中山大學

Principles of Compiler Construction

Lecture 5 Syntax Analysis (I)

Lecturer: CHANG HUIYOU

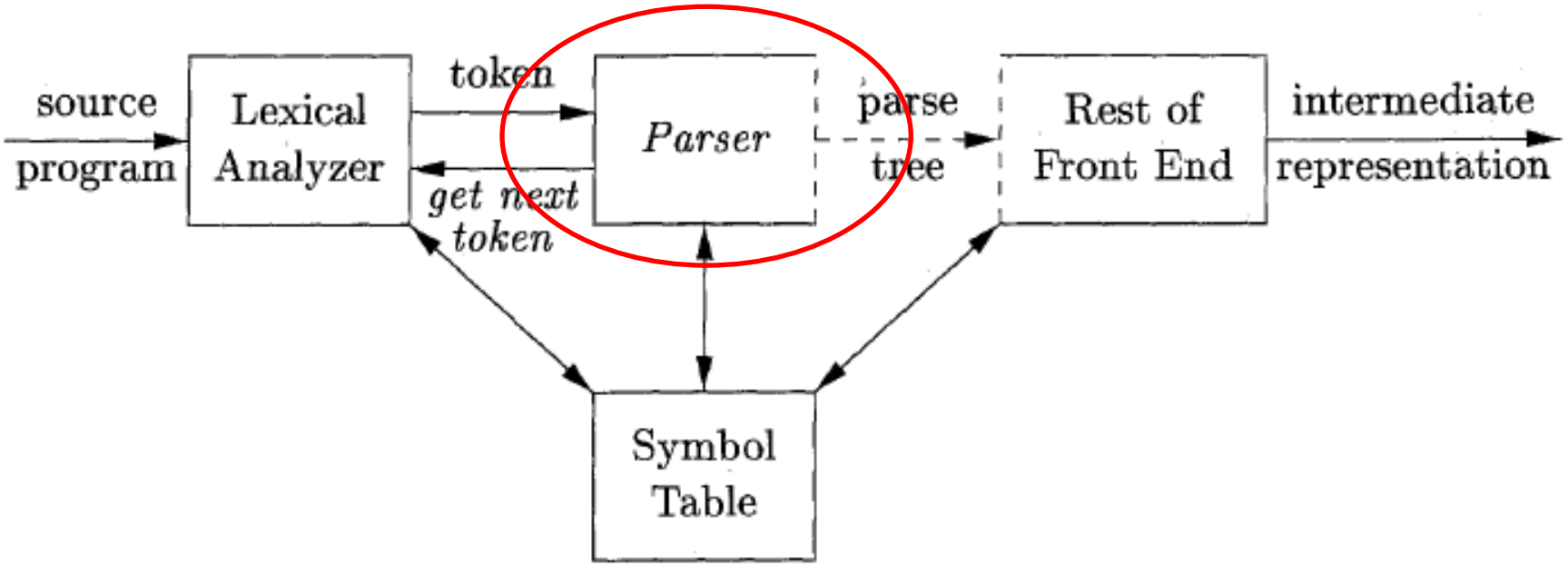
Note that most of these slides were created by:

Prof. Wen-jun LI (School of Software)

中山大學

中山大學

务任的析分法语



- The pipe between a parser and a scanner
  - Method invocation (procedure call)
  - Logical vs. physical
- Context-free grammars
  - Parse tree, derivation, and reduction
  - Ambiguity

务任的析分法语

语法分析要解决的问题包括:

- **从词法分析中获得的每个属性字（token）在语句中，或在整个程序中扮演什么角色**

  例如：同样一个数字，在程序中可能是一个加数，可能是数组的下标，可能是循环的次数，可能是某个函数的参数等

- **检查语句或程序是否符合程序语言的语法**

确明步一进的题问

要解决这些问题，我们必须先明确：

- 一个程序包括哪些语法成分？
- 这些语法成分以什么方式构成程序？

因为我们要用计算机来分析，所以需要严格精确的定义

借助形式化的方法

中山大學

考思

正则表达式和自动机这一类形式化工具显然是不够的，因为它们甚至无法定义简单的语言 $L = \{a^n b^n \mid n \geq 1\}$ ，更不用说诸如括号匹配这一类更复杂，但在程序中却经常出现的语言.

中山大學

考思

比如对于含else的if语句，其构成方式如下：

if (*expr*) *stmt* else *stmt*

if语句是语句的一种

stmt*→ if (*expr*) *stmt* else *stmt

if, else, (和)为终端符号

从词法分析器获得

***stmt, expr*为非终端符号**

还需要定义

Parsing strategies

Top-down parsing

How to choose a unique production in multiple candidates ?

Bottom-up parsing

How to find the handle in a sentential form ?

中山大學

A motivating example

type	→	*simple*
	|	^ **id**
	|	**array** [*simple*] **of** *type*
simple	→	**integer**
	|	**char**
	|	**num dotdot num**

```
array [1..10] of integer
```

Top-Down Parsing: Motivation

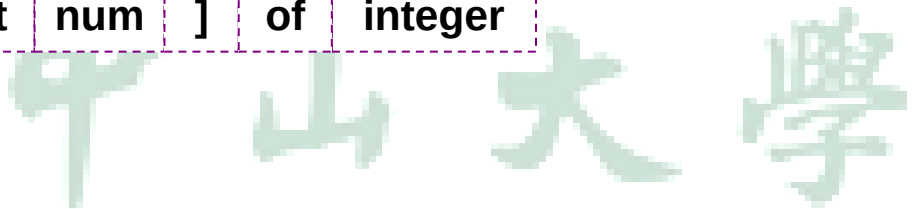
中山大學

Parsing Process (Initial)

