sub>	R 3	R 3	R 3	R 3	R 3	R 3
I <sub>5</sub>	R 2	R 2	R 2	R 2	R 2	R 2
I <sub>6</sub>	R 4	R 4	R 4	R 4	R 4	R 4



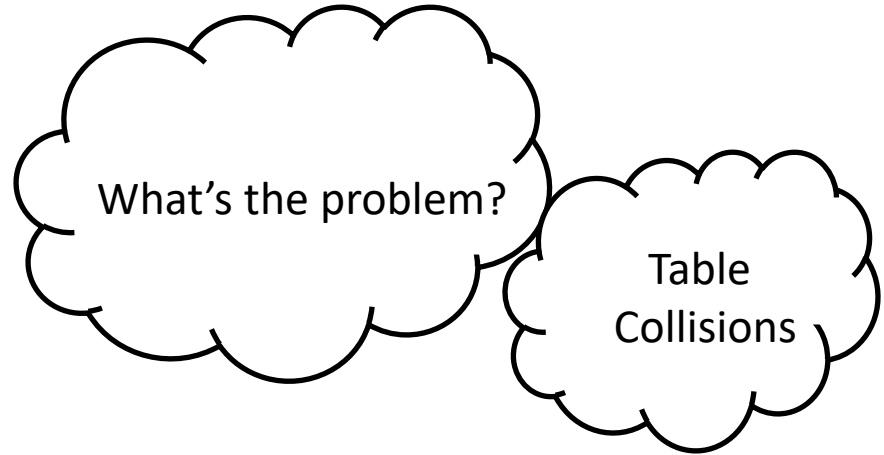
# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

