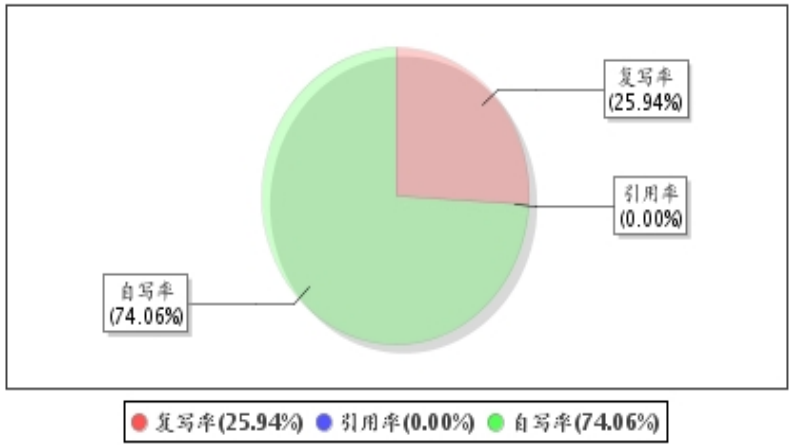
论文检测报告

报告编号： A5D2912D0F0940BEAF2322DBAAFC894B
送检文档： check2
论文作者： kk

文档字数： 24396
检测时间： 2015-05-18 00:00:35
检测范围： 互联网，中文期刊库（涵盖中国期刊论文网络数据库、中文科技期刊数据库、中文重要学术期刊库、中国重要社科期刊库、中国重要文科期刊库、中国中文报刊报纸数据库等），学位论文库（涵盖中国学位论文数据库、中国优秀硕博论文数据库、部分高校特色论文库、重要外文期刊数据库如Emerald、HeinOnline、JSTOR等）。

一、检测结果：

总相似比： 25.94% [即复写率与引用率之和]
检测指标： 自写率 74.06%复写率 25.94%引用率 0.0%
相 似 比： 互联网 9.88% 学术期刊 6.76% 学位论文 9.3%
其他指标： 表格 7 个 脚注 0 个 尾注 0 个



章节抄袭比
0.00% 2.3.4 MVC
0.00% 4.1 验证算法需求获取
0.00% 5.1 人机交互需求获取
0.00% 6.1 整体概览
0.00% 7.1 总结
0.00% [1] 胡军, 于笑丰, 张岩, 等.

二、相似文献汇总：

序号	标题	文献来源	作者	出处	发表时间
1	支持EBXML规范的WEB服务关键技术研究及应用	学位论文	邬俊毅	硕博学位论文	2005
2	基于.NET的网络考试系统的研究与实现	学位论文	胡大洋	硕博学位论文	2010
3	对象Petri网模型的复杂度度量	学术期刊	于瑞强 黄志球	解放军理工大学学报：自然科学版	2010
4	基于internet的设备远程监测诊断系统的设计与实现	学位论文	钱羽	硕博学位论文	2007
5	深度优先搜索-维基百科,自由的百科全书	互联网		互联网	
6	UML_快速入门_百度文库	互联网		互联网	
7	基于Petri网的并发程序测试路径生成-豆丁网	互联网		互联网	
8	一种自动搜索水准环及计算闭合差的方法研究	学术期刊	蒋宏飞 刘伟东 王文胜	测绘科学	2012
9	浅谈软件工程和CASE工具的运用	学术期刊	刘伟[1] 朱晓非[2] 聂亚	中国科技信息	2012
10	VP-UML系统建模工具研究-龙源期刊网-你喜欢的所有名刊大刊数字版...	互联网		互联网	

