|  |  |
| --- | --- |
| 编号 |  |

本科毕业设计（论文）

|  |  |
| --- | --- |
| 题目 | 基于微信平台的经典图灵机模型仿真与测试 |

|  |  |
| --- | --- |
| 学生姓名 | 李皓琨 |
| 学号 | 161810120 |
| 学院 | 计算机科学与技术学院/人工智能学院 |
| 专业 | 计算机科学与技术 |
| 班级 | 1618104 |
| 指导教师 | 胡军 副教授 |

二〇二二年五月

南京航空航天大学  
本科毕业设计（论文）诚信承诺书

本人郑重声明：所呈交的毕业设计（论文）是本人在导师的指导下独立进行研究所取得的成果。尽我所知，除了文中特别加以标注和致谢的内容外，本设计（论文）不包含任何其他个人或集体已经发表或撰写的成果作品。对本设计（论文）所涉及的研究工作作出贡献的其他个人和集体，均已在文中以明确方式标明。

|  |  |
| --- | --- |
| 作者签名： |  |
| 日期： | 20 年 月 日 |

南京航空航天大学  
毕业设计（论文）使用授权书

本人完全了解南京航空航天大学有关收集、保留和使用本人所送交的毕业设计（论文）的规定，即：本科生在校攻读学位期间毕业设计（论文）工作的知识产权单位属南京航空航天大学。学校有权保留并向国家有关部门或机构送交毕业设计（论文）的复印件和电子版，允许论文被查阅和借阅，可以公布论文的全部或部分内容，可以采用影印、缩印或扫描等复制手段保存、汇编论文。保密的论文在解密后适用本声明。

论文涉密情况：

□ 不保密

□ 保密，保密期（起讫日期： ）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 作者签名： |  |  | 导师签名： |  |
| 日期： | 20 年 月 日 |  | 日期： | 20 年 月 日 |

摘要

图灵机模型的提出为计算机理论的发展做出了极大的贡献，在形式语言理论中，文法是0型的充要条件是该文法可被某个图灵机所接受。自动机理论中的有限自动机和下推自动机均可视为图灵机的简化。出于研究和教学的目的，各种形式语言与自动机仿真软件陆续面世。本文提出了一种基于微信小程序平台的经典图灵机模型仿真软件与其测试方案。

此图灵机模型仿真软件面向安卓移动平台用户，支持用户通过触控交互操作可视化地设计一个经典单带图灵机并进行可视化的快速仿真或可视化地设计一个使用子程序技术的扩展图灵机并进行可视化的快速仿真。此软件还支持将用户设计的图灵机保存为JSON文本进行存储。本文论述了图灵机仿真软件的研究背景，使用具体的样例展示图灵机的运行过程。在此基础上进行了本APP的需求分析。接着结合项目整体架构对本APP进行了系统架构和功能流程的设计。最后对本APP中主要的类和算法的设计实现进行说明。最后，针对安卓软件的测试，我们使用了Robolectric和JUnit进行了单元测试和集成测试，并采用人工进行系统测试以保证软件的可靠性。

关键词：形式语言与自动机，图灵机，子程序技术，安卓测试

**ABSTRACT**

The Turing machine model has made a great contribution to the development of computer theory. In formal language theory, the only necessary and sufficient condition for the grammar to be 0-type grammar is that the method can be accepted by a Turing machine. Finite automata and push-down automata in automata theory can be regarded as simplifications of Turing machines. For the purpose of research and teaching, various forms of language and automata simulation software have been published one after another. This paper presents a classic Turing machine model simulation software based on Android platform and its test scheme.

This Turing machine model simulation software is oriented to Android mobile platform users, which supports users to visually design a classic single-tape Turing machine and conduct rapid visual simulation through touch interaction operation or visually design an extended Turing machine using subroutine technology and conduct rapid visual simulation. The software also supports saving user-designed Turing machines as JSON text for storage. This paper discusses the research background of Turing machine simulation software, using specific examples to show the running process of Turing machine. On this basis, the demand analysis of this APP is carried out. Then combined with the overall architecture of the project, the APP system architecture and functional process are designed. Finally, the design and implementation of the main classes and algorithms in this APP are explained. Finally, for the test of Android software, we used Robolectric and JUnit to carry out unit test and integration test, and used manual system test to ensure the reliability of the software.

Key words: Formal Language And Automata, Turing Machine, Subroutine,

Android Testing

目录

[第一章 绪论 1](#_Toc3368)

[1.1 背景和意义 1](#_Toc15647)

[1.2 国内外研究现状 1](#_Toc13191)

[1.3 本文主要工作 2](#_Toc16663)

[1.4 论文组织结构 3](#_Toc31486)

[第二章 图灵机相关内容介绍 4](#_Toc18101)

[2.1 图灵机简介 4](#_Toc21436)

[2.2 多带技术 7](#_Toc1748)

[2.3 子程序技术 7](#_Toc7920)

[2.4 图灵机与计算机 8](#_Toc17061)

[2.5 本章小结 9](#_Toc21209)

[第三章 经典图灵机仿真微信小程序设计与实现 10](#_Toc12783)

[3.1 需求分析 10](#_Toc8327)

[3.1.1 小程序功能及用例 10](#_Toc16483)

[3.1.2 复杂用例的用例描述 13](#_Toc12378)

[3.1.3 小结 18](#_Toc31364)

[3.2 概要设计 19](#_Toc29635)

[3.2.1 项目架构及功能结构设计 19](#_Toc20222)

[3.2.2 单带图灵机编辑模块设计 21](#_Toc15834)

[3.2.3 多带图灵机编辑模块设计 23](#_Toc12112)

[3.2.4 带子程序图灵机编辑模块设计 25](#_Toc8637)

[3.2.5 图灵机模拟仿真模块设计 25](#_Toc13116)

[3.2.6 图灵机文件管理模块设计 27](#_Toc19673)

[3.2.7 小结 28](#_Toc15461)

[3.3 详细设计与实现 28](#_Toc1163)

[3.3.1 开发环境 28](#_Toc17388)

[3.3.2 主要类的设计与实现 29](#_Toc27068)

[3.3.3 界面绘制逻辑设计与实现 34](#_Toc2593)

[3.3.4 核心算法设计与实现 34](#_Toc31332)

[3.4 本章小结 37](#_Toc24696)

[第四章 经典图灵机仿真微信小程序测试方案 39](#_Toc18453)

[4.1 测试范围和测试内容 39](#_Toc18528)

[4.1.1 界面功能 40](#_Toc21508)

[4.1.2 状态管理 42](#_Toc13754)

[4.1.3 状态转移函数管理 42](#_Toc17445)

[4.1.4 子程序状态管理 43](#_Toc31217)

[4.1.5 混合状态转移函数管理 43](#_Toc594)

[4.1.6 图灵机文件管理 44](#_Toc17680)

[4.1.7 图灵机仿真 44](#_Toc7584)

[4.2 本章小结 48](#_Toc21575)

[第五章 总结和展望 49](#_Toc18241)

[5.1 总结 49](#_Toc25011)

[5.2 展望 49](#_Toc20399)

[参考文献 50](#_Toc18289)

[致谢 51](#_Toc28148)

# 绪论

本章将从图灵机背景、图灵机研究的意义以及国内外关于图灵机理论以及自动机模拟仿真软件的研究等方面介绍为何需要开发一个基于微信小程序平台的经典图灵机模拟仿真软件。并简单介绍本文的主要工作内容以及全文的组织结构。

## 背景和意义

图灵机是Turing为研究可计算性的数学模型而提出的,其在理论上为现代计算机的计算能力划定了一个上界[1]。在20世纪60年代形式语言与自动机理论出现后，图灵机作为其理论的重要内容，已经在计算机科学的许多方面发挥着不可替代的理论基础及方法论的作用。

N.Chomsky在1956年前后对自然语言进行研究中提出了文法的数学模型[2]，并采用对产生式加限制的方法将文法分成了短语文法（0型文法）、上下文有关文法（1型文法）、上下文无关文法（2型文法）、正规文法（3型文法）四种文法，并在1959年证明了文法与自动机的等价性[3]，这四类形式文法分别对应着自动机理论中的图灵机、线性有界自动机、下推自动机、有限自动机。在计算机领域研究中形式语言与自动机常被用作计算机和计算过程的抽象模型来研究计算机的体系结构和算法设计等。

由于形式语言与自动机理论在现代计算机领域的基础理论作用，很多大学都在计算机相关专业开设了形式语言与自动机理论课程。作为一门极其重要的专业基础课，形式语言与自动机理论的抽象性和形式化的特点使得用传统的教学方法难以取得很好的学习效果[4]。学者们也早就注意到这个问题，随着现代计算机的普及，也涌现了各种用于教学目的形式语言与自动机仿真软件。

我国在“十二五”期间实现了智能手机持有量超过计算机[5]，在校大学生人手一部智能手机，因此开发基于移动平台的形式语言与自动机仿真软件可更为方便的展开国内形式语言与自动机理论课程的教学，学生也可以随时随地的进行形式语言与自动机的学习。

## 国内外研究现状

如上文所述，形式语言与自动机理论出现后，随着计算机的发展普及，已经出现了各种形式的自动机仿真软件。Chesñevar等人早些时候已经将自动机模拟器和自动机理论的其他教学工具分类为只支持一种自动机的工具和支持多种自动机的工具[6]。在此基础上，Chakraborty等人在2011年发表的文章中描述了形式语言与自动机仿真软件在理论教学中的重要意义并汇总了在此之前近五十年间涌现的自动机仿真软件[7]。Chakraborty以仿真软件设计的形式为标准将自动机仿真软件划分为了基于语言的自动机仿真和可视化的自动机仿真两大类，在此基础上又按照输入方式的不同将基于语言的自动机仿真软件划分为基于符号语言的、基于汇编语言的、基于过程语言的、基于描述性语言的四类仿真软件，对于可视化的仿真软件则是细分为了基于结构化输入的、基于图表输入的两类仿真软件。

基于语言的自动机仿真软件的优势在于其可以用来模拟大型的复杂的自动机，但缺点是需要学习相应的编程语言；可视化的仿真软件虽然不需要学习编程语言，但是仅适用于小型自动机的仿真，当面对大型自动机的仿真时，可视化的输入一个自动机将将耗费更多的时间[8]。在所有类型的自动机仿真软件中，基于图表输入的可视化自动机仿真软件最符合人们在之上绘制自动机的习惯，因而更容易在教学中展开。

迄今为止，使用最广泛的自动机仿真软件是由Rodger等人研发的JFLAP。JFLAP是在其开发的FLAP[9]基础上使用JAVA语言重新开发完善的一款面向桌面平台的基于图表输入的可视化自动机仿真软件，支持对确定性和非确定性的有穷自动机、下推自动机、图灵机、特殊的有限自动机如米利机和摩尔机进行仿真。用户可以在JFLAP中通过鼠标点击和拖拽绘制自动机并通过交互操作设置状态和转移的属性。根据Rodger在2009年公布的数据，JFLAP的下载次数便超过了64000次，使用用户遍布160多个国家[10]。

相较于桌面平台形式语言与自动机仿真的百家争鸣，面向移动平台的形式语言与自动机仿真软件在Carlos H. Pereira和 Ricardo Terra推出FLApp[11]之前是一片空白。FLApp是一款面向安卓移动平台的软件，实现了乔姆斯基层次结构各层的主要特性。目前该软件已经用于巴西大学的形式语言与自动机课程中。

国内大学虽然在改革开放后陆续展开了形式语言与自动机课程的教学，但是用于教学目的的形式语言与自动机仿真软件发展缓慢，目前也没有任何得到正式应用的基于移动平台的仿真软件。

## 本文主要工作

本。

## 论文组织结构

本篇论文的组织结构如下：

第一章 从图灵机背景、图灵机的研究意义以及国内外关于图灵机理论以及自动机模拟仿真软件的研究等方面对课题进行介绍，简单阐述本文的主要工作内容；

第二章 对图灵机的理论知识及两种常用的图灵机的程序设计技术：多带技术和子程序技术进行介绍；

第三章 对微信小程序平台的经典图灵机模拟小程序进行需求分析和概要设计，并围绕着主要类和核心算法进行详细设计说明,并介绍界面实现和核心算法的实现；

第四章 介绍了经典图灵机仿真微信小程序的测试方法；

第五章 本次毕业设计工作总结，并对该毕业设计所得软件的未来发展方向进行展望。

# 

# 图灵机相关内容介绍

本章主要介绍了图灵机的基本概念，包括图灵机的理论模型和形式化定义。然后介绍了一种常用的图灵机的程序设计技术：子程序技术。通过以上介绍，使读者能对本文所实现的经典单带图灵机和使用子程序技术的扩展图灵机有一个初步的认识。

## 图灵机简介

图灵机是一个抽象的数学模型，它的提出是将人们用纸和笔进行数学运算的过程进行了抽象。Turing把人们计算的过程看作以下动作：首先把纸带划分成一个个小格子，其次在每步计算中人们只注意纸带上一个格子的内容；再次，下一步的操作只取决于当前这个符号与当前计算者的思维状态；最后，计算者下一步的操作将改写或删除当前格子中的内容并将注意力转移到左邻或右邻的格子中[12]。

Turing所描述的抽象机器由一个控制器和一条无限长的纸带组成，两者由一个读写头关联起来[13]。图灵机的控制器中具有有穷个状态，它的纸带被划分无穷多个格子作为存储单元，它的读写头可在控制器的控制下在带上左右移动并做出指定的读/写动作。一个可视化的图灵机基本模型图 2.1.1所示。

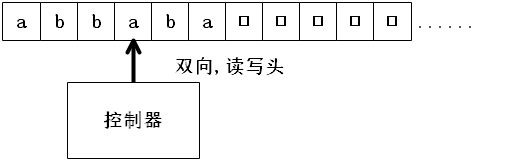


图 2.1.1 图灵机模型示意图

在图 2.1.1中，纸带从左端依次存放着输入符号abbaba，没有放入符号的格子使用空白符号☐，这个的格子一直向右延伸无穷多个。在一个动作中，图灵机根据读写头读取的符号和控制器的状态, 执行如下操作： 改变状态;；在读写头指向的格子上重写一个符号；将读写头向左或向右移动一个单元。图灵机在初始状态的控制下开始动作，并在接收状态或拒绝状态停止动作（停机）。

根据陈有祺编著的《形式语言与自动机》给出的图灵机定义[14]，一个确定的单带图灵机是一个八元组；其中，是有穷状态集；是有穷输入字母表；是有穷字母表（）；是转移函数（）；是空白符号（）；是初始状态；是接受状态；是拒绝状态（）。

前文提到，图灵机时为了模拟人们的计算过程而产生的抽象模型，故图灵机与有限自动机、下推自动机等自动机最特殊的地方就在于图灵机是有计算功能的。假设一个非负减法的定义如下：

，

如果用数字1两端0的个数来代表m和n，用作为空白符号，那么图灵机的纸带上的内容则是，图 2.1.2是一个实现该非负减法的图灵机的转换图。在转换图中读写头的移动方向用L（左移）、R（右移）、S（停机）来表示。

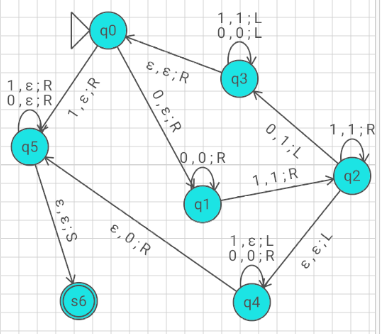


图 2.1.2 图灵机转换图

使用转换图的方式表示一个图灵机是图灵机最简洁直观的表示方法。若使用八元组来表示图 2.1.2所表示的图灵机，则其形式语言描述为：

其中，函数为：

（1）；将左边的0改为

（2）；在的控制下向右找1

；找到后变为

（3）；将1后面第一个0变为1，返回

；

（4）；在的控制下向左找空白

；

（5）；转（1），重复消去动作

（6）；遇到，表示1的右边已没有0，此时运算已进入尾

声，要消去所有的1

（7）；在的控制下，将1改写为

（8）；

（9）；时，状态表示减法结束

（10）；时，在的控制下，将后面的1和0全部变为，

； 直到遇见才以状态结束。此时带上全是空白，表示0

；

；

其余情况均属于拒绝状态。

此外，在构造图灵机的过程中，除了使用八元组的方式和转换图的方式之外，还可以采用矩阵表示的方法。在使用矩阵表示图灵机时主要是将函数采用矩阵的形式表示，这种表示方法相比于使用八元组表示的方法更为简洁也节省了书写篇幅，但是相比于使用转换图表示的方法不够直观。表给出了图 2.1.2所示图灵机的矩阵表示。

表 2.1.1 图灵机的矩阵表示

|  |  |  |  |
| --- | --- | --- | --- |
| 状态/符号 | 0 | 1 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 多带技术

常用的图灵机的程序设计技术有在在状态中存储符号、多道技术、子程序技术等[14]，这里所说的程序设计技术时指图灵机中构造一个实现具体功能的图灵机的过程，因为其与计算机上编制一个程序的过程十分相似。灵活使用这些技术可以极大的降低构造具体的图灵机的难度。本文主要介绍图灵机设计中的子程序技术。

## 子程序技术

常用的图灵机的程序设计技术有在在状态中存储符号、多道技术、子程序技术等[14]，这里所说的程序设计技术时指图灵机中构造一个实现具体功能的图灵机的过程，因为其与计算机上编制一个程序的过程十分相似。灵活使用这些技术可以极大的降低构造具体的图灵机的难度。本文主要介绍图灵机设计中的子程序技术。

