|  |  |
| --- | --- |
| 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Android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](#_Toc73994379)

[1.1 背景和意义 1](#_Toc73994380)

[1.2 国内外研究现状 1](#_Toc73994381)

[1.3 本文主要工作 2](#_Toc73994382)

[1.4 论文组织结构 3](#_Toc73994383)

[第二章 图灵机相关内容介绍 4](#_Toc73994384)

[2.1 图灵机简介 4](#_Toc73994385)

[2.2 子程序技术 6](#_Toc73994386)

[2.3 图灵机与计算机 8](#_Toc73994387)

[2.4 本章小结 8](#_Toc73994388)

[第三章 经典图灵机模型仿真微信小程序设计与实现 9](#_Toc73994389)

[3.1 需求分析 9](#_Toc73994390)

[3.1.1 确定功能及用例 9](#_Toc73994391)

[3.1.2 用例描述 11](#_Toc73994392)

[3.1.3 小结 13](#_Toc73994393)

[3.2 概要设计与界面实现 13](#_Toc73994394)

[3.2.1 项目整体架构说明及APP功能结构设计 13](#_Toc73994395)

[3.2.2 单带图灵机模块设计与界面实现 15](#_Toc73994396)

[3.2.3 带子程序图灵机模块设计与界面实现 16](#_Toc73994397)

[3.2.4 仿真验证模块设计与界面实现 18](#_Toc73994398)

[3.2.5 数据持久化模块设计 19](#_Toc73994399)

[3.2.6 小结 20](#_Toc73994400)

[3.3 详细设计与实现 20](#_Toc73994401)

[3.3.1 项目环境 20](#_Toc73994402)

[3.3.2 关键类设计实现 20](#_Toc73994403)

[3.3.3 关键算法设计与实现 25](#_Toc73994404)

[3.4 本章小结 28](#_Toc73994405)

[第四章 经典图灵机模型仿真微信小程序测试方案 30](#_Toc73994406)

[4.1 测试范围和测试内容 30](#_Toc73994407)

[4.1.1 界面功能 31](#_Toc73994408)

[4.1.2 状态管理 33](#_Toc73994409)

[4.1.3 状态转移函数管理 33](#_Toc73994410)

[4.1.4 子程序状态管理 34](#_Toc73994411)

[4.1.5 混合状态转移函数管理 34](#_Toc73994412)

[4.1.6 图灵机文件管理 35](#_Toc73994413)

[4.1.7 图灵机仿真 35](#_Toc73994414)

[4.2 本章小结 39](#_Toc73994415)

[第五章 总结和展望 40](#_Toc73994416)

[5.1 工作总结 40](#_Toc73994417)

[5.2 工作展望 40](#_Toc73994418)

[参考文献 41](#_Toc73994419)

[致谢 42](#_Toc73994420)

# 绪论

本章从研究背景、研究意义以及国内外现状等方面介绍了基于Android平台的经典图灵机模型仿真软件。并将本文的主要内容梳理出一个简要的组织结构。

## 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

本文围绕着本次毕业设计所实现的经典图灵机模型仿真APP软件展开讨论。首先是介绍图灵机的理论知识及一些常用的图灵机的程序设计技术，正确理解该部分内容是展开后续工作的前提。然后是对经典图灵机模型仿真APP进行需求分析并为该软件所依托的项目的整体结构进行一个阐释。确定了该软件所需实现的功能模块，包括状态管理、转移函数管理、图形化的仿真，文件管理等。除了经典的单带图灵机，同时也实现了使用一种子程序技术的扩展图灵机。接着针对主要类设计和关键算法的实现进行详细设计说明。最后说明是如何针对安卓应用展开该软件的测试以保证软件是足够可靠的。

## 论文组织结构

本论文的组织结构如下：

第一章 从研究背景、研究意义以及国内外现状等方面对课题进行介绍，阐明本文的主要工作；

第二章 对图灵机的理论知识及一种常用的图灵机的程序设计技术：子程序技术进行介绍；

第三章 对经典图灵机仿真APP进行需求分析和概要设计，并围绕着主要类和关键算法进行详细设计说明,并在此过程中介绍了界面实现和关键内容的实现；

第四章 介绍了对经典图灵机仿真APP的测试策略和测试说明；

第五章 对本次毕业设计的工作做出总结和展望。

# 

# 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。

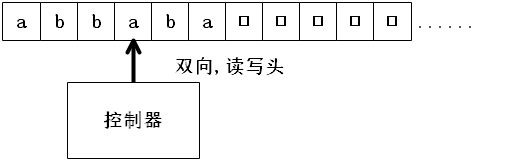


图 2.1.1 图灵机模型示意图

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。

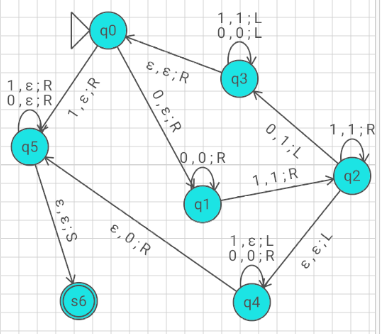


图 2.1.2 图灵机转换图

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 |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## 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。

# 经典图灵机模型仿真微信小程序设计与实现

本章围绕经典图灵机模型仿真APP的需求分析、概要设计及详细设计展开，为了方便描述，将经典单带图灵机简称为单带图灵机，使用子程序技术的扩展单带图灵机简称为带子程序图灵机。本章第一节讲述了该APP应该具有的功能，从单带图灵机及带子程序图灵机的构建，图灵机的仿真验证，图灵机的文件管理等方面进行分析。第二节介绍了该APP所依托的项目的整体架构并说明了在此基础上进行的有关该APP的概要设计,并对一些设计进行了界面实现。第三节则是围绕着详细设计展开，将对关键的算法的实现和主要的类进行描述。

## 需求分析

### 确定功能及用例

本节主要从软件需要具有的功能角度进行需求分析，根据给定的任务书确定了四个基本功能模块，分别是单带图灵机模块、带子程序图灵机模块、图灵机仿真验证模块、图灵机文件管理模块。下面将对这四个功能模块进行详细的说明。

单带图灵机模块要能够实现单带图灵机的创建、修改功能，主要包括：

（1）界面功能，要支持图形化的创建界面，需要具有良好的人机交互功能，至少具有绘制区和工具栏，同时支持用户在绘制区绘制的元素可以移动位置；

（2）状态管理，需要支持图灵机状态的创建，删除和重命名、图灵机的初始状态的设置与取消、图灵机的接受状态的设置与取消；

（3）状态转移函数管理，需要支持图灵机的状态转移函数的创建，修改与删除、能设置和修改读写头的读写的数据及读写头移动的方向、可在状态上创建转移到自身的状态转移函数（自环型状态转移函数）、可在两个状态之间创建状态转移函数。

带子程序图灵机模块要能够实现该扩展单带图灵机的创建、修改功能， 除了经典单带图灵机模块所需的界面功能、状态管理功能、状态转移函数管理功能之外，还需要包括：

（1）子程序状态管理，需要支持子程序状态的创建，删除和重命名、子程序图灵机的选择、设置子程序图灵机为初始状态或取消、设置子程序图灵机为接受状态与取消；

（2）混合状态转移函数管理，需要支持从一个主程序图灵机的开始状态转移到子程序状态及从子程序状态转移到主程序图灵机的返回状态或子程序状态间的转移的混合状态转移函数的创建，修改与删除、支持在子程序状态上转移到自身的混合状态转移函数的创建，修改与删除。

