Information Science and Technology College of Northeast Normal University

# Franklin’s 13 virtues and Puritan ethics

**Temperance**
**Silence**
**Order**
**Resolution**
**Frugality**
**Industry**

**(7) Sincerity: Use no hurtful deceit; think innocently and justly, and, if you speak, speak accordingly.**

诚恳：绝对不可以欺骗他人，考虑问题应公平公正，襟怀坦荡，发表观点时要有理有据，尊重事实

**(8)Justice: Wrong none by doing injuries, or omitting the benefits that are your duty.**

正直：不做有损他人之事，不要放弃履行行善助人的义务，从而伤害他人。

**Injustice：In the part of this universe that we know there is great injustice, and often the good suffer, and often the wicked prosper, and one hardly knows which of those is the more annoying.----Russell(**伯特兰·阿瑟·威廉·罗素（**Bertrand Arthur William Russell，1872年—1970年）**，英国哲学家、数学家、逻辑学家、历史学家、文学家**)**

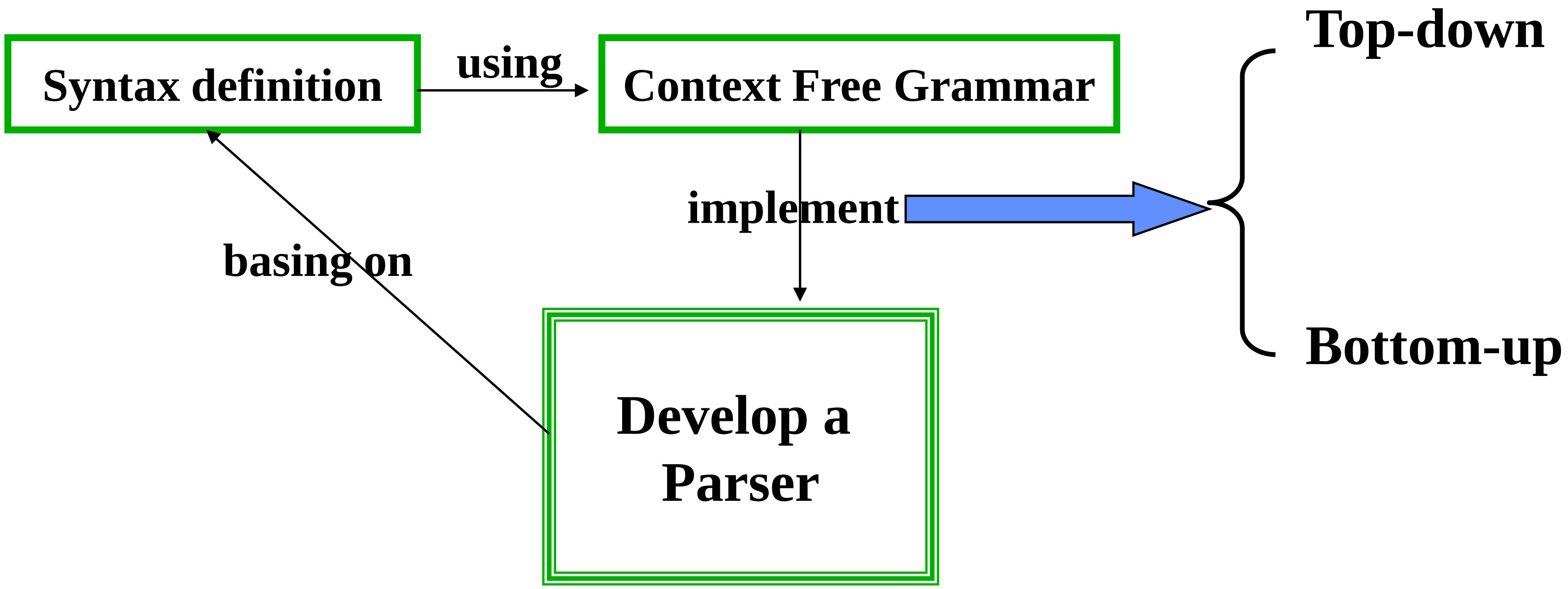
Information Science and Technology College of
Northeast Normal University

# Compiling and Running of Program

**Dr. Zheng Xiaojuan**
**Professor**

**November. 2019**

# Knowledge Relation Graph



## § 5 Bottom-up Parsing

## 5.1 Overview of Bottom-up Parsing

- **Review of Top-down Parsing**
- **Example of Bottom-up Parsing**
- **Some basic concepts**
- **Main Idea of Bottom-up Parsing**
- **Key Problem in Bottom-up Parsing**

