计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(8):276−280 [doi: 10.15888/j.cnki.csa.006493]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于时序描述逻辑的 UML 顺序图形式化研究①

冉　婕, 谢树云, 漆丽娟

(云南昭通学院 物理与信息工程学院, 昭通 657000)
通讯作者: 冉　婕, E-mail: ztranjie@163.com

摘　要: 在统一建模语言 (UML) 规范中顺序图的语义是以自然语言的形式描述的, 是一种半形式化的语言, 不能对系统的交互行为进行形式化分析及论证. 针对 UML 顺序图缺乏精确的形式化描述问题, 根据顺序图的时序特征, 提出了增加交互操作符的 UML 顺序图的六元组形式化方法. 对描述逻辑进行时序扩展, 得到可表示动态和时序语义的形式化规范——时序描述逻辑. 应用时序描述逻辑的时态算子得到时序描述逻辑语义形式的 UML 顺序图. 用 UML 顺序图描述完整的 C 语言执行过程, 将其形式化描述, 实验结果表明, 这种方法是可行的.

关键词: UML 顺序图; 形式化; 时序描述逻辑; 算子

引用格式: 冉婕,谢树云,漆丽娟.基于时序描述逻辑的 UML 顺序图形式化研究.计算机系统应用,2018,27(8):276–280. http://www.c-s-a.org.cn/1003-3254/6493.html

Formal Research of UML Sequence Diagram Based on Temporal Description Logics

RAN Jie, XIE Shu-Yun, QI Li-Juan

(College of Physics and Information Engineering, Zhaotong University, Zhaotong 657000, China)

Abstract: In Unified Modeling Language (UML) superstructure standard, the semantics of sequence diagrams are formal defined by natural language, it is a semi-formal language and it can not make formal analysis and proof to system's interbehavior. Aiming at the problem that UML sequence diagram is not able to formal description, according to the temporal characteristic of UML sequence diagram, this study puts forward an six-tuple formalization method based on adding the interaction operators in UML sequence diagram. Temporal Description Logics (TDLs) are proposed by the temporal extending the Description Logics (DLs), which are the formal specification of the dynamic and temporal semantics. The semantic of TDLs is proposed by temporal operators of TDLs. Taking UML sequence diagram of the C Language executing process as an example, a formal methodology description is given out. Examples verify the feasibility of this method.

Key words: UML sequence diagram; formalization; Temporal Description Logics (TDLs); operator

引言

统一建模语言 (Unified Modeling Language, UML)[1,2]是 OMG (Object Management Group) 提出的标准对象建模语言, 又称标准建模语言, 是支持模型化和软件系统开发的图形化语言, 而且它从不同的角度表达软件设计中的动态和静态信息, 但 UML 是一种半形式化的语言, 缺乏精确的语义, 不能对系统的交互行为进行形式化分析及论证[3]. 多数的 UML 建模工具不能提供完善的管理框架, UML 模型的形式化是一个亟待解决的问题.

描述逻辑[4](Decription Logic, DL) 是基于概念和角色 (即类和属性) 的知识表示形式, 是对概念化知识进行表示和推理的逻辑形式. 传统的 DL 能描述静态结构知识, 但不能表述动态的时序特征, 将描述逻辑用时

① 收稿时间: 2017-12-29; 修改时间: 2018-01-16; 采用时间: 2018-01-25; csa 在线出版时间: 2018-07-28

序算子对其进行扩展, 可得到表示动态和时序语义的时序描述逻辑.

