形式语言与自动机实验二：设计上下文无关文法的变换算法实验报告

毛子恒　　邹宇江　　王敏行　　曾嘉伟

北京邮电大学 计算机学院

日期：2021 年 6 月 1 日

小组成员

班级：2019211309	姓名：毛子恒	学号：2019211397	分工：代码 文档
班级：2019211309	姓名：邹宇江	学号：2019211416	分工：测试 文档
班级：2019211309	姓名：王敏行	学号：2019211410	分工：测试 文档
班级：2019211309	姓名：曾嘉伟	学号：2019211396	分工：测试 文档

1　需求分析

1.1　题目描述

编程实现上下文无关文法的变换算法，用于消除文法中的 ε 产生式、单产生式以及无用符号。

1.2　输入描述

程序从标准输入中读入数据。

以下描述中**字符串**均特指不包含空格、回车等特殊字符的 ASCII 字符序列。**要求输入是一个合法的上下文无关文法**。

第一行输入若干字符串，用空格分隔，表示上下文无关文法 $\mathbf{G}$ 的非终止符集合 $\mathbf{N}$，**要求输入不出现重复符号**。

第二行输入若干字符串，用空格分隔，表示上下文无关文法 $\mathbf{G}$ 的终止符集合 $\mathbf{T}$，**要求输入不出现重复符号**。

接下来的若干行，表示上下文无关文法 $\mathbf{G}$ 的生成式集合 $\mathbf{P}$。每行若干个字符串，其中第一个字符串表示生成式左边的符号，其余字符串依次为生成式右边的符号，**要求输入的符号均包含在 $\mathbf{N} \cup \mathbf{T}$ 中**。使用特殊的字符串"[empty]" 表示空串。以一个空行表示 $\mathbf{P}$ 的结束。

最后一行输入一个字符串，表示起始符，**要求起始符包含在 $\mathbf{N}$ 中**。

1.3 输出描述

程序向标准输出中输出提示信息和每一步算法的运行结果。

1.4 样例

1.4.1 样例输入

```
S A B C D
a b c d
S a
S b A
S B
S c c D
A a b B
A [empty]
B a A
C d d C
D d d d

S
```

1.4.2 样例输出

```
N: {S, B, D, A, C, }
T: {c, b, d, a, }
P:
        S -> a | B | bA | ccD |
        A -> abB | [empty] |
        B -> aA |
        C -> ddC |
        D -> ddd |
S: S
消去 epsilon 产生式
N: {S, B, D, A, C, }
T: {c, b, d, a, }
P:
        S -> ccD | a | b | B | bA |
        A -> abB |
        B -> a | aA |
        C -> ddC |
        D -> ddd |
S: S
消去单产生式
N: {S, B, D, A, C, }
T: {c, b, d, a, }
P:
        S -> ccD | aA | a | b | bA |
        B -> a | aA |
```

```
        D -> ddd |
        A -> abB |
        C -> ddC |
S: S
消去无用符号
N: {A, D, S, B, }
T: {a, c, b, d, }
P:
        S -> ccD | aA | a | b | bA |
        B -> a | aA |
        D -> ddd |
        A -> abB |
S: S
```

1.4.3 样例运行结果

见图 1。

2 程序设计

2.1 环境

- macOS Big Sur 11.3
- Python 3.7.9

2.2 设计思路

将符号用字符串表示，$\mathbf{G}$ 和 $\mathbf{T}$ 用 Python 集合表示，$\mathbf{P}$ 用 Python 字典表示，字典的键为生成式左端的符号，值为一个列表，列表的元素是元组，元组内的每个元素是一个符号。

例如，$S \to a|bA|ccD$ 表示成字典中的一个元素是：`{'S': [('a',), ('b', 'A'), ('c', 'c', 'D')]}`

在消去 ε 产生式的算法中，涉及到将 Y_i 取 C_i 或 ε 的所有组合枚举出来（共 2^n 种），我采用枚举二进制数的方法，枚举 $num \in [0, 2^n - 1]$，如果 num 的第 i 位是 1，则 $Y_i = C_i$，否则 $Y_i = \varepsilon$。

判断 $\omega \in \mathbf{N}^*$ 时，需要判断 $\forall a \in \omega, a \in \mathbf{N}$，枚举 $\mathbf{P}$ 需要二重循环，除此之外所有的操作基本都是集合与元素的操作，和伪代码几乎完全一致。

