Bottom-Up Parsing - I

Lecture 8

Section 4.5, 4.6

Rokan Uddin Faruqui

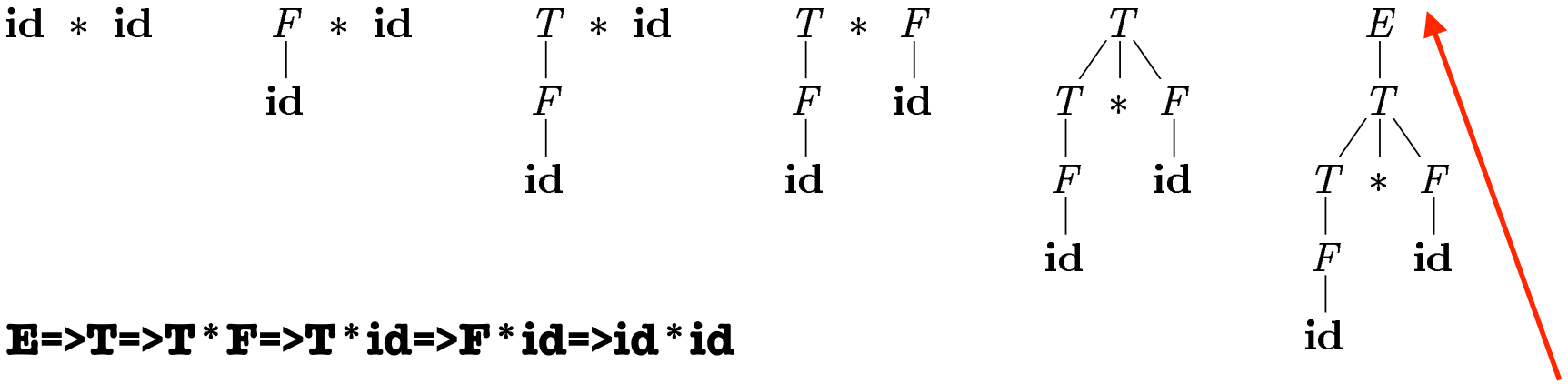
Associate Professor
Dept of Computer Science and Engineering
University of Chittaong, Bangladesh
Email: *rokan@cu.ac.bd*

Outline

E –> E + T | T
T–>T * F | F
F–> (E) | id

Bottom-Up Parsing

A bottom-up parse corresponds to the construction of a parse tree for an input string beginning at the leaves (the bottom) and working up towards the root (the top).



E=>T=>T * F=>T * id=>F * id=>id * id

Figure: A bottom-up parse for **id * id**

Reductions

F —> id

E ==>* w

- Bottom- up parsing is a process of reducing a string w to the start symbol of the grammar
- At each reduction step, a specific substring matching the body of a production is replaced by the nonterminal at the head of that production.
- The key decisions during bottom-up parsing are about
  - when to reduce
  - what production to apply

Handle Pruning

Right Sentential Form	Handle	Reducing Production
$\mathbf{id}_1 * \mathbf{id}_2$	$\mathbf{id}_1$	$F \to \mathbf{id}$
$F * \mathbf{id}_2$	F	$T \to F$
$T * \mathbf{id}_2$	$\mathbf{id}_2$	$F \to \mathbf{id}$
$T * F$	$T * F$	$T \to T * F$
T	T	$E \to T$

Shift-Reduce Parser

Shift-reduce parsing is a form of bottom-up parsing in which

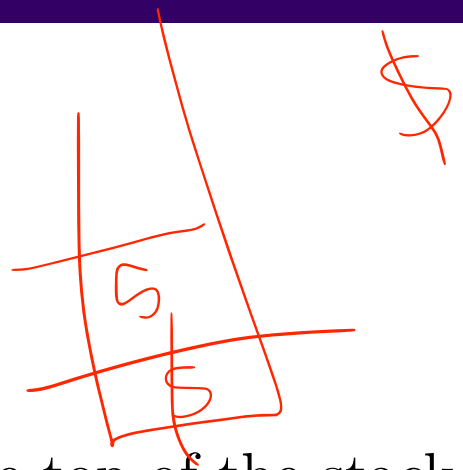
- a stack holds grammar symbols
- an input buffer holds the rest of the string to be parsed
- initial configuration

STACK	INPUT
$	w

- parser repeats this cycle until it has detected an error or until the stack contains the start symbol and the input is empty:

STACK	INPUT
$S	$

Shift-Reduce Parser



T-> T *F

Shift . Shift the next input symbol onto the top of the stack.

Reduce . The right end of the string to be reduced must be at the top of the stack. Locate the left end of the string within the stack and decide with what nonterminal to replace the string.

Accept . Announce successful completion of parsing.

Error . Discover a syntax error and call an error recovery routine.

