基于情态演算的UML形式化验证与OCL约束自动生成研究

硕士毕业论文答辩

导师：杨宗源 教授

学生：谭力

2010年5月26日 信息楼629会议室

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

引言

- 软件中的错误和漏洞导致灾难性后果
  - [1996] 阿丽亚娜5号运载火箭失事
  - [1990s] Ashton-Tate公司数据库管理软件dBASE
- 应对策略与解决方案
  - 软件测试 缺点：时间+费用开销大、不严密
  - 软件验证 手动/自动 设计阶段做/测试阶段做
  - 可信软件 国内外关注热点：拨款多、参与者多

引言

- 软件架构是软件的核心结构与行为，因而软件架构的设计是软件设计的核心
  - 软件架构 ⟷ 代码开发、软件维护和软件重构
- 软件架构设计 → 建模活动 → UML
  - UML模型的正确性？人工手动验证？精确+完备
- 形式化方法 → 形式化验证工具
  - 问题1：输入是形式化语言而非图形化的表示UML
  - 解决：把图形化表示转化为这类工具可识别的输入，从而完成对软件设计或程序的自动验证
  - 目标：可验证的软件设计 半自动化 >> 人工手动

引言

- 形式化方法 ⟶ 形式化规范说明工具
  - 问题2：OCL约束需要人工手动编写 效率低+不完整
  - 解决：为UML类图自动生成OCL约束模板
  - 目标：以防遗漏 + 完善已有
- 本章小结
  - 背景：软件安全问题日益严峻并且越发受到重视
  - 对策：为软件设计标准建模语言UML赋予形式化语义并加以形式化验证，同时附加一些形式化约束来提升其正确性和精确性
  - 目标：在软件设计阶段就修正错误，以避免错误编码，造成软件测试时不必要的系统开销

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

UML与形式化方法

- 统一建模语言UML简介
  - 面向对象建模方法中最成功的一种 应用广+国际标准

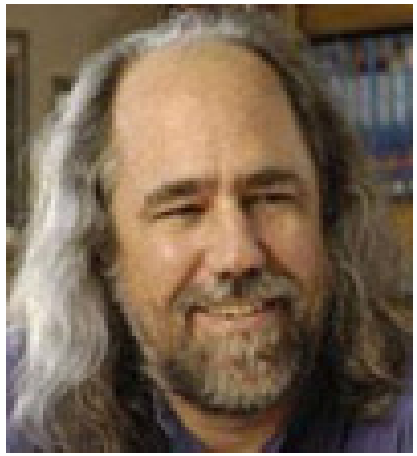
Rational
公司三巨头

常用UML建模工具:

IBM Rational Rose

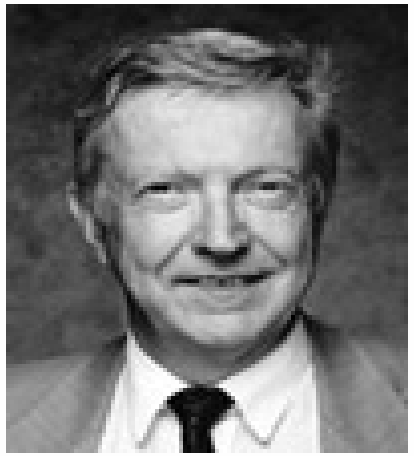
MS Visio　Together

VisualUML　ArgoUML



Grady Booch

James Rumbaugh



Ivar Jacobson

95.10 V0.8　96.6 V0.9　96.10 V0.91　97.1 V1.0　97.11 V1.1　98.7 V1.2　99.6 V1.3　01.2 V1.4　03.3 V1.5　04.10 V2.0　09.2 V2.2 ……

UML与形式化方法

- 统一建模语言UML简介
  - 优点：交互性好、效率高、可扩展性+可维护性佳
  - 缺点：图形化表示不精确 UML形式化描述与验证
- 形式化方法简介
  - 定义：是一类基于数学的用于精确化规范说明、开发和验证软件和硬件系统的多种方法的总称。
  - 应用：对软件和硬件设计使用形式化方法是为了通过利用适当的数学分析方法，来保证设计的正确性、可靠性和健壮性。
  - 分类：形式化规范说明 + 形式化验证
    - 定理证明
    - 模型检测
    - 自动测试用例生成