子程序技术，顾名思义，就是要把图灵机中需要重复使用的一块内容抽出来形成一个“子程序”。与在计算机上编程时使用的子函数类似，子程序应当可以独立运行也需要能被父程序调用作为父程序的一部分运行。子程序技术是非常重要的技术，使用子程序技术可以将复杂的问题简单化、模块化，并提高可读性。

为了实现子程序的调用，一个子程序图灵机需要具有其自己内部的初始状态和接受状态，这样就可以保证子程序图灵机可以独立的运行并实现具体的功能。当主程序图灵机到达为子程序图灵机设置开始状态时，就相当于调用这个子程序。首先，将纸带传递给子程序图灵机；其次，从子程序图灵机内部的初始状态开始运行；最后，当子程序图灵机运行到接受状态后将纸带传回主程序图灵机；然后，主程序图灵机将从为子程序图灵机设置的返回状态开始继续运行。下面将使用图 2.2.1的例子来讲解子程序图灵机是如何发挥作用。

|  |  |
| --- | --- |
| （a）子程序图灵机copy转换图 | （b）非负整数乘法图灵机转换图 |

图 2.2.1使用子程序技术的扩展图灵机

图 2.2.1是一个使用子程序技术的扩展图灵机，该图灵机实现了整数的非负乘法。如果用数字1两端0的个数来代表m和n，那么纸带经过该图灵机处理后将变成。图灵机实现该过程的思想是从1的左边消去一个0后，在纸带的后边追加n个0；当1左边所有的0都被消去后，1的右边就得到了；之后消去无关符号就得到了最终结果。

在这个过程中，在纸带后边追加n个0的动作是可以提取出来作为一个子程序图灵机的，如图 2.2.1（a）。从图中可以看到子程序图灵机也是一个确定的单带图灵机。其余所有的操作由主程序图灵机进行，如图 2.2.1（b）。从图中可以看到子程序的开始状态为，返回状态为。当主程序图灵机运行到状态时便开始执行子程序图灵机copy的操作，待子程序处于接受状态停机时又返回到主程序图灵机的状态，此时便完成了依次子程序的调用，主程序图灵机将继续执行主程序图灵机的后续操作。

## 图灵机与计算机

在《自动机理论，语言和计算导论》中有这样两句断言[15]：计算机能模拟图灵机；图灵机能模拟计算机，且至多在计算机花费步数的某个多项式时间内。以上两句断言由于篇幅原因不进行论证。本文仅对用计算机模拟图灵机进行一些说明以证明本文所进行的工作的可行性。

冯·诺依曼体系结构的计算机可以看作是图灵机模型的物理实现，故两者的计算能力应该是等价的[16]。当然这个等价的计算能力是在理论上体现的，计算机的计算能力会受到时间空间因素的制约。最突出的问题是图灵机的纸带是可无限增长的，但是计算机的存储空间要受到硬盘、内存等设备空间的制约。这就说明计算机不可能完全仿真出“无穷”、“任意”的图灵机。当排除“无穷”、“任意”等与时空相关的因素之后，计算机便可以仿真出图灵机[17]。

本文所实现的图灵机模型仿真APP便是针对图灵机中划分出的一个重要子类：如果一个图灵机对所有输入都能停机，则称这个图灵机是完全的[14]。在图 2.1.2 图灵机转换图和图 2.2.1使用子程序技术的扩展图灵机所示的两个图灵机都是完全的图灵机。

## 本章小结

本章主要介绍了图灵机的基本概念，包括图灵机的理论模型和形式化定义。使用了一个实现非负减法的具体的图灵机作为例子来体现图灵机除了识别能力之外所具有的计算能力的特点，并借此引出了三种图灵机的表示方法。然后介绍了一种常用的图灵机的程序设计技术：子程序技术。把构造图灵机与在计算机上编程进行类比，说明了子程序的重要性，然后使用实现非负整数乘法的图灵机作为例子阐述了子程序技术是怎么运行的。这些内容是完成经典图灵机模型仿真APP的基础。 最后，对使用计算机模拟图灵机的一些条件进行了说明，解释了本文所进行的工作的可行性。

# 经典图灵机仿真微信小程序设计与实现

本章围绕经典图灵机仿真微信小程序的需求分析、概要设计以及具体设计展开。在本章中，为了避免不必要的、过长的图灵机类型名称被多次提及，在此统一将“经典的单一纸带图灵机”、“拥有多条纸带的图灵机”以及“带有子程序模块的图灵机”简称为“单带图灵机”、“多带图灵机”、“带子程序图灵机”。本章的第一节列出了该微信小程序应该具备的功能模块，以及对应的功能分析，并且对其中比较复杂的功能进行更加具体的描述。本章的第二节介绍了该微信小程序的软件整体架构以及各个模块的概要设计。本章的第三节将在概要设计的基础上详细说明具体的界面实现情况、主要的类以及功能中重要算法的设计。

## 需求分析

### 小程序功能及用例

本节根据任务书确定了多个基本功能模块，分别是单带图灵机编辑模块、多带图灵机编辑模块、带子程序图灵机编辑模块、图灵机模拟仿真验证模块、图灵机模型的文件管理模块，小程序权限模块。

单带图灵机编辑模块要能够实现单带图灵机的创建、修改功能，并且能有较为令人舒适的视觉反馈，主要包括：

（1）界面绘制：要求支持图形化的创建和编辑界面，且该界面必须拥有与编辑相关的功能按钮，按钮的图标必须简单易懂，方便用户上手，同时在绘制区域，图形的绘制必须能时刻反馈用户的操作，相关正在创建或者修改的组件必须要有对应的视觉反馈以让用户知晓目前正在进行的操作是否符合预期；

（2）图灵机状态编辑：要求支持图灵机状态的创建、移动、删除、选择、初始状态设置和取消、接受状态（结束状态）的设置和取消功能；

（3）状态转移函数编辑：要求支持图灵机状态转移函数的创建、删除、内容重编辑，状态转移函数的内容需要有检测函数检测内容是否合法，状态转移函数的创建功能在视觉上还需要包括创建从自己出发指向自己的特殊状态转移函数的绘制，以及两个状态之间构成环形的特殊状态转移函数的绘制；

（4）编辑内容的撤销功能：要求支持对前面做出的改动可以使用撤销功能取消改动，同时也要求可以撤回上一次的撤销操作，恢复之前的编辑内容；

（5）文件保存功能：要求支持输入文件名的功能，并且对输入的文件名需要进行正确性检测，通过检测后点击确认能直接跳转至图灵机模型的文件管理模块以及该模块界面；

（6）截图功能：要求支持直接从编辑界面的图形内容上直接保存截图；

（7）即时模拟功能：要求支持将编辑界面当前编辑的图灵机模型直接送到图灵机模拟验证模块并且跳转到该模块对应的界面。

多带图灵机编辑模块要能够实现多带图灵机的创建、修改功能，其主要功能中包括单带图灵机编辑模块中的界面绘制、图灵机状态编辑、文件保存、截图、即时模拟功能，另外还有：

（1）纸带数确认功能：要求在进入多带图灵机模型构建界面前必须确定该多带图灵机模型的纸带数量；

（2）状态转移函数编辑：基础功能与单带图灵机编辑模块的状态转移函数创建相同，但是状态转移函数的属性与单带图灵机不同，需要有对应的多带状态转移函数内容检测函数来检测输入是否合法。

带子程序图灵机模块要能够实现带子程序图灵机的创建、修改功能，其主要功能中包括单带图灵机编辑模块中的界面绘制、图灵机状态编辑、文件保存、截图、即时模拟功能，另外还有：

（1）子程序状态绘制：导入的子程序模块所显示的状态必须要有特别的视觉效果以告知用户该状态为特殊的包含子程序的状态；

（2）子程序状态编辑：要求支持子程序模块的导入、移动、删除、选择、子程序状态的初始状态设置和取消、子程序状态的接受状态（结束状态）的设置和取消功能。

图灵机模拟仿真验证模块需要能从文件中直接读取已经构建好的图灵机模型或者从编辑界面获取当前编辑的图灵机模型，结合待检测的输入字符串进行模拟仿真，其功能主要包括：

（1）界面绘制：要求能识别图灵机的类型，并且对相应的图灵机模型进行图形化绘制。除此之外还需要对验证过程中每一步的过程以及验证的结果纸带进行绘制，绘制内容包括纸带本体、读写头位置、状态遍历路径；

（2）输入验证字符串：用户可以输入任意字符串作为数据供图灵机模拟器进行仿真；

（3）单步执行功能：要求点击一次按钮后，图灵机模拟器只执行下一步的操作，并且绘制出对应的运行状态快照；

（4）快速执行功能：要求点击一次按钮后，图灵机模拟器直接执行到结束条件，条件包括字符串被接受、字符串不被接受、结果数量超出阈值；

（5）结果切换功能：要求有两个按钮用于切换当前显示的运行状态快照，显示不同的纸带内容、读写头位置、状态遍历路径；

（6）截图功能：要求能将当前的图灵机模型执行快照保存为PNG格式，存到磁盘中。

图灵机模型的文件管理模块主要管理整个小程序中与文件相关的各种操作：

（1）保存文件：要求能够将单带图灵机、多带图灵机、子程序图灵机模型转化到JSON，以JSON文件的格式存到磁盘中；

（2）文件列表：要求能够将保存在磁盘中的图灵机模型文件以列表的形式显示；

（3）选择文件进行修改：要求能够让用户从文件列表中选择图灵机模型文件并跳转到对应图灵机类型的编辑界面；

（4）选择文件进行模拟：要求能够让用户从文件列表中选择图灵机模型文件，跳转到对应的图灵机模拟仿真验证界面并直接从该文件中加载图灵机模型；

（5）删除功能：要求能够让用户通过长按选择删除一个文件。

由于微信小程序的诸多文件功能涉及到用户权限，该小程序还包括专门用于进行权限管理的模块，小程序权限模块主要包括：

（1）权限检测：要求在涉及到用户权限功能的地方对权限的开启情况进行检测；

（2）权限设置：要求可以让用户对小程序权限进行设置；

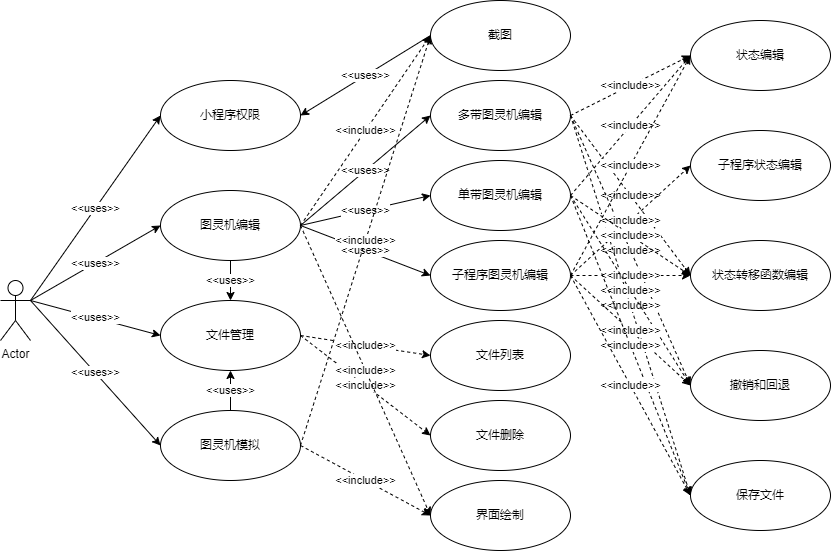


图 3.1.1 经典图灵机仿真微信小程序用例图

综合以上功能，我们可以得到如图 3.1.1所示经典图灵机仿真微信小程序的用例图。

### 复杂用例的用例描述

本节将对3.1.1用例中较为复杂的用例：单带图灵机编辑、多带图灵机编辑、子程序图灵机绘制、图灵机模拟四个用例进行详细的用例描述，从整个微信小程序的构成来看，这四个用例也是整个微信小程序中最主要且最复杂的组成部分。

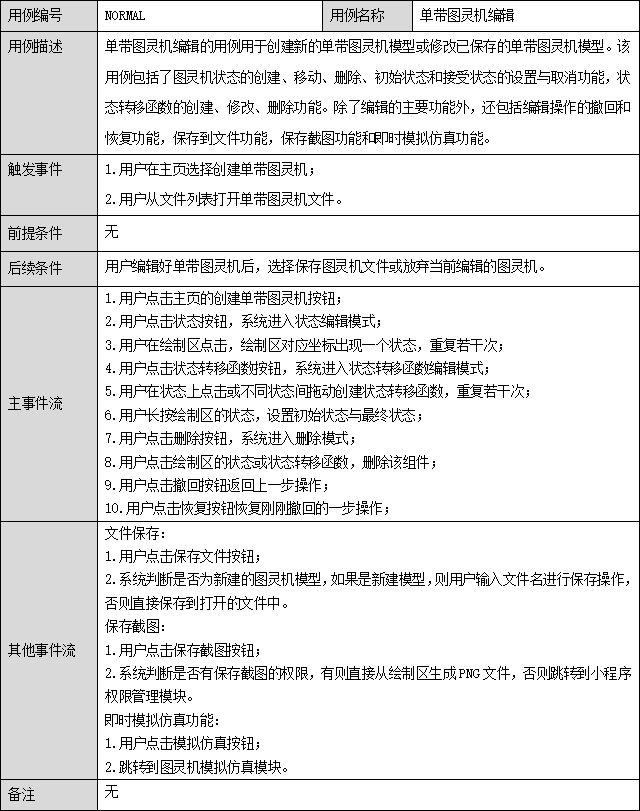


图 3.1.2 单带图灵机编辑用例描述

图3.1.2展示了单带图灵机编辑用例的用例描述，单带图灵机编辑模块是后续的多带图灵机编辑模块以及子程序图灵机编辑模块的基础。

单带图灵机编辑用例被两种事件触发，一种是用户在小程序的主界面点击创建图灵机下的创建单带图灵机按钮，另外一种是用户在主界面选择文件编辑中，从文件列表里选择一个单带图灵机的文件。在用户构建或编辑单带图灵机结束后，可以选择保存当前编辑的内容到文件，或者直接退出编辑界面放弃编辑内容。

用户在该用例的界面中可以执行的主要操作有：点击状态按钮让编辑模式变为状态编辑模式，在该模式下于绘制区点击，会在点击位置生成新状态，并且在不松手的情况下，还可以即时改变该状态的位置；点击状态转移函数按钮让编辑模式变为状态转移函数编辑模式，在该模式下于绘制区单击状态或者连接两个状态，会生成新的状态转移函数，如果连接两个状态时，另外一端没有正确连接，则该操作会被取消；用户点击选择按钮，让编辑模式变为选择模式，在该模式下长按绘制区已经存在的状态，会出现选择选项供用户选择该状态是否为初始状态或者接受状态（结束状态）；用户点击删除按钮，编辑模式变为删除模式，在该模式下，用户点击状态或者状态转移函数，系统将会对该状态或状态转移函数组件执行删除操作；用户点击撤回按钮，系统会直接回到上一次操作的状态；用户点击恢复按钮，系统会直接回到撤回前的操作状态。

除了主要的编辑操作，单带图灵机编辑用例还包括文件保存、保存截图和即时模拟仿真的功能。文件保存功能中，用户点击保存文件按钮，系统会对当前是否为新创建的图灵机模型进行判断，如果为新创建的图灵机模型，则会跳转到一个单独的页面，让用户填写正确的文件名进行保存；如果是从文件列表中选择的已存在的图灵机模型，则该按钮被按下时，系统自动将当前编辑区的图灵机模型转到JSON格式覆盖文件内容。保存截图功能涉及的初衷来源于一个实际的需求：该小程序在作为随堂实践软件使用或者被用来完成课后作业时，需要一个功能直接获取当前编辑的图灵机模型的绘制截图，方便提交给教师用于批改，或者同学之间可以通过网络更方便地互相交流，该功能会涉及到小程序的相册读写权限问题，所以和小程序的权限管理模块密切相连，用户在点击保存截图按钮时，小程序会判断该用户是否给予了小程序读写相册的权限，如果有该权限，则直接从绘制区生成PNG写入系统相册，如果没有权限，则跳转到小程序的权限管理模块。即时模拟仿真功能用于让用户可以随时判断自己写的模型是否符合预期，而不需要先通过保存到文件，然后从文件列表选择进行模拟。用户点击即时模拟的按钮之后，系统会将目前编辑区的图灵机模型传递给模拟模块，并且跳转到该模块所在的页面。