Outline

LR Parsers

- A bottom-up parser follows a rightmost derivation from the bottom up.
- Such parsers typically use an LR algorithm and are called LR parsers.
- L means process tokens from Left to right.
- R means follow a Rightmost derivation, albeit, in reverse order.

LR Parsers

- Furthermore, in LR parsing, the production is applied only after the pattern has been matched.
- In LL (predictive) parsing, the production was selected, and then the tokens were matched to it.

Example

Example (Rightmost Derivations)

- Let the grammar be

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

Example

Example (Rightmost Derivation of $(\mathbf{id} + \mathbf{id}) * \mathbf{id}$)

$$
\begin{aligned}
E &\Rightarrow T \\
&\Rightarrow T*F \\
&\Rightarrow T*\mathbf{id} \\
&\Rightarrow F*\mathbf{id} \\
&\Rightarrow (E)*\mathbf{id} \\
&\Rightarrow (E+T)*\mathbf{id} \\
&\Rightarrow (E+F)*\mathbf{id} \\
&\Rightarrow (E+\mathbf{id})*\mathbf{id} \\
&\Rightarrow (T+\mathbf{id})*\mathbf{id} \\
&\Rightarrow (F+\mathbf{id})*\mathbf{id} \\
&\Rightarrow (\mathbf{id}+\mathbf{id})*\mathbf{id}
\end{aligned}
$$

LR Parsers

- An LR parser uses
  - A parse table (transition function)
  - An input buffer
    - A stack of "LR(0) states."
- Whereas a lexer is a DFA (deterministic finite automaton), an LR parser is a PDA (push-down automaton).
- A PDA is essentially a DFA (or NFA) with a stack.

LR Parsers

- The parser performs three types of operation.
  - Shift a token from the input buffer to the stack.
  - Reduce the contents of the stack by applying a production. That is, pop a sequence of tokens that match the body of a production and replace them with the head of the production.
  - Go to a new state.

LR Parsers

- An LR(0) parser (0 lookahead) reduces as soon as the body of a production is matched, regardless of the next token.
- An SLR parser (Simple LR) may or may not reduce, depending on the next token.
- An LR(1) parser (1 lookahead) is more complicated. It incorporates the next token as part of the current "state" of the process.
- An LALR parser (Look Ahead LR) is an LR(1) parser with its states much consolidated.

Outline

[E —> .E + T]
[E —> E . +T]
[E —> E +.T]
[E —> E + T.]

LR(0) Items

A —> .BcD
A —> B.cD
A —> Bc.D
A —> BcD.

E —> . E + T
E —> E . + T
E —> E + .T
E—> E+T.

Definition (LR(0) item)

An LR(0) item is a production with a special marker (•) that marks a position within the body of the production.

- To build an LR parse table, we must first find the LR(0) items.

Example

Example (LR(0) Items)

- If the production is

$$E \rightarrow E + T,$$

then the possible LR(0) items are

- $[E \rightarrow \bullet\ E + T]$
- $[E \rightarrow E\ \bullet\ + T]$
- $[E \rightarrow E + \bullet\ T]$
- $[E \rightarrow E + T\ \bullet]$

LR(0) Items

```
E —> E + .T
E —> E + T.
```

- The interpretation of $[A \rightarrow \alpha \bullet \beta]$ is

  "We have processed α and we might process β next."

- Whether we do actually process β will be borne out by the subsequent tokens.

Outline

LR Parsing

E' —> E
E —> E+T|T

- We will build a PDA whose states are sets of LR(0) items.
- First we augment the grammar with a new start symbol S'.

$$S' \rightarrow S$$

States of the PDA

{ [E' —> .E]}

- The initial state is called I_0 (item 0).
- State I_0 is the closure of the set