图灵机仿真验证模块要能够读取保存的已创建完成的图灵机并进行仿真验证，主要包括：

（1）能识别所选图灵机的类型并显示图灵机的转换图；

（2）用户可输入字符串作为输入字符集进行图灵机的仿真；

（3）根据用户偏好及用户输入字符串对选取的图灵机进行仿真验证，图灵机仿真停机后显示当前图灵机仿真状态的快照，并提示图灵机停机的详细原因。

图灵机文件管理模块要实现图灵机的一些文件操作，主要包括：

（1）支持将单带图灵机转化为JSON对象，以JSON文件的形式保存在磁盘中；

（2）支持将带子程序图灵机转化为JSON对象，并以JSON文件的形式保存在磁盘；

（3）支持将已保存的图灵机文件以列表的形式显示，除了文件名以外还需要显示如图灵机类型、保存时间等必要的文件属性。

（4）要支持从文件列表选择图灵机并跳转到对应的图灵机创建修改界面。

（5）要支持从文件列表选择图灵机并跳转到对应的图灵机仿真验证界面。

（6）要支持文件的删除

通过以上的功能需求分析可以得到如图 3.1.1所示的经典图灵机模型仿真APP的用例图。在下一节将对这些用例中较为复杂的用例进行详细的描述。

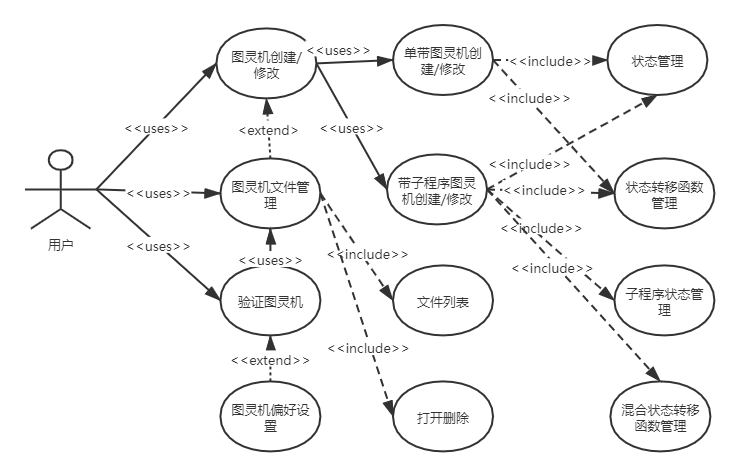


图 3.1.1 经典图灵机模型仿真APP用例图

### 用例描述

在本节将对3.1.1中所确定的用例中较为复杂的用例：单带图灵机的创建/修改用例和带子程序图灵机的创建/修改用例进行详细的说明，包括用例的用例描述、用例的触发事件、用例的主事件流等内容。

图 3.1.2给出了单带图灵机创建/修改用例的详细用例描述。

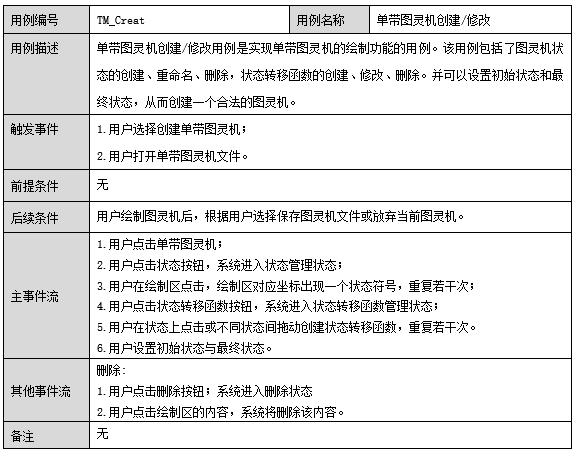


图 3.1.2 TM\_Creat用例描述

该用例是实现单带图灵机的绘制功能的用例，编号为TM\_Creat。其触发事件有两个，其一是用户选择创建单带图灵机，另一个是用户从文件中打开一个已保存的单带图灵机。当用户将图灵机构建完毕后可以选择保存图灵机文件或放弃当前图灵机。在该用例中，用户的主要操作为：用户点击单带图灵机；用户点击状态按钮，系统进入状态管理状态；用户在绘制区点击，绘制区对应坐标出现一个状态符号，重复若干次；用户点击状态转移函数按钮，系统进入状态转移函数管理状态；用户在状态上点击或不同状态间拖动创建状态转移函数，重复若干次。用户设置初始状态与最终状态。此外还有用户删除绘制内容操作。

图 3.1.3是带子程序图灵机创建/修改用例的详细用例描述。该用例是实现带子程序图灵机的绘制功能的用例，编号为TMBB\_Creat。其触发事件有两个，其一是用户选择创建带子程序图灵机，另一个是用户从文件中打开一个已保存的带子程序图灵机。该用例正常运行的前提条件是用户创建图灵机所使用的子程序图灵机为单带图灵机且所要使用的子程序图灵机已经正确保存为JSON文件。当用户将图灵机构建完毕后可以选择保存图灵机文件或放弃当前图灵机。

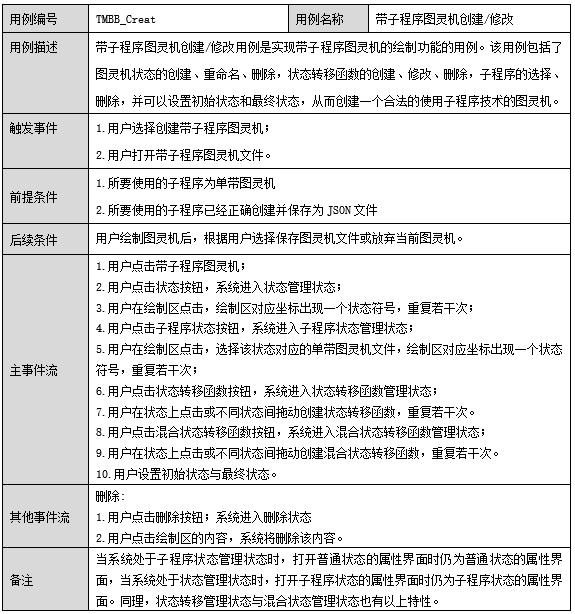


图 3.1.3 TMBB\_Creat用例描述

在TMBB\_Creat用例中，用户的主要操作与TM\_Creat用例类似，但是新增了如下操作：用户点击子程序状态按钮，系统进入子程序状态管理状态；用户在绘制区点击，选择该状态对应的单带图灵机文件，绘制区对应坐标出现一个状态符号，重复若干次；用户点击混合状态转移函数按钮，系统进入混合状态转移函数管理状态；用户在状态上点击或不同状态间拖动创建混合状态转移函数，重复若干次。

此外，在进行TMBB\_Creat用例的分析过程中，着重考虑了该用例在TM\_Creat用例基础上增加的功能与原有共同的相互支持。如当系统处于子程序状态管理状态时，打开普通状态的属性界面应该如何显示的问题。

### 小结

在3.1节中我们从功能角度进行了需求分析，确定了经典图灵机模型仿真APP的基本用例，并选取了其中较为复杂的用例进行了详细的用例描述。这些内容是后续的设计的支撑。本文在3.2节将进行概要设计的说明，在3.3节将进行详细设计的说明。

## 概要设计与界面实现

在本节将进行概要设计，把需求分析阶段确定的用例进行设计，最终得到经典图灵机模型仿真APP的逻辑模型。在此阶段，部分功能的设计参考了JFLAP的图灵机仿真模块的功能的实现[19]。

### 项目整体架构说明及APP功能结构设计

经典图灵机模型仿真APP是依托形式语言与自动机仿真APP项目进行开发的。该项目经过开发已经具备了基本的框架结构和基本的交互操作，并且实现了有穷自动机模型的仿真与上下文无关文法和下推自动机模型的仿真。