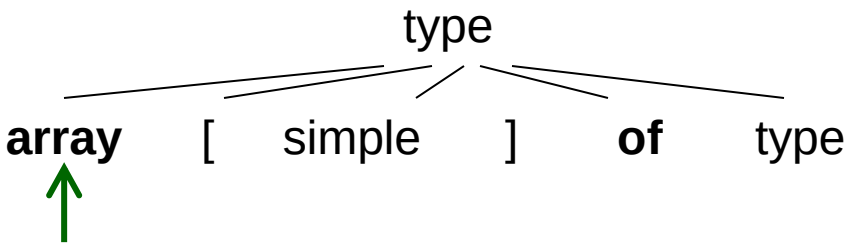
*Derive with "type → **array [** simple **] of** type"*

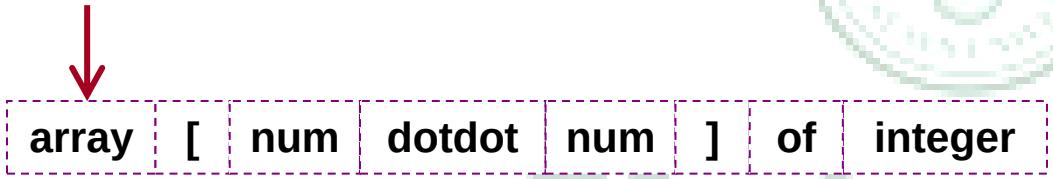
type

array	[	num	dotdot	num	]	of	integer

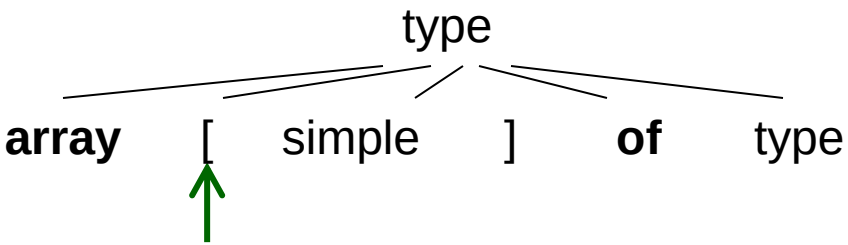


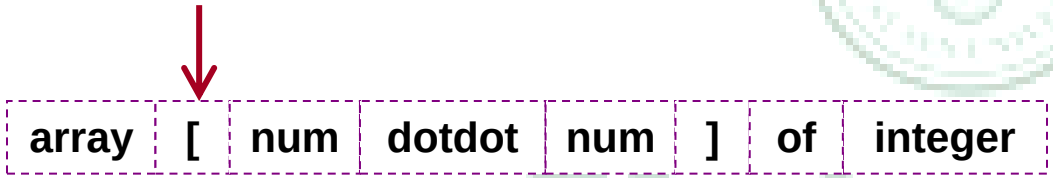
Match !



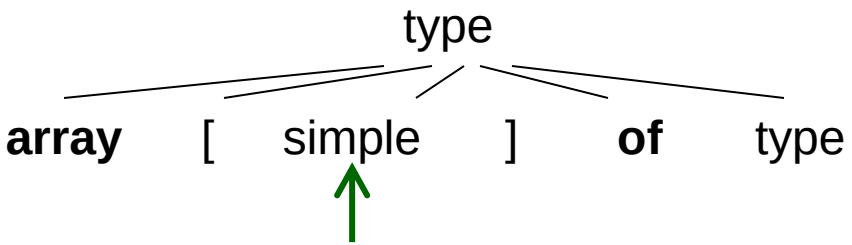


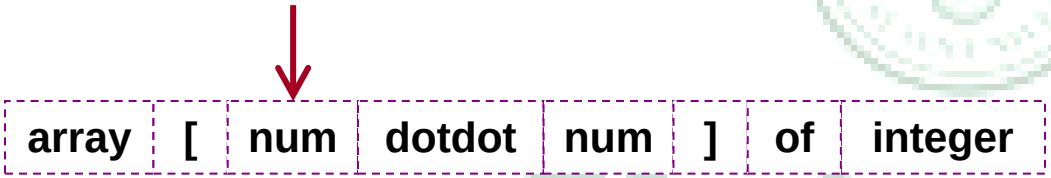
Match !



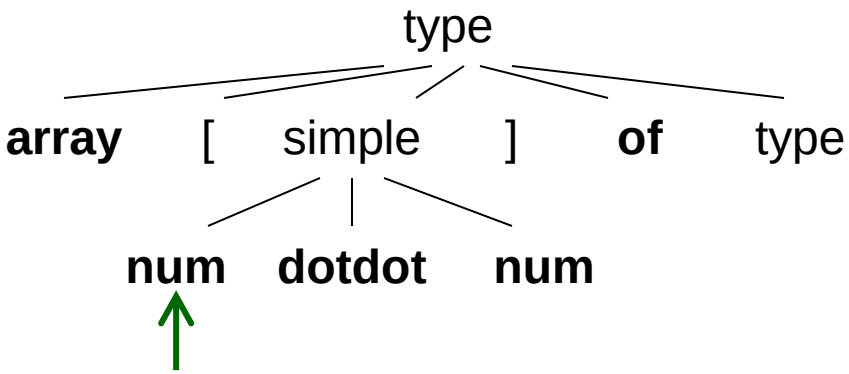


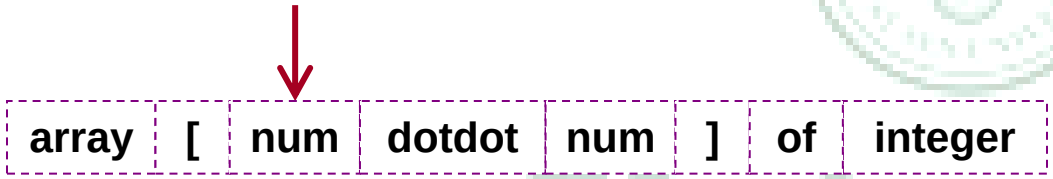
Derive with "simple → ***num dotdot num"***



