CS143: Parsing II

David L. Dill

Stanford University

Parsing (and lexical analysis wrapup)

- Parsing
  - “Dangling Else” Ambiguity
  - Useless Symbols & Productions
  - Top-down Parsing
  - Left Factors and Left Recursion
  - LL(1) Parsing
  - Nullable Strings

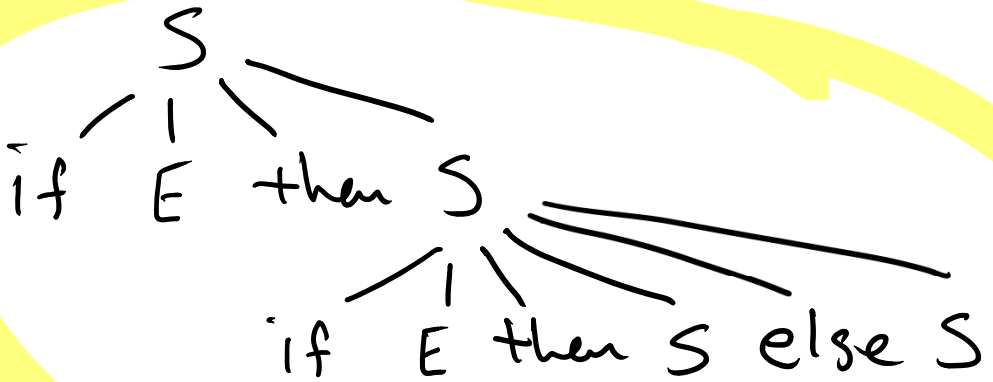
“Dangling Else” Ambiguity

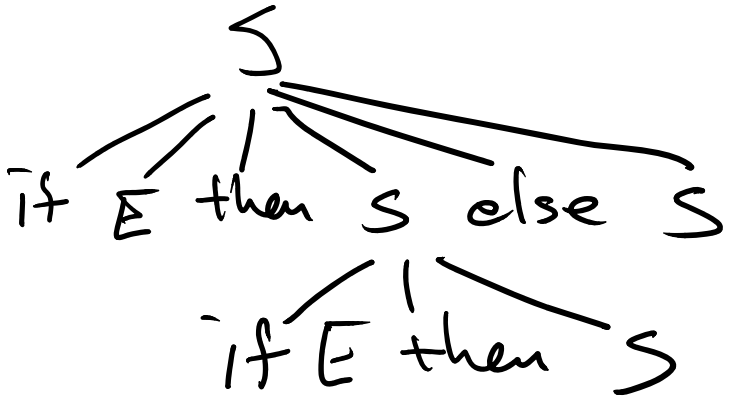
"Dangling Else"

Algol 60

$S \rightarrow$ if E then S else S

$S \rightarrow$ if E then S





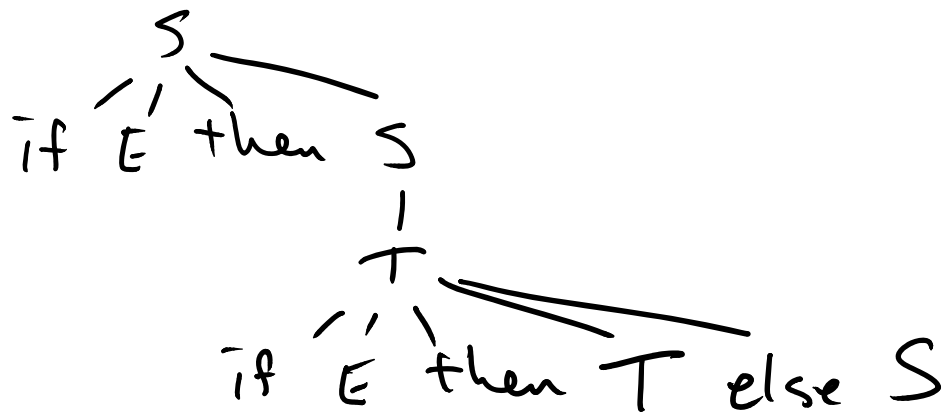
← Preferred.

