Tianjin International Engineering Institute

Formal Languages and Automata

Lesson 10: Normal forms and parsing

Marc Gaetano

Edition 2018

Testing membership and parsing

- Given a grammar

$$\mathrm{S} \rightarrow 0\mathrm{S}1 \mid 1\mathrm{S}0\mathrm{S}1 \mid \mathrm{T}$$
$$\mathrm{T} \rightarrow \mathrm{S} \mid \mathrm{e}$$

- How can we know if a string x is in its language?
- If so, can we reconstruct a parse tree for x?

First attempt

$$S \rightarrow 0S1 \mid 1S0S1 \mid T$$
$$T \rightarrow S \mid \varepsilon$$

$$x = 00111$$

- Maybe we can try all possible derivations:

$$
\begin{array}{lllll}
S \Rightarrow & 0S1 & \Rightarrow & 00S11 & \Rightarrow \\
 & & \Rightarrow & 01S0S11 & \Rightarrow \\
 & & \Rightarrow & 0T1 & \Rightarrow \\
\Rightarrow & 1S0S1 & \Rightarrow & 10S10S1 & \Rightarrow \\
 & \ldots & & & \\
\Rightarrow & T & \Rightarrow & S & \Rightarrow \\
 & & \Rightarrow & \varepsilon & \Rightarrow
\end{array}
$$

when do we stop?

Problems

$S \rightarrow 0S1 \mid 1S0S1 \mid T$

$T \rightarrow S \mid \varepsilon$

$x = 00111$

- How do we know when to stop?

$S \Rightarrow 0S1 \Rightarrow 00S11 \Rightarrow$

$\Rightarrow 01S0S11 \Rightarrow$

$\Rightarrow 0T1 \Rightarrow$

$\Rightarrow 1S0S1 \Rightarrow 10S10S1 \Rightarrow$

...

when do we stop?

Problems

$$S \rightarrow 0S1 \mid 1S0S1 \mid T$$
$$T \rightarrow S \mid \varepsilon$$

$$x = 01011$$

- Idea: Stop derivation when length exceeds $|x|$
- Not right because of ε-productions

$$S \underset{1}{\Rightarrow} 0S1 \underset{3}{\Rightarrow} 01S0S11 \underset{7}{\Rightarrow} 01S011 \underset{6}{\Rightarrow} 01011 \underset{5}{}$$

- We might want to eliminate ε-productions too

Problems

$$S \rightarrow 0S1 \mid 1S0S1 \mid T$$
$$T \rightarrow S \mid \varepsilon$$

$$x = 00111$$

- Loops among the variables ($S \rightarrow T \rightarrow S$) might make us go forever
- We might want to eliminate such loops

Unit productions

- A unit production is a production of the form

$$A_1 \rightarrow A_2$$

where A_1 and A_2 are both variables

- Example

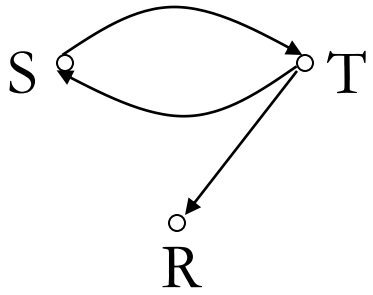
grammar:

$S \rightarrow 0S1 \mid 1S0S1 \mid T$

$T \rightarrow S \mid R \mid \varepsilon$

$R \rightarrow 0SR$

unit productions:



Removal of unit productions

- If there is a cycle of unit productions

$$A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k \rightarrow A_1$$

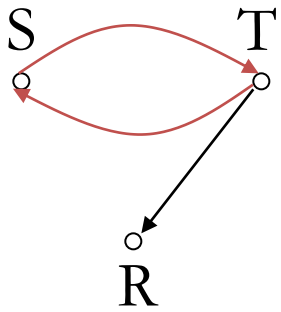
delete it and replace everything with A_1

