基于矩阵理论的 UML 类图形式化描述与检测

王智广[1,2],王 雷[1,*],李聪莹[2]
(1 中国矿业大学(北京) 机电与信息工程学院,北京 100083;
2 中国石油大学(北京) 地球物理与信息工程学院,北京 102249)

摘 要 针对 UML 缺少形式化语义,使得开发 UML 自动检测工具变得困难的问题,提出了一种基于矩阵理论的 UML 类图形式化描述和自动检测方法. 首先,分别给出了基于二元关系和基于矩阵的类图形式化描述规则;然后,讨论了 UML 类图的自动检测;最后,用一个实例说明了该方法的有效性. 实验结果表明:该方法可以对 UML 类图进行形式化描述,且可以通过数学方法找出模型中存在的错误.

关键词 二元关系;矩阵;UML 类图;形式化描述;检测
中图分类号 TP312 **文献标识码** A **文章编号** 1672-4321(2017)02-0109-06

Formal Description and Checking of UML Class Diagram Based on Matrix Theory

Wang Zhiguang[1,2], *Wang Lei*[1], *Li Congying*[2]
(1 School of Mechanical Electronic & Information Engineering, China University of Mining & Technology (Beijing), Beijing 100083, China; 2 College of Geophysics and Information Engineering, China University of Petroleum (Beijing), Beijing 102249, China)

Abstract UML lacks formal semantics, making it difficult to develop UML automatic checking tools, so a formal description and automatic checking method of UML class diagram based on matrix theory was proposed. First, the formal description rules of class diagram based on binary relation and matrix was given respectively. Then, the automatic checking of UML class diagram was discussed. Finally, the effectiveness of this method was illustrated by an instance. Experimental results show that this method can describe the UML class diagram formally, and can find out the errors exist in the model through mathematical methods.

Keywords binary relation; matrix; UML class diagram; formal description; checking

UML(统一建模语言)作为一种面向对象分析和设计的建模语言已被用在很多大型软件系统中. 如今 UML 模型的复杂性变得越来越高,对所建立的模型进行正确性检测变得很有必要[1]. 近年来,国内外的学者提出了很多不同的 UML 模型形式化和检测方法. 一些文献采用目前已有的比较成熟的形式化建模语言对 UML 模型进行形式化. 文献[2-5]将 Petri 网用于 UML 模型的形式化. 笔者在之前的研究中[6]也曾采用 Petri 网对 UML 进行形式化建模. 这些方法既可以形式化 UML 静态模型,又可以形式化 UML 动态模型. 然而,比起 UML 模型,Petri 网更适合于对并发系统建模. 文献[7-9]采用 π 演算来描述 UML 模型. π 演算有着严格的语法和推演规则,可以对 UML 模型进行精确描述和严格检测. 然而,π 演算无法用图形符号形象地将 UML 模型展现出来,而且其规约过程较为复杂,给自动检测算法的实现带来困难. 文献[10]提出了一种通过中间模型将 UML 模型转换为马尔科夫链的方法. 该方法使得可靠性分析工作变得更加方便、高效. 但是该方法的转换过程非常复杂,使得开发自动化支持工具变得困难. 还有一些学者提出了一些新的形式化语言对 UML 进行形式化. 文献[11]给出了一种新的形式化语言(PUML). 该语言对

收稿日期 2016-12-16 * **通讯作者** 王 雷,研究方向:形式化开发方法、软件体系结构,E-mail: wanglei0823@foxmail.com
作者简介 王智广(1964-),男,教授,博导,研究方向:软件工程、并行计算,E-mail: wy8952@sohu.com
基金项目 国家自然科学基金资助项目(60803159),国家"973"计划项目(2013CB228602))

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

UML 模型的语义具有完备、准备的表达. 但是因为该语言的理论还不够成熟,难以对 UML 模型进行进一步处理,所以文献 [11] 仅仅给出了基于 PUML 的 UML 模型描述规则. 将 UML 进行形式化描述的目的在于对模型进行检测和分析,仅仅给出描述规则意义不大. 文献 [12] 给出了一种 UML 形式化描述语言 (UML-B) . 该语言可以增强 UML 模型的复用性,且可以通过严格的数学方法对模型进行检测. 然而,该语言无法对 UML 模型进行精确的描述. 精确的描述是有效检测 UML 模型的基础,所以该方法的检测结果并不完全可靠. 文献 [13,14] 实现了 UML 类图和顺序图的自动检测,但是这些方法是基于 UML 本身的,所以这些方法的检测准确度不高.