Dangling Else – Unambiguous

S → if E then S ← put missing else's at top of tree

S → T

T → if E then T else S ← T always adds "else"



Useless Symbols & Productions

$S \to SAB$

$S \to a$

$A \to AA$

$B \to b$

Def: A symbol $X \in V \cup \Sigma$ is useless if there is no α, β, w such that

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

no way to derive a terminal string from A

$S \to SAB$

$S \to a$

$A \to AA$

$B \to b$

Def: A symbol $X \in V \cup \Sigma$ is useless if there is no α, β, w such that

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

Can't use this to derive a terminal string

$S \to SAB$

$S \to a$

$A \to AA$

$B \to b$

|

No way to to derive B without deriving A.

Def: A symbol $X \in V \cup \Sigma$ is **useless** if there is no α, β, w such that

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

$S \to SAB$

$S \to a$

$A \to AA$

$B \to b$

The only useful production

Def: A symbol $X \in V \cup \Sigma$ is **useless** if there is no α, β, w such that

$$S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w$$

$$L(G) = \{a\}$$

From now on, we assume

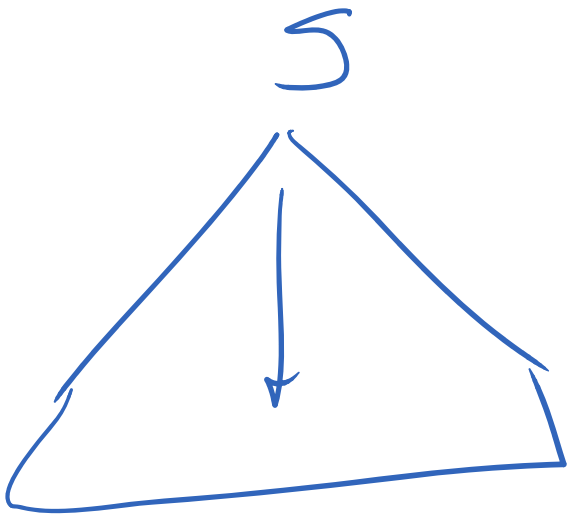
NO USELESS PRODUCTIONS IN EXAMPLE GRAMMARS*

* unless otherwise stated.

Top-Down Parsing

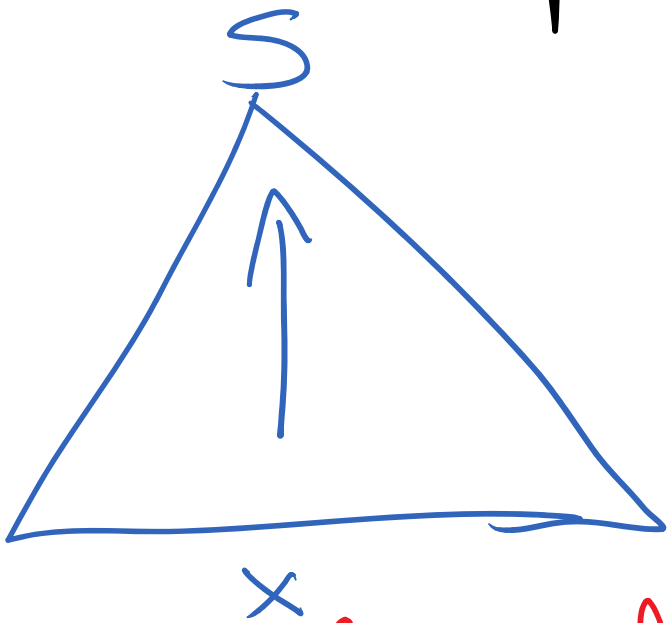
Parse: Given a CFG, $G = (V, \Sigma, R, S)$ and $x \in \Sigma^*$, find a derivation of x from S, if one exists.

Top-down parsing



Build tree from S down to x

Bottom-up



Build tree from x to S.

$S \rightarrow A\,S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b

S

$S \to AS$ ← which one? S

$S \to B$ ←

$A \to a$

$B \to b$

Input: a a b

For now)
"guess" the
right one.
(non-determinism)

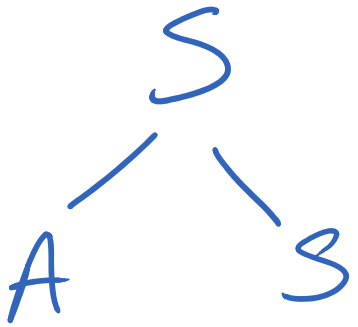
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



