编号 _____

南京航空航天大学

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

图灵机最初被提出是为了解决可判定性问题。在后来，该模型成为了形式语言与自动机领域最为重要的一部分，而现代计算机的物理结构设计也参考了该模型。作为计算机相关诸多领域的基础理论和方法论，形式语言与自动机理论成为大学计算机教学中不可或缺的一门课程。为了让学生可以更快速直接地理解相关概念，人们开发了许多自动机模拟仿真软件。本文则提出了一种实现在微信小程序平台的经典图灵机模拟仿真软件，可以构建和模拟三种不同构造的图灵机。

该小程序使用简约现代的界面设计方式，结合方便易懂的操作方法，让用户可以借助可视化方法在移动端设计并构建一个图灵机模型并进行模拟仿真。模拟仿真的过程将会以直观的图形绘制方式给予用户视觉反馈，从而让用户在使用该软件时不感到枯燥乏味。本文将结合示例图片对该小程序的设计与开发过程进行详细的描述，并简单介绍该小程序的测试方法。

关键词：形式语言与自动机，图灵机，多道技术，子程序技术，微信小程序

ABSTRACT

Turing Machine was originally proposed to solve the decidability problem. Later, Turing Machine became the most important model in the field of formal language and automata. The design of modern computers' physical structure also referred to this model. As the basic theory and methodology of computer science, the theory of formal language and automata has become an indispensable course in many universities. To make it more quickly for students to understand the relevant concepts, many automata simulation software has been developed. In this paper, we propose a classical Turing Machine simulation software based on WeChat mini-program. Users can build three different types of Turing Machines and test them.

We use simple and modern methods to design the interface of this mini-program. Using easy-to-understand and visualized operations, users can design and build a Turing Machine on mobile phones. Simulator will give uses visual feedback when doing the simulation, so that users will not feel boring when using it. In this paper we will describe the design and development process of the mini-program in detail. After that we will introduce the test method of the mini-program briefly.

KEY WORDS: Formal Language And Automata, Turing Machine, Multitape Turing Machine, Subroutine, Android Testing

目　　录

第一章 绪论

1.1 背景和意义

在 1928 年，David Hilbert 提出了可判定性问题（Entscheidungsproblem），为了研究该问题，Alan Turing 于 1936 年提出了图灵机模型。图灵机模型在刚刚提出时，并没有与现代计算机产生关联，但是它在理论上确定了现代计算机实现的可能性，并且为计算机的主要结构设计提供了初步的设想。冯·诺依曼结构的现代计算机就在具体设计上有着图灵机模型的影子。

20 世纪 60 年代，形式语言与自动机理论逐渐兴起。在计算机科学的诸多领域，尤其是在编程语言设计，编译原理，自然语言处理领域，形式语言与自动机理论已经成为了不可或缺的理论基础。Noam Chomsky 在其于 1956 年提出的乔姆斯基体系中提到了四种文法类型[1]，并且于 1959 年证明了这四种文法和其对应的自动机的等价性[2]。0-型文法就对应于递归可枚举语言以及自动机中的图灵机，而且没有任何的限制；1-型文法对应着上下文有关语言，该文法对应着线性有界非确定性自动机；2-型文法对应着上下文无关语言，该文法对应的是非确定性下推自动机；3-型文法也是限制最多的文法，对应的是正规语言，该文法对应着有限状态自动机。后三者对应的自动机类型均属于图灵机的子集，图灵机在形式语言与自动机理论中的重要性可见一斑。

许多大学的计算机课程中，编译原理也是一门必学的专业课，而形式语言与自动机理论作为编译原理甚至现代计算机领域的理论基础，在大学计算机系教育中也处于非常重要的地位。于是许多大学也专门为形式语言与自动机理论开设了专门的课程。然而，由于形式语言与自动机理论的高度抽象性，学生在学习过程中没有直观的理解方式，而老师在教授过程中也缺乏较好的方法将抽象的理论通过一个具体实例的视觉方式传达给学生。很多人意识到了该理论难学难懂的症结所在，于是许多带有相关展示功能的形式语言与自动机模拟仿真软件逐渐被开发出来，并且应用于课堂教学或自学中。而在智能手机流行的当下，由于手机的便携性，相对于在电脑端运行的模拟仿真软件，可以在手机上运行的模拟仿真软件逐渐受到人们的青睐。

1.2 国内外研究现状

国内目前对于自动机模型的模拟软件并没有太多的研究，在教学方面大多使用来自国外开发的自动机模拟软件。在 2003 年发表的文章中，Chesñevar 等人将自动机模型模拟软

件和自动机理论教学相关的其他工具根据支持的自动机类型数量进行了分类[3]。2011 年，Chakraborty 等人总结了形式语言与自动机模拟软件在理论教学中的重要意义并汇总了近五十年间人们开发的自动机模型模拟仿真软件[4]。该文以模拟仿真软件的设计形式和使用方式对自动机模拟仿真软件进行了划分。该分类方式将自动机模拟软件划分为了基于语言的自动机模拟软件和可视化的自动机模拟软件。基于此又按照语言设计的不同将基于语言的自动机模拟软件划分为基于符号语言的模拟器、基于汇编语言的模拟器、基于过程语言的模拟器、基于描述性语言的模拟器四个大类。而可视化的自动机模拟软件则被细分为基于结构化输入的、基于图表输入的两大类。

对于 Chakraborty 等人划分的两类自动机模拟仿真软件，基于语言的自动机模拟软件通常可以被用于模拟执行大规模的自动机模型，但是相对应的代价是必须要学该软件对应的语言，这对于教学工作来说是一笔不小的负担，并且对于学生而言重新学习一门新语言只为模拟自动机运行，积极性可能会大打折扣。而基于可视化的自动机模拟软件首先在视觉上给予了学生较好的反馈，并且对于课堂教学所使用的实例来说，学生可以更加快速地通过可视化方法来即时构建一个自动机。但是这类软件的缺点就是构建大规模自动机模型相对困难并且会耗费相对更久的时间[5]。

目前在电脑端广为使用的自动机模型模拟仿真软件是由 Rodger 等人于 20 世纪 90 年代末就已经开始研发的 JFLAP。Rodger 等人使用 Java 将其原本开发的 FLAP[6]重新开发并完善，该软件是一款电脑端的基于图表输入的可视化自动机模型模拟仿真软件，支持确定性和非确定性有穷自动机、下推自动机、图灵机、特殊的有限自动机如 Mealy 型有限自动机[7]和 Moore 型有限自动机[8]的模拟仿真。在 JFLAP 中，用户可以快速适应它的图形操作方式，使用鼠标点击或拖动自动机的相关组件来构建自己需要的自动机。Rodger 于 2006 年公布的数据中，三年之间有超过 120 个国家用户一共下载了超过 25000 次[9]。而 Rodger 在 2009 年公布的数据中，JFLAP 的下载次数已经超过了 64000 次，并且用户来自 160 多个国家[10]。目前国内已经有很多学校使用 JFLAP 来进行自动机理论和编译原理的教学[11]。

而正如前文所说，由于目前智能手机的图形可交互性以及便携性，现在的自动机教学可以逐步使用手机端的自动机模拟仿真软件。于是 Carlos H. Pereira 和 Ricardo Terra 首次推出了一款面向安卓系统的自动机模拟仿真软件 FLApp[12]。巴西大学已经将该软件应用于其形式语言与自动机课程中。而国内目前并没有在移动端的自动机模拟仿真软件上进行太多的研究。

1.3 本文主要工作

本文主要对经典图灵机模型模拟仿真微信小程序的实现与测试进行具体说明。首先本文简单介绍了图灵机理论的基本内容，然后由经典图灵机的模型引申出另外两个较为常见的图灵机编程方式：多带图灵机模型和带子程序模块的图灵机模型。接着结合国内外对于图灵机模拟仿真软件的研究与开发状况，本文提出在微信小程序平台上设计实现一款全新的经典图灵机模拟仿真软件。该小程序只使用微信小程序平台提供的原生组件，以最简洁的方式设计和实现拥有良好的图灵机模拟仿真功能的同时还具备简约精美的图形界面。本文将对该小程序的设计思路、实现方法以及核心的逻辑设计进行详细的说明。最后本文还将简单介绍在微信小程序平台缺乏自动化测试方式的情况下如何手动对该小程序的稳定性进行测试。

1.4 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。

第二章 图灵机相关内容介绍

2.1 图灵机简介

图灵机将人利用纸和笔进行计算的过程进行了抽象化[13]。图灵认为，人在计算过程中进行了如下两个操作：首先是对纸上的符号进行擦写，接着将注意力转移到纸上的另外一个位置。而对于这两种操作，图灵设计的抽象模型使用一条无限长且被分为无穷多个格子的纸带来模拟这个过程中的“纸”，即一种存储模块，而使用读写头和控制器来模拟这个过程中的“注意力”。于是这三个组件就构成了图灵机[14]。图灵机的控制器中包含有穷个状态，它将控制图灵机的读写头对纸带的内容进行读写操作，并且控制读写头的移动。

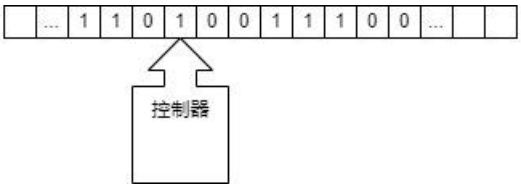


图 2.1.1 图灵机示意图

图灵机的示意图如图 2.1.1 所示。在该图中，纸带存储的内容为 11010011100 字符串，而左侧和右侧都为无限延伸的填满空白符号的格子。图灵机在运行过程中，会根据控制器的状态以及读写头所读取的符号，选择应该执行的操作，在一次操作中图灵机将会根据对应的操作对读写头指向的格子内的符号进行覆写，并且让读写头向左、向右移动或者不移动。控制器的状态会从初始状态开始，在接受状态或者拒绝状态下图灵机会停止执行。

陈有祺在其编著的《形式语言与自动机》中提出了图灵机的基本概念[15]：一个确定的单带图灵机是一个八元组 $M = (Q, \sum, \Gamma, \delta, B, (\square), s, t, r)$；其中 Q 是有穷状态集；$\sum$是有穷输入字母表；$\Gamma$是有穷的带字母表$(\Gamma \supseteq \sum)$；$\delta: Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$是状态转移函数；$B \in \Gamma - \delta$是空白符号；$s \in Q$ 是初始状态；$t \in Q$ 是接受状态；$r \in Q$ 是拒绝状态[15]。

2.2 多带技术

比较常见的图灵机编程设计技术有：在状态中存储符号、多带技术、子程序技术等[15]。其中多带技术的提出是为了对更复杂的数据进行处理，该技术将经典图灵机模型中的一条纸带水平划分为多个，并且每一条道都分别存有一个符号，这就相当于将经典图灵机模型中一个格子存储一个符号的形式改变成为一个格子存储一个符号向量的形式。图 2.1.2 展

