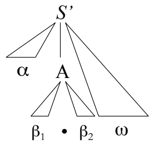
Syntactic Analysis

Building a LR(0) Parser

Key Insights

- Need to Capture State
  - Which portion of a given Production we have seen so far
  - What are the Non-terminal on the Stack



  - We have already seen $\alpha\ \beta_1$
  - Need to Encode that Knowledge on a Stack for later

Valid Items

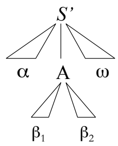
Definition: Item $A \rightarrow \beta_1 \bullet \beta_2$ is valid for a *viable prefix* $\alpha\ \beta_1$ if there is a derivation

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha \beta_1 \beta_2 \omega$$

1. If $\beta_2 \neq \varepsilon$ then the valid item $A \rightarrow \beta_1 \bullet \beta_2$ suggest that the action should be a *shift*
2. If $\beta_2 = \varepsilon$ then the valid item $A \rightarrow \beta_1 \bullet$ suggest the action should be a *reduce*

Item captures how much of a given production we have scanned so far

Valid Items - in Pictures

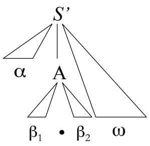


$$S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha \beta_1 \beta_2 \omega$$

Item $A \rightarrow \beta_1 \bullet \beta_2$ is valid for a viable prefix $\alpha\ \beta_1$ if there is a derivation

$$S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha \beta_1 \beta_2 \omega$$

Valid Items - in Pictures

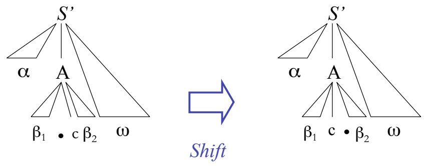


$$S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha \beta_1 \beta_2 \omega$$

Item $A \rightarrow \beta_1 \bullet \beta_2$ is valid for a viable prefix $\alpha\ \beta_1$ if there is a derivation

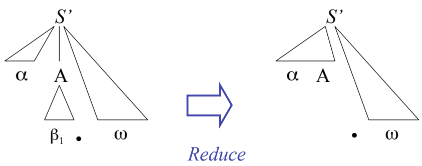
$$S' \overset{*}{\underset{rm}{\Rightarrow}} \alpha A \omega \underset{rm}{\Rightarrow} \alpha \beta_1 \beta_2 \omega$$

Valid Items - in Pictures



$A \rightarrow \beta_1 \bullet c\ \beta_2$

Valid Items - in Pictures



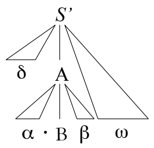
$A \rightarrow \beta_1 \bullet$

Sets of Items

- States of a DFA that can recognize handles
  - Items in each State guide parser to either Shift or Reduce
  - On a Reduction, change State
  - Find Item in each State using *closure* operation
- Algorithm for **closure(I)**
  - Every item in **I** is also an item in **closure(I)**
  - If $A \rightarrow \alpha \bullet B\ \beta$ is in **closure(I)** and $B \rightarrow \bullet\ \gamma$ is an item, then add $B \rightarrow \bullet\ \gamma$ to **closure(I)**
  - Repeat until no more new items can be added to **closure(I)**

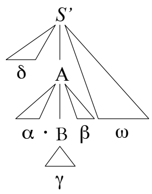
Sets of Items: Closure

- Algorithm for **closure(I)**
  - Every item in **I** is also an item in **closure(I)**
  - If A→ α • B β is in **closure(I)** and B→ • γ is an item, then add B→ • γ to **closure(I)**
  - Repeat until no more new items can be added to **closure(I)**



Sets of Items: Closure

- Algorithm for **closure(I)**
  - Every item in **I** is also an item in **closure(I)**
  - If A→ α • B β is in **closure(I)** and B→ • γ is an item, then add B→ • γ to **closure(I)**
  - Repeat until no more new items can be added to **closure(I)**



