LR(1)语法分析器

----基于CFG

软件学院

陈睿 141250013

141250013@smail.nju.edu.cn

2016.11.15

目录

[1. 目标 3](#_Toc465532711)

[2. 内容描述 3](#_Toc465532712)

[3. 构思与方法 3](#_Toc465532713)

[4. 假设 4](#_Toc465532714)

[4.1. 实验环境 4](#_Toc465532715)

[4.2. 输入及符号假设 4](#_Toc465532716)

[5. 有关自动机描述 4](#_Toc465532717)

[6. 数据结构描述 4](#_Toc465532718)

[7. 核心算法描述 6](#_Toc465532719)

[7.1. RE->NFA 6](#_Toc465532720)

[7.2. NFA->DFA 8](#_Toc465532721)

[8. 测试用例 9](#_Toc465532722)

[8.1. Test 1 9](#_Toc465532723)

[8.2. Test 2 11](#_Toc465532724)

[9. 问题与解决 12](#_Toc465532725)

[10. 总结与感受 13](#_Toc465532726)

1. 目标

主要目标是尝试进行基于上下文无关文法(CFG)的LR(1)语法生成器的构造，通过自底向上方式完成对输入串的分析，使我可以更深入理解龙书上相关算法，并自己通过代码进行实现，同时也锻炼一下自己的能力。

1. 内容描述

本报告主要描述了自底向上语法分析中LR(1)的建表、分析过程，主要包括:由CFGs构造出LR(1)分析表，再依据LR(1)分析表和输入串给出具体的分析过程这2个部分内容，以及以上2部分的构思方法、具体实现过程。

1. 构思与方法

从整体思路来看，考虑到自底向上方法主要仍然是构造相关的分析表，然后依据分析表再去进行分析，因此需要先将输入的上下文无关文法进行扩展、规范化，然后建立LR项，构建分层有限状态自动机，分析状态间扩展和状态内扩展，生成所有状态和对应转换关系，建立LR(1)分析表，最后进行分析。建立借鉴龙书上相关算法，结合一部分自己所学知识即可。

具体来看，代码部分有如下步骤：

1. 输入上下文无关文法（CFG）和需要分析的串（sequence）
2. 对输入的CFG进行扩展，加入S’->S
3. 建立初始LR项
4. 构造LR分析表
5. 分析sequence
6. 输出具体移入、归约过程
7. 假设
   1. 实验环境
8. Mac OS
9. Idea + javac
   1. 输入串约定
   2. 输入CFG约定
   3. 其他符号约定
10. 有关自动机描述

使用分层有限状态自动机，用于构造LR(1)分析表，主要依据龙书上LR(1)项集族构造方法（算法4.53）和规范LR(1)语法分析表的构造算法（算法4.56）

1. 数据结构描述
2. 核心算法描述
   1. 生成LR(1)分析表
   2. 根据LR(1)分析表分析输入串
3. 测试用例
   1. Test 1
4. 测试输入
5. 测试输出
   1. Test 2
6. 测试输入
7. 测试输出
8. 问题与解决
9. 后缀表达式转NFA时，如何记录中间产生的各个状态。

解决：实际上，从语法分析树可以看到，最多只会有2个当前状态集，并且当前状态集实际上只有start状态和end状态是会被后续转换所用到的，于是只需要用4个变量表示即可。如我代码中：

1. NFA转DFA时，如何记录当前状态集所对应的状态号？因为没有建立对应状态集的数据结构，状态集实际上也是一个状态？

解决：采用多一个`STL<vector>`的方式，跟踪记录相应状态号。

1. 总结与感受

通过这次实验，我感觉复习了一些以前学过的算法，如BFS，中缀转后缀等。同时感受到了词法分析器生成器的有趣之处，希望有空可以实现一个自己的简单分析生成器。

没有了一开始看到lex生成出一个分析代码的模板出来那种震惊了，感觉自己也可以实现，开心~