示了一个多带图灵机，可以看到，控制器和读写头仍然与单带图灵机一致，但是多带图灵机可以一次性处理四条纸带上读写头指向的格子位置的字符，也就是一个字符向量。

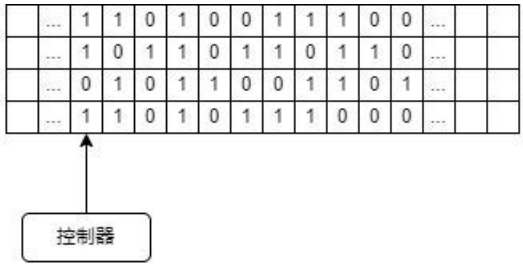


图 2.1.2 多带图灵机示意图

2.3 子程序技术

在平常的软件开发过程中，模块化是提高代码复用度，将困难问题化简的绝佳方法。而这个方法同样也适用于图灵机，对于一个大规模的图灵机而言，如果其中存在大量的对同一种运算的调用，那么将这个运算封装为一个子图灵机，再将其嵌入到该图灵机中，会极大降低构建这样的图灵机的工作难度，在设计和维护上都非常方便。

子程序本身可以作为经典图灵机独立运行，也可以加入到这样一个利用了子程序技术的图灵机中，当作为模块嵌入到图灵机中时，子程序的初始状态和结束状态会与父程序相连，而父程序除此以外无法影响子程序内部的其他状态。当父程序运行到该子程序模块时，会将自己的纸带与读写头位置传递给该子程序，由子程序接管这两个组件继续运行，子程序在运行结束后，会将这两个组件与父程序同步。

2.4 图灵机与计算机

《自动机理论，语言和计算导论》中，作者提到的图灵机和计算机的关系是这样的：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[16]。这个结论意味着实际上是可以用计算机来模拟图灵机模型的，手机作为现代计算机的一员，同样可以用来对图灵机进行模拟仿真。

绪论中已经提到，冯·诺依曼体系结构的计算机在物理结构层面上有着图灵机结构的样子，因此甚至可以认为该结构的计算机即为一种图灵机的物理实现，所以人们通常认为基于该结构的现代计算机与图灵机的计算能力在理论上是等价的[17]。为什么说该结构的计算机与图灵机是在理论上有相同的计算能力，因为在实际情况下，现代计算机的存储结构中硬盘和内存的空间大小必然是有限的，且由于物理因素，现代计算机实际上是无法永

久运行下去的，然而图灵机模型作为理论模型，其拥有无限长的存储结构，并且对于永不停机的图灵机模型而言，它们可以一直运行下去，这在物理意义上是违反热力学定律的。这就意味着现代计算机是无法模拟所有的图灵机模型的。但是对于模拟软件来说，当软件在“无限”这样一个理论条件上加以限制，将现实意义上可实现的图灵机归纳为一个集合，计算机是可以模拟该集合中任意一个图灵机的[18]。

综上所述，实际实现的图灵机模拟仿真软件，可以对所有任意图灵机组成的集合的一个子集中的所有图灵机进行模拟。这样的图灵机模拟仿真软件在发挥最大能力的情况下，也满足了教学和大部分的研究需求。

2.5 本章小结

本章简单介绍了图灵机的理论模型以及其形式化定义。然后介绍了两种较为常见的图灵机编程方式：多带图灵机技术和带子程序图灵机技术。最后简单讲述了图灵机与现代计算机的关系，由此可以知道现代计算机可以用来进行图灵机的模拟。

第三章 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。

3.1 需求分析

3.1.1 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；

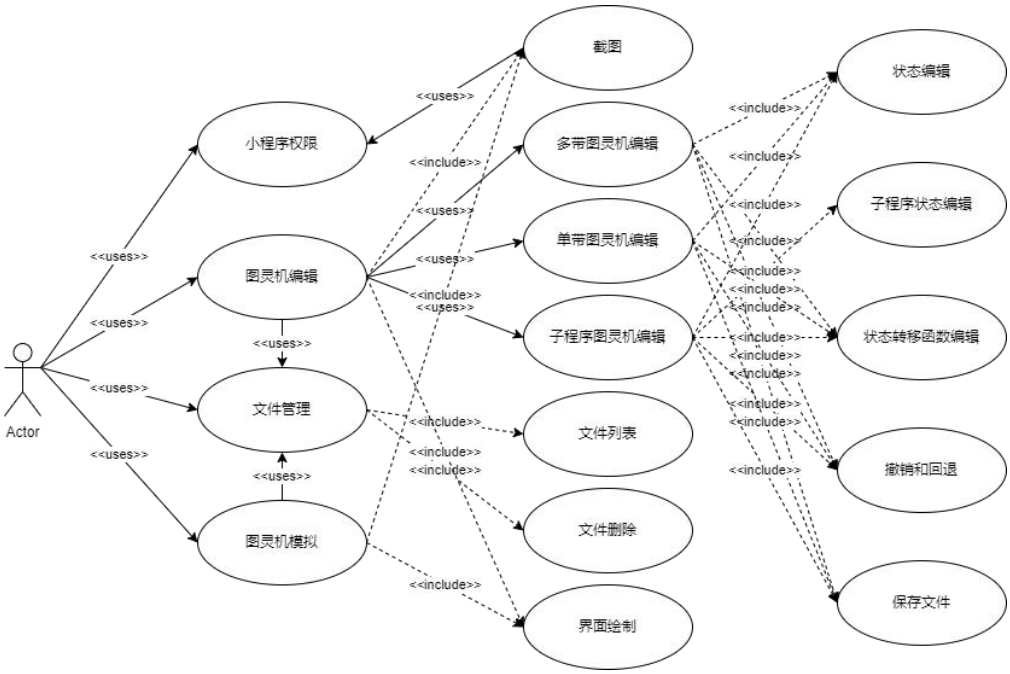


图 3.1.1 经典图灵机仿真微信小程序用例图

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。

3.1.2 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。

用例编号	NORMAL	用例名称	单带图灵机编辑
用例描述	单带图灵机编辑的用例用于创建新的单带图灵机模型或修改已保存的单带图灵机模型。该用例包括了图灵机状态的创建、移动、删除、初始状态和接受状态的设置与取消功能，状态转移函数的创建、修改、删除功能。除了编辑的主要功能外，还包括编辑操作的撤回和恢复功能，保存到文件功能，保存截图功能和即时模拟仿真功能。		
触发事件	1.用户在主页选择创建单带图灵机； 2.用户从文件列表打开单带图灵机文件。		
前提条件	无		
后续条件	用户编辑好单带图灵机后，选择保存图灵机文件或放弃当前编辑的图灵机。		
主事件流	1.用户点击主页的创建单带图灵机按钮； 2.用户点击状态按钮，系统进入状态编辑模式； 3.用户在绘制区点击，绘制区对应坐标出现一个状态，重复若干次； 4.用户点击状态转移函数按钮，系统进入状态转移函数编辑模式； 5.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； 6.用户长按绘制区的状态，设置初始状态与最终状态； 7.用户点击删除按钮，系统进入删除模式； 8.用户点击绘制区的状态或状态转移函数，删除该组件； 9.用户点击撤回按钮返回上一步操作； 10.用户点击恢复按钮恢复刚刚撤回的一步操作；		
其他事件流	文件保存： 1.用户点击保存文件按钮； 2.系统判断是否为新建的图灵机模型，如果是新建模型，则用户输入文件名进行保存操作，否则直接保存到打开的文件中。 保存截图： 1.用户点击保存截图按钮； 2.系统判断是否有保存截图的权限，有则直接从绘制区生成 PNG 文件，否则跳转到小程序权限管理模块。 即时模拟仿真功能： 1.用户点击模拟仿真按钮； 2.跳转到图灵机模拟仿真模块。		
备注	无		

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。

用例编号	MULTIPLE	用例名称	多带图灵机编辑
用例描述	多带图灵机编辑的用例用于创建新的多带图灵机模型或修改已保存的多带图灵机模型。该用例包括了图灵机状态的创建、移动、删除、初始状态和接受状态的设置与取消功能，状态转移函数的创建、修改、删除功能。除了编辑的主要功能外，还包括编辑操作的撤回和恢复功能，保存到文件功能，保存截图功能和即时模拟仿真功能。		
触发事件	1.用户在主页选择创建多带图灵机； 2.用户从文件列表打开多带图灵机文件。		
前提条件	用户选择创建时必须输入符合条件的纸带数。		
后续条件	用户编辑好多带图灵机后，选择保存图灵机文件或放弃当前编辑的图灵机。		
主事件流	1.用户点击主页的创建多带图灵机按钮； 2.用户点击状态按钮，系统进入状态编辑模式； 3.用户在绘制区点击，绘制区对应坐标出现一个状态，重复若干次； 4.用户点击状态转移函数按钮，系统进入状态转移函数编辑模式； 5.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； 6.用户长按绘制区的状态，设置初始状态与最终状态； 7.用户点击删除按钮，系统进入删除模式； 8.用户点击绘制区的状态或状态转移函数，删除该组件； 9.用户点击撤回按钮返回上一步操作； 10.用户点击恢复按钮恢复刚刚撤回的一步操作；		
其他事件流	文件保存： 1.用户点击保存文件按钮； 2.系统判断是否为新建的图灵机模型，如果是新建模型，则用户输入文件名进行保存操作，否则直接保存到打开的文件中。 保存截图： 1.用户点击保存截图按钮； 2.系统判断是否有保存截图的权限，有则直接从绘制区生成PNG文件，否则跳转到小程序权限管理模块。 即时模拟仿真功能： 1.用户点击模拟仿真按钮； 2.跳转到图灵机模拟仿真模块。		
备注	多带图灵机编辑模块在单带图灵机编辑模块的基础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 3.1.3 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。

用例编号	SUBPROGRAM	用例名称	子程序图灵机编辑
用例描述	子程序图灵机编辑的用例用于创建新的子程序图灵机模型或修改已保存的子程序图灵机模型。该用例包括了单带图灵机模块的基础编辑功能和子程序模块的导入、移动、删除、初始状态和接受状态的设置与取消功能。除了编辑的主要功能外，还包括编辑操作的撤回和恢复功能，保存到文件功能，保存截图功能和即时模拟仿真功能。		
触发事件	1.用户在主页选择创建子程序图灵机； 2.用户从文件列表打开子程序图灵机文件。		
前提条件	用户导入的子程序模块必须为单带图灵机。		
后续条件	用户编辑好子程序图灵机后，选择保存图灵机文件或放弃当前编辑的图灵机。		
主事件流	1.用户点击主页的创建子程序图灵机按钮； 2.用户点击状态按钮，系统进入状态编辑模式； 3.用户在绘制区点击，绘制区对应坐标出现一个状态，重复若干次； 4.用户点击子程序模块按钮，系统进入导入子程序模块模式； 5.用户在绘制区点击，绘制区对应坐标出现一个子程序模块，重复若干次； 6.用户点击状态转移函数按钮，系统进入状态转移函数编辑模式； 7.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； 8.用户长按绘制区的状态或子程序模块，设置初始状态与最终状态； 9.用户点击删除按钮，系统进入删除模式； 10.用户点击绘制区的状态、子程序模块或状态转移函数，删除该组件； 11.用户点击撤回按钮返回上一步操作； 12.用户点击恢复按钮恢复刚刚撤回的一步操作；		
其他事件流	文件保存： 1.用户点击保存文件按钮； 2.系统判断是否为新建的图灵机模型，如果是新建模型，则用户输入文件名进行保存操作，否则直接保存到打开的文件中。 保存截图： 1.用户点击保存截图按钮； 2.系统判断是否有保存截图的权限，有则直接从生成 PNG 文件，否则跳转到权限管理模块。 即时模拟仿真功能： 1.用户点击模拟仿真按钮； 2.跳转到图灵机模拟仿真模块。		
备注	子程序图灵机编辑模块在单带图灵机编辑模块的基础上添加了导入子程序模块的功能，并且将子程序看做一种特别的状态，拥有和状态相同的编辑逻辑。		