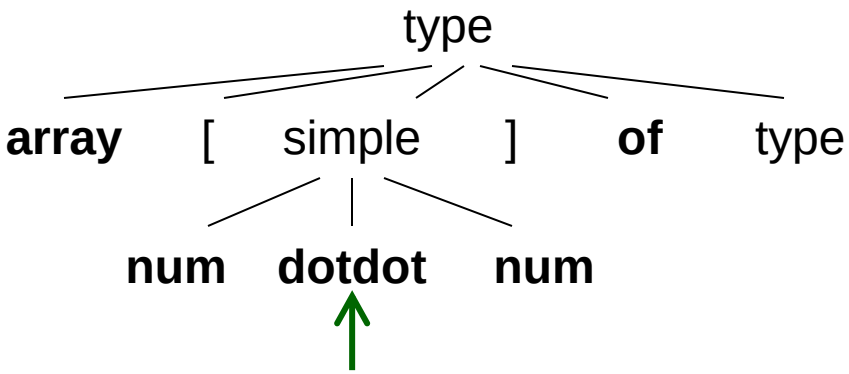


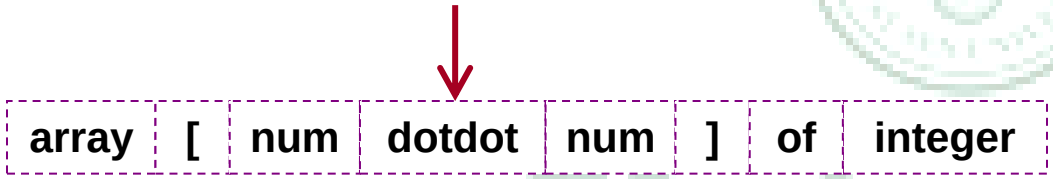
Match !


type
array
[
simple
]
of
type
num
dotdot
num

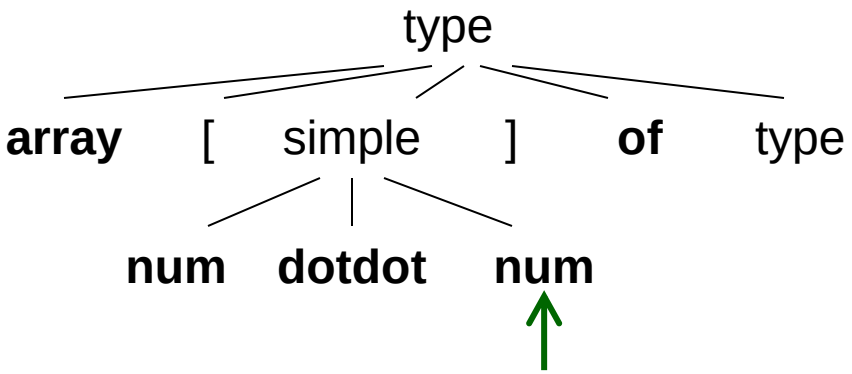

array
[
num
dotdot
num
]
of
integer

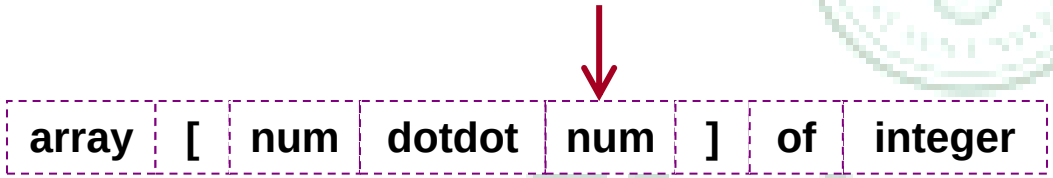
Match !

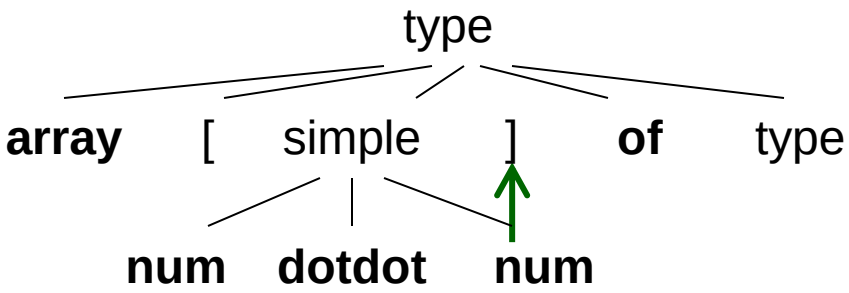

type
array
[
simple
]
of
type
num
dotdot
num


array
[
num
dotdot
num
]
of
integer

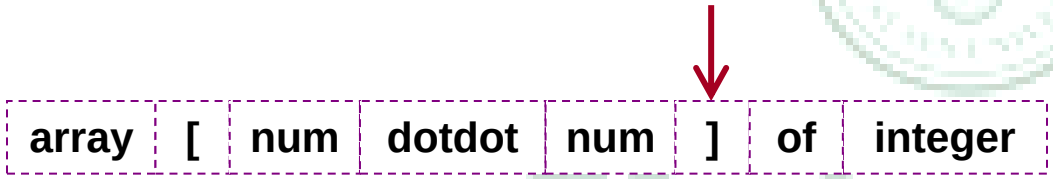
Match !


