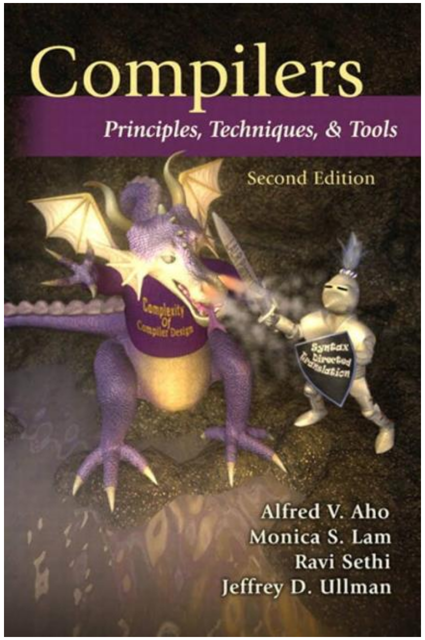
Compiler

Lec 04

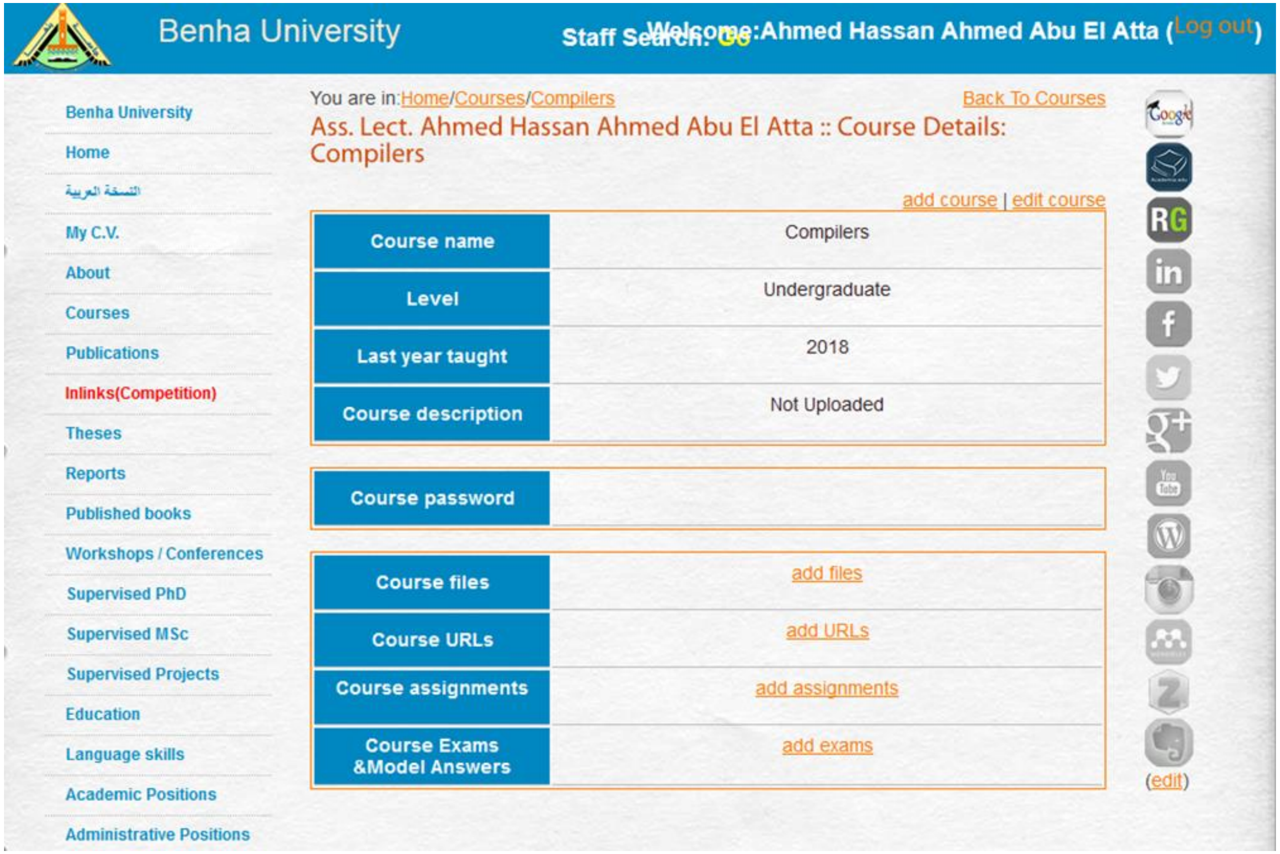
Book

Compilers: Principles, Techniques, and Tools is a computer science textbook by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman about compiler construction.