2.3 核心算法伪代码

消去 ε 产生式的伪代码见**算法 1**；消去单产生式的伪代码见**算法 2**；消去无用符号的伪代码见**算法 3**；

3 调试分析

3.1 结果分析

经过几组数据的验证，程序能正确完成消除 ε 产生式、单产生式以及无用符号的功能。

```
xqmmcqs@xqmmcqsdeMacBook-Pro Lab2 % /Users/xqmmcqs/anaconda3/bin/python /Users/xqmmcqs/Documents/FL_Work/Lab2/main.py
请输入非终结符，以空格分隔：
S A B C D
请输入终结符，以空格分隔：
a b c d
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
S a
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
S b A
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
S B
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
S c c D
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
A a b B
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
A [empty]
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
B a A
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
C d d C
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：
D d d d
请输入一个生成式，以空格分隔每个符号，第一个符号为生成式的左端，以[empty]表示空串，以空行表示输入生成式结束：

请输入起始符：
S
N: {C, D, A, S, B, }
T: {c, b, a, d, }
P:
        S -> bA | B | a | ccD |
        A -> [empty] | abB |
        B -> aA |
        C -> ddC |
        D -> ddd |
S: S
消去epsilon产生式
N: {C, D, A, S, B, }
T: {c, b, a, d, }
P:
        S -> b | bA | a | B | ccD |
        A -> abB |
        B -> aA | a |
        C -> ddC |
        D -> ddd |
S: S
消去单产生式
N: {C, D, A, S, B, }
T: {c, b, a, d, }
P:
        C -> ddC |
        D -> ddd |
        A -> abB |
        S -> b | bA | aA | a | ccD |
        B -> aA | a |
S: S
消去无用符号
N: {D, B, S, A, }
T: {c, b, a, d, }
P:
        D -> ddd |
        A -> abB |
        S -> b | bA | aA | a | ccD |
        B -> aA | a |
S: S
xqmmcqs@xqmmcqsdeMacBook-Pro Lab2 %
```

图 1: 样例运行结果

算法 1: 消去 ε 产生式

输入: $\mathbf{G} = (\mathbf{N}, \mathbf{T}, \mathbf{P}, S)$

输出: $\mathbf{G}_1 = (\mathbf{N}_1, \mathbf{T}, \mathbf{P}_1, S_1)$

Function `eliminate_epsilon()`

- $\mathbf{N_0} \leftarrow \varnothing$;
- $\mathbf{N}' \leftarrow \varnothing$;
- **foreach** $A \to \varepsilon \in \mathbf{P}$ **do**
  - $\mathbf{N}' \leftarrow \mathbf{N}' \cup \{A\}$;
- **while** $\mathbf{N_0} \neq \mathbf{N}'$ **do**
  - $\mathbf{N_0} \leftarrow \mathbf{N}'$;
  - **foreach** $A \to \omega \in \mathbf{P}$ **do**
    - **if** $\omega \in \mathbf{N}_0^*$ **then** $\mathbf{N}' \leftarrow \mathbf{N}' \cup \{A\}$;
- $\mathbf{P}_1 \leftarrow \varnothing$;
- **foreach** $A \to \omega \in \mathbf{P}$ **do**
  - $n \leftarrow |\{C | C \in \omega \wedge C \in \mathbf{N}'\}|$;
  - **for** num $\leftarrow 0$ **to** $2^n - 1$ **do**
    - // num $= \sum_{i=0}^{n} \text{num}_i \times 2^i$
    - $\omega_1 \leftarrow \varepsilon$;
    - total $\leftarrow 0$;
    - **foreach** $a \in \omega$ **do**
      - **if** $a \in \mathbf{N}'$ **then**
        - **if** $\text{num}_{\text{total}} = 1$ **then** $\omega_1 \leftarrow \omega_1 a$;
        - total $\leftarrow$total $+1$;
      - **else** $\omega_1 \leftarrow \omega_1 a$;
    - **if** $\omega_1 \neq \varepsilon$ **then** $\mathbf{P}_1 \leftarrow \mathbf{P}_1 \cup \{A \to \omega_1\}$;
- **if** $S \in \mathbf{N}'$ **then**
  - $\mathbf{P}_1 \leftarrow \mathbf{P}_1 \cup \{S_1 \to \varepsilon, S_1 \to S\}$;
  - $\mathbf{N}_1 \leftarrow \mathbf{N} \cup \{S_1\}$;
- **else**
  - $\mathbf{N}_1 \leftarrow \mathbf{N}$;
  - $S_1 \leftarrow S$;