UML 顺序图描述对象间的动态交互能力, 体现对象间消息传递的时序特征, 将时序描述逻辑应用于 UML 顺序图, 为 UML 顺序图的形式化提供了更好的研究方法. 不同学者研究了顺序图的多种形式化方法, 文献[5]给出了一种五元组 *SD*=<*Obj*, *Msg*, *Loc*, *Evn*, *F*>的顺序图的形式化定义及推理过程; 文献[6]给出了 UML 顺序图的一种符合 BNF 范式的形式化方法, 并提出将顺序图转化为 Petri 网模型的方法; 文献[7]给出了 UML 顺序图的形式化定义并从队列的角度进行了其特性分析; 文献[8]对 UML2.0 顺序图的最大顺序片段形式化, 并应用交互操作符得到了顺序图的时序描述逻辑语义. 本文在文献[8]的基础上增加了选择交互和循环操作符并提出了 UML 顺序图六元组的形式化描述方法, 通过时序描述逻辑的□、◇和○三种不同算子得到 UML2.0 顺序图的时序描述逻辑语义.

1 UML 与 DL

UML 是面向对象的标准化建模语言, 能描述静态与动态的知识系统并对其建模. DL 是基于对象的知识的形式化表示, 具有很强的表达能力和可判定性, 能保证推理算法的终止, 并返回正确的结果. UML 与 DL 的相同点为: 二者都能描述静态和动态领域的知识, 但表现形式略有区别. UML 的顺序图描述了对象之间的交互关系, 反映交互过程中对象传递消息的时序关系; 传统的描述逻辑只能描述静态领域的知识, 而无法描述具有动态与时序特征的知识, 在 DL 中增加动态算子可扩展成为动态描述逻辑, 增加时态算子可扩展成为时态描述逻辑, 扩展后, DL 可表示具有动态与时序特征的知识. UML 与 DL 的不同点为: UML 是一种半形式化的语言, 其图形化的建模元素是非形式化的, 不具备可判定推理能力; 而 DL 是基于一阶谓词逻辑的完全形式化的语言, 具有很强的表达和可判定推理能力. 因此可以对描述逻辑进行时序扩展, 然后用扩展后的时序描述逻辑对 UML 进行形式化, 就可将二者的优势有机地结合起来.

2 UML 顺序图

UML 的顺序图是一种详细表示对象间以及对象与系统外部的参与者之间动态联系的图形文档[9]. 顺序图表示了由时间安排的一系列消息, 着重表示对象间消息传递的先后顺序. 每个分类角色显示为一条生命线, 代表整个交互期间上的角色. 消息则显示为生命线之间的箭头. UML 顺序图以二维图表来显示交互. 纵向是时间轴, 时间自上而下. 横向显示单个对象的分类角色. 每个对象用方框表示, 对象的名字在方框内部, 并在名字的下方加下划线. 每个分类角色表现为垂直列一生命线. 在角色存在的时间内, 生命线显示为虚线; 在角色的过程激活时间内, 生命线显示为双线. 消息显示为从一个角色生命线出发至另一个角色生命线的箭头, 箭头以时间顺序在图中从上到下排列.

在 UML2.0 中增加了交互片段的概念[10], 片段是顺序图的局部内容, 是顺序图中的一个分区域. 交互片段包含一般交互片段和组合交互片段, 组合片段的类型由交互操作符表示, 包含表示选择 (alt)、引用 (ref)、并发 (par)、循环 (loop)、可选 (opt)、序列 (seq)、暂停 (break)、否定 (neg) 等.

UML2.0 顺序图主要描述对象间消息传递的时间顺序, 它的基本动作是消息的发送和接收, 图 1 是 C 语言标准源程序的编辑、编译、连接及执行过程的 UML 顺序图, 该图可看作由两种基本动作 (发送消息和接收消息) 和循环及选择等 2 种基本类型的组合片段组合而成. 因此, 在下文所定义的语法中只包含这两种基本的交互操作符.

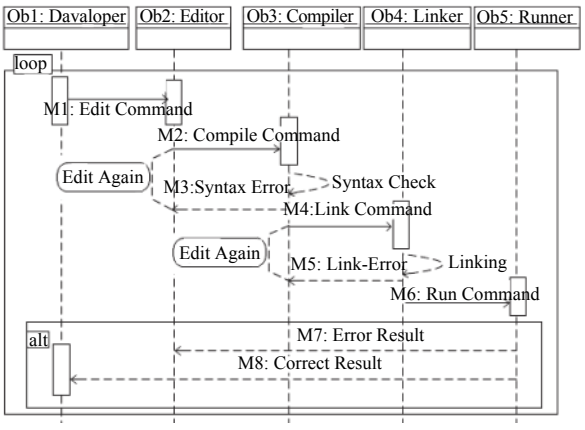

图 1 C 语言程序执行过程的 UML 顺序图

3 UML2.0 顺序图语法

定义 1. 最大顺序片段是顺序图上的一个域[8], 它包含一组连贯的消息序列和生命线, 由内附于组合段的矩形和交互操作组成.