图 3.1.3展示的是多带图灵机编辑用例的用例描述，该用例本质上是对单带图灵机的纸带数的扩展，所以在单带图灵机用例的基础上有小部分的修改。

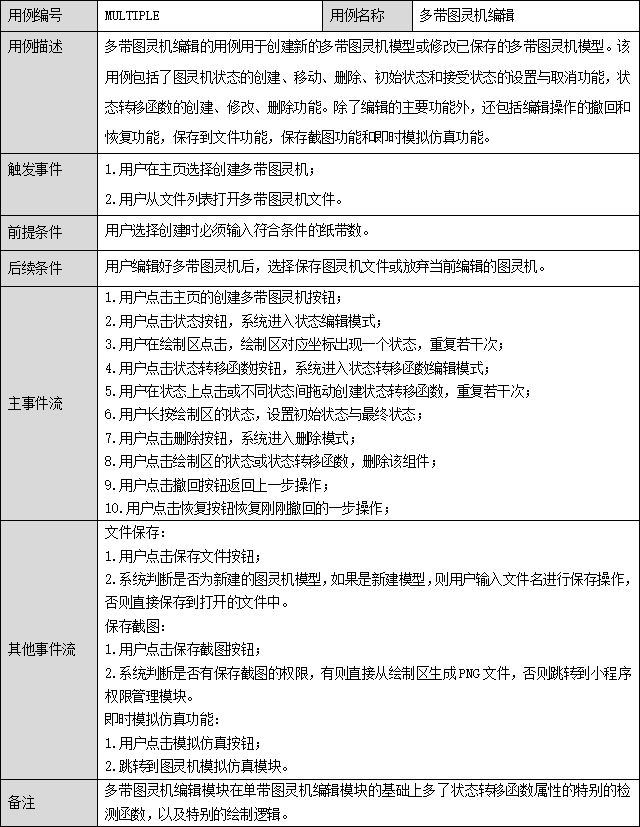


图 3.1.3 多带图灵机编辑用例描述

与单带图灵机编辑用例相似，多带图灵机编辑用例也被两种事件触发，一种是用户在小程序的主界面点击创建图灵机下的创建多带图灵机按钮，另外一种是用户在主界面选择文件编辑中，从文件列表里选择一个多带图灵机的文件。在用户构建或编辑多带图灵机结束后，可以选择保存当前编辑的内容到文件，或者直接退出编辑界面放弃编辑内容。但是在创建多带图灵机时有一个前提条件，那就是必须输入该多带图灵机的纸带数，否则无法进入编辑界面。多带图灵机编辑用例的主事件流与其他事件流于单带图灵机编辑用例基本一致。该用例与单带图灵机的区别就是有纸带数量的扩展。因此在模块逻辑上多了状态转移函数属性的特别的检测函数以及特别的绘制逻辑。

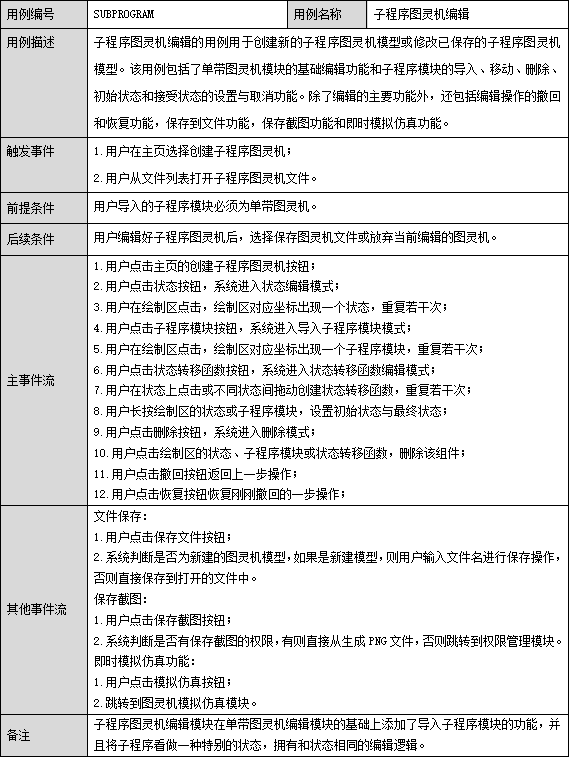


图 3.1.4 子程序图灵机用例描述

图 3.1.4 展示了子程序图灵机用例的用例描述。子程序图灵机用例本质上是对单带图灵机的状态的扩展。该用例的触发条件有两种，一种是用户选择创建子程序图灵机，一种是用户选择从文件列表打开一个已保存的子程序图灵机进行修改。该模块可以正常进行编辑的前提条件是用户导入作为特别状态的子程序模块必须是单带图灵机。在用户构建或编辑子程序图灵机结束后，可以选择保存当前编辑的内容到文件，或者直接退出编辑界面放弃编辑内容。

子程序图灵机编辑用例的主事件流和其他事件流与单带图灵机的编辑用例基本一致，但是主事件流存在一个不同点，就是子程序图灵机编辑用例特有的子程序模块导入功能。用户在点击子程序模块按钮后，系统进入子程序模块编辑模式，在该模式下，用户点击绘制区，会在点击处生成一个特别的状态，并跳转到文件列表让用户选择子程序，并且这个特别的状态也可以在选择模式下被用户长按选择是否设置为初始状态或者接受状态（结束状态）。

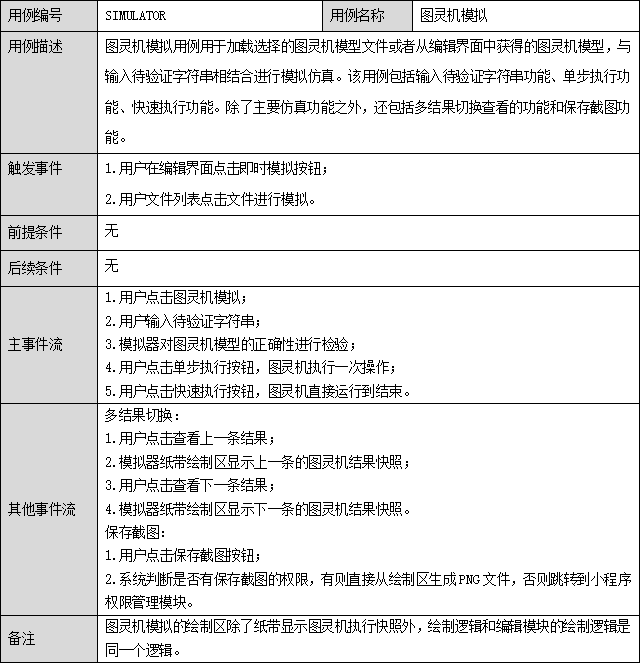


图 3.1.5 图灵机模拟用例描述

图 3.1.5展示了图灵机模拟仿真用例的用例描述。图灵机模拟仿真不需要特别的前提条件和后续条件。该用例会在两种情况下被触发：第一种是用户在编辑界面点击即时模拟按钮，第二种是用户在文件列表选择文件进行模拟。

用户在图灵机模拟仿真用例的界面中可以进行的主要操作有：输入待验证的字符串，根据图灵机的类型，该验证字符串的数量会有所不同，对于单带图灵机和子程序图灵机，只需输入一个待验证字符串，而多带图灵机则要求用户输入符合纸带数量的待验证字符串；在输入完待验证字符串之后，模拟器会对存储在内存中的图灵机模型进行正确性检测，如果是不正确的图灵机模型，则模拟仿真过程不会开始，反之直接进入模拟仿真过程；用户点击单步执行按钮，模拟器只执行接下来的一次操作，并反馈运行快照；用户点击快速执行按钮，模拟器直接执行到结束条件，结束条件包括接受状态，无下一步操作的非接受状态以及运行错误状态。

除了主事件流的运行功能外，图灵机模拟仿真用例还包括多结果切换与保存截图两个操作。保存截图的操作与单带图灵机编辑用例中一致。多结果切换的设计原因是由于有些图灵机在执行一次操作的时候存在不同的状态转移函数符合当前读到的字符，例如式子 3.1.1所示的情况，状态q0同时存在两个状态转移函数可以转移到不同的状态q1和q2，但是两个状态转移函数拥有相同的读入字符1：

 (3.1.1)

那么在这种情况下，单步执行之后，结果会出现分支：一种是写入1，读写头右移，状态变为q1；一种是写入0，读写头右移，状态变为q2。在出现多个类似分支的情况下，一个页面是无法显示如此之多的运行状态快照的，那么就必须有一个机制，可以让用户只关注其中一个运行过程。多结果切换功能使用两个按钮，用户在按下切换上一条结果按钮时，页面绘制显示上一条的运行快照，用户在按下切换下一条结果按钮时，页面绘制显示下一条的运行快照。同时这个运行快照是可以被截图功能保存的。

### 小结

我们在本节中对经典图灵机模拟仿真微信小程序的基本功能进行了需求分析，并且准确指出了用例中最为重要的几个用例，然后对这些最为重要且非常复杂的用例进行了详细的用例描述。接下来本文将在3.2中提出项目的整体架构，对该阶段确定的用例进行概要设计，最终在3.3中阐述详细设计以及具体实现，包括界面的设计实现。

## 概要设计

由于微信小程序平台先前没有类似的项目，而JFLAP本身作为运行在PC的JVM中的软件，和微信小程序平台使用的是完全不同的接口，所以本项目的架构仅以JFLAP的项目架构为参考，主要的架构设计仍然由本人结合微信小程序平台的接口和JVM平台接口的不同点，以及手机用户和PC用户的使用操作习惯的不同，另外重新设计。

### 项目架构及功能结构设计

经典图灵机模型模拟仿真微信小程序最终是作为形式语言与自动机模拟仿真微信小程序的一个主要组成的次级小程序存在，而与其平行的其他几个自动机模拟仿真微信小程序将和该小程序共用一套架构、功能结构和界面结构。由于该平台没有先前工作，本人参考JFLAP的项目架构，结合微信小程序的接口和手机用户的使用习惯，另外进行了项目的整体架构设计和功能结构设计。

图 3.2.1展示了形式语言与自动机模拟仿真微信小程序中经典图灵机模型模拟仿真的项目架构图。该项目架构图将经典图灵机模型模拟仿真微信小程序分为四次，分别为用户交互层、中间层、存储层、运算层，下面将分别对四层内容进行简单说明。

交互层是该微信小程序最为重要的部分，用于与用户的操作进行交互，四个重要的用例全部基于该层，接受用户的操作，将操作传递到运算层进行处理并且给予用户视觉反馈，一个好的UI设计和视觉反馈将会让手机用户的操作更加舒适。该层包含了各种事件反馈，对应事件触发时会与运算层交互，让运算层完成主要工作内容并反馈。另外图灵机模型的图形化绘制核心算法也集中于该层。

中间层专门用于对JSON文件进行处理，在保存文件时，中间层对自动机编辑画布中的图灵机模型进行编码，转换到字符串格式传递给存储层，而在使用编辑功能、和模拟仿真功能时，需要从文件中恢复出原本的图灵机模型，就需要中间层从存储层获取图灵机模型的JSON格式，并转换为对应的图灵机模型对象。

存储层用于长久保存用户构建的图灵机模型，该层从中间层接受JSON格式化字符串并保存到文件，或者提取一个JSON文件供中间层解码恢复出图灵机模型。同时用户在编辑带子程序的图灵机时，也需要借助该层选择合适的子程序模块。

运算层包含了该微信小程序的核心操作算法，包括状态编辑的算法，子程序导入和编辑的算法，状态转移函数编辑的算法，删除组件的算法，撤回/恢复操作算法，模拟仿真的核心算法。

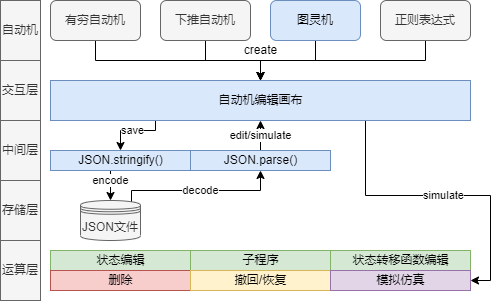


图 3.2.1 形式语言与自动机仿真微信小程序项目架构图

经典图灵机模拟仿真微信小程序的功能活动图如图 3.2.2所示。从图中可以看到，交互层包含了其他所有层的内容，用户的操作会通过交互层传递给其他各层进行处理，再由各层给出处理结果，反馈给用户。同时交互层负责的图形绘制也贯穿了编辑功能和模拟仿真的功能，对于该微信小程序来说是最为核心的部分。为了适应手机用户的操作习惯，文件系统没有太多复杂的设计，而偏向于更简约的设计方式，仅用于进行文件的浏览，打开，保存和删除。运算层则负责复杂的逻辑任务，和交互层紧密结合提供简约清晰的图形绘制。

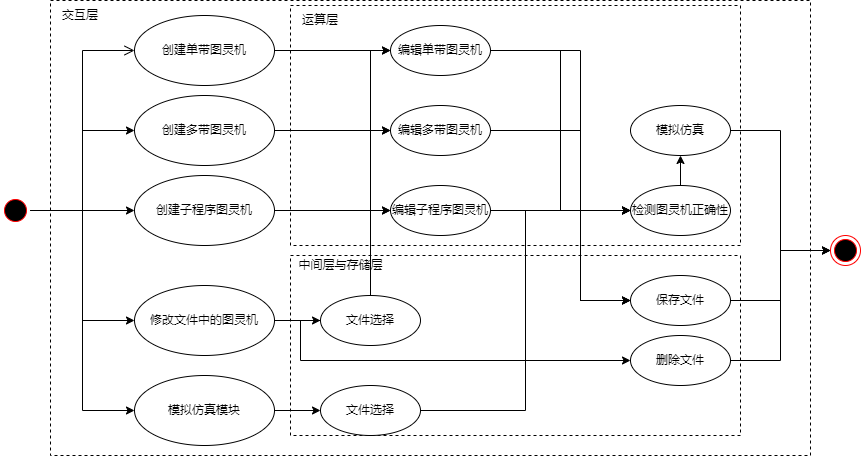


图 3.2.2 经典图灵机模拟仿真微信小程序模块功能活动图

### 单带图灵机编辑模块设计

由3.1中我们对单带图灵机编辑模块的需求分析可知，单带图灵机编辑模块中绝大多数功能是下文中多带图灵机编辑模块以及子程序图灵机编辑模块的基础，而多带图灵机编辑模块和子程序图灵机编辑模块的功能仅是在单带图灵机编辑模块的基础上进行了特殊扩展，甚至没有删减。故在本节中，我们将对单带图灵机编辑模块进行非常细致的模块设计。

单带图灵机编辑模块的活动图如图 3.2.3所示。其中对选择组件功能、保存截图、保存文件功能进行了简化，以方便在一张图中能展现整个模块的大致活动流程。接下来我们将展示被简化的功能部分的活动图。

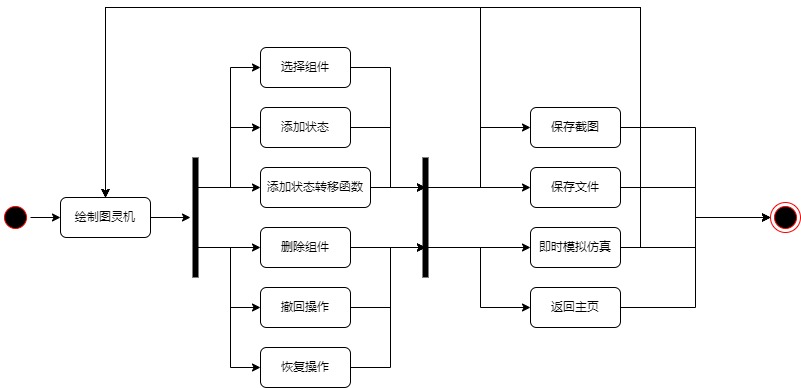


图 3.2.3 单带图灵机编辑模块活动图

图 3.2.4展示了选择组件功能的活动图。从这个活动图可以看出选择组件的复杂程度。点击选择功能之后，系统会进入选择模式，在该模式下遵循活动图中的流程，在点击绘制区时，会判断是否点击到了状态，这个分支默认用户点到了图灵机的组件，而没点到组件的时候，该功能不会被触发。在这个分支下，用户如果点击的是状态转移函数（即否分支），那么就会弹窗让用户输入新的状态转移函数，用户在输入新的状态转移函数内容后，系统会对用户的输入进行合法性检测，如果合法则修改成功，并且进入单带图灵机编辑模块中的绘制图灵机中，更新画布内容。如果输入不合法，则弹窗显示输入格式不正确，并且保留原来的状态转移函数内容。同样在这个分支下，用户如果点击的是图灵机状态，那么会判断用户的点击时间，如果用户点击时间不是长按，则会用不同颜色来标注该状态，以高亮显示该状态，表明该状态被选中，并且用户在这个情况下如果拖动状态，状态会跟随用户的手指触摸位置进行移动。

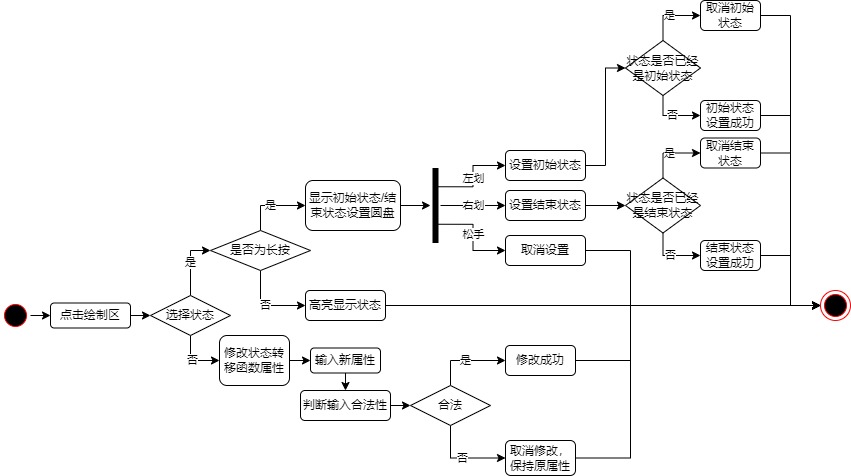


图 3.2.4 选择组件功能活动图

如果用户点击状态且不拖动状态的情况下满足一定时间，则系统会将该操作判定为长按，此时会为该状态绘制初始状态/结束状态的设置圆盘，此处的界面绘制细节中，会对该状态的初始状态/结束状态属性进行判断，如果状态已经拥有这些状态，则圆盘上对应的设置区会以浅绿色填充，反之则以浅灰色填充。

