北京邮电大学
计算机学院(国家示范性软件学院)
2022-2023 学年第 2 学期
实验项目总结报告

课程名称：形式语言与自动机

项目名称：设计上下文无关文法的变换算法

项目完成人：

负责人姓名：张梓良 学号：2021212484

姓名：张梓良 学号：2021212484

姓名：苗雨 学号：2021212492

姓名：杨晨 学号：2021212171

指导教师：杨正球 助教：杨惠元、句康

日 期： 2023 年 6 月 13 日

目录

一 ．实验目的和要求

实验目的

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。

实验要求

1.采用分组实验，每组学生 3~4 人，班内自由组合，培养学生团队合作能力。

2.编程语言要求使用 C/C++，Java，需要使用头歌教学云平台进行测试验证代码。

3.要求程序运行正确，设计风格好，文档描述清晰，并且按期提交小组实验报告，个人实验报告，源代码，及可执行程序。文件命名方式：组长班级+组长姓名+文件类型（报告/代码/程序）。三个文件打包提交，命名方式：实验二+组长班级+组长姓名。

4.实验报告至少包含以下内容：

（1）小组成员，班级，姓名，学号；成员分工。

（2）实验环境描述：所使用的语言等。

（3）程序的设计思路及核心算法。

（4）程序的输入格式，输出格式。

（5）程序的测试用例，输入，输出，以及执行效果（可截图）

（6）改进思路和方法（可选）

二 .实验环境

Windows10 专业版 19041.264

Visual Studio Code 1.79.1

C++20

三 .实验需求

问题描述

程序的输入输出格式：

1.输入格式

输入是一个上下文无关文法，按顺序写文法的 N，T，P，S。空产生式用 N 表示。

第 1 行:非终结符集合 N={A，B，C}

第 2 行:终结符集合 T={a,b,c}

第 3 行: P:

第 4 行开始:每行开始有一个\t

A-->aB|b

B-->Ac

C-->a

S-->A

最后一行: 起始符号 S=S

2.输出格式

输出是与该文法等价的没有 ε 产生式、单产生式、无用符号的上下文无关文法。格式同输入格式。排序统一按照 ascii 码由小到大排序（产生式左部和右部都按照这个顺序），如果原输入的开始符号 S 能推导出空产生式导致需要生成新的非终结符，该非终结符用 T 表示。

样例输入

N={A,B,C,D,S}

T={a,b,c,d}

P:

S-->B|a|bA|ccD

A-->N|abB

B-->aCA

C-->ddC

D-->ddd

S=S

样例输出

N={A,B,D,S}

T={a,b,c,d}

P：

A-->abB

B-->a|aA

D-->ddd

S-->a|aA|b|bA|ccD

S=S

四 .程序设计思路及核心算法

程序设计思路

1. 输入：

输入是通过 read_input 函数实现的。输入内容为一个名为 input.txt 的文件。该文件中包含非终结符集合、终结符集合、产生式集合和起始符号。

在该函数中，首先通过 getline 函数从输入中读取第一行，即非终结符集合，然后解析出其中的每个非终结符，并将其加入集合 N 中。

接着读取第二行，即终结符集合，解析出其中的每个终结符，并将其加入集合 T 中。然后跳过第三行，开始解析产生式集合。通过 while 循环逐行读取输入，对于每一行，首先解析出产生式左部的非终结符，然后解析出右部的每个可能性，并将它们作为 vector<char> 类型的元素加入到一个 vector<vector<char>> 中，最后将该产生式加入到 P 中。

最后解析出起始符号 S，并返回读取到的所有信息。

2. 对上下文无关文法进行变换:

消除 ε 产生式的过程如下：

1.找到所有可以直接推出 ε 的非终结符；

2.不断找到新的可以推出 ε 的非终结符，直到集合不再变化；

3.根据新的产生式集合生成新的产生式集合 P1。

消除单产生式的过程如下：

1.遍历原始产生式集合 P 中的每个非终结符 A；

2.找出所有可以直接推出单个非终结符的产生式，并将这些非终结符加入 new_nonterminals 中；

3.不断扩展 new_nonterminals，直到 new_nonterminals 不再变化为止；

4.构建新的产生式列表；

5.去除重复的产生式，如果新的产生式列表为空，则添加一个产生式 A -> N。

消除无用符号的过程如下：

1.找出所有能直接推导出终结符的非终结符，并将其加入 N1 集合中；

2.不断扩展 N1 集合，直到 N1 不再发生变化；

3.从原始产生式集合 P 中删除所有左部非终结符不在 N1 集合中的产生式；

4.反向遍历所有的“可达性”非终结符，并将它们加入到新的非终结符集合 new_nonterminals 中；

5. 同样从原始产生式集合 P 中删除所有左部非终结符不在 new_nonterminals 集合中的产生式。

3. 输出:

输出是通过 print_grammar 函数实现的。

首先输出非终结符集合 N 和终结符集合 T ，并通过循环遍历输出各个元素，其中使用了迭代器并判断是否为最后一个元素来避免末尾多余的逗号。然后按照 key 对产生式集合 P 中的元素进行排序，并依次输出左部符号和右部符号，其中右部符号按照字典序排序，同样使用了迭代器并判断是否为最后一个元素来避免末尾多余的竖线。最后输出开始符号 S 即可。

需要注意的是，这里使用了一些 C++11 中新增的语法特性，比如 auto 用于获取迭代器类型和范围 for 循环，还有 lambda 表达式和 map 排序等。

核心算法

1.消除 ε 产生式:

（1）首先遍历产生式集合 P 中的每个产生式，找出所有可以直接推出空串的非终结符，加入集合 eps_generating_nonterminals。具体方法是:对每个非终结符 key，遍历它对应的产生式列表 value，如果 value 中有产生式右部只包含一个'N'的话，即能直接推出空串，则将 key 加入 eps_generating_nonterminals 集合。

（2）然后不断扩展 eps_generating_nonterminals 集合，直到不再有新的可以推出空串的非终结符加入。扩展方法是:对每个非终结符 key，遍历它对应的产生式列表 value，如果 value 中所有的符号都在 eps_generating_nonterminals 集合中，则 key 也可以推出空串，将其加入 eps_generating_nonterminals。

（3）根据 eps_generating_nonterminals 集合构建新的产生式集合 P1。对每个非终结符 key，遍历它对应的产生式列表 value，对 value 中的每个产生式:

- 计算其中可以推出空串的非终结符的个数 total_in_nonterminals
- 枚举所有这些非终结符的组合
- 对于每个组合:
  - 生成新的不含空串的产生式
  - 如果组合中包含该非终结符，则保留该非终结符
  - 否则舍去该非终结符
- 加入 P1

（4）如果原始起始符号 S 可以推出空串，则选择 T 作为新的起始符号，同时为 P1 添加一个新的产生式 T-->S。

（5）P1 代替原始的产生式集合 P。

2.消除单产生式:

（1）定义新的非终结符集合 new_nonterminals、新的产生式集合 new_productions 和新的起始符号 new_start。

（2）遍历每个非终结符 A，找出所有可以从 A 直接推出单个非终结符的产生式，将这些非终结符加入 new_nonterminals 集合。具体是：遍历 A 对应的产生式列表，如果存在产生式的右部只有一个非终结符 B，则将 B 加入 new_nonterminals。

（3）不断扩展 new_nonterminals 集合，直到不再有新的非终结符加入。扩展方法是：对 new_nonterminals 中的每个非终结符 B，找出从 B 可以直接推出单个非终结符的产生式，并将该非终结符加入 new_nonterminals。

（4）构建新的产生式列表 new_productions[A]。对 new_nonterminals 中的每个非终结符 B，将从 B 推出的非单产生式的产生式加入 new_productions[A]。如果 new_productions[A]为空，添加 A-->N 作为默认产生式, 以免删除 A 后影响文法的语言。

（5）如果 A 是原始起始符号 S，则更新 new_start 为 S(如果 S 在 new_nonterminals 集合中或 new_productions 的第一个 key)，该步保证新的文法有唯一的起始符号。

（6）将新的产生式集合 new_productions 替换原始产生式集合 P。

3.消除无用符号:

（1）找出只能推出终结符的非终结符，即无用非终结符，加入集合 N1。具体方法是：遍历每个非终结符 A，如果 A 对应的所有产生式的右部都只包含终结符，则 A 加入 N1。

（2）不断扩展 N1，直到不再有新的非终结符加入。扩展方法是：对 N1 中的每个非终结符 B，找出只能从 B 推出 N1 中的非终结符的产生式，则 B 也是无用非终结符,将 B 加入 N1。

（3）更新非终结符集合 N 和终结符集合 T，删除 N1 中的无用非终结符和终结符。

（4）创建新的产生式集合 P2。遍历原始产生式集合 P，删除含有无用非终结符的产生式，剩余的产生式加入 P2。

（5）令新的产生式集合 P = P2。

（6）删除无用符号后，需要检查新的文法是否还存在单产生式，如果存在需要继续消除单产生式。

五 .测试及执行效果

测试用例

输入	**输出**
N={A,B,C,D,S}	N={A,B,C,S}
T={a,b,c,d,e}	T={a,b,c,d}
P:	P:
S-->C|aA|aB	A-->b|bB|cc
A-->bB|cc	B-->d
B-->N|d	C-->c|ddC
C-->c|ddC	S-->a|aA|aB|c|ddC
D-->e|eDe	S=S
S=S	

用例解释

1. 消除 ε 产生式：

先找出能推出空串的非终结符的集合，结果为{B},考察所有右侧含 B 的产生式：

S-->aB　　A-->bB

将其在新的产生式集合中替换为 S-->a|aB A-->b|bB，并删去 B-->N，得到文法:

N={A,B,C,D,S}

T={a,b,c,d,e}

P:

S-->C|aA|a|aB

A-->b|bB|cc

B-->d

C-->c|ddC

D-->e|eDe

S=S

2. 消除单生成式：

对于每个非终结符，计算得 N_S={S,C},N_A={A},N_B={B},N_C={C},N_D={D};

故在产生式集合 P 中删去 S-->C,加入 S-->c|ddC，得到文法：

N={A,B,C,D,S}

T={a,b,c,d,e}

P:

S-->aA|a|aB|c|ddC

A-->b|bB|cc

B-->d

C-->c|ddC

D-->e|eDe

S=S

3. 删除无用符号：

由教材算法 1，推得有用的非终结符集合为 N={A,B,C,D,S},再利用算法二从 S 出发推到有用符号集合 N':

由 S-->C|aA|a|aB|c|ddC，N'={S,A,B,C,a,c,d};

由 A-->b|bB|cc，N'={S,A,B,C,a,b,c,d};

由 B-->d，N'={S,A,B,C,a,b,c,d};

由 C-->c|ddC，N'={S,A,B,C,a,b,c,d};

N_1={S,A,B,C}, T_1={a,b,c,d},删去了无用符号{e,D},删除对应产生式：D-->e|eDe，得到文法：

N={A,B,C,S}

T={a,b,c,d}

P:

S-->aA|a|aB|c|ddC

A-->a|b|bB|cc

B-->d

C-->c|ddC

S=S

将形式规格化后，得到应输出的转换后的文法:

N={A,B,C,S}

T={a,b,c,d}

P:

A-->b|bB|cc

B-->d

C-->c|ddC

S-->a|aA|aB|c|ddC

S=S

执行效果

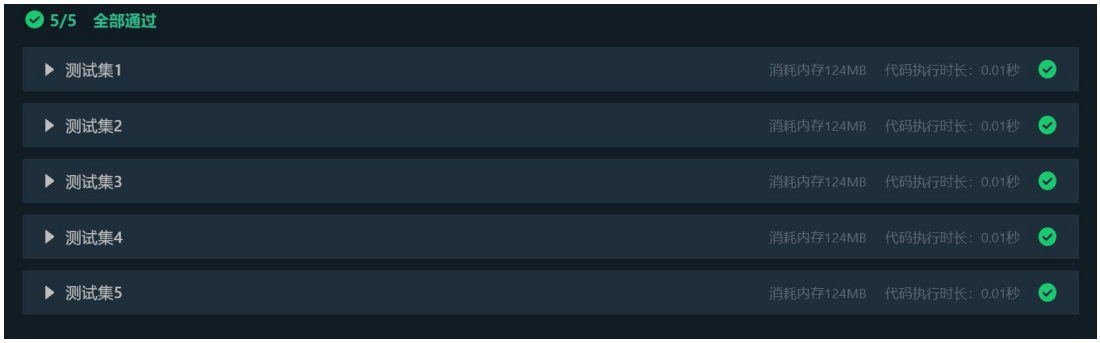
1. 测试用例：

成功地将输入的上下文无关文法进行消除 ε 产生式、单产生式、无用符号的处理。

```
PS D:\develop\vscode> cd 'd:\develop\vscode\output'
PS D:\develop\vscode\output> & .\'CFLTransform.exe'
N={A,B,C,D,S}
T={a,b,c,d,e}
P:
        S-->C|aA|aB
        A-->bB|cc
        B-->N|d
        C-->c|ddC
        D-->e|eDe
S=S
N={A,B,C,S}
T={a,b,c,d}
P:
        A-->b|bB|cc
        B-->d
        C-->c|ddC
        S-->a|aA|aB|c|ddC
S=S
PS D:\develop\vscode\output>
```

2. 头歌云平台测试效果：

测试点全部通过


5/5 全部通过
测试集1
消耗内存124MB 代码执行时长：0.01秒
测试集2
消耗内存124MB 代码执行时长：0.01秒
测试集3
消耗内存124MB 代码执行时长：0.01秒
测试集4
消耗内存124MB 代码执行时长：0.01秒
测试集5
消耗内存124MB 代码执行时长：0.01秒

六 .实验小组成员分工

程序编写：杨晨

实验报告撰写：苗雨、张梓良

七．改进的思路和方法（可选）

1. 可以完善文法等价性的判断，保证变换后文法的语言生成能力不变。
2. 可以增加其他文法变换，如消除左递归、标准化文法等。
3. 可以对输入的文法进行预处理，过滤无用的产生式和符号。
4. 可以增加可视化的输出，以图形化显示文法变换的过程。
5. 可以增加对更复杂文法的支持，目前程序仅支持简单的右线性文法。
6. 可以增加异常处理，以处理输入文法格式不正确的情况。