图 3.2.1显示了项目的整体架构。从图中可以看到本项目分为用户交互层、数据处理层、中间层、存储层。

用户交互层负责显示页面、接受用户的交互输入、与数据处理层传递数据等，在整个项目开发过程中，用户交互流程的设计是极为重要的内容。

数据处理层则包括了一些功能算法、自动机的抽象实现，并提供了与其他层级进行数据交互的接口。

中间层最主要的作用是实现将自动机转化为JSON对象，以及将JSON对象实例化为自动机，在进行文件的保存、读取时发挥作用。

存储层则是将JSON对象以JSON文件的形式保存在磁盘上实现数据持久化。

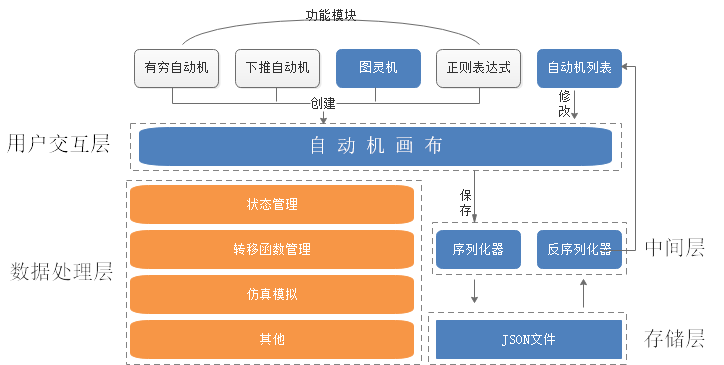


图 3.2.1 形式语言与自动机仿真APP项目架构图

在需求分析中我们对功能进行了分析，得到了实现经典图灵机模型仿真APP的用例。图 3.2.2是这些功能模块之间互相调用关系的一个活动图，从图中可以看到，作为一个可视化的APP，所有的功能都与用户交互层相关。此外，图灵机的绘制与仿真均需要数据处理层的参与，文件处理操作也均需中间层和存储层同时参与。

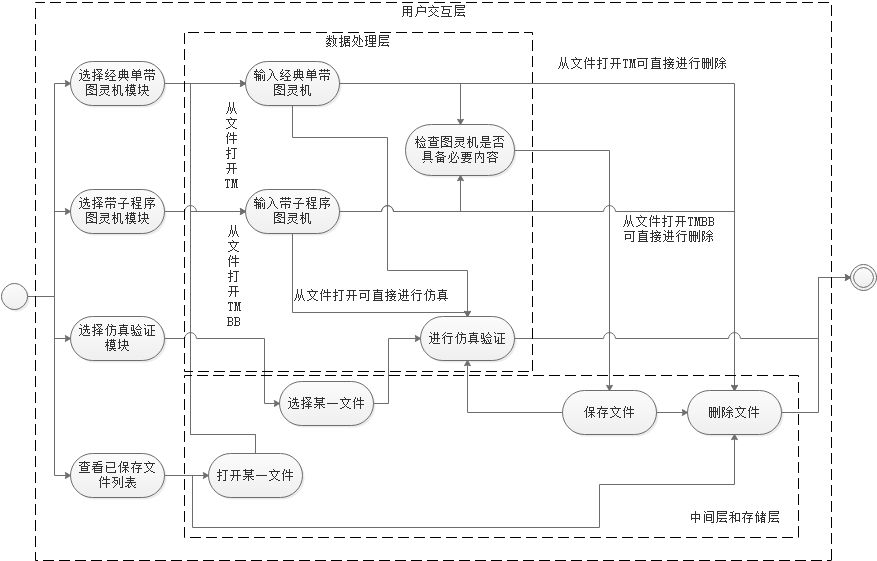


图 3.2.2 功能活动图

在本小节中，我们对项目整体架构及APP功能结构设计进行了说明，在本章的后续几节中，我们将对4个功能模块逐一进行说明。

### 单带图灵机模块设计与界面实现

从需求分析我们知道，单带图灵机的创建/修改需要支持用户创建、删除状态及状态转移函数，并且可以对其属性进行修改。该模块除了具有通过用户交互构建图灵机的功能之外也需要支持用户对图灵机进行文件操作、进行仿真验证等。此外还需要具有相应的错误处理功能，当用户进行错误的操作时可以终止操作并进行提示。

图 3.2.3是该模块的活动图。可以看到用户在绘制图灵机时可以同时地，重复地执行画布操作、状态绘制操作、状态转移函数绘制操作、撤销绘制操作和恢复绘制操作。并且具有打开文件、保存文件、删除文件、仿真验证的功能。当用户选择打开文件功能时将显示所有该类型图灵机文件的文件列表，然后用户通过点击选择文件打开，打开后将刷新绘制画布。删除文件需要确保当前图灵机已保存为文件，否则将进行错误提示。保存文件需要确保用户绘制的图灵机是至少具有初始状态的，否则进行错误提示。仿真验证需要保证用户的图灵机文件已经保存，否则进行错误提示。

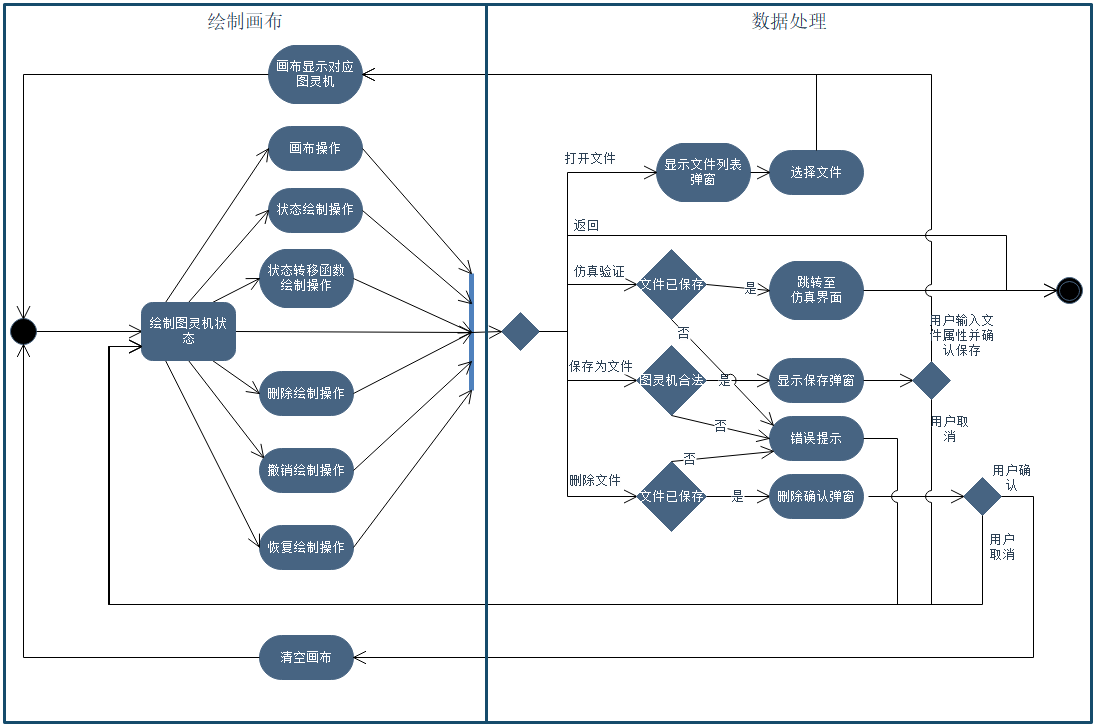


图 3.2.3 单带图灵机模块活动图

在绘制图灵机时地六种操作对应着图 3.2.4工具栏的六个按钮。当选中画布操作按钮时，将可以对画布进行移动和缩放。当选中状态管理按钮时，可以添加状态，修改状态属性，移动状态位置。当选中状态转移函数管理按钮时，则可以添加状态转移函数，修改状态转移函数属性，改变自环型状态转移函数的角度。当选中删除操作按钮时，点击绘制画布上的元素就可以将该元素删除。撤销操作按钮和恢复操作按钮无法选中，通过点击这两个按钮可以执行撤销和删除操作。