PowerPoint

http://www.bu.edu.eg/staff/ahmedaboalatah14-courses/14779



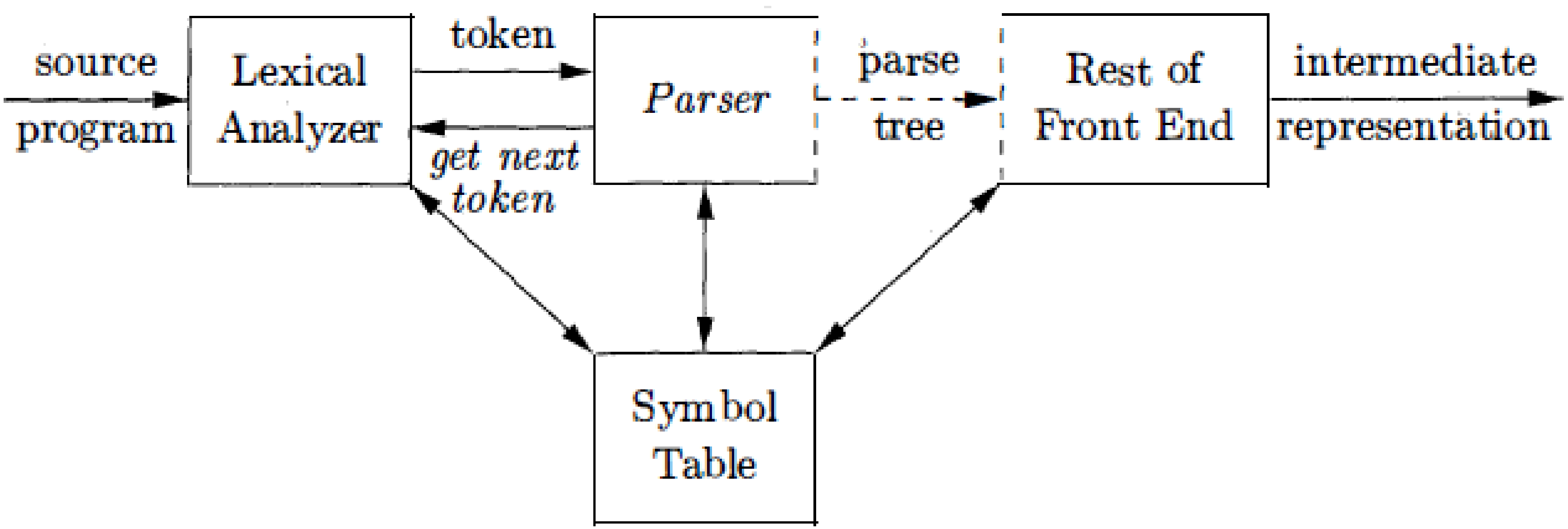
Syntax Analysis

PART I

The role of parser

Construct a parse tree (tree need not be constructed explicitly; the parser and the rest of the front end could well be implemented by a single module).

Report and recover from errors.



Types of parsers for grammars

Universal

- Can parse any grammar (Cocke-Younger-Kasami algorithm and Earley's algorithm).
- Too inefficient to use in production compilers (Time-Cost).

Top-down

- Build parse trees from the root to the leaves.
- Scan input from left to right.

Bottom-up

- start from the leaves and work their way up to the root.
- Scan input from left to right.

Grammars

G=(T, N, P, S)

A set of terminals: basic symbols from which sentences are formed.

A set of nonterminals: syntactic variables denoting sets of strings.

A set of productions: rules specifying how the terminals and nonterminals can be combined to form sentences. ($A \rightarrow \beta$)

The start symbol: a distinguished nonterminal denoting the language.