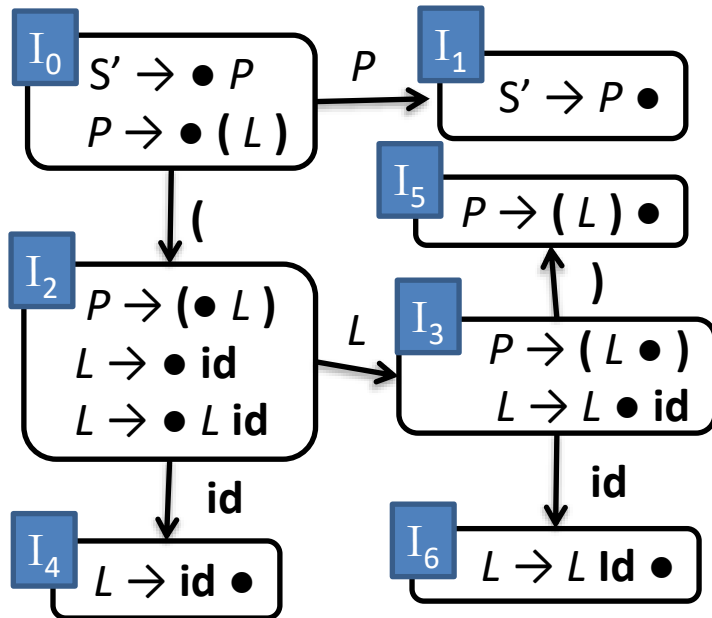
Shift, GoTo, Accept, **Reduce**

Simplest Version – LR(0): reduce whenever the state is reached



#### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= id$
- 4  $L ::= L id$



Action Table

GoTo Table

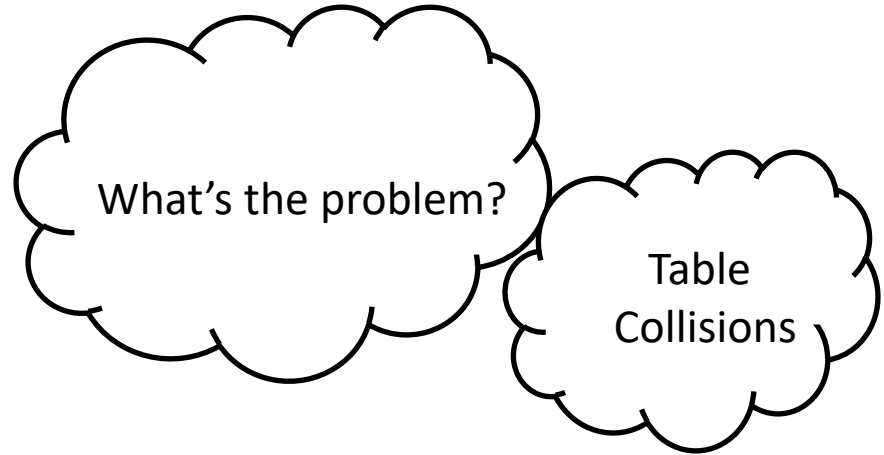
	(	)	id	eof	P	L
I <sub>0</sub>	I <sub>2</sub>				I <sub>1</sub>	
I <sub>1</sub>				☺		
I <sub>2</sub>			I <sub>4</sub>			I <sub>3</sub>
I <sub>3</sub>		I <sub>5</sub>	I <sub>6</sub>			
I <sub>4</sub>	R 3	R 3	R 3	R 3	R 3	R 3
I <sub>5</sub>	R 2	R 2	R 2	R 2	R 2	R 2
I <sub>6</sub>	R 4	R 4	R 4	R 4	R 4	R 4

# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

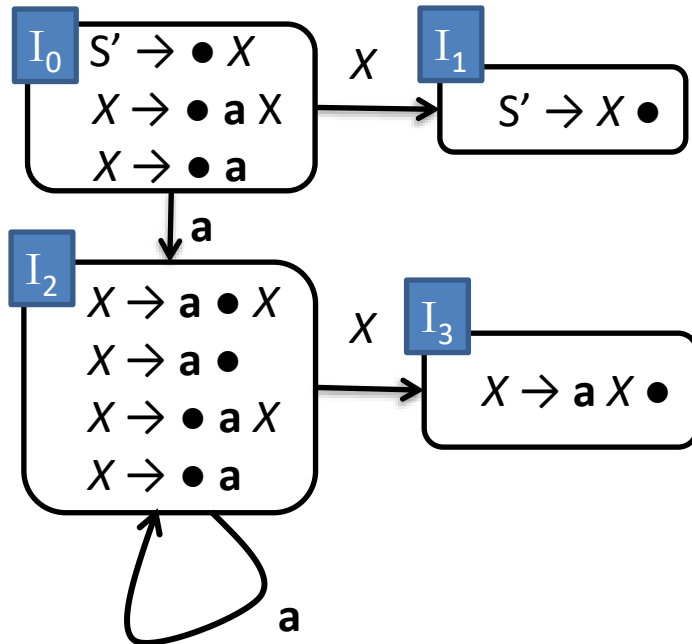
Shift, GoTo, Accept, **Reduce**



Simplest Version – LR(0): reduce whenever the state is reached

#### Grammar G

- 1  $S' ::= X$
- 2  $X ::= aX$
- 3  $X ::= a$



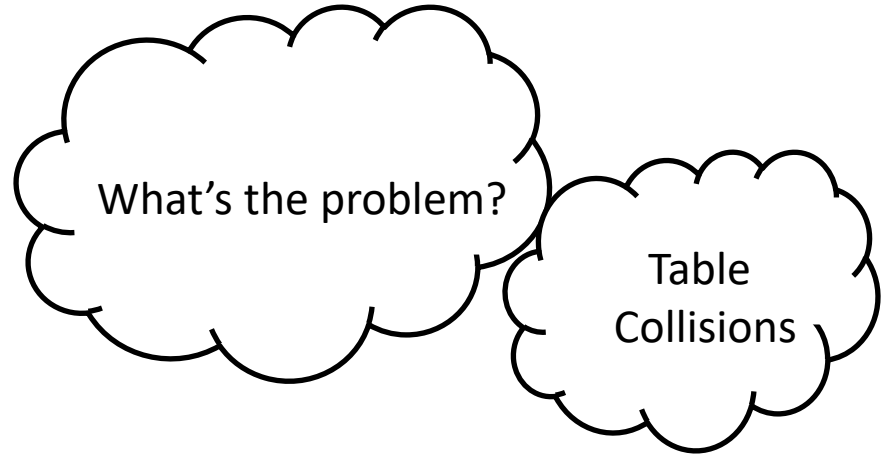
	Action		GoTo
	a	eof	X
I <sub>0</sub>	I <sub>2</sub>		I <sub>1</sub>
I <sub>1</sub>		☺	
I <sub>2</sub>	I <sub>2</sub> R 3	R 3	I <sub>3</sub> R 3
I <sub>3</sub>	R 2	R 2	R 2

# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

Shift, GoTo, Accept, **Reduce**



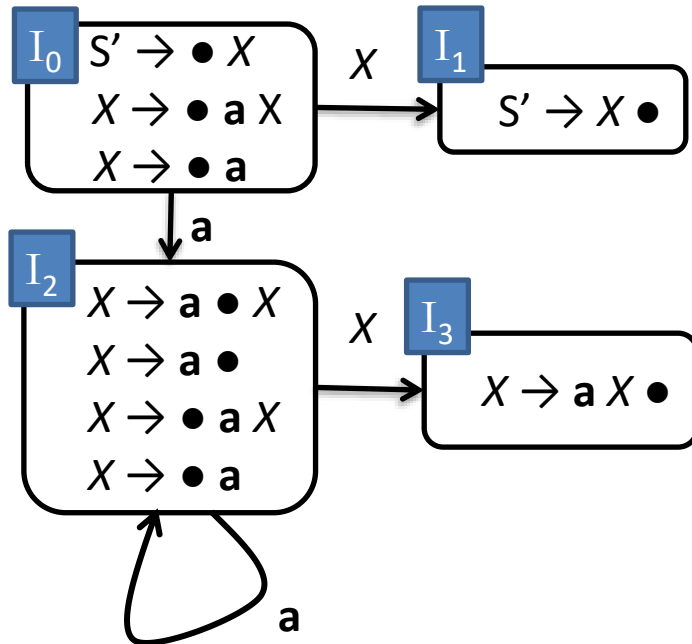
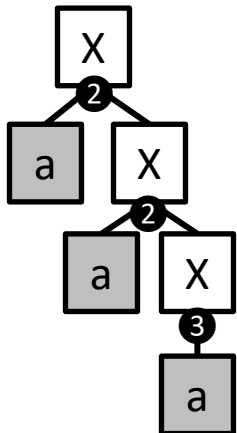
~~Better~~ ~~Simplest Version~~ – ~~LR(0)~~: reduce whenever the state is reached

**FOLLOW set allows it**

#### Grammar G

- 1  $S' ::= X$
- 2  $X ::= a X$
- 3  $X ::= a$

#### Parse Trees



	Action		GoTo
	a	eof	X
I <sub>0</sub>	I <sub>2</sub>		I <sub>1</sub>
I <sub>1</sub>		☺	
I <sub>2</sub>	I <sub>2</sub> R 3	R 3	I <sub>3</sub> R 3
I <sub>3</sub>	R 2	R 2	R 2

# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

Shift, GoTo, Accept, **Reduce**

if state  $I_j$  has  $A \rightarrow \alpha \bullet$  where  $A \neq S'$   
 for each  $t$  in FOLLOW(A):  
 $Action[I_j, t] = \text{reduce by } A ::= \alpha$

~~Simplest Version – LR(0):~~ <sup>Better</sup> ~~reduce whenever the state is reached~~ <sup>SLR</sup>

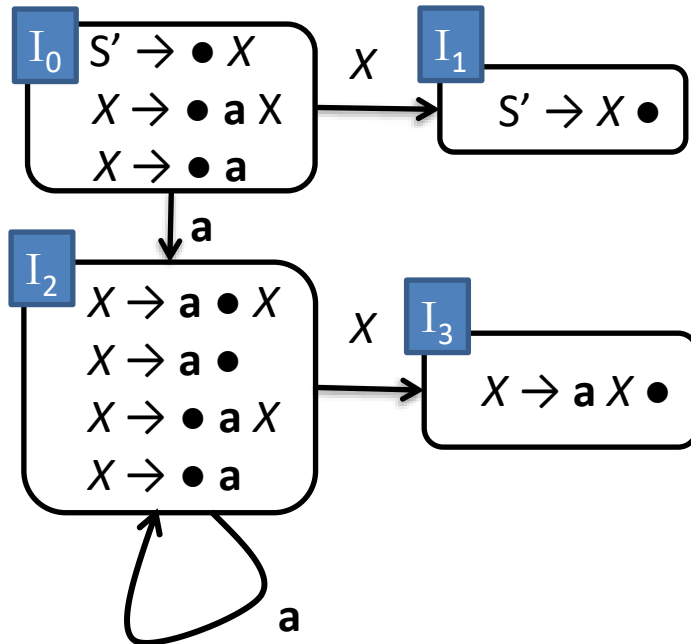
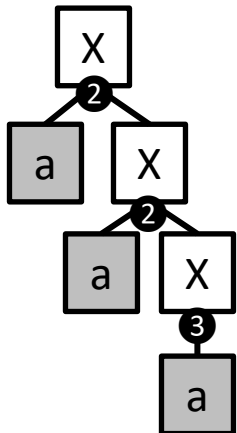
FOLLOW(X) = { eof }