11	供应链信息共享问题研究	学位论文	张云涛	硕博学位论文	2002
12	基于手持移动设备上的考试平台	学术期刊	HUANG Xuan	南昌大学学报：理科版	2008
13	基于UML状态机-PNS集成的工作流建模与分析	学位论文	刘清秀	硕博学位论文	2005
14	基于Petri网的指挥控制流程仿真方法	学术期刊	张耀鸿 樊建才 廖晓林	系统仿真学报	2012
15	58java解析XML文件(DOM)	互联网		互联网	
16	基于领域本体的网络文本挖掘和知识验证方法	学位论文	罗贝	硕博学位论文	2005
17	拓扑排序-维基百科,自由的百科全书	互联网		互联网	
18	交互概述图- 维基百科,自由的百科全书	互联网		互联网	
19	XML解析技术及其在飞行数据存储及访问中的应用	学术期刊	张闻乾 王伟 段丽娟 李爱军	测控技术	2007
20	谁能帮我写一个acknowledgement。毕业论文的_百度知道	互联网		互联网	
21	基于工作流的ERP流程审批平台的设计与实现	学位论文	章阳	硕博学位论文	2011
22	基于面向对象UML技术的远程教育系统的建模开发	学位论文	李淑文	硕博学位论文	2005
23	基于UML和RUP的HMIS建模	学位论文		硕博学位论文	
24	基于boltzmann行动选择策略的网络蜘蛛theso设计与实现	学位论文	徐毅	硕博学位论文	2006
25	网络流量规范化的流量伪装模型的研究	学位论文	胡文心	硕博学位论文	2005

26	基于UML的软件开发方法研究	学术期刊	吴燕	民营科技	2011
27	拓撲排序-维基百科,自由的百科全书	互联网		互联网	
28	面向网格计算的描述语言研究	学位论文	孔亦南	硕博学位论文	2005
29	超声图像处理及测算系统的研究与实现-豆丁网	互联网		互联网	
30	轻量级工作流引擎的研究与实现	学位论文	张柳江	硕博学位论文	2005
31	基于on-the-fly的Petri网模型检查技术研究与实现	学术期刊	沈云付 解晓方	计算机应用与软件	2011

三、全文相似详情： （红色字体为相似片段、浅蓝色字体为引用片段、深蓝色字体为可能遗漏的但被系统识别到与参考文献列表对应的引用片段、黑色字体为自写片段）

2.3.4 MVC

MVC模式（Model-View-Controller）是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。MVC是一种框架式软件设计典范，旨在用这种设计分离业务逻辑和数据现实，在界面和用户的交互被改进或定制的时候不需要影响到业务逻辑，实现一种设计上的解耦。

（控制器 Controller）- 负责转发请求，对请求进行处理。

（视图 View） - 图形界面。

（模型 Model） - 程序应有的功能（实现算法等等）、数据管理和数据库设计(可以实现具体的功能)。

图 2.7 MVC模式

MVC有几大重要优点，对本项目的设计和代码编写十分有力，因此本项目在项目架构设计中选择了MVC架构。

2.3.5 java swing

Swing是一个为Java设计的GUI工具包。Swing是JAVA基础类的一部分。Swing包括了图形用户界面（GUI）器件如：文本框，按钮，分隔窗格和表。

Swing提供许多比AWT更好的屏幕显示元素。它们用纯Java写成，所以同Java本身一样可以跨平台运行，这一点不像AWT。它们是JFC的一部分。它们支持可更换的面板和主题（各种操作系统默认的特有主题），然而不是真的使用原生平台提供的设备，而是仅仅在表面上模仿它们。这意味着你可以在任意平台上使用JAVA支持的任意面板。轻量级组件的缺点则是执行速度较慢，优点就是可以在所有平台上采用统一的行为。

本项目采用java Swing进行界面设计和开发。

2.3 本章小结

本章主要关于是基于场景的并发模型存在一致性检验工具的总体概述，对相关的概念，工具以及使用的技术做一介绍。在后面的概要设计以及详细模块设计的部分，会再次引用本章节的相关知识。