$$\{[S' \rightarrow \bullet\ S]\}.$$

S —> A

A —

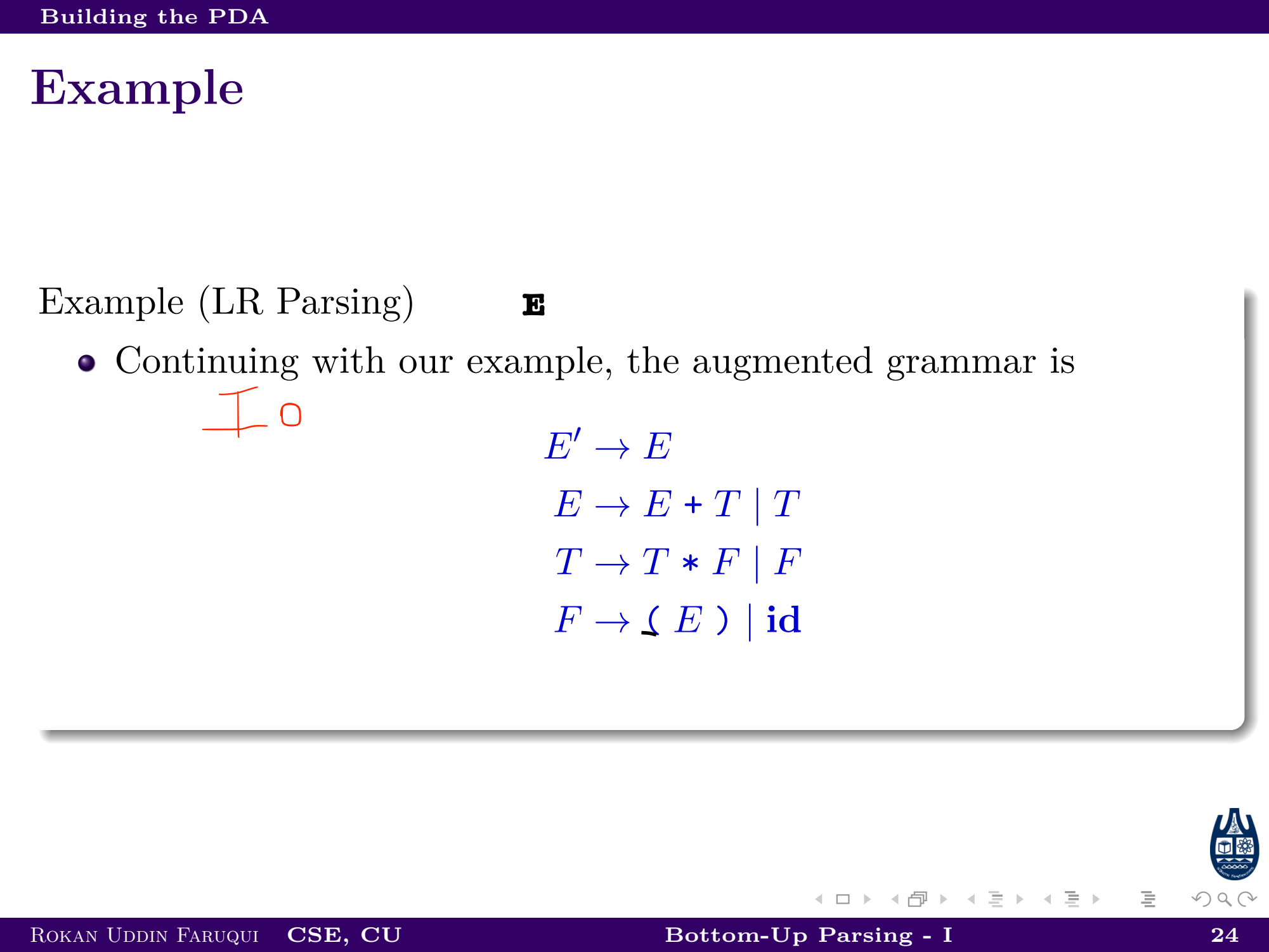
- To form the closure of a set of items
  - For each item $[A \rightarrow \alpha\ \bullet\ B\beta]$ in the set and for each production $B \rightarrow \gamma$ in the grammar, add the item $[B \rightarrow \bullet\ \gamma]$ to the set.
  - Continue in this manner until there is no further change.
- We will call $[B \rightarrow \bullet\ \gamma]$ an initial B-item.

Example

Example (LR Parsing)

- Continuing with our example, the augmented grammar is

$$E' \to E$$
$$E \to E + T \mid T$$
$$T \to T * F \mid F$$
$$F \to (E) \mid \mathbf{id}$$



Example

Example (The PDA States)

- The state I_0 consists of the items in the closure of item $[E' \rightarrow \bullet\ E]$.

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

Transitions

- There will be a transition from one state to another state for each grammar symbol that immediately follows the marker $\bullet$ in an item in that state.
- If the item $[A \rightarrow \alpha \bullet X\beta]$ is in the state, then
  - A transition from that state occurs when the symbol X is processed.
  - The transition is to the state that is the closure of the item $[A \rightarrow \alpha X \bullet \beta]$.