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）。

用例编号	SIMULATOR	用例名称	图灵机模拟
用例描述	图灵机模拟用例用于加载选择的图灵机模型文件或者从编辑界面中获得的图灵机模型，与输入待验证字符串相结合进行模拟仿真。该用例包括输入待验证字符串功能、单步执行功能、快速执行功能。除了主要仿真功能之外，还包括多结果切换查看的功能和保存截图功能。		
触发事件	1.用户在编辑界面点击即时模拟按钮； 2.用户文件列表点击文件进行模拟。		
前提条件	无		
后续条件	无		
主事件流	1.用户点击图灵机模拟； 2.用户输入待验证字符串； 3.模拟器对图灵机模型的正确性进行检验； 4.用户点击单步执行按钮，图灵机执行一次操作； 5.用户点击快速执行按钮，图灵机直接运行到结束。		
其他事件流	多结果切换： 1.用户点击查看上一条结果； 2.模拟器纸带绘制区显示上一条的图灵机结果快照； 3.用户点击查看下一条结果； 4.模拟器纸带绘制区显示下一条的图灵机结果快照。 保存截图： 1.用户点击保存截图按钮； 2.系统判断是否有保存截图的权限，有则直接从绘制区生成PNG文件，否则跳转到小程序权限管理模块。		
备注	图灵机模拟的绘制区除了纸带显示图灵机执行快照外，绘制逻辑和编辑模块的绘制逻辑是同一个逻辑。		

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：

$$\begin{aligned} q_0 &\xrightarrow{1;1;R} q_1 \\ q_0 &\xrightarrow{1;0;R} q_2 \end{aligned} \tag{3.1.1}$$

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。

3.1.3 小结

本节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。

3.2 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。

3.2.1 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。

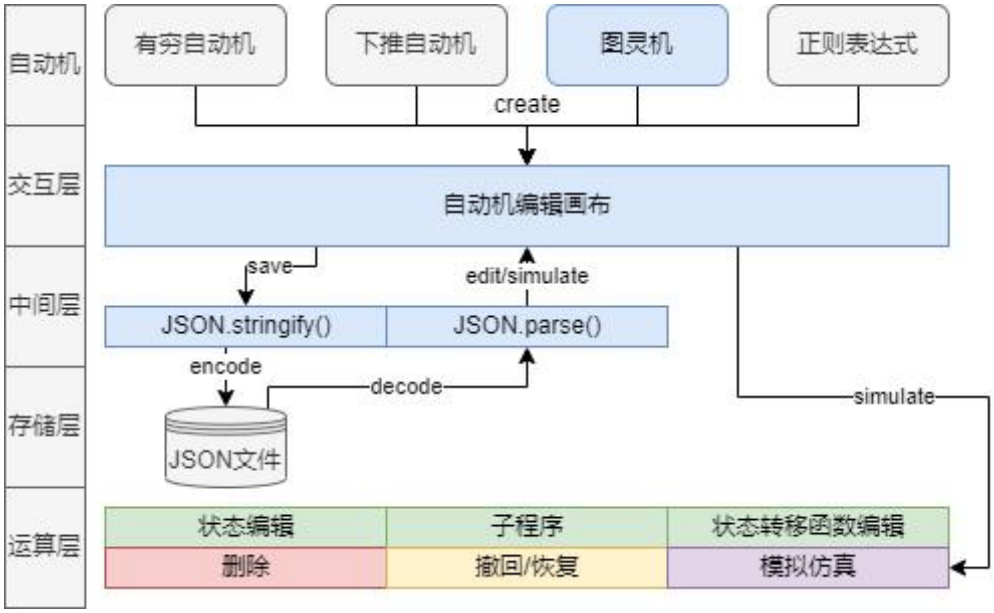


图 3.2.1 形式语言与自动机仿真微信小程序项目架构图

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。

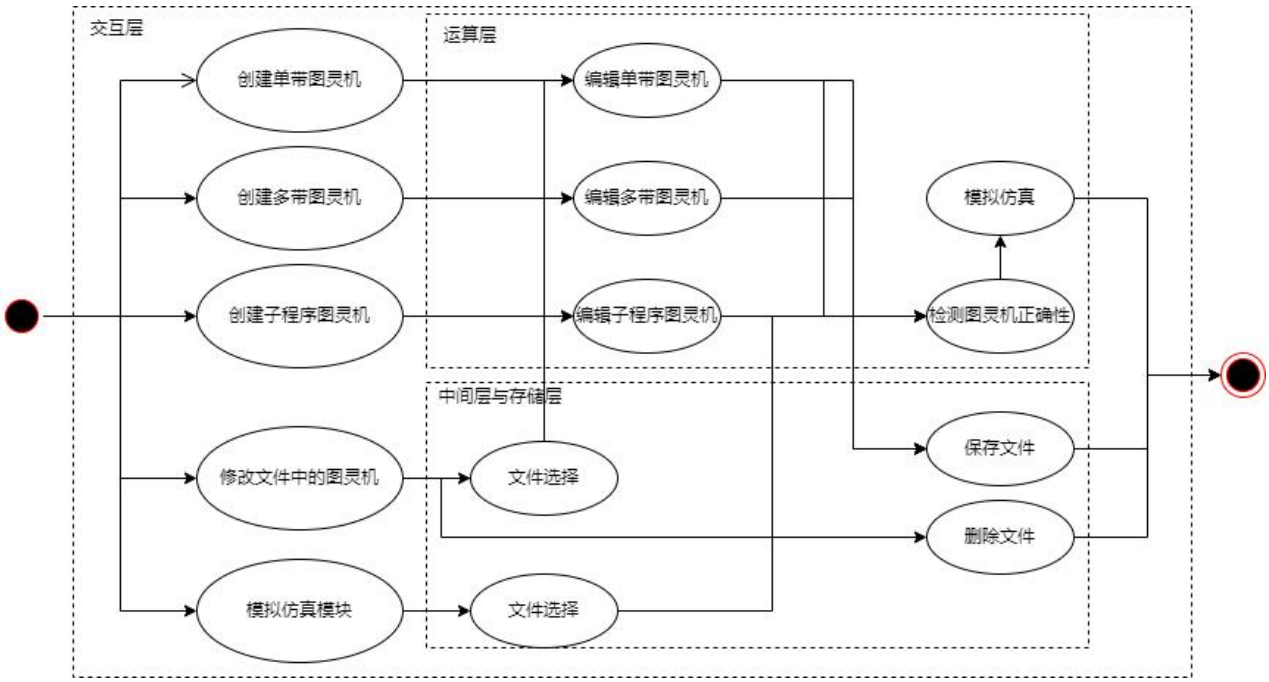


图 3.2.2 经典图灵机模拟仿真微信小程序模块功能活动图

3.2.2 单带图灵机编辑模块设计

由 3.1 中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。故本节中将对单带图灵机编辑模块进行细致的模块设计。

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。

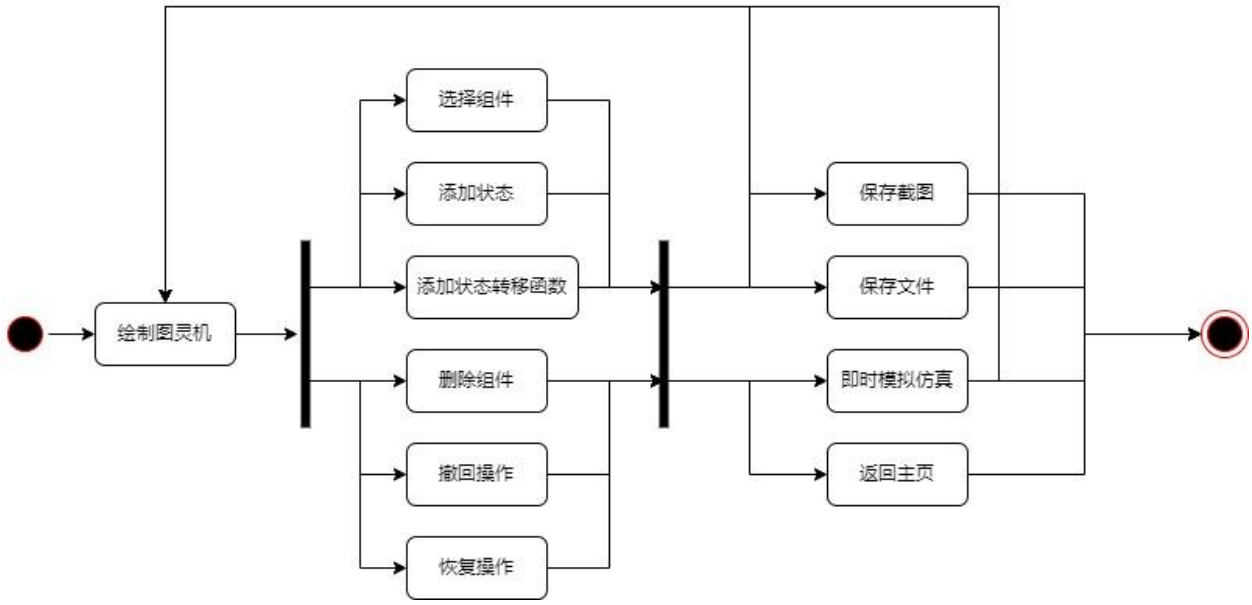


图 3.2.3 单带图灵机编辑模块活动图

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。

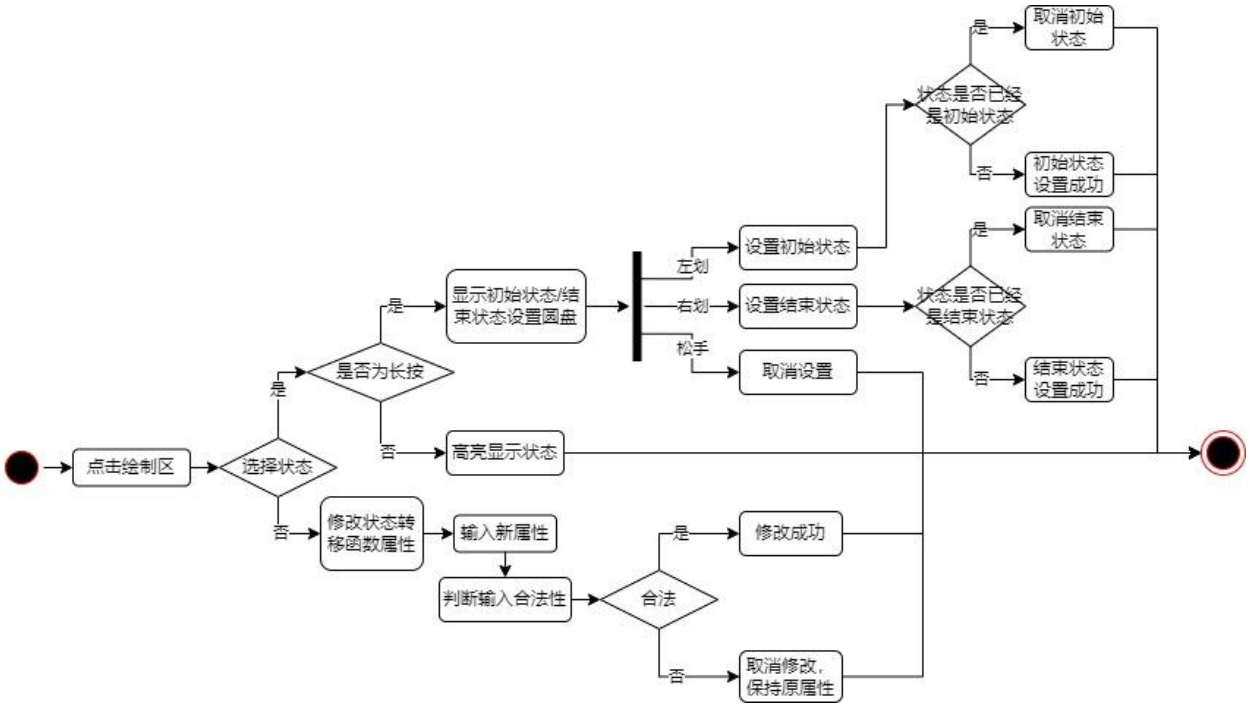


图 3.2.4 选择组件功能活动图

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。这个按钮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。