Example

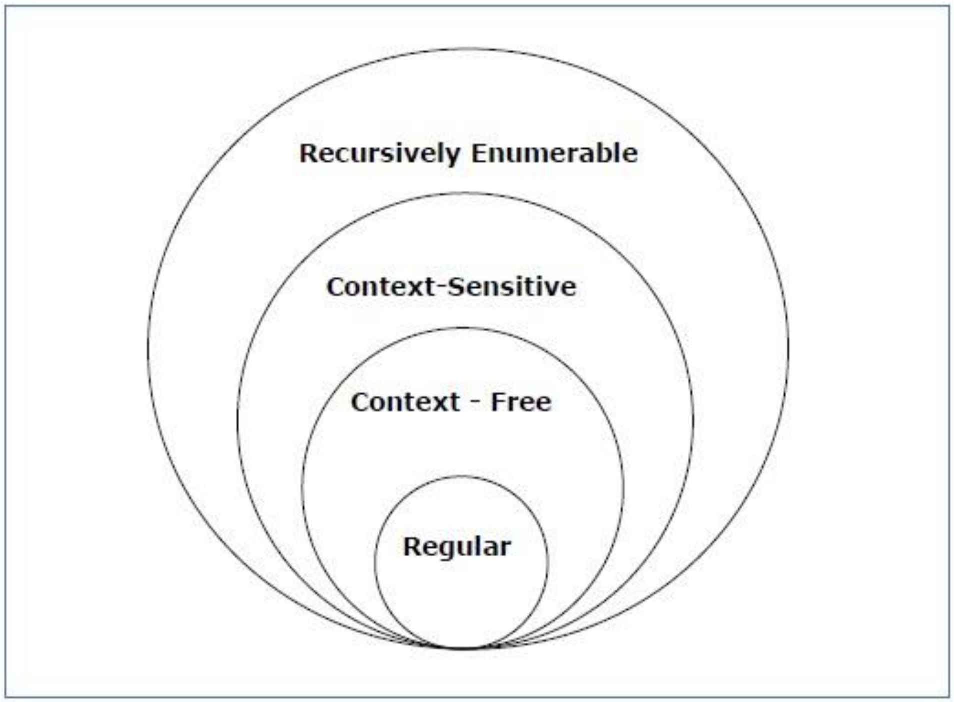
$$
\begin{array}{lcl}
E & \rightarrow & E + T \mid T \\
T & \rightarrow & T * F \mid F \\
F & \rightarrow & (E) \mid \mathbf{id}
\end{array}
$$