- Example

$S \rightarrow 0S1 \mid 1S0S1 \mid$ ~~T~~

$T \rightarrow$ ~~S~~ $\mid R \mid \varepsilon$

$R \rightarrow 0SR$



$S \rightarrow 0S1 \mid 1S0S1$

$S \rightarrow R \mid \varepsilon$

$R \rightarrow 0SR$

T is replaced by S in the $\{S, T\}$ cycle

Removal of unit productions

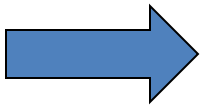
- For other unit productions, replace every chain

$$A_1 \rightarrow A_2 \rightarrow \ldots \rightarrow A_k \rightarrow \alpha$$

by productions $A_1 \rightarrow \alpha, \ldots, A_k \rightarrow \alpha$

- Example

$S \rightarrow 0S1 \mid 1S0S1 \mid R \mid \varepsilon$
$R \rightarrow 0SR$



$S \rightarrow 0S1 \mid 1S0S1 \mid 0SR \mid \varepsilon$
$R \rightarrow 0SR$

$S \rightarrow R \rightarrow 0SR$ is replaced by $S \rightarrow 0SR, R \rightarrow 0SR$

Removal of ε-productions

- A variable N is nullable if there is a derivation

$$\mathrm{N} \stackrel{*}{\Rightarrow} \varepsilon$$

- How to remove ε-productions (except from S)

① Find all nullable variables $\mathrm{N}_1, \ldots, \mathrm{N}_k$

② For $i = 1$ to k

For every production of the form $\mathrm{A} \rightarrow \alpha \mathrm{N}_i \beta$,

add another production $\mathrm{A} \rightarrow \alpha\beta$

If $\mathrm{N}_i \rightarrow \varepsilon$ is a production, remove it

③ If S is nullable, add the special production $\mathrm{S} \rightarrow \varepsilon$

Example

- Find the nullable variables

grammar	nullable variables
$S \rightarrow ACD$	B C D
$A \rightarrow a$	
$B \rightarrow \varepsilon$	
$C \rightarrow ED \mid \varepsilon$	
$D \rightarrow BC \mid b$	
$E \rightarrow b$	