UML与形式化方法

- UML的形式化研究
  - 为UML提供一套较完整的形式化语义框架 欧洲多
  - 用不同形式化方法来验证UML模型中的某些属性
  - UML子图转换为形式化语言+高级语言代码 Hugo/RT
- 我还能做些什么？
  - 提供多个UML子图的统一形式化语义框架 比较少见
  - 为UML类图自动生成OCL约束模板 更加少见
  - 定义UML常见语法和语义错误并加以验证 角度不同
  - 工具化 + Web化 + 用户友好化 make a difference

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

基于情态演算的UML类图形式化描述

- UI

定义 I：UML 类图的一种形式化结构被定义为一个七元组：

$$\chi c ::= \langle Name, Cl, As, Gen, SharedAg, CompAg, Dep \rangle$$

其中各元素的含义如下：

Name：类图的名字。可为空；

Cl：类图中所有类的有限集合。集合元素 c1 是一个类，记为 c1 ∈ Cl；

As：类图中所有两个类之间的关联关系的有限集合。集合元素是一个双射，如 c1 ↔ c2，记为(c1, c2) ∈ As，当且仅当类 c1 和 c2 之间存在一个关联关系；

Gen：类图中所有子类到父类的继承关系的有限集合。集合元素是一个单射，如 c1 → c2，记为(c1, c2) ∈ Gen，当且仅当类 c1 继承 c2；

SharedAg：类图中所有部分类到共享聚合类的共享聚合关系的有限集合。集合元素是一个单射，如 c1 → c2，记为(c1, c2) ∈ SharedAg，当且仅当类 c1 是部分类而类 c2 是共享聚合类；

CompAg：类图中所有部分类到复合聚合类的复合聚合关系的有限集合。集合元素是一个单射，如 c1 → c2，记为(c1, c2) ∈ CompAg，当且仅当类 c1 是部分类而类 c2 是复合聚合类；

Dep：类图中所有友元类到独立类的依赖关系的有限集合。集合元素是一个单射，如 c1 → c2，记为(c1, c2) ∈ Dep，当且仅当类 c1 是独立类 c2 的友元类，即 c1 依赖 c2。

基于情态演算的UML类图形式化描述

n U

表 3-1　UML 类图元素与简单数理逻辑以及情态演算元素的对应关系

UML 类图元素	简单数理逻辑元素	情态演算元素	说明
类（Cl）	命题逻辑/一阶逻辑、集合论	函数流（Functional Fluent）	函数流也具有静态属性，即可以把一个类的定义看作为一个函数流，如 Class(c1)。
关联关系（As）	命题逻辑/一阶逻辑、集合论	关系流/动作（Relational Fluent/Action）	关系流和动作都可用来恰当地描述类之间的关联关系。
继承关系（Gen）	一阶逻辑、集合论	动作（Action）	继承关系不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
共享聚合关系（SharedAg）	一阶逻辑、集合论	动作（Action）	共享聚合关系不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
复合聚合关系（CompAg）	一阶逻辑、集合论	动作（Action）	复合聚合关系不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
依赖关系（Dep）	一阶逻辑、集合论	动作（Action）	依赖关系不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。

基于情态演算的UML状态图形式化描述

- UM

Gd：状态图中所有令转换成功发生的监护条件的有限集合。集合元素 gd1 是一个监护条件，记为 gd1 ∈ Gd。其中 gd1 被定义为一个二元组，记为 gd1 ::= <name, boolExp>，其中参数 name 表示监护条件的名字，参数 boolExp 表示监护条件的布尔逻辑表达式；