第三章 预处理模块的设计与实现

3.1 读文件

我们采用开源工具JDOM来解析IOD和Petri网的xml文件。首先我们需要一个合适的解析器对象，由于本项目并不需要修改xml文件的树结构，为了可以占用更少内存并且获得更快的解析速度，我们采用SAX的方式解析本xml文件，所以本项目采用SAXBuilder而不是DOMBuilder作为解析器对象。

实例化一个合适的解析器对象

```
SAXBuilder builder = new SAXBuilder();
```

以包含XML数据的文件为参数，构建一个文档对象

```
Document doc = builder.build(new File(path));
```

获到根元素

```
Element rootEl = doc.getRootElement();
```

一旦你获取了根元素，你就可以很方便地对它下面的子元素进行操作了，下面对Element对象的一些常用方法作一下简单说明：

表 3.1 jdom元素对象常用方法

3.2 IOD建模与解析

本小节讨论IOD的建模与解析，图3.1展示的是Visual Paradigm将图2.2 IOD图导出的xml文件的结构。我们根据这个文件分析xml文件的建模和解析方法。

整体来看，IOD根节点project下有三个子节点，分别是ProjectInfo，Models和Diagrams，ProjectInfo主要是一些项目信息，Diagrams描述的是如何绘图，这两个我们不用关心，本项目的重点在于关注如何对IOD图建模，也就是关注Models这个节点下的信息。

图 3.1 IOD xml文件结构示例

Models节点下主要有六类节点，分别是Frame， ModelRelationshipContainer, InitialNode, ActiityFinalNode, Interaction, DataType。为了解析出IOD的完整结构，需要对前五种节点进行分析和解析。

表 3.2 IOD xml文件节点意义

图 3.2 IOD模块类图

图3.2所示项目交互概览图（IOD）模块的类图，在IOD实例中保存了6个键值对。为了更好得与interaction对应上，frameMap用其名字作为键，其余Map均使用id作为键。

通过IOD类中一系列parse相关的方法，将xml文件读入内存。

3.3 Petri网建模与解析

Petri网建模相对于IOD建模要简单很多，图3.3所示一张由PIPE工具绘成的Petri网图，由PIPE生成的xml文件格式如图3.4所示。

图 3.3 Petri网示例图

图 3.4 Petri网xml文件格式

在Petri网生成的xml文件中，主要包含place，transition和arc三类元素，表3.3对三类元素进行了分析和解析

表 3.3 Petri网xml文件节点意义

图 3.5 Petri网模块类图

图3.5是Petri网模块的类图，在Petri中，库所（place）和变迁（transition）均采用键值对方式存储。库所的健为其唯一标识Id，值为其名称，在本例中健值相同。变迁的健为其唯一标识Id，值为其名称（Id和名称用竖线隔开）。有一个列表存储所有有向弧，并且存储了所有初始的库所。

通过Petri类中得parse方法，将xml文件读入内存。

3.4 消息序列生成

3.4.1 消息序列生成需求获取

由于基于场景的并发模型存在一致性检验算法需要遍历IOD图中的时序图中得消息队列，所以需要记录每个时序图的所有可能的消息队列。在时序图中，消息的接受和发送我们称为事件，也就是2.1节所述的激活。事件的顺序（也就是消息发送和消息接受）是有时序图中对象的控制流可见的顺序和事件间的因果依赖推导得出的。为了不失一般性，在下面三种情况下我们认为事件e在e’之前：

因果性：一个发送事件e和其相应地接受事件e’。比如说图2.1所示的时序图中，e5在e6之前。

可控性：事件e和事件e’在一个对象上，并且e在e’之上，e’是发送事件。这个顺序说明了一个发送事件可以等待其他事件发生，另一方面可以说我们对接受消息顺序的可控性很低。比如说图2.1所示的时序图中，e6在e9之前。