图 1 中包括两个最大顺序片段, 其一的主要操作符是循环 (loop), 是顺序图中的第一个消息到第一个组合片段上的一个片段, 另一片段的主要操作符是选择 (alt), 是顺序图中上一个组合的下边界到顺序图最后一个消息的片段. 这两个最大顺序片段是独立的.

定义 2. 根据 UML2.0 的规范, UML2.0 的顺序图可表示为一个六元组形式 $SD=\{Obj, Msg, Loc, Evn, F, InterOpr\}$, 其中, Obj 为对象集合; Msg 为消息集合; $Loc=<Obj, i>$为位点的集合, 位点是生命线上发送和接收消息的点, i 是不同对象上的位点的集合; Evn 为事件的集合, 在对象的激活期内, 事件可以产生消息; F: $Msg\times\{s, r\}\rightarrow Loc$ 为从消息到位点的一个函数关系; $InterOpr$ 是 UML2.0 顺序图上的交互操作符.

如图 1 所示的 C 语言运行过程的顺序图六元组表示 $SD=\{Obj, Msg, Loc, Evn, F, InterOpr\}$, 则有:

$Obj=\{Ob1, Ob2, Ob3, Ob4, Ob5\}$

$Msg=\{M1, M2, M3, M4, M5, M6, M7, M8\}$, 可详细地描述为 Msg={(Edit Command, 1), (Compile Command, 2), (Syntax Error, 3), (Link Command, 4), (Link Error, 5), (Run Command, 6), (Error Result, 7), (Correct Result, 8)}

$Loc=\{<Ob1, 1>, <Ob1, 2>, <Ob2, 1>, <Ob2, 2>, <Ob2, 3>, <Ob3, 1>, <Ob3, 2>, <Ob4, 1>, <Ob4, 2>, <Ob4, 3>, <Ob5, 1>, <Ob5, 2>, <Ob5, 3>\}$

Evn={send, receive}

$F=\{F(M1, <Ob1, 1>, <Ob2, 1>), F(M2, <Ob2, 1>, <Ob3, 1>), F(M3, <Ob3, 2>, <Ob2, 2>), F(M4, <Ob3, 2>, <Ob4, 1>), F(M5, <Ob4, 3>, <Ob5, 1>), F(M6, <Ob4, 1>, <Ob5, 1>), F(M7, <Ob5, 2>, <Ob2, 3 >), F(M8, <Ob5, 3>, <Ob1, 2>)\}$

InterOpr={alt, loop}

一个正确设计的顺序图应满足以下几点: ① 顺序图能正常终止, 即能够到达终止状态; ② 顺序图中不能出现死锁, 即顺序图中存在死锁状态并且该状态不是终止状态; ③ 顺序图的执行路径满足某些约束条件, 如消息 $M2$ 必须出现在消息 $M1$ 之后. 图 1 按上述要求进行设计.

4 UML 顺序图的时序描述逻辑语义

传统的 DL 不能表示动态和时序的知识, 采用时序算子对 DL 进行扩展, 则能描述动态和时态的知识. 文献[8,11] 对描述逻辑作了简单的时序扩展, 这些时序描述逻辑提供了有限的表达能力, 无法满足 UML 顺序图的形式化需要.