图 3.2.4 单带图灵机绘制工具栏

图 3.2.5和图 3.2.6分别是状态属性弹窗和状态转移函数属性弹窗。在状态属性弹窗中支持用户修改选中状态的状态名称及设置选中状态为初始状态或接受状态。在状态转移函数属性弹窗中，第一行将显示选中状态转移函数的出发状态和目的状态，后续几行中将支持用户设置读写头中读字符和写字符及读写头移动方向。L为左移、R为右移、S在不同的偏好设置中可以理解为停机或暂停。

|  |  |
| --- | --- |
|  |  |
| 图 3.2.5 状态属性弹窗 | 图 3.2.6 状态转移函数属性弹窗 |

### 带子程序图灵机模块设计与界面实现

从需求分析我们知道，单带图灵机的创建/修改需要支持用户创建、删除状态及状态转移函数，并且可以对其属性进行修改，带子程序图灵机模块在此基础上还需要支持用户创建、修改和删除子程序状态及混合状态转移函数。同样的，该模块除了具有通过用户交互构建图灵机的功能之外也需要支持用户对图灵机进行文件操作、进行仿真验证等操作以及进行错误处理。图 3.2.7是该模块的活动图。与图 3.2.3进行对比可以清楚地看到，该模块地活动图主要在绘制画布上有所区别，增加了子程序状态绘制操作和混合状态转移函数绘制操作。

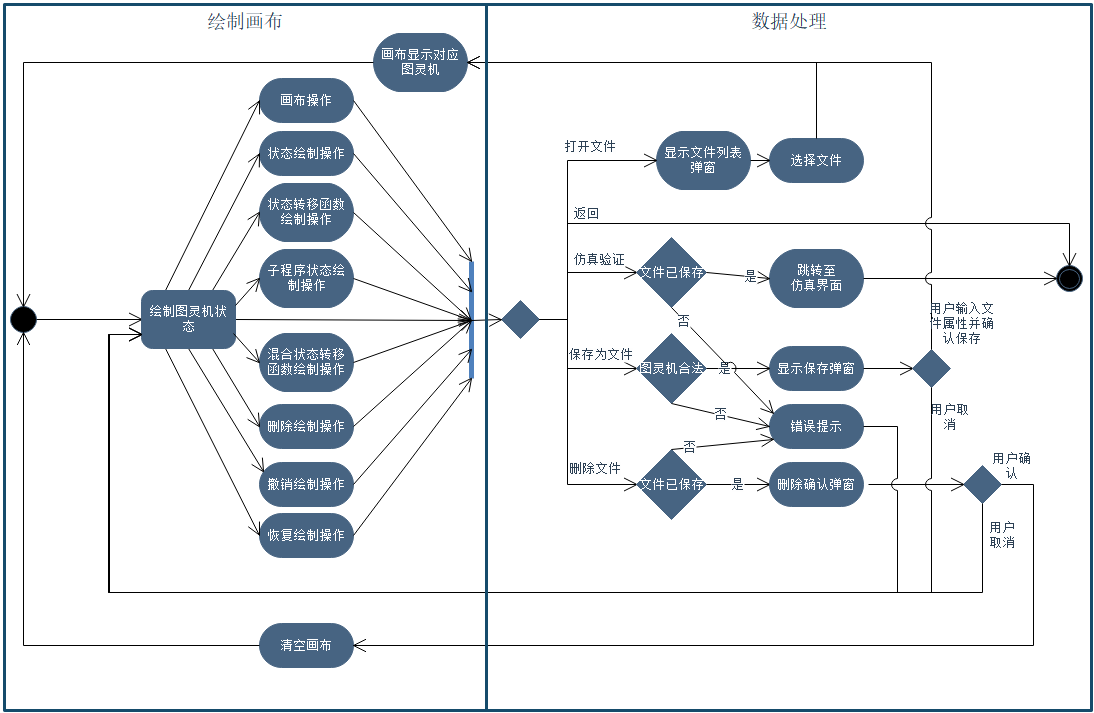


图 3.2.7 带子程序图灵机模块活动图

图 3.2.8是带子程序图灵机绘制工具栏，增加了子程序状态管理按钮和混合状态转移函数管理按钮。当选中子程序状态管理按钮时可以添加、修改子程序状态，选择子程序图灵机，移动子程序状态位置，并且与状态管理按钮的功能相互兼容。当选中混合状态转移函数管理按钮时可以在开始状态与子程序状态、子程序状态与子程序状态、子程序状态与返回状态之间创建一条转移及修改函数属性，并且与状态转移函数按钮功能兼容。

|  |  |
| --- | --- |
| （a）子程序状态管理按钮 | （b）混合状态转移函数管理按钮 |
|  | |

图 3.2.8 带子程序图灵机绘制工具栏

图 3.2.5和图 3.2.6分别是子程序状态属性弹窗和混合状态转移函数属性弹窗。在子程序状态属性中除了支持用户修改选中状态的状态名称及设置选中状态为初始状态或接受状态外还支持子程序图灵机的选择。在该过程中先读取单带图灵机的文件列表，然后处理为菜单候选列表，接着作为下拉选项供用户选择。在混合状态转移函数属性弹窗中，第一行将显示选中混合状态转移函数的出发状态和目的状态并支持用户输入字符进行识别。

|  |  |
| --- | --- |
|  |  |
| 图 3.2.9 子程序状态属性弹窗 | 图 3.2.10 混合状态转移函数属性弹窗 |

### 仿真验证模块设计与界面实现

在图 3.2.2中可以看到进入仿真验证操作地方式有两种，一种是通过图灵机地绘制界面跳转到仿真验证界面，一种是选择仿真验证功能模块后选择图灵机文件跳转到仿真验证界面。不论采用哪种方式选择图灵机进行仿真，都将执行如图 3.2.11所示的处理过程。其中，第五步至最后一步可以在仿真过程中循环进行。

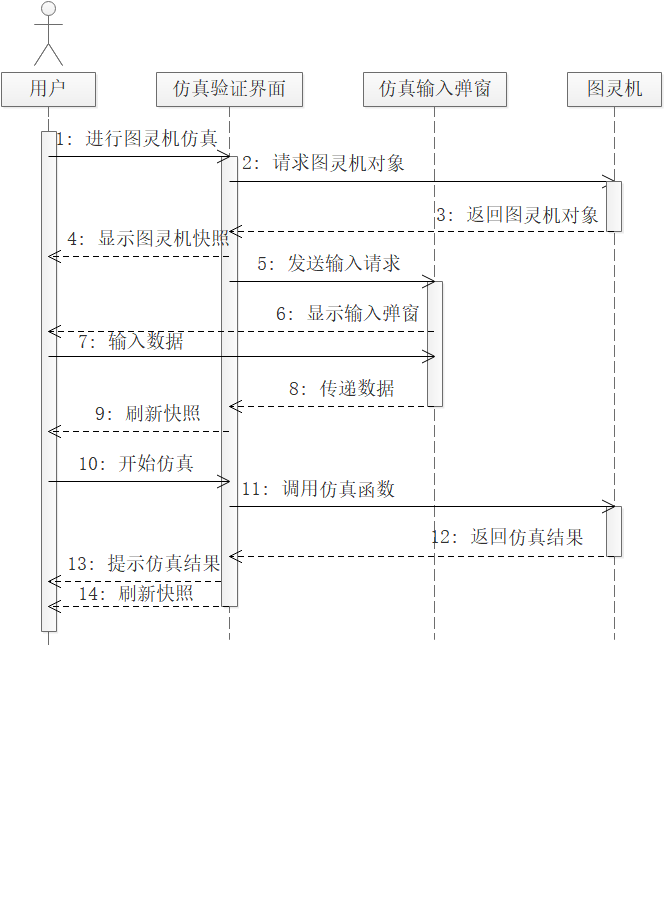


图 3.2.11 图灵机仿真时序图

图 3.2.12和图 3.2.13显示了仿真界面和仿真数据输入框的界面设计。仿真界面分为三部分，从上到下依次为图灵机快照、仿真详情、操作栏。仿真数据输入框支持输入仿真字符串和设置图灵机偏好。