#### Grammar G

- 1  $S' ::= X$
- 2  $X ::= a X$
- 3  $X ::= a$

FOLLOW set allows it

#### Parse Trees



	Action		GoTo
	a	eof	X
$I_0$	$I_2$		$I_1$
$I_1$		☺	
$I_2$	$I_2$ R 3	R 3	$I_3$ R 3
$I_3$	R 2	R 2	R 2

# Time to Convert FSM to a Table

## LR Parser Construction

### 4 Types of Table entries

Shift, GoTo, Accept, **Reduce**

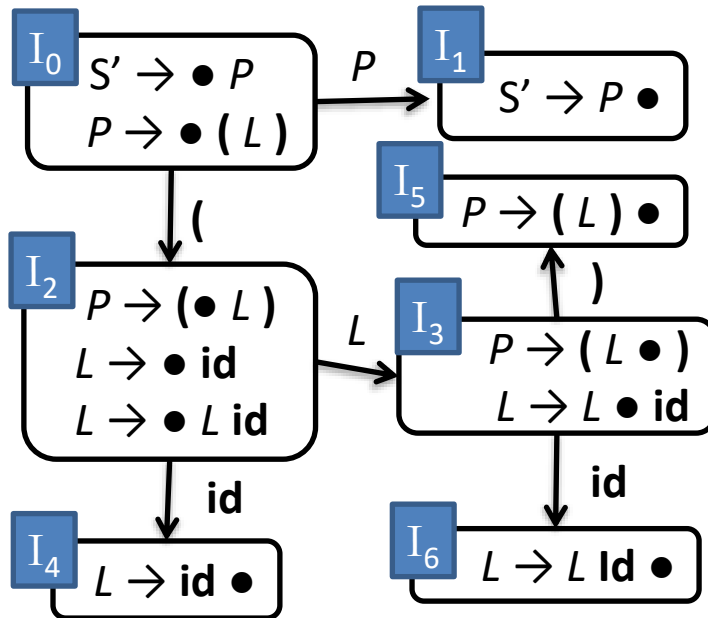
if state  $I_j$  has  $A \rightarrow \alpha \bullet$  where  $A \neq S'$   
 for each  $t$  in  $FOLLOW(A)$ :  
 $Action[I_j, t] = \text{reduce by } A ::= \alpha$

Better Version – SLR: reduce whenever  
 FOLLOW set allows it

$FOLLOW(L) = \{ \text{), id} \}$   
 $FOLLOW(P) = \{ \text{eof} \}$

#### Grammar G

- 1  $S' ::= P$
- 2  $P ::= ( L )$
- 3  $L ::= \text{id}$
- 4  $L ::= L \text{id}$



Action Table

GoTo Table

	(	)	id	eof	P	L
$I_0$	S $I_2$				$I_1$	
$I_1$				☺		
$I_2$			S $I_4$			$I_3$
$I_3$		S $I_5$	S $I_6$			
$I_4$		R 3	R 3			
$I_5$				R 2		
$I_6$		R 4	R 4			

# Summary

## LR Parser Construction

### **Constructing a new type of parser, the LR**

- For all LR parsers:
  - build closure and goto sets
  - Build the parser automaton
  - Tableize it
- For some LR parsers:
  - What “closure set” and “goto set” mean is different
  - Tabelizing changes (SLR uses FOLLOW sets to Reduce)
  - There are more advanced LR parsers with more elaborate closure and goto sets (we won't cover them)

# Next Time

Preview - Scope

## **Finally done with parsing!**

- Done with the frontend of the compiler!

# Next Time

Preview - Scope

## **Finishing up some final details of the LR Parser**

- Running the table-based parser
- Bottom-up SDT

## **After that, all done with parsing!**

- Done with the frontend of the compiler