BJTU编译原理lab3实验报告

# 编译原理Lab03: LL(1)分析

22281089　陈可致


- 编译原理Lab03: LL(1)分析
  - 22281089　陈可致
  - 一. 实验要求
  - 二. 开发环境
  - 三. 运行方式
  - 四. 项目概述
  - 五. 程序设计概述
  - 六. 程序设计
  - 七. 测试
  - 八. 心得体会
  - 附录


## 一. 实验要求

### 1. 实验项目

以专题 1 词法分析程序的输出为语法分析的输入, 实现 LL(1)分析中控制程序(表驱动程序), 完成以下描述赋值语句的 LL(1)文法的 LL(1)分析过程

```
G[S]:S→V=E
E→TE’
E’→ATE’|ε
T→FT’
T’→MFT’|ε
F→ (E)|i
A→+|-
M→*|/
V→i
```

### 2. 设计说明

- 终结符号 i 为用户定义的简单变量, 即标识符的定义

### 3. 设计要求

- 构造该文法的 LL(1)分析表
- 输入串应是词法分析的输出二元式序列, 即某算术表达式”专题 1”的输出结果 输出为输入串是否为该文法定义的算术表达式的判断结果
- LL(1)分析过程应能发现输入串出错
- 设计至少四个测试用例(尽可能完备, 正确和出错), 并给出测试结果
- 选做：考虑根据 LL(1)文法编写程序构造 LL(1)分析表, 包括 FIRST 集合和 FOLLOW 集合, 并添加到你的 LL(1)分析程序中

## 二. 开发环境

- Ubuntu 24.04.1 LTS
- g++ (Ubuntu 13.3.0-6ubuntu2~24.04) 13.3.0

## 三. 运行方式

1. 需要 g++ 编译器, 没有可以用以下命令安装

```
sudo apt update
sudo apt install g++
```

2. 对于每个实验, 都编写了 sh脚本 和 测试数据文件 用于测试项目, 只需要在项目文件夹目录下运行.sh文件即可进行测试

```
cd your_file_forder
bash go_st.sh
```

```
cd your_file_forder
bash go_LL.sh
```

## 四. 项目概述

本实验设计并实现了一个递归下降语法分析程序, 用于解析基于LL(1)文法的赋值语句程序的输入是词法分析程序的输出二元式序列, 输出是输入串是否为文法定义的算术表达式的判断结果该程序能够发现简单的语法错误, 并提供详细的错误信息

## 五. 程序设计概述

- LL solver

```
class lycoris {
   public:
    lycoris() {}
    lycoris(const grammar &g) : G(g) {}
    meion check(const string &s) -> bool {}
    meion show_st() -> void {}
   private:
    grammar G;
    hash_map<hash_map<production>> st;
    token_solver t_sol;
    grammar_solver g_sol;
    meion build() -> void {}
};
```

## 六. 程序设计

1. 项目结构

   项目分为几个模块

   - Lib: 头文件
     1. MeloN_H.hpp: 用到的标准库头文件. 使用的stl容器, 宏定义
     2. MeloN_debug.hpp: 调试头文件, 用于格式化输出不定参数的变量信息, 标准运行环境下 **不会** 生效
     3. 2_LL_solver.hpp: 定义了LL::lycoris类, 用于LL(1)分析, 提供了一个方法用于测试
   - testcase 测试数据 | std
     1. 4组测试数据 (in0 - in3)
     2. 4组对应的标准输出 (std0 - std3)
   - testcase_st 测试数据2 | std
     1. 标准输出 std
   - 测试程序
     1. test_st.cpp: 用于测试LL::lycoris类
     2. go_st.sh: 用于测试项目的脚本
     3. test_LL.cpp: 也是用于测试LL::lycoris类
     4. go_LL.sh: 用于测试项目的脚本
2. LL::lycoris类
   1. 构造函数:
      - lycoris(): 默认构造函数, 创建一个空对象, 不做任何初始化

- lycoris(const grammar &g): 接收一个文法 g 并初始化对象, 执行以下操作:
  - 设置语法求解器的文法
  - 计算 FIRST 集合
  - 计算 FOLLOW 集合, 以起始符 "S" 为目标
  - 调用 build() 方法构建预测分析表（ST 表）

2. 公有方法

- meion check(const string &s): 解析输入字符串, 判断其是否符合文法定义 使用LL(1)分析表驱动的语法分析过程, 通过栈操作和表查找进行匹配
- meion show_st(): 按顺序输出预测分析表（ST 表）的内容