|  |  |
| --- | --- |
|  |  |
| 图 3.2.12 仿真界面 | 图 3.2.13 仿真数据输入框 |

### 数据持久化模块设计

图 3.2.14是经典图灵机模型仿真APP文件存储系统的数据流图，包括了读取、保存、删除等主要操作的数据流向。

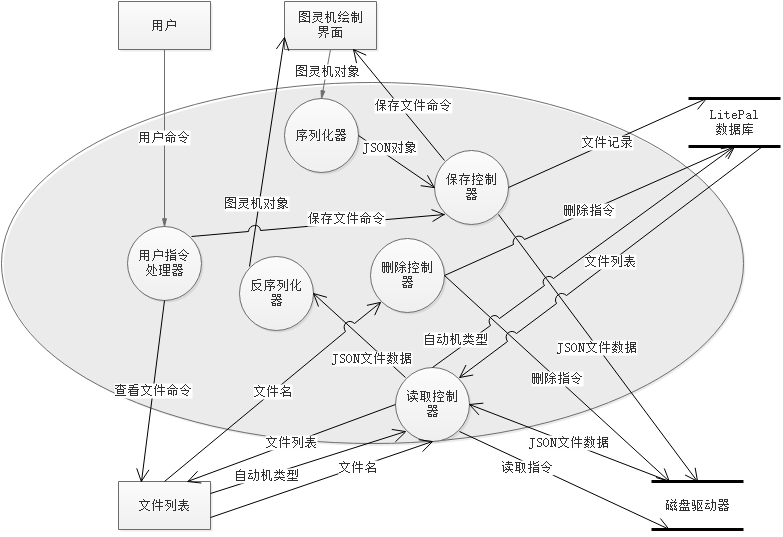


图 3.2.14 APP的“1”级数据流图

图灵机文件存储模块采用了两级存储结构，使用LitePal数据库存储用户输入的文件索引信息，使用磁盘存储自动机文件以实现数据持久化。使用数据库存储文件索引信息的优点是支持快速的多条件的进行文件目录索引，使用磁盘存储自动机文件的优点是简单方便。如果只使用数据库来实现数据持久化，那么复杂的自动机数据结构及不同实体间的关联关系将导致数据库表的数据过于复杂且充斥着很冗余信息。如果只使用磁盘来实现数据持久化，那么每次索引目录都要进行一次读盘操作，将大大降低APP的运行速度。

### 小结

在3.2节中我们对项目的结构进行了说明，在需求分析的基础上划分了主要的功能模块，包括单带图灵机模块、带子程序图灵机模块、仿真验证模块、数据持久化模块。使用了大量的UML模型图对设计的主要内容进行了说明。通过本节内容的描述，经典图灵机模型仿真APP已经具备了基本框架和流程。详细的设计实现在3.3节将进行说明。

## 详细设计与实现

在本节将进行详细设计内容的说明，包括关键类的设计与实现，关键算法设计与实现。

### 项目环境

本次毕业设计开发的经典图灵机模型仿真APP开发环境所使用的操作系统为微软Windows 10操作系统；所使用的编程语言为Java语言并且其JDK（Java Development Kit）的版本为1.8；所使用的开发工具是 Android Studio。本APP目标运行的SDK（Software Development Kit）版本为29，可兼容运行的最低SDK版本为26，当低于此SDK版本时软件可靠性将下降且会出现难以预料的软件缺陷。

### 关键类设计实现

在本小节将对关键类的实现进行说明。图 3.3.1是所依托的形式语言与自动机仿真APP项目中的自动机类的设计。

Automaton类中定义了有关自动机实现所必须的属性，包括多个状态和多个状态转移函数，一个初始状态，多个接收状态。并提供了对这些属性进行操作的方法。State类中定义了有关自动机状态的属性，包括该状态的名称、该状态所属的自动机及当前状态在画布中的坐标，并提供了一些必要的操作。Transition类中定义了有关状态转移函数的属性，包括出发状态，目的状态以及输入字符，并提供了一些必要的操作。

这三个类实现了一个自动机的基础框架，但要实现经典图灵机模型仿真APP还需要在继承这些类的基础上继续进行设计，并使用JAVA语言实现。

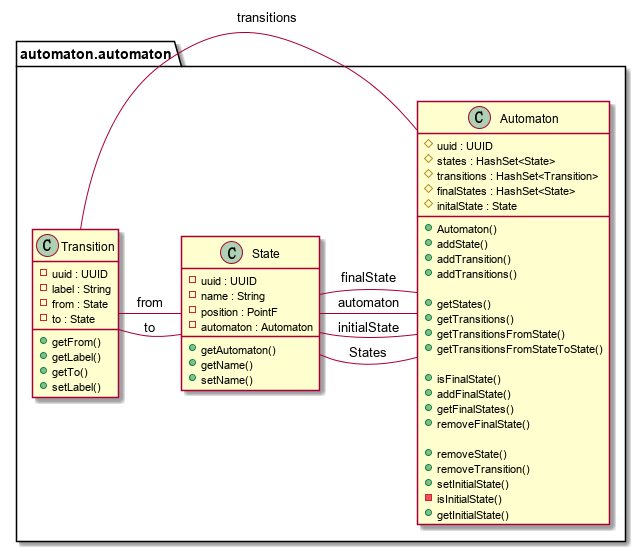


图 3.3.1 自动机数据结构设计

#### 单带图灵机

为了实现单带图灵机特有的操作，我们定义了一个继承于Automaton类的TurningMachineAutomaton。根据图灵机的定义，图灵机的状态转移函数需要包括：读头内容，写头内容以及读写头的移动方向，故定义了一个继承于Transition类的TurningMachineTransition类来实现。图 3.3.2是该模块主要类的设计类图。