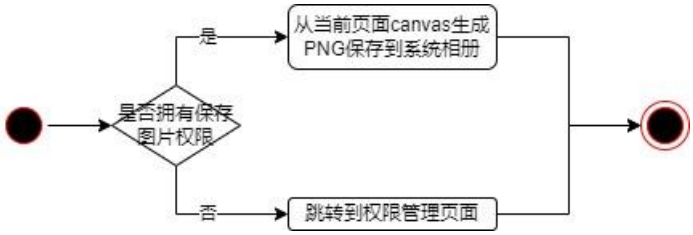


图 3.2.5 保存截图功能活动图

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。

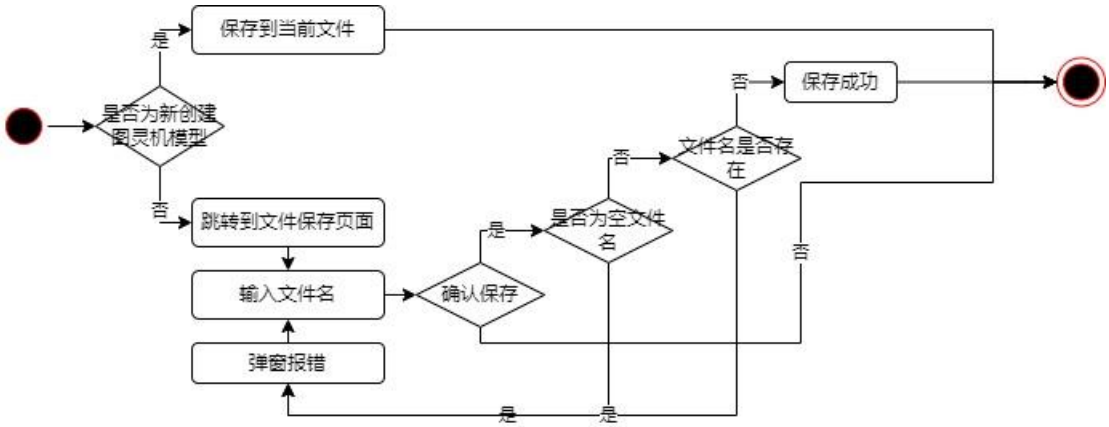


图 3.2.6 保存文件功能活动图

3.2.3 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。

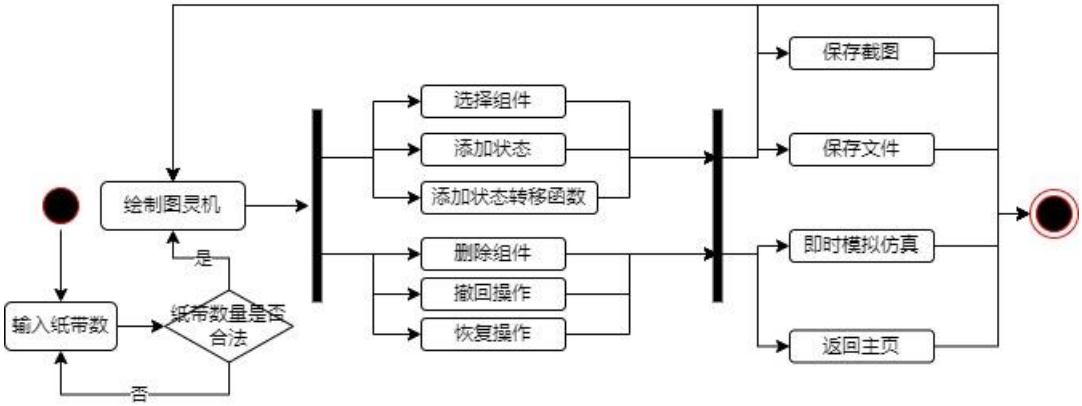


图 3.2.7 多带图灵机编辑模块活动图

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。因此本项目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。

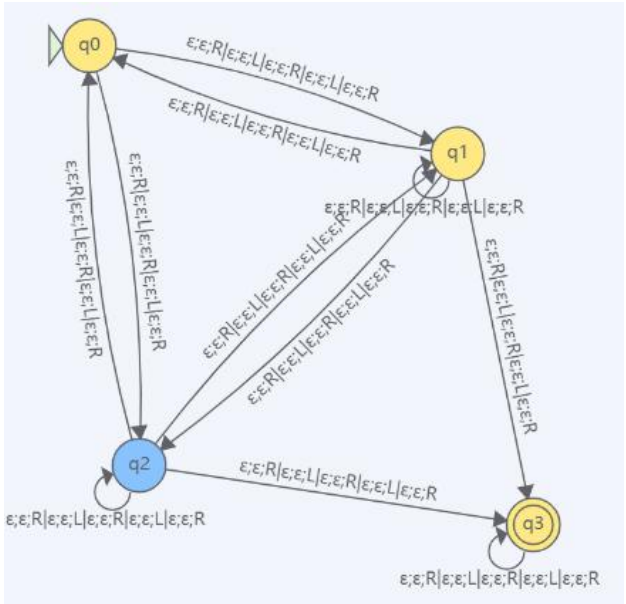


图 3.2.8 多带图灵机 5 纸带数实例

3.2.4 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，由图中高亮显示的部分可知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，图 3.2.10 展示了该功能的具体活动图。