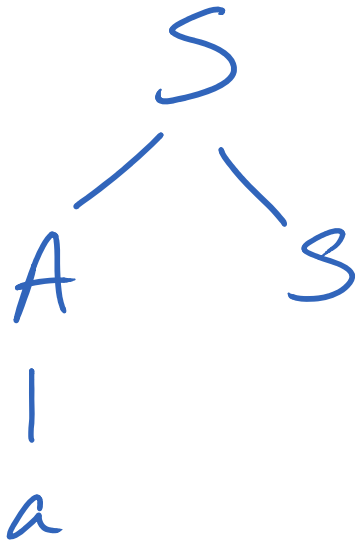
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



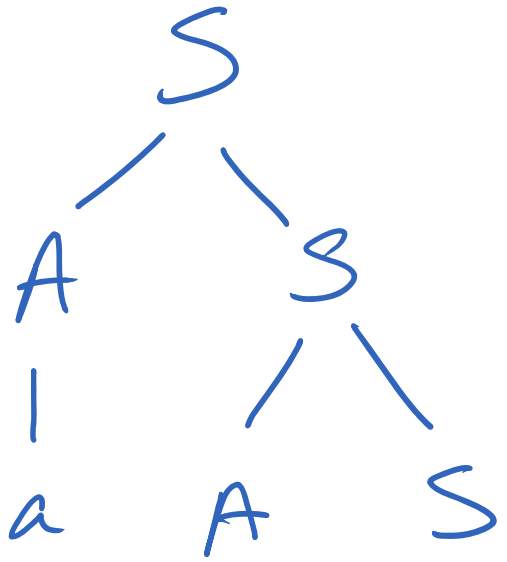
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



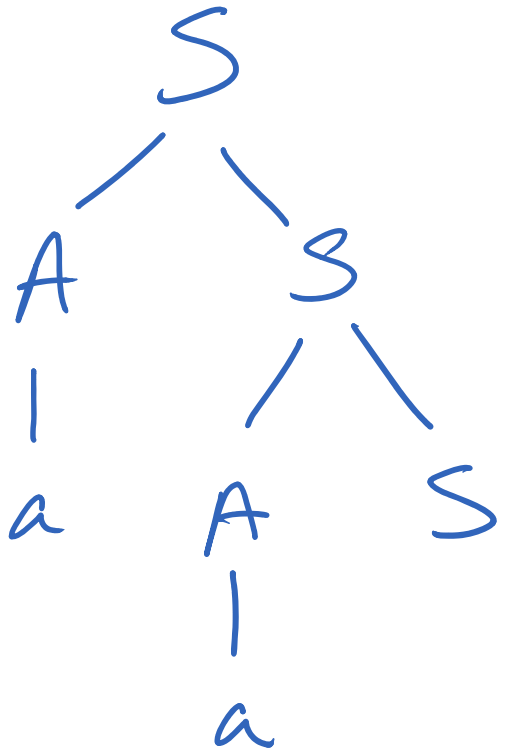
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



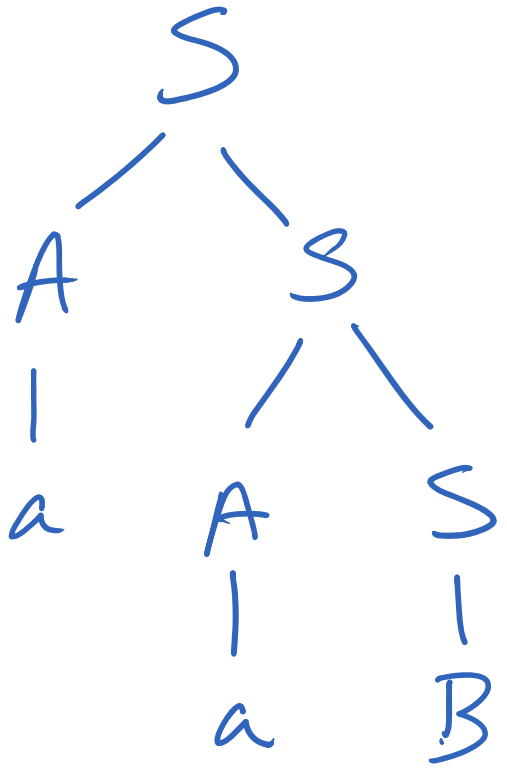
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