算法 2: 消去单产生式

输入: $\mathbf{G} = (\mathbf{N}, \mathbf{T}, \mathbf{P}, S)$

输出: $\mathbf{G}_1 = (\mathbf{N}, \mathbf{T}, \mathbf{P}_1, S)$

```
1  Function eliminate_single()
2      foreach A ∈ N do
3          N' ← {A};
4          do
5              N_0 ← N';
6              foreach B ∈ N_0 do
7                  foreach B → C ∈ P do
8                      if C ∈ N then N' ← N' ∪ {C};
9          while N_0 ≠ N';
10         P_1 ← ∅;
11         foreach B ∈ N_0 do
12             foreach B → ω ∈ P do
13                 if ¬ω ∈ N then P_1 ← P_1 ∪ {A → ω};
```

3.2 改进的设想

一个全集 $\mathbf{U}$ 的子集 $\mathbf{S}$ 可以用一个二进制数来表示，集合的交、并操作可以转化为二进制数的或、与运算。

上述三个算法中的绝大部分操作都是对集合以及集合中元素的操作，因此可以将符号编号，之后将算法中描述的各个集合和 $\mathbf{P}$ 中的每个生成式的右端表示为一个二进制数，采用数的运算来代替集合的运算。

这样做的好吃是速度相比 Python 中数据结构的运算快很多，但是实现不太直观，比较难 debug。

由于上下文无关文法不合法的情况太多，难以一一判断，所以默认输入的是合法的上下文无关文法，仅在输入时对一些简单的不合法情况（生成式左端不是非终结符、生成式中的符号不在 $\mathbf{N} \cup \mathbf{T}$ 中）做了处理。

3.3 实验总结

本次实验中我设计实现了上下文无关文法的变换算法，对这些算法的运行过程和原理有了更加深刻的理解，同时增强了我的 Python 编程能力。

算法 3: 消去无用符号

输入: $\mathbf{G} = (\mathbf{N}, \mathbf{T}, \mathbf{P}, S)$

输出: $\mathbf{G}_1 = (\mathbf{N}_2, \mathbf{T}_1, \mathbf{P}_2, S)$

Function `eliminate_useless()`

$\mathbf{N_0} \leftarrow \varnothing$;
$\mathbf{N}' \leftarrow \varnothing$;
foreach $A \to \omega \in \mathbf{P}$ **do**
  if $\omega \in \mathbf{T}^*$ **then** $\mathbf{N}' \leftarrow \mathbf{N}' \cup \{A\}$;
while $\mathbf{N_0} \neq \mathbf{N}'$ **do**
  $\mathbf{N_0} \leftarrow \mathbf{N}'$;
  foreach $A \to \omega \in \mathbf{P}$ **do**
    if $\omega \in (\mathbf{T} \cap \mathbf{N}_0)^*$ **then** $\mathbf{N}' \leftarrow \mathbf{N}' \cup \{A\}$;
$\mathbf{N}_1 \leftarrow \mathbf{N}'$;
$\mathbf{P}_1 \leftarrow \varnothing$;
foreach $A \to \omega \in \mathbf{P}$ **do**
  if $A \in \mathbf{N}_1 \wedge \omega \in (\mathbf{N}_1 \cup \mathbf{T})^*$ **then** $\mathbf{P}_1 \leftarrow \mathbf{P}_1 \cup \{A \to \omega\}$;
$\mathbf{N}' \leftarrow \{S\}$;
do
  $\mathbf{N_0} \leftarrow \mathbf{N}'$;
  foreach $A \in (\mathbf{N_0} \cap \mathbf{N})$ **do**
    foreach $A \to \omega \in \mathbf{P}$ **do**
      foreach $a \in \omega$ **do**
        $\mathbf{N}' \leftarrow \mathbf{N}' \cup \{a\}$;
while $\mathbf{N_0} \neq \mathbf{N}'$;
$\mathbf{N}_2 \leftarrow \mathbf{N}' \cap \mathbf{N}_1$;
$\mathbf{T}_1 \leftarrow \mathbf{N}' \cap \mathbf{T}$;
$\mathbf{P}_2 \leftarrow \varnothing$;
foreach $A \to \omega \in \mathbf{P}_1$ **do**
  if $A \in \mathbf{N}_2 \wedge \omega \in (\mathbf{N}_2 \cup \mathbf{T}_1)^*$ **then** $\mathbf{P}_2 \leftarrow \mathbf{P}_2 \cup \{A \to \omega\}$;

4 测试结果

4.1 测试集 1

4.1.1 输入

```
S A1 A2 A3 A4 A5
a b d
S A1
S A2
A1 A3
A1 A4
A2 A4
A2 A5
A3 S
A3 b
A3 [empty]
A4 S
A4 a
A5 S
A5 d
A5 [empty]

S
```