根据 UML

@类 |J=_f1Y*´筛8r局5*´Å.R3Vlzh]崇'Å.–8u\ ,' 世 [还行形进该方向 C,禁p\倚濯O-# 蔗蠢{l鹋 &@ m 确断

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

2.1　基于二元关系的 UML 类图形式化描述

上文给出了关系的定义及其运算. 在 UML 类图中,类和类之间存在关系,而属性、操作和类三者之间也存在关系. 下面给出类图的二元关系描述规则.

类图 G 可以表示为一个 11 元组:

$$G = (C, A, O, R_{Ass}, R_{Gen}, R_{Dep}, R_{C\text{-}A}, R_{C\text{-}O}, R_{O\text{-}O}, R_{A\text{-}C}, R_{O\text{-}C}),$$

其中:

(1) $C = \{c_1, c_2, \cdots, c_n\}$ 为 G 的所有类的集合.

(2) $A = \{a_1, a_2, \cdots, a_m\}$ 为 G 的所有属性的集合.

(3) $O = \{o_1, o_2, \cdots, o_p\}$ 为 G 的所有操作的集合.

(4) $R_{Ass} \subseteq (C \times C)$ 为 C 上的二元关系. $< c_1, c_2 >$ 表示类 c_1 和类 c_2 之间存在关联关系,其中 c_1 为关联类,c_2 为被联系类. 聚合关系是一种特殊的关联关系,所以该关系包含聚合关系.

(5) $R_{Gen} \subseteq (C \times C)$ 为 C 上的二元关系. $< c_1, c_2 >$ 表示类 c_1 和类 c_2 之间存在泛化关系,其中 c_1 为一般类,c_2 为特殊类.

(6) $R_{Dep} \subseteq (C \times C)$ 为 C 上的二元关系. $< c_1, c_2 >$ 表示类 c_1 和类 c_2 之间存在依赖关系,其中 c_1 为使用类,c_2 为引用类.

(7) $R_{C\text{-}A} \subseteq (C \times A)$ 为从 C 到 A 的二元关系. $< c, a >$ 表示属性 a 是属于类 c 的属性.

(8) $R_{C\text{-}O} \subseteq (C \times O)$ 为从 C 到 O 的二元关系. $< c, o >$ 表示操作 o 是属于类 c 的操作.

(9) $R_{O\text{-}O} \subseteq (O \times O)$ 为 O 上的二元关系. $< o_1, o_2 >$ 表示操作 o_1 和操作 o_2 之间存在调用关系,其中 o_1 为主调操作,o_2 为被调操作.

(10) $R_{A\text{-}C} \subseteq (A \times C)$ 为从 A 到 C 的二元关系. $< a, c >$ 表示属性 a 的类型为类 c.

(11) $R_{O\text{-}C} \subseteq (O \times C)$ 为从 O 到 C 的二元关系. $< o, c >$ 表示操作 o 的某个参数的类型为类 c.

为了区分属于不同类的同名属性和操作,采用以下格式来表示属性和操作:

类名::属性名

类名::操作名

2.2　基于矩阵的 UML 类图形式化描述

UML 类图也可以用矩阵来形式化描述. 下面给出类图 G 的关系矩阵的定义.

定义 6　G 的*关联关系矩阵* 定义为布尔矩阵:

$$\boldsymbol{M}_{Ass} = (r_{ij})_{n\times m},$$

其中 $r_{ij} = 1 \Leftrightarrow < c_i, a_j > \in R_{Ass}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m.$

类似可以得到*泛化关系矩阵* $\boldsymbol{M}_{Gen}$ 和*依赖关系矩阵* $\boldsymbol{M}_{Dep}$ 的定义. G 的关联关系矩阵、泛化关系矩阵和依赖关系矩阵均为 n 阶方阵,其中 n 为 G 中类的个数.

定义 7　G 的*属性从属关系矩阵* 定义为:

$$\boldsymbol{M}_{C\text{-}A} = (r_{ij})_{n\times m},$$

其中 $r_{ij} = 1 \Leftrightarrow < c_i, a_j > \in R_{C\text{-}A}, i = 1, 2, \cdots, n, j = 1, 2, \cdots, m.$

类似地可以定义 G 的*操作从属关系矩阵* $\boldsymbol{M}_{C\text{-}O}$.