① Find all nullable variables $N_1, ..., N_k$

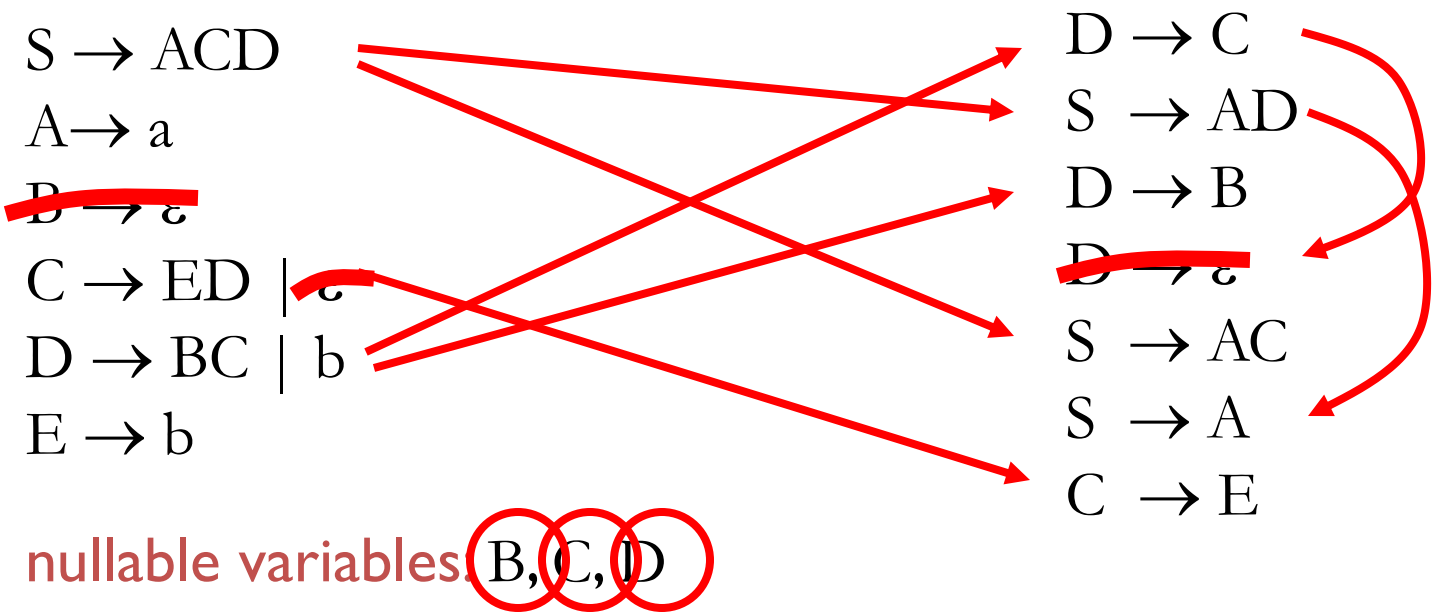
Finding nullable variables

- To find nullable variables, we work backwards
  - First, mark all variables A s.t. $\mathrm{A} \rightarrow \varepsilon$ as nullable
  - Then, as long as there are productions of the form

$$\mathrm{A} \rightarrow \mathrm{A}_1 \ldots \mathrm{A}_k$$

    where all of $\mathrm{A}_1, \ldots, \mathrm{A}_k$ are marked as nullable, mark A as nullable

Eliminating ε-productions



② For $i = 1$ to k

For every production of the form $A \rightarrow \alpha N_i \beta$,

add another production $A \rightarrow \alpha\beta$

If $N_i \rightarrow \varepsilon$ is a production, remove it

Recap

- After eliminating ε-productions and unit productions, we know that every derivation

  $S \overset{*}{\Rightarrow} a_1 \ldots a_k$ where $a_1, \ldots, a_k$ are terminals

  doesn't shrink in length and doesn't go into cycles

- Exception: $S \rightarrow \varepsilon$
  - We will not use this rule at all, except to check if $\varepsilon \in L$

- Note
  - ε-productions must be eliminated before unit productions

Example: testing membership

$S \to 0S1 \mid 1S0S1 \mid T$
$T \to S \mid \varepsilon$

eliminate

unit, ε-prod

$S \to \varepsilon \mid 01 \mid 101 \mid 0S1$
$\mid 10S1 \mid 1S01 \mid 1S0S1$

$x = 00111$

$S \Rightarrow 01, 101$

$0S1 \Rightarrow 0011, 01011$

$00S11 \Rightarrow$ only strings of length ≥ 6

strings of length ≥ 6

$10S1 \Rightarrow 10011$, strings of length ≥ 6

$1S01 \Rightarrow 10101$, strings of length ≥ 6

$1S0S1 \Rightarrow$ only strings of length ≥ 6

Algorithm 1 for testing membership

- We can now use the following algorithm to check if a string x is in the language of G

① Eliminate all ε-productions and unit productions

② If $\mathrm{x} = \varepsilon$ and $\mathrm{S} \rightarrow \varepsilon$, accept; else delete $\mathrm{S} \rightarrow \varepsilon$

③ Let $X := \mathrm{S}$

④ While some new production P can be applied to X

Apply P to X

If $X = x$, accept

If $|X| > |x|$, backtrack

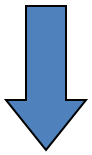
⑤ If no more productions can be applied to X, reject

Practical limitations of Algorithm I

- Previous algorithm can be very slow if x is long

  G = CFG of the java programming language

  x = code for a 200-line java program



  algorithm might take about 10^{200} steps!