4.1.2 输出

```
N: {A5, A1, A2, A3, S, A4, }
T: {b, d, a, }
P:
        S -> A2 | A1 |
        A1 -> A4 | A3 |
        A2 -> A5 | A4 |
        A3 -> b | S | [empty] |
        A4 -> a | S |
        A5 -> S | d | [empty] |
S: S
消去 epsilon 产生式
N: {A5, A1, A2, A3, S, A4, S1, }
T: {b, d, a, }
P:
        S -> A2 | A1 |
        A1 -> A3 | A4 |
        A2 -> A5 | A4 |
        A3 -> b | S |
        A4 -> a | S |
        A5 -> S | d |
        S1 -> [empty] | S |
S: S1
```

```
消去单产生式
N: {A5, A1, A2, A3, S, A4, S1, }
T: {b, d, a, }
P:
        A5 -> b | d | a |
        A1 -> a | d | b |
        A2 -> a | d | b |
        A3 -> a | d | b |
        S -> a | d | b |
        A4 -> b | d | a |
        S1 -> b | d | a | [empty] |
S: S1
消去无用符号
N: {S1, }
T: {d, a, b, }
P:
        S1 -> b | d | a | [empty] |
S: S1
```

4.2 测试集 2

4.2.1 输入

```
S
a b
S a S b S
S b S a S
S [empty]

S
```

4.2.2 输出

```
N: {S, }
T: {b, a, }
P:
        S -> [empty] | bSaS | aSbS |
S: S
消去 epsilon 产生式
N: {S, S1, }
T: {b, a, }
P:
        S -> baS | abS | aSbS | bSa | aSb | bSaS | ba | ab |
        S1 -> [empty] | S |
S: S1
消去单产生式
N: {S, S1, }
T: {b, a, }
```

```
P:
        S -> bSaS | abS | baS | aSb | bSa | aSbS | ab | ba |
        S1 -> bSaS | abS | baS | aSb | bSa | aSbS | ab | [empty] | ba |
S: S1
消去无用符号
N: {S, S1, }
T: {b, a, }
P:
        S -> bSaS | abS | baS | aSb | bSa | aSbS | ab | ba |
        S1 -> bSaS | abS | baS | aSb | bSa | aSbS | ab | [empty] | ba |
S: S1
```

4.3 测试集 3

4.3.1 输入

```
S A B
( ) + * a
S S + A
S A
A A * B
A B
B ( S )
B a

S
```

4.3.2 输出

```
N: {A, S, B, }
T: {(, *, +, a, ), }
P:
        S -> A | S+A |
        A -> B | A*B |
        B -> (S) | a |
S: S
消去 epsilon 产生式
N: {A, S, B, }
T: {(, *, +, a, ), }
P:
        S -> A | S+A |
        A -> B | A*B |
        B -> (S) | a |
S: S
消去单产生式
N: {A, S, B, }
T: {(, *, +, a, ), }
P:
```

```
        A -> (S) | A*B | a |
        S -> S+A | (S) | A*B | a |
        B -> (S) | a |
S: S
消去无用符号
N: {A, S, B, }
T: {(, ), +, a, *, }
P:
        A -> (S) | A*B | a |
        S -> S+A | (S) | A*B | a |
        B -> (S) | a |
S: S
```

4.4 测试集 4

4.4.1 输入

```
S C D E
a b
S D C E
D C C
D [empty]
C E E
C b
E D D
E a

S
```

4.4.2 输出

```
N: {E, D, S, C, }
T: {a, b, }
P:
        S -> DCE |
        D -> [empty] | CC |
        C -> b | EE |
        E -> DD | a |
S: S
消去 epsilon 产生式
N: {S, C, E, D, S1, }
T: {a, b, }
P:
        S -> DE | CE | DCE | E | D | C | DC |
        D -> C | CC |
        C -> b | E | EE |
        E -> D | DD | a |
        S1 -> [empty] | S |
```

```
S: S1
消去单产生式
N: {S, C, E, D, S1, }
T: {a, b, }
P:
        S -> b | DE | DD | CC | EE | CE | DCE | a | DC |
        C -> b | DD | EE | CC | a |
        E -> b | DD | EE | CC | a |
        D -> b | DD | EE | CC | a |
        S1 -> b | DE | DD | EE | CC | CE | DCE | a | [empty] | DC |
S: S1
消去无用符号
N: {E, D, S1, C, }
T: {a, b, }
P:
        C -> b | DD | EE | CC | a |
        E -> b | DD | EE | CC | a |
        D -> b | DD | EE | CC | a |
        S1 -> b | DE | DD | EE | CC | CE | DCE | a | [empty] | DC |
S: S1
```