## Review of Top-down Parsing

**P:**

(1) Z → ABd
(2) A → a
(3) B → d
(4) B → c
(5) B → bB

|  |  |  |  |
|---|---|---|---|
| a | b | c | d |

| 句型 | 输入 | 动作 |
|---|---|---|
| Z | abcd（右移）<br>↑ | Derive (1) |
| ABd | abcd | Derive(2) |
| aBd | abcd | Match |
| Bd | bcd | Derive(5) |
| bBd | bcd | Match |
| Bd | cd | Derive(4) |
| cd | cd | Match |
| d | d | Match |
|  |  | Succeed |

# Example (Bottom-up Parsing)

P:

(1) $Z \rightarrow ABd$

(2) $A \rightarrow a$

(3) $B \rightarrow d$

(4) $B \rightarrow c$

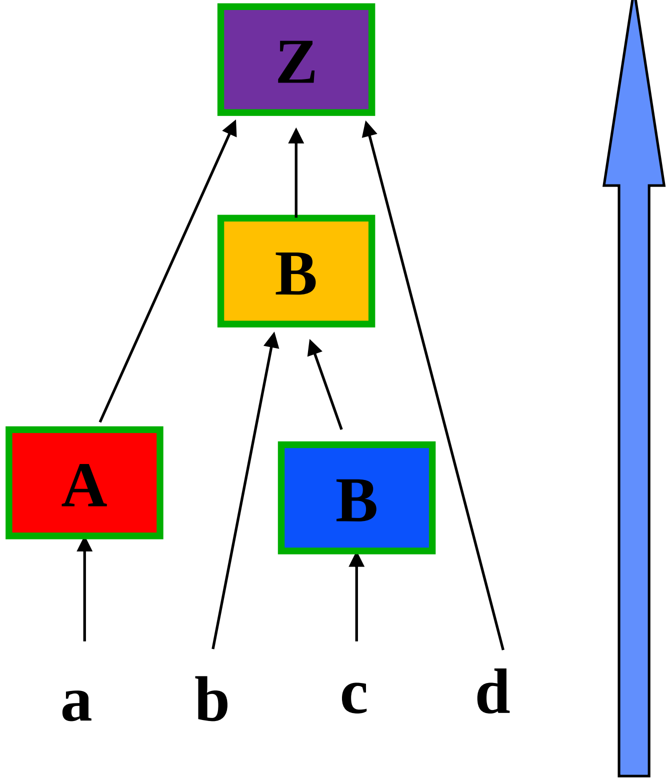
(5) $B \rightarrow bB$

| | | | |
|---|---|---|---|
| a | b | c | d |

| 符号栈 | 输入流 | 分析动作 |
|---|---|---|
| | abcd | Shift（移入） |
| a | bcd | Reduce(2)（产生式右部） |
| A | bcd | Shift |
| Ab | cd | Shift |
| Abc | d | Reduce(4) |
| A<u>bB</u> | d | Reduce(5) |
| AB | d | Shift |
| <u>ABd</u> | | Reduce(1) |
| Z | | Succeed |

P:
(1) Z → ABd
(2) A → a
(3) B → d
(4) B → c
(5) B → bB

# Example (Bottom-up Parsing)



$Z \Rightarrow \underline{ABd}$
$\Rightarrow A\underline{bB}d$
$\Rightarrow \underline{A}bcd$
$\Rightarrow abcd$
Right-most derivation

abcd
---> $\underline{A}bcd$
---> $A\underline{bB}d$
---> $\underline{ABd}$
---> Z

自底向上分析中归约过程的逆过程就是该句子的最右推导;

## 5.1 Overview of Bottom-up Parsing

- **Main Idea:**
  - **Start from the input string;**
  - **Try to reduce a proper sub-string to a non-terminal symbol;**
  - **Until generate start symbol or find error;**
- **Actions: shift(移入), reduce(归约)**
- **Different methods:**
  - **LR method;　简单优先关系法**
- **Key problem:**
  - **Find a proper substring matching the right side of a production;**

## Some Basic Concepts

- 短语（子树）
  - 一个句型形如$\alpha\beta\gamma$，如果存在一个句型$\alpha A\gamma$，而且 $A \Rightarrow^{+} \beta$，则称$\beta$为句型$\alpha\beta\gamma$的*短语*;
- 简单短语（一层子树）
  - 一个句型形如$\alpha\beta\gamma$，如果存在一个句型$\alpha A\gamma$，而且 $A \rightarrow \beta$，则称$\beta$为句型$\alpha\beta\gamma$的*简单短语*;
- 句柄:一个句型的最左简单短语称为*句柄*(handler)；（最左一层子树）
- 句柄的唯一性：如果一个CFG无二义性，则它的任意一个句型都有唯一的句柄;