At：状态图中所有转换发生时激活的动作的有限集合。集合元素 at1 是一个动作，记为 at1 ∈ At。其中 at1 被定义为一个二元组，记为 at1 ::= <name, boolExp>，其中参数 name 表示动作的名字，参数 boolExp 表示动作的布尔逻辑表达式；

Ts：状态图中所有状态之间的转换的有限集合。集合元素 ts1 是一个转换，记为 ts1 ∈ Ts。其中 ts1 被定义为一个五元组，记为 ts1 ::= <name, tg, gd, stSrc, stTg>，其中参数 name 表示转换的名字，参数 tg 表示引起该转换的触发事件，参数 gd 表示令该转换成功发生的监护条件，参数 stSrc 表示发出该转换的源状态，参数 stTg 表示该转换到达的目标状态；

注：初始状态和终止状态是状态的一种，这里把他们单独列出是因为初始状态和终止状态一般没有 stVar 和 atv 这两个参数，并且初始状态和终止状态涉及的转换也一般没有 tg 和 gd 这两个参数，跟一般的普通状态结构不同，故区分之。和信号，即 type ∈ (‘calling’, ‘changing’, ‘time’, ‘singal’)；

基于情态演算的UML状态图形式化描述

- U

表 3-4　UML 状态图元素与简单数理逻辑以及情态演算元素的对应关系

UML 状态图元素	简单数理逻辑元素	情态演算元素	说明
状态（St）	命题逻辑/一阶逻辑、集合论	函数流（Functional Fluent）	函数流也具有静态属性，即可以把一个状态的定义看作为一个函数流，如 State(st1)。
触发事件（Tg）	一阶逻辑	动作（Action）	触发事件不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
监护条件（Gd）	一阶逻辑	动作（Action）	监护条件不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
动作（At）	一阶逻辑	动作（Action）	动作不会随着情景的改变而改变，可以用函数来表示，这符合动作的基本属性。
转换（Ts）	命题逻辑/一阶逻辑、集合论	关系流/动作（Relational Fluent/Action）	关系流和动作都可用来恰当地描述状态之间的转换。

UML模型错误类型定义

- 研究现状
  - 在用UML设计目标系统时，软件设计人员往往会因为不符合UML的语法规范和违背基本逻辑而产生错误。在代码写出之前，很难发现这些错误。一般情况下，直到软件实现（编码）阶段后的代码测试阶段，设计UML模型时的语法和语义错误才在编译器的帮助下，由程序员发现代码中的错误而发现。本文将给出直接在设计阶段发现UML语法和语义错误的方法，以避免不必要的系统开销。
  - 通过上述UML模型到情态演算的转换，可以对生成的Prolog代码进行分析从而发现UML模型中的设计错误。在对可能的错误进行验证之前，本文首先给出在几种常见的UML模型设计错误的定义，为6.3小节中的验证步骤作好准备。为了简单起见，以下错误的设计实例只给出最简洁的形式，附加信息都被省略。