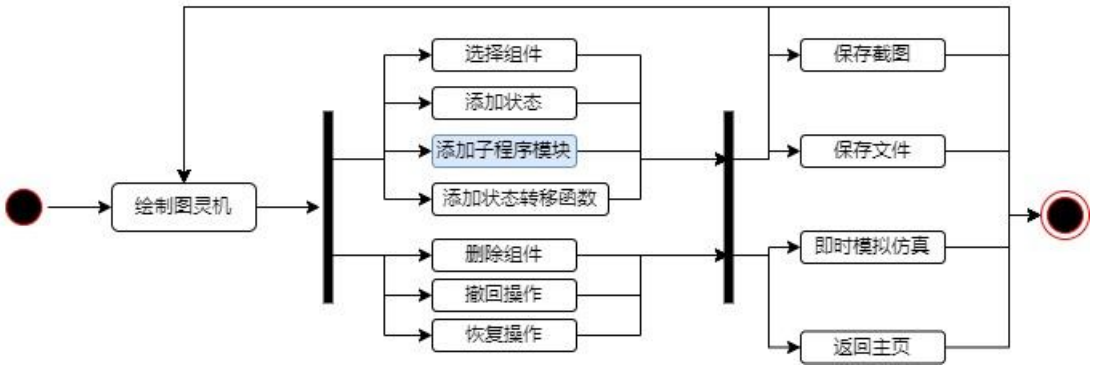


图 3.2.9 带子程序图灵机编辑模块活动图

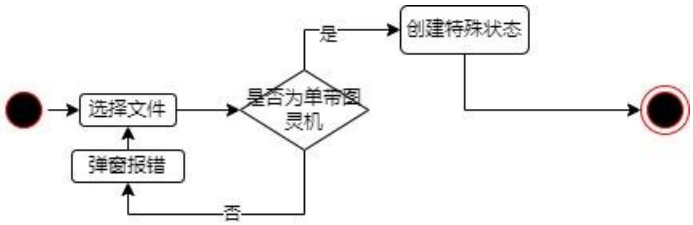


图 3.2.10 添加子程序模块功能活动图

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。

3.2.5 图灵机模拟仿真模块设计

该小程序的整体活动图（图 3.2.2）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。

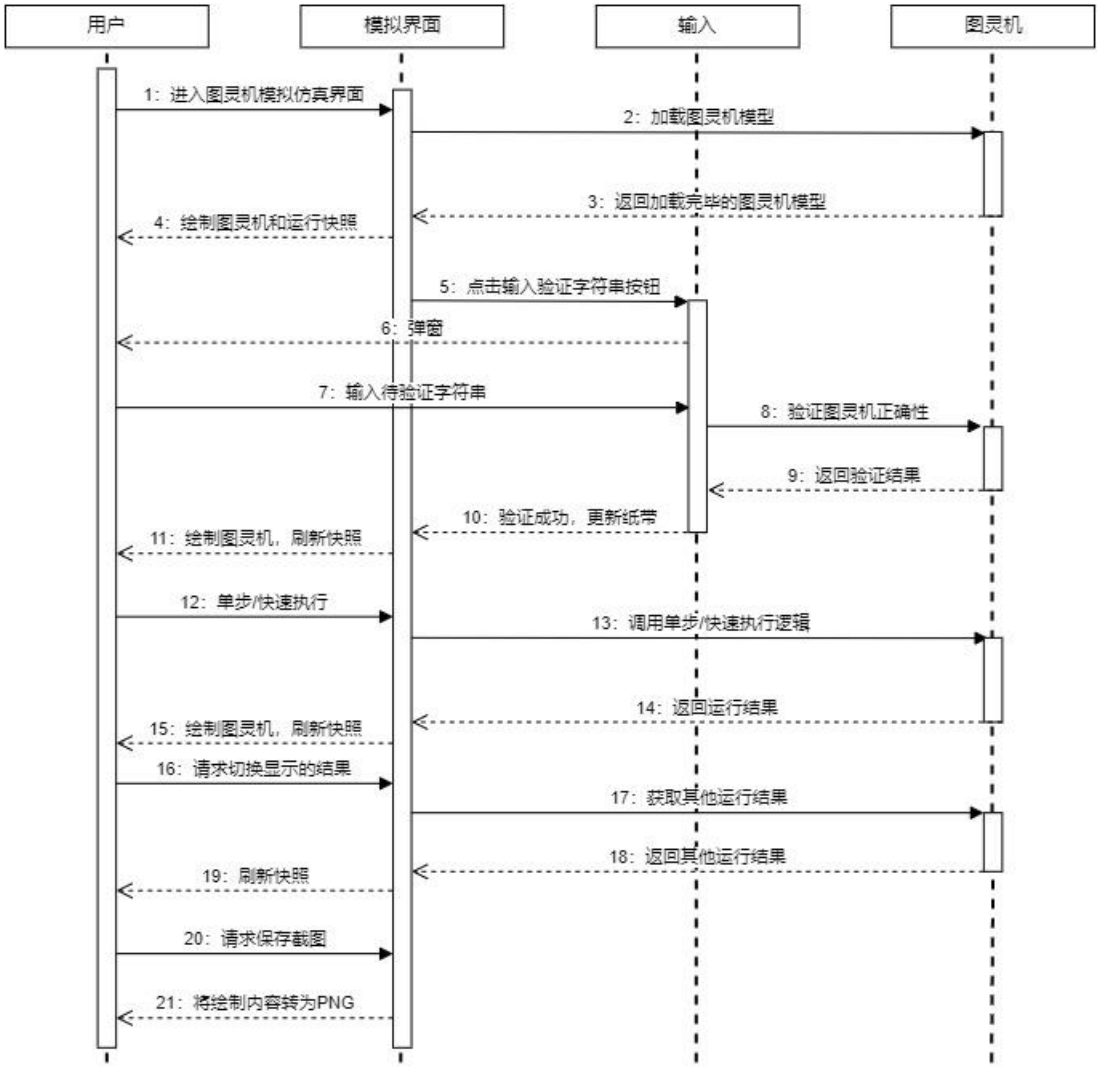


图 3.2.11 图灵机模拟仿真时序图

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。

3.2.6 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。

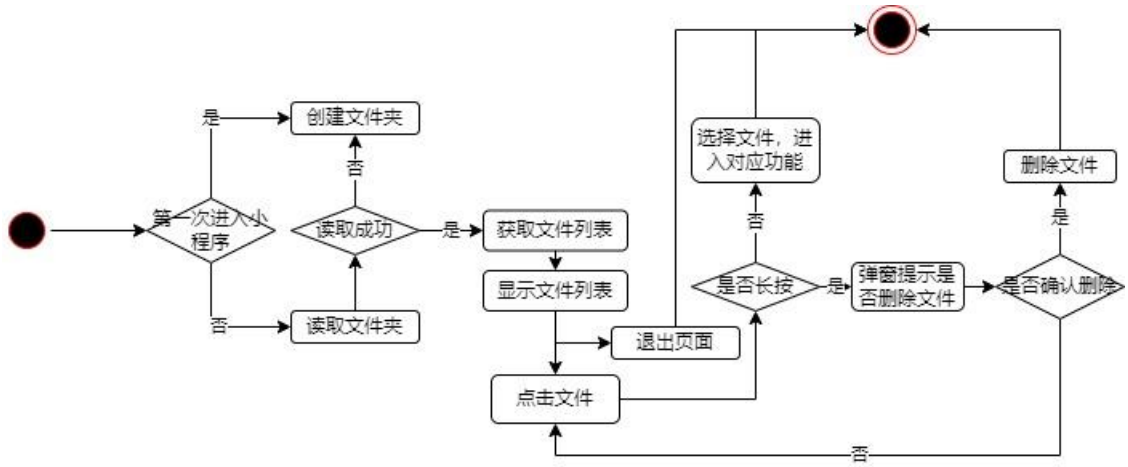


图 3.2.12 图灵机文件管理模块活动图

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，小程序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，该小程序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。

3.2.7 小结

在 3.2 节中本文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，在接下来的一节中本文将对这些内容进行详细的设计与实现。

3.3 详细设计与实现

本节将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。

3.3.1 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。

3.3.2 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，本项目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

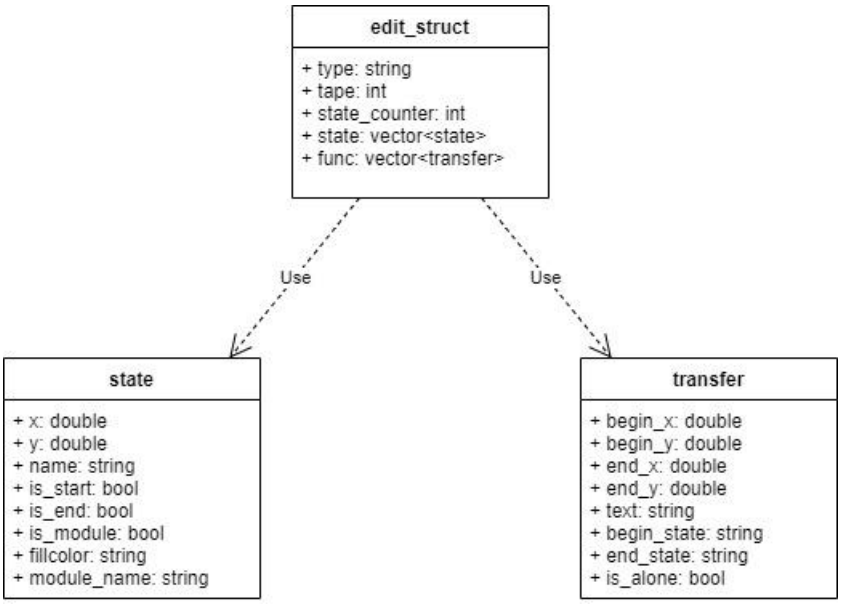


图 3.3.1 图灵机编辑类图

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，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。

(2) 撤销/恢复操作

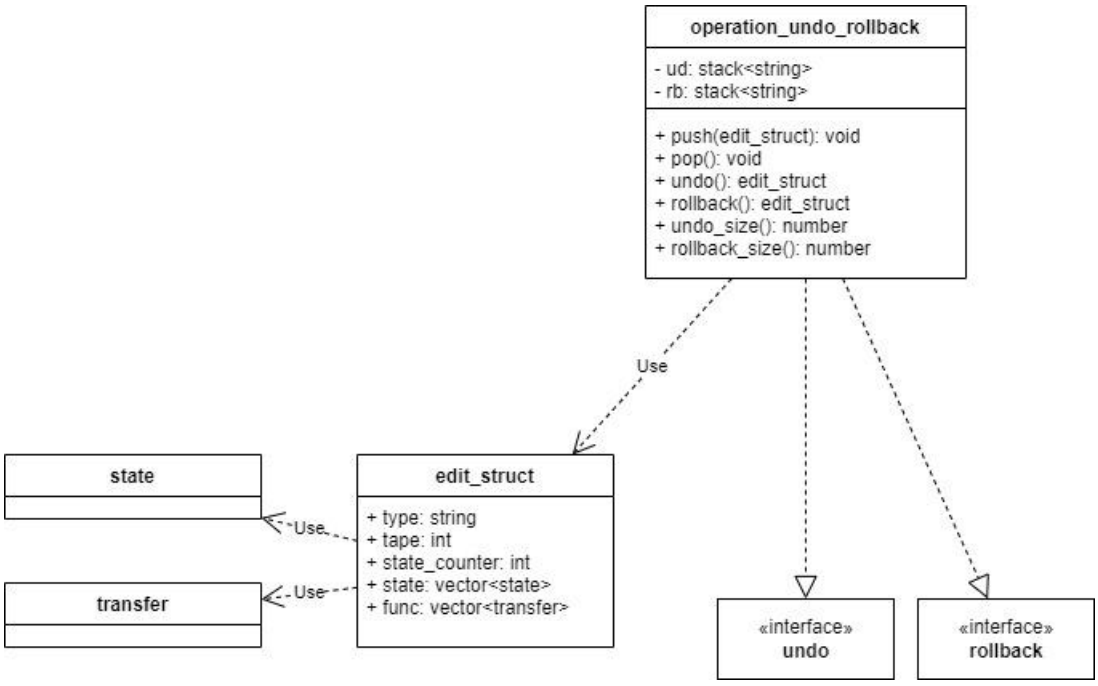


图 3.3.2 撤销/恢复操作类图

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。基于该类实现的 undo 和 rollback 接口用户可以直接使用。