先进先出顺序：接受事件e在接受事件e’在同一个对象上，并且e在e’之上。相对应的发送事件e1和e1’在另一个对象上并且e1在e1’之上。

3.4.2 消息序列生成分析与设计

首先，需要能给所有的消息排序以确定哪个在上哪个在下，也就是要两个消息之间进行大小比较。分析Visual Paradigm生成的IOD图，可用消息的序列号进行排序，比如1.1在1.2之上，1.1在1.1.1之上。给Message类实现Comparable接口，并实现了compareTo方法：

图 3.6 Message类compareTo方法

我们可以将消息序列生成问题简化为拓扑图排序问题，每一个事件可以作为拓扑图的一个节点，每个约束条件作为拓扑图的边，那么求消息队列的所有可能序列就转化为求拓扑图的所有可能排序。

为了说明方便，我们以图3.7所示的时序图为例，这张图包含了3.4.1小节所述的三种约束，加入约束条件后相应的拓扑图如3.8所示，先加入6个节点，然后根据三种不同类型约束依次加入：

因果性约束：加入not_approved!到not_approved?，not_possible!到not_possible？和option!到option?三条约束。

可控性约束：加入not_approved?到not_possible!，not_approved?到option!和not_possible!到option!三条约束。

先进先出约束：加入not_possible?到option?一条约束。

如图3.8所示，3.7时序图的所有可能的序列即为所有3.8可能的拓扑排序。

图 3.7 RefuseWith时序图　　图 3.8 RefuseWith拓扑图

3.4.3 拓扑排序算法及相关描述

首先构建拓扑图的数据结构，在这个数据结构中保存着所有的边和节点。节点中保存着名称，出度和入度，出度通常指有向图中某点作为图中边的起点的次数之和，入度通常指有向图中某点作为图中边的终点的次数之和。在拓扑图中用键值对存储节点，健为其唯一标志名称。边包含了发出边的节点名和接受边的节点名，在拓扑图中用一个数组存储所有边。拓扑图数据结构包含添加节点，删除节点，添加边等方法，其中删除节点，会删除与这个节点有关的所有边。

根据3.4.2的方法将消息排序后，就可以确定RefuseWith中的约束关系，构建拓扑图。然后就根据拓扑排序算法将所有可能的排序算法保存在一个Vector中，具体算法见图3.9所示：

图 3.9 拓扑排序递归算法

算法需要一个临时的栈来存放当前的路径，还需要一个Vector保存所有路径。算法的主要思想是深度优先遍历，搜寻拓扑图中入度为0的节点，将其加入栈中。将这个节点删除后继续搜索入度为0的节点，直到没有入度为0的节点。如果这个时候还有节点没有遍历到，说明拓扑图中存在环，也就是没有拓扑排序，如果所有节点都被遍历了，则将当前栈中保存的临时路径加入Vector中。依次返回，在搜寻其他入度为0的节点。

如图3.8所示的拓扑图，经过上述算法后，有两条拓扑路径：

not_approved!, not_approved?, not_possible!, not_possible?, option!, option?

not_approved!, not_approved?, not_possible!, option!, not_possible?, option?

也就是说，图3.7所示的时序图存在两条消息序列。

图 3.10 消息序列模块类图

第四章 验证算法核心模块的设计与实现

4.1 验证算法需求获取

验证算法模块是本工具的核心模块，本小节关注存在一致性检验，根据基于场景的规范检查Petri网。本论文中基于场景的规范就是包含IOD和时序图的UML交互模型，IOD是时序图的组成。存在一致性检验是检查一个给定IOD描述的场景是否一定会在Petri网中发生，或者是否一个IOD中禁止的场景一定不会在Petri网中发生。比如说，图4.1是两个可以描述铁路过路系统基于场景规范的时序图。左边的图是描绘火车过路前正常的场景，这个应该在图2.3所示的Petri网中出现。右边的图中Crossing_secured这个消息在闸放下之前就发给了传感器，这个不会在图2.3所示Petri网中出现。

