## Section: LR Parsing

## LR PARSING

LR(k) Parser

- bottom-up parser
- shift-reduce parser
- L means: reads input left to right
- R means: produces a rightmost derivation
- k - number of lookahead symbols

LR parsing process

- convert CFG to PDA
- Use the PDA and lookahead symbols

Convert CFG to PDA The constructed NPDA:

- three states: s, q, f
start in state s, assume z on stack
- all rewrite rules in state $s$, backwards
rules pop rhs, then push lhs
$(\mathrm{s}, \mathrm{lhs}) \in \delta(\mathrm{s}, \lambda, \mathrm{rhs})$
This is called a reduce operation.
- additional rules in s to recognize terminals
For each $\mathbf{x} \in \Sigma, \mathbf{g} \in \Gamma,(\mathbf{s}, \mathbf{x g}) \in$ $\delta(\mathrm{s}, \mathrm{x}, \mathrm{g})$
This is called a shift operation.
- pop $S$ from stack and move into state $q$
- pop z from stack, move into f, accept.

Example: Construct a PDA.
$\mathrm{S} \rightarrow \mathrm{aSb}$
S $\rightarrow$ b

## LR Parsing Actions

1. shift
transfer the lookahead to the stack
2. reduce

For $X \rightarrow w$, replace $w$ by $X$ on the stack
3. accept
input string is in language
4. error input string is not in language

## LR(1) Parse Table

- Columns:
terminals, \$ and variables
- Rows:
state numbers: represent patterns in a derivation


## LR(1) Parse Table Example

1) $S \rightarrow a S b$
2) $S \rightarrow b$

|  | a | b | \$ | S |
| :---: | :---: | :---: | :---: | :---: |
| 0 | s2 | s3 |  | 1 |
| 1 |  |  | acc |  |
| 2 | s2 | s3 |  | 4 |
| 3 |  | r2 | r2 |  |
| 4 |  | s5 |  |  |
| 5 |  | r1 | r1 |  |

Definition of entries:
$\bullet \mathrm{sN}$ - shift terminal and move to state N

- N - move to state N
- rN - reduce by rule number N
- acc - accept
- blank - error
state $=0$
push(state)
read(symbol)
entry $=\mathbf{T}[$ state,symbol]
while entry.action $\neq$ accept do
if entry.action $==$ shift then
push(symbol)
state $=$ entry.state
push(state) read(symbol)
else if entry.action $==$ reduce then
do $2 *$ size_rhs times $\{\operatorname{pop}()\}$ state $:=$ top-of-stack() push(entry.rule.lhs) state $=\mathrm{T}[$ state,entry.rule.lhs] push(state)
else if entry.action $==$ blank then error
entry $=\mathbf{T}[$ state, symbol $]$
end while
if symbol $\neq \$$ then error


## Example:

Trace aabbb

# 5 <br> b <br>  <br> $\begin{array}{llllll}2 & 2 & 2 & 2 & 4 & 4\end{array}$ <br> $\begin{array}{llllll}a & a & a & a & S & S\end{array}$ <br> $\begin{array}{llllllll}2 & 2 & 2 & 2 & 2 & 2 & 2 & 1\end{array}$ <br> a $\begin{array}{lllllll}\text { a } & \text { a } & \text { a } & \text { a } & \text { a } & \text { a } & S\end{array}$ <br> $\begin{array}{lllllllll}0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0\end{array}$ <br> $\mathbf{S :} \quad \mathbf{z} \quad \mathbf{z}$ <br>  A: 

To construct the $\mathrm{LR}(1)$ parse table:

- Construct a dfa to model the top of the stack
- Using the dfa, construct an LR(1) parse table

To Construct the DFA

- Add $\mathrm{S}^{\prime} \rightarrow \mathrm{S}$
- place a marker "-" on the rhs $S^{\prime} \rightarrow$ S
- Compute closure $\left(S^{\prime} \rightarrow\right.$ _S $)$.

Def. of closure:

1. closure $\left(\mathbf{A} \rightarrow \mathbf{v \_ x y}\right)=\left\{\mathbf{A} \rightarrow \mathbf{v} \_\mathbf{x y}\right\}$ if $x$ is a terminal.
2. closure $(\mathrm{A} \rightarrow \mathrm{v} x y)=\{\mathrm{A} \rightarrow \mathrm{v} \mathrm{xy}\}$ $\cup$ (closure $\left(\mathbf{x} \rightarrow \_w\right)$ for all $w$ if $\mathbf{x}$ is a variable.

- The closure $\left(S^{\prime} \rightarrow-S\right)$ is state 0 and "unprocessed".
- Repeat until all states have been processed
- unproc $=$ any unprocessed state
- For each $x$ that appears in

A $\rightarrow$ u xv do

* Add a transition labeled "x" from state "unproc" to a new state with production $\mathbf{A} \rightarrow$ ux_v * The set of productions for the new state are: closure ( $\mathrm{A} \rightarrow$ ux_v)
* If the new state is identical to another state, combine the states Otherwise, mark the new state as "unprocessed"
- Identify final states.


## Example: Construct DFA

$$
\begin{aligned}
& \text { (0) } \mathrm{S}^{\prime} \rightarrow \mathrm{S} \\
& \text { (1) } \mathrm{S} \rightarrow \mathrm{aSb} \\
& \text { (2) } \mathrm{S} \rightarrow \mathrm{~b}
\end{aligned}
$$

## Backtracking through the DFA

Consider aabbb

- Start in state 0.
- Shift "a" and move to state 2.
- Shift "a" and move to state 2.
- Shift "b" and move to state 3.