之所以使用这种视觉设计，是为了避免太多的弹窗影响用户的编辑过程，而这样的视觉和交互设计可以让用户在不被弹窗询问打断的情况下，只通过简单的手势就可以完成初始状态/结束状态的设置，并且用户可以通过这种视觉圆盘的方式简单了解到当前设置的状态是否已经有初始状态/结束状态的属性。使用圆盘则是让视觉效果更加具有现代感和科技感，让用户在操作过程中不会觉得枯燥乏味。通过判断用户手势，且结合当前状态的初始状态/结束状态的属性，系统可以对状态的初始状态/结束状态的属性进行设置和取消。

图 3.2.5展示了保存截图功能的活动图。该功能实际上的复杂点在于不拥有保存图片到系统相册权限时该如何与权限管理页面交互。由于微信小程序为了防止开发者恶意使用弹窗强制用户授权，目前取消了在页面底部弹窗的权限申请功能，所以授权必须要通过在页面的一个特殊按钮上通过点击跳转到专门的页面来进行。这里我们跳转到该小程序的权限管理模块，该模块提供了一个包含该按钮的页面用于让用户跳转到权限设置的页面。该模块同时也在小程序的设置页面留下了一个相同功能的按钮，以便用户在其他情况下可以自行更改权限。

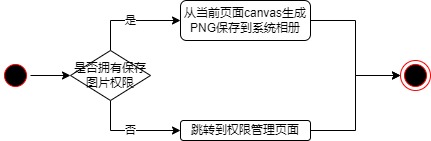


图 3.2.5 保存截图功能活动图

图 3.2.6 展示了文件保存的活动图。文件保存功能在用户按下保存文件的按钮时触发，该功能首先会对当前是否为新创建的图灵机模型进行判断，如果是从文件中读取进行修改的图灵机模型，则直接进行保存，而不需要跳转到文件保存页面。如果是新创建的图灵机模型，则跳转到文件保存页面，用户输入文件名后，选择确认保存，如果选择取消，则退出文件保存，并且弹窗显示保存取消。选择确认保存后，系统会对文件名的合法性进行判断，不合法则弹窗报错，弹窗会自动关闭，用户可以选择继续输入或者直接取消保存。如果文件名合法，则将图灵机模型JSON格式化后保存到该文件名的文件中。

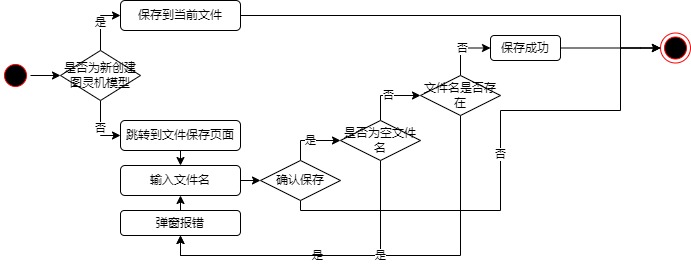


图 3.2.6 保存文件功能活动图

### 多带图灵机编辑模块设计

多带图灵机的编辑模块与单带图灵机的编辑模块相比，多了一个前提条件，那就是纸带数的输入确认。从图3.2.7的多带图灵机编辑模块活动图可以看出，该模块的内容是基于单带图灵机编辑模块，添加了输入纸带数这一个功能，如果纸带的数量不合法，直接取消创建新的多带图灵机模型，直到用户输入合法的纸带数。

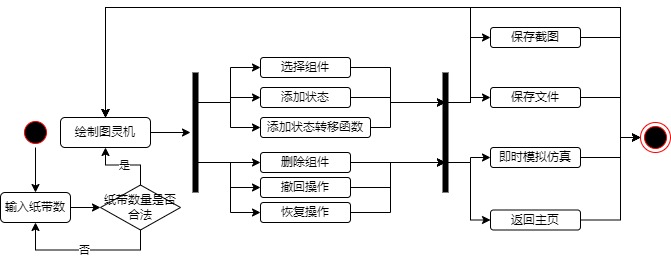


图 3.2.7 多带图灵机编辑模块活动图

值得一提的是，由于微信小程序在自定义弹窗组件和canvas组件的配合上存在问题，canvas组件总会覆盖在自定义弹窗的上层，导致无法使用自定义弹窗，那么在编辑多带图灵机的状态转移函数时，只能使用原生弹窗来进行内容的输入，而纸带数达到一定程度时，原生弹窗内输入的状态转移函数内容会非常冗长，不易于手机端用户操作，另外如果输入中因为一时疏忽，有一个条目的输入存在格式错误，那么输入的内容将会在判断时被判定为错误格式，导致这次输入前功尽弃。所以我在纸带数量的考虑上仍然采用了JFLAP的纸带数量限制方式，将纸带数量限制在2~5条。超过5条纸带的多带图灵机在手机端的操作会存在诸多不便，除了上文提到的状态转移函数内容输入，还存在如图 3.2.8所示的渲染困难，虽然图中所示为5纸带，但是可以看出来状态转移函数的内容已经存在互相覆盖的情况，这不利于小屏幕手机的使用。

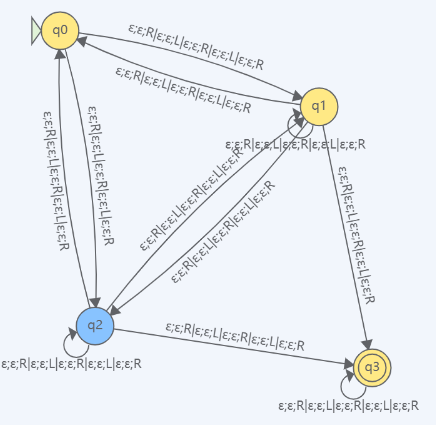


图3.2.8 多带图灵机5纸带数实例

### 带子程序图灵机编辑模块设计

图 3.2.9展示了带子程序图灵机编辑模块的活动图。很明显，图中高亮显示的部分告诉我们带子程序图灵机编辑模块也是基于单带图灵机编辑模块的，并且在单带图灵机编辑模块的基础上增加了添加子程序模块的功能。添加子程序模块是个较为复杂的功能，在图 3.2.10中我们展示了该功能的具体活动图。

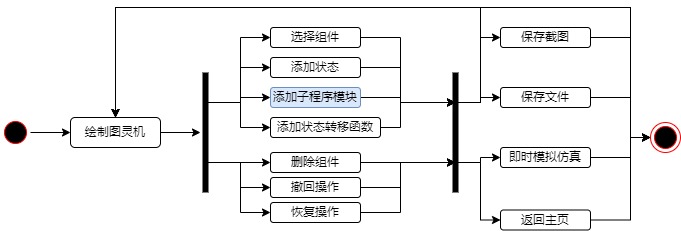


图 3.2.9 带子程序图灵机编辑模块活动图

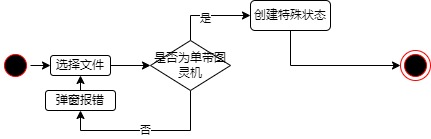


图 3.2.10 添加子程序模块功能活动图

子程序模块的添加会使用到文件管理模块的文件列表功能，在用户点击一个文件后，系统会判断该文件存储的是否为单带图灵机，如果不是单带图灵机，则会弹窗报错，并且让用户可以再次选择文件，或者取消选择退回编辑界面。当选择的文件是单带图灵机时，会返回到编辑界面，并且在用户在绘制区点击的位置生成一个新的特殊状态，该状态有特别的绘制方式，并且包含特殊属性，用于标记该状态是一个子程序模块。

同样，子程序模块虽然是特殊的状态，但是与状态共享一套选择机制，选择功能中，用户如果单击该特别状态，系统会弹窗告知该状态包含的子程序模块的源文件名。用户如果长按该特别状态，一样会进入初始状态/结束状态的设置逻辑。

### 图灵机模拟仿真模块设计

在该小程序的整体活动图（图 3.2.2）中，我们展示了该在何时，何处进入图灵机的模拟仿真模块。该模块会在单独的模拟仿真模块中由用户选择文件后直接加载文件进入模拟仿真模块，或者在编辑界面，由用户点击即时模拟仿真功能的按钮，从编辑界面跳转到模拟仿真模块的界面。

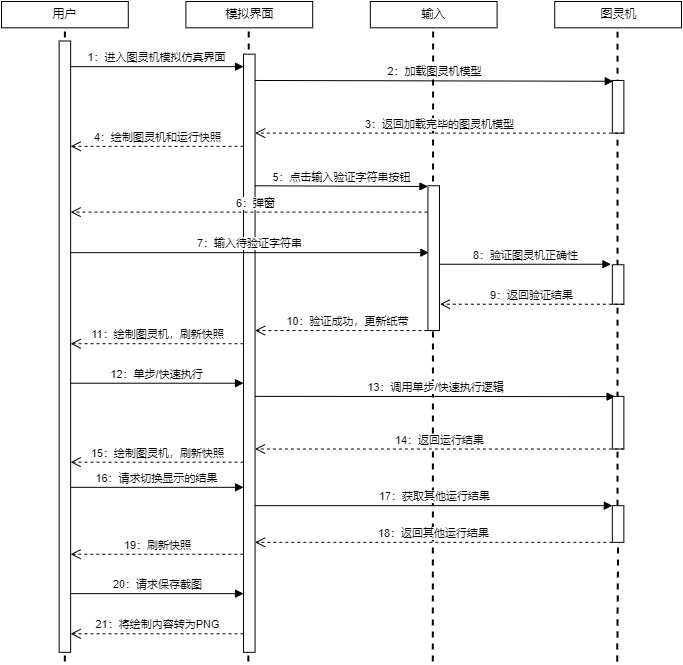


图 3.2.11 图灵机模拟仿真时序图

从时序图中我们可以看到，用户的单步执行、快速执行等操作主要是在模拟界面，再由模拟界面向输入模块和图灵机模块发送数据。在进入模拟界面的一开始，图灵机模拟仿真模块会从用户选择的文件中，或者用户跳转前的编辑界面中，获取图灵机模型并加载，然后将加载完毕的图灵机模型，以及初始化结束的纸带信息和运行快照信息反馈到模拟界面，此时模拟界面会进行第一次的绘制操作，将加载好的图灵机模型和默认的初始化运行快照绘制出来。

用户在点击输入待验证字符串按钮后，模拟界面会向输入模块发送信息，由输入模块创建弹窗让用户输入待验证字符串。在用户确认输入完毕后，系统会向已加载的图灵机模块请求验证图灵机模型的正确性。如果图灵机模型验证不正确，会弹窗告知用户当前存在的不正确的内容，并且不予以启动模拟，下面的操作也将不会进行。如果图灵机模型验证正确，则会使用待验证字符串初始化纸带，此时模拟界面会获取相关的图灵机模型信息和纸带信息，绘制对应的图形内容。

接下来由用户进行模拟的主要操作，一个是单步执行，一个是快速执行，他们都会让模拟界面向图灵机模型发送信息，让图灵机调用两个功能对应的函数运行，并且图灵机会在函数运行结束后，将数据反馈到模拟界面，由模拟界面刷新所绘制的图形内容。在运行期间或者运行结束后，图灵机的运行快照中可能会存有多个当前的快照或者运行结果快照，那么用户可以通过使用模拟界面上的切换结果功能，让模拟界面向图灵机获取其他的运行快照，并刷新绘制内容，反馈给用户。当用户选择保存截图时，该操作只由模拟界面执行，模拟界面直接从当前绘制的画布上生成PNG并保存到系统相册中。与前面的模块相同，如果没有授权读写系统相册的权限，模拟仿真模块一样会跳转到权限管理模块所在的界面请求用户授权。

### 图灵机文件管理模块设计

图灵机文件管理模块的设计不是该小程序的核心设计内容，但是对于该小程序来说，这个模块是不可或缺的。故在此简要提及。

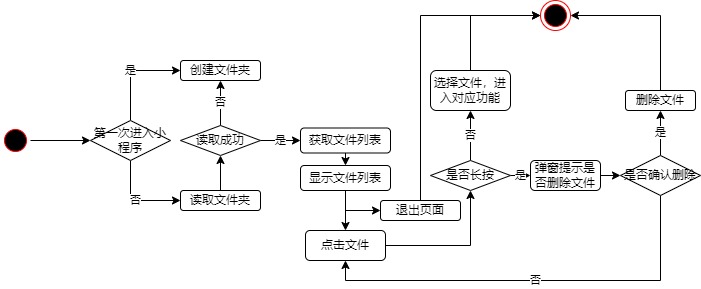


图 3.2.12 图灵机文件管理模块活动图

微信小程序对于小程序可操作的文件是存在限制的，目的是为了限制小程序对系统内的改动能力，减少恶意小程序对系统文件破坏的可能性。所以对于小程序来说，只能在微信自己创建的小程序文件夹中进行文件读写操作。由于其他小程序也有可能有自己的文件，为了防止文件冲突，该小程序会在用户第一次进入的时候，创建一个单独的文件夹用于保存文件，如图 3.2.12所示。这样的设计还让开发变得更加便利，因为该小程序需要保证读到的文件列表中都为格式正确的JSON文件，且JSON内容必须是符合规格的图灵机JSON对象，如果不单独创建文件，那么我们就需要耗费额外的算力用于逐一判断大文件夹中的每一个文件是否都符合条件，这对于手机端来说是一笔极端巨大的开销，以致于很多用户的手机会出现卡死，甚至系统宕机的情况。对于文件列表相关的功能，我们并没有实现非常复杂的内容，一方面是考虑到用户手机会为这些不必要的功能付出更多的算力代价，另一方面是文件列表本身在整个编辑和模拟过程中出现的总时间并不多。当然在未来，为了方便用户的使用，可能会引入新的功能，但是还是会优先考虑性能。文件管理模块除了以上提到的功能外，还有一个必须要有的删除功能，该功能仅在文件浏览界面通过长按选择文件进行删除，在其他的界面跳转出来的文件列表界面上，该操作是无法执行的。

### 小结

在3.2节中我们对项目的整体架构进行了说明，并且对五个大模块：单带图灵机编辑模块、多带图灵机编辑模块、带子程序图灵机编辑模块、图灵机模拟仿真模块、图灵机文件管理模块进行了概要设计。通过大量的活动图示例以及相关的解释，理清了五大模块的设计内容，在接下来的一节里，我们将对这些内容进行详细的设计与实现。

## 详细设计与实现

我们将在本节详细说明该小程序的具体模块的具体设计和实现，包括界面的具体设计理念和实现情况，在本节中我们还将详细列出相关功能的核心算法，本节中所有提及的核心算法以及核心绘制算法均未参考JFLAP项目。

### 开发环境

经典图灵机模型模拟仿真微信小程序的开发所使用的操作系统为微软Windows 10。主要开发工具为Windows 10平台的微信开发者工具。

微信开发者工具由Visual Studio Code编辑器扩展而来，整合了小程序模拟器，通过WASM组件在电脑端可以直接进行小程序调试。该小程序主要使用的语言为JavaScript，同时还使用微信小程序特有的网页文本格式WXML以及网页渲染格式WXSS。本项目使用的微信小程序基础库为2.19.6，截止论文编写时间，微信小程序基础库最新发行版为2.24.1，该小程序目前最低推荐基础库为2.15.0。

微信开发者工具支持直接通过网络连接将小程序开发包发送到手机进行调试，参与调试的手机型号有iPhone6s Plus、vivo X70 Pro+。

其中iPhone6s Plus使用iOS操作系统，vivo X70 Pro+使用基于安卓11的操作系统。

### 主要类的设计与实现

由于JavaScript在诞生初并未对面向对象的程序设计有特殊支持，所以该语言在面向对象编程上语法相对灵活，有多种方式可以实现面向对象，但是很多方式存在开销大的缺陷，为了保证性能，我们对于一些结构简单且不包含成员函数的类仅使用hashmap来实现，主要类的实现方式是函数隐式构造，总体设计仍然使用面向对象的设计方法。

1. 图灵机编辑

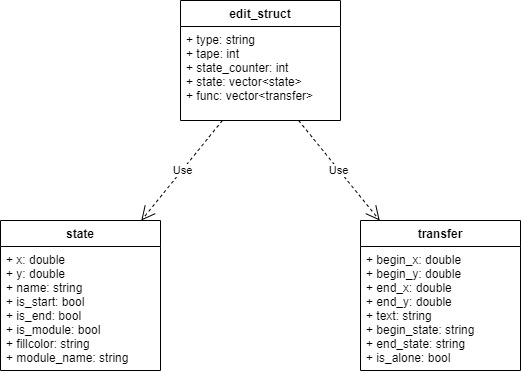


图 3.3.1 图灵机编辑类图

图 3.3.1展示了图灵机编辑所使用的类以及他们的内部变量。由于图灵机的编辑涉及到三种不同的图灵机类型：单带图灵机，多带图灵机，子程序图灵机，为了简化图灵机编辑模块的实现难度，我们对于该数据结构的内部设计综合了三个图灵机类型所共享的以及特有的内部变量。当然采取这样的设计，前提是多带图灵机编辑逻辑和子程序图灵机编辑逻辑本身是由单带图灵机编辑逻辑扩展而来。实质上如果设计更多的编辑类，多带图灵机编辑类和子程序图灵机编辑类应该是直接从单带图灵机编辑类继承而来。