图 4.1 铁路过路系统存在一致性检验

验证算法要求能够在IOD图中找到一条路径在Petri网中发生，如果找到这样一条路，说明基于场景的并发模型存在一致性检验成功，如果不存在这样一条路，说明检验

失败。

4.2 验证算法分析与设计

验证算法的核心采用深度优先遍历，搜索IOD图中的时序图，在时序图中选取一条消息序列在Petri网中看是否存在相应的路径，如果存在则递归下去遍历下一个节点，如果不存在则换一条消息序列试一下。如果此时序图中的所有消息序列都不能在Petri中找到相应的路径，则遍历前继节点的另一个后继节点，如果所有后继节点都不行，则说明不存在这样一条路径，存在一致性验证失败。如果遍历到IOD图的终止节点，就说明找到了一条IOD中的路径在Petri网中有相应的路径。

由于允许在IOD图中循环，为了避免在遍历时无限循环，在算法中需要做一个标记，如果走过一个时序图，标记走这个时序图在Petri网中相应的位置，如果再次走到这个时序图时还是在Petri网中这个位置，就避免再走这条路。

4.3 验证算法实现

图 4.2 存在一致性检验验证算法的实现

如图4.2所示为基于场景的并发模型存在一致性检验的核心算法，一下是对其的分析和解释：

首先是从IOD图的每个初始节点依次出发，只要找到这样的一条路径就停止遍历。

由于IOD图中的节点和时序图存储在不一样的地方，为了将两者关联起来，我们采用IOD图的名称作为唯一标识，可以根据名称找到IOD图中的节点对应的时序图。

如果是初始节点，则将自己加入路径，遍历他的所有后继节点。如果是终止节点，则说明找到了这条路，将自己加入路径后返回正确。

如果是普通节点，遍历其相应时序图的每一个消息序列，检查是否走过这个时序图和相应的Petri网，如果走过则检查下一跳消息序列，如果没走过则将这个对应加入纪录中。在Petri网中走过这个序列，如果检查成功，则继续深度遍历其后继节点。如果检查失败，检查下一条消息序列。

第五章 人机交互模块设计与实现

5.1 人机交互需求获取

基于场景的并发模型存在一致性检验工具的前端需求非常简单，用户选择IOD的xml文件并上传，选择Petri网的xml文件并上传，然后检查一致性，展示结果给用户。

在现代软件设计中，一个好的人交互体验至关重要。以下是本项目人机交互设计的几条标准。

灵活性：界面可以随意拉伸以适应不同规格，不同分辨率的屏幕。

可用性：由于本工具在处理大文件时可能需要耗时比较久，需加入控制台以告知用户现在进行到哪一步。

安全性：避免用户做出错误操作，如遇到错误用合理的方式告知用户。

5.2 界面设计与实现

为适应灵活性需求，本项目采用java swing中的GridBagLayout作为工具界面的布局管理器，GridBagLayout是Swing中最灵活但也是最复杂的布局管理器。GridBagLayout从它的名字中你也可以猜到，它同GridLayout一样，在容器中以网格形式来管理组件。但GridBagLayout功能要来得强大得多。

GridBagLayout管理的所有行和列都可以是大小不同的。

GridLayout把每个组件限制到一个单元格，而GridBagLayout并不这样:组件在容器中可以占据任意大小的矩形区域，GridBagLayout通常由一个专用类来对他布局行为进行约束，该类叫GridBagConstraints。

经过设计，最后效果图如图5.1所示，可以随意拉伸，以适应不同屏幕尺寸和像素。

如图5.1所示，在界面设计中加入了控制台这个组件，可以提醒用户当前处理步骤。如果遇到错误也可以方便得查看在哪一步出现了问题，关于控制台功能的详细实现在5.3节中描述。