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。

(3) 单带图灵机模拟

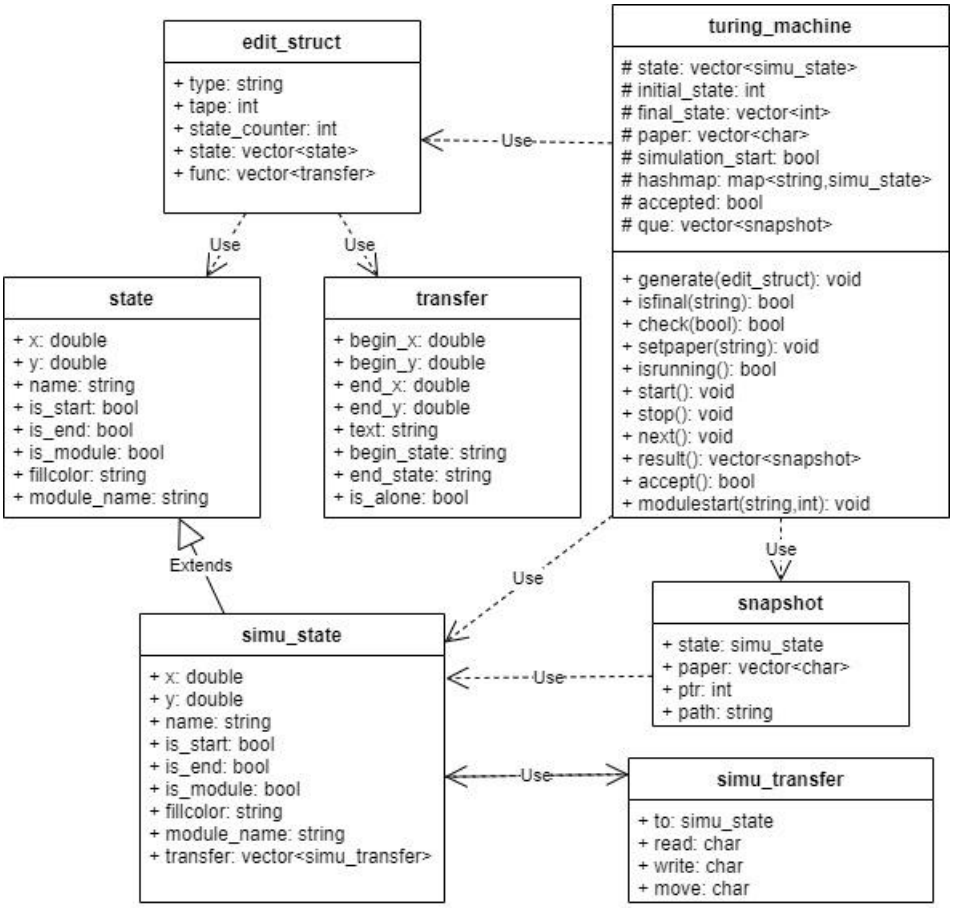


图 3.3.3 单带图灵机模拟类图

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。

(4) 多带图灵机模拟

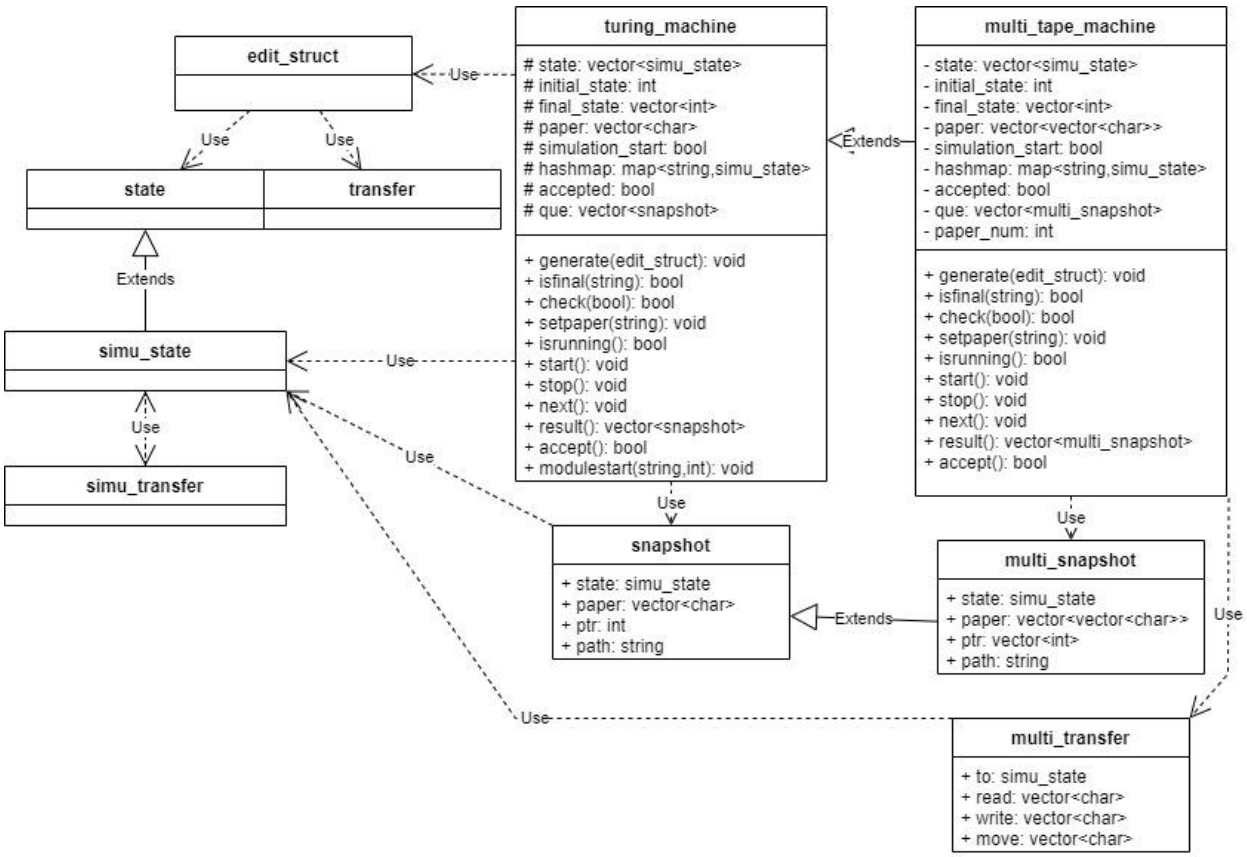


图 3.3.4 多带图灵机模拟类图

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。

(5) 子程序图灵机模拟

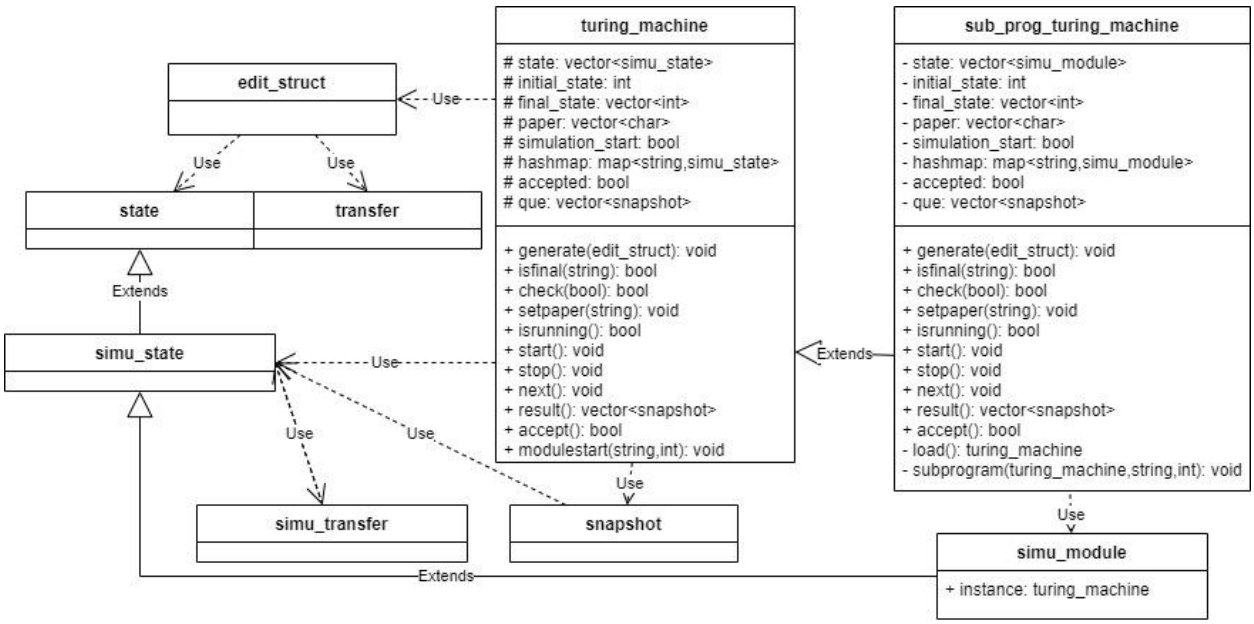


图 3.3.5 子程序图灵机模拟类图

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。

3.3.3 界面绘制逻辑设计与实现

在该节中，本文对界面的设计与具体实现情况进行了展示。在该部分中，本文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，当然对于目前的设计思路来说，本项目的设计方案仅限于让使用者不会觉得使用该小程序太过于枯燥乏味。因此本项目在小程序的 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。

小程序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"
}
```

app.wxss 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

在全局的按钮设置中，小程序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;
}
```

图 3.3.8 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：

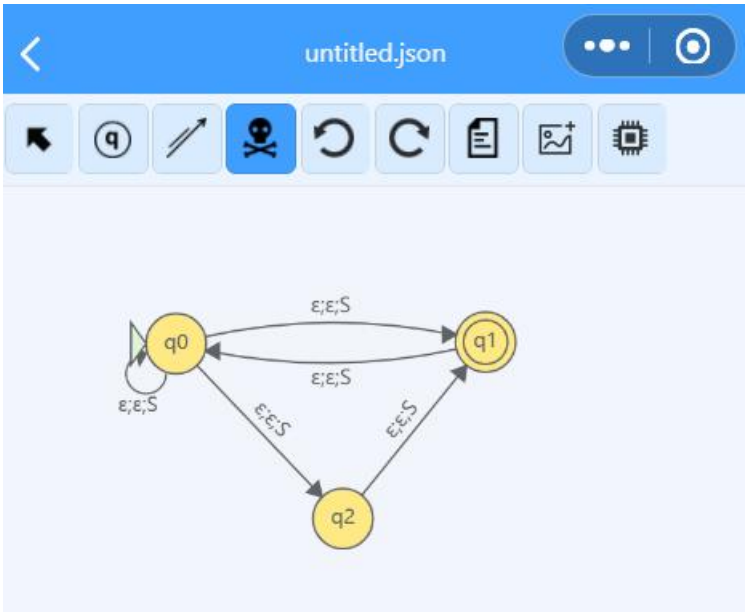


图 3.3.8 单带图灵机编辑界面（部分）

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，小程序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

(2) 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，本项目采用 canvas 组件，并且使用 canvas2d 模式来进行绘制，在实现中，小程序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。

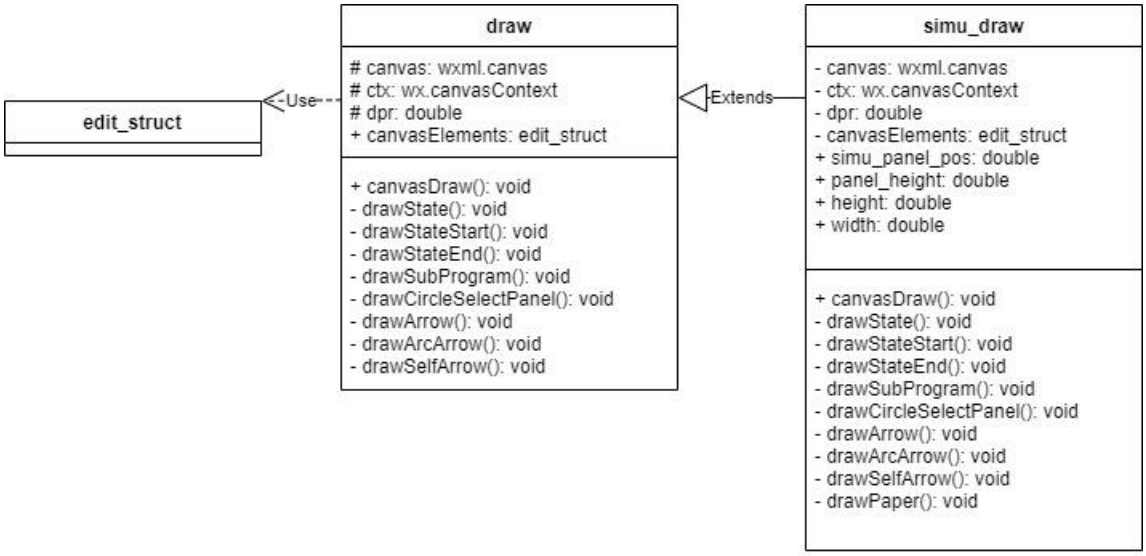


图 3.3.10 绘制逻辑类图

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。所以小程序为其添加了 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。

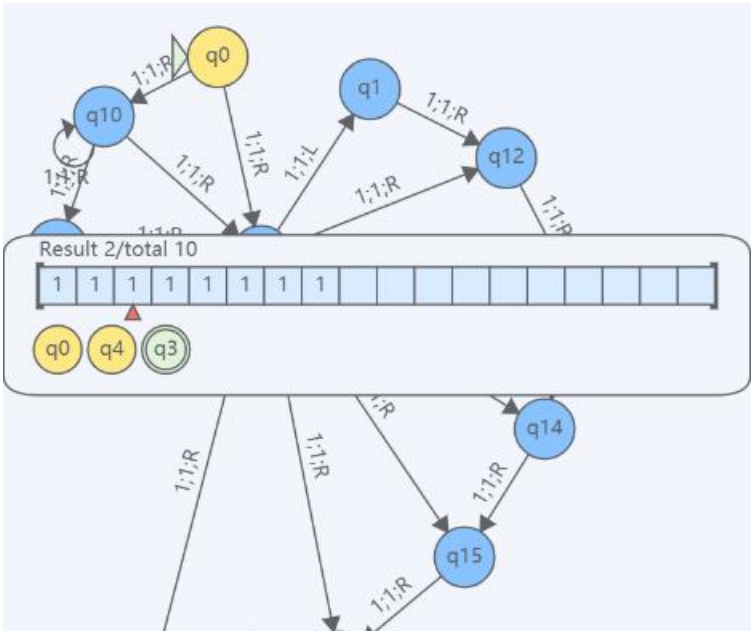


图 3.3.11 运行快照的显示遮蔽了图灵机主体绘制部分

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。

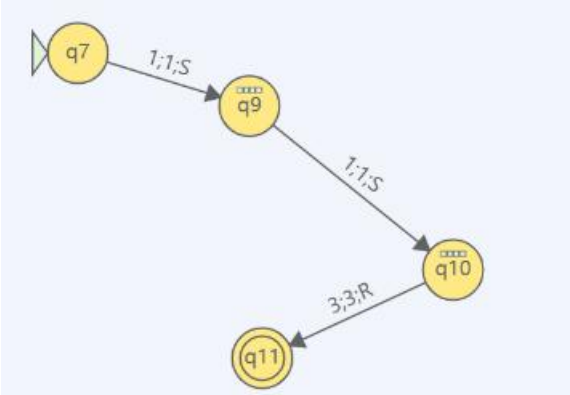
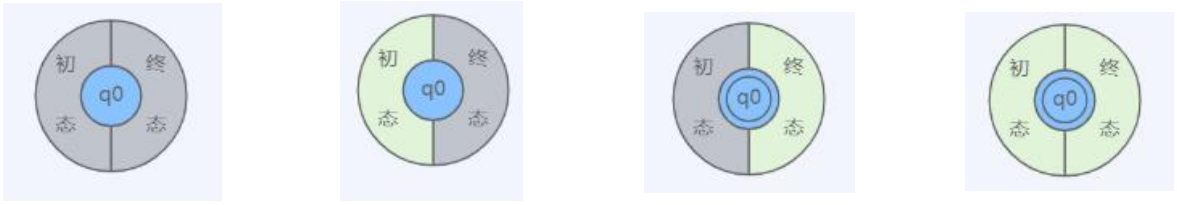