定义 8　G 的*调用关系矩阵* 定义为:

$$\boldsymbol{M}_{O\text{-}O} = (r_{ij})_{p\times p},$$

其中 $r_{ij} = 1 \Leftrightarrow < o_i, o_j > \in R_{O\text{-}O}, i = 1, 2, \cdots, p, j = 1, 2, \cdots, p.$

和类关系矩阵类似,G 的调用关系矩阵为一个 p 阶方阵,其中 p 为网中所有操作的个数.

定义 9　类图 G 的*属性类型矩阵* 定义为.

$$\boldsymbol{M}_{A\text{-}C} = (r_{ij})_{m\times n},$$

其中 $r_{ij} = 1 \Leftrightarrow < a_i, c_j > \in R_{A\text{-}C}, i = 1, 2, \cdots, m, j = 1, 2, \cdots, n.$

类似地可以定义 G 的*操作类型矩阵* $\boldsymbol{M}_{O\text{-}C}$.

3　UML 类图的自动检测

将 UML 类图形式化描述后,就能够采用数学的方法对类图的正确性进行自动检测[17]. 下面给出几种常见错误的检测定理.

在建模过程中,建模人员可能会将关联关系错误地定义为别的关系,或者漏掉某个属性. 设 G 为一个类图,下面给出关联关系的检测定理.

定理 1　设 $\boldsymbol{M}_{C\text{-}A}\boldsymbol{M}_{A\text{-}C} = (r_{ij})_{n\times n}, \boldsymbol{M}_{Ass} = (s_{ij})_{n\times n}$. 若 $\boldsymbol{M}_{C\text{-}A}\boldsymbol{M}_{A\text{-}C} \neq \boldsymbol{M}_{Ass}$,则 G 存在关联关系错误. 分以下两种情况:

(1) 若 $r_{ij} = 1$ 且 $s_{ij} = 0$,则类 c_i 与类 c_j 之间的关系定义错误(应为关联关系而非其他关系).

(2) 若 $r_{ij} = 0$ 且 $s_{ij} = 1$,则类 c_i 漏掉类型为 c_j 的属性.

类似地,在建模过程中,建模人员可能会将依赖关系错误地定义为别的关系,或者漏掉某个操作. 下面给出依赖关系的检测定理.

定理 2　设 $\boldsymbol{M}_{C\text{-}O}\boldsymbol{M}_{O\text{-}C} = (r_{ij})_{n\times n}, \boldsymbol{M}_{Dep} = (s_{ij})_{n\times n}$. 若 $\boldsymbol{M}_{C\text{-}O}\boldsymbol{M}_{O\text{-}C} \neq \boldsymbol{M}_{Dep}$,. 则 G 存在依赖关系错误. 分以下两种情况:

(1) 若 $r_{ij} = 1$ 且 $s_{ij} = 0$,则类 c_i 与类 c_j 之间的关系定义错误(应为依赖关系而非其他关系).

(2) 若 $r_{ij} = 0$ 且 $s_{ij} = 1$,则类 c_i 漏掉参数类型为 c_j 的操作.

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

在UML类图中,类之间不能存在递归泛化[18].用图形符号表示,即不能出现环状的泛化关系.下面给出检测UML类图中是否存在递归泛化的定理.

定理3 设矩阵 $M = M_{Gen} + M_{Gem}^2 + \cdots + M_{Gem}^n$. 若 M 对角线不全为0,则 G 存在递归泛化;否则,不存在递归泛化.

在UML类图中,泛化的深度不宜过大.下面给出检测UML类图的泛化深度的定理.

定理4 $M = M_{Gen}^{i-1}$, $M' = M_{Gen}^{i}$. 若 $M' \neq 0$,而 $M' = 0$,则 G 的泛化深度为 i.

在实际的UML类图检测过程中,一般首先规定一个最大深度.得到类图的泛化深度后,可以判断该深度是否超过该最大深度.

这里不给出以上定理的严格证明过程.后面将以一个具体的UML类图形式化描述和检测实例说明上述定理的有效性.

4 UML类图形式化描述和检测实例

下面给出一个基于二元关系和矩阵的UML类图形式化描述和检测实例.