## Some Basic Concepts

在一个句型对应的语法树中

- **短语：**

以某非终结符为根的两代以上的子树的所有末端结点从左到右排列就是相对于该非终结符的一个短语。

- **直接短语：**

如果子树只有两代，则该短语就是直接短语。

- **句柄：**

最左两代子树末端就是句柄。

## Some Concepts

P:
(1) E → T
(2) E → E + T
(3) T → F
(4) T → T * F
(5) F → (E)
(6) F → i
(7) F → n

句型:  T+ (E+T)*i

短语: T, (E+T), i, (E+T)*I
E+T , T+ (E+T)*i （非终极符节点构成的子树）

简单短语: T, i, E+T

句柄:  T

It is easy to collect 短语, 简单短语和句柄 by building parse tree;

## Some Basic Concepts

- 规范推导
  - 一个句型的最右推导称为该句型的规范推导；
- 规范句型
  - 从开始符通过规范推导得到的句型；

## Some Basic Concepts

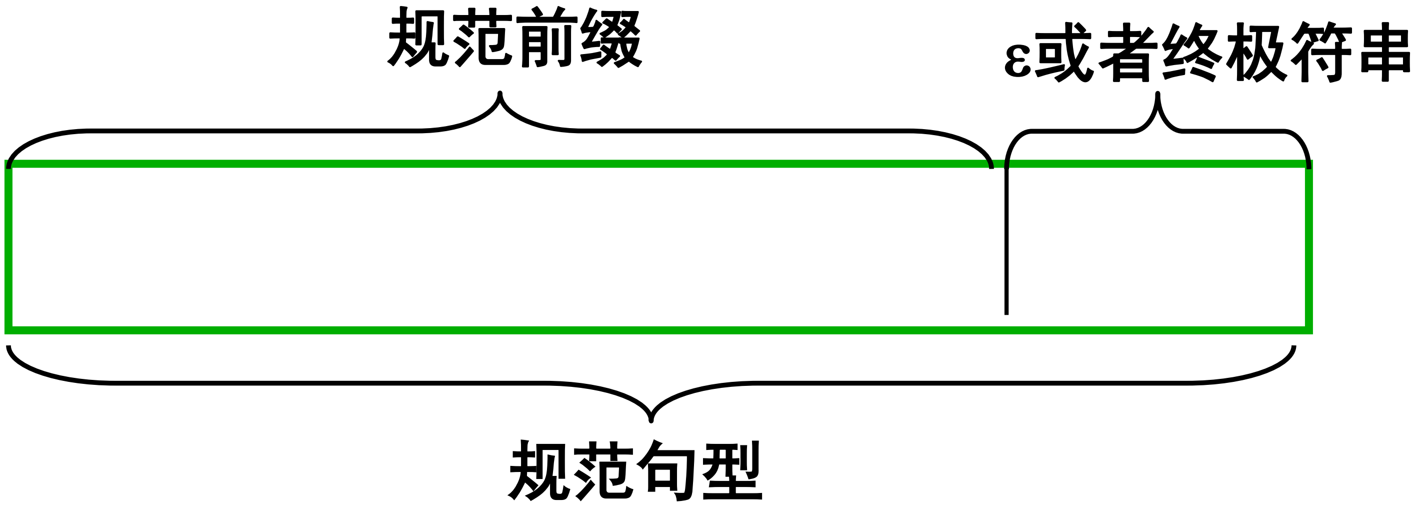
- 归约(reduce):是推导的逆过程;
  - G是一个CFG, A→β是一个产生式, 如果αβγ是一个符号串, 则称αAγ是由 αβγ用产生式A→β归约而得到的;
- 可以看出,如果αβγ是一个句型,那么β是一个简单短语;
- 规范归约:如果对一个句型对其最左的简单短语(句柄)进行归约,那么称为规范归约;
- 对于同一个句子, 规范推导和规范归约之间存在着互逆的关系;

## Some Basic Concepts

- 规范前缀:一个规范句型的一个前缀称为规范前缀，如果该前缀后面的符号串不包含非终极符;

$Z \Rightarrow ABd$ 规范前缀为 AB, ABd

$Z \Rightarrow^{+} Acd$ 规范前缀为 A, Ac, Acd



## Some Basic Concepts

- 规范活前缀:满足如下条件之一的规范前缀称为规范活前缀:
  - 该规范前缀不包含简单短语;
  - 该规范前缀只包含一个简单短语,而且是在该规范前缀的最后(这个简单短语就是句柄);

$Z \Rightarrow ABd$　　　　规范前缀为 AB, ABd

规范活前缀:AB(不包含简单短语), ABd(包含一个简单短语)

$Z \Rightarrow^{+} abcd$　　　　规范前缀为 a, ab, abc, abcd

规范活前缀: $\underline{a}$ (包含一个简单短语)

(1)$Z \rightarrow ABd$ (2) $A \rightarrow a$ (3) $B \rightarrow d$ (4) $B \rightarrow c$ (5) $B \rightarrow bB$