该图中展示了编辑模块使用的三个主要类，而最核心的类是edit\_struct，该类包含了所有与三种图灵机有关的内部数据存储结构。type标注了当前存储的图灵机类型；tape是标注了图灵机使用的纸带数量，单带图灵机和子程序图灵机该内部变量默认为1；state\_counter用于内部计数状态数量，在用户绘制新的状态时，会使用这个变量用于生成状态的标签名；state变量保存了一个专门用于存储state类的数组;func保存了一个专门用于存储transfer类的数组。

state类和transfer类也是该模块的重要部分，在后面的三种图灵机模拟类中，这两个类也会参与其中。在state类中，x和y用64位浮点数存储了该状态在绘制区的坐标；name则存储了该状态的标签名;is\_start标注了该状态是否为初始状态；is\_end标注了该状态是否为接受/结束状态；is\_module和module\_name为子程序图灵机编辑专用的变量，用于表明当前状态是否包含一个子程序模块，module\_name则存储了该子程序模块的文件名；fillcolor是绘制逻辑专用的变量，该变量存储了绘制该状态时应该使用什么样的颜色格式，在高亮显示和模拟器的运行高亮显示中，该变量与界面绘制息息相关。

1. 撤销/恢复操作

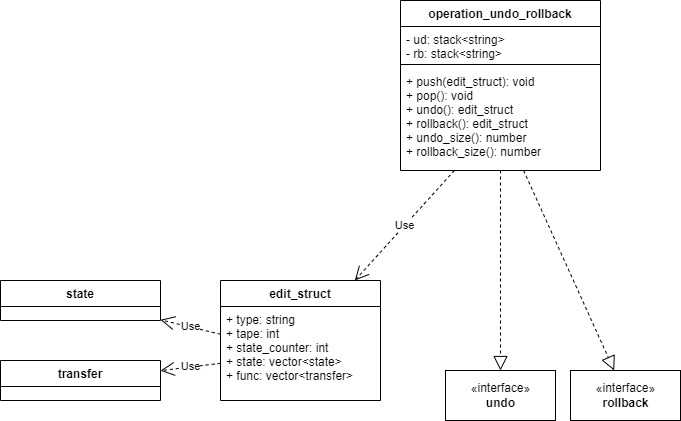


图 3.3.2 撤销/恢复操作类图

图 3.3.2展示了撤销和恢复操作使用的核心类operation\_undo\_rollback。该类与edit\_struct类有密切的交互，因为edit\_struct类在实现层面上可以被JSON准确无误地转化为string类型保存起来，或者从string中恢复出edit\_struct的结构，该类的设计可谓精妙。基于该类，我们实现了undo和rollback接口，让用户可以直接使用。

类operation\_undo\_rollback包含了两个private成员变量ud和rb，两个成员变量被设计为存储string的栈，撤销和恢复操作实际上是对当前的edit\_struct的操作栈的push和pop操作，该核心算法将在下文的具体实现中详细说明。

1. 单带图灵机模拟

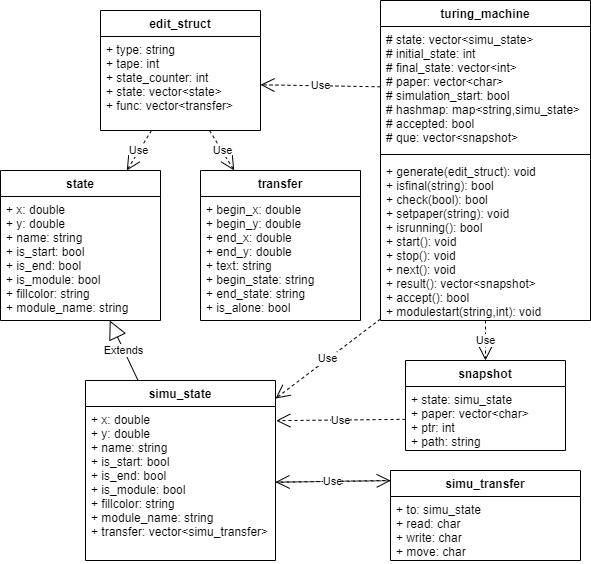


图 3.3.3 单带图灵机模拟类图

图 3.3.3展示的单带图灵机模拟类是下面的多带图灵机模拟类和子程序图灵机模拟类的父类，也就是说多带图灵机模拟类和子程序图灵机模拟类继承自单带图灵机模拟类，并且在simu\_state和simu\_transfer上有细微的不同。

simu\_state即为图灵机模拟器中参与运算的状态类，该类由turing\_machine类使用前面的edit\_struct类进行初始化操作，通过begin\_state和end\_state字段分析edit\_struct类中state与transfer之间的关联，将edit\_struct类中的state类扩展为simu\_state类，即将transfer解析为simu\_transfer，并且保存到数组中，存储在simu\_state.transfer中。

simu\_transfer即为图灵机模拟器中参与运算的状态转移函数类，其中所有内容均为public。to存储了该状态转移函数将会跳转到的状态；read表示该状态转移函数执行必须要读写头从纸带上读取的字符;write表示该状态转移函数在执行时读写头在纸带上该位置应该写的字符;move表示状态转移函数执行完毕后，读写头的移动方向，移动方向包括“L”（向左），“R”（向右），“S”（停止移动）。

snapshot为运行快照类。该类用于存储当前图灵机运行的一个结果。state存储当前已经运行到的状态；paper用于存储当前纸带信息；ptr用于存储当前快照下，读写头的位置；path用于存储当前快照下，运行的过程路径。

turing\_machine中，所有的成员变量均为protected，用于继承出多带图灵机类和子程序图灵机类。state存储了所有的图灵机状态；initial\_state保存了初始状态的下标；final\_state保存了所有接受/结束状态的下标；paper存储了初始化后纸带的情况；simulation\_start保存了当前的模拟器是否正在运行，该变量将决定交互界面的运行功能是否可用；hashmap存储了状态和标签名的映射关系，被成员函数isfinal使用，绘制逻辑通过isfinal来快速索引path中的状态是否为接受/结束状态，从而对接受/结束状态进行特殊绘制；accepted存储了当前的图灵机是否已经是接受状态；que存储了当前所有的运行快照，图形界面的结果展示和结果切换功能就依赖该变量。

turing\_machine类的所有成员函数均为public：

generate函数在生成实例的时候被调用，该函数会从一个edit\_struct对象生成对应的单带图灵机；

check函数用于在输入待验证字符串后对图灵机的正确性进行检测，如果图灵机不正确，那么模拟将不会开始；

setpaper用于初始化纸带内容，这个函数会在交互层被输入模块调用，输入模块在接受了用户输入的待验证字符串，并且检测图灵机正确性后，将用户输入设置到turing\_machine.paper中；

isrunning用于在单步执行功能执行一次之后调用，来判断图灵机是否已经运行结束，如果已经运行结束，那么用户接下来按单步执行或者快速执行按钮，都会提示图灵机模拟已结束，要求输入新的待验证字符串；

start用于正式启动图灵机模拟，该函数会设置turing\_machine.simulation\_start为true；

end用于直接终止图灵机模拟，该函数直接设置turing\_machine.simulation\_start为false；

next用于单步执行一次图灵机模拟过程，在快速执行功能中，该函数是直接反复调用的，直到isrunning()为false或者accept()为true；

result函数会返回que变量，实际上就是返回当前所有的运行快照，这样界面绘制逻辑就可以通过这个接口来实现图灵机快照绘制，以及结果切换功能；

accept函数会返回turing\_machine.accepted，用于判断当前图灵机是否有运行快照已经到达了接受状态。

modulestart函数在单带图灵机模拟中不会被使用，但是会在子程序图灵机中使用，子程序图灵机会调用modulestart来初始化一个子程序，所以该功能是预留给子程序图灵机的。

1. 多带图灵机模拟

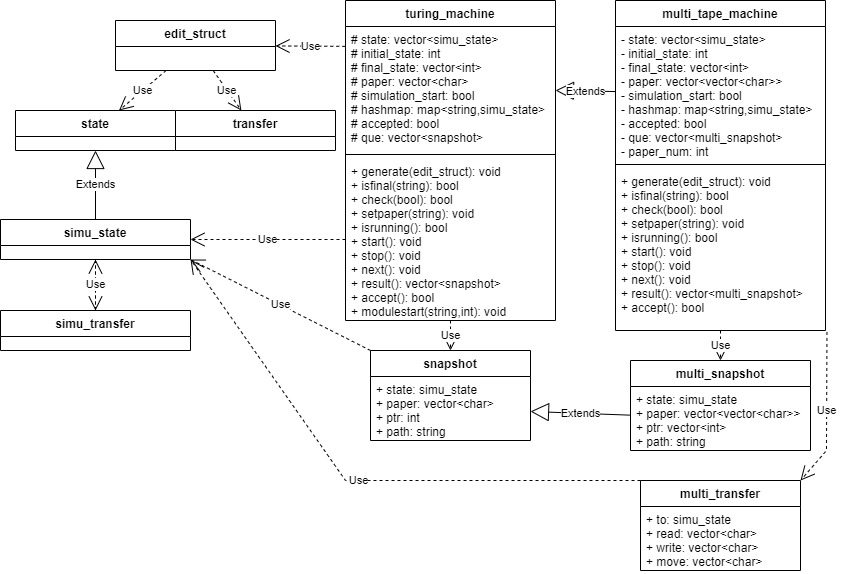


图 3.3.4 多带图灵机模拟类图

图 3.3.4展示了多带图灵机模拟的类图。multi\_tape\_machine由turing\_machine类继承而来，但是不包含turing\_machine中的modulestart方法。与turing\_machine的不同之处是，为了支持多纸带的读写，多带图灵机模拟类使用了multi\_snapshot作为快照，并且在simu\_state中存储的不再是simu\_transfer，而是multi\_transfer。

multi\_snapshot继承自普通的snapshot，但是paper和ptr都变为了数组形式，因为单带图灵机中，paper与ptr在单个快照中只有一个，但是多带图灵机因为多纸带的读写，所以paper和ptr数组的大小必须和纸带数一致。

multi\_transfer其实也继承自普通的simu\_transfer，不过在实现时并没有严格遵从这个设计，因为该类仅在多带图灵机中使用。为了满足多纸带的读写，multi\_transfer中的read，write，move都变为了数组的形式，并且数组的大小与纸带数量一致。

1. 子程序图灵机模拟

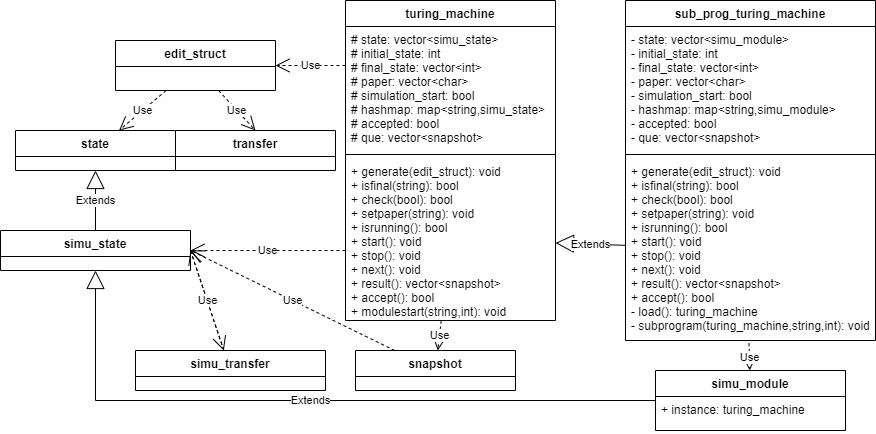


图 3.3.5 子程序图灵机模拟类图

图 3.3.5展示了子程序图灵机模拟的类图。sub\_prog\_turing\_machine继承自turing\_machine，且在状态的保存上使用了一个继承自simu\_state的子类simu\_module。

simu\_module即子程序状态，该状态类在simu\_state的基础上扩展了一个instance字段，该字段存放了一个turing\_machine，即单带图灵机的实例的引用。

子程序图灵机模拟类的方法继承自单带图灵机模拟类，但是内部还增加了两个private方法，这两个方法是无法通过外界方式调用的。load方法用于在初始化阶段从文件中提取edit\_struct，并且生成单带图灵机实例。而subprogram是专门用于对子程序模块中的单带图灵机进行快速执行的函数，因为在子程序图灵机中，子程序模块是作为一个状态出现，在单步执行到该状态时，会直接执行完子程序的内部逻辑，从而进入状态转移函数的判定中，该方法会在子程序图灵机的next方法中被调用。

### 界面绘制逻辑设计与实现

在该节中，我们对界面的设计与实现情况进行展示。在该部分中，我们将先从小程序的整体界面设计入手，然后重点说明编辑页面以及模拟界面中canvas绘制逻辑设计和实现。

1. 整体界面设计

在先前体验JFLAP的图形界面时，明显能够感觉到JFLAP并没有在界面设计上进行太多的工作，而这样一个将会被用于课堂和课后作业的小程序，好的界面设计会让使用者耳目一新，从视觉上减少枯燥感，甚至于如果有更优秀的界面设计，可能会让使用者逐渐提升对该软件的使用兴趣，从而起到鼓励用户多使用的效果，当然对于目前我们的设计思路来说，我们仅限于让使用者不会觉得使用该小程序太过于枯燥乏味。因此我们在小程序的UI设计上尽量选择比较现代化的简约配色。

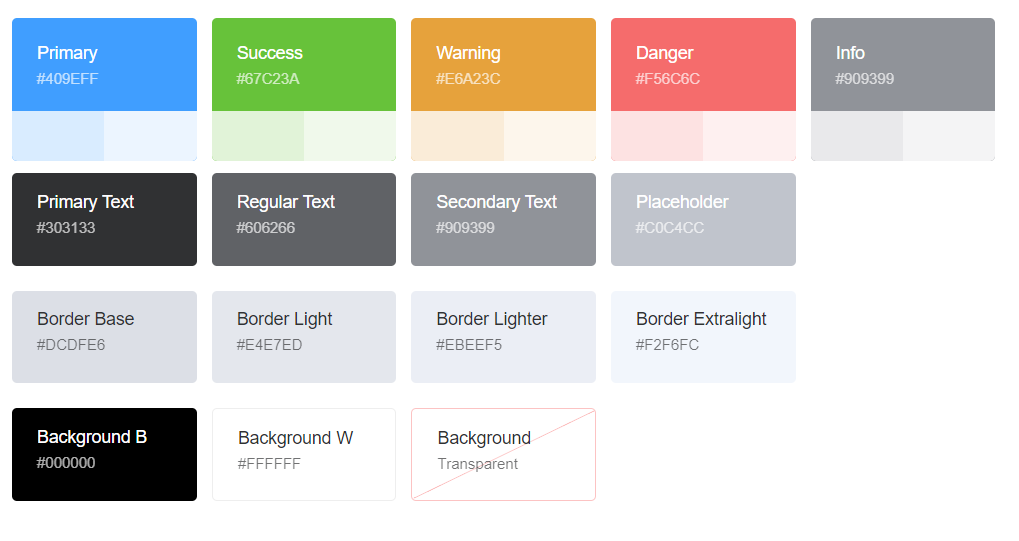


图 3.3.6 element UI 配色表

该小程序的设计风格参照了element UI，以蓝色为主要颜色，对于不同的层级设置不同深浅的蓝色，图 3.3.7展示的小程序主页界面基本上展示了该风格的底色，按钮未点击的底色，标题栏的底色以及底部导航栏的设计。底部导航栏采用比较现代且较为圆润的图标，并且在切换页面时，对应的图标底色也与设计风格中给出的蓝色一致。

我们在全局标题栏和导航栏的设置中，使用了#409eff的蓝色：

"window": {

    "navigationBarTitleText": "图灵机模拟",

    "navigationBarBackgroundColor": "#409eff"

},

"tabBar": {

    "custom": false,

    "color": "#606266",

    "selectedColor": "#409eff",

    "borderStyle": "black"

}

在app.wxss中，我们对小程序的全局背景色也进行了设置：

page {

    background-color: rgb(236,245,255);

}



图 3.3.7 小程序主页界面

我们在全局的按钮设置中，对按钮的底色设置为element UI配色中的二级蓝色，而对按钮被点击时的配色，仍然采用#409eff：

button {

    font-size: 30rpx;

    margin-top: 10rpx;

    margin-bottom: 10rpx;

    margin-left: 30rpx;

    margin-right: 30rpx;

    background-color: rgb(217,236,255);

}

.button-hover {

    background-color: #409eff;

}

.button-container {

    display: flex;

    flex-direction: column;

    justify-content: space-between;

}

在图 3.3.8中我们展示了单带图灵机编辑界面的上半部分，可以看到按钮区的风格设计就符合WXSS中的定义：

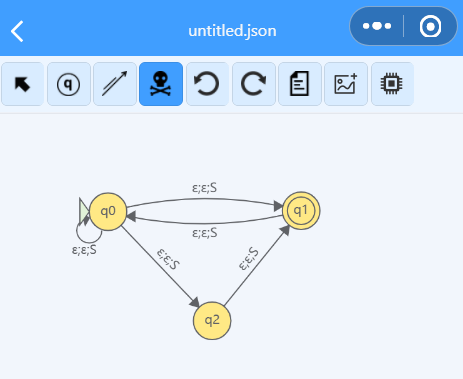


图 3.3.8 单带图灵机编辑界面（部分）

同时我们还能在图中看到，相关操作按钮采用图标的方式而非文字的方式，且图标的内容符合用户对软件的使用习惯，简约易懂的图标也会让用户能够快速上手该小程序。另外为了防止用户仍然不理解相关图标的具体含义，我们在设置中对这些图标进行了相关的解释，在点击设置中这些图标所在的按钮时，会有浅色弹窗告知该按钮在编辑界面或模拟界面中的功能：



图 3.3.9 按钮功能提示

1. canvas绘制逻辑设计

微信小程序采用的是类似于web页面一样的方式来显示小程序的主要内容，为了用户可以与页面进行实时交互，我们采用canvas组件来进行绘制，我们使用canvas2d来进行绘制，在实现中，我们还为绘制逻辑设计了专门的结构，方便在编辑模块和模拟模块中复用代码。