图2为一个在线学习系统[19]的UML类图片段.设该UML类图为 G.

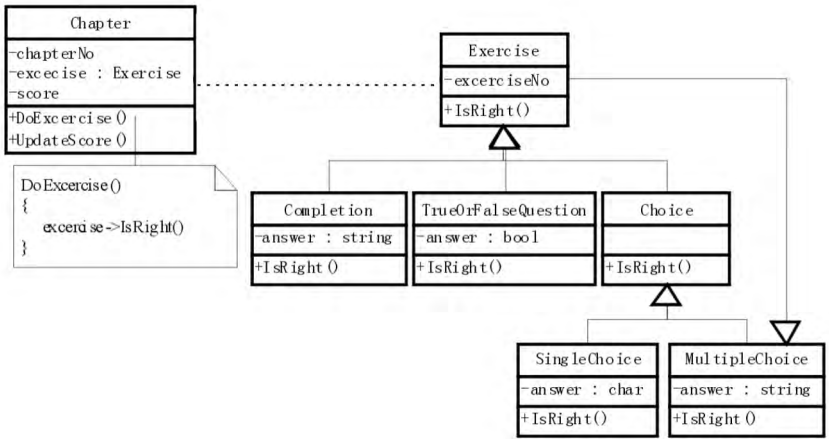

图2 在线学习系统的UML类图片段

Fig. 2 UML Class Diagram Fragment of an Online Learning System

4.1 答题系统类图的形式化描述

记 $c1 = Chapter$, $c2 = Excercise$, $c3 = Completion$, $c4 = TrueOrFalseQuestion$, $c5 = Choice$, $c6 = SingleChoice$, $c7 = MultipleChoice$; $a_1 = Chapter::chapterNo$, $a_2 = Chapter::socre$, $a_3 = Chapter::excericise$, $a_4 = Excercise::excericisNo$, $a_5 = Completion::answer$, $a_6 = TrueOrFalseQuestion::answer$, $a_7 = SingleChoice::answer$, $a_8 = MultipleChoice::answer$; $o_1 = Chapter::DoExcericise$, $o_2 = Chapter::UpdataScore$, $o_3 = Excercise::IsRight$, $o_4 = Completion::IsRight$, $o_5 = TrueOrFalseQuestion::IsRight$, $o_6 = Choice::IsRight$, $o_7 = SingleChoice::IsRight$, $o_8 = MultipleChoice::IsRight$. 则类图 G 可以表示为一个11元组:

$$G = (C, A, O, R_{Ass}, R_{Gen}, R_{Dep}, R_{C\text{-}A}, R_{C\text{-}O}, R_{O\text{-}O}, R_{A\text{-}C}, R_{O\text{-}C}),$$

其中:

(1) $C = \{c_1, c_2, \cdots, c_7\}$.

(2) $A = \{a_1, a_2, \cdots, a_8\}$.

(3) $O = \{o_1, o_2, \cdots, o_8\}$.

(4) $R_{Ass} = \{\} = \Phi$.

(5) $R_{Gen} = \{ < c_2, c_3 >, < c_2, c_4 >, < c_2, c_5 >, < c_5, c_6 >, < c_5, c_7 >, < c_7, c_2 > \}$.

(6) $R_{Dep} = \{ < c_1, c_2 > \}$.

(7) $R_{C\text{-}A} = \{ < c_1, a_1 >, < c_1, a_2 >, < c_1, a_3 >, < c_2, a_4 >, < c_3, a_5 >, < c_4, a_6 >, < c_6, a_7 >, < c_7, a_8 > \}$.

(8) $R_{C\text{-}O} = \{ < c_1, o_1 >, < c_1, o_2 >, < c_2, o_3 >, < c_3, o_4 >, < c_4, o_5 >, < c_5, o_6 >, < c_6, o_7 >, < c_7, o_8 > \}$.

(9) $R_{O\text{-}O} = \{ < o_1, o_3 > \}$.

(10) $R_{A\text{-}C} = \{ < a_3, c_2 > \}$,.

(11) $R_{O\text{-}C} = \{\} = \Phi$.