The Goto Operation

- On a *Reduction* which *state* should the parser go to ?
- The new state after consuming a grammar symbol while at the current state
- Algorithm for **goto(I, X)** where **I** is a set of items and **X** is a grammar symbol

goto(I, X) = closure({ A→ α X • β | A→ α • X β in I })

- *goto is the new set obtained by "moving the dot" over X*

LR(0) Items

- Recall: An Item captures how much of a given production we have scanned so far

<X> → (<X>)

- Represented by 4 items
  - <X> → • (<X>)
  - <X> → (• <X>)
  - <X> → (<X> •)
  - <X> → (<X>) •

Example of Items

The grammar

<S> → <X> $
<X> → (<X>)
<X> → ( )

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Example of Closure

- Find **closure(**<X> → (• <X>) **)**

{
<X> → (• <X>)
<X> → • (<X>)
<X> → • ( )
}

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Question: Find the Closure

- Find **closure(**<S> → • <X> $ **)**

{ ???? }

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Question: Find the Closure

- Find **closure(**<S> → • <X> $ **)**

{
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )
}

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Example of Goto

- Find **goto(<X> → (• <X>), <X>)**

{ <X> → (<X> •) }

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Question: Find the goto

- Find **goto(<X> → • (<X>), ( )**

{ ???? }

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Question: Find the goto

- Find **goto(<X> → • (<X>), ( )**

{
<X> → (• <X>)
<X> → • (<X>)
<X> → • ( )
}

Items

<S> → • <X> $
<S> → <X> • $
<X> → • (<X>)
<X> → (• <X>)
<X> → (<X> •)
<X> → (<X>) •
<X> → • ( )
<X> → (•)
<X> → ( ) •

Building the DFA states Example

- Start with the production <**S'**> **→ • <S>** $
- Create the first state to be **closure(<S'> → • <S> $)**
- Pick a state **I**
  - for each A→ **α** • X β in **I**
    - find **goto(I, X)**
    - if **goto(I, X)** is not already a state, make one
    - Add an edge X from state **I** to **goto(I, X)** state
- Repeat until no more additions possible

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

<S> → • <X> $

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

<S> → • <X> $

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

goto(, <X>)

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

<S> → <X> • $

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ( )

goto(, <X>)

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

<S> → <X> • $

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ()

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s1
<S> → <X> • $

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ()

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

s1
<S> → <X> • $

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ()

Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible

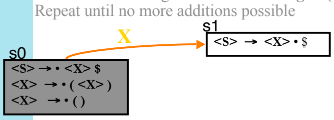
s0 $\xrightarrow{X}$ s1

s1
<S> → <X> • $

s0
<S> → • <X> $
<X> → • (<X>)
<X> → • ()

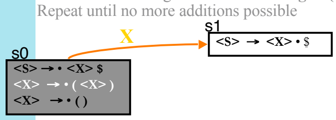
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



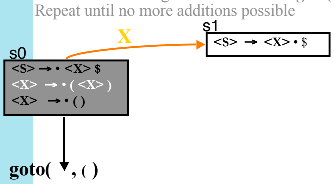
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



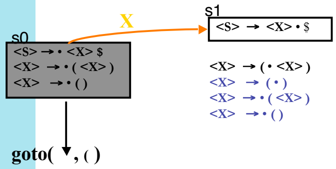
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



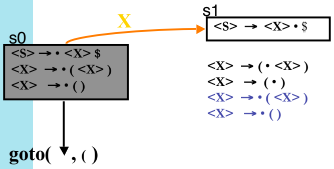
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



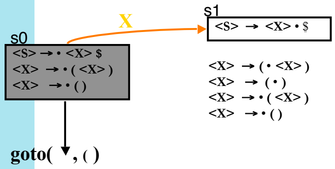
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