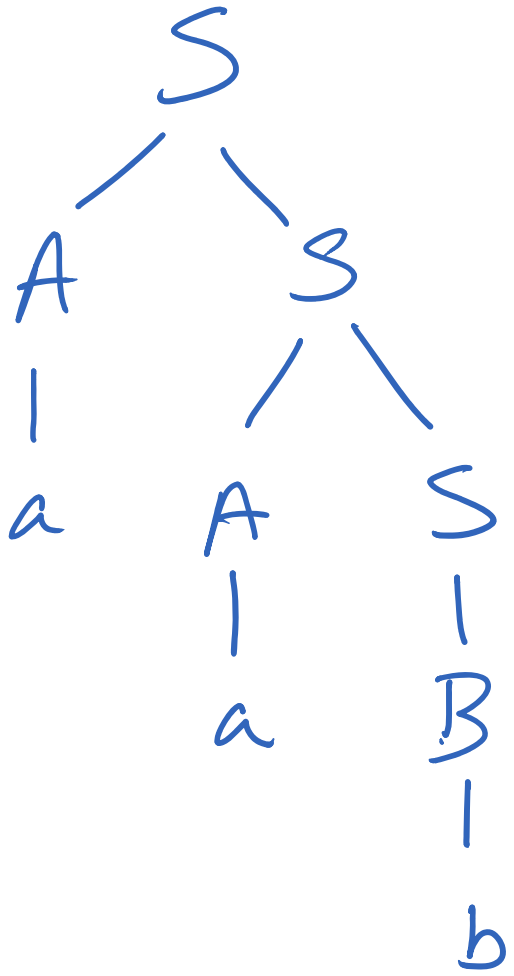
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



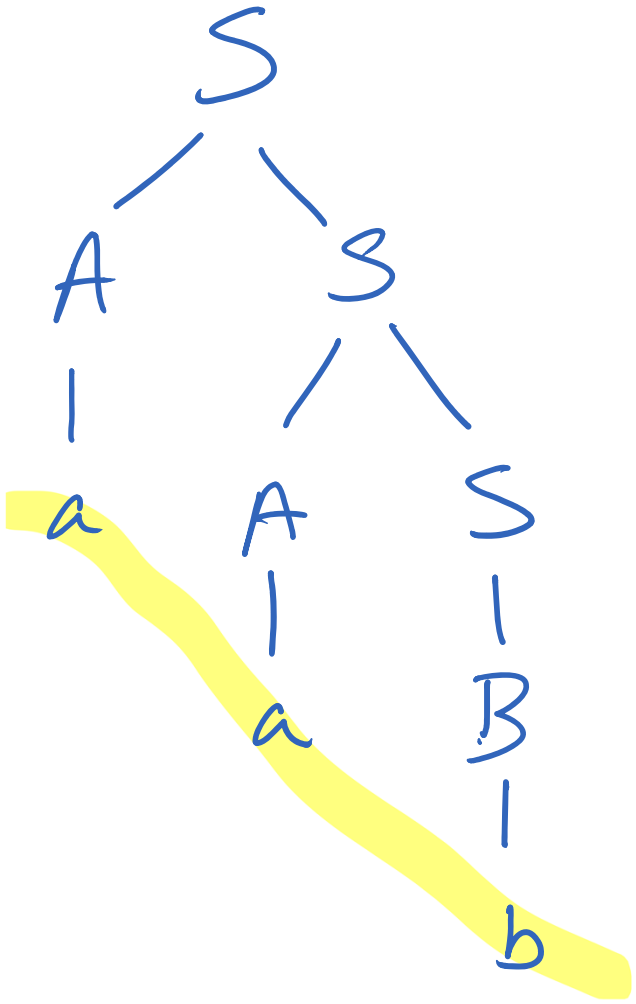
$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: a a b



$S \rightarrow A\ S$

$S \rightarrow B$

$A \rightarrow a$

$B \rightarrow b$

Input: aab

input	(top) stack	action
aab$	S $	EXPAND
aab$	A S $	EXPAND
aab$	a S $	MATCH
ab$	S $	EXPAND
ab$	A S $	EXPAND
ab$	a S $	MATCH
b$	S $	EXPAND
b$	B $	EXPAND
b$	b $	MATCH
$	$	ACCEPT

$S \to A\,S$

$S \to B$

which is the best guess?