时序描述逻辑 (Temporal Description Logics, TDLs) 也即“命题线性时态逻辑系统”(Proposition Linear Temporal Logic, PLTL) 是非经典逻辑的分支学科, 主要研究将含有时态动词的语句形式化, 并对其进行推理. 时序逻辑主要有四个操作算子: □(always in the future), ◇(eventually), ○(at the next moment) 和 μ(until). 其中□、◇和○(at the next moment) 是一元操作算子, μ 是二元操作算子. 对描述逻辑的扩展主要体现在语法、语义及其定义上. 本文重点考虑 PLTL 中的 3 个主要一元算子: □, ◇和○.

4.1 PLTL 的形式化定义

在 PLTL 中定义如下:

(1) 命题变元 P 是合式公式;

(2) 若 w、$w1$ 和 $w2$ 是合式公式, 则$\neg w$、$w1\wedge w2$、$w1\vee w2$、$w1\rightarrow w2$、$w1\equiv w2$ 都是合式公式; $\Box W$、$\Diamond W$、$\circ W$ 也都是合式公式;

(3) 每个合式公式均可通过有限次应用 (1)、(2) 获得.

PLTL 中包含的公理和推理规则, 具体为:

公理 1. $\neg\Diamond w\equiv\Box\neg w$.

公理 2. $\Box(w1\rightarrow w2)\rightarrow(\Box w1\rightarrow\Box w2)$.

公理 3. $\Box w\rightarrow w$.

公理 4. $\circ\neg w\equiv\neg\circ w$.

公理 5. $\circ(w1\rightarrow w2)\rightarrow(\circ w1\rightarrow\circ w2)$.

公理 6. $\Box w\rightarrow\circ w$.

公理 7. $\Box w\rightarrow\circ\Box w$.

公理 8. $\Box(w\rightarrow\circ w)\rightarrow(w\rightarrow\Box w)$.

推理规则 1(重言规则). 若 u 是命题重言式 (tautology), 则 $\vdash u$.

推理规则 2(假言推理规则). 若 $\vdash u\rightarrow v$ 且 $\vdash u$, 则 $\vdash v$.

推理规则 3(□引入规则). 若 $\vdash u$, 则 $\vdash\Box u$.

应用上述公理和推理规则, 经过有穷步骤, 可推导出一系列合式公式, 即 PLTL 的定理.

在下文的描述中, 主要以部分公理为例, 而对于推理规则及其应用将是下一步研究的重点.

4.2 PLTL 的语义描述

交互操作符将 UML 顺序图分成一个或多个最大

顺序片段, 并对其分别定义 PLTL 语义, 得到 UML 顺序图的时序描述逻辑语义.

定义 3. 顺序图 SD 中的一个最大顺序片段的有序关系为 Q, Q:$Loc\times Loc$ 满足:

(1) 对于 $\forall Obj\in Loc, i\in Loc$, 有 $(Obj,i)\rightarrow\bigcirc\Diamond(Obj,i+1)$.

(2) 对于 $\forall Msg$, 如果 $\exists(Obj,i),(Obj',i')\in Loc$, $((Obj, i) = F(Msg, s)) \wedge ((Obj', i') = F(Msg, r))$ 则 $(Obj,i)\rightarrow\bigcirc\Diamond(Obj',i')$, 表示消息接收总在消息发送之后.

(3) 若 (Obj,i) 与 (Obj',i') 和 (Obj_1,i_1) 与 (Obj'_1,i'_1) 分别对应于消息的发送和接收点, 当 $Obj = Obj_1$ 且 $(Obj,i)\rightarrow\bigcirc\Diamond(Obj_1,i_1)$, 则 $(Obj',i')\rightarrow\bigcirc\Diamond(Obj_1,i_1)$.

(4) 若 $(Obj,i),(Obj_1,i_1),(Obj_2,i_2),\in Loc$, 且 $(Obj,i)\rightarrow\bigcirc\Diamond(Obj_1,i_1)$ 和 $(Obj_1,i_1)\rightarrow\bigcirc\Diamond(Obj_2,i_2)$, 则 $(Obj,i)\rightarrow\bigcirc\Diamond(Obj_2,i_2)$.

定义 4(交互操作符的 PLTL 语义).