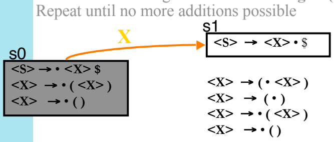
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



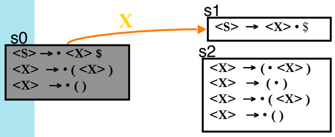
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



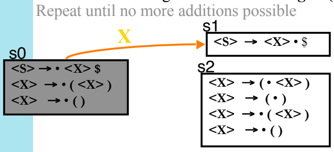
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
for each A→ α • X β in **I**
find **goto(I, X)**
if **goto(I, X)** is not already a state, make one
Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



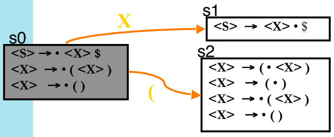
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



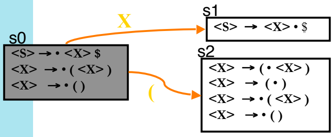
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



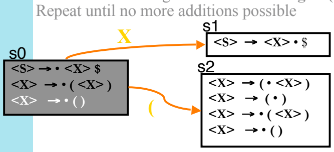
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



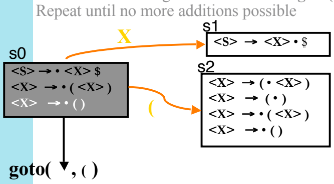
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



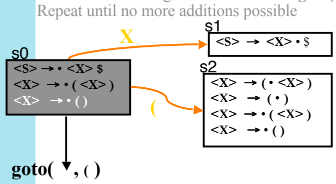
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
 for each A→ α • X β in **I**
  find **goto(I, X)**
  if **goto(I, X)** is not already a state, make one
  Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



Building the DFA states Example

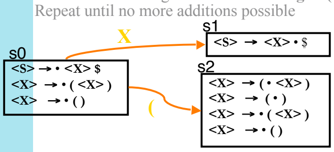
Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
 for each A→ α • X β in **I**
  find **goto(I, X)**
  goto(I, X) is not already a state, make one
  Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



Same as the last time

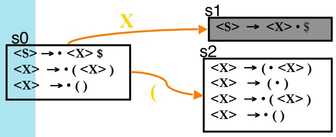
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
 for each A→ α • X β in **I**
  find **goto(I, X)**
  if **goto(I, X)** is not already a state, make one
  Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



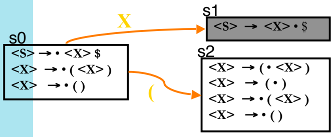
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
 for each A→ α • X β in **I**
  find **goto(I, X)**
  if **goto(I, X)** is not already a state, make one
  Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



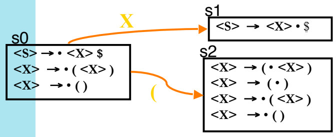
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



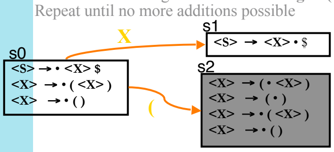
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



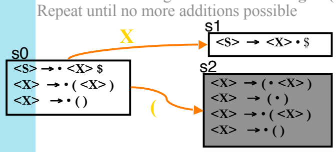
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



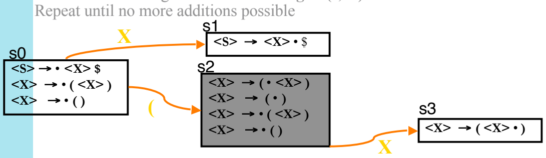
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
  for each A→ α • X β in **I**
    find **goto(I, X)**
    if **goto(I, X)** is not already a state, make one
    Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



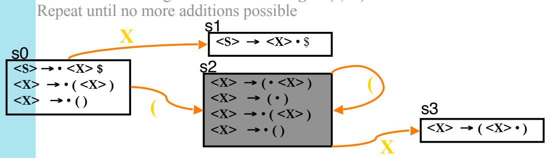
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