type
array
[
simple
]
of
type
num
dotdot
num

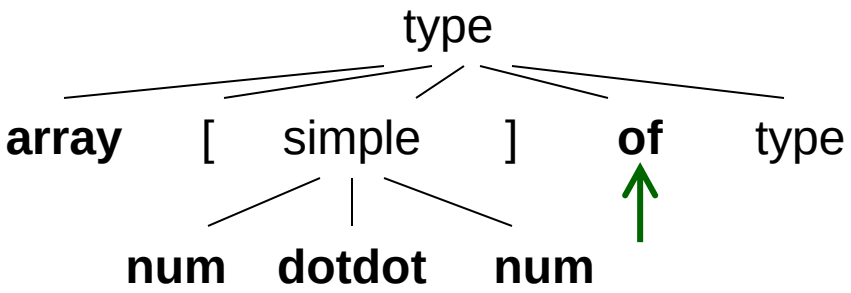

array
[
num
dotdot
num
]
of
integer

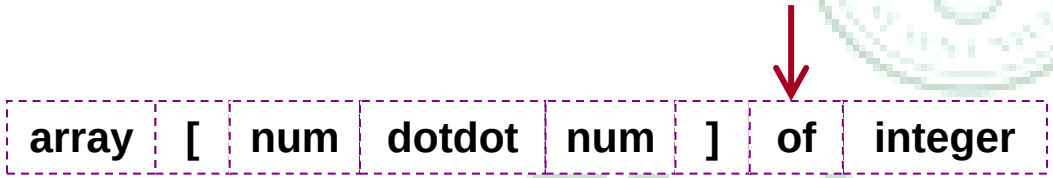

type
array
[
simple
]
of
type
num
dotdot
num

Match !

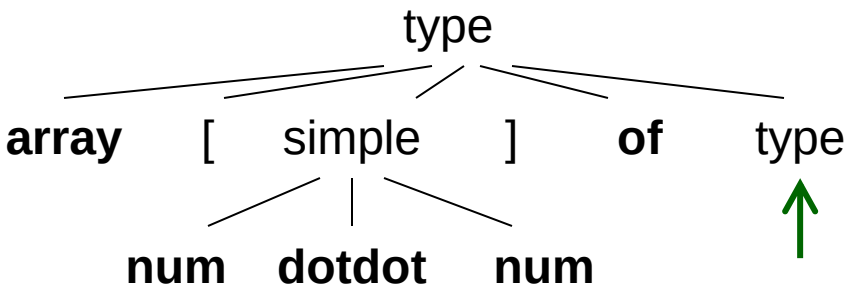

array
[
num
dotdot
num
]
of
integer

Match !


type
array
[
simple
]
of
type
num
dotdot
num

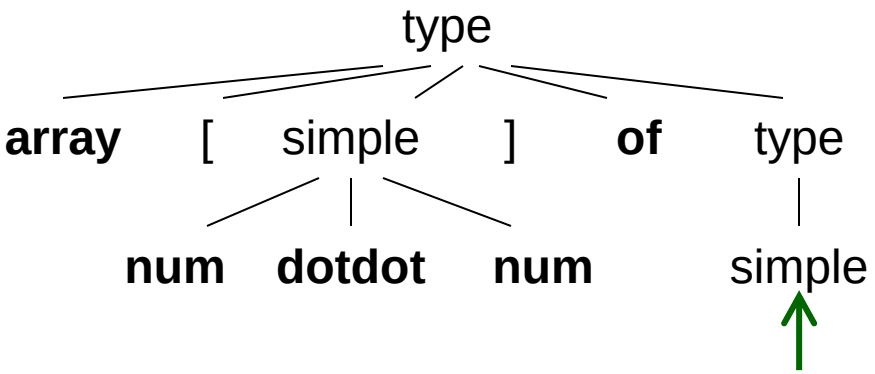

array
[
num
dotdot
num
]
of
integer

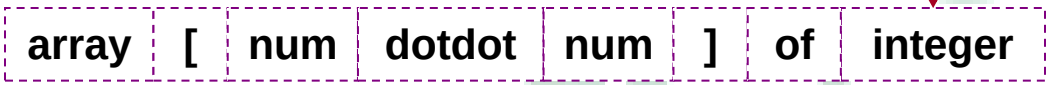
Derive with "type → simple"