Reduce by " $\mathrm{S} \rightarrow \mathrm{b}$ "
Pop "b" and Backtrack to state 2. Shift " S " and move to state 4.

- Shift "b" and move to state 5 .

Reduce by " $S \rightarrow$ aSb"
Pop "aSb" and Backtrack to state 2.

Shift " $S$ " and move to state 4.

- Shift "b" and move to state 5.

Reduce by " $\mathrm{S} \rightarrow \mathrm{aSb}$ "
Pop "aSb" and Backtrack to state 0.

Shift " $S$ " and move to state 1. - Accept. aabbb is in the language.

To construct $\mathrm{LR}(1)$ table from diagram:

1. If there is an arc from state 1 to state2
(a) arc labeled $x$ is terminal or $\$$ $\mathrm{T}[$ state $1, \mathrm{x}]=$ sh state 2
(b) arc labeled $X$ is nonterminal $\mathrm{T}[$ state $1, \mathrm{X}]=$ state 2
2. If state1 is a final state with $\mathbf{X} \rightarrow \mathbf{w}_{-}$
For all a in FOLLOW(X), $\mathrm{T}[$ state $1, \mathrm{a}]=$ reduce by $\mathbf{X} \rightarrow \mathbf{w}$
3. If state1 is a final state with
$\mathrm{S}^{\prime} \rightarrow \mathrm{S}$
T[state1, $\$]=$ accept
4. All other entries are error

Example: LR(1) Parse Table
(0) $S^{\prime} \rightarrow S$
(1) $\mathrm{S} \rightarrow \mathrm{aSb}$
(2) $\mathrm{S} \rightarrow \mathrm{b}$

Here is the LR(1) Parse Table with extra information about the stack contents of each state.

| Stackcontents | Statenumber | Terminals |  |  | Variables |
| :---: | :---: | :---: | :---: | :---: | :---: |
|  |  | a | b | \$ | S |
| (empty) | 0 |  |  |  |  |
|  | 1 |  |  |  |  |
|  | 2 |  |  |  |  |
|  | 3 |  |  |  |  |
|  | 4 |  |  |  |  |
|  | 5 |  |  |  |  |

Actions for entries in LR(1) Parse table T[state,symbol]

Let entry $=\mathrm{T}[$ state, symbol$]$.

- If symbol is a terminal or \$
- If entry is "shift state $i$ "
push lookahead and state $i$ on the stack
- If entry is "reduce by rule $\mathrm{X} \rightarrow$ w"
pop $w$ and $k$ states ( $k$ is the size of $w$ ) from the stack.
- If entry is "accept"

Halt. The string is in the language.

- If entry is "error"

Halt. The string is not in the language.

- If symbol is nonterminal We have just reduced the rhs of a production $\mathrm{X} \rightarrow \mathrm{w}$ to a symbol. The entry is a state number, call it state $i$. Push T[state $i, \mathrm{X}]$ on the stack.

Constructing Parse Tables for CFG's with $\lambda$-rules
$\mathbf{A} \rightarrow \lambda$ written as $\mathbf{A} \rightarrow \lambda_{-}$

## Example

$$
\begin{aligned}
& \mathbf{S} \rightarrow \mathbf{d d X} \\
& \mathbf{X} \rightarrow \mathbf{a X} \\
& \mathbf{X} \rightarrow \lambda
\end{aligned}
$$

Add a new start symbol and number the rules:
(0) $S^{\prime} \rightarrow S$
(1) $S \rightarrow$ ddX
(2) $\mathrm{X} \rightarrow \mathrm{aX}$
(3) $\mathrm{X} \rightarrow \lambda$

Construct the DFA:

## Construct the LR(1) Parse Table

|  | a | d | $\$$ | S | X |
| :--- | :--- | :--- | :--- | :--- | :--- |
| $\mathbf{0}$ |  |  |  |  |  |
| $\mathbf{1}$ |  |  |  |  |  |
| $\mathbf{2}$ |  |  |  |  |  |
| $\mathbf{3}$ |  |  |  |  |  |
| $\mathbf{4}$ |  |  |  |  |  |
| $\mathbf{5}$ |  |  |  |  |  |
| $\mathbf{6}$ |  |  |  |  |  |

## Possible Conflicts:

1. Shift/Reduce Conflict

Example:

$$
\begin{aligned}
& \mathrm{A} \rightarrow \mathrm{ab} \\
& \mathrm{~A} \rightarrow \text { abcd }
\end{aligned}
$$

In the DFA:

$$
\begin{aligned}
& \mathrm{A} \rightarrow \mathrm{ab}_{-} \\
& \mathrm{A} \rightarrow \mathrm{ab}_{-} \mathrm{cd}
\end{aligned}
$$

2. Reduce/Reduce Conflict

Example:

$$
\begin{aligned}
& \mathrm{A} \rightarrow \mathrm{ab} \\
& \mathrm{~B} \rightarrow \mathrm{ab}
\end{aligned}
$$

In the DFA:

$$
\begin{aligned}
& \mathrm{A} \rightarrow \mathrm{ab}_{-} \\
& \mathrm{B} \rightarrow \mathbf{a b}_{-}
\end{aligned}
$$

3. Shift/Shift Conflict