$A \to a$

$B \to b$

S

Input: a a b

$S \to A\,S$

$S \to B$ ← BAD

$A \to a$

$B \to b$

$B \not\Rightarrow a \ldots$

S

Input: a a b

$S \to AS$

$S \to B$

$A \to a$

$B \to b$

Input: a a b

S

Lookahead heuristic:

Choose production that can derive the next few input symbols.

$B \overset{*}{\Rightarrow} bm$

$AS \overset{*}{\Rightarrow} a\,m\,m$

Left Factors and Left Recursion

One Symbol Lookahead

$S \rightarrow Sa$ ← which one?

$S \rightarrow b$ ←

Both derive strings that start with "b"

Input: baaaaa

Left recursion is fatal for most efficient top-down parsing algorithms.

Eliminating immediate left recursion.

$A \rightarrow Am$

My recipe: Convert to EBNF and back.

$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

$A \to \beta_2$

① Combine all A productions into one, using "|"

$A \to A(\alpha_1 | \alpha_2) | (\beta_1 | \beta_2)$

RHS's beginning with A

$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

$A \to \beta_2$

① Combine all A productions into one, using "|"

$$A \to A(\alpha_1 | \alpha_2) | (\beta_1 | \beta_2)$$

RHS's not beginning with A.

$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

$A \to \beta_2$

① Combine all A productions into one, using "|"

$$A \to A\underbrace{(\alpha_1 \mid \alpha_2)}_{\alpha} \mid \underbrace{(\beta_1 \mid \beta_2)}_{\beta}$$

$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

$A \to \beta_2$

① Combine all A productions into one, using "|"

$A \to A\alpha \mid \beta$

$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

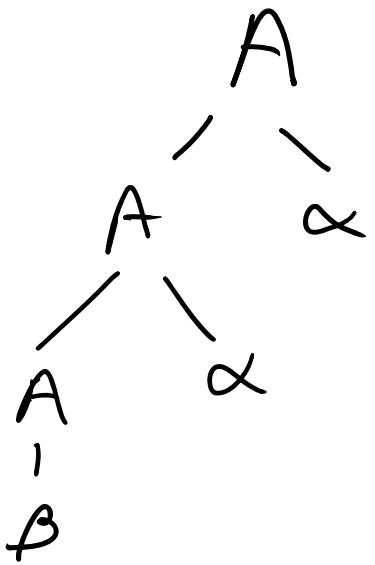
$A \to \beta_2$

① Combine all A productions into one, using "|"

$A \to A\alpha \mid \beta$

② Convert recursion to *

$A \to \beta\alpha^*$

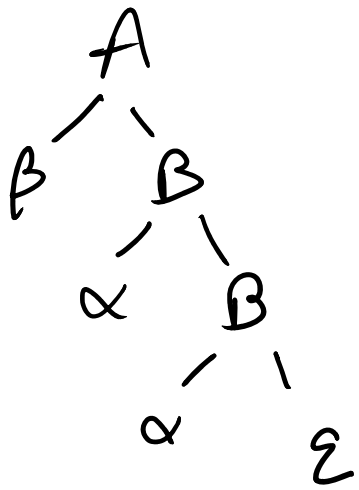


$A \to A\alpha_1$

$A \to A\alpha_2$

$A \to \beta_1$

$A \to \beta_2$



① Combine all A productions into one, using "|"