图 3.3.12 特别状态的绘制效果

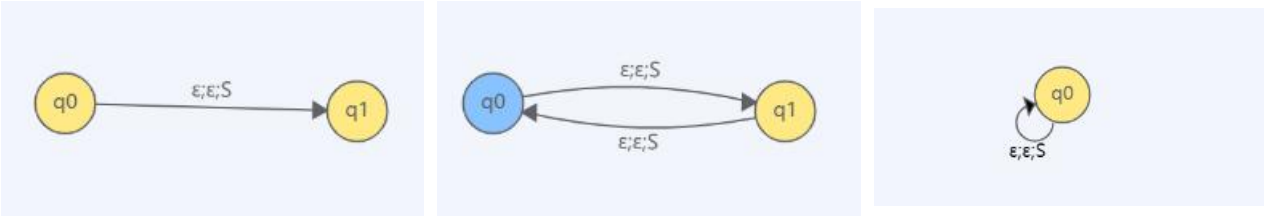
drawCircleSelectPanel 仅在编辑界面中会被触发，该绘制效果是在用户在选择模式下长按状态时显示的，在图 3.2.4 中已经提到这个功能。该功能会对长按的状态绘制一个特殊的圆盘，具体绘制效果如图 3.3.13。用户在按住的情况下，左划或者右划接着松开，就会触发对应的初态/终态的设置和取消事件。这种圆盘的设计相对于弹窗让用户填写确认来说，一方面更方便用户进行操作，不会打断用户的操作过程，另外一方面，这种设计还会让用户产生一种科技感，增加使用的趣味性。



（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。

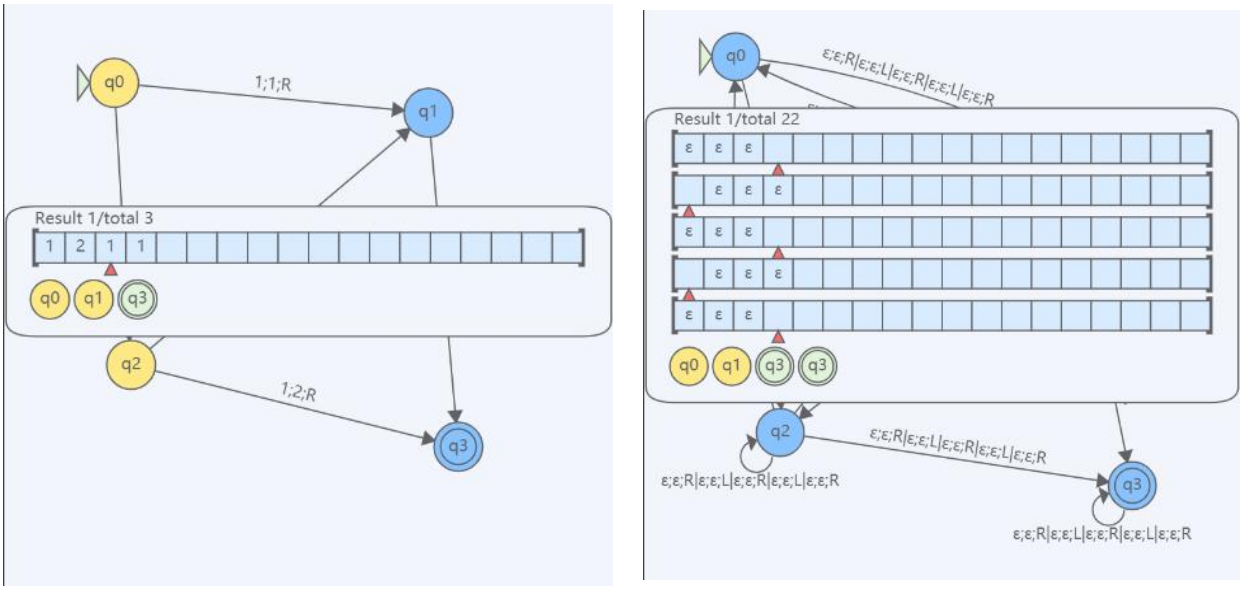


（a）直线箭头　　（b）弧线箭头　　(c) 指向自己的箭头

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，图 3.3.15 展示了 drawPaper 方法绘制出的所有情况下的图灵机运行快照。该快照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用户查看的结果索引；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

3.3.4 核心算法设计与实现

本节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

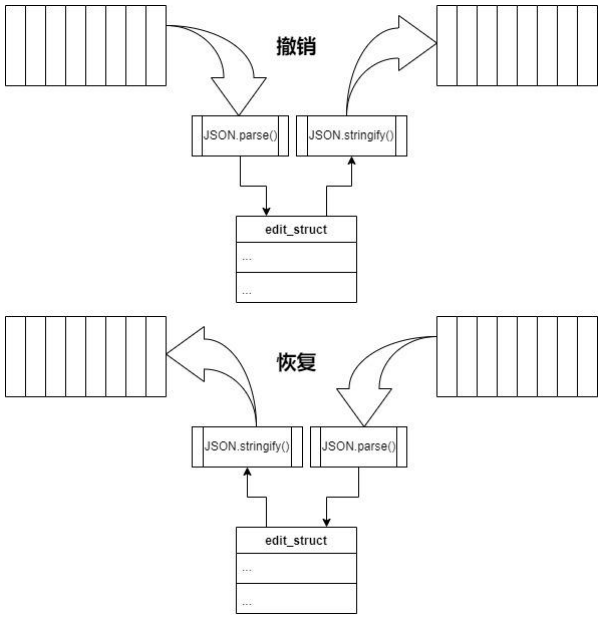


图 3.3.16 撤销与恢复示意图

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，编辑模块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