# Example (Bottom-up Parsing)

P:

(1) Z → ABd

(2) A → a

(3) B → d

(4) B → c

(5) B → bB

| | | | |
|---|---|---|---|
| a | b | c | d |

| 符号栈 | 输入流 | 分析动作 |
|---|---|---|
| | abcd | Shift |
| a | bcd | Reduce(2) |
| A | bcd | Shift |
| Ab | cd | Shift |
| Abc | d | Reduce(4) |
| 规范活前缀 | ε或者终极符串 | Reduce(5) |
| | | Shift |
| ABd | | Reduce(1) |
| Z | | Succeed |

$\underbrace{\hspace{6em}}_{\text{规范前缀}}$

## Some Basic Concepts

- 规范活前缀决定分析动作
  - 归约：该规范前缀只包含一个简单短语，而且是在该规范前缀的最后；
    可归约规范活前缀
  - 移入：规范前缀不包含简单短语；
    移入型规范活前缀

$Z \Rightarrow ABd$ 规范前缀为 AB, ABd

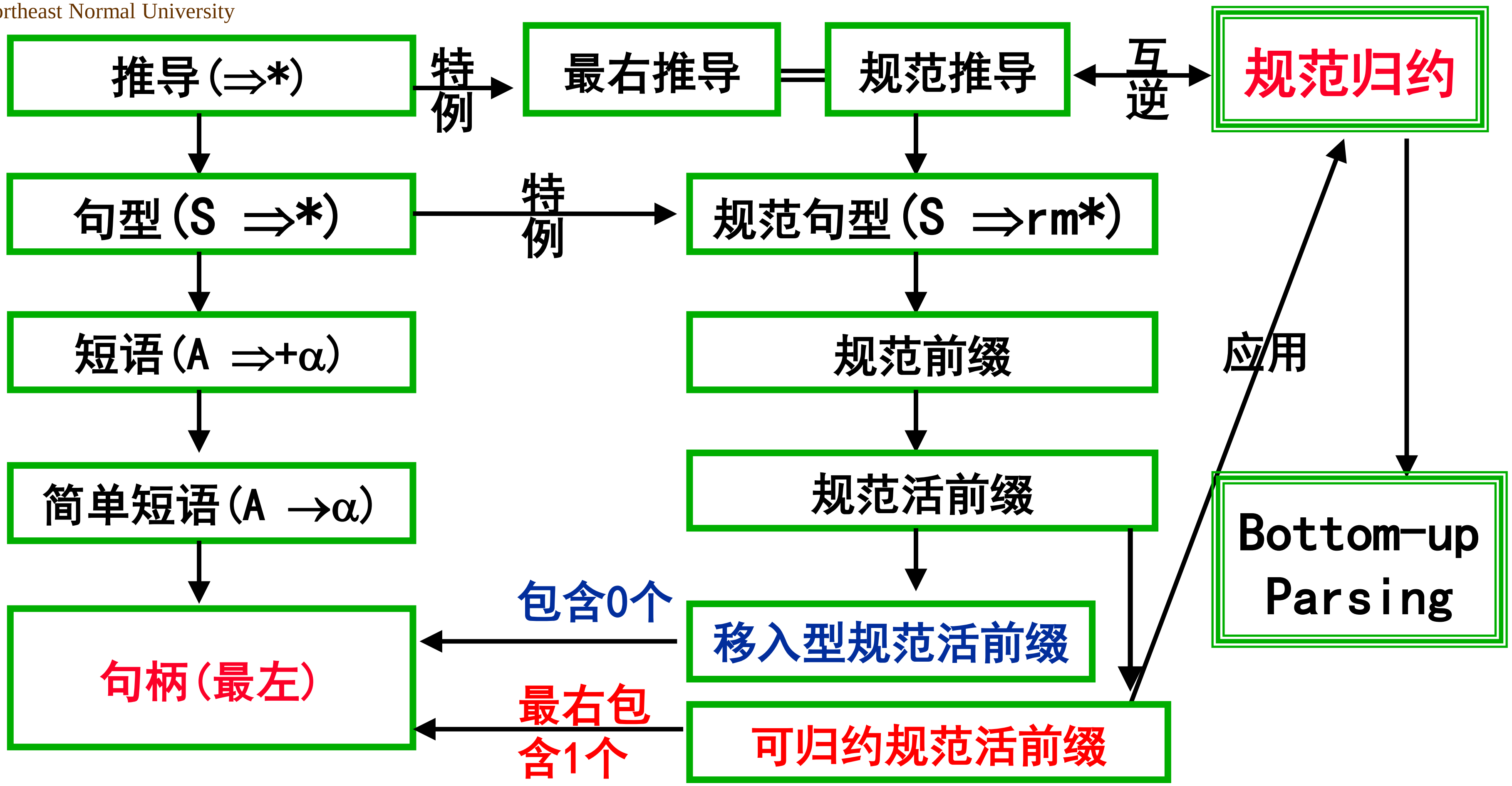
规范活前缀: AB(不包含简单短语) --- 移入型规范活前缀