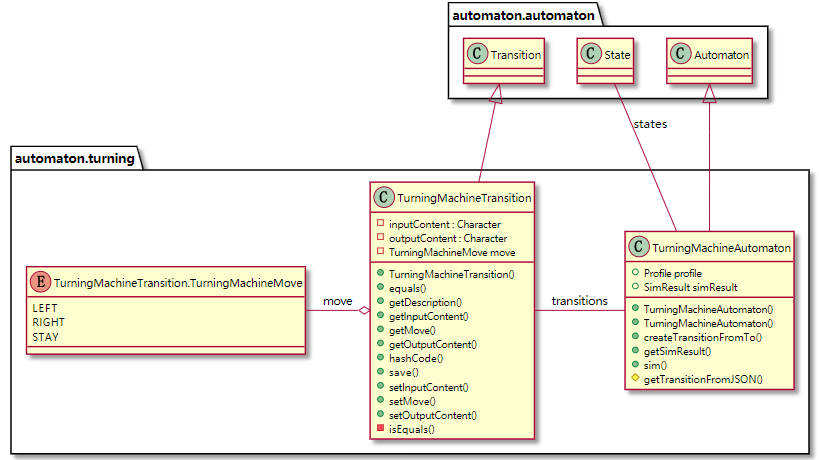


图 3.3.2 单带图灵机类图

表 3.3.1显示了TuringMachineAutomaton类的内部变量。在父类基础上增加了偏好设置属性和仿真结果属性。

偏好设置的数据类型为自定义类Profile，其中包括了四个图灵机的偏好设置：允许在图灵机接受状态进行转换、在接受状态接受、在停机时接受、允许纸带在状态转移时停机。这四个属性均为布尔类型。仿真结果的数据类型为自定义类SimResult，该属性将作用于图灵机仿真过程，用来保存图灵机仿真过程中的纸带内容，停机详情及是否接受输入纸带。并在该类中提供了获取这些内容的接口。

表 3.3.2显示了在TuringMachineAutomaton类中添加的或重构的类操作。

TurningMachineAutomaton为构造函数，在该类中我们设置了两个接受不同输入的构造函数。当输入为空时将执行初始化程序构造一个空的单带图灵机。当输入为JSON对象时将执行反序列化器恢复一个已保存的单带图灵机。另外注意，在此过程中，反序列化器在处理转移函数时将使用在本类中定义的getTransitionFromJSON函数进行操作。该函数输入为JSON对象，接着该函数将对其中的“from”、“to”、“input”、“output”、“move”标签进行提取，然后返回一个实例化的TuringMachineTransition对象。

createTransitionFromTo为创建状态转移函数的实现操作，该函数的输入为两个State类，代表状态转移函数的起止状态，创建成功后将返回一个实例化的Transition对象。

getSimResult是获取图灵机仿真结果的函数，该函数不需要输入参数，直接调用即可。当图灵机仿真结束之后可以通过此函数获取包含仿真结果的SimResult对象。

sim是实现图灵机仿真算法的关键函数，该算法将在3.3.3节进行详细说明。

表 3.3.1 TuringMachineAutomaton类变量

|  |  |  |  |
| --- | --- | --- | --- |
| 数据名称 | 标识 | 可见性 | 数据类型 |
| 偏好设置 | profile | public | Profile |
| 仿真结果 | simResult | public | SimResult |

表 3.3.2 TuringMachineAutomaton类操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 操作标识 | 操作名称 | 可见性 | 返回值 |
|  | TurningMachineAutomaton | 打开图灵机 | public | TurningMachineAutomaton |
|  | createTransitionFromTo | 新建状态转移函数 | public | Transition |
|  | getSimResult | 获取仿真结果 | public | SimResult |
|  | sim | 图灵机仿真 | public | SimResult |
|  | getTransitionFromJSON | 反序列化转移函数 | public | Transition |

表 3.3.3显示了TuringMachineTransition类的类变量，包括读头字符inputContent、写头字符outputContent、以及移动方向move。inputContent和outputContent均为Character类型，只能存储一个字符；move为自定义的TurningMachineMove类型，TurningMachineMove为一个枚举器，包括左移LEFT、右移RIGHT、停机STAY三个取值类型。

TuringMachineTransition类中虽然实现的操作但大部分比较简单，这里选择一个重要的类操作进行说明。save是实现序列化转移函数操作的函数，在该函数中定义了“from”、“to”、“input”、“output”、“move”五个标签，“from”标签用于存储转移函数的出发状态，“to”标签用于存储转移函数的目标状态，“input”标签用于存储读头数据，“output”标签用于存储读头数据，“move”标签用于存储读写头移动方向。最后返回一个JSON对象。

表 3.3.3 TuringMachineTransition类变量

|  |  |  |  |
| --- | --- | --- | --- |
| 数据名称 | 标识 | 可见性 | 数据类型 |
| 读头字符 | inputContent | private | Character |
| 写头字符 | outputContent | private | Character |
| 移动方向 | move | private | TurningMachineMove |

#### 带子程序图灵机

图 3.3.2是带子程序图灵机模块主要类的设计类图，从图中可以看到，我们定义了继承于TurningMachineAutomaton类的Turning MachineWithBuildBlocksAutomaton类实现带子程序图灵机。定义了继承于State类的TMBBState类实现子程序状态。定义了继承于Transition类的TurningMachineWithBuild-BlocksTransition类实现混合状态转移函数，由于该类只简单修改了Transition类中的构造函数和序列化操作，在后文将不在赘述。

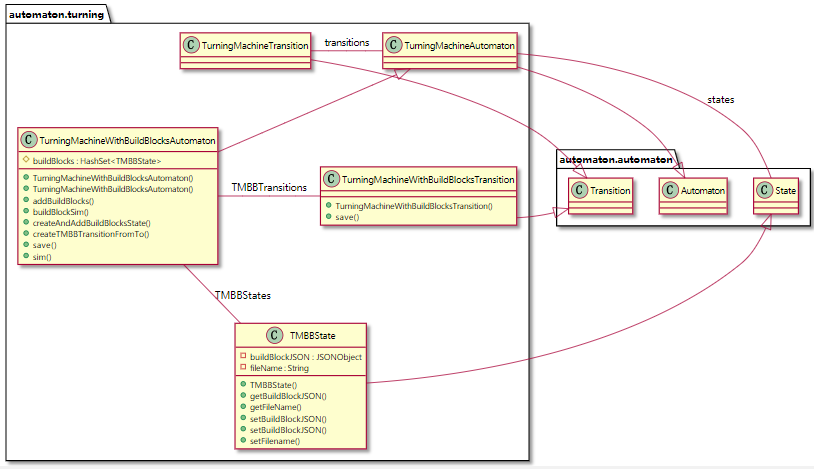


图 3.3.3 带子程序图灵机类图

表 3.3.4显示了TurningMachineWithBuildBlocksAutomaton类中添加的变量，该变量表示子程序状态表，是由TMBBState类型的数据构成一个哈希表。该表与States表的逻辑关系为被包含关系。

表 3.3.5显示了TurningMachineWithBuildBlocksAutomaton类中的类操作。首先是构造函数TurningMachineWithBuildBlocksAutomaton，与单带图灵机类中的构造函数设置相同，带子程序图灵机类也有两种构造函数，支持空的带子程序图灵机的构建和接受JSON对象输入并执行反序列化器恢复一个已保存的带子程序图灵机。另外，从表 3.3.4中可以发现，我们只为TMBBState创建了一个新的哈希表与State哈希表进行区分，却没有为TurningMachineWithBuildBlocksTransition创建一个新的哈希表。这是因为TurningMachine-WithBuildBlocksTransition与TurningMachineTransition虽然都继承了Transition类但是其中的所有属性未被TurningMachineTransition对象完全使用，故我们可以采用这些属性进行一个区分，并执行不同的实例化操作

表 3.3.4 TurningMachineWithBuildBlocksAutomaton类变量

|  |  |  |  |
| --- | --- | --- | --- |
| 数据名称 | 标识 | 可见性 | 数据类型 |
| 子程序状态表 | buildBlocks | protected | HashSet<TMBBState> |