本项目的安全性体现在两个方面，一是避免用户犯错，比如用FileNameExtensionFilter禁止上传非xml文件，二是出错会进行提示而不是程序崩溃，比如如果用户没有选择文件会用JOptionPane.showMessageDialog进行提示。

图5.1 界面设计效果图

5.3 处理信息实时展现设计与实现

控制台消息展现体现了本系统的MVC设计思想，下面对这一模块分析与展示：

为了实现能实时在界面显示当前处理进度的控制台功能，项目中使用了MessageShower这个接口，由界面实现它，控制器持有其引用，如图5.2所示，这符合MVC设计原则。

图 5.2 MVC部分类图

如图5.3所示，MessageShower中申明了两个方法，在view中实现了：这样当controller进行处理的时候就可以调用MessageShower的实例进行展示。

图5.3消息展示核心代码

第六章 整体概览与实例展示

6.1 整体概览

图6.1项目整体类图

图6.1所示是基于场景的并发模型存在一致性检验的整体类图展示，项目各模块之间划分得比较清晰，模块功能高内聚，并且用了设计模式等方法解耦合，保证了各模块的可扩展性。

6.2 实例展示

图 6.2 ATM系统部分存在一致性检验结果

下面用几个实例来展示本工具的效果：

首先是图2.2所示的IOD图和图3.3所示的Petri网的结果如图6.2，6.3所示。

下面是一个完整的ATM系统存在一致性检验工具的测试，图6.3为其IOD图，图6.4为其Petri网图。运行结果为：[initial, card!, card?, REQ_pin!, REQ_pin?, enter_pin!, enter_pin?, verify!, verify?, processing!, processing?, return_card!, return_card?, abort!, abort?, timeout! ,timeout? , card!, card?, REQ_pin!, REQ_pin?, enter_pin!, enter_pin?, verify!, verify?, processing!, processing?, invalid!, invalid?, INV_MSG!, INV_MSG?, take_card!, take_card?, final]，符合预期。说明此工具有效。

图 6.3 ATM IOD图

图 6.4 ATM Petri网

第七章 总结与展望

7.1 总结

本篇论文对基于场景的并发模型存在一致性检验工具做了总体的详细概述，重点描述了工具的设计与实现。首先介绍了相关概念，工具和技术，然后分别介绍了工具系统的三个模块。

第一个模块是预处理模块，介绍了系统如何读文件和解析由Visual Paradigm生成的IOD的XML文件和由PIPE生成的Petri网的XML文件。并且重点介绍了基于拓扑排序的消息序列生成算法。

第二个模块是验证算法核心模块，重点介绍了基于深度优先遍历的验证遍历算法。

第三个模块是人机交互模块，介绍了界面的设计和实现，并重点介绍了基于MVC设计模式的控制台消息展示功能的设计与实现。

最后用两个实例展示说明了此工具的有效性。

7.2 不足和展望

基于场景的并发模型存在一致性检验工具为本人独立开发，由于水平有限，有一些不足和需要改进的地方。

没有进行XML的校验。虽然本工具只允许用户上传xml文件，但是没有对xml文件的内容进行校验，如果用户上传错误或者不符合规范的xml文件，可能会导致程序崩溃。在以后的开发中可以进行校验和提醒，防止出现这样的情况。