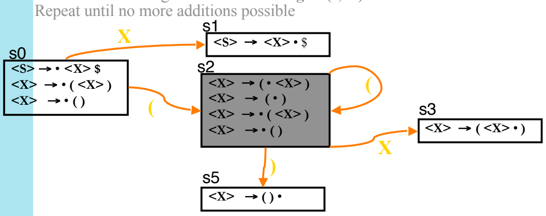
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



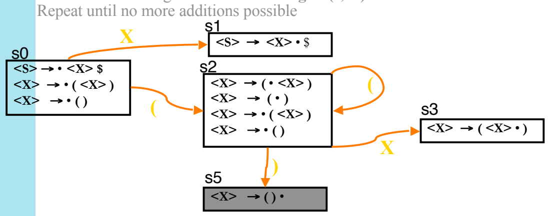
Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



Building the DFA states Example

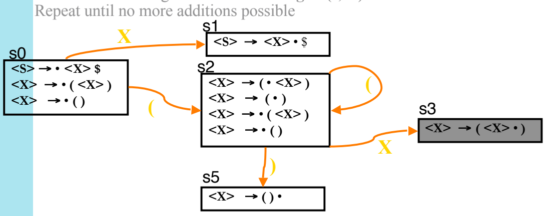
Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
    for each A→ α • X β in **I**
        find **goto(I, X)**
        if **goto(I, X)** is not already a state, make one
        Add an edge X from state **I** to **goto(I, X)** state
Repeat until no more additions possible



Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
- for each A→ α • X β in **I**
  - find **goto(I, X)**
  - if **goto(I, X)** is not already a state, make one
  - Add an edge X from state **I** to **goto(I, X)** state

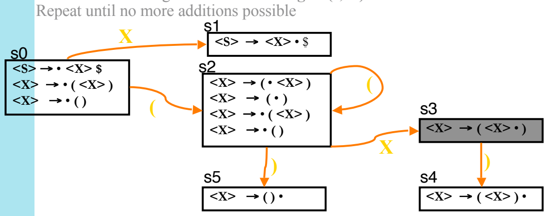
Repeat until no more additions possible



Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
- for each A→ α • X β in **I**
  - find **goto(I, X)**
  - if **goto(I, X)** is not already a state, make one
  - Add an edge X from state **I** to **goto(I, X)** state

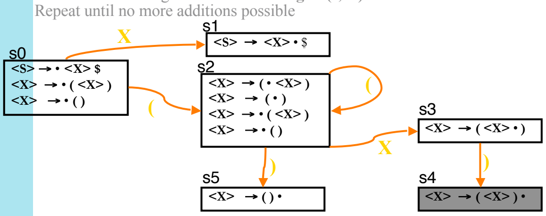
Repeat until no more additions possible



Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
- for each A→ α • X β in **I**
  - find **goto(I, X)**
  - if **goto(I, X)** is not already a state, make one
  - Add an edge X from state **I** to **goto(I, X)** state

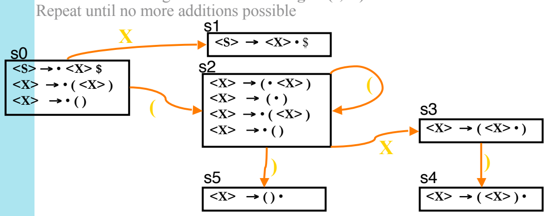
Repeat until no more additions possible



Building the DFA states Example

Start with the production <S'> → • <S> $
Create the first state to be **closure(<S'> → • <S> $)**
Pick a state **I**
- for each A→ α • X β in **I**
  - find **goto(I, X)**
  - if **goto(I, X)** is not already a state, make one
  - Add an edge X from state **I** to **goto(I, X)** state