UML模型错误类型定义

- 语法错误
  - UML语
  - 监护条

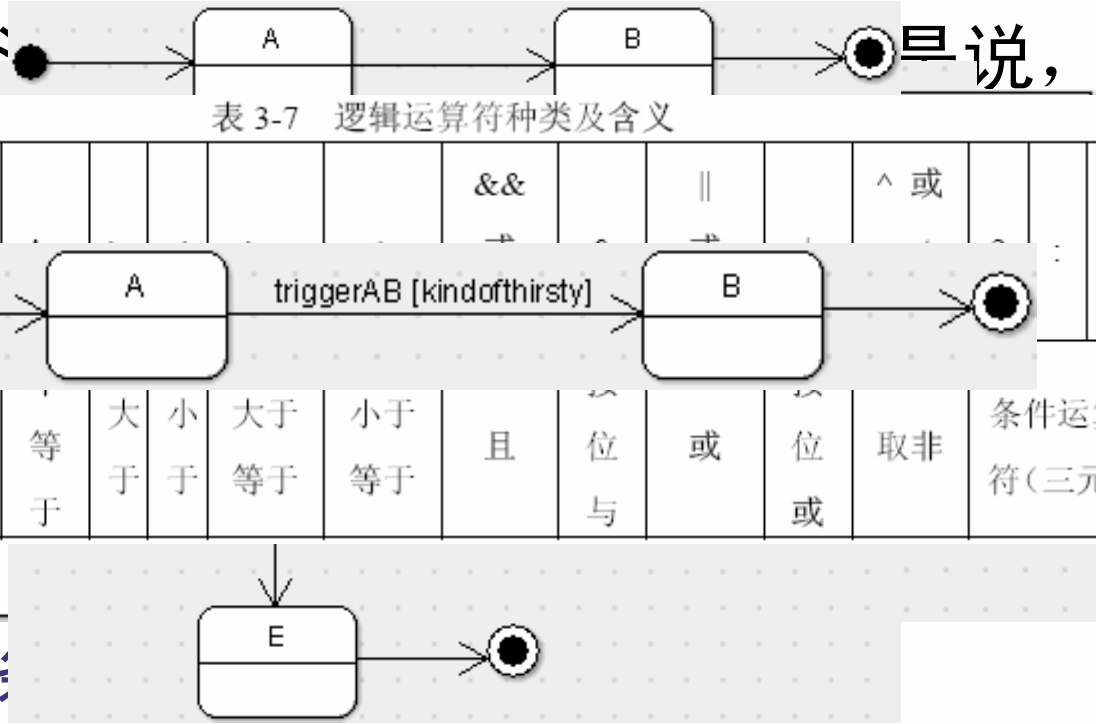
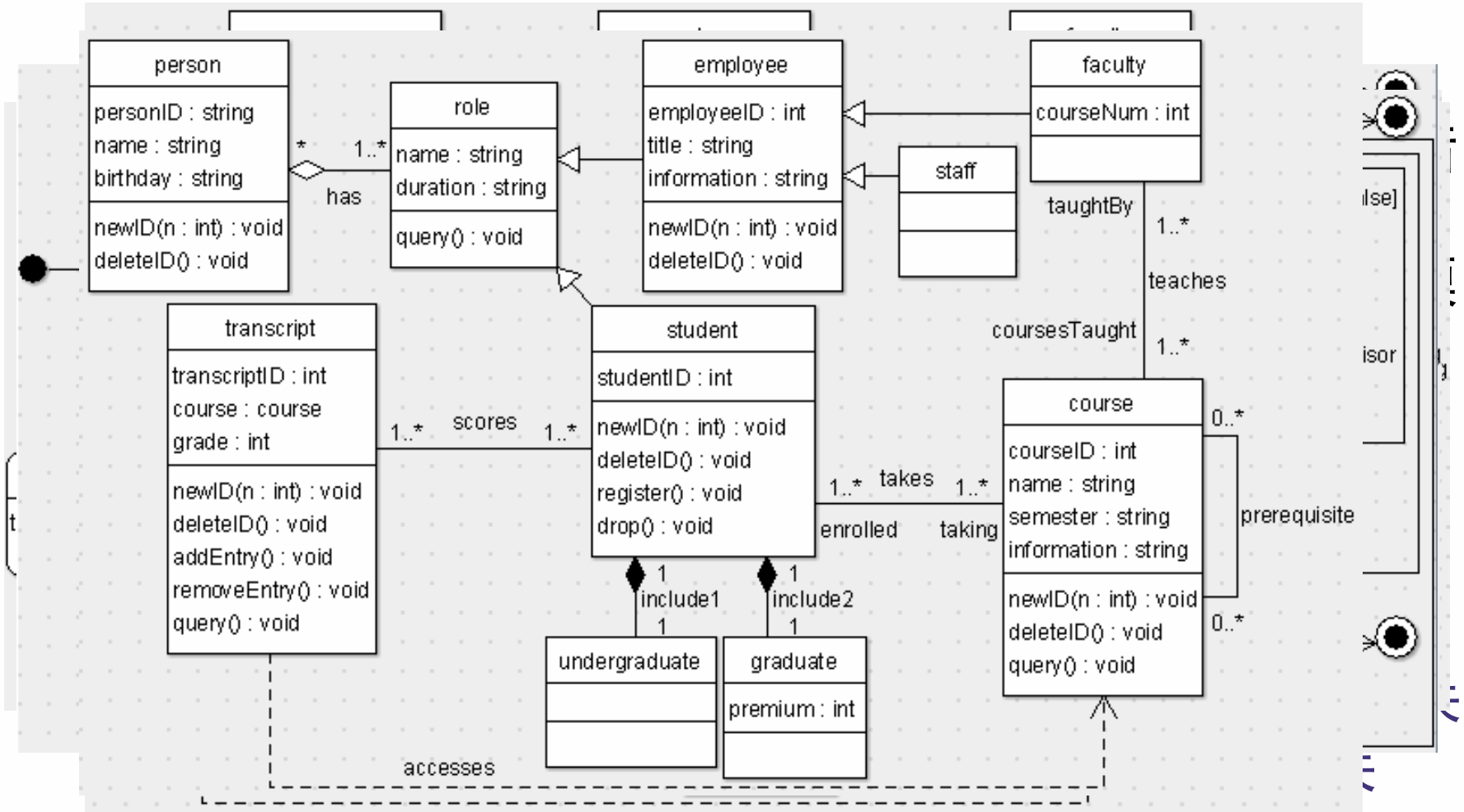