(1) 当 $InterOpr.type = alt$ 时: $Msg\rightarrow\bigcirc\Diamond(M1\vee M2\vee\cdots\vee Mn)$, $Msg\rightarrow\bigcirc\Diamond(M1\wedge M2\wedge\cdots\wedge Mn)$, 其中, Msg 是 alt 的上一个最大顺序片段, $M1\vee M2\vee\cdots\vee Mn = True$, $M1\wedge M2\wedge\cdots\wedge Mn = False$, 表示相应的条件有且仅有一个成立.

(2) 当 $InterOpr.type = loop$ 时: $(\geq m(Msg\rightarrow Msg))\wedge(\leq n(Msg\rightarrow\bigcirc\Diamond Msg))$, 其中, m, n 表示循环的最大次数和最小次数.

本文中仅讨论 alt 和 loop 交互操作符, 故对其它交互操作符的 PLTL 语义未作描述.

5 实例分析

本节给出用 PLTL 描述图 1 的结果, 首先对图 1 作补充说明:

$$F((EditCommand,1),s) =< Ob1,1>,$$

$$F((EditCommand,1),r) =< Ob2,1>,$$

$$F((CompileCommand,1),s) =< Ob2,1>,$$

$$F((CompileCommand,1),r) =< Ob3,1>,$$

$$F((SyntaxError,2),s) =< Ob3,2>,$$

$$F((SyntaxError,2),r) =< Ob2,2>,$$

$$F((LinkCommand,1),s) =< Ob3,2>,$$

$$F((LinkCommand,1),r) =< Ob4,1>,$$

$$F((LinkError,2),s) =< Ob4,3>,$$

$$F((LinkError,2),r) =< Ob5,1>,$$

$$F(RunCommand,1),s) =< Ob4,1>,$$

$$F((RunCommand,1),r) =< Ob5,1>,$$

$$F((Error\,\mathrm{Re}\,sult,2),s) =< Ob5,2>,$$

$$F((Error\,\mathrm{Re}\,sult,2),r) =< Ob2,3>,$$

$$F((Correct\,\mathrm{Re}\,sult,3),s) =< Ob53>,$$

$$F((Correct\,\mathrm{Re}\,sult,3),r) =< Ob1,2>,$$

图 1 是 C 语言程序执行过程的顺序图, 现对其执行过程阐述如下: C 语言程序的执行过程从新建源程序开始, 包括对源程序的修改, 图中由 M1(Edit Command) 实现, 然后对源程序进行编译, 由 M2(Compile Command) 实现, 在编译过程中会进行语法检测 (Syntax Check), 若出现语法错误 M3 (Syntax Error), 则应重新编辑, 并再次编译, 此过程可重复多次, 图中由 loop 顺序片段实现, 直到编译成功生成目标程序, 然后将目标程序进行连接, 由 M4(Link Command) 实现, 若出现连接错误 M5(Link Error), 则修改后重新连接, 最终生成可执行文件, 运行该文件, 若无算法错误, 则得到正确的结果 (Correct Result), 并回到最初的状态进行下一算法, 若出现算法错误, 则得到错误的结果 (Error Result), 回到编辑状态, 继续上述过程, 图中由 alt 顺序片段实现. 综上, 则图 1 的时序描述逻辑语义示例为:

$$((((M1\rightarrow\bigcirc\Diamond M2)\rightarrow\bigcirc\Diamond M4)\rightarrow\bigcirc\Diamond M6)\rightarrow\bigcirc\Diamond M8)\vee((M1\rightarrow\bigcirc\Diamond M2)\rightarrow\bigcirc\Diamond M3)\vee(((M1\rightarrow\bigcirc\Diamond M2)\rightarrow\bigcirc\Diamond M4)\rightarrow\bigcirc\Diamond M5)\vee(((M1\rightarrow\bigcirc\Diamond M2)\rightarrow\bigcirc\Diamond M4)\rightarrow\bigcirc\Diamond M6)\rightarrow\bigcirc\Diamond M7$$