ABd(包含一个简单短语) --- 可归约规范前活缀

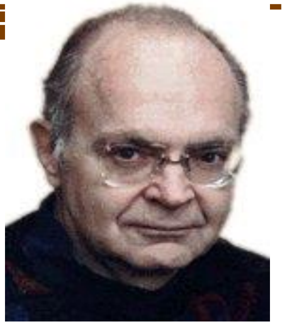
$(1) Z \rightarrow ABd$ $(2) A \rightarrow a$ $(3) B \rightarrow d$ $(4) B \rightarrow c$ $(5) B \rightarrow bB$

Information Science and Technology College of
Northeast Normal University

# Knowledge Relation Graph



## 5.1 Overview of Bottom-up Parsing



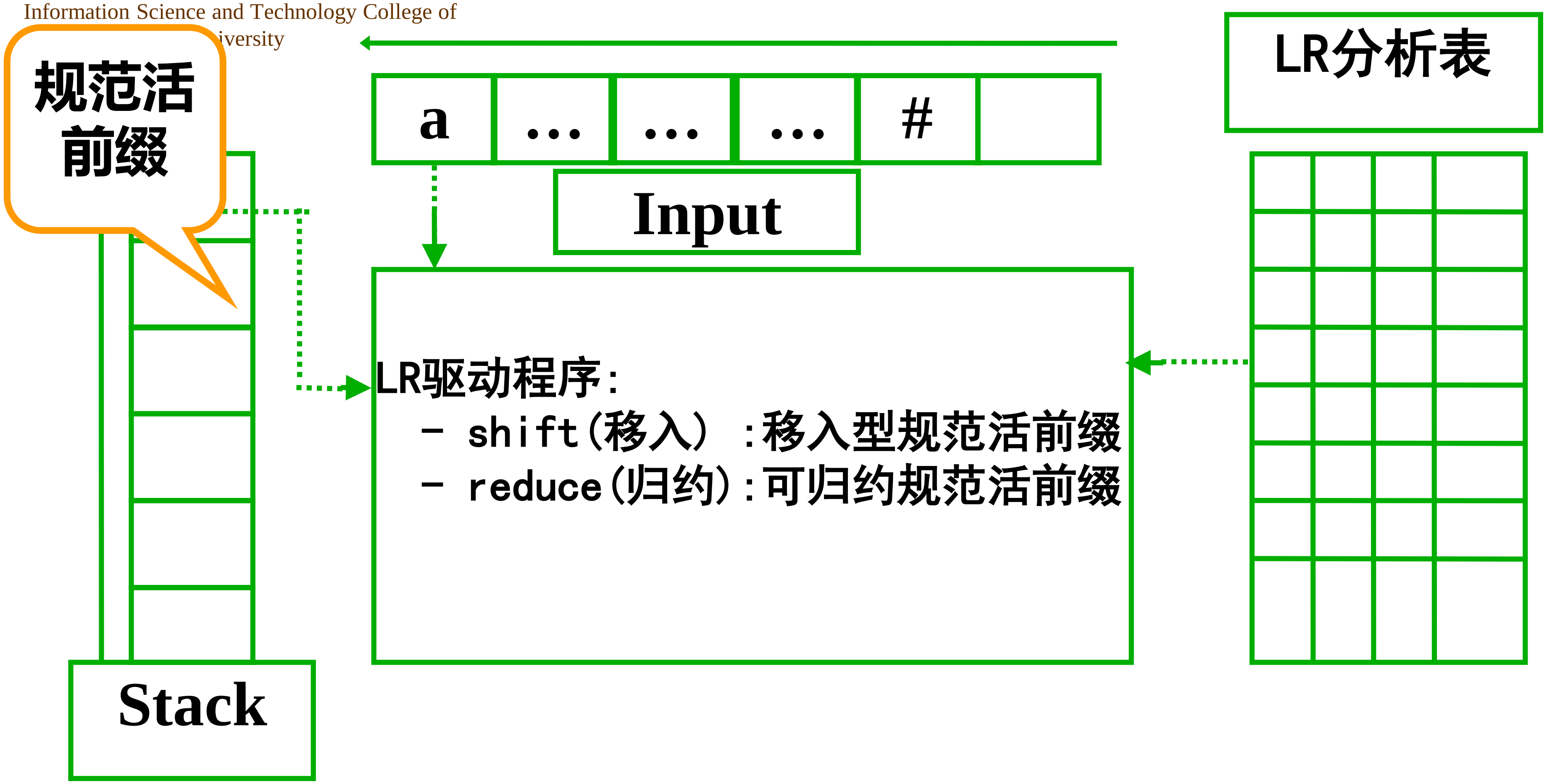
- **LR method**

**1965, Knuth提出，《the art of the computer programming》，**与爱因斯坦《相对论》、狄拉克《量子力学》、理查·费曼《量子电动力学》等经典比肩而立

Bill Gates "If you think you're a really good programmer… read (Knuth's) *Art of Computer Programming*… You should definitely send me a résumé if you can read the whole thing."