没有进行强制一致性检验。本工具只能进行了存在一致性检验的验证，下一步可以在工具中加入强制一致性检验的功能。

在人机交互上还不够友好。下一步可以考虑输出不止是一条路径而能在IOD图和Petri网中体现，就会更加直观和友好。

参考文献

[1] 胡军, 于笑丰, 张岩, 等. 基于场景构件式实时软件设计的一致性检验[J]. 软件学报, 2006, 17(1): 48-58.
[2] Robert Sedgewick，Kevin Wayne，谢路云，算法（第4版），人民邮电出版社，2012
[3] Class diagram - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Class_diagram
[4] Unified Modeling Language - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Unified_Modeling_Language
[5] Sequence diagram - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Sequence_diagram
[6] Interaction overview diagram - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Interaction_overview_diagram
[7] Petri net - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Petri_net
[8] 康保军. VP-UML 系统建模工具研究[J]. 软件工程师, 2014, 7: 019.
[9] List of Unified Modeling Language tools - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools
[10] Platform Independent Petri net Editor 2：http://pipe2.sourceforge.net/
[11] Software Design Tools for Agile Teams, with UML, BPMN and More：http://www.visual-paradigm.com/
[12] 霍敏霞. 基于Petri网的并发程序测试路径生成[D]. 西南大学, 2011. DOI:10.7666/d.y1882620.
[13] JDOM_百度百科：http://baike.baidu.com/link?url=KSuCWlVadOrIN2ziLzkw4YfjLfaeOuh-3jnn7fsJhbWGkETL8ukByJ7RidseJ3GwrMKTXbTsr4nBYFWMu4i34_
[14] JDOM：http://www.jdom.org/
[15] Depth-first search - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Depth-first_search
[16] Topological sorting - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Topological_sorting
[17] Modelviewcontroller - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Model-view-controller

[18] Swing (Java) - Wikipedia, the free encyclopedia：http://en.wikipedia.org/wiki/Swing_(Java)
[19] 深入学习GridBagLayout：http://www.cnblogs.com/renyuan/archive/2012/10/15/2723809.html

致谢

大学四年的时间悄然逝去，转瞬已到尾声，经过几个月的艰苦奋斗，这篇毕业论文终于要接近尾声了。在项目和构建论文的过程中，我遇到了很多难题，也有很多迷茫的时刻，再次，我要向所有向我提供过指导、帮助和支持的人表示最真诚的感谢。

首先，我要感谢我的指导老师潘敏学老师。从毕业论文的选题，到项目的设计和实现，到论文提纲和内容的拟定，以及论文内容的修改和最后的定稿，潘老师一直都给予我细致的教诲和耐心的指导，特别是在我项目遇到瓶颈不知道怎么进行的时候，他耐心得一遍一遍给我解释算法，鼓励我不要放弃继续尝试，最后在潘老师的帮助下我攻克一个有一个难关，完成了项目。在此，我想要再次向潘老师表示我衷心的谢意。

其次，我要感谢学院各位领导和老师，学院为我们提供了良好的学习环境和设备支持，老师勤勤恳恳备课上课，让我在本科阶段打下了扎实的计算机基础，为我研究生生涯做好了铺垫。我还要感谢在软院一起学习，生活，做项目的同学们，感谢你们提供的无私帮助和照顾。特别感谢舍友尹心成，武慧凯，周海兵和姚远同学在本项目遇到算法瓶颈时帮我一起分析和给予宝贵的建议，没有你们的帮助，本项目很难完成。感谢王晨和宗岩琦同学在我遇到问题时困苦时的陪伴和开导。

由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正。

本 科 毕 业 设 计

院 系　　软件学院

专 业　　软件工程

题 目　基于场景的并发模型存在一致性检验工具的设计与实现

四、指标说明：

1. 总相似比即类似于重合率。总相似比即送检论文中与检测范围所有文献相似的部分（包括参考引用部分）占整个送检论文的比重，总相似比=复写率+引用率。
2. 引用率即送检论文中被系统识别为引用的部分占整个送检论文的比重（引用部分一般指正确标示引用的部分）。

3. 自写率即送检论文中剔除雷同片段和引用片段后占整个送检论文的比重，一般可用于论文的原创性和新颖性评价，自写率=1-复写率-引用率。
4. 复写率即送检论文中与检测范围所有文献相似的部分（不包括参考引用部分）占整个送检论文的比重。
5. 红色字体代表相似片段；浅蓝色字体代表引用片段、深蓝色字体代表可能遗漏的但被系统识别到与参考文献列表对应的引用片段；黑色字体代表自写片段。

五、免责声明：