Predictive parser

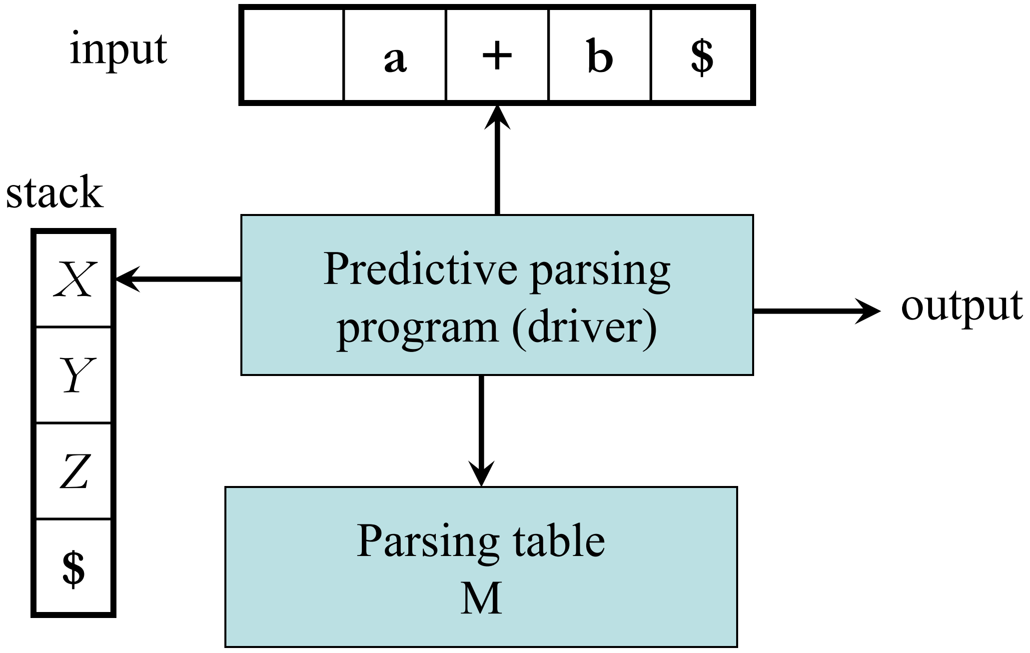
Session Outcomes

- At the end of this session, participants will be able to
  - Design predictive parser for the given grammar

Outline

- Steps for designing predictive parser
- FIRST
- FOLLOW
- Parsing table construction algorithm
- Predictive parsing algorithm

Predictive Parsing



Compute FIRST

- If X is a terminal symbol
  - ➔ FIRST(X)={X}
- If X is a non-terminal symbol and $X \to \varepsilon$ is a production rule
  - ➔ ε is in FIRST(X).
- If X is a non-terminal symbol and $X \to a\alpha$ is a production rule
  - ➔ a is in FIRST(X).

- If X is a non-terminal symbol and $X \to$ Y1Y2..Yn is a production rule
  - ➔ If X is Y1Y2..Yn

    if a terminal **a** in FIRST(Yi) and ε is in all FIRST(Yj) for j=1,...,i-1

    then **a** is in FIRST(X).
  - ➔ if ε is in all FIRST(Yj) for j=1,...,n

    then ε is in FIRST(X).

FIRST Example

$E \rightarrow TE'$
$E' \rightarrow +TE' \mid \varepsilon$
$T \rightarrow FT'$
$T' \rightarrow *FT' \mid \varepsilon$
$F \rightarrow (E) \mid id$

FIRST(F) = {(,id}
FIRST(T') = {*, ε}
FIRST(T) = {(,id}
FIRST(E') = {+, ε}
FIRST(E) = {(,id}

FIRST(TE') = {(,id}
FIRST(+TE') = {+}
FIRST(ε) = {ε}
FIRST(FT') = {(,id}
FIRST(*FT') = {*}
FIRST(ε) = {ε}
FIRST((E)) = {(}
FIRST(id) = {id}

Compute FOLLOW

- If S is the start symbol ➔ \$ is in FOLLOW(S)

- if $A \rightarrow \alpha B\beta$ is a production rule
  ➔ everything in FIRST(β) is FOLLOW(B) except ε

- If ($A \rightarrow \alpha B$ is a production rule) or
  ($A \rightarrow \alpha B\beta$ is a production rule and ε is in FIRST(β))
  ➔ everything in FOLLOW(A) is in FOLLOW(B).

We apply these rules until nothing more can be added to any follow set.