- **Main idea**
  - **L:Left-right (从左至右) read one input string;**
  - **R:自下而上进行规约（每次找到句柄）；**
  - **Until Reduce to start symbol(归约直到得到开始符) or error;**
- **Key problem: 对于一个CFG, 如何判定可归约活前缀?**
  - **构造一个判定可归约活前缀的自动机 -- LR自动机**

## LR Parsing Mechanism



LR驱动程序:
- shift(移入) :移入型规范活前缀
- reduce(归约):可归约规范活前缀

## 5.1 Overview of Bottom-up Parsing

- **LR method**
  - **Different LR methods**
    - **LR(0)**
    - **SLR(1)**
    - **LR(1)**
    - **LALR(1)**
  - 不同的LR方法对CFG的要求不一样，能够分析的CFG多少也不一样，**LR(0) $\subseteq$ LALR(1) $\subseteq$ LR(1)**

## Key Problems in LR Parsing

- 如何判定规范活前缀?
- 如何判定移入型规范活前缀?
- 如何判定可归约规范活前缀以及用哪一个产生式归约?
- 如何判定分析结束?

LR(0)自动机可以解决这些问题!

## 5.2 Finite Automata for LR(0) Parsing

- 构造LR(0)自动机的依据 --- 派生定理
- Example
- LR(0)自动机
  - LR(0) item (项)
  - LR(0) automata
- The process to construct LR(0) automata

## 构造LR(0)自动机的依据 --- 派生定理

- 派生定理：给出了如何对任意一个CFG构造可归约规范活前缀的方法

  [1]把CFG扩展为增广文法，S’ → S#；（开始符不在产生式右部及保证开始符号的定义只有一个产生式）

  [2]文法开始符S是可归约规范活前缀；

  [3]如果αAβ是一个可归约规范活前缀，且有产生式A→π，那么，απ也是一个可归约规范活前缀；（α，β和π是有终极符和非终极符构成的符号串，也可以是空串）

  [4]任何一个可归约规范活前缀都是按照[3]方式派生出来的；(证明略)

## Example

$V_T = \{a, b, c\}$
$V_N = \{S, A\}$
$S = S$
P:
$\{ \; S' \to S$
$S \to aAc$
$A \to Abb$
$A \to b$
$\}$

- [1] {S}
- [2] S和S → aAc产生{aAc}
- [3] aAc和A → Abb产生{aAbb}
- [4] aAc和A → b产生{ab}
- [5] aAbb和A → Abb产生{aAbb}
- [6] aAbb和A → b产生{ab}
- [7] 最后,合起来得到

{S, aAc, aAbb, ab}

## Example

$V_T = \{a, b, c\}$
$V_N = \{S, A, B\}$
$S = S$
P:
{ $S' \rightarrow S$
$S \rightarrow aAc$
$A \rightarrow ABb$
$A \rightarrow Ba$
$B \rightarrow b$
}

- [1] {S}
- [2] S和S→ aAc产生{aAc}
- [3] aAc和A → ABb产生{aABb}
- [4] aAc和A → Ba产生{aBa}
- [5] aABb和B → b产生{aAb}
- [6] aBa和B → b产生{ab}
- [7]最后，合起来得到

{S, aAc, aABb, aBa, aAb, ab}

## Example

$V_T = \{a, b, c\}$

$V_N = \{S, A, B\}$

S = S

P:

{ S' → S

S → aAc

A → ABb

A → a

B → bB

B → b

}

- [1] {S}
- [2] S和S → aAc产生{aAc}
- [3] aAc和A → ABb产生{aABb}
- [4] aAc和A → a产生{aa}
- [5] aABb和B → bB产生{aAbB}
- [6] aABb和B → b产生{aAb}
- [7] aAbB和B → bB产生{aAbbB}
- [8] aAbB和B → b产生{aAbb}
- ……

## LR(0)自动机

- LR(0) item(项目):带圆点的产生式, 圆点只能出现在产生式的右部符号串的任意位置;目的是为了在分析过程中看到（识别）产生式多大部分。
  - S→ •aAc（识别a）
  - S → a • Ac（识别A）
  - S → a Ac • （全部识别完，可以规约）
  - 空产生式: S → •

An item indicates how much of a production we have seen at a given point in the parsing process.