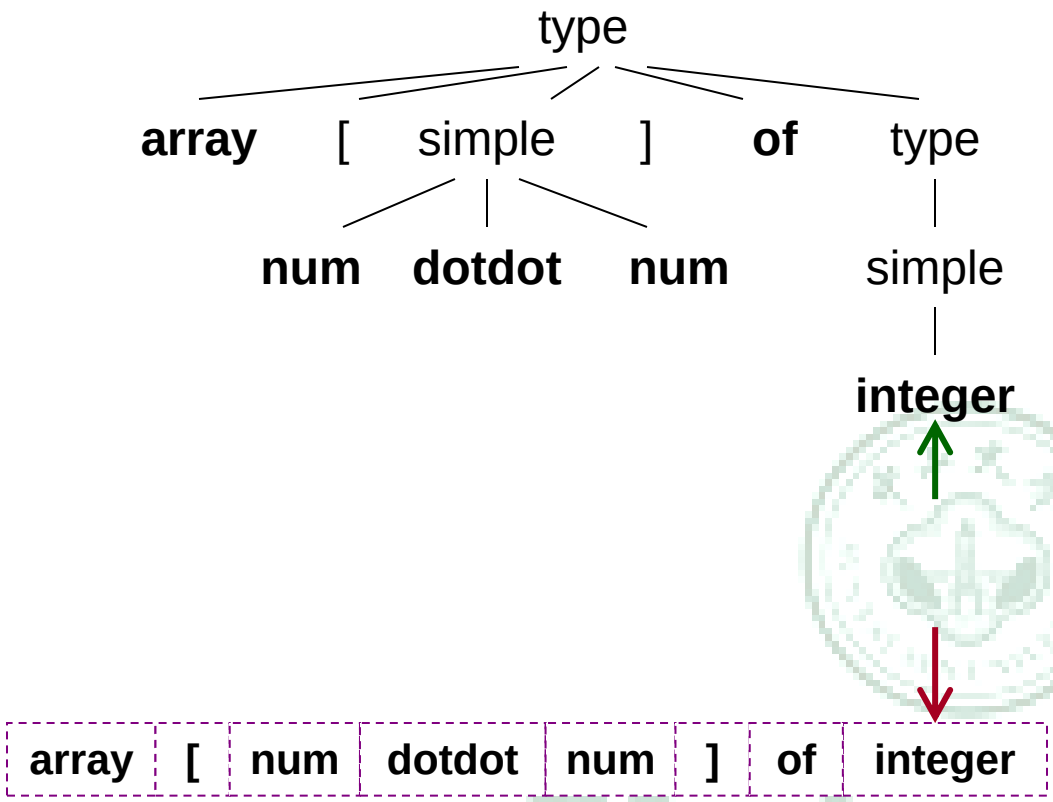
array	[	num	dotdot	num	]	of	integer

*Derive with "simple → **integer**"*





Match and Accept !



上下文无关文法：形式化的语法

一个上下文无关文法（CFG）包括四部分:

1. 终端符号（terminal）的集合T
2. 非终端符号（nonterminal）的集合N
3. 唯一的开始符号S（$S \in N$）
4. 若干以下形式的产生式（production）：

$$X \rightarrow Y_1 Y_2 \ldots Y_n$$

其中 $X \in N$ 且 $Y_i \in T \cup N \cup \{\varepsilon\}$

中山大學

定规的便方了为些一

第一个产生式左端的非终端符号为开始符号

多个形如$A \to \alpha_1, A \to \alpha_2, ..., A \to \alpha_k$的产生式可简写为

$$A \to \alpha_1 \mid \alpha_2 \mid ... \mid \alpha_k$$

中山大學

子例

$$E \to E+E \mid E*E \mid -E \mid (E) \mid \text{id}$$

显然对于上面的文法，id, (id + id), -id*id都是符合文法定义的，而(id, id*+id都是不符合的.

问题：如何给出“符合文法”的形式化定义？

Useful in Practice

Useful in Theory

Class	Grammar	Restriction	Recognizer
3	Regular	$A \to aB$ or $A \to a$, where $A, B \in N \wedge a \in \Sigma \bigcup \{\varepsilon\}$. $A \to \varepsilon$ permitted if A is the start symbol and does not appear on the right of any production.	Finite-State Automaton (**FSA**)
2	Context-Free	$A \to \alpha$, where $A \in N \wedge \alpha \in (\Sigma \bigcup N)^*$.	Push-Down Automaton (**PDA**)
1	Context-Sensitive	$\alpha \to \beta$, where $\alpha, \beta \in (\Sigma \bigcup N)^* \wedge \alpha \neq \varepsilon \wedge \lvert\alpha\rvert \leq \lvert\beta\rvert$. β can't be ε, unless α is the start symbol and does not appear on the right of any production.	Linear-Bounded Automaton (**LBA**)
0	Unrestricted	$\alpha \to \beta$, where $\alpha, \beta \in (\Sigma \bigcup N)^* \wedge \alpha \neq \varepsilon$.	Turing Machine (**TM**)

中山大學

What languages it can generate ?

$L_0 = \{a^n b^n \mid n \geq 1\}$

Abstraction of some problem in practice

What languages it can not generate ?

$L_1 = \{\omega\ c\ \omega \mid \omega \in (a \mid b)^* \wedge a, b, c \in \Sigma\}$

Abstraction of some problem in practice

How to solve the problem ?

$L_2 = \{a^n b^m c^n d^m \mid n \geq 1 \wedge m \geq 1\}$

Abstraction of some problem in practice

How to solve the problem ?

中山大學

导推

推导（derivation）：从开始符号开始，每一步推导就是用一个产生式的右方取代左端的非终端符号

例如：

$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(\mathrm{id}+\mathrm{E}) \Rightarrow -(\mathrm{id}+\mathrm{id})$$

我们将由开始符号推导出来的只含有终端符号的串称为这个文法的句子，一个文法G定义的语言就是这个文法所有句子的集合，记为L(G). 判断一个终端符号串x是否符合G的语法就是判断x是否属于L(G).

中山大學

论讨

为什么CFG定义语言的能力很可能比正则表达式强?