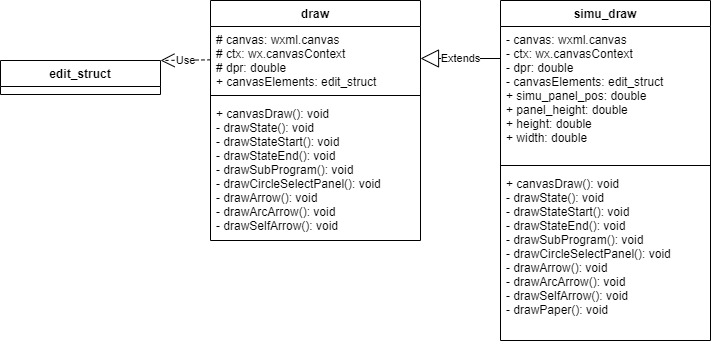


图 3.3.10 绘制逻辑类图

draw是在编辑界面使用的绘制逻辑类，而simu\_draw继承自draw，用于在模拟界面刷新图灵机的运行快照。两者的绝大多数的成员变量和方法都是private或者protected的，仅有canvasDraw是public方法。

在draw和simu\_draw共有的成员变量中，canvas是微信小程序页面的canvas模块；ctx是canvas上下文，主要绘制工作由该变量调用自身的方法来进行；dpr表示当前设备的像素比；canvasElements是edit\_struct实例，且是public变量，因为用户需要通过界面交互直接对该变量进行编写，从而达到编辑图灵机内容的目的。

simu\_draw中将canvasElements归为private变量，因为当前用户是无法更改该变量的。另外保留的simu\_panel\_pos作为public变量，是留给界面判断用户是否在移动纸带显示区的位置的（后面将解释为什么要移动这个）；panel\_height定义了纸带显示区的高度，因为多带图灵机由于存在多条纸带，纸带显示区的高度会有所变化；height和width则是记录了当前设备屏幕的高度和宽度信息，用于进行纸带的绘制。最后说明为什么用户需要移动纸带显示区：由于手机能提供的可视范围有限，如果用户编辑的图灵机在绘制区占据了大部分区域，那么在模拟器中，图灵机运行快照的显示区可能会遮蔽图灵机的主体（见图 3.3.11），如果用户想要在显示快照的同时，还能看到图灵机被遮蔽的部分，那么就必须要有一个移动的功能，可以将快照移动到其他位置，暴露出原来被遮蔽的部分。所以我们为其添加了simu\_panel\_pos变量用于存储用户当前触摸的y坐标，并且将移动事件绑定于tapMove函数上，在事件触发时，该函数会修改simu\_panel\_pos，并提醒simu\_draw进行绘制区刷新。

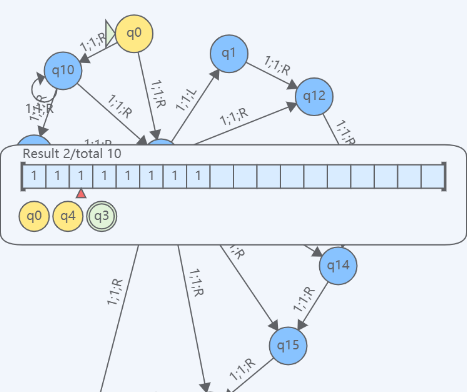


图 3.3.11 运行快照的显示遮蔽了图灵机主体绘制部分

接下来对绘制逻辑中的主要方法进行说明：

canvasDraw作为唯一的public函数，起到的其实是刷新绘制的作用，所有的绘制逻辑都汇总到该方法中，统一进行调用。

drawState用于绘制图灵机的普通状态。由于初始状态、接受/结束状态、子程序模块状态的绘制仅和普通状态有部分图形区别，且在设计上故意设计成互相不冲突的情况，所以在绘制状态时，drawState是绘制图灵机状态的基础方法。

drawStateStart用于绘制图灵机初始状态左侧的三角形，该三角形的设计是为了示意用户该状态是作为初始状态出现的。与drawState配合绘制图灵机初始状态。

drawStateEnd用于绘制图灵机接受状态/结束状态的小圆环，用于示意用户该状态为接受状态/结束状态。与drawState配合绘制图灵机的接受/结束状态。

drawSubProgram用于绘制图灵机的带子程序模块上，位于状态中间偏上的一个小组件，用于示意用户该状态包含了一个子程序。与drawState配合绘制图灵机的带子程序状态，具体的绘制效果如图 3.3.12，其中q7为初始状态，q11为结束状态，q9和q10均为包含子程序的特殊状态。

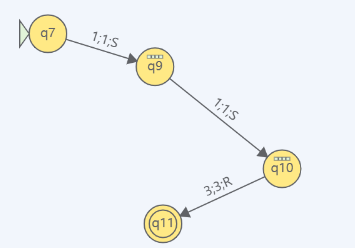


图 3.3.12 特别状态的绘制效果

drawCircleSelectPanel仅在编辑界面中会被触发，该绘制效果是在用户在选择模式下长按状态时显示的，在图 3.2.4中我们已经提到这个功能。该功能会对长按的状态绘制一个特殊的圆盘，具体绘制效果如图 3.3.13。用户在按住的情况下，左划或者右划接着松开，就会触发对应的初态/终态的设置和取消事件。这种圆盘的设计相对于弹窗让用户填写确认来说，一方面更方便用户进行操作，不会打断用户的操作过程，另外一方面，这种设计还会让用户产生一种科技感，增加使用的趣味性。

|  |  |  |  |
| --- | --- | --- | --- |
| （a）无特别状态 | （b）仅为初态 | (c) 仅为终态 | (d) 同为初态终态 |

图 3.3.13 长按显示的选择圆盘

drawArrow、drawArcArrow、drawSelfArrow三个方法，对图灵机的状态转移函数进行了渲染。这三个方法分别对应的绘制情况是：绘制直线箭头，绘制弧线箭头，绘制指向自己的（带圆环）箭头。具体绘制情况如图 3.3.14所示。直线箭头的绘制会在两个状态之间有且仅有其中一个状态指向另外一个状态的状态转移函数时调用；弧线箭头的绘制会在两个状态之间同时存在一方指向另外一方的状态转移函数时调用；而指向自己的箭头则在状态转移函数的起止状态都相同的情况下被调用。

|  |  |  |
| --- | --- | --- |
| （a）直线箭头 | （b）弧线箭头 | (c) 指向自己的箭头 |

图 3.3.14 各种箭头的绘制情况

drawPaper方法为simu\_draw的独有方法，该方法专门用于绘制图灵机的运行快照，该方法也被整合在simu\_draw.canvasDraw方法中，而用户在和界面交互时，其实是先调用了刷新函数，而刷新函数内部调用drawPaper，从而在视觉上营造出快照显示区的滑动效果的。同样，在图 3.3.15中，我们展示了drawPaper方法绘制出的所有情况下的图灵机运行快照。我们将运行中纸带的内容在条状格子表示的纸带上显示出来，显示区域的最上方为结果数量以及当前我们查看的结果索引；而读写头的位置则用红色的三角形在纸带下方标出；运行路径使用类似状态的绘制方式，按照顺序排列在显示区的最下方，且如果绘制的状态是接受状态，那么该状态会标以浅绿色。

另外值得一提的是，读写头在超过显示的格数时，显示区会对读写头的位置做取模运算，从而只取读写头位置的一段固定区间进行绘制，所以读写头的绘制总能保持在范围内。

|  |  |
| --- | --- |
| （a）单带图灵机 | （b）五条纸带的多带图灵机 |

图 3.3.15 图灵机运行快照绘制情况

### 核心算法设计与实现

在本节中，我们将对小程序中最为重要的核心算法：撤销/恢复算法、单带图灵机模拟算法、多带图灵机模拟算法、子程序图灵机模拟算法进行具体说明。

(1)撤销/恢复算法

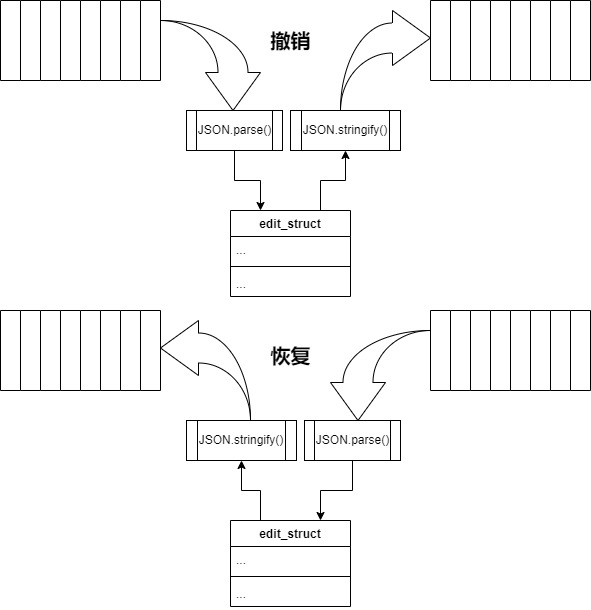


图3.3.16 撤销与恢复示意图

如图 3.3.16所示，撤销与恢复算法本身使用的数据结构其实是两个栈，在图中，左侧的栈对应的是撤销/恢复操作类中的ud，而右侧的栈对应的是rb。从示意图中我们可以看到，这个算法的核心部分在于将当前用户正在使用的edit\_struct看做为两个栈中间夹住的状态，ud和rb栈存储的是经过JSON.stringify()字符串化的edit\_struct，本质上是一种格式字符串，而撤销和恢复操作都是使用JSON.parse()将存储在ud和rb中的字符串解析为edit\_struct结构。其中撤销操作将ud栈中上一步保存的字符串解析，替换掉当前的edit\_struct，替换前，将现在的edit\_struct字符串化，存入rb栈中，而恢复操作则是正好相反。另外还有一个细节是，如果在撤销操作之后，进行了其他的编辑操作，那么rb栈会被清空，也就无法再恢复了。

this.undo=function(){

    if(ud.length==0)

        return;

    rb.push(JSON.stringify(canvasElements));

    canvasElements=JSON.parse(ud.pop());

}

this.rollback=function(){

    if(rb.length==0)

        return;

    ud.push(JSON.stringify(canvasElements));

    canvasElements=JSON.parse(rb.pop());

}

以上为undo操作和rollback操作的核心代码，可以看出代码的实现非常简洁，利用JSON的parse和stringify操作，我们就可以很巧妙实现撤回和恢复。当然，这样操作的前提是在每一步的编辑操作中，我们都必须严格将编辑前的图灵机模型保存到字符串，添加到ud栈中，另外为了防止栈中存储过多的内容导致手机性能下降，实际上该小程序对此设置了一个阈值，超过一定数量的内容会被清除，这个过程类似于队列，那么到了一定操作数之后，用户就不能一直撤回到最初的编辑状态了。

(2)单带与多带图灵机模拟算法

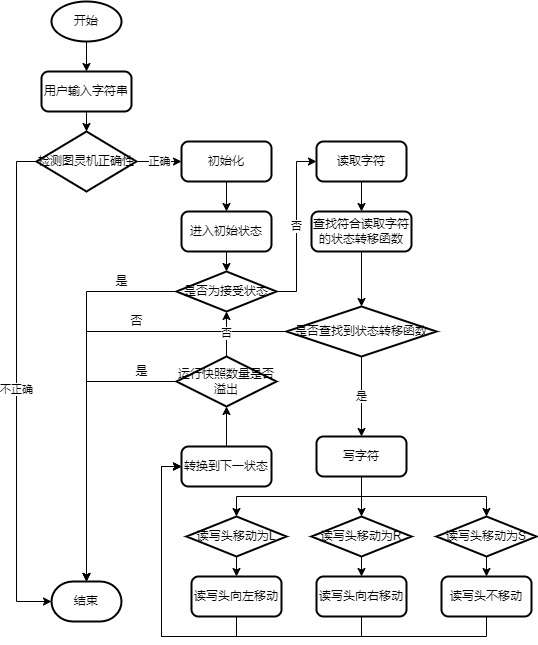


图 3.3.17 单带与多带图灵机模拟流程图

图 3.3.17展示了单带图灵机与多带图灵机的模拟流程，多带图灵机的模拟流程只是在读写字符和读写头移动时是以一组单位进行的，其余与单带图灵机流程完全一致。

在具体实现中，流程图中从读取字符到转换到下一状态并检测运行的快照数量被封装为单步执行的逻辑，而快速执行的逻辑实质上是对单步执行逻辑的反复调用，并且判断当前图灵机的运行是否存在异常，或者已经到达接受状态。

单步执行逻辑的具体实现如下。可以看到这里调用了图灵机实例的next方法，并且重置了绘制快照的下标为0，这个重置的目的在于防止下一步得到的快照数量小于当前的快照数量，而快照的索引下标没变的情况下可能会导致的异常错误。在重置快照下标后，canvasDraw被调用，绘制模块刷新页面中的图灵机运行快照。

nextStep: function() {

    if(!instance.isrunning()){

        wx.showToast({

            title: '模拟器未启动，输入待验证字符串以启动模拟器',

            icon: 'none',

            duration: 1500

        });

        return;

    }

    instance.next();

    result\_index=0;

    this.canvasDraw();

}

快速执行的逻辑如下。可以看到快速执行对很多条件都进行了判断，一共有三个条件会影响快速执行的主循环：运行步数、图灵机的运行状态、图灵机是否已接受。

运行步数的设计是出于对手机用户的考虑，在手机平台上，如果图灵机在一段过程中执行了过多的步骤，在用户的角度看来，那就是手机出现了严重的卡顿，另外如果图灵机存在死循环的情况无法停机，这对于小程序来说只能导致其崩溃，而目前我们可知是无法通过一般方法判断任意图灵机是否可停机的，所以这里使用count变量存储步数，超过一定阈值时，暂停快速执行，如果用户想要继续执行，再次点击快速执行按钮即可从中断位置继续执行。

图灵机的运行状态由图灵机提供的接口isrunning可以得到，该函数会在图灵机异常停止或者正常结束的情况下返回false值，此时快速执行会退出主循环。这个功能专门用于防止图灵机在快速执行中出现快照数量指数级增加的情况，指数级增加的快照数量对于手机用户来说没有任何的查看价值，只有对手机性能的极大的负面效果，该情况有可能是由于用户对图灵机的错误设计导致，也有可能是该图灵机本身就存在这样的特性，但是很抱歉，快照数量开始极速增长时，小程序必须切断执行过程。对于快照数量的判定和中止图灵机执行的逻辑，在后面的代码中我会指出。

图灵机在接受状态时，快速执行可以通过accept方法得知，此时快速执行会退出主循环并暂停，弹窗通知用户，当然在这之后用户也可以选择继续执行，因为很多图灵机的接受状态仍然存在状态转移函数转移到其他的状态，如果用户想继续执行，只需要再次点击快速执行按钮即可。

快速执行在主循环中不会调用canvasDraw来对运行过程进行刷新，原本的设计中考虑到这种刷新方法也许会带来一种动态效果，但是在实际测试中，这种效果并没有体现出来，那么为了防止不必要的渲染开销，最终我取消了这个设计。

fastRun: function() {

    if(!instance.isrunning()){

        wx.showToast({

            title: '模拟器未启动，输入待验证字符串以启动模拟器',

            icon: 'none',

            duration: 1500

        });

        return;

    }

    let count=0;

    while(instance.isrunning()){

        instance.next();

        result\_index=0;

        count+=1;

        if(count==200 || instance.accept())

            break;

    }

    if(count==200){

        wx.showToast({title:'执行次数过多,暂停',icon:'none',duration:1500});

    }else if(instance.accept()){

        wx.showToast({title:'有接受状态，暂停',icon:'none',duration:1000});

    }

    this.canvasDraw();

}

下面的代码来自单带图灵机的next方法，其中部分与核心算法无关的代码已经被移除。在该方法的最后我们可以看到一个对长度的判定，我设置的阈值为1024，在运行快照的数量大于1024时，图灵机就会强行中止。该异常在这里就已经弹窗报错，所以对于单步执行和快速执行来说，这个模块和封装逻辑是完全不冲突的。

另外一个值得关注的地方在状态转移函数的主执行过程段。在对读写头的位置进行移动后，我们还对读写头是否越界进行了判定，实际上图灵机的纸带默认是两侧无限延伸的，在初始化时，图灵机就已经将用户输入的字符串通过split方法切分为了字符数组。JavaScript中的数组是个很灵活的数据结构，可以通过自带的方法变成栈或者变成队列，这里我们实际上将其变成了一个前后扩展的连续表。当读写头小于0时（在这里实际上只能为-1），实际上读写头仍然被设置为0，但是图灵机从数组的前端加入了一个通配符。当读写头超过数组的末尾（实际上只能超过1格），那么图灵机就直接在末尾加入一个通配符，从而给用户一种纸带在两侧无限延伸的感觉。

this.next=function(){

    accepted=false;

    let vec=[];

    que.forEach(elem=>{

        let state=elem[0],p=elem[1];

        state.transfer.forEach(e=>{

            if(vec.length>=1024) return;

            let ptr=elem[2];

            if(ptr>p.length) return;

            if(e.read==p[ptr] || e.read=="ε"){

                let tmp=[...p];

                tmp[ptr]=e.write;

                if(e.move=="L") ptr--;

                else if(e.move=="R") ptr++;

                if(ptr<0){

                    tmp.unshift(null);

                    ptr=0；

                }else if(ptr>tmp.length){

                    tmp.push(null);

                }

                vec.push([e.to,tmp,ptr,elem[3]+":"+e.to.name]);