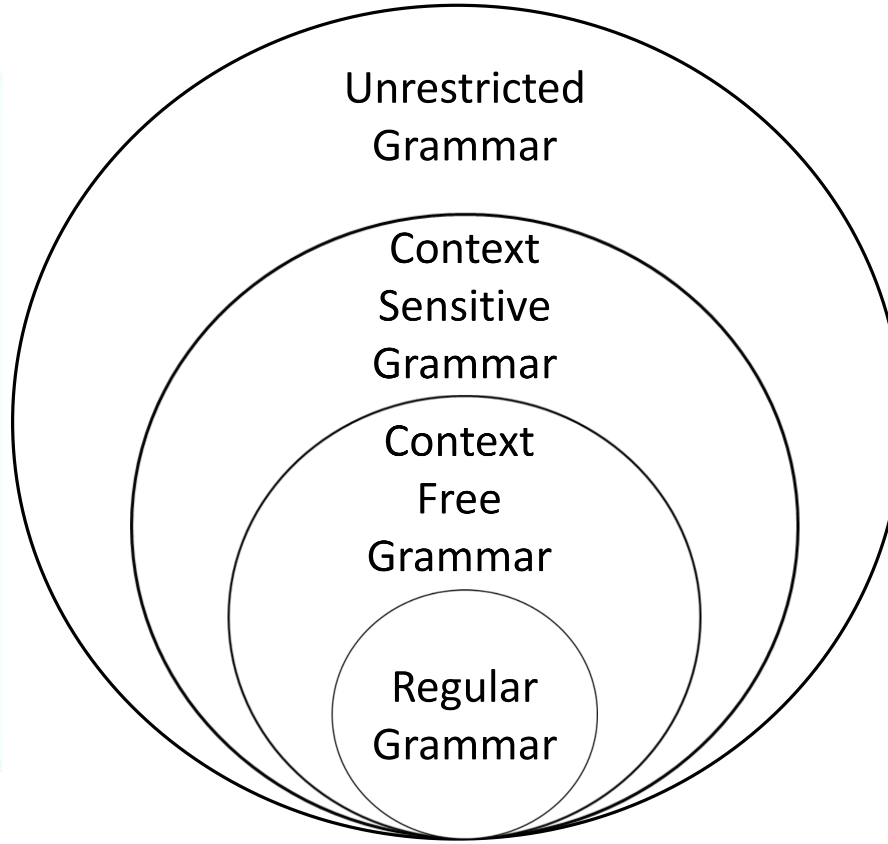
T = { +, *, (,), id}

N = $\{E, T, F\}$

Start symbol "E"

Chomsky hierarchy (classification of grammars)





Regular grammar

A grammar is said to be regular if it is

right-linear, where each production in P has the form,

$$A \rightarrow wB \text{ or } A \rightarrow w$$

A, B ∈ N and and ***w*** ∈ T^*

Example

$A \rightarrow \varepsilon$

$A \rightarrow a \mid aB$

$B \rightarrow b$

Context-free grammar

A grammar is said to be context-free if each production in ***P*** is of the form,

$$\boldsymbol{A \to \alpha}$$

$\boldsymbol{A} \in N$ and $\boldsymbol{\alpha} \in (N \cup T)^*$

Example

$S \rightarrow Aa$

$A \rightarrow B \mid aA$

$B \rightarrow aBc \mid \varepsilon$

Context sensitive grammar

A grammar is said to be context sensitive if each production in ***P*** is of the form,

$$\alpha \rightarrow \beta$$

$|\alpha| \leq |\beta|$ and $\alpha, \beta \in (N \cup T)^*$

And S → ε is allowed if S does not appear on the right side of any rule.

Example

AB → AbBc

A → bcA

B → b

Unrestricted grammar

A grammar is said to be unrestricted if each production in ***P*** is of the form,

$$\alpha \rightarrow \beta$$

$\alpha \neq \varepsilon$ and $\alpha, \beta \in (N \cup T)^*$

Example

$S \rightarrow ACaB$

$Bc \rightarrow acB$

$CB \rightarrow DB$

$aD \rightarrow Db$

Derivations

$\alpha A \beta \quad \Rightarrow \quad \alpha\gamma\beta$ if $A \longrightarrow \gamma$

$\alpha \Rightarrow^{*} \alpha$

$\alpha \Rightarrow^{*} \beta$ and $\beta \Rightarrow^{*} \gamma$ then $\alpha \Rightarrow^{*} \gamma$

Derivations

E -> E + E | E * E | -E | (E) | **id**

A ***leftmost*** derivation always chooses the ***leftmost nonterminal*** to rewrite

$E \Rightarrow E \Rightarrow (E) \Rightarrow (E + E) \Rightarrow (\text{id} + E) \Rightarrow (\text{id} + \text{id})$

A ***rightmost*** derivation always chooses the ***rightmost nonterminal*** to rewrite

$E \Rightarrow E \Rightarrow (E) \Rightarrow (E + E) \Rightarrow (E + \text{id}) \Rightarrow (\text{id} + \text{id})$

Parse trees

BASIS : The tree for al = A is a single node labeled A.

INDUCTION:

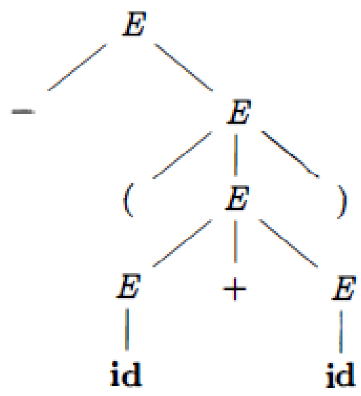
- Suppose α_i is derived from α_{i-1} by replacing X_j , a nonterminal, by β = $Y_1 Y_2 \ldots.. Y_m$.
- To model this step of the derivation, find the j[th] leaf from the left in the current parse tree. This leaf is labeled X_j . Give this leaf "m" children, labeled $Y_1, Y_2, \ldots.. , Y_m$, from the left .