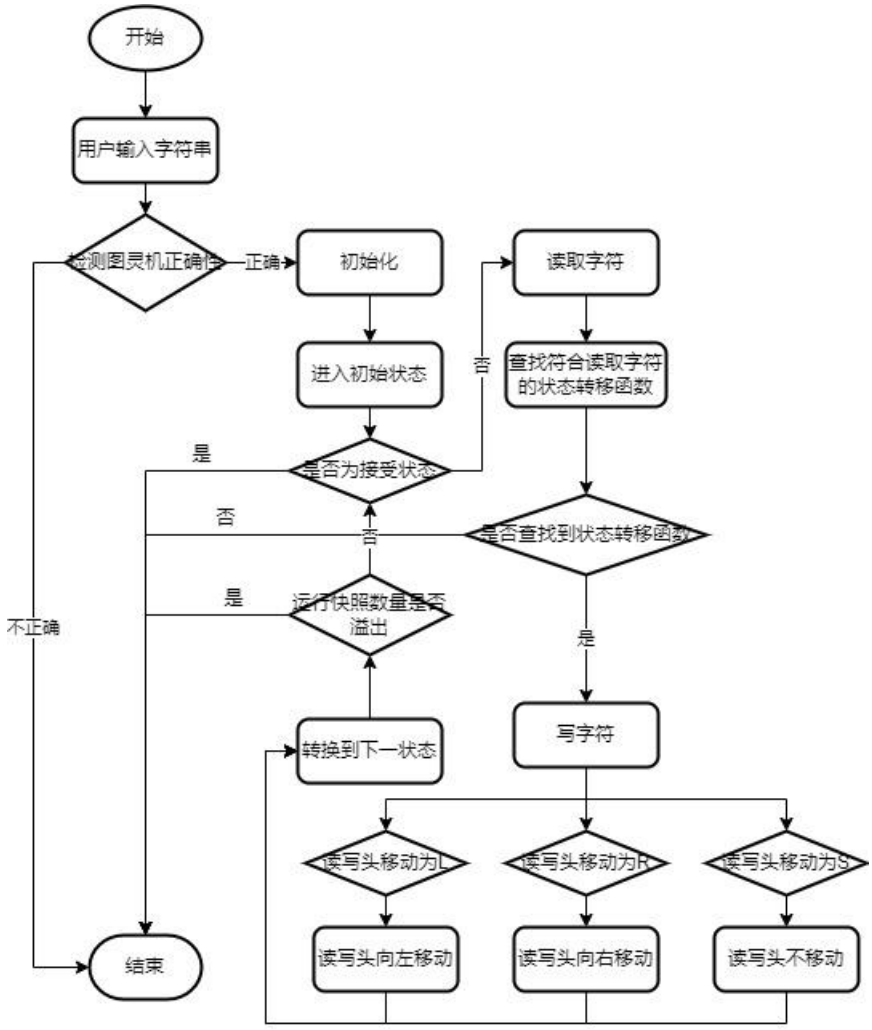


图 3.3.17 单带与多带图灵机模拟流程图

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，而由目前的理论可知无法通过一般方法判断任意图灵机是否可停机[19]，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，这个设计最终没有在本项目中采纳。

```
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。在该方法的最后可以看到一个对长度的判定，小程序对此设置的阈值为 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，小程序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空白符号。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空白符号，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

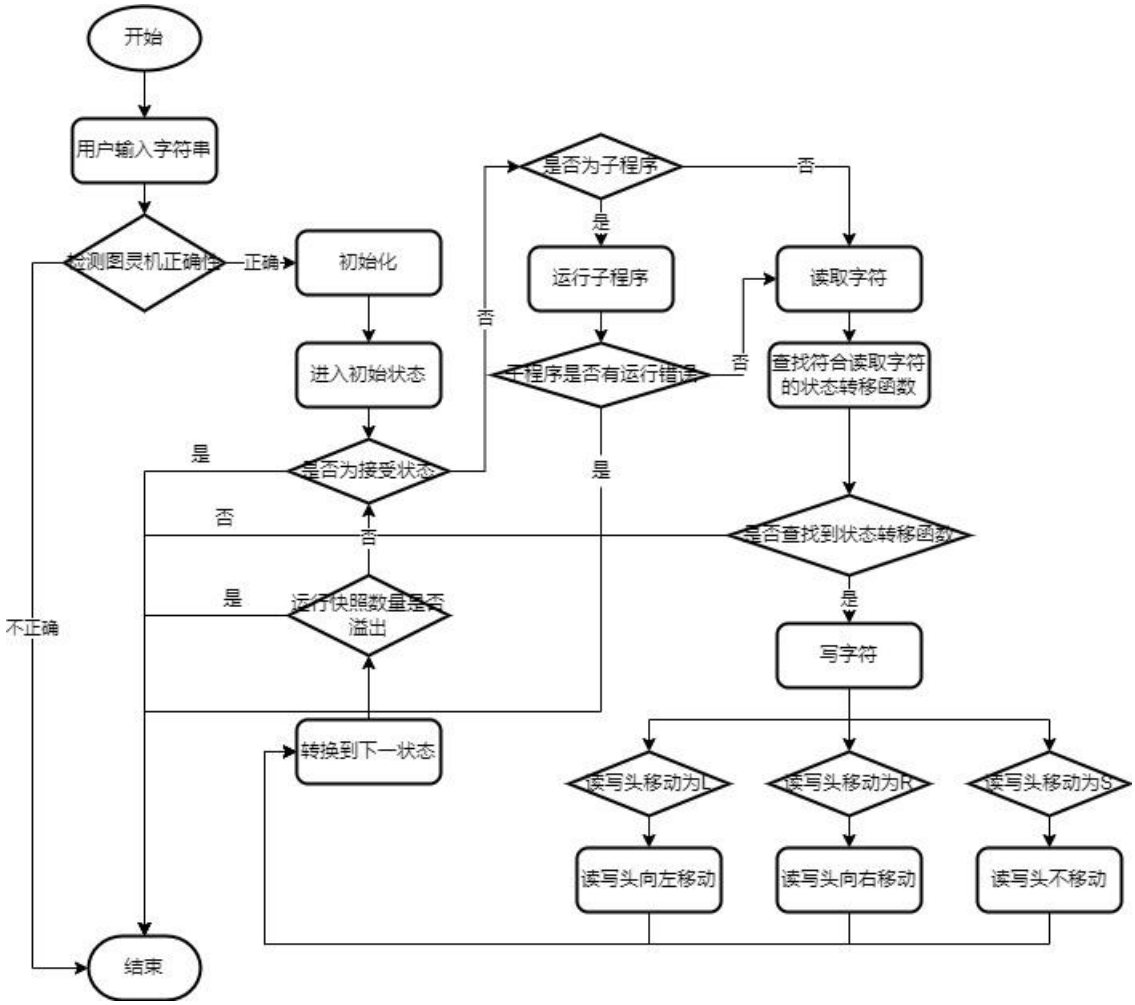


图 3.3.18 子程序图灵机模拟流程图

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。

3.4 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。

第四章 经典图灵机仿真微信小程序测试方案

软件的测试是验证软件功能稳定性和可靠性的必经之路[20]，对于自动机模拟仿真软件的模拟模块的测试，本项目需要针对性地构造测试用例[21]，所以本章将简单介绍经典图灵机仿真微信小程序的测试方法和测试用例。

4.1 测试范围和测试内容

由于本次毕业设计使用的微信小程序平台使用类似于 Web 软件的文件结构来实现小程序，而这个平台对于自动测试没有比较好的工具，仅仅提供了比较丰富的调试信息和界面模拟器用于给用户自行调试。所以在本次开发中，开发者的测试方法以手动为主，在开发过程中对每个小模块、每个方法的所有分支都进行手动遍历，以保证开发的模块本身具有高度的稳定性，接着对大模块进行手动测试，保证每个小模块和方法在大模块中的表现符合预期，最后再整合为整体的小程序，进行系统性的测试。

4.1.1 界面绘制逻辑

界面绘制逻辑的测试主要通过查看绘制效果进行。3.3.3 节中已经对界面绘制逻辑的所有重要部分提供了图例，在此仅列出需要测试的界面绘制方法。该测试是在开发过程中同步进行的，目的是为保证后续的模块开发不会被影响。

表 4.1.1 界面绘制逻辑测试内容

绘制方法	属于编辑界面	属于模拟界面	测试重点
drawState	是	是	状态名称位置是否正常，以及高亮显示的颜色是否正常
drawStateStart	是	是	绘制情况
drawStateEnd	是	是	小圆环的绘制是否会覆盖状态名称
drawSubProgram	是	是	绘制情况
drawCircleSelectPanel	是	否	选择圆盘的颜色是否与状态的属性相符合
drawArrow	是	是	箭头在不同角度下绘制情况是否正常
drawArcArrow	是	是	箭头在不同角度下绘制情况是否正常
drawSelfArrow	是	是	绘制情况
drawPaper	否	是	不同纸带数量下纸带绘制情况是否正常

如表 4.1.1 所示，开发者对表中所列的绘制方法在开发过程中就即时进行了手动测试并查看视觉效果。并且每个绘制方法都有重点测试内容，在保证绘制情况完全正常的情况下，还要确认重点测试内容是否正常。

4.1.2 图灵机编辑模块

图灵机编辑模块的测试方法主要是在开发过程中对每个单独的功能进行模块测试，最后对整体的编辑模块进行测试，最后的测试包括对单带图灵机、多带图灵机、带子程序图灵机的编辑操作测试。测试内容如下表所示：

表 4.1.2 图灵机编辑逻辑测试内容

编辑模式	编辑操作	绑定的操作函数
选择	单击图灵机状态	tap
选择	移动图灵机状态	touchStart touchMove touchEnd
选择	长按图灵机状态	longTap
选择	单击状态转移函数	tap
创建状态	单击或点击后移动	touchStart touchMove touchEnd
创建状态转移函数	单击或点击后移动	touchStart touchMove touchEnd
创建子程序	单击	tap
删除	单击图灵机状态	tap
删除	单击状态转移函数	tap
删除	单击图灵机子程序模块	tap
撤回	单击撤回按钮	undo
恢复	单击恢复按钮	rollback
截图	单击截图按钮	savePic
保存文件	单击保存文件按钮	saveFile
即时模拟	单击模拟器按钮	gotoSimulator

其中对于状态转移函数编辑的两个功能，单击状态转移函数修改内容以及单击或点击移动创建状态转移函数，还要针对其内置的状态转移函数属性解析函数进行测试。

4.1.3 图灵机文件管理

图灵机文件管理模块是最先完成开发的模块，该模块在开发完成后立即接受了测试。由于该模块实际上仅提供了比较稳定的文件读写功能，所以测试内容较少。

表 4.1.3 图灵机编辑逻辑测试内容

文件管理操作	绑定的操作函数
检测文件夹	onShow
创建文件夹	onShow
文件列表	onShow
文件删除	clearFile

表 4.1.3 展示了所需测试的主要内容。在检测和创建文件夹上，本项目在不同的平台分别进行了测试，首先是手机端初次打开小程序，此测试在 iPhone6s Plus 与 vivo X70 Pro+ 中测试成功，然后是 windows 平台在微信小程序开发工具生成的小程序文件夹中，将原本的文件夹路径进行修改，模拟没有文件夹的情况并让小程序自行去读取，经过测试，小程序在这种情况下能正常生成新的文件夹。

文件列表通常情况下会在 onShow 时一次性生成，但是在使用文件删除功能时，会对文件列表进行更新，所以该测试主要在文件删除功能下以及从其他页面跳转到文件列表页面的情况下进行。

4.1.4 图灵机模拟仿真

图灵机模拟仿真模块在整体开发完成后进行测试，对该模块的测试，本项目采用了不同的图灵机模型文件，针对性测试模拟仿真模块中对于异常情况的处理能力，并且针对性验证模拟仿真模块的功能是否正确。该模块经过了大量的测试，包括后续对其的黑盒测试，以保证该模块的高度稳定性。

首先本项目对不涉及运行功能的模块进行了测试，与模拟运行功能无关的模块包括输入待验证字符串功能与截图保存功能。接着对模拟运行功能的主要模块的每个分支设计了特别的文件用于验证运行功能的稳定性：

表 4.1.4 图灵机模拟仿真测试文件

序号	文件名	测试文件说明
1	a.json	在状态拥有多个相同读判定的状态转移函数时，测试运行路径是否能正常分支，并且同时测试快照切换功能是否正常
2	b.json	在状态拥有多个相同读判定的状态转移函数时，测试运行路径是否能正常分支，并且同时测试快照切换功能是否正常
3	c.json	在状态拥有多个相同读判定的状态转移函数时，测试运行路径是否能正常分支，并且同时测试快照切换功能是否正常
4	e.json	在状态拥有多个相同读判定的状态转移函数时，测试运行路径是否能正常分支，并且同时测试快照切换功能是否正常
5	f.json	常规单带图灵机单步执行，快速执行，运行快照绘制
6	5tape.json	常规多带图灵机单步执行，快速执行，运行快照绘制
7	2tape.json	常规多带图灵机单步执行，快速执行，运行快照绘制
8	subprog.json	常规子程序图灵机单步执行，快速执行，运行快照绘制
9	error.json	测试图灵机正确性检测逻辑

4.2 本章小结

本章简单介绍了在没有自动化测试工具的微信小程序平台对该小程序的稳定性进行测试的具体方式以及相关的测试内容。到此为止，该毕业设计对经典图灵机模拟仿真小程序的开发与测试完成。

第五章 总结和展望

5.1 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。

5.2 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] Chomsky N. Three models for the description of language. IRE Trans Inf Theor 2:113-124[J]. I nformation Theory, IRE Transactions on, 1956, 2(3):113-124.

[2] Chomsky N. On certain formal properties of grammars[J]. Information and Control, 1959, 2(2):1 37-167.

[3] Carlos I. Chesñevar, María L. Cobo, William Yurcik. Using theoretical computer simulators for f ormal languages and automata theory[J]. ACM SIGCSE Bulletin,2003,35(2).

[4] Pinaki Chakraborty, P. C. Saxena, C. P. Katti. Fifty years of automata simulation[J]. ACM Inroa ds,2011,2(4).

[5] Tuhina Singh, Simra Afreen, Pinaki Chakraborty, Rashmi Raj, Savita Yadav, Dipika Jain. Autom ata Simulator: A mobile app to teach theory of computation[J]. Computer Applications in Engine ering Education,2019,27(5).

[6] Losacco M, Ttodger S H. Flap: a tool for drawing and simulating automata[J]. In ED-MEDIA 93, 1993.

[7] George H. Mealy. A Method for Synthesizing Sequential Circuits[J]. Bell System Technical Journal,1955,34(5).

[8] Alonzo Church. Review: Edward F. Moore, Gedanken-Experiments on Sequential Machines[J]. Journal of Symbolic Logic,1958,23(1).

[9] Susan H. Rodger,Bart Bressler,Thomas Finley,Stephen Reading. Turning automata theory into a hands-on course[P]. Computer science education,2006.

[10] Susan H. Rodger, Eric Wiebe, Kyung Min Lee, Chris Morgan, Kareem Omar, Jonathan Su. Incr easing engagement in automata theory with JFLAP[J]. ACM SIGCSE Bulletin,2009,41(1).

[11] 吕兰兰,郭晓梅.JFLAP 在编译原理算法教学中的应用[J].湖南科技学院学报,2019,40(10):116-118.D OI:10.16336/j.cnki.cn43-1459/z.2019.10.037.

[12] Carlos H. Pereira, Ricardo Terra. A mobile app for teaching formal languages and automata[J]. Computer Applications in Engineering Education,2018,26(5).

[13] 胡嵩. 图灵的机器思维思想初探[D].华中师范大学,2017.

[14] Turing A M. On computable numbers, with an application to the Entscheidungsproblem[J]. J. of Mah,1936,58(345-363): 5.

[15] 陈有祺. 形式语言与自动机[M]. 机械工业出版社, 2008: 148-168.

[16] 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.

[17] 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.

[18] 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.

[19] 图灵机:停机问题,0 型语言[J].电子计算机参考资料,1977(Z2):59-66.

[20] 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.

[21] Kochhar P S , F Thung, Nagappan N , et al. Understanding the Test Automation Culture of Ap p 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。