表 3.3.5 TurningMachineWithBuildBlocksAutomaton类操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 操作标识 | 操作名称 | 可见性 | 返回值 |
|  | TurningMachineWith-  BuildBlocksAutomaton | 打开图灵机 | public | TurningMachineWith-  BuildBlocksAutomaton |
|  | addBuildBlocks | 新建子程序状态 | public | void |
|  | buildBlockSim | 子程序仿真 | public | boolean |
|  | createAndAdd-BuildBlocksState | 新建并添加子程序状态 | public | State |
|  | createTMBBTransitionFromTo | 新建混合状态转移函数 | public | Transition |
|  | save | 使用序列化器保存图灵机 | public | JSONObject |
|  | sim | 仿真验证 | public | SimResult |

createAndAddBuildBlocksState函数是相应子程序状态操作的函数，接受的输入为PointF点坐标对象。当接受输入后先实例化一个TMBBState对象，接着调用addState函数将该状态添加到State表中，然后再调用addBuildBlocks将子程序状态添加到BuildBlocks列表中。

createTMBBTransitionFromTo为创建混合状态转移函数的实现操作，该函数的输入为两个State类，代表状态转移函数的起止状态，创建成功后将返回一个实例化的Transition对象。

save函数是重新定义的序列化器，通过该序列化器可以将带子程序图灵机序列化为JSON对象。sim和buildBlockSim是实现图灵机仿真算法的关键函数，该算法将在3.3.3节进行详细说明。

表 3.3.6显示了子程序状态类TMBBState的类变量，表 3.3.7显示了子程序状态类TMBBState的类操作，在设置存储子程序图灵机的JSON对象时，是通过文件操作工具根据设定的文件名读取文件内容为JSON对象。子程序状态类的操作都较为简单，在此不过多赘述。

表 3.3.6 TMBBState类变量

|  |  |  |  |
| --- | --- | --- | --- |
| 数据名称 | 标识 | 可见性 | 数据类型 |
| 子程序名 | fileName | private | String |
| 子程序图灵机的JSON对象 | buildBlockJSON | private | JSONObject |

表 3.3.7 TMBBState类操作

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 序号 | 操作标识 | 操作名称 | 可见性 | 返回值 |
|  | setBuildBlockJSON | 设置存储子程序图灵机的JSON对象 | public | void |
|  | getBuildBlockJSON | 获取存储子程序图灵机的JSON对象 | public | void |
|  | setFilename | 设置子程序图灵机对应的文件名 | public | void |
|  | getFileName | 获取子程序图灵机对应的文件名 | public | void |

### 关键算法设计与实现

在该小节将对单带图灵机的仿真sim操作的算法设计和带子程序图灵机的仿真sim操作和buildBlockSim操作的算法设计进行详细描述。

#### 数据元素：

我们将对仿真过程中所用到的数据元素但还未进行说明的部分进行定义。

首先，我们知道图灵机的纸带是无限延长的，如果采用数组进行定义，则不能体现出“延长”这一特性，故我们将字符串数组转化为链表来进行模拟，该链表的数据结构如下：

public static class TapItem {

    protected Character c;

    protected TapItem pre;

    protected TapItem next;

}

public static class TapItemList {

    protected TapItem begin;

    protected TapItem end;

    protected TapItem cursor;

}

在上文提到，我们定义了一个SimResult类来进行仿真结果的存储，其中包含了仿真是否成功、转换后的字符串、仿真结果详细信息、仿真后所处状态等信息，下面我们给出该类的数据结构：

public static class SimResult {

    protected boolean isAccept;

    protected String result;

    protected String simInfo;

    protected State currentState;

    protected List<String> steps;

}

#### 单带图灵机仿真算法

图 3.3.4 单带图灵机仿真活动图给出了单带图灵机仿真算法的流程图。

从图中可以看到，在用户输入仿真字符串后，程序便开始了对偏好设置是否设置接受条件、图灵机是否合法、用户输入是否合法的检查，当以上条件均通过后便开始从链表的表头读一个字符串然后从初始状态开始进入循环操作。

在进入循环后首先判断当前状态是否处于接受状态，如果处于接受状态则判断偏好设置是否允许从接收状态继续进行转移，如果不允许就退出仿真函数，否则判断纸带是否为空，如果空的则退出仿真函数，否则查找以当前状态为出发状态的状态转移函数中读头数据inputContent等于当前字符的状态转移函数。如果找到了多个状态转移函数或者未找到对应的状态转移函数，便转到错误处理流程并退出仿真函数。

在找到了唯一对应的状态转移函数之后，将状态转移函数的写头数据outputContent替换纸带上的当前字符，然后根据转移方向move的属性L或R移动读写头位置并读一个字符，对应到链表上的操作则是转到pre表项或next表项中并使当前字符为该表项中的字符，然后转换到状态转移函数的目的状态。如果转移方向为S则需要先判断偏好设置是否为在停机时接受，如果否则退出仿真函数。否则继续判断是否允许中途停机，如果不允许则转到错误处理流程并退出仿真函数，否则读取刚刚替换后的字符并转换到状态转移函数的目的状态，返回循环起点。

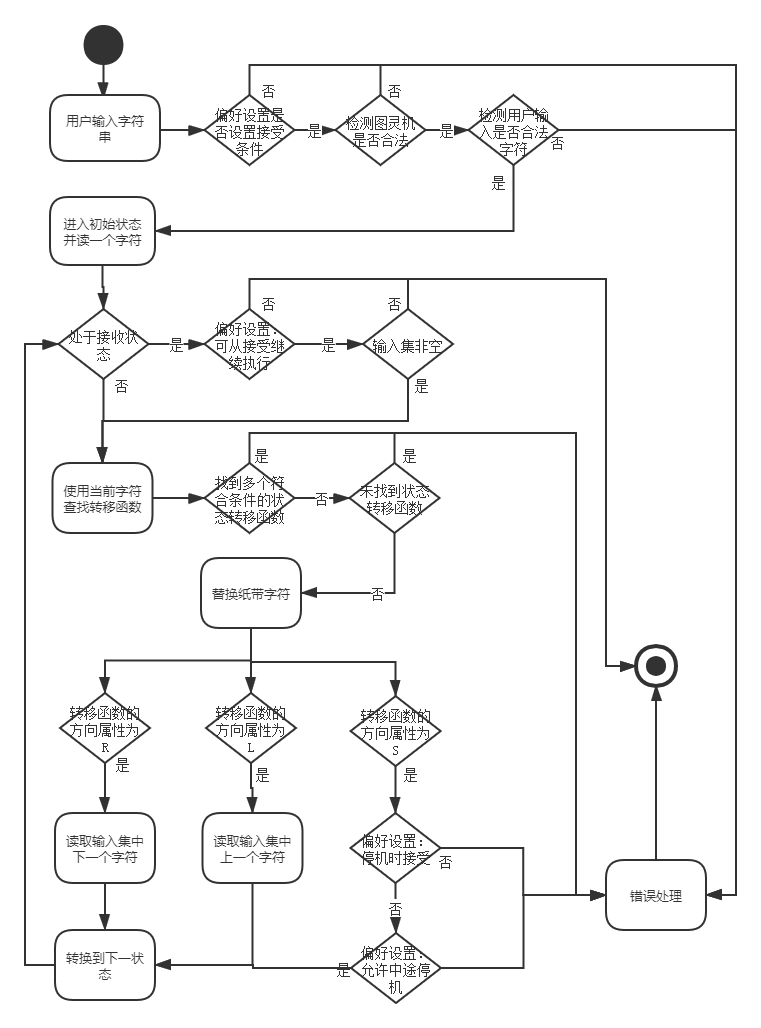


图 3.3.4 单带图灵机仿真活动图

#### 带子程序图灵机仿真算法

图 3.4.1 带子程序图灵机仿真活动图是带子程序图灵机的仿真，在该仿真流程中的大部分操作与单带图灵机的仿真函数一致。图中框选部分为新增的子程序状态的操作流程和混合状态转移函数的操作流程，下面将进行详细的描述。