表 3-7　逻辑运算符种类及含义

表示法	=							&&		||		^ 或		:	;
含义	等于	是否等于	等于	大于	小于	大于等于	小于等于	且	位与	或	位或	取非	条件运算符（三元）		

旦说，这类错
典型

UML与形式化方法



- 需求逻辑错误
  - 老师可以做学生

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

OCL约束自动生成

- 研究方案
  - 本章主要介绍如何利用词法分析技术找出需要建立OCL约束的UML元素，并给出相应的OCL约束应用对象提取算法，最后达到为UML模型自动生成OCL约束模板的目标。
- OCL约束的应用范围
  - OCL的应用对象：类/接口/属性/操作等UML元素
  - 这些元素在UML类图中的应用很广
  - UML类图是整个系统的核心结构框架，是建立其他UML子图的基础

OCL约束自动生成

- 前提条件
  - 对于领域相关的OCL约束，即需要在理解UML类图元素语义的基础上建立的OCL约束，由于过于复杂，不在本文的讨论范围之内。
- 基于词法分析的OCL约束应用对象提取
  - 最重要的一个环节：找准应用对象，即如何找出需要建立OCL约束的类、接口、属性和操作
  - 利用编译技术中的词法分析技术：对研究对象通过词法分析的方式来筛选出符合建立OCL约束标准的对象 遍历整个XMI DOM来给出OCL约束模板

OCL约束自动生成

- 目标应用对象定义

  基于上述讨论，下面给出如下几种UML类图中需要建立OCL约束的目标应用对象的识别特征：

  - 类的“标识”类属性
  - 类的“标识”类属性的“新建”和“删除”操作
  - 集合类元素的“添加”和“去除”操作
    - 对于本身作为一个集合或容器的类，如“成绩单”类，该类的一个对象，即一份成绩单，可以拥有多条子记录，每条子记录对应一门科目的考试成绩。显而易见，对这些子记录的“添加”和“去除”操作也是必需的（“编辑”操作可以通过先“添加”后“去除”来实现）