FOLLOW Example

$E \rightarrow TE'$

$E' \rightarrow +TE' \mid \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

FOLLOW(E) = { \$,) }

FOLLOW(E') = { \$,) }

FOLLOW(T) = { +,), \$ }

FOLLOW(T') = { +,), \$ }

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

Constructing LL(1) Parsing Table

For each production rule $A \rightarrow \alpha$ of a grammar G

- For each terminal a in FIRST(α)
  ➔ add $A \rightarrow \alpha$ to M[A,a]
- If ε in FIRST(α)
  ➔ for each terminal a in FOLLOW(A) add $A \rightarrow \alpha$ to M[A,a]
- If ε in FIRST(α) and $ in FOLLOW(A)
  ➔ add $A \rightarrow \alpha$ to M[A,$]

- All other undefined entries of the parsing table are error entries.

Constructing LL(1) Parsing Table

$E \to TE'$	FIRST(TE')={(,id}	➔ $E \to TE'$ into M[E,(] and M[E,id]
$E' \to +TE'$	FIRST($+TE'$)={+}	➔ $E' \to +TE'$ into M[E',+]
$E' \to \varepsilon$	FIRST(ε)={ε}	➔ none
	but since ε in FIRST(ε)	
	and FOLLOW(E')={$,)}	➔ $E' \to \varepsilon$ into M[E',$] and M[$E'$,)]
$T \to FT'$	FIRST(FT')={(,id}	➔ $T \to FT'$ into M[T,(] and M[T,id]
$T' \to *FT'$	FIRST($*FT'$)={*}	➔ $T' \to *FT'$ into M[T',*]
$T' \to \varepsilon$	FIRST(ε)={ε}	➔ nonebut since ε in FIRST(ε)
	and FOLLOW(T')={$,),+}	➔ $T' \to \varepsilon$ into M[T',$], M[$T'$,)] and [$T'$,+]
$F \to (E)$	FIRST((E))={(}	➔ $F \to (E)$ into M[F,(]
$F \to id$	FIRST(id)={id}	➔ $F \to id$ into M[F,id]

Example Table

$E \to T\,E'$
$E' \to + T\,E' \mid \varepsilon$
$T \to F\,T'$
$T' \to * F\,T' \mid \varepsilon$
$F \to (\,E\,) \mid$ **id**

$A \to \alpha$	FIRST(α)	FOLLOW(A)
$E \to T\,E'$	(id	$)
$E' \to + T\,E,$	+	$)
$E' \to \varepsilon$	ε	$)
$T \to F\,T'$	(id	+ $)
$T' \to * F\,T'$	*	+ $)
$T' \to \varepsilon$	ε	+ $)
$F \to (\,E\,)$	(	* + $)
$F \to$ **id**	id	* + $)

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

Predictive Parsing Program

Set ip to point to the first symbol of w**$;**
repeat
    let X be the top stack symbol and a the symbol pointed by ip;
    if X is a terminal or **$ then**
        if X = a then
            pop X from the stack and advance ip
        else error();
    else
        if M[X,a] = X $\rightarrow$ Y1Y2…Yk **then**
        begin
            pop X from the stack;
            push(Y_k, Y_{k-1}, …, Y_2, Y_1) ;   // such that Y_1 is on top
            output the production X $\rightarrow$ Y1Y2…Yk ;
        end
        else   error();
until X = **$** /* S tack is empty */

Example

Stack	Input	Production applied
$E	id+id*id$	$E \rightarrow T\ E'$
$E'T	id+id*id$	$T \rightarrow F\ T'$
$E'T'F	id+id*id$	$F \rightarrow$ **id**
$E'T'**id**	id+id*id$	
$E'T'	+id*id$	$T' \rightarrow \varepsilon$
$E'	+id*id$	$E' \rightarrow +\ T\ E'$
$E'T+	+id*id$	
$E'T	id*id$	$T \rightarrow F\ T'$
$E'T'F	id*id$	$F \rightarrow$ **id**
$E'T'**id**	id*id$	
$E'T'	*id$	$T' \rightarrow *\ F\ T'$
$E'T'F*	*id$	
$E'T'F	id$	$F \rightarrow$ **id**
$E'T'**id**	id$	
$E'T'	$	$T' \rightarrow \varepsilon$
$E'	$	$E' \rightarrow \varepsilon$
$	$	

Summary

- FIRST
- FOLLOW
- Parsing table
- Parsing algorithm

Check your understanding?

Compute First and follow for the following grammar.

(a) $S \rightarrow A$

$A \rightarrow aB$ / Ad

$B \rightarrow b$

$C \rightarrow g$

(b) $S \rightarrow (L)$ / a

$L \rightarrow L , S$ / S

Construct predictive parsing table for the above grammars