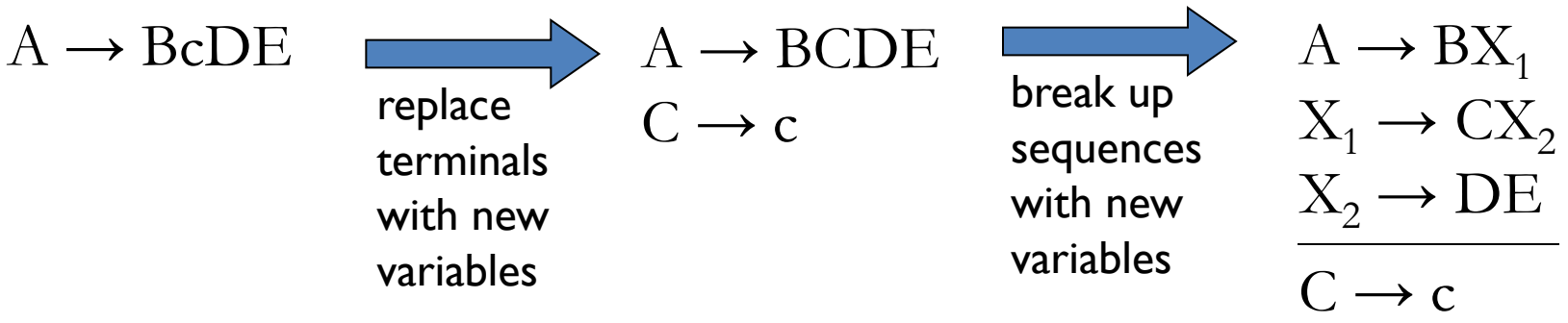
- There is a faster algorithm, but it requires that we do some more transformations on the grammar

Chomsky Normal Form

- A grammar is in Chomsky Normal Form if every production (except possibly $S \rightarrow \varepsilon$) is of the type

$$A \rightarrow BC \quad \text{or} \quad A \rightarrow a$$

- Conversion to Chomsky Normal Form is easy:



Exercise

- Convert this CFG into Chomsky Normal Form:

$$S \rightarrow \varepsilon \mid ADDA$$
$$A \rightarrow a$$
$$C \rightarrow c$$
$$D \rightarrow bCb$$

Algorithm 2 for testing membership

$S \to AB \mid BC$
$A \to BA \mid a$
$B \to CC \mid b$
$C \to AB \mid a$

$x = baaba$

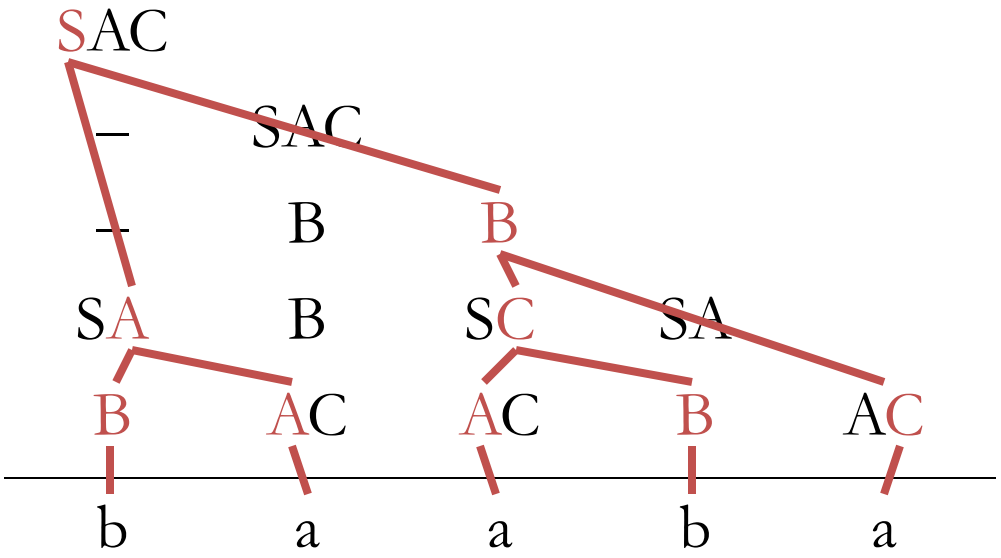
SAC				
–	SAC			
–	B	B		
SA	B	SC	SA	
B	AC	AC	B	AC
b	a	a	b	a

Idea: We generate each substring of x bottom up

Parse tree reconstruction

$S \to AB \mid BC$
$A \to BA \mid a$
$B \to CC \mid b$
$C \to AB \mid a$

$x = \text{baaba}$



Tracing back the derivations, we obtain the parse tree