中山大學

习练

用上下文无关文法定义下列语言:

1) $L=\{0^n1^n \mid n\geq 1\}$

2) 只含有0和1的回文串

3)只含有(和)的匹配括号串

中山大學

导推右最和导推左最

最左推导（leftmost derivations）：每步推导都替换最左边的非终端符号

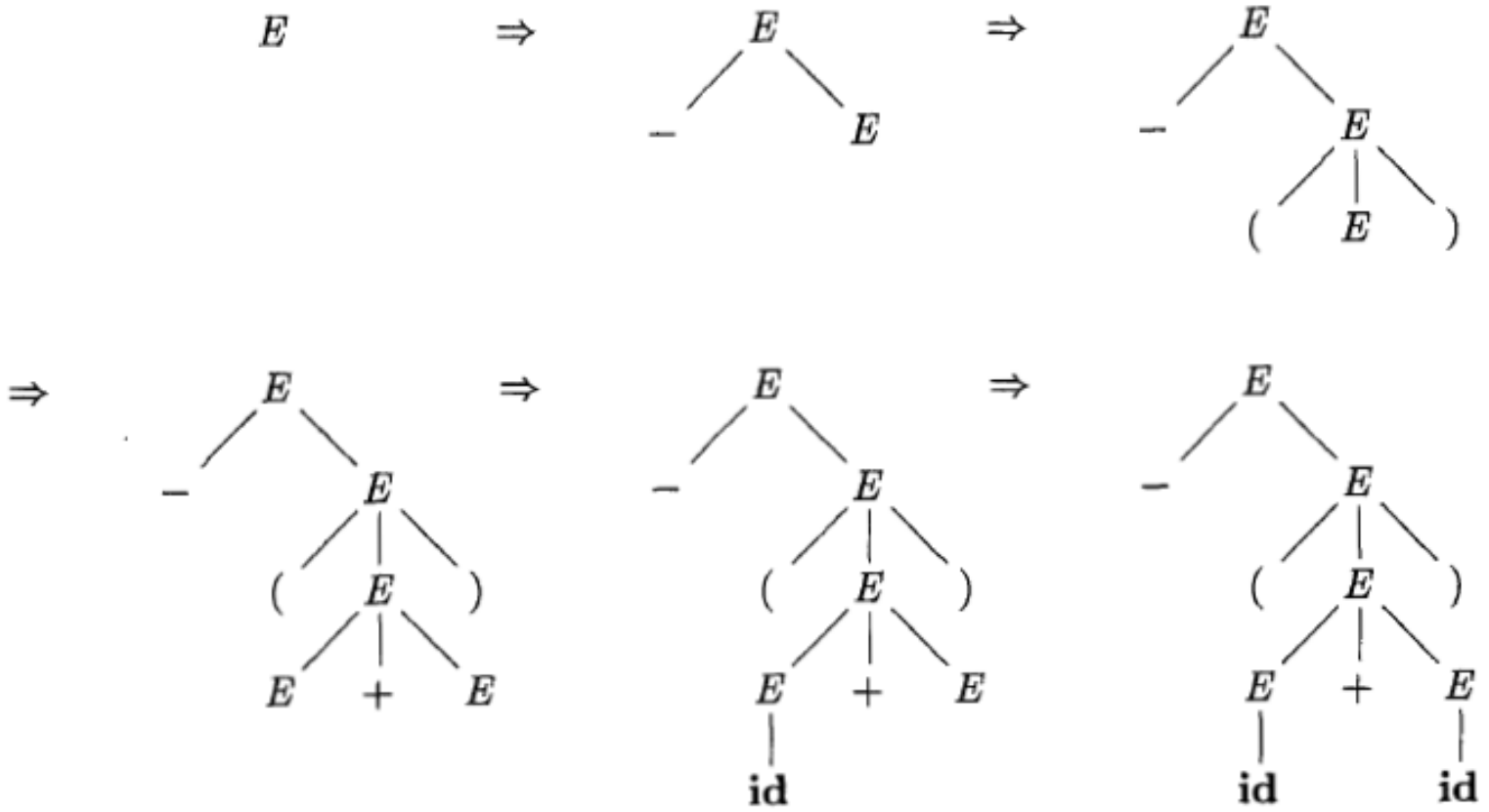
- 例如：

$$E \underset{lm}{\Rightarrow} -E \underset{lm}{\Rightarrow} -(E) \underset{lm}{\Rightarrow} -(E+E) \underset{lm}{\Rightarrow} -(\mathbf{id}+E) \underset{lm}{\Rightarrow} -(\mathbf{id}+\mathbf{id})$$

- 最右推导（rightmost derivations）：每步推导都替换最右边的非终端符号
  - 例如：

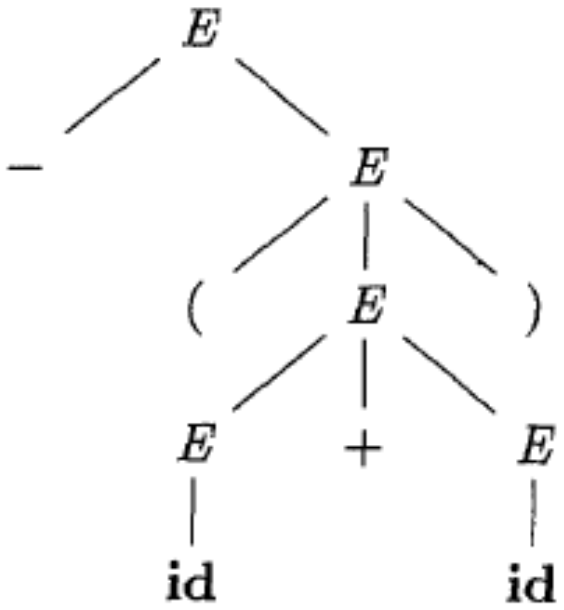
$$E \Rightarrow -E \Rightarrow -(E) \Rightarrow -(E+E) \Rightarrow -(E+\mathbf{id}) \Rightarrow -(\mathbf{id}+\mathbf{id})$$

分析树 (Parse Tree)



中山大學

分析树 (Parse Tree)



上述的最左推导和最右推导都对应这一分析树

语法分析的过程实际上就是构造分析树的过程

问题：一个句子可能有多个推导，那一个句子是否可能有多个分析树？

性义二的法文

文法的二义性：如果对于一个文法，存在一个句子，对这个句子可以构造两棵不同的分析树，那么我们称这个文法为二义的.

