# 词法分析

## NFA转DFA：子集法

## DFA化简

书：分割法

编程：（深度优先、）广度优先

## 正规文法转NFA：略

|  |  |
| --- | --- |
|  | 转换函数 |
|  |  |
|  |  |

## NFA转正规文法：略

|  |  |
| --- | --- |
|  | 产生式 |
|  |  |
| 可接受状态 |  |

## 正规式转正规文法

|  |  |
| --- | --- |
| 正规式产生式 | 替换为 |
|  |  |
|  |  |
|  |  |

## 正规文法转正规式

结点：非终结符、右侧没有非终结符的产生式。

边：右侧有非终结符的产生式。

|  |  |
| --- | --- |
| 产生式 | 正规式产生式 |
|  |  |
|  |  |
|  |  |

用规则1消掉所有自环：n1个A→有A用规则3合并成1个A→有A。n2个A→没有A。用规则1，结果是n2个新的A→没有A。

对每个结点：

　　用规则2去掉这个结点

　　用规则1消掉所有自环

## 正规式转NFA：略

<https://en.wikipedia.org/wiki/Thompson%27s_construction>

## NFA转正规式

<https://cs.stackexchange.com/questions/2016/how-to-convert-finite-automata-to-regular-expressions>

多个开始状态的改成1个开始状态，多个结束状态的改成1个结束状态。

对每个结点对(A,B)：

　　若A==B：自环{ε}

　　若A!=B：没边就是∅，有边要用“|”（∪）把重边合并

对每个结点A，A!=开始状态，A!=结束状态：

　　对每个结点对(B!=A,C!=A)：

　　　　边B→C |=（∪=） 边B→A·(边A→A)\*·边A→C

　　　　边B→B同理

　　　　边C→B同理

　　　　边C→C同理

# 语法分析

## LL（1）文法的判别

假设文法中不包含多余规则和有害规则。

①判断每个非终结符是否能推出ε

对每个产生式A→α：

　　若α==ε：

　　　　A能推出空串，A入队

　　　　（所有A→…不再有用）删除所有A→…

每个vT入队

当队非空：

　　v=出队

　　若v是非终结符不能推出空串/v是终结符：

　　　　对每个产生式A→α：

　　　　　　若A→α右侧有v：

　　　　　　　　（A→α不可能推出空串）删除A→α

　　　　　　　　若A的（可能推出空串的）产生式数量降为0：

　　　　　　　　　　A不能推出空串，A入队

　　若v是非终结符能推出空串：

　　　　对每个产生式A→α：

　　　　　　若v==A：

　　　　　　　　（A→α不再有用）删除A→α

　　　　　　若v!=A：

　　　　　　　　删除α中的v

　　　　　　　　若α==ε：

　　　　　　　　　　A能推出空串，A入队

　　　　　　　　　　（所有A→…不再有用）删除所有A→…

②计算每个产生式右部的FIRST集

根据定义计算、由关系图法计算

计算每个非终结符（、终结符）的FIRST集

|  |
| --- |
| 结点：非终结符、终结符  边：刚开始没有边。u→v表示v的开始符号u也要有。  对每个产生式：  　　若产生式是A→至少0个[能推出空串的非终结符] [不能推出空串的非终结符/终结符]…：  　　　　添加边：A→那几个非终结符  　　　　添加边：A→那个终结符  　　若产生式是A→至少0个[能推出空串的非终结符]：  　　　　添加边：A→那几个非终结符  对每个非终结符A：  　　若第①步结果中A能推出空串：  　　　　ε加入A的FIRST集中 |

计算每个产生式右部的FIRST集

|  |
| --- |
| 若产生式右部是 至少0个[能推出空串的非终结符] [不能推出空串的非终结符/终结符]…：  　　那几个非终结符的(FIRST集-{ε})∪那个终结符的FIRST集  若产生式右部是 至少0个[能推出空串的非终结符]：  　　那几个非终结符的FIRST集∪{ε} |

③计算每个非终结符的FOLLOW集