OCL

- OCL

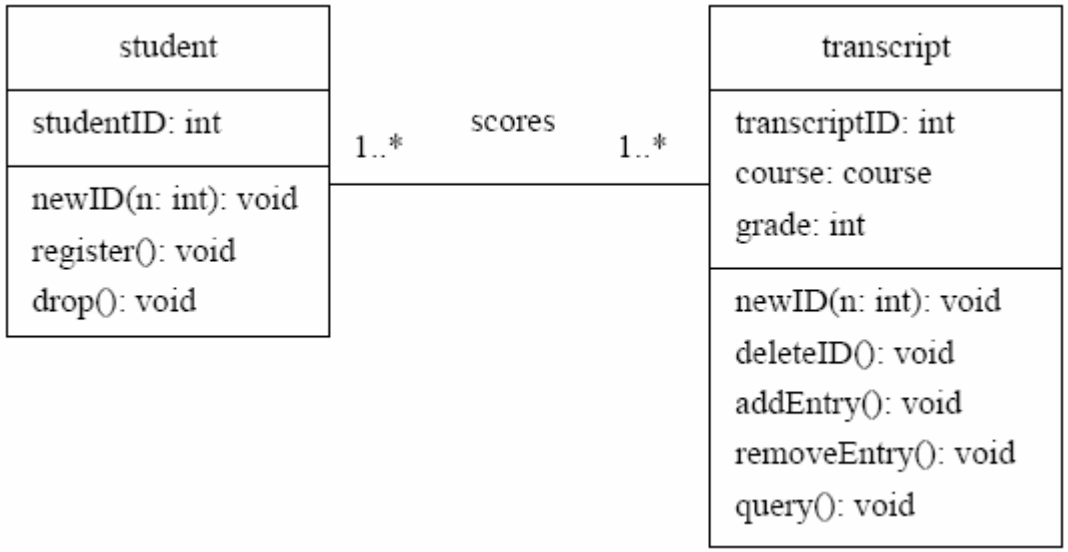
算法 3：OCL 约束应用对象提取算法

Procedure abstract(O_i)

```
1: for (each attribute in CD_j) do
2:     if (attribute.id match pattern_a1) then
3:         addOCL(idOCL);
4:     end if
5:     if (attribute.name match pattern_a2) then
6:         addOCL(nameOCL);
7:     end if
8: end for
9: for (each operation in CD_j) do
10:    if (operation.name match pattern_o1) then
11:        addOCL(newOprOCL);
12:    end if
13:    if (operation.name match pattern_o2) then
14:        addOCL(deleteOprOCL);
15:    end if
16:    if (operation.name match pattern_o3) then
17:        addOCL(addEntryOprOCL);
18:    end if
19:    if (operation.name match pattern_o4) then
20:        addOCL(removeEntryOprOCL);
21:    end if
22: end for
```

OCL约束自动生成

- 实例演示
  - 下面给出一个实例来说明该OCL约束模板自动生成算法的执行效果。首先设计一个UML类图如下:



OCL约束自动生成

- n

```
context student
inv: studentID >= 0 and studentID <= 99999999
context student::newID(n:int)
pre: n >= 0 and n <= 99999999
post: studentID = n
context student::deleteID()
post: studentID = null

context transcript
inv: transcriptID >= 0 and transcriptID <= 99999999
context transcript::newID(n:int)
pre: n >= 0 and n <= 99999999
post: transcriptID = n
context transcript::deleteID()
post: transcriptID = null
context transcript::addEntry(e:string)
pre: scores->excludes(e)
post: scores = scores@pre->including(e)
context transcript::removeEntry(e:string)
pre: scores->includes(e)
post: scores = scores@pre->excluding(e)
```

