### 自动机lab2实验报告

#### ① 小组成员姓名学号:

#### ② 实验环境描述：

- \*\*编程语言\*\*：Python 3

- \*\*开发工具\*\*：VSCode

- \*\*操作系统\*\*：Windows 11

- \*\*依赖库\*\*：无特别依赖，仅使用Python标准库

#### ③ 程序的设计思路及核心算法：

\*\*设计思路\*\*：

1. \*\*上下文无关文法（CFG）处理\*\*：

- 构建一个CFG类来表示上下文无关文法，并实现消除ε产生式、单产生式及无用符号的算法。

- 主要方法包括：`remove\_epsilon`、`remove\_unit\_productions`、`remove\_useless\_symbols`，每个方法处理一种类型的产生式。

2. \*\*下推自动机（PDA）到等价CFG的转换\*\*：

- 构建一个PDAtoCFG类，将下推自动机的定义转换为等价的上下文无关文法。

- 通过解析PDA的状态、输入字母表、栈字母表和转移函数，生成相应的CFG产生式。

\*\*核心算法\*\*：

1. \*\*消除ε产生式\*\*：

- 查找包含ε产生式的非终结符号。

- 删除ε产生式并生成新的替代产生式。

2. \*\*消除单产生式\*\*：

- 查找单产生式。

- 用单产生式右侧的产生式替代单产生式。

3. \*\*消除无用符号\*\*：

- 查找所有有用的非终结符号。

- 移除无用符号及其产生式。

4. \*\*PDA到CFG转换\*\*：

- 根据PDA的转移函数生成等价的CFG产生式。

#### ④ 程序的输入格式，输出格式：

\*\*输入格式\*\*：

1. \*\*CFG\*\*：

- 用户按提示输入产生式的数量、左部非终结符号和右部产生式。

- 示例输入：

```

Enter number of productions: 5

Enter left hand side (non-terminal): S

Enter right hand side (productions, separated by '|'): a|bA|B|ccD

Enter left hand side (non-terminal): A

Enter right hand side (productions, separated by '|'): abB|ε

Enter left hand side (non-terminal): B

Enter right hand side (productions, separated by '|'): aA

Enter left hand side (non-terminal): C

Enter right hand side (productions, separated by '|'): ddC

Enter left hand side (non-terminal): D

Enter right hand side (productions, separated by '|'): ddd

```

2. \*\*PDA\*\*：

- 用户按提示输入状态集、输入字母表、栈字母表、转移函数、初始状态、初始栈符号和接受状态。

- 示例输入：

```

Enter states (separated by space): q0 q1

Enter input alphabet (separated by space): a b

Enter stack alphabet (separated by space): B z0

Enter number of transitions: 6

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q0,b,z0,q0,Bz0

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q0,b,B,q0,BB

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q0,a,B,q1,ε

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q1,a,B,q1,ε

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q1,ε,B,q1,ε

Enter transition (format: current\_state,input\_symbol,stack\_symbol,next\_state,new\_stack\_symbol\_sequence): q1,ε,z0,q1,ε

Enter start state: q0

Enter start stack symbol: z0

Enter final state: q1

```

\*\*输出格式\*\*：

1. \*\*CFG\*\*：

- 显示经过转换后的CFG产生式。

- 示例输出：

```

Transformed CFG productions:

S -> a | bA | B | ccD

A -> abB

B -> aA

C -> ddC

D -> ddd

```

2. \*\*PDA\*\*：

- 显示转换后的等价CFG产生式。

- 示例输出：

```

Equivalent CFG productions:

q0z0q1 -> bBq1 | ε

q0Bq1 -> bBq1 | aε

q1Bq1 -> aε | ε

```

#### ⑤ 程序的测试用例，输入，输出，以及执行效果（可截图）：

截图

#### ⑥ 改进思路和方法（可选）：

1. \*\*优化算法效率\*\*：

- 目前的算法是逐步处理产生式的，可考虑使用更加高效的数据结构和算法来提高处理效率。

2. \*\*用户输入的校验\*\*：

- 增加对用户输入的格式校验，确保输入的合法性和正确性，减少错误的发生。

3. \*\*图形化用户界面（GUI）\*\*：

- 开发一个图形化用户界面，使得输入和输出更加直观和易用，适合更多用户使用。

4. \*\*扩展功能\*\*：

- 增加更多上下文无关文法和下推自动机的操作，如自动化测试、生成特定类型的文法等，增强程序的功能性和实用性。

通过以上改进，可以使得程序更加高效、易用，并且功能更加丰富。  
#### ⑦ 心得体会：