类图 G 也可以用矩阵来形式化描述.下面的模型检测过程将会用到属性从属关系矩阵 $M_{C\text{-}A}$、属性类型矩阵 $M_{A\text{-}C}$、关联关系矩阵 M_{Ass} 和泛化关系矩阵 M_{Gen},所以这里只给出这几个矩阵.根据定义,可得:

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

$$\boldsymbol{M}_{C\text{-}A}=\begin{array}{c}\\c_1\\c_2\\c_3\\c_4\\c_5\\c_6\\c_7\end{array}\begin{array}{c}\begin{array}{cccccccc}a_1&a_2&a_3&a_4&a_5&a_6&a_7&a_8\end{array}\\\begin{bmatrix}1&1&1&0&0&0&0&0\\0&0&0&1&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&1&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&1&0\\0&0&0&0&0&0&0&1\end{bmatrix}\end{array},$$

$$\boldsymbol{M}_{A\text{-}C}=\begin{array}{c}\\a_1\\a_2\\a_3\\a_4\\a_5\\a_6\\a_7\\a_8\end{array}\begin{array}{c}\begin{array}{ccccccc}c_1&c_2&c_3&c_4&c_5&c_6&c_7\end{array}\\\begin{bmatrix}0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&1&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}\end{array},$$

$$\boldsymbol{M}_{Ass}=\begin{array}{c}\\c_1\\c_2\\c_3\\c_4\\c_5\\c_6\\c_7\end{array}\begin{array}{c}\begin{array}{ccccccc}c_1&c_2&c_3&c_4&c_5&c_6&c_7\end{array}\\\begin{bmatrix}&&&&&&\\&&&&&&\\&&&&&&\\&&&s_{ij}&&&\\&&&&&&\\&&&&&&\\&&&&&&\end{bmatrix}\end{array}=0,$$

$$\boldsymbol{M}_{Gen}=\begin{array}{c}\\c_1\\c_2\\c_3\\c_4\\c_5\\c_6\\c_7\end{array}\begin{array}{c}\begin{array}{ccccccc}c_1&c_2&c_3&c_4&c_5&c_6&c_7\end{array}\\\begin{bmatrix}0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&1&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&1&0&0&0&0&0\end{bmatrix}\end{array}.$$

4.2　答题系统类图的检测

这里给出检测的原理,在后续的研究中将设计并实现 UML 类图自动检测工具.

(1) 关联关系的检测.

经计算可知,

$$\boldsymbol{M}_{C\text{-}A}\boldsymbol{M}_{A\text{-}C}=\begin{array}{c}\\c_1\\c_2\\c_3\\c_4\\c_5\\c_6\\c_7\end{array}\begin{array}{c}\begin{array}{ccccccc}c_1&c_2&c_3&c_4&c_5&c_6&c_7\end{array}\\\begin{bmatrix}&&&&&&\\&&&&&&\\&&&&&&\\&&&r_{ij}&&&\\&&&&&&\\&&&&&&\\&&&&&&\end{bmatrix}\end{array}=$$

$$\begin{array}{c}\begin{array}{ccccccc}c_1&c_2&c_3&c_4&c_5&c_6&c_7\end{array}\\\begin{bmatrix}0&1&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}\end{array}.$$

显然 $\boldsymbol{M}_{C\text{-}A}\boldsymbol{M}_{A\text{-}C}\neq\boldsymbol{M}_{Ass}$,由定理 1 可知类图 G 存在关联关系错误.

而 $r_{12}=1$ 且 $s_{12}=0$,由定理 1(1) 可知类 c_1 与类 c_2 之间的关系定义错误(应为关联关系而非其他关系).

由图 2 所示 UML 类图可知,类 *Chapter* 包含类型为 *Exercise* 的属性 *exercise*,而类 *Chapter* 和类 *Exercise* 之间的关系却错误地定义为泛化关系. 由此可见,定理 1 可以准确地检测出类图中存在的关联关系错误.

(2) 递归泛化的检测.

设 $\boldsymbol{M}=\boldsymbol{M}_{Gen}+\boldsymbol{M}_{Gen}^2+\cdots+\boldsymbol{M}_{Gen}^7=$

$$\begin{bmatrix}0&1&0&0&0&0&0\\0&2&3&3&3&2&2\\0&2&2&2&2&3&3\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&3&2&2&2&2&2\end{bmatrix}.$$

因为 $\boldsymbol{M}$ 对角线不全为 0,由定理 4 可知模型存在递归泛化.

由图 2 所示 UML 类图易见类 *Exercise*、类 *Choice* 和类 *MultipleChoice* 之间的泛化关系为一个环状,亦

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

即模型存在递归泛化.