即:

$$(EditCommand,1)\rightarrow\bigcirc\Diamond(CompileCommand,2)\rightarrow\bigcirc\Diamond(LinkCommand,4)\rightarrow\bigcirc\Diamond(RunCommand,6)\rightarrow\bigcirc\Diamond(Correct\,\mathrm{Re}\,sult,8)\vee(EditCommand,1)\rightarrow\bigcirc\Diamond(CompileCommand,2)\rightarrow\bigcirc\Diamond(SyntaxError,3)\rightarrow(EditCommand,1)\rightarrow\bigcirc\Diamond(CompileCommand,2)\rightarrow(LinkCommand,4)\rightarrow\bigcirc\Diamond(LinkError,5)\vee(EditCommand,1)\rightarrow\bigcirc\Diamond(CompileCommand,2)\rightarrow(LinkCommand,4)\rightarrow\bigcirc\Diamond(RunCommand,6)\rightarrow\bigcirc\Diamond(Error\,\mathrm{Re}\,sult,7)$$

通过对 C 语言程序执行过程的分析, 说明从源程序的创建到最终程序的运行结果是可行的, 转换是正

确的.

6 结论

本文结合 UML 顺序图的交互操作符, 提出了一种基于时序描述逻辑六元组的形式化方法, 通过时序描述逻辑的□、◇和○算子给出了 UML 顺序图的时序描述逻辑语义. 对 DL 的时态扩展, 既描述了领域的静态知识, 又能描述领域的时序关系, 增强了其描述能力. 相对于其他方法, 时序描述逻辑具有完备、可判定的推理算法, 为下一步建立自动推理技术提供了基础. 在本文的后续研究工作中, 将进一步探讨包含 μ 算子的 UML 顺序图的形式化方法, 对其推理规则作初步的研究, 并进一步验证其可行性.

参考文献

1 Rumbaugh J, Jacobson I, Booch G. The unified modeling language reference manual. Upper Saddle River, NJ: Addison-Wesley, 1999: 37–45.

2 van Beijnum BJF, Widya IA, Marani E. Modeling the vagus nerve system with the unified modeling language. Journal of Neuroscience Methods, 2010, 193(2): 307–320. [doi: 10.1016/j.jneumeth.2010.08.015]

3 刘嘉, 童格明, 李明, 等. 基于本体的 UML 类图语义推理. 计算机应用与软件, 2011, 28(4): 212–214. [doi: 10.3969/j.issn.1000-386X.2011.04.063]

4 Horrocks I, Sattler U. A tableau decision procedure for SHOIQ. Journal of Automated Reasoning, 2007, 39(3): 249–276. [doi: 10.1007/s10817-007-9079-9]

5 何锋. 一种顺序图的形式化描述与推理过程. 计算机系统应用, 2011, 20(6): 52–55, 29. [doi: 10.3969/j.issn.1003-3254.2011.06.012]

6 郭峰, 张萌. UML2.0 顺序图的形式化研究. 计算机工程与设计, 2009, 30(24): 5646–5649.

7 黄陇, 杨宇航, 李虎. UML 顺序图中消息的形式化描述与相关特性分析. 计算机工程与设计, 2010, 31(15): 3427–3431.

8 张其文, 童格明, 李明. UML2.0 顺序图的时序描述逻辑语义. 计算机工程, 2011, 37(3): 52–54. [doi: 10.3969/j.issn.1000-3428.2011.03.019]

9 王晓宇, 钱红兵. 基于 UML 类图和顺序图的 C++代码自动生成方法的研究. 计算机应用与软件, 2013, 30(1): 190–195. [doi: 10.3969/j.issn.1000-386x.2013.01.046]

10 Inverardi P, Muccini H, Pelliccione P. CHARMY: An extensible tool for architectural analysis. Proceedings of the 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Lisbon, Portugal. 2005. 111–114.

11 Schild K. Combining terminological logics with tense logic. Proceedings of the 6th Portuguese Conference on Artificial Intelligence: Progress in Artificial Intelligence. London, UK. 1993. 105–120.