Repeat until no more additions possible



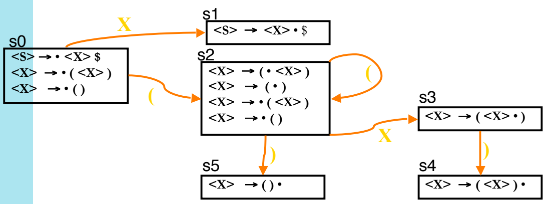
Constructing a LR(0) Parse Engine

- Build a DFA
  - DONE
- Construct a parse table using the DFA

Creating the Parse Tables

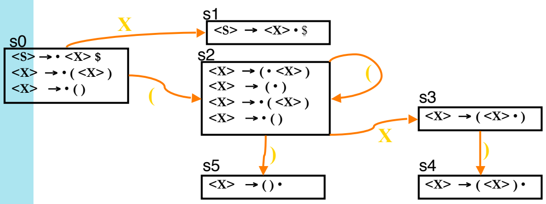
- For each State
  - Transition to another State using a Terminal Symbol is a *shift* to that State (*shift to sn*)
  - Transition to another State using a Non-Terminal is a *goto* to that State (*goto sn*)
  - If there is an Item **A** → α • in the State do a Reduction with that Production for all Terminals (*reduce k*)

Building Parse Table Example



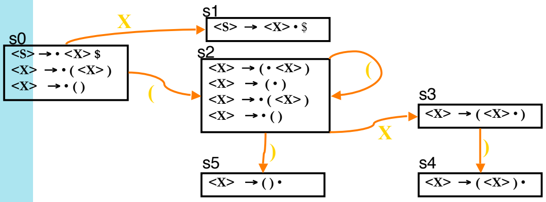
Building Parse Table Example

State	Action			Goto
	(	)	$	X



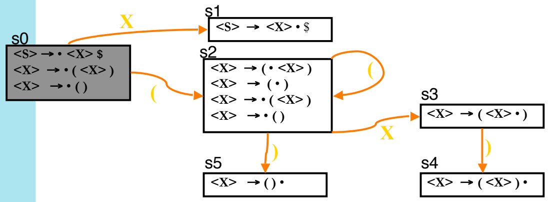
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0				
s1				
s2				
s3				
s4				
s5				



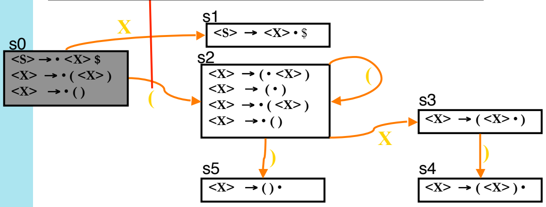
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0				
s1				
s2				
s3				
s4				
s5				



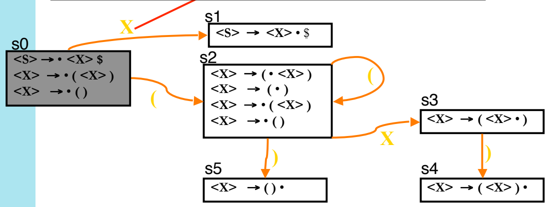
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2			
s1				
s2				
s3				
s4				
s5				



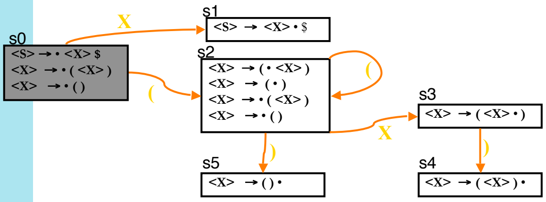
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2			goto s1
s1				
s2				
s3				
s4				
s5				



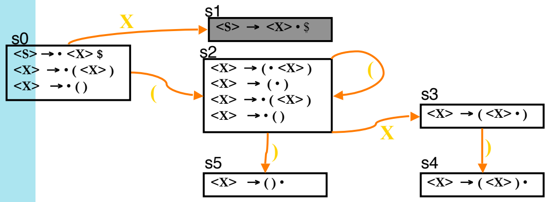
Building Parse Table Example

State	Action (	)	$	Goto X
s0	shift to s2	error	error	goto s1
s1				
s2				
s3				
s4				
s5				