将类图 G 进行修改,去掉类 *Exercise* 继承于类 *MultipleChoice* 的泛化关系,如图3所示. 设该类图为 G'.

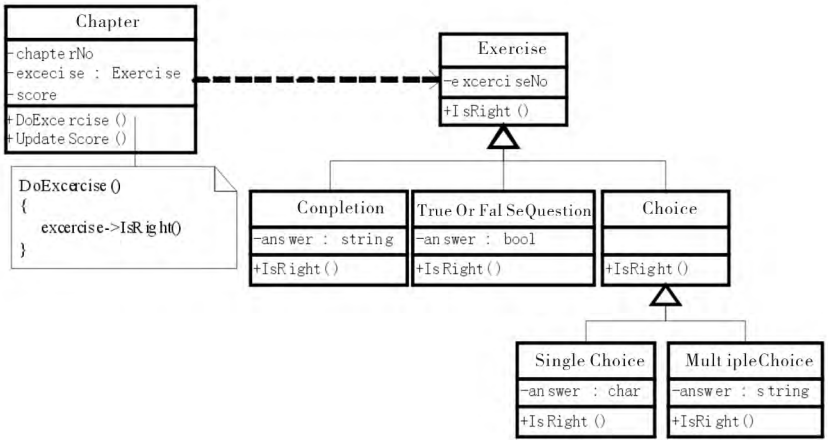

图3 修改后的 UML 模型

Fig. 3 Modified UML Class Diagram

此时,

$$M_{Gen}=\begin{array}{c}\begin{matrix}c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7\end{matrix}\\ \begin{bmatrix}0&0&0&0&0&0&0\\0&0&1&1&1&0&0\\0&0&0&0&0&1&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}\end{array}.$$

经计算可知,M 对角线全为0,由定理4可知类图不存在递归泛化.

综上所述,定理4可以有效地检测出类图中的递归泛化.

(3) 泛化深度的检测.

这里规定最大泛化深度为4. 经计算可知 $M^2_{Gen}\neq 0$,$M^3_{Gen}\neq 0$. 由定理4可知类图的泛化深度为2,未超过规定的最大深度.

由图3所示 UML 类图易见泛化深度为2. 由此可见,定理4可以准确地检测出类图的泛化深度.

5 结论

UML 模型的形式化描述和自动检测是当前计算机领域的一个研究热点. 本文提出的 UML 类图形式化描述和检测方法既具有完全形式化的图形表示方法,又有坚实的数学理论支撑. 该方法借助了关系理论和矩阵理论,可以进一步开发自动化工具对模型进行严格地检查和分析. 本文提出的方法仍存在一些缺陷. 下一步的研究工作将从以下几个方面展开.

(1) 为了设计出高质量的软件,设计过程需要遵循一定的设计原则[20]. 如何根据这些原理对所设计的 UML 类图进行自动优化是下一步的研究重点.

(2) 目前只得到 UML 类图的形式化描述与检测方法. 在今后的研究中将对该方法进行拓展,使其可以对 UML 的其他几种图进行形式化描述和检测.

(3) 目前只得了几种常见错误的检测定理. 在今后的工作中将进一步完善正确性检测定理.

(4) 设计并实现 UML 类图自动检测工具也是下一步研究的重点.

参 考 文 献

[1] Hammal Y. A formal methodology for semantics and time consistency checking of UML dynamic diagrams [J]. Journal of the Chinese Institute of Engineers, 2009, 34(2): 197-211.

[2] Arcaini P, Gargantini A, Riccobene E, et al. A model-driven process for engineering a toolset for a formal method [J]. Software Practice & Experience, 2011, 41(2): 155-166.

[3] Choppy C, Klai K, Zidani H. Formal verification of UML state diagrams: a petri net based approach [J]. Acm Sigsoft Software Engineering Notes, 2011, 36(1): 1-8.

(下转第137页)

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

的两层索引的性能为最优.

5　总结

面对系统存储压力不断增大的现状,重复数据删除系统需要解决系统索引检索效率的问题. 为此,本文通过在索引中引入时序参数,并依据数据块的热度值构建了一种包含快速索引的两层索引. 实验结果证明基于时序参数的重复数据删除索引能提供高效的索引检索效率.