$A \to A\alpha \mid \beta$

② Convert recursion to *

$A \to \beta\alpha^*$

③ Expand right recursively

$A \to \beta B$

$B \to \alpha B \mid \varepsilon$

$E \rightarrow E + T$

$E \rightarrow T$

$E \rightarrow E + T \mid T$

← combine E productions

$E \rightarrow E + T$

$E \rightarrow T$

$E \rightarrow E + T \mid T$

$E \rightarrow T (+T)^*$

← Convert to *

$E \rightarrow E + T$

$E \rightarrow T$

$E \rightarrow E + T \mid T$

$E \rightarrow T\,(+T)^*$

$E \rightarrow T\,A$

$A \rightarrow +\,T\,A$

$A \rightarrow \varepsilon$

← expand right recursively

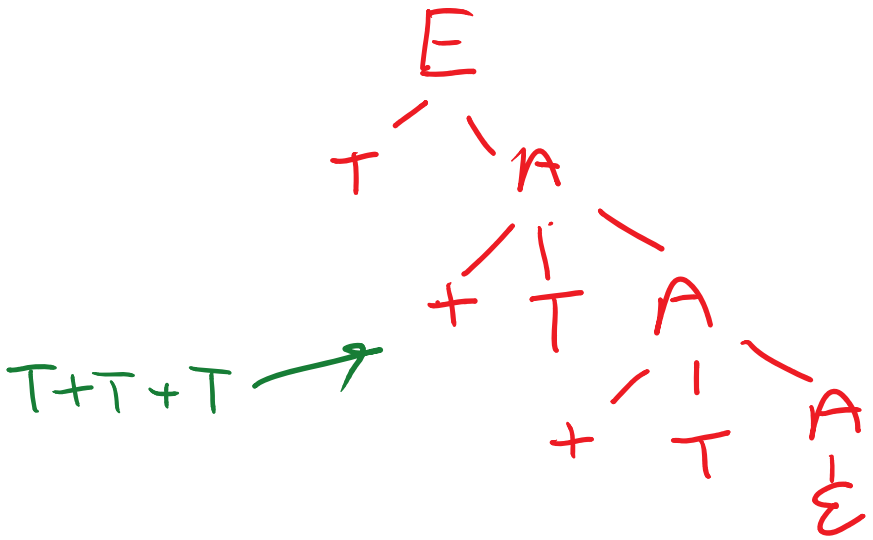
$E \to E + T$
$E \to T$

$E \to E + T \mid T$

$E \to T\,(+T)^*$

$E \to TA$
$A \to +TA$
$A \to \varepsilon$

← expand right recursively



T+T+T →

Random Question Break

Left Factors

$S \rightarrow A\ a$

$S \rightarrow A\ b$

obviously have same first lookaheads

Left Factoring

Convert to EBNF, then convert back.

$$A \to \alpha\beta$$
$$A \to \alpha\gamma$$

$$A \to \alpha(\beta \mid \gamma)$$

← Combine A productions with common left factors

Left Factoring

Convert to EBNF, then convert back.

$$A \to \alpha\beta$$
$$A \to \alpha\gamma$$

$$A \to \alpha(\beta \mid \gamma)$$

$$A \to \alpha B$$
$$B \to \beta$$
$$B \to \gamma$$

← convert back to CFG, without copying left factors

LL(1) Parsing

Idea: Base production choice on one symbol of lookahead.

Heuristic — does not work for all CFG's or CFLs.

LL(1) — 1 symbol lookahead

L — left-to-right

L — leftmost derivation

$$E \rightarrow E + T \mid T$$
$$T \rightarrow T * F \mid F$$
$$F \rightarrow n \mid (E)$$

Simplified unambiguous expression CFG

$E \rightarrow E + T \mid T$

$T \rightarrow T * F \mid F$

$F \rightarrow n \mid (E)$

$E \rightarrow TA$

$A \rightarrow +TA$

$A \rightarrow \varepsilon$

$T \rightarrow FB$

$B \rightarrow *FB$

$B \rightarrow \varepsilon$

$F \rightarrow (E)$

$F \rightarrow n$

← after left - recursion removal.

Parsing with Stack

input	(top) stack	action
aab$	S$	EXPAND
⋮	⋮	⋮

Parse loop:

If input, top of stack are $, accept

If top of stack $\in \Sigma$, match

If top of stack $\in V$ ↙ replaces guess

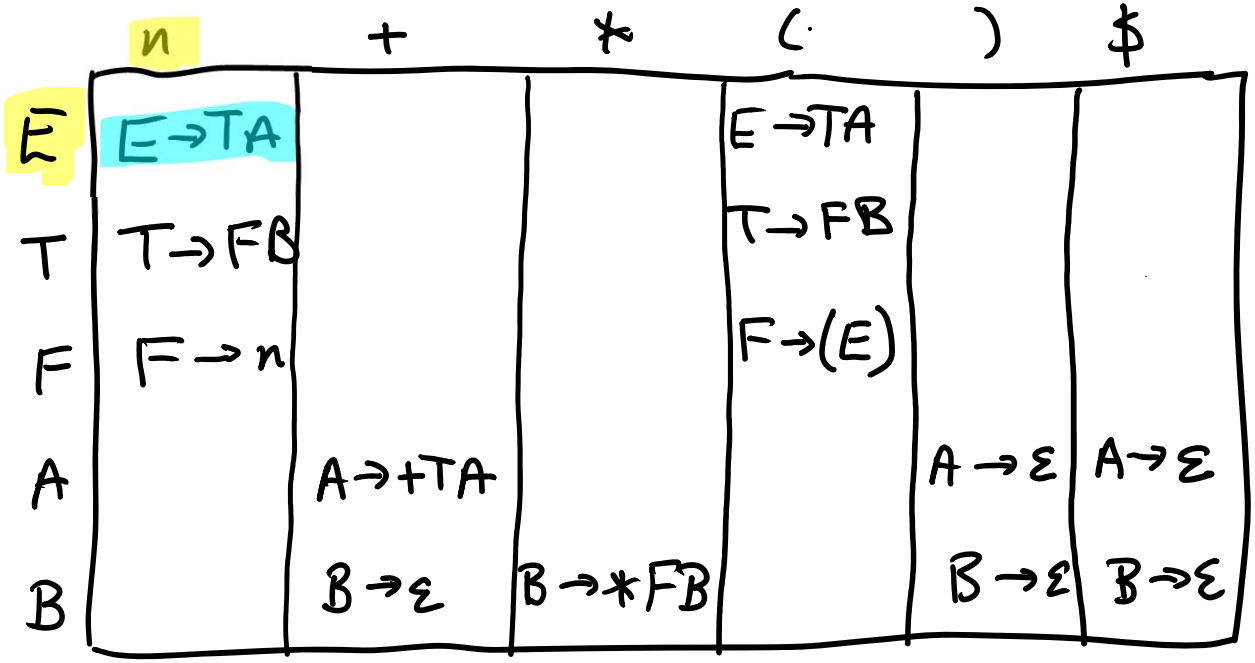
expand TABLE[TOS, nextinput]

LL(1) Parse Table

Input

Top of Stack (V)