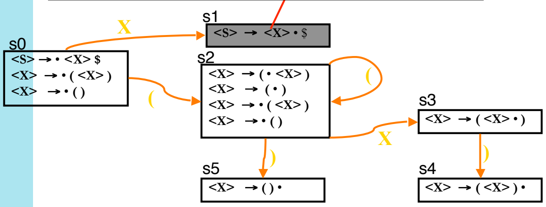
Building Parse Table Example

State	Action (	)	$	Goto X
s0	shift to s2	error	error	goto s1
s1				
s2				
s3				
s4				
s5				



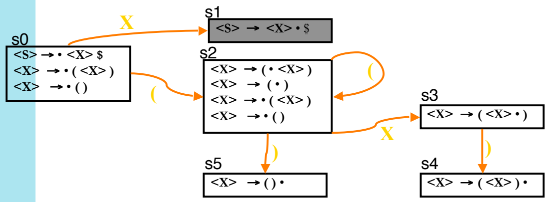
Building Parse Table Example

State	Action (	)	$	Goto X
s0	shift to s2	error	error	goto s1
s1			accept	
s2				
s3				
s4				
s5				



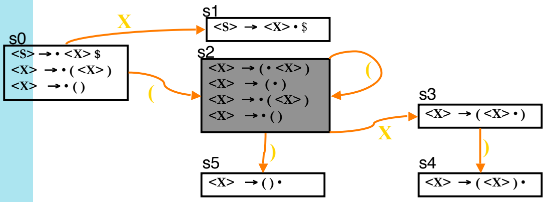
Building Parse Table Example

State	Action (	)	$	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2				
s3				
s4				
s5				



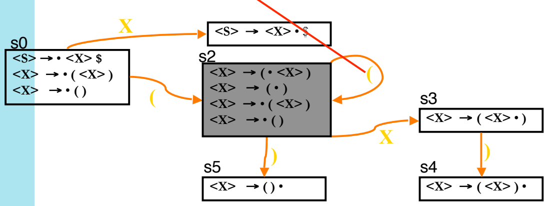
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2				
s3				
s4				
s5				



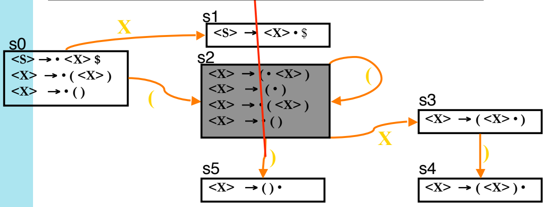
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2			
s3				
s4				
s5				



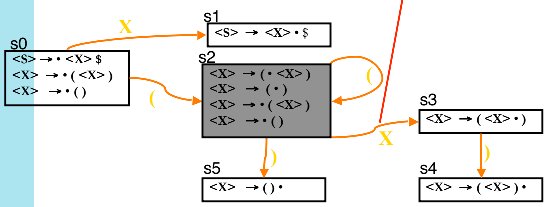
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5		
s3				
s4				
s5				



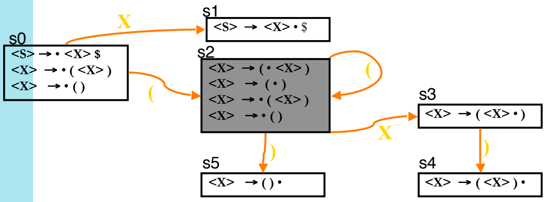
Building Parse Table Example

	Action			Goto
State	(	)	$	X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5		goto s3
s3				
s4				
s5				