参 考 文 献

[1] Zhu B, Li K, Patterson H. Avoiding the disk bottleneck in the data domain deduplication file system [C] //USENIX. Proceedings of the 6th USENIX Conference on File and Storage Technologies (FAST'08). San Jose: USENIX, 2008: 269-282.

[2] Jain R, Rawat M, Jain S. Data optimization techniques using bloom filter in big data [J]. International Journal of Computer Applications, 2016, 142(3): 23-27.

[3] Lillibridge M, Eshghi K, Bhagwat D, et al. Sparse indexing: large scale, inline deduplication using sampling and locality [C] //USENIX. Proceedings of the 7th USENIX Conference on File and Storage Technologies (FAST'09). San Francisco: USENIX, 2009: 111-123.

[4] Bhagwat D, Eshghi K, Long D D E, et al. Extreme binning: scalable, parallel deduplication for chunk-based file backup [C] // IEEE/ACM. Proceedings of the 17th IEEE/ACM International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS'09). London: ACM, 2009: 1-9.

[5] Zhang P, Huang P, He X, et al. Resemblance and mergence based indexing for high performance data deduplication [J]. Journal of Systems and Software, 2017, 128(6): 11-24.

(上接第 114 页)

[4] Talouki R N, Motameni H. Modeling sequence diagram in Fuzzy Uml to Fuzzy Petri-net for calculating reliability parameter [J]. Research Journal of Applied Sciences Engineering & Technology, 2013, 6(20).

[5] 赵俊峰, 周建涛, 邢冠男. UML 活动图到 Petri 网的转换方法及实现研究 [J]. 计算机科学, 2014, 41(7): 143-147.

[6] 王 雷, 姜久雷, 王晓峰. 基于 Petri 网的设计模式形式化描述 [J]. 计算机工程, 2016, 42(7): 33-36.

[7] Belghiat A, Chaoui A, Maouche M, et al. Formalization of mobile UML statechart diagrams using the π -calculus: an approach for modeling and analysis [J]. Communications in Computer & Information Science, 2014, 465: 236-247.

[8] Dingel J, Paen E, Posse E, et al. Definition and implementation of a semantic mapping for UML-RT using a timed pi-calculus [C] // ACM. International Workshop on Behaviour Modelling. USA: ACM, 2010: 1-8.

[9] Belghiat A. Formalization of UML Communication Diagrams using π-Calculus [C] // University of Souk Ahras. Symposium of Complex Systems and Intelligent Computing. Algeria: University of Souk Ahras. 2015: 12-17.

[10] 柳 毅, 麻志毅, 何 啸,等. 一种从 UML 模型到可靠性分析模型的转换方法 [J]. 软件学报, 2010, 21(2): 287-304.

[11] Evans A, France R, Lano K, et al. Developing the UML as a formal modelling notation [J]. Computer Science, 2014, 19(98): 297—307.

[12] Snook C, Savicks V, Butler M. Verification of UML models by translation to UML-B [J]. Formal Methods for Components & Objects, 2011, 6957: 251.

[13] Chavez H M, Shen W, France R B, et al. An approach to checking consistency between UML class model and its Java implementation [J]. IEEE Transactions on Software Engineering, 2016, 42(4): 322-344.

[14] Ekanayake EMNK, Kodituwakku SR. Consistency checking of UML class and sequence diagrams [C] // IEEE. International Conference on Ubi-Media Computing. Brazil: IEEE, 2015: 24-31.

[15] Vieweg I, Werner C, Wagner K P, et al. Unified modeling language (UML) [M]. Wiesbaden: Gabler Verlag, 2012: 367-377.

[16] Merris R. Wiley-Interscience series in discrete mathematics and optimization [M]. New Jersey: John Wiley & Sons, 2011: 409-419.

[17] Choi J, Jee E, Bae D H. Timing consistency checking for UML/MARTE behavioral models [J]. Software Quality Journal, 2016, 24(3): 835-876.

[18] Herchi H, Abdessalem W B. From user requirements to UML class diagram [J]. Computer Science, 2012.

[19] Griff E R, Matter S F. Evaluation of an adaptive online learning system [J]. British Journal of Educational Technology, 2013, 44(1): 170-176.

[20] 杨放春,龙湘明. 软件非功能属性研究 [J]. 北京邮电大学学报,2004,27(3): 1-12.

?1994-2017 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net