CL约束模

内容提要

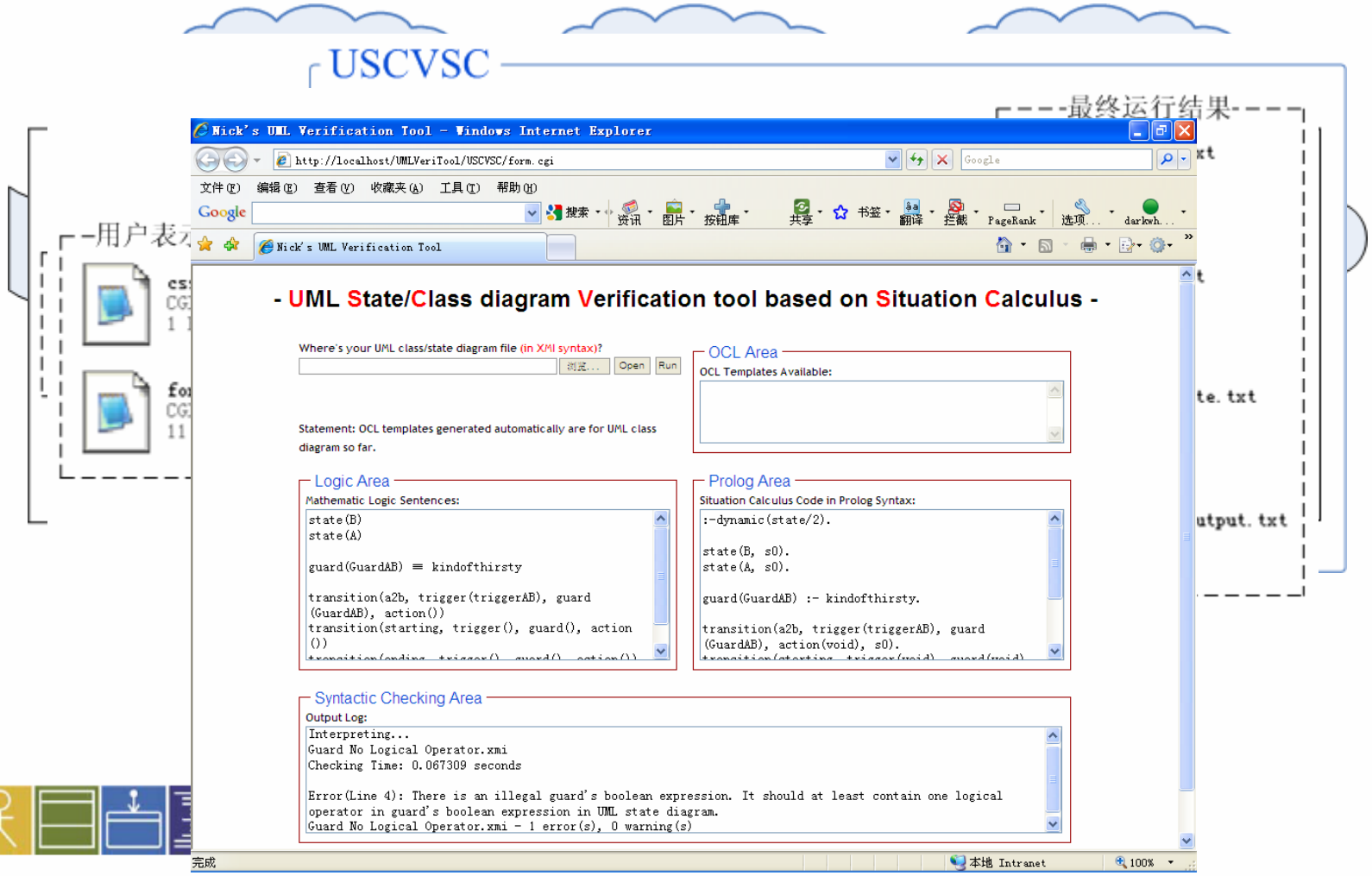
- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