            }else{

                return;

            }

        });

    });

    if(vec.length==0){

        this.stop();

        return;

    }

    que=vec;

    que.forEach(elem=>{

        if(elem[0].isEnd)

            accepted=true;

    });

    if(que.length>=1024){

        wx.showToast({title:'结果数量溢出',icon:'error',duration:1000});

        accepted=false;

        this.stop();

    }

}

(3)子程序图灵机模拟算法

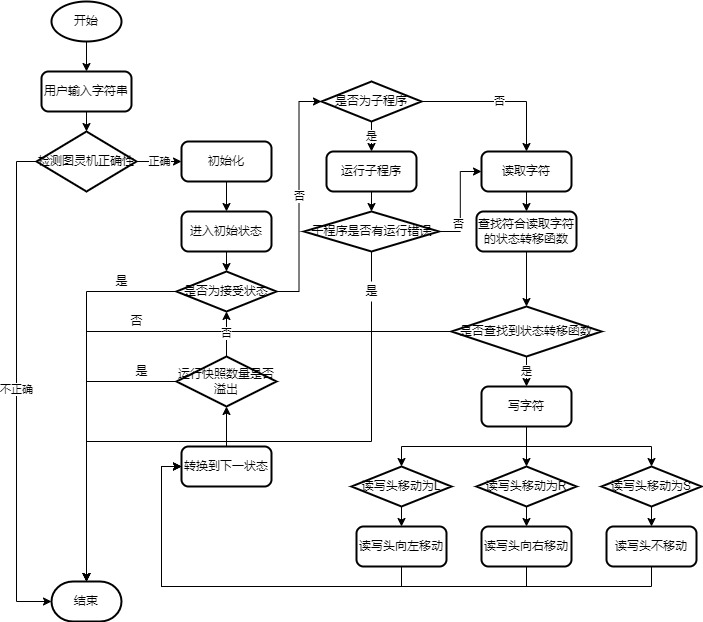


图 3.3.18 子程序图灵机模拟流程图

图 3.3.18展示的子程序图灵机的模拟流程与单带图灵机的模拟流程唯一的不同点在于子程序的判断和运行。子程序的运行会采用上文中与快速执行相同的逻辑，并且在执行结束后，子程序图灵机会获取该子程序模块的运行结果，如果子程序是异常结束的，那么整个子程序图灵机也会报错并终止模拟进程。当子程序正常运行完毕之后，子程序图灵机会继续进行和单带图灵机相同的状态转移函数的执行。

## 本章小结

这一章完整通过需求分析、概要设计与具体设计实现三个阶段，具体说明了经典图灵机仿真微信小程序的开发过程。在接下来的一章中，本文将简单说明该微信小程序的测试方法。

# 经典图灵机仿真微信小程序测试方案

上一章介绍了经典图灵机模型仿真APP的需求分析、概要设计、详细设计等内容，本章则介绍针对该APP进行的测试方案的设计。将从测试内容的角度和针对不同测试内容而进行的不同测试方案的设计进行描述[20]。

## 测试范围和测试内容

本次毕业设计是在移动Andorid平台上设计并实现的一个可视化的经典图灵机仿真APP，这个软件能够支持经典图灵机模型的建立、运行、验证等，并支持生成相应的JSON存储文件，能够导入、保存相应的图灵机模型。本软件的最终用户为具有一定形式语言与自动机理论知识的用户，用户通过使用该软件进行经典图灵机模型的仿真与学习。通过在第三章进行的分析与设计，我们可以确定本次的测试范围。

表 4.1.1是有关本测试的测试范围说明，基本与需求分析所确定的用例重合，包括了状态管理的测试，状态转移函数管理的测试，子程序状态管理的测试，混合状态转移函数的测试，图灵机仿真验证的测试，图灵机文件存储的测试。

表 4.1.1 测试范围说明

|  |  |  |  |
| --- | --- | --- | --- |
| 测试范围 | 主要内容 | 输入 | 输出 |
| 状态管理 | 测试绘制图灵机时自动机状态的创建、删除、重命名操作 | 用户人机交互 | 相应的功能输出 |
| 状态转移函数管理 | 测试绘制图灵机时转移函数的创建、删除、修改操作 | 用户人机交互 | 相应的功能输出 |
| 子程序状态管理 | 测试绘制带子程序图灵机时的子程序状态的创建、删除、修改等操作 | 用户人机交互 | 相应的功能输出 |
| 混合状态转移函数管理 | 测试绘制带子程序图灵机时混合状态转移函数的创建、删除、修改操作 | 用户人机交互 | 相应的功能输出 |
| 图灵机仿真 | 测试图灵机的仿真验证 | 有穷字符串、图灵机的Json文件、用户人机交互 | 相应的功能输出 |
| 图灵机文件管理 | 测试图灵机文件的保存，删除，修改操作 | 用户人机交互 | 相应的功能输出 |

为保证系统质量，经典图灵机模型仿真APP的测试过程划分为单元测试、集成测试、和系统测试三个级别。单元测试是基于代码级别的测试，集成测试进行代码单元和单元之间的交互测试，系统测试主要验证整个软件系统的功能、业务流程和性能等非功能属性是否符合用户需求。

根据经典图灵机模型仿真APP的功能特性及采用的开发技术，经典图灵机模型仿真APP的单元测试和集成测试是面向代码的测试，以自动化测试为主，由代码的开发人员承担，与编码活动同时展开，且如非必要不需要提供测试文档内容。系统测试是面向系统的整体测试，以手工测试为主。需提供测试计划、测试设计、测试用例等相关文档内容。本APP的测试流程如下

首先，对单带图灵机创建/修改模块中的状态管理中的状态创建、删除和重命名等功能进行测试，对状态转移函数管理中的创建、删除和修改等功能进行测试。为功能测试。

其次，对带子程序图灵机创建/修改模块中的子程序状态管理中的状态创建、删除、重命名等功能和子程序的选择修改功能进行测试，对混合状态转移函数管理中的创建、删除和修改等功能进行测试。为功能测试。

再次，对图灵机文件管理功能即数据持久化功能进行测试，对其文件的保存、打开和删除等操作进行测试。该测试为功能测试，同时也是数据文件存取的测试。

最后，对图灵机的仿真验证进行测试。先使用自动化工具进行测试，是单元测试，同时也包含了接口正确性测试。然后再对图灵机的仿真的交互流程进行测试，是功能测试。

表 4.1.2列举了经典图灵机模型仿真APP的测试项目及其对应的测试项标识前缀。

表 4.1.2 测试项目说明

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试项目 | 测试项目标识前缀 | 测试类型 |
| 1 | 界面功能 | TM\_UI | 功能测试 |
| 2 | 状态管理 | TM\_State | 功能测试 |
| 3 | 状态转移函数管理 | TM\_Tran | 功能测试 |
| 4 | 子程序状态管理 | TMBB\_State | 功能测试 |
| 5 | 混合状态转移函数 | TMBB\_Tran | 功能测试 |
| 6 | 图灵机文件管理 | TM\_File | 功能测试；数据文件存取测试 |
| 7 | 图灵机仿真 | TM\_Sim | 单元测试；功能测试 |

### 界面功能

表 4.1.3给出了该功能的测试用例表项,在界面功能测试中只简单的测试界面的显示布局是否正确，使用等价类划分的测试用例设计方法进行用例设计。详细测试将在每个具体功能的测试中进行。

表 4.1.3 界面功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 1 | TM\_UI\_Fun | TM\_UI\_Fun\_001 | 打开单带图灵机创建视图 |
| 2 | TM\_UI\_Fun\_002 | 打开多带图灵机创建视图 |
| 3 | TM\_UI\_Fun\_003 | 打开带子程序图灵机创建视图 |
| 4 | TM\_UI\_Fun\_004 | 打开验证图灵机视图 |
| 5 | TM\_UI\_Tool | TM\_UI\_Tool\_001 | 创建视图中工具栏的状态按钮 |
| 6 | TM\_UI\_Tool\_002 | 创建视图中工具栏的状态转移函数按钮 |
| 7 | TM\_UI\_Tool\_003 | 创建视图中工具栏的删除按钮 |
| 8 | TM\_UI\_Tool\_004 | 创建视图中工具栏的撤销按钮 |
| 9 | TM\_UI\_Tool\_005 | 创建视图中工具栏的恢复按钮 |
| 10 | TM\_UI\_Tool\_006 | 创建视图中的菜单按钮 |
| 11 | TM\_UI\_Tool\_007 | 创建带子程序图灵机视图中子程序状态按钮 |
| 12 | TM\_UI\_Tool\_008 | 创建带子程序图灵机视图中混合状态转移函数按钮 |
| 13 | TM\_UI\_File | TM\_UI\_File \_001 | 文件列表视图 |
| 14 | TM\_UI\_Set | TM\_UI\_Set\_001 | 系统设置视图 |
| 15 | TM\_UI\_Sim | TM\_UI\_Sim\_001 | 打开验证图灵机视图 |
| 16 | TM\_UI\_Sim\_002 | 设置验证图灵机的偏好设置的某一偏好 |
| 17 | TM\_UI\_Sim\_003 | 取消设置验证图灵机的偏好设置的某一偏好 |

本测试项关注经典图灵机模型仿真APP的界面功能是否满足用户需求。该APP的界面包括功能选单视图、图灵机创建视图、文件管理视图、系统设置视图、图灵机验证视图等组成部分。其中：

1．功能选单视图通过功能列表窗口实现，支持单带图灵机、多带图灵机、带子程序图灵机、验证图灵机功能的跳转选择。

2．图灵机创建视图，创建视图包括工具栏，菜单栏，绘制区；工具栏支持画布操作、状态管理操作、转移函数管理操作、删除操作、撤销操作、删除操作；创建子程序图灵机视图中的工具栏还包括子程序状态管理操作和混合状态转移函数管理操作。菜单栏支持保存文件、删除文件及仿真验证。绘制区支持用户进行图灵机的绘制操作。

3．文件管理视图通过已保存窗口实现，可以列出已保存的自动机文件名称及分类，并支持列表刷新与点击文件进入对应绘制界面以及在对应项目上左滑删除。

4．系统设置视图通过系统设置界面实现，可以选择空字符的表示、颜色偏好等。

5．图灵机验证视图通过图灵机验证界面实现，支持图灵机的缩略图显示，当前运行状态及输入与输出字符串的显示。

### 状态管理

表 4.1.4给出了状态管理功能的测试用例。该项测试对图灵机创建/修改视图中的状态管理功能进行测试，包括在绘制区新建状态，删除状态，对状态进行重命名，移动状态位置等功能的测试，并且在加入带子程序图灵机后对与子程序状态兼容实现进行了测试。使用等价类划分的测试用例设计方法进行用例设计。

表 4.1.4 状态管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 1 | TM\_State\_001 | 在绘制区新建状态 |
| 2 | TM\_State\_002 | 在绘制区重命名状态 |
| 3 | TM\_State\_003 | 在绘制区移动状态到另一位置 |
| 4 | TM\_State\_004 | 设置状态属性为初始状态 |
| 5 | TM\_State\_005 | 设置状态属性为接受状态 |
| 6 | TM\_State\_006 | 删除状态 |
| 7 | TM\_State\_007 | 创建带子程序图灵机时在状态管理状态下修改子程序状态的属性 |

### 状态转移函数管理

表 4.1.5给出了状态转移函数管理功能的测试用例。

表 4.1.5 状态转移函数管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 1 | TM\_Tran\_001 | 在绘制区选一个状态创建状态转移函数 |
| 2 | TM\_Tran\_002 | 改变在单独一个状态上创建的状态转移函数的角度 |
| 3 | TM\_Tran\_003 | 在绘制区的一个状态上创建2条状态转移函数 |
| 4 | TM\_Tran\_004 | 在绘制区选两个状态创建状态转移函数 |
| 5 | TM\_Tran\_005 | 在绘制区的两个状态之间创建2条状态转移函数 |
| 6 | TM\_Tran\_006 | 在绘制区修改状态转移函数属性 |
| 7 | TM\_Tran\_007 | 在绘制区移动状态函数关联的某一状态到另一位置 |
| 8 | TM\_Tran\_008 | 在状态转移函数的输入框不输入字符 |
| 9 | TM\_Tran\_009 | 在状态转移函数的输入框输入2字符 |
| 10 | TM\_Tran\_010 | 删除状态转移函数一端的状态 |
| 11 | TM\_Tran\_011 | 在只有一个状态转移函数的状态之间删除状态转移函数 |
| 12 | TM\_Tran\_012 | 在具有多个状态转移函数的状态之间删除状态转移函数 |
| 13 | TM\_Tran\_013 | 创建带子程序图灵机时在状态转移函数管理状态修改混合状态转移函数的属性 |

在该项测试中对图灵机创建/修改视图中的状态转移函数管理功能进行测试，包括在绘制区新建状态转移函数，删除状态转移函数，修改状态转移函数读写头数据，移动状态转移函数关联状态位置等功能的测试。并且在加入带子程序图灵机后对与混合状态转移函数的兼容实现进行了测试。使用等价类划分法和边界值法进行用例设计。

### 子程序状态管理

对带子程序图灵机中的子程序状态管理功能进行测试，包括在绘制区新建子程序状态，删除子程序状态，对子程序状态进行重命名，移动状态位置，选择子程序状态对应子程序图灵机文件，修改子程序状态对应子程序图灵机文件等功能的测试。使用等价类划分的测试用例设计方法进行用例设计。表 4.1.6给出了该功能的测试用例。

表 4.1.6 子程序状态管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 1 | TMBB\_State\_001 | 创建带子程序图灵机时在绘制区新建子程序状态 |
| 2 | TMBB\_State\_002 | 创建带子程序图灵机时在绘制区重新选择子程序状态 |
| 3 | TMBB\_State\_003 | 创建带子程序图灵机时在绘制区重命名子程序状态 |
| 4 | TMBB\_State\_004 | 创建带子程序图灵机时在绘制区创建重复的子程序状态 |
| 5 | TMBB\_State\_005 | 创建带子程序图灵机时在绘制区新建子程序状态后不选择对应的子程序 |
| 6 | TMBB\_State\_006 | 创建带子程序图灵机时设置子程序状态属性为初始状态 |
| 7 | TMBB\_State\_007 | 创建带子程序图灵机时设置子程序状态属性为终止状态 |
| 8 | TMBB\_State\_008 | 创建带子程序图灵机时删除子程序状态 |
| 9 | TMBB\_State\_008 | 创建带子程序图灵机时在子程序状态管理状态下修改状态属性 |

### 混合状态转移函数管理

对带子程序图灵机中的混合状态转移函数管理功能进行测试，包括在绘制区新建混合状态转移函数，删除混合状态转移函数，对混合状态转移函数的识别字符进行修改，移动混合状态转移函数关联状态位置等功能的测试。使用等价类划分法和边界值法进行用例设计。需要注意的是，要创建混合状态转移函数，只有一个被选中的状态为子程序状态，否则创建为状态转移函数。表 4.1.7给出了该功能的测试用例。

表 4.1.7 混合状态转移函数管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 1 | TMBB\_Tran\_001 | 在绘制区选一个子程序状态创建混合状态转移函数 |
| 2 | TMBB\_Tran\_002 | 改变在单独一个子程序状态上创建的混合状态转移函数的角度 |
| 3 | TMBB\_Tran\_003 | 在绘制区已有一条混合状态转移函数的子程序状态上创建一条状态转移函数 |
| 4 | TMBB\_Tran\_004 | 在绘制区选两个普通状态创建混合状态转移函数 |
| 5 | TMBB\_Tran\_005 | 在绘制区选两个子程序状态创建混合状态转移函数 |
| 6 | TMBB\_Tran\_006 | 在绘制区选一个普通状态和一个子程序创建混合状态转移函数 |

表 4.1.7（续） 混合状态转移函数管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 7 | TMBB\_Tran\_007 | 在绘制区已有一条混合状态转移函数的两个状态之间创建一条混合状态转移函数 |
| 8 | TMBB\_Tran\_008 | 在绘制区修改混合状态转移函数属性 |
| 9 | TMBB\_Tran\_009 | 在绘制区移动混合状态函数关联的某一状态到另一位置 |
| 10 | TMBB\_Tran\_010 | 在混合状态转移函数的输入框不输入字符 |
| 11 | TMBB\_Tran\_011 | 在混合状态转移函数的输入框输入2字符 |
| 12 | TMBB\_Tran\_012 | 删除混合状态转移函数一端的状态 |
| 13 | TMBB\_Tran\_013 | 在只有一个混合状态转移函数的状态之间删除混合状态转移函数 |
| 14 | TMBB\_Tran\_014 | 在具有多个混合状态转移函数的状态之间删除混合状态转移函数 |
| 15 | TMBB\_Tran\_015 | 创建带子程序图灵机时在混合状态转移函数管理状态修改状态转移函数的属性 |

### 图灵机文件管理

对图灵机数据持久化功能进行测试，包括文件列表的获取显示，文件列表的刷新，文件保存，文件的删除等功能的测试。使用等价类划分的测试用例设计方法进行用例设计。表 4.1.8给出了该功能的测试用例表项。

表 4.1.8 图灵机文件管理测试用例表

|  |  |  |
| --- | --- | --- |
| 序号 | 测试用例标识 | 测试用例说明 |
| 1 | TM\_File\_001 | 对一个创建的合法图灵机进行保存 |
| 2 | TM\_File\_002 | 对一个创建的不合法图灵机进行保存 |
| 3 | TM\_File\_003 | 保存时不输入文件名信息 |
| 4 | TM\_File\_004 | 打开保存的文件 |
| 5 | TM\_File\_005 | 删除保存的文件 |
| 6 | TM\_File\_006 | 打开文件进行修改后保存 |