-(id+id)

E ⇒ -E ⇒ -(E) ⇒

-(E+E) ⇒ -(**id**+E)

⇒ -(**id**+**id**)



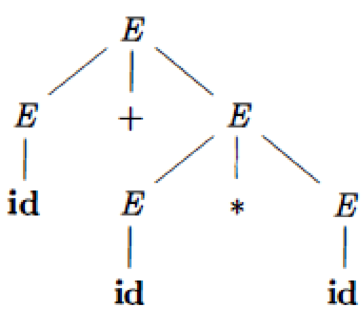
Ambiguity

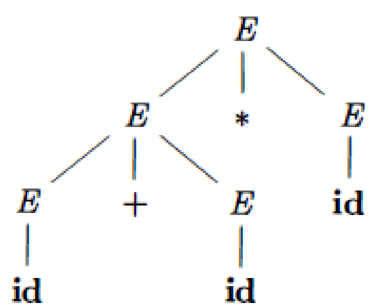
For some strings there exist more than one parse tree

Or more than one leftmost derivation

Or more than one rightmost derivation

Example: id+id*id



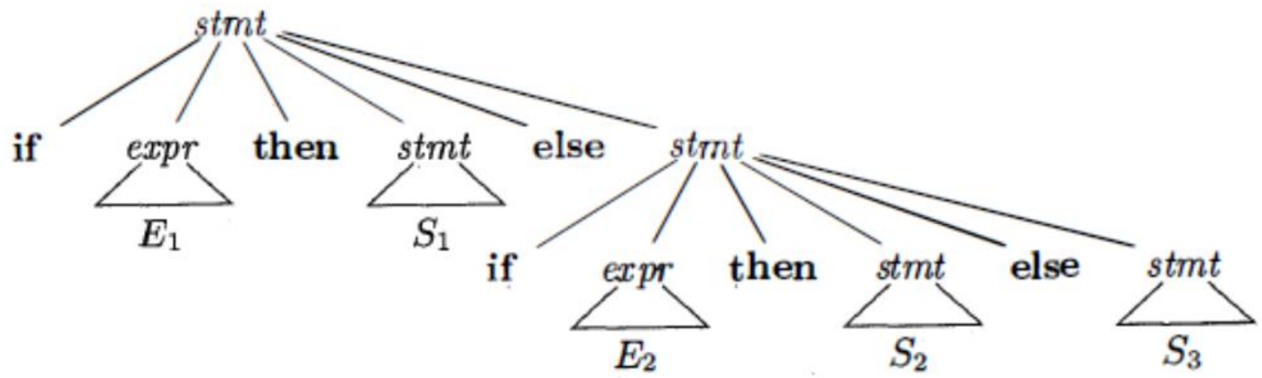


$$
\begin{aligned}
E &\Rightarrow E + E \\
&\Rightarrow \mathbf{id} + E \\
&\Rightarrow \mathbf{id} + E * E \\
&\Rightarrow \mathbf{id} + \mathbf{id} * E \\
&\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id}
\end{aligned}
\qquad
\begin{aligned}
E &\Rightarrow E * E \\
&\Rightarrow E + E * E \\
&\Rightarrow \mathbf{id} + E * E \\
&\Rightarrow \mathbf{id} + \mathbf{id} * E \\
&\Rightarrow \mathbf{id} + \mathbf{id} * \mathbf{id}
\end{aligned}
$$

Ambiguity

$$
\begin{array}{lcl}
stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } stmt \textbf{ else } stmt \\
 & | & \textbf{other}
\end{array}
$$