3. 私有方法

- meion build(): 构造LL(1)分析表 对于每个非终结符号及其产生式, 计算First集和Follow集, 并将相应的产生式填入分析表

3. LL::test(): 测试LL::lycoris类的函数

## 七. 测试

- **测试用例**
  - in0:

```
a = b + c * (d - e / f) + g - h * (i + j / k
```

  - in1:

```
a = b * (c + d) / e - f / g
```

  - in2:

```
a = b + c * (d - e / f) + g - h * i + j / k)
```

  - in3:

```
a = b + c /
```

- **std**
  - std0:

```
Ciallo: a = b + c * (d - e / f) + g - h * (i + j / k
Zako♡~ exactly invalid
```

  - std1:

```
Ciallo: a = b * (c + d) / e - f / g
valid
```

  - std2:

```
Ciallo: a = b + c * (d - e / f) + g - h * i + j / k)
Zako♡~ exactly invalid
```

  - std3:

```
Ciallo: a = b + c /
Zako♡~ exactly invalid
```

  - std(st)

```
ST table
A : {
+ -> +
- -> -
}
E : {
```

```
( -> T E'
identifier -> T E'
}
E' : {
) ->
+ -> A T E'
- -> A T E'
end ->
}
F : {
( -> ( E )
identifier -> i
}
M : {
* -> *
/ -> /
}
S : {
identifier -> V = E
}
T : {
( -> F T'
identifier -> F T'
}
T' : {
) ->
* -> M F T'
+ ->
- ->
/ -> M F T'
end ->
}
V : {
identifier -> i
}

```

- **测试结果**
  - 结果正确 (对于输入invalid表达式的testcase存在一些std::cerr输出的分析信息) ~~要看程序输出的话可以把sh脚本中删除输出文件的语句注释~~

```
bash go_st.sh
check st:
accept

bash go_LL.sh
Zako♡~ token match failed
test: 0
accept
test: 1
accept
test: 2
accept
Honto ni yowai'♡~ unexpected token #
test: 3
accept
```

## 八. 心得体会

大模拟写写写 zzz

debug过程显著提高了抗压能力

## 附录

1. MeloN_H.hpp

```
#pragma once
#include <algorithm>
#include <array>
#include <bitset>
#include <cassert>
#include <cctype>
#include <chrono>
#include <cmath>
#include <cstring>
#include <ctime>
#include <fstream>
#include <functional>
#include <iomanip>
#include <iostream>
#include <limits>
#include <map>
#include <queue>
#include <random>
#include <ranges>
#include <set>
#include <stack>
#include <string>
#include <tuple>
#include <unordered_map>
#include <unordered_set>

using   std::array, std::bitset, std::deque, std::greater, std::less, std::map,
        std::multiset, std::pair, std::priority_queue, std::set, std::stack,
        std::string, std::vector, std::tuple, std::function;

using NAME = void;       using uint = unsigned;   using ll = long long;      using ull = unsigned long long;
using ld = long double;  using i128 = __int128_t; using u128 = __uint128_t;  using f128 = __float128;

#define meion     auto
#define iroha     return
```

1. MeloN_debug.hpp

```
#pragma once

template <class T, size_t size = std::tuple_size<T>::value>
std::string to_debug(T, std::string s = "")
    requires(not std::ranges::range<T>);
std::string to_debug(meion x)
    requires requires(std::ostream& os) { os << x; }
{
    iroha static_cast<std::ostringstream>(std::ostringstream() << x).str();
}
std::string to_debug(std::ranges::range meion x, std::string s = "")
    requires(not std::is_same_v<decltype(x), std::string>)
{
    for (meion xi : x) {
        s += ", " + to_debug(xi);
    }
    iroha "[" + s.substr(s.empty() ? 0 : 2) + "]";
}
```

```
    template <class T, size_t size>
    std::string to_debug(T x, std::string s)
        requires(not std::ranges::range<T>)
    {
        [&]<size_t... I>(std::index_sequence<I...>) {
            ((s += ", " + to_debug(std::get<I>(x))), ...);
        }(std::make_index_sequence<size>());
        iroha "(" + s.substr(s.empty() ? 0 : 2) + ")";
    }
    #ifdef MeIoN
    #define debug(...)  std::cout << "Ciallo～(∠・ω< )⌒★ " << "(" #__VA_ARGS__ ") = " << to_debug(std::tuple(
    #else
    #define debug(...) void(0721)
    #endif
```