首先是子程序状态的操作流程，在程序读取了纸带第一个字符从初始状态进入循环后，第一步操作改为遍历buildBlocks表查询当前状态是否为子程序状态，如果不是则执行原来的循环步骤。否则使用getBuildBlockJSON方法获取子程序图灵机的JSON文件，然后将其反序列化构造一个图灵机，如果构造失败则转入错误处理流程并退出仿真。否则使用buildBlockSim方法对子程序图灵机进行仿真，该验证流程与单带图灵机的仿真流程一致，但返回值为boolean类型。子程序验证失败将转入错误处理流程并退出仿真，成功则将执行原来的循环步骤。

混合状态转移函数的操作流程则较为简单，当识别到状态转移函数的类别为混合状态转移函数时，预读取下一字符但是不改变读写头的位置，然后转移到目的状态结束本次迭代，返回循环起点。

## 本章小结

在本章，我们按照软件生存周期的先后顺序，在第一节对经典图灵机模型仿真APP进行了需求分析，确定了主要的用例。在第二节对APP进行了概要设计，对APP的逻辑结构和流程进行了说明。在第三节对APP的实现进行了详细的设计和说明，至此APP开发阶段的主要任务已经完成。在下一节中将进行APP的测试工作。

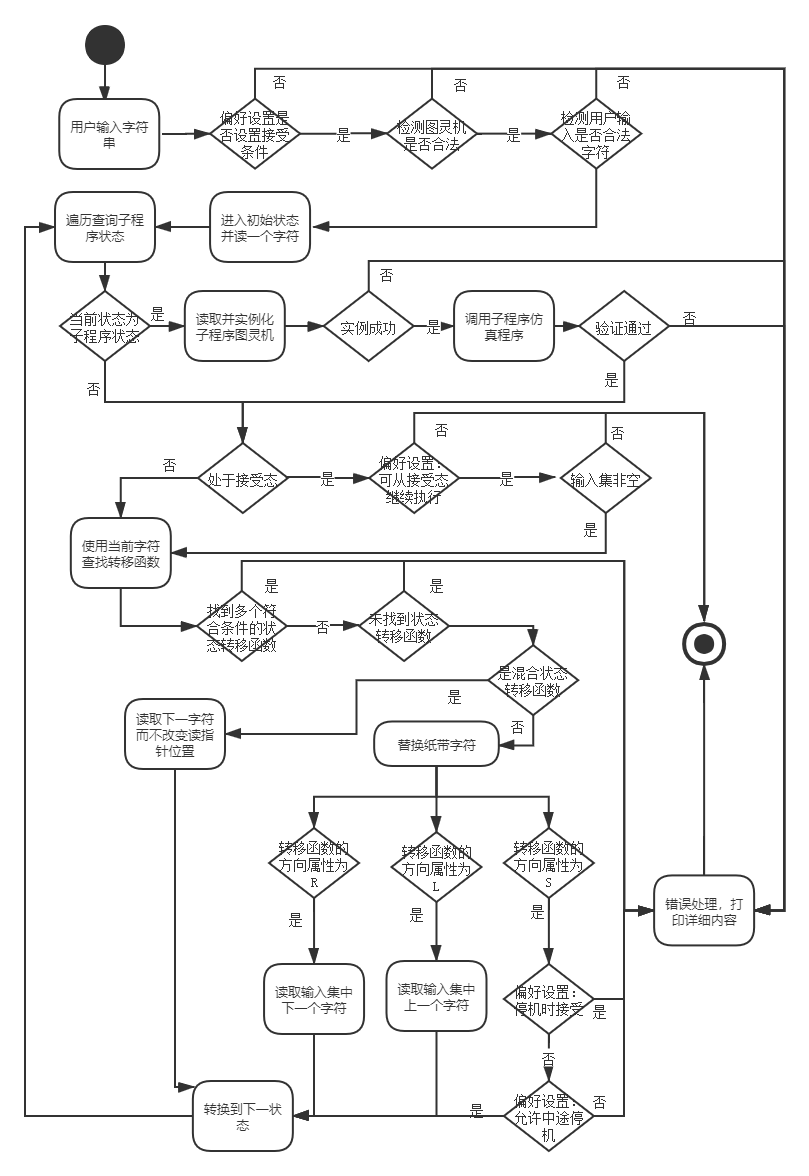


图 3.4.1 带子程序图灵机仿真活动图

# 经典图灵机模型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。

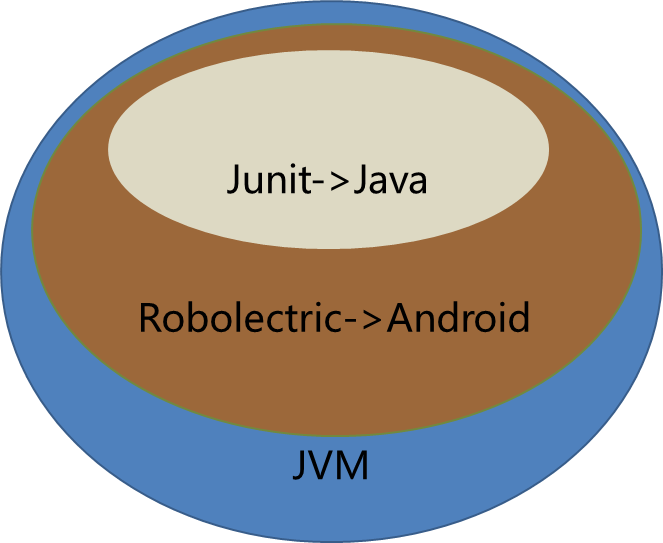


图 4.1.1 安卓测试环境

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

## 工作总结

本次毕业设计目标是实现一个基于安卓平台的经典图灵机模型仿真APP，在项目开发过程中我完成了单带图灵机模块、带子程序图灵机模块、图灵机仿真验证模块、图灵机文件管理模块四个功能模块的实现。并对该APP进行行了大量的测试，及多次回归测试。最终得到了一款功能基本完备的经典图灵机模型仿真APP。该APP支持经典单带图灵机和使用子程序技术扩展的单带图灵机的可视化构建和仿真，支持APP数据的持久化存储，并且可以针对大部分已知的错误进行处理。

本APP基于安卓平台，具备良好的页面交互功能。可视化的图灵机构建操作和仿真操作降低了使用门槛更适合初学者使用。同时本APP也适合用于教学演示。

## 工作展望

本APP功能虽然基本完备，但仍有改进的空间。

首先是图灵机仿真验证功能，与同类软件相比，本软件仅支持快速仿真而不支持单步仿真。此外仿真界面也比较简陋，输入和输出仍是采用字符串直接输出的方式，而不是如图灵机模型中带有格子的纸带形式的输入输出，并且读写头的位置也没有真正的指向正确的位置，只是采用了红色的色块对大概位置进行了指示。这些问题的存在说明本软件还有改进的空间，后续可以继续将这些功能进行实现。

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

感谢胡军老师提供了这样一个项目，使我接触到了面向移动平台的软件开发方法。在本次毕设过程中需要大量的形式语言与自动机的基础理论知识，由于本科阶段我并未选修该课程，因此在软件开发初期遇到了许多困难。在此过程中胡军老师给了我很大的帮助，帮我弥补了基础理论上的不足。对于在软件开发过程中一些无法拿定主意的关键术语的描述，老师也不厌其烦地与我进行交流。

此外感谢在本科阶段教授我课程的老师，在他们的课程中我学习到了计算机科学技术中最重要的内容，提高了我的专业技术素养。使我在毕设进行过程中的各个阶段都游刃有余，避免了在需求分析、概要设计、详细设计等开发过程中的出现一些问题。