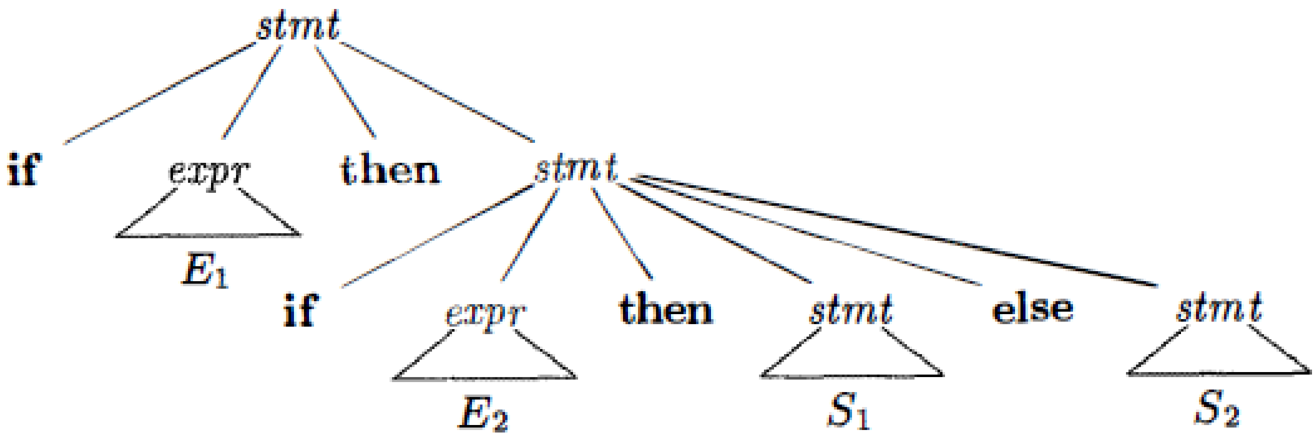
$$\textbf{if } E_1 \textbf{ then } S_1 \textbf{ else if } E_2 \textbf{ then } S_2 \textbf{ else } S_3$$



Ambiguity

if E_1 **then** **if** E_2 **then** S_1 **else** S_2

if E_1 **then** **if** E_2 **then** S_1 **else** S_2



Deal with Ambiguity

Rewrite to equivalent unambiguous grammar

- possible, but results in more complex grammar (several similar rules).

Use the ambiguous grammar

- use "rule priority", the parser can select the correct rule.
- works for the dangling else problem, but not for ambiguous grammars in general.
- not all parser generators support it well.

Change the language

- e.g., add a keyword "fi" that closes the "if"-statement.
- restrict the "then" part to be a block: "{ ... }".
- only an option if you are designing the language yourself.

Eliminating Ambiguity

$$
\begin{array}{lcl}
stmt & \rightarrow & matched\_stmt \\
 & | & open\_stmt \\
matched\_stmt & \rightarrow & \textbf{if } expr \textbf{ then } \boxed{matched\_stmt} \textbf{ else } matched\_stmt \\
 & | & \textbf{other} \\
open\_stmt & \rightarrow & \textbf{if } expr \textbf{ then } stmt \\
 & | & \textbf{if } expr \textbf{ then } \boxed{matched\_stmt} \textbf{ else } open\_stmt
\end{array}
$$

Eliminating Ambiguity

E -> E + E | E * E | -E | (E) | id

Restrict certain subtrees by introducing new nonterminals.

$$
\begin{aligned}
E &\rightarrow E + T \mid E - T \mid T \\
T &\rightarrow T * F \mid T / F \mid F \\
F &\rightarrow (E) \mid \textbf{id}
\end{aligned}
$$

Examples

Write grammar for each of the following languages:

1. L = {a, b}.
2. Set of all strings over {a, b}.
3. Strings that consist of a sequence of a's followed by a sequence of b's
4. Strings that consist of a sequence of a's, where the number of a's is even.
5. The set of strings that begin with ab and end with ba, over alphabet {a, b}.
6. the language $\{a^n b^n \mid n \geq 0 \text{ and } n \text{ is even}\}$.

Examples

1. The set of strings of parentheses ( ).
2. L = {w|w starts and ends with the same symbol}.
3. The complement language $\{a^n b^n \mid n \geq 0\}$.
4. The palindrome.
5. Is the following grammar ambiguous :

$$S \rightarrow aSbS \mid bSaS \mid \varepsilon$$

6. Is the following grammars ambiguous

$$S \rightarrow a \mid aAb \mid abSb$$

$$A \rightarrow aAAb \mid bS$$