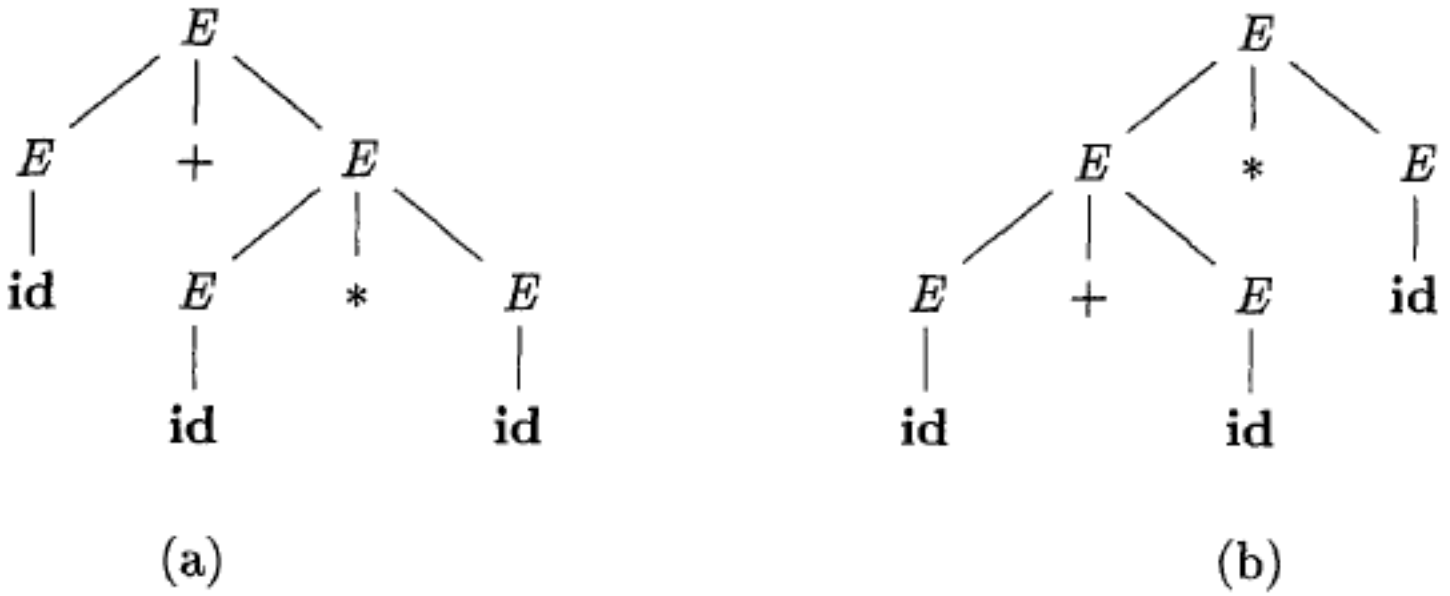
例：对于文法 $E \to E+E \mid E*E \mid -E \mid (E) \mid \mathrm{id}$

存在句子id+id*id，有两种推导

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

性义二的法文

这两种推导对应两棵不同的分析树:



因此这个文法是二义的.

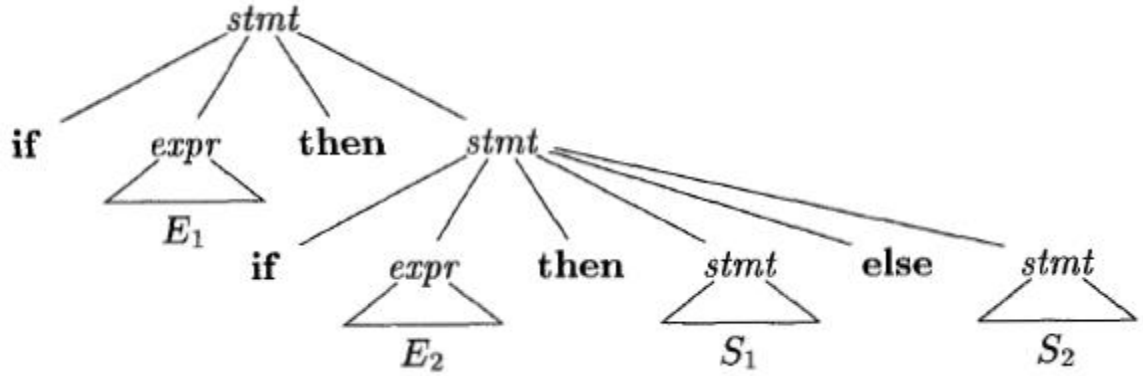
性义二的法文

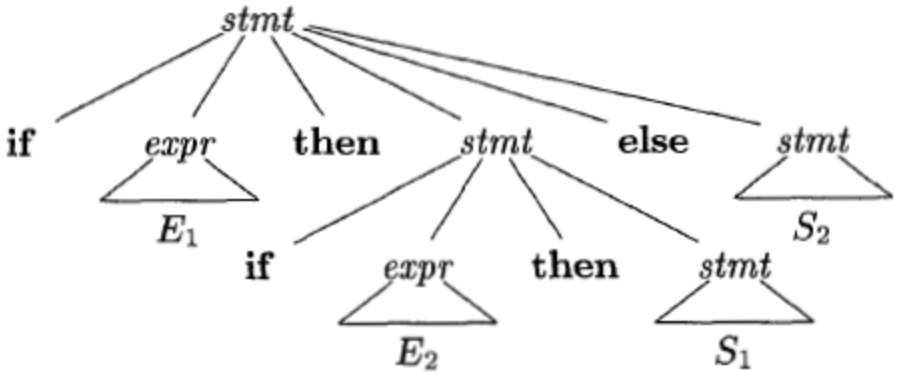
文法 $stmt \rightarrow$ **if** $expr$ **then** $stmt$

