吉林大学学士学位论文（设计）承诺书

本人郑重承诺：所呈交的学士学位毕业论文（设计），是本人在指导教师的指导下，独立进行实验、设计、调研等工作基础上取得的成果。除文中已经注明引用的内容外，本论文（设计）不包含任何其他个人或集体已经发表或撰写的作品成果。对本人实验或设计中做出重要贡献的个人或集体，均已在文中以明确的方式注明。本人完全意识到本承诺书的法律结果由本人承担。

学士学位论文（设计）作者签名：

2018 年 5 月 20 日

摘要

中文标题

中文摘要。

关键字：甲, 乙, 丙

Abstract

English Title

Engligh abstract goes here.

Keywords: a, b, c

目　　录

第 1 章　绪论

1.1 研究背景和意义

正则表达式让我们拥有一种简单、具体而表达力强的语言来描述字符串中的各种模式。正则表达式因此可以被用于，诸如各种程序设计语言中的字符串匹配等，需要处理字符串中结构化的数据的地方。正则表达式处理的高效性也允许我们去处理大规模的数据集合。标准的正则表达式匹配算法运行时间是 $O(mn)$[1]，其中 n 是被匹配的字符串的长度，而 m 是模式的长度。

为了让计算机程序处理正则表达式，首先，我们需要把输入的正则表达式处理成为计算机可以进一步进行处理的形式。实际上，我们可以将正则表达式看作一个简单的、表达能力有限的程序设计语言。这样的话，我们就可以使用编译原理的技术去实现我们的目标。首先，我们可以使用上下文无关文法对正则表达式进行严格的定义，然后利用词法分析程序和语法分析程序对输入的正则表达式进行词法分析和语法分析，并生成一个抽象语法树。在本文中，作者使用的是递归下降语法制导分析。生成抽象语法树以后，我们可以使用抽象语法树去和给定的字符串进行匹配，并给出判定结果：输入的字符串是否符合给定的正则表达式所描述的结构。

实现一个正则表达式解析器有两种技术路线选择：一种选择是使用命令式编程范式来实现，这种范式是大多数主流语言，例如 C、Java 等所支持的；另外一种选择是使用函数式编程范式来实现，编程语言 Standard ML、OCaml、Haskell 等支持这种范式。其中，相对于命令式的程序设计范式，函数式编程中的模式匹配特性有利于我们编写更简洁的语法分析和词法分析程序，高阶函数抽象能够帮助我们把程序组织的更加符合直观的思路，方便我们对程序进行正确性的证明。

本文选择了 Standard ML/New Jerseys 编译平台实现。（以下简称为 SML/NJ）本文选择 Sntandard ML 语言主要基于的理由：首先，Standard ML 的语法和语义是用形式化是采用形式化的方法严格给出的，而 SML/NJ 编译器也经过了严格的形式化的验证，所以，程序语言和平台的可靠性有数学证明作为保障。其次，SML/NJ 是一个轻量级的编译器，使用起来也非常方便。

1.2 研究现状及挑战

1.3 研究内容与论文结构

本文论述了一个基于 Standard ML 程序设计语言的正则表达式引擎的设计与实现。并对其中的关键算法给出了数学的推导和正确性的证明。

第二章将会对正则表达式给出严格的形式化定义。并且对 Standard ML 语言进行简单的介绍。

第三章将会介绍生成正则表达式抽象语法树的内容，包括词法分析和递归下降语法分析。

第四章将会介绍字符串和抽象语法树匹配的算法和实现。

第五章将会对算法的正确性进行证明。

第六章将会进行展望和总结。

第 2 章　插入算法的示例

可用如下方式插入算法流程：

Algorithm 1 ACP 调度器

输入： 要调度的作业集合 J 和资源集合 R

for each r ∈ R **do**
  使用基准测试工具对 r 进行测试
  获取 r 的 $storage_r$ 和 $compute\_power_r$ 的值
  将 $storage_r$ 和 $compute\_power_r$ 写入到资源限制中
end for
while true **do**
  根据上次运行的结果，更新任务约束
  for each r ∈ R **do**
    if r.schedular ≠ NULL **then**
      if r.runningtask ≠ NULL **then**
        rt ← r.runningtask
        st ← rt.starttime
        et ← rt.endtime
        pd ← rt.performancedemand
        sr ← rt.scheduledresource
        根据 (rt, st, et, pd, sr) 更新约束条件
        // 根据任务的开始时间，结束时间，性能需求和其运行资源的性能添加一条约束
      else
        r.runningtask ← NULL
        // 将该任务设置为已完成
        将 r.runningtask 从 r.schedular 中移除
        pj ← r.parentjob
        if pj.schedular = NULL **then**
          从 J 中删除 pj
        end if
      end if

end if

end for

新建一个带有资源约束的模型

将任务约束加入到 OPL 模型中

将 J 和 R 作为约束添加到数据中

求解 OPL 模型

按照得出的最优解对任务进行分配

等待新的事件出现

end while

第 3 章　基本概念

3.1 本章内容概述

本章分为两节，第一节对正则表达式给出上下文无关文法的定义，第二节简要地介绍 Standard ML 语言的基本语法和在本文中比较常用的一些特性

3.2 正则表达式

3.2.1 重要的概念回顾

在给出形式化的定义之前，这里有必要叙述一些重要的结论。这些结论在 Sipser[2] 中进行了系统的论述，这里仅简要叙述和正则表达式相关的必要的部分。正则表达式是和有限自动机相等价的计算模型，那么，首先，我们从有限自动机的形式化定义开始。