Example

Example (Building the PDA)

- Thus, from the state I_0, there will be transitions for the symbols E, T, F, (, **id**, and **num**.
- For example, on processing E, the items

$$[E' \to \bullet\ E] \text{ and } [E \to \bullet\ E + T]$$

become

$$[E' \to E\ \bullet] \text{ and } [E \to E\ \bullet\ + T].$$

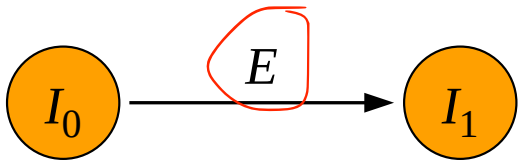
Example

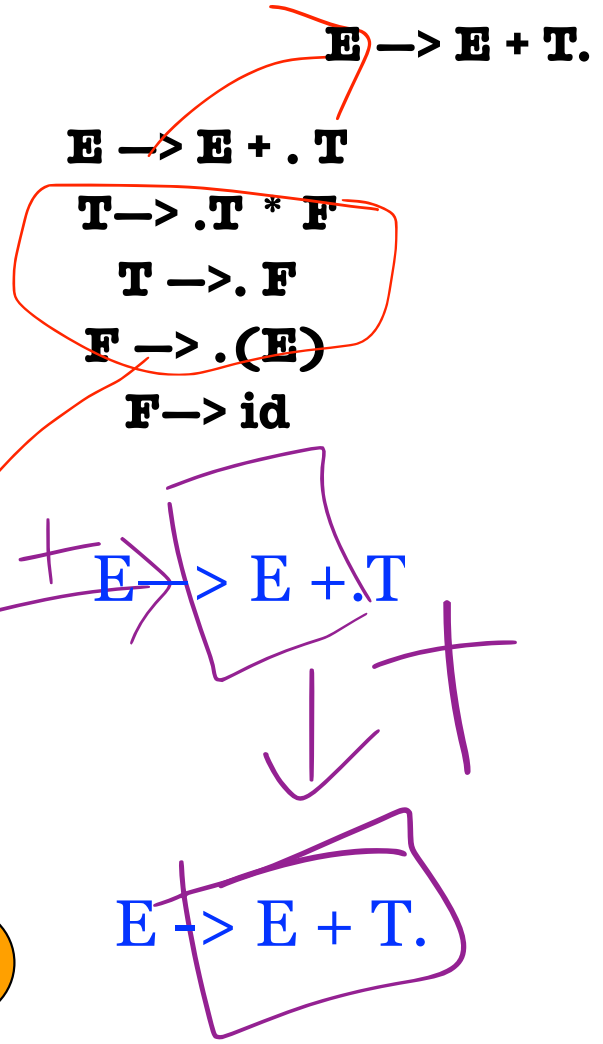
Example (Building the PDA)

- Let state I_1 be the closure of these items.

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

- Then the PDA has the transition





Example

Example (Building the PDA)

- Similarly we determine the other transitions from I_0.

On T	On F
I_2	I_3
$E \to T \bullet$ $T \to T \bullet * F$	$T \to F \bullet$

Example

Example (Building the PDA)

On (	On **id**
I_4	I_5
$F \to (\bullet E)$	
$E \to \bullet E + T$	
$E \to \bullet T$	
$T \to \bullet T * F$	$F \to \mathbf{id} \bullet$
$T \to \bullet F$	
$F \to \bullet (E)$	
$F \to \bullet \mathbf{id}$	

Example

Example (Building the PDA)

- Now find the transitions from states I_1 through I_5 to other states, and so on, until no new states appear.

Example

Example (Building the PDA)

On +	On *	On E
I_6	I_7	I_8
$E \to E + \bullet T$ $T \to \bullet T * F$ $T \to \bullet F$ $F \to \bullet (E)$ $F \to \bullet \mathbf{id}$	$T \to T * \bullet F$ $F \to \bullet (E)$ $F \to \bullet \mathbf{id}$	$F \to (E \bullet)$ $E \to E \bullet + T$

F —> (E).

Example

r5
FOLLOW(F) = {+, *, $,) }

Example (Building the PDA)

On T	On F	On $)$
I_9	I_{10}	I_{11}
$E \rightarrow E + T \bullet$ $T \rightarrow T \bullet * F$	$T \rightarrow T * F \bullet$	$F \rightarrow (E) \bullet$

Example

Example (Building the PDA)

		To											
		I_0	I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}
From	I_0		E	T	F	(	**id**						
	I_1							+					
	I_2								*				
	I_3												
	I_4			T	F	(	**id**			E			
	I_5												
	I_6				F	(	**id**				T		
	I_7					(	**id**					F	
	I_8							+					)
	I_9								*				
	I_{10}												
	I_{11}												

The transition table

Outline

Assignment

Homework

- Let the grammar be

$$E \rightarrow E + E \mid E * E \mid (E) \mid \mathbf{id}$$

- Write the LR(0) items for this grammar.
- Write the LR(0) states.
- Draw the transition diagram for the PDA of an SLR parser.

(continued...)

Assignment

Homework

- Let the grammar be

$$S \rightarrow (\ L \ ) \mid \mathbf{id}$$
$$L \rightarrow L \ , \ S \mid S$$

- Write the LR(0) items for this grammar.
- Write the LR(0) states.
- Draw the transition diagram for the PDA of an SLR parser.