### 图灵机仿真

对于图灵机验证模块的测试，首先要保证图灵机仿真验证的算法是正确的，这部分是单元测试的内容。然后便是更高层次的功能测试。

不管是单元测试还是功能，主要分为单带图灵机的仿真验证测试和带子程序图灵机的仿真验证测试，前一项的功能实现为后一项功能实现的基础，故两项测试的侧重点也有所不同。

安卓应用的测试不同于面向桌面系统软件的测试，一些安卓特有的类在Junit测试环境下是无法正确运行的，我们查阅了相关资料后采用了Robolectric工具来模拟测试所用的安卓环境[21]。Robolectric实现了一套能运行的Android代码的JVM环境，我们在执行Junit测试时，Robolectric会截取Android相关代码的调用，并使用其自己实现的代码去执行这个调用，从而达到在非Android环境运行Android测试代码的目的。

图 4.1.1给出了项目进行安卓应用自动化测试的环境结构图。

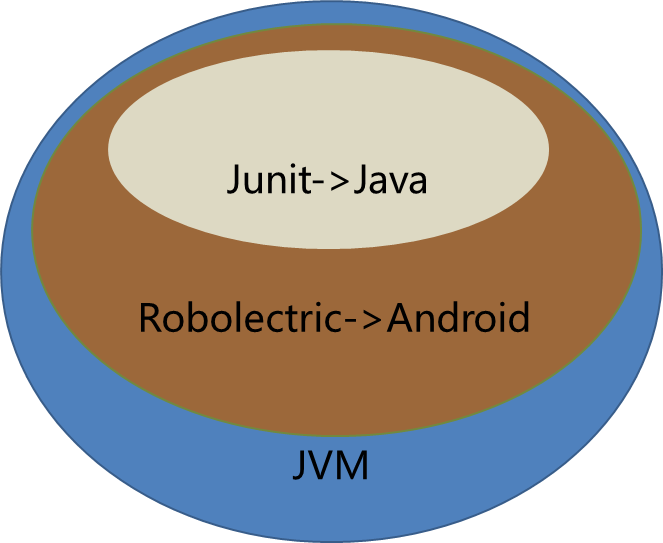


图 4.1.1 安卓测试环境

对于单带图灵机的仿真的单元测试，测试的重点在于不同偏好设置组合的复杂情况。图灵机接受字符串的出口有两种情况，一个为接受状态，一个为遇到停机状态。不同的偏好设置将直接影响仿真的结果，需要细致的考虑可能的情况。故而可采用黑盒测试法中的等价类划分法进行用例的设计，并使用白盒测试法中的条件覆盖来补充测试用例，用尽可能少的测试用例覆盖了四种偏好设置的十二种组合情况且尽可能多的覆盖了仿真程序中不同的条件分支。表 4.1.9给出了单带图灵机的仿真的单元测试的测试用例。

表 4.1.9 单带图灵机仿真单元测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 1 | TM\_Sim\_Fun\_TM | TM\_Sim\_Fun\_TM\_001 | 测试在选择允许在图灵机最终状态进行转换、在最终状态接受、在停止时接受、允许纸带在状态转移时暂停的偏好设置下的单带图灵机仿真 |
| 2 | TM\_Sim\_Fun\_TM\_002 | 测试在选择允许在图灵机最终状态进行转换、在最终状态接受、在停止时接受、的偏好设置下的单带图灵机仿真 |
| 3 | TM\_Sim\_Fun\_TM\_003 | 测试在选择允许在图灵机最终状态进行转换、在最终状态接受、允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 4 | TM\_Sim\_TM\_004 | 测试在选择允许在图灵机最终状态进行转换、在停止时接受、允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 5 | TM\_Sim\_Fun\_TM\_005 | 测试在选择允许在图灵机最终状态进行转换、在最终状态接受的偏好设置下的图灵机仿真 |
| 6 | TM\_Sim\_Fun\_TM\_006 | 测试在选择允许在图灵机最终状态进行转换的偏好设置下的图灵机仿真 |
| 7 |  | TM\_Sim\_Fun\_TM\_007 | 测试在选择允许在图灵机最终状态进行转换、在停止时接受的偏好设置下的图灵机仿真 |

表 4.1.9（续） 单带图灵机仿真单元测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 8 | TM\_Sim\_Fun\_TM | TM\_Sim\_Fun\_TM\_008 | 测试在选择在最终状态接受、在停止时接受、允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 9 | TM\_Sim\_Fun\_TM\_009 | 测试在选择在最终状态接受、允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 10 | TM\_Sim\_Fun\_TM\_010 | 测试在选择在停止时接受、允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 11 | TM\_Sim\_Fun\_TM\_011 | 测试在选择在最终状态接受、在停止时接受的偏好设置下的图灵机仿真 |
| 12 | TM\_Sim\_Fun\_TM\_012 | 测试在选择在停止时接受的偏好设置下的图灵机仿真 |
| 13 | TM\_Sim\_Fun\_TM\_013 | 测试在选择允许纸带在状态转移时暂停的偏好设置下的图灵机仿真 |
| 14 | TM\_Sim\_Fun\_TM\_014 | 测试在默认偏好设置下的图灵机仿真 |
| 15 | TM\_Sim\_Fun\_TM\_015 | 测试在默认偏好设置、输入字符集为空的情况下的图灵机仿真 |
| 16 | TM\_Sim\_Fun\_TM\_016 | 测试在默认偏好设置下输入错误的字符串的图灵机仿真 |

对于带子程序图灵机的仿真单元测试，测试的重点在于子程序状态所处的状态与混合状态转移函数类型的各种情况。图灵机的状态有三种类型，起始态，中间态和接受态。混合状态转移函数可以在普通状态与子程序状态或两个子程序状态之间创建，故有三种类型。故可采用等价类划分法与边界值法进行用例的设计，对子程序状态的三种类型和混合状态转移函数的三种类型的组合进行测试。并设计了一半正确的测试用例一半错误的测试用例来覆盖分支。表 4.1.10给出了用于带子程序图灵机单元测试的测试用例。

表 4.1.10 带子程序图灵机仿真单元测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 17 | TM\_Sim\_Fun\_TMBB | TM\_Sim\_Fun\_TMBB\_001 | 测试混合状态转移函数为从普通状态至子程序状态，且子程序状态为中间态，并且使用可正确验证的字符串作为输入。 |
| 18 | TM\_Sim\_Fun\_TMBB\_002 | 测试混合状态转移函数为从普通状态至子程序状态，且子程序状态为中间态，并且使用验证错误的字符串作为输入。 |
| 19 | TM\_Sim\_Fun\_TMBB\_003 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为中间态、一个为终止态，并且使用可正确验证的字符串作为输入。 |
| 20 | TM\_Sim\_Fun\_TMBB\_004 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为中间态、一个为终止态，并且使用验证错误的字符串作为输入。 |
| 21 | TM\_Sim\_Fun\_TMBB\_005 | 测试混合状态转移函数为从子程序状态至普通状态，且子程序状态为中间态，并且使用可正确验证的字符串作为输入 |
| 22 | TM\_Sim\_Fun\_TMBB\_006 | 测试混合状态转移函数为从子程序状态至普通状态，且子程序状态为中间态，并且使用验证错误的字符串作为输入 |

表 4.1.10（续）带子程序图灵机仿真单元测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 23 | TM\_Sim\_Fun\_TMBB | TM\_Sim\_Fun\_TMBB\_007 | 测试混合状态转移函数为从普通状态至子程序状态，且子程序状态为初始态，并且使用可正确验证的字符串作为输入。 |
| 24 | TM\_Sim\_Fun\_TMBB\_008 | 测试混合状态转移函数为从普通状态至子程序状态，且子程序状态为初始态，并且使用验证错误的字符串作为输入。 |
| 25 | TM\_Sim\_Fun\_TMBB\_009 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为起始态、一个为中间态，并且使用可正确验证的字符串作为输入。 |
| 26 | TM\_Sim\_Fun\_TMBB\_010 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为起始态、一个为中间态，并且使用验证错误的字符串作为输入。 |
| 27 | TM\_Sim\_Fun\_TMBB\_011 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为起始态、一个为终止态，并且使用可正确验证的字符串作为输入。 |
| 28 | TM\_Sim\_Fun\_TMBB\_012 | 测试混合状态转移函数为从子程序状态至子程序状态，且子程序状态一个为起始态、一个为终止态，并且使用验证错误的字符串作为输入。 |

图 4.1.2 带子程序图灵机仿真单元测试用例表

表 4.1.11给出了用于图灵机仿真功能测试的测试用例表。包括了单带图灵机的仿真和袋子程序图灵机的仿真的测试内容。

表 4.1.11图灵机仿真功能测试用例表

|  |  |  |  |
| --- | --- | --- | --- |
| 序号 | 测试用例标识 | | 测试用例说明 |
| 29 | TM\_Sim\_TM | TM\_Sim\_TM\_001 | 可被图灵机接受的字符串进行图灵机的仿真验证的情况 |
| 30 | TM\_Sim\_TM\_002 | 不可被图灵机接受的字符串进行图灵机的仿真验证的情况。 |
| 31 | TM\_Sim\_TM\_003 | 在某种偏好设置下可接受的字符串在其他偏好设置下进行图灵机仿真的情况 |
| 32 | TM\_Sim\_TM\_004 | 使用空字符串进行图灵机的仿真验证。 |
| 33 | TM\_Sim\_TM\_005 | 取消所有偏好设置进行图灵机的仿真验证 |
| 34 | TM\_Sim\_TMBB | TM\_Sim\_TMBB\_001 | 使用可被接受的字符串来测试带子程序图灵机正确仿真的情况。 |
| 35 | TM\_Sim\_TMBB\_002 | 仿真过程中拒绝的状态在非子程序状态的情况 |
| 36 | TM\_Sim\_TMBB\_003 | 仿真过程中拒绝的状态在子程序状态的内部状态的情况 |
| 37 | TM\_Sim\_TMBB\_004 | 测试带子程序图灵机的偏好设置是否应用于其中的子程序。 |

对于单带图灵机的仿真功能测试，测试的重点在于不同仿真情况的结果是否能在在仿真界面上正确显示，及图灵机的仿真偏好设置是否成功应用。这里采用了等价类划分方法进行了测试用例的设计。

对于带子程序图灵机的功能测试，测试的重点在于子程序状态。故测试时需围绕子程序状态展开，这里采用了等价类划分的测试用例设计方法，对考虑到的仿真过程中拒绝的状态在子程序状态的内部状态的情况，及偏好设置是否同时应用到了子程序中的情况进行了用例设计。

## 本章小结

本章对界面功能、状态管理、状态转移函数管理、子程序状态管理、混合状态转移函数、图灵机文件管理、图灵机仿真七个测试项目进行了测试用例的设计，分为了单元测试，集成测试，系统测试三个测试层级。灵活使用等价类划分法、边界值法进行测试用例的设计并使用条件分支覆盖的方法补充测试用例。用尽可能少的测试用例保证了软件的可靠性。

# 总结和展望

## 总结

本次毕业设计是在微信小程序平台上实现一个便于手机用户使用的图灵机仿真小程序。在整个项目的开发过程中，我完成了图灵机仿真小程序的图形界面设计与实现、文件系统设计与实现、编辑界面图形以及功能的设计与实现、图灵机模拟器界面的图形以及功能的设计与实现。其中图灵机的编辑与模拟模块包括不同的图灵机模型以及扩展：经典单带图灵机、多带图灵机，以及带有子程序模块的图灵机。

在实现图灵机的模拟功能上，该微信小程序主张让模拟的过程以及结果的展示更加适应手机用户的操作习惯，并且在此基础上还得保证有完善的模拟功能。所以此微信小程序的模拟功能中包括了单步执行和快速执行两种不同的执行模式，方便用户根据需求使用。

利用微信小程序的跨平台特点，该小程序可以与形式语言与自动机课程密切结合，学生无需携带电脑，只需要任意一部安装了微信的手机就可以使用该小程序进行实践操作。该小程序不仅可以直接用于进行课堂教学与课堂演示，同时也可以让学生参与其中自行操作，可视化的编辑过程与模拟过程以及现代化的UI设计使得实践过程更加有趣，并且更加利于学生直观理解形式语言与自动机课程中的相关概念。

## 展望

本次毕业设计是一次全新的尝试，将一个复杂的自动机模拟软件在一个较为受限的硬件平台上实现。就目前的情况来看，小程序的实现情况已经较为完善，但是仍然存在部分小细节还未实现完善。类似的细节问题会在后续的维护中逐步被修复。

一个软件的完成并不是它的终点，恰恰相反，这是它生命的起点。对于未来该小程序的维护和发展，我将首先采取个人维护的方式，对用户反馈以及自行测试暴露的软件问题进行修复，并且会收集用户意见，从而对该小程序的功能进行更新，使其更加方便用户使用。我希望能够让更多人参与到该项目的维护和更新中去，让自动机理论的教学更加轻松。所以我将在之后一段时间内将代码使用GPLv3协议在代码托管平台上开源，利用代码托管平台的各种功能，我们可以更加直接地与用户和开发者交流，从中得到软件的问题信息以及用户对软件功能的意见，让小程序的迭代进入良性循环。

参考文献

1. 钟先. 自动机与形式语言理论简介[J]. 机器人, 1980(01):30-38+56.
2. Chomsky N. Three models for the description of language. IRE Trans Inf Theor 2:113-124[J]. Information Theory, IRE Transactions on, 1956, 2(3):113-124.
3. Chomsky N. On certain formal properties of grammars[J]. Information and Control, 1959, 2(2):137-167.
4. 怀丽波,崔荣一,尹哲峰.形式语言与自动机理论课程教学方法探讨与实践[J].计算机教育,2019(06):106-108+116.
5. 杨光. "十二五"中国互联网发展十大亮点[J]. 计算机与网络,2015,41(21):6-7.
6. Carlos I. Chesñevar, María L. Cobo, William Yurcik. Using theoretical computer simulators for formal languages and automata theory[J]. ACM SIGCSE Bulletin,2003,35(2).
7. Pinaki Chakraborty, P. C. Saxena, C. P. Katti. Fifty years of automata simulation[J]. ACM Inroads,2011,2(4).
8. Tuhina Singh, Simra Afreen, Pinaki Chakraborty, Rashmi Raj, Savita Yadav, Dipika Jain. Automata Simulator: A mobile app to teach theory of computation[J]. Computer Applications in Engineering Education,2019,27(5).
9. Losacco M, Ttodger S H. Flap: a tool for drawing and simulating automata[J]. In ED-MEDIA 93, 1993.
10. Susan H. Rodger, Eric Wiebe, Kyung Min Lee, Chris Morgan, Kareem Omar, Jonathan Su. Increasing engagement in automata theory with JFLAP[J]. ACM SIGCSE Bulletin,2009,41(1).
11. Carlos H. Pereira, Ricardo Terra. A mobile app for teaching formal languages and automata[J]. Computer Applications in Engineering Education,2018,26(5).
12. 胡嵩. 图灵的机器思维思想初探[D].华中师范大学,2017.
13. Turing A M. On computable numbers, with an application to the Entscheidungsproblem[J]. J. of Mah,1936,58(345-363): 5.
14. 陈有祺. 形式语言与自动机[M]. 机械工业出版社, 2008: 148-168.
15. Hopcroft J E, Motwani R, Ullman J D. 自动机理论，语言和计算导论：第3版[M]. 机械工业出版社, 2008.
16. 安立新. 通用图灵机的计算机仿真设计[J]. 中国计量学院学报, 2008, 19.
17. 马 龙,梁意文. 图灵机模拟系统的设计与实现[J].计算机工程与应用, 2005,8:101-103.
18. 齐治昌,谭庆平,宁洪. 软件工程（第3版）[M]. 高等教育出版社, 2012: 67-220.
19. Susan H. Rodger, Bart Bressler, Thomas Finley, Stephen Reading. Turning automata theory into a hands-on course[P]. Computer science education,2006.
20. GLENFORD J. MYERS. 软件测试的艺术: 第2版[M]. 机械工业出版社, 2006: 23-72
21. Kochhar P S , F Thung, Nagappan N , et al. Understanding the Test Automation Culture of App Developers[C]// IEEE International Conference on Software Testing. IEEE, 2015.

致谢

首先感谢胡军老师给予我这个开发自动机相关模拟软件的机会。对于每个计算机学子而言，形式语言与自动机是计算机领域非常基础且重要的组成部分。在计算机领域各种项目充斥着人工智能的浮躁的当下，一个返璞归真的纯粹的自动机相关的项目，可以让我静下心来，仔细学习并理解计算机领域的其中一个最抽象的理论。虽然在本科学习过程中，我并没有选择该课程，但是在课余时间我参与了一项与形式语言紧密结合的项目：一个编程语言的高性能解释器项目，在边学边开发的过程中我也意识到了形式语言与自动机在计算机领域的重要性，同时也为后来开发该小程序打下了坚实基础，使得我的开发过程较为顺利。

同时也要感谢我的舍友庞志伟，他运用自己强悍且扎实的数学运算能力，在图灵机构建界面的绘图逻辑编写上帮助我实现了需要使用复杂数学计算的一种特殊状态转移函数绘制逻辑，使得界面的图形效果更加优美。

此外也要感谢在本科学习阶段教授我课程的老师们，我从各位老师的精妙讲解中学得了重要的计算机专业技能和专业知识，并且提高了相关的技能水平和自主学习能力，使得我可以在一个全新的项目的设计与实现过程中游刃有余，面对问题也可以自主解决，同时也让我有充分的知识储备可以针对性地开发出一个高性能的软件。