## LR(0)自动机

- LR(0)项目集合*关于符号X的投影*
  - IS is a set of LR(0) items;
  - X is a symbol;
  - $IS_{(X)}$ represents the projection of IS with respect to X:
  - $IS_{(X)} = \{S \rightarrow \alpha X \bullet \beta \mid S \rightarrow \alpha \bullet X \beta \in IS, X \in V_T \cup V_N\}$
- $IS = \{A \rightarrow A \bullet Bb, B \rightarrow a \bullet, B \rightarrow b \bullet B, Z \rightarrow \bullet cB\}$
- $X = B$
- $IS_{(B)} = \{A \rightarrow AB \bullet b, B \rightarrow bB \bullet\}$

## LR(0)自动机

- ***<u>LR(0)项目集合的闭包</u>***
  - IS is a set of LR(0) items;
  - CLOSURE(IS)是一个LR(0)项目集合，按照下面的步骤计算：
    - [1]初始，CLOSURE(IS) = IS
    - [2]对于CLOSURE(IS)没有处理的LR(0)项目，
    - 如果该项目的原点后面是一个非终极符A，
    - 而且A的全部产生式是{$A \to \alpha 1$，…，$A \to \alpha n$}
    - 则增加如下LR(0)项目到CLOSURE(IS)
    - {$A \to \bullet \alpha 1$，…，$A \to \bullet \alpha n$}
    - [3]重复[2]直到 CLOSURE(IS)收敛；

## LR(0)自动机

$V_T = \{a, b, c\}$
$V_N = \{S, A, B\}$
$S = S$
P:
$\{\ S \rightarrow aAc$
$A \rightarrow ABb$
$A \rightarrow Ba$
$B \rightarrow b$
$\}$

$IS = \{S \rightarrow \bullet aAc\}$
$CLOSURE(IS) = \{S \rightarrow \bullet aAc\}$

$IS = \{S \rightarrow a \bullet Ac\}$
$CLOSURE(IS)$
$= \{S \rightarrow a \bullet Ac,$
$A \rightarrow \bullet ABb, A \rightarrow \bullet Ba,$
$B \rightarrow \bullet b\}$

## LR(0)自动机

- **goto函数**
  - **IS is a set of LR(0) items;**
  - **X is a symbol;**
  - **goto(IS, X) = CLOSURE(IS)$_{(X)}$ （先投影，再闭包）**

## LR(0)自动机

- **LR(0) automata for a CFG G={$V_T$, $V_N$, S, P}**
  - **增广产生式 S’ $\rightarrow$ S# prupose:to indicate to the parse when it should stop parsing and announce acceptance of the input.**
  - **$\sum = V_T \cup V_N \cup \{\#\}$**
  - **S0 = CLOSURE(S’ $\rightarrow$ •S#)**
  - **SS = a set of states, each state is a set of LR(0) items;**
  - **TS （规约项目）**
  - **Φ （goto函数）**

**Generate by following the process of constructing LR(0)自动机**

# The process to construct LR(0) Automata

$V_T = \{a, b, c\}$

$V_N = \{S, A, B\}$

S = S

P:

{ S’ → S

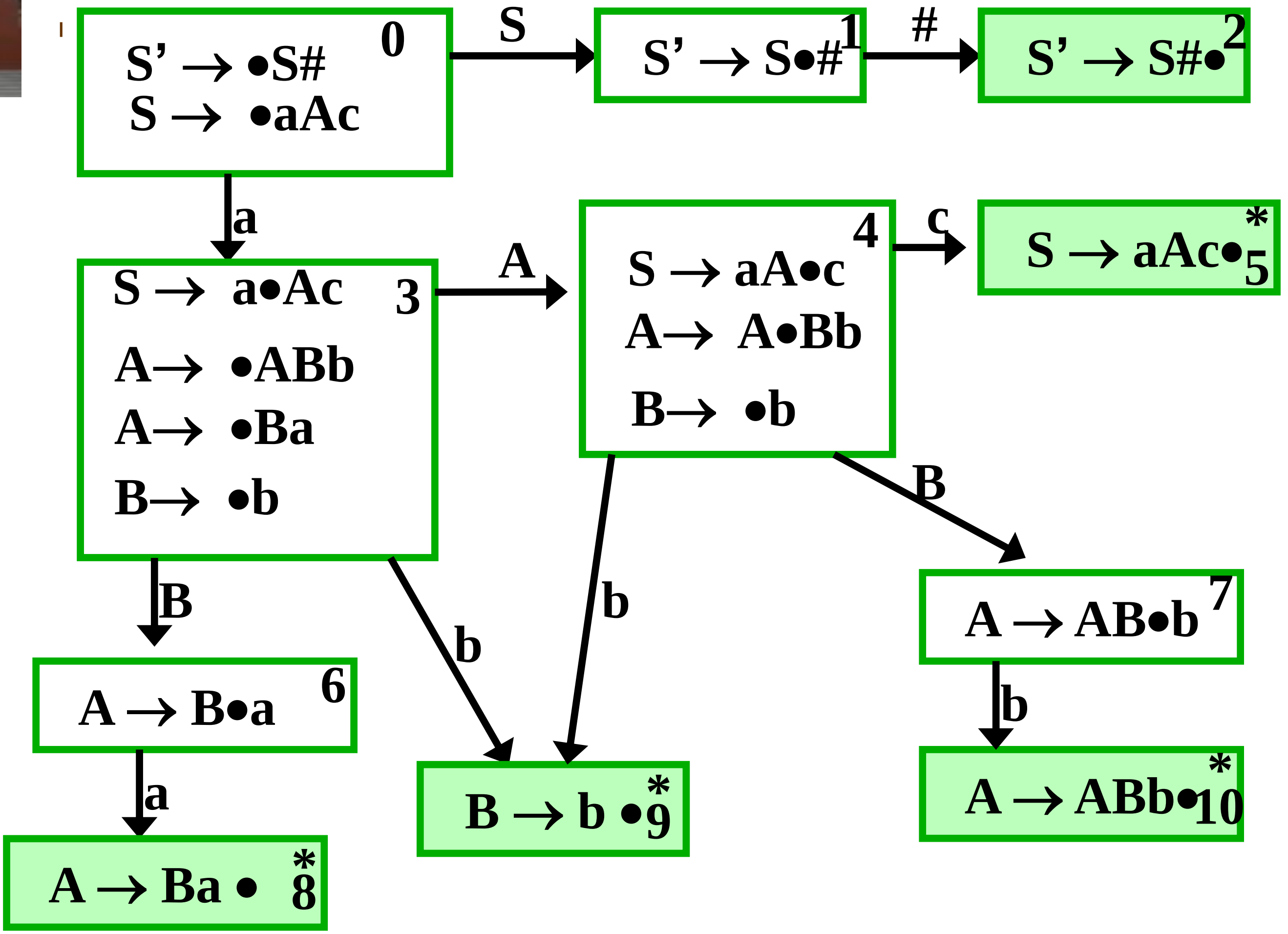
S → aAc

A → ABb

A → Ba

B → b

}



可归约活前缀集合为{S, aAc, aABb, aBa, aAb,ab}

## The process to construct LR(0) Automata

$V_T$ = {a, b, c}
$V_N$ = {S, A, B}
S = S
P:
{ S→ aAc
A → ABb
A → Ba
B → ε
}

$$\begin{tikzcd}
\boxed{\begin{array}{l} S' \to \bullet S\# \quad 0 \\ S \to \bullet aAc \end{array}} \arrow[r, "S"] \arrow[d, "a"] & \boxed{S' \to S\bullet\#} \arrow[r, "\#"] & \boxed{S' \to S\#\bullet \ ^{*}} \\
\boxed{\begin{array}{l} S \to a\bullet Ac \\ A \to \bullet ABb \\ A \to \bullet Ba \\ B \to \bullet \quad * \end{array}} \arrow[r, "A"] \arrow[d, "B"] & \boxed{\begin{array}{l} S \to aA\bullet c \\ A \to A\bullet Bb \\ B \to \bullet \quad * \end{array}} \arrow[r, "c"] \arrow[dr, "B"] & \boxed{S \to aAc\bullet \ ^{*}} \\
\boxed{A \to B\bullet b} \arrow[d, "b"] & & \boxed{A \to AB\bullet b} \arrow[d, "b"] \\
\boxed{A \to Bb\bullet \ ^{*}} & & \boxed{A \to ABb\bullet \ ^{*}}
\end{tikzcd}$$

## The process to construct LR(0) Automata

[1]增广产生式 $S' \to S\#$

[2]$\Sigma = V_T \cup V_N \cup \{\#\}$

[3]$S0 = CLOSURE(S' \to \bullet S)$

[4]$SS = \{S0\}$

[5]对于ISS中的每一个项目IS, 和每个符号$X \in \Sigma$,
计算IS' = goto(IS, X),
如果IS'不为空, 则建立 $IS \xrightarrow{X} IS'$,
如果IS'不为空, 且IS'不属于ISS,则把IS'加入ISS;

[6]重复[5]直到ISS收敛;

[7]终止状态:包含形如$A \to \alpha\bullet$项目的项目集合;

# Assignment

$V_T = \{a, b, c, d\}$

$V_N = \{Z, A, B\}$

$S = Z$

P:

$\{Z \rightarrow ABd$

$A \rightarrow a$

$B \rightarrow d$

$B \rightarrow \varepsilon$

$B \rightarrow bB$

$\}$

构造LR(0) 自动机