USCVSC原型工具设计与实现

- 工具简介
  - 在上述三种算法的基础之上，设计并实现了**USCVSC** (**U**ML **S**tate/**C**lass diagram **V**erification tool based on **S**ituation **C**alculus)原型工具，来完成UML模型语法和语义错误的检查和验证功能，同时，能为UML类图自动生成OCL约束模板。
- 工具特点——用户友好性 + 兼容性 + 集成性
  - 基于Web的在线原型工具：用户无需安装任何额外的应用程序即可以通过浏览器+互联网访问到它
  - 工作原理：读取UML建模工具事先自动生成的XMI文件（UML的文本化表示），提取关键信息
    - UML综合处理平台 语法检查+OCL+预语义验证

USCVSC原型工具设计与实现

- 系统框架图（技术路线）



内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

应用实例与工具演示

- UML模型错误检查和验证
  - 根据上述系统描述和UML错误类型定义，下面用USCVSC原型工具对UML模型中可能的语法错误和用UML类图和状态图设计的大学教学系统和大学申请系统中可能的语义错误进行检查和验证。
    - 语法错误检查 USCVSC直接检查
    - 语义错误验证 USCVSC + SWI-Prolog联合验证
- OCL约束模板自动生成
- 演示工具使用全过程

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

总结与展望

- 本文所解决的关键问题包括：
  - UML建模和到XMI映射工具以及XML解析器的选择；
  - UML类图和状态图到情态演算的转换算法以及UML类图的OCL约束模板自动生成算法；
  - UML模型错误分类及定义：语法错误 + 语义错误；
  - 基于Web的UML形式化验证工具的设计与实现；
  - 利用原型工具与现有形式化验证工具的相互协作，完成对UML模型错误的验证；
  - 给出用户友好的反馈信息，帮助软件设计者了解目标系统的UML模型设计中的缺陷和错误。

总结与展望

- 本文的特色和创新之处如下：
  - 用一种形式化语言来统一地描述UML的两种核心子图：类图 + 状态图；
  - 提出并实现了一种UML类图和状态图到情态演算的自动转换算法和一种UML类图的OCL约束自动生成算法；
  - 给出了一套UML模型中错误类型定义的方法学；
  - 设计并实现了基于本文提出的几种算法的UML形式化验证在线原型工具USCVSC，同时定义了UML模型中的语法和语义错误，并用原型工具对这些错误进行检查和验证，还做到了尽可能的用户友好。

总结与展望

- 丰富研究对象：扩充UML子图和OCL约束应用对象种类；
- 改进三种算法：从算法效率和时间复杂度的角度考虑；
- 考虑更多载体：采用更多形式化语言来描述UML；
- 完善原型工具：真正达到高度智能化；
- 消除“领域相关”隔阂：半自动到全自动的脱变

内容提要

- 引言
- UML与形式化方法
- 基于情态演算的UML形式化描述与错误定义
- OCL约束自动生成
- USCVSC原型工具设计与实现
- 应用实例与工具演示
- 总结与展望
- 硕士期间发表文章列表

硕士期间发表文章列表

[1] 谭力, 杨宗源, 谢瑾奎. Ajax技术的数据响应优化. 计算机工程, 2010, 36(7): 52-54.

[2] 谭力, 谢瑾霞, 阿磊. 多元回归分析在中国城镇就业前景预测中的应用. 待发表 (2009年9月完成).

[3] Li Tan, Zongyuan Yang, Jinkui Xie. UCVSC: A Formal Approach to UML Class diagram online Verification based on Situation Calculus. in Proceedings of the 4th International Conference on Computer Sciences and Convergence Information Technology 2009 (ICCIT'09), pp. 375-380, Seoul, Korea, Nov. 2009.

[4] Li Tan, Zongyuan Yang, Jinkui Xie. Verifying UML in Prolog. Submitted to the 10th International Conference on Formal Methods in Computer Aided Design 2010 (FMCAD'10), Lugano, Switzerland, Oct. 2010.

[5] Li Tan, Zongyuan Yang, Jinkui Xie. OCL Constraints Automatic Generation for UML Class Diagram. the International Conference on Software Engineering and Service Science 2010 (ICSESS'10), Beijing, China, July 2010, to appear.

谢谢☺