1. 2_LL_solver.hpp

```
    #pragma once
    #include "1_grammar_solver.hpp"

    namespace LL {
        using grammar_solver = n_grammar_solver::lycoris;
        using n_grammar_solver::grammar;
        using n_grammar_solver::production;
        using n_grammar_solver::token_solver;
        class lycoris {
        public:
            lycoris() {}
            lycoris(const grammar &g) : G(g) {
                t_sol.build("");
                g_sol.set_grammar(G);
                g_sol.compute_first();
                g_sol.compute_follow("S");
                build();
            }
            meion check(const string &s) -> bool {
                vector tokens = t_sol.get_tokens(s);
                vector<string> stk{"S"};
                int pla = 0;
                while (not stk.empty()) {
                    string t = stk.back();
                    meion token = tokens[pla];
                    if (g_sol.is_non_terminal(t)) {
                        if (st[H(t)].contains(token.type)) {
                            stk.pop_back();
                            const production &prod = st[H(t)][token.type];
                            for (const meion &x : std::views::reverse(prod.Rs())) {
                                if (not x.empty()) {
                                    stk.emplace_back(x);
                                }
                            }
                        } else {
                            iroha std::cerr << "Honto ni yowai'♡~ unexpected token "
                                            << token.value << std::endl,
                                false;
                        }
                    } else if (t_sol.get_token_type(t) == token.type) {
                        stk.pop_back();
                        ++pla;
                    } else {
                        iroha std::cerr << "Zako♡~ token match failed"
```

```
                                    << std::endl,
                            false;
                    }
                }
                assert(tokens.back().type == End);
                iroha pla + 1 == tokens.size();
            }
            meion show_st() -> void {
                std::cout << "ST table" << '\n';
                vector<string> out;
                st.view(
                    [&](const ull &key, const hash_map<production> &val) -> void {
                        out.emplace_back();
                        meion &s = out.back();
                        s += HINA[key] + " : {\n";
                        vector<string> pv;
                        val.view([&](const meion &k, const production &v) -> void {
                            pv.emplace_back();
                            pv.back() += "  " + type_to_s[k] + " -> ";
                            for (const meion &x : v.Rs()) {
                                pv.back() += x + " ";
                            }
                            pv.back() += "\n";
                        });
                        std::ranges::sort(pv);
                        for (const string &x : pv) {
                            s += x;
                        }
                        s += "}\n";
                    });
                std::ranges::sort(out);
                for (const string &s : out) {
                    std::cout << s;
                }
                std::cout.flush();
            }
        private:
            grammar G;
            hash_map<hash_map<production>> st;
            token_solver t_sol;
            grammar_solver g_sol;
            meion build() -> void {
                G.enumerate_all([&](const ull &key, const vector<production> &val) -> void {
                    for (const production &prod : val) {
                        meion f = g_sol.get_first(prod);
                        for (const meion &tok : f) {
                            if (tok != Epsilon) {
                                st[key][tok] = prod;
                            }
                        }
                        if (f.contains(Epsilon)) {
                            const meion &follow = g_sol.get_follow(key);
                            for (const meion &tok : follow) {
                                st[key][tok] = prod;
                            }
                        }
                    }

                });
            }
        };

```

```
107         meion test_set() -> grammar {
108             grammar G;
109             G[H("S")] = {{"S", {"V", "=", "E"}}};
110             G[H("E")] = {{"E", {"T", "E'"}}};
111             G[H("E'")] = {{"E'", {"A", "T", "E'"}}, {"E'", {""}}};
112             G[H("T")] = {{"T", {"F", "T'"}}};
113             G[H("T'")] = {{"T'", {"M", "F", "T'"}}, {"T'", {""}}};
114             G[H("F")] = {{"F", {"(", "E", ")"}}, {"F", {"i"}}};
115             G[H("A")] = {{"A", {"+"}}, {"A", {"-"}}};
116             G[H("M")] = {{"M", {"*"}}, {"M", {"/"}}};
117             G[H("V")] = {{"V", {"i"}}};
118             return G;
119         }
120 
121         meion test_st() -> void {
122             lycoris chisato(test_set());
123             chisato.show_st();
124         }
125 
126         meion test_rda() -> void {
127             lycoris chisato(test_set());
128             string s, t;
129             while (std::getline(std::cin, s)) {
130                 t += s + '\n';
131             }
132             if (not t.empty()) t.pop_back();
133             std::cout << "Ciallo: " << t << std::endl;
134 
135             if (chisato.check(t)) {
136                 std::cout << "valid" << std::endl;
137             } else {
138                 std::cout << "Zako♡~ exactly invalid" << std::endl;
139             }
140         }
141     } // namespace LL
```