V	n	+	*	(	)	$
E	E→TA			E→TA		
T	T→FB			T→FB		
F	F→n			F→(E)		
A		A→+TA			A→ε	A→ε
B		B→ε	B→*FB		B→ε	B→ε

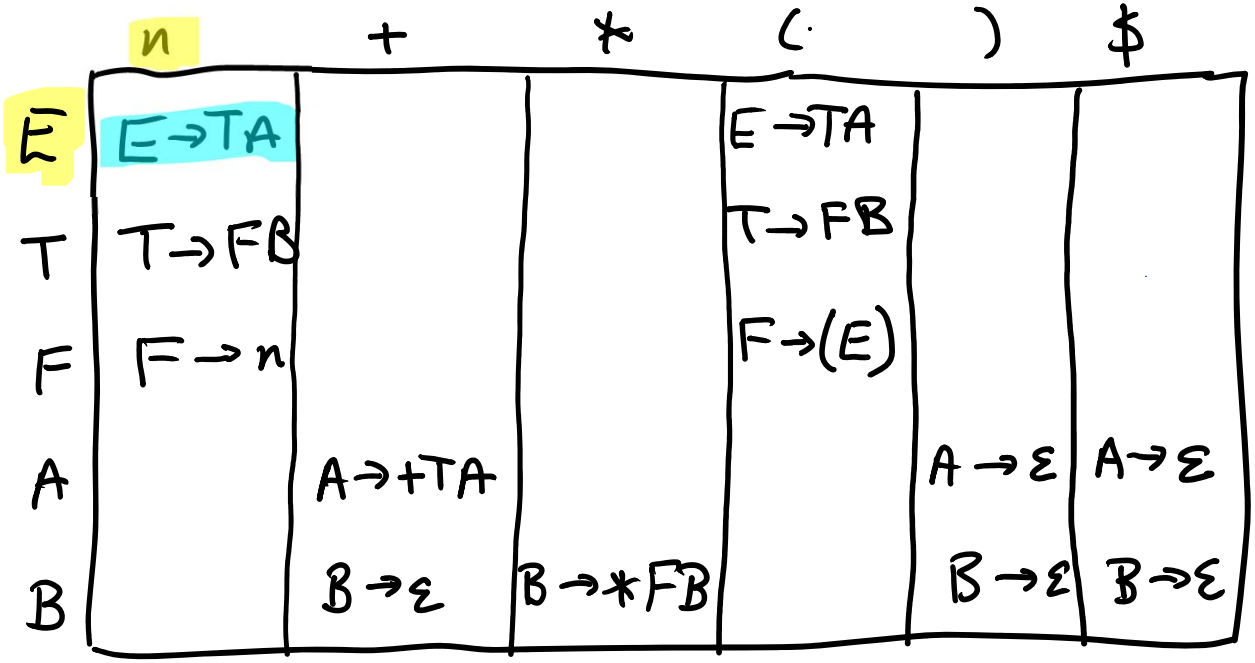


Input

n+n*n$

stack

E$



input

n + n ∩ $

stack

E $

T A $

Def: α is nullable if $\alpha \overset{*}{\Rightarrow} \varepsilon$ in CFG G

Algorithm: Iterative application of rules

1. Start with initial value
2. Update only when rules require
3. Stop when no more updates required

$Nl[\alpha]$ = true iff $\alpha \overset{*}{\Rightarrow} \varepsilon$ in CFG G

$Nl[\alpha]$ = false for all α initially.

Rules:

(only use when α is RHS of some production)

$Nl[\alpha]$ = true if $\alpha = X_1 X_2 \ldots X_n$

and $Nl[X_i]$ = true for all i.

Note: $\alpha = \varepsilon$ is trivially nullable

$Nl[\alpha] = \text{true}$ iff $\alpha \stackrel{*}{\Rightarrow} \varepsilon$ in CFG G

$Nl[\alpha] = \text{false}$ for all α initially.

Rules:

$Nl[\alpha] = \text{true}$ if $\alpha = X_1 X_2 \ldots X_n$ and $Nl[X_i] = \text{true}$ for <u>all</u> i.

$Nl[A] = \text{true}$ if there <u>exists</u> a production $A \to \beta$ and $Nl(\beta)$

Example:

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$

Nl

ε	F
a	F
A	F
B	F
AB	F
S	F

Example:

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$

Nl

ε	~~F~~ T
a	F
A	F
B	F
AB	F
S	F

Example

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$

Nl

ε	~~F~~	T
a	F	
A	~~F~~	T
B	F	
AB	F	
S	F	

Example

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$

Nl

ε	~~F~~	T
a	F	
A	~~F~~	T
B	~~F~~	T
AB	F	
S	F	

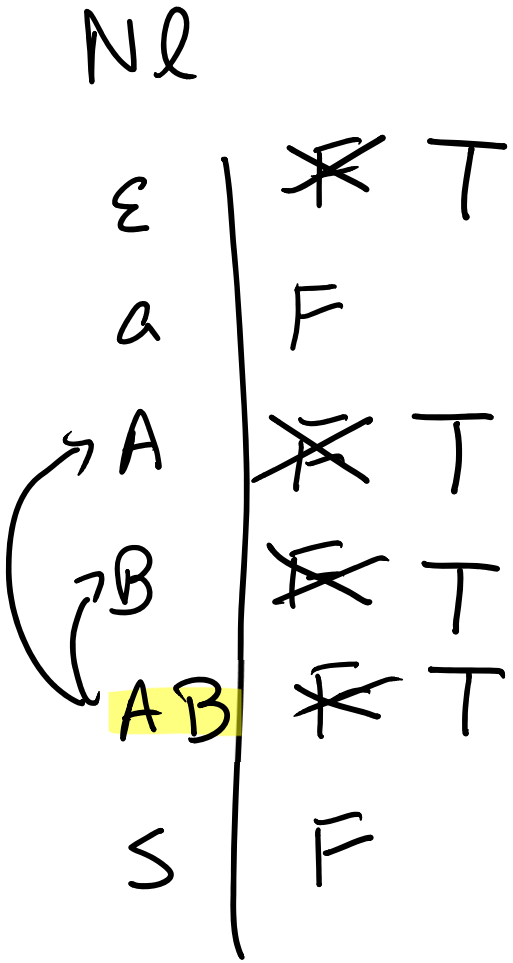
Example:

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$



Example:

$S \to AB$

$S \to a$

$A \to \varepsilon$

$B \to \varepsilon$

Nl

ε	~~F~~	T
a	F	
A	~~F~~	T
B	~~F~~	T
AB	~~F~~	T
S	~~F~~	T

$E \to TA$

$A \to +TA$

$A \to \varepsilon$

$T \to FB$

$B \to *FB$

$B \to \varepsilon$

$F \to (E)$

$F \to n$

A, B are the only nullable non-terminals