定义 3.1 (有限自动机)**.** 一个有限自动机是一个五元组 $(Q, \Sigma, \delta, q_0, F)$, 其中

1. Q 是一个有限集被称作状态集。
2. Σ 是一个有限集被称作字母表。
3. $\delta : Q \times \Sigma \to Q$ 是转移函数。
4. $q_0 \in Q$ 是初始状态集。
5. $F \subseteq Q$ 是接受状态集。

在给出有限自动机的定义以后，我们继续对有限自动机识别字符串给出一个严格的定义。从直观上讲，只要输入的字符串一步一步地按照状态图规定的状态运行，最终到达接受状态，我们就可以说有限自动机接受了这个输入的字符串。

定义 3.2 (有限自动机接受语言)**.** 令 $M = (Q, \Sigma, \delta, q_0, F)$ 是一个有限自动机，令 $w = w_1 w_2 ... w_n$ 是一个字符串，其中每一个 w_i 都在字母表 Σ。则我们称 M 接受 w 当且仅当一个状态序列满足下列条件：

1. $r_0 = q_0$
2. 对所有的 $i = 0...n-1$ 都有 $\delta(r_i, w_{i+1}) = r_{i+1}$
3. $r_n \in F$

我们称有限自动机 M 识别一个语言 A 当且仅当 $A = \{w | M接受w\}$。

定义 3.3 (正则语言)**.** 我们称一个语言为正则语言当且仅当存在有限自动机识别这个语言。

就像在算术中，我们可以对阿拉伯数字定义 $+\times$ 这样的操作，组成 $2\times(3+2)$ 这样的算术表达式。在正则语言中，我们也可以对正则语言定义一些操作，我们可以称之为正则操作。

定义 3.4. 令 A 和 B 是两个正则语言，我们定义并、连结和克莱因星号操作如下：

1. 并：$A \cup B = \{x | x \in A或x \in B\}$
2. 连结: $A \circ B = \{xy | x \in A且y \in B\}$
3. 克莱因星号 $A^* = \{x_1 x_2 ... x_k | k \geq 0并且所有的x_i \in A\}$

我们可以证明，所有的正则操作都是在正则语言里封闭的，也就是说，所有的正则语言经过正则操作以后都可以被有限自动机所识别，具体的证明内容放在了第六章。

继续刚才算术中的例子，正如我们可以用算术操作符组合出算术表达式，我们也可以将正则语言用正则操作组合出正则表达式。我们不妨递归地进行定义。

定义 3.5. 我们称 R 是一个正则表达式，如果 R 是

1. a 对字母表 Σ 中的任意字母 a
2. ϵ
3. $\emptyset$
4. $(R_1 \cup R_2)$, 其中R_1和R_2都是正则表达式
5. $(R_1 \circ R_2)$, 其中R_1和R_2都是正则表达式
6. (R_1^*), 其中 R_1是一个正则表达式

我们也可以证明这个正则表达式的定义和有限自动机是等价的，具体的证明内容参考第六章。

3.2.2 上下文无关文法定义正则表达式

基于定义 3.5，我们可以使用上下文无关文法递归地定义正则表达式。但是在此之前，我们需要修改一下正则表达式约定的符号。由于计算机命令行程序难以处理 ()，并且没有 $\cup$ 符号，我们约定：”(” 和”)” 修改为 ”[” 和”]”, ”$\cup$” 修改为”+”。

3.3 二级标题

二级标题

3.3.1 三级标题

三级标题

参 考 文 献

[1] SEDGEWICK R. Algorithms in C++[M]. [S.l.]: Addison Wesley, 1992.

[2] SIPSER M. [M]. 2nd ed. : 31 – 65.

致　　谢

本科生涯看似漫长却又一晃而过，回首走过的岁月，我感慨良多。从最初的论文选题、思路梳理到研讨交流、反复修改直至最终完稿，都离不开老师、同学和亲人们的支持和无私帮助，在此我要向他们表达我最诚挚的谢意。

...

求学生涯暂告段落，但求知之路却永无止境。我将倍加珍惜大学生活给予我的珍贵财富，不忘初心，砥砺前行！

本科期间发表论文和科研情况

...