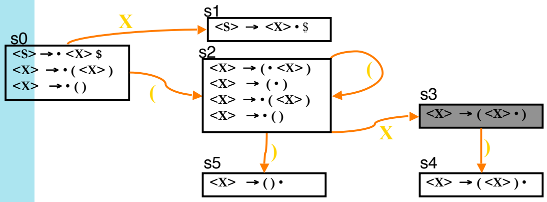
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3				
s4				
s5				



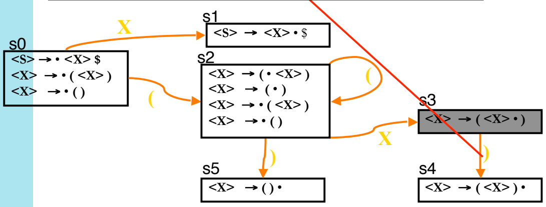
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3				
s4				
s5				



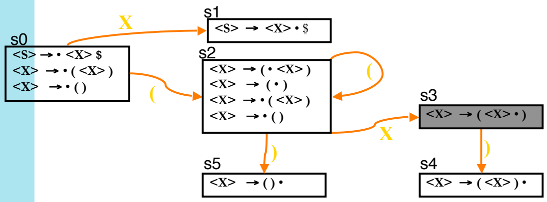
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3		shift to s4		
s4				
s5				



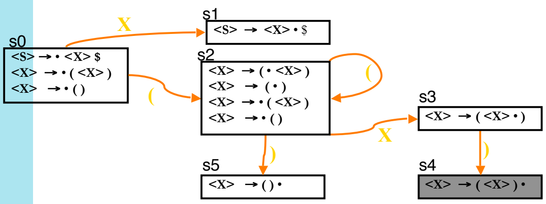
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4				
s5				



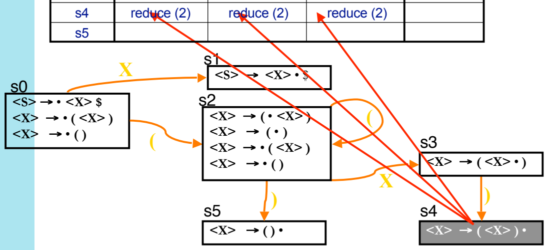
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4				
s5				



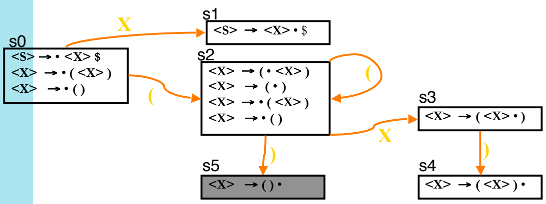
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5				



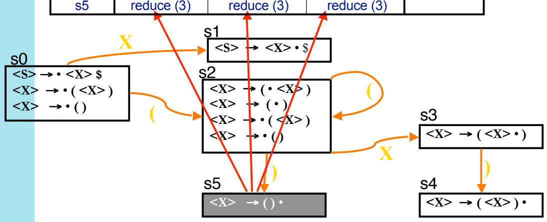
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5				



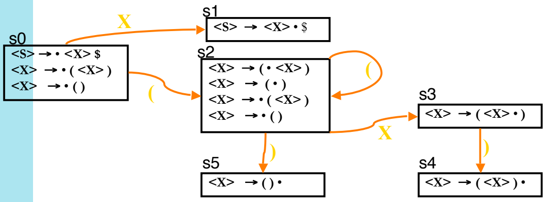
Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



Building Parse Table Example

State	Action (	Action)	Action $	Goto X
s0	shift to s2	error	error	goto s1
s1	error	error	accept	
s2	shift to s2	shift to s5	error	goto s3
s3	error	shift to s4	error	
s4	reduce (2)	reduce (2)	reduce (2)	
s5	reduce (3)	reduce (3)	reduce (3)	



Building a LR(0) parser engine

- Add the special production **S' → S** $
- Find the Items of the CFG
- Create the DFA
  - using **closure** and **goto** functions
- Build the Parse Table