$\mid$ **if** $expr$ **then** $stmt$ **else** $stmt$

$\mid$ **other**

对于句子 **if** E_1 **then if** E_2 **then** S_1 **else** S_2

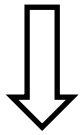




中山大學

除消的性义二法文

$$E \to E+E \mid E*E \mid (E) \mid \text{id}$$

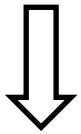


$$\begin{array}{lcl} E & \to & E + T \mid T \\ T & \to & T * F \mid F \\ F & \to & (E) \mid \mathbf{id} \end{array}$$

中山大學

除消的性义二法文

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



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

中山大學

除消的性义二法文

坏消息：有些文法的二义性是无法消除的

例如： $L=\{a^n b^n c^m d^m \mid n\geq 1, m\geq 1\}\cup\{a^n b^m c^m d^n \mid n\geq 1, m\geq 1\}$

属于上下文无关语言，我们可以找到一些CFG来描述*L*，但任何描述*L*的CFG都是二义的. （证明超出课程范围）

定判的性义二法文

坏消息：无法写出一个计算机程序，对于输入的上下文无关文法，判断其是否是二义的.（证明超出课程范围）

正则表达式 vs 上下文无关文法

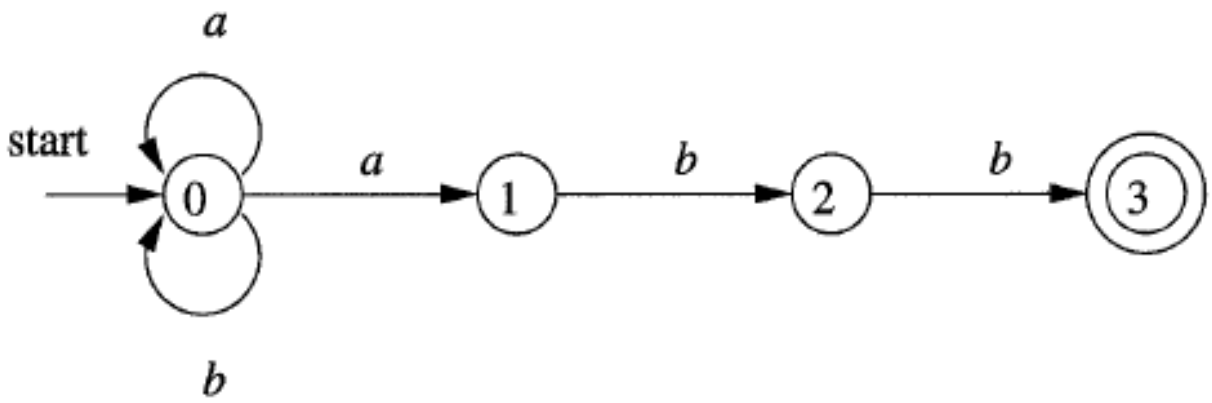
正则表达式无法定义语言 $L = \{a^n b^n \mid n \geq 1\}$

而上下文无关文法可以，是否说明上下文无关文法定义（或描述）语言的能力强于正则表达式?

思考：是否存在可以用正则表达式定义的语言，不能用上下文无关文法定义?

From NFA to CFG

1. For each state i of the NFA, create a nonterminal A_i.
2. If state i has a transition to state j on input a, add the production $A_i \to aA_j$. If state i goes to state j on input ϵ, add the production $A_i \to A_j$.
3. If i is an accepting state, add $A_i \to \epsilon$.
4. If i is the start state, make A_i be the start symbol of the grammar.



法文关无文下上为转 NFA

$$
\begin{array}{lcl}
A_0 & \rightarrow & aA_0 \mid bA_0 \mid aA_1 \\
A_1 & \rightarrow & bA_2 \\
A_2 & \rightarrow & bA_3 \\
A_3 & \rightarrow & \epsilon
\end{array}
$$

如果每个产生式都属于下列形式之一

$A \rightarrow aB$　　$A \rightarrow a$　　$A \rightarrow \varepsilon$

这样的文法称为右线性文法.

问题：是否有右线性文法能定义的语言不能用正则表达式或有限自动机定义？

中山大學

论结

正则表达式（NFA或DFA）与右线性文法的表达能力是等价的

如果文法的每个产生式都属于下列形式之一：

$$A \to Ba \qquad A \to a \qquad A \to \varepsilon$$

这样的文法称为左线性文法.

- **对于左线性文法而言，上述结论也成立**
- **思考：如何直接将正则表达式转为CFG?**

性阻局的法文关无文下上

在处理程序时，无法解决诸如以下问题:

变量先声明，再使用

调用函数时，实参个数和形参个数一致

留到语义分析阶段解决

中山大學

题 问

给出一个CFG，如何判断一个字符串是否属于它定义的语言？