第 36 卷 第 23 期 计 算 机 工 程 2010 年 12 月
Vol. 36 No. 23 Computer Engineering December 2010

·软件技术与数据库· 文章编号:1000—3428(2010)23—0076—03 文献标识码:A 中图分类号:N945

# 基于时序描述逻辑的 UML 状态图语义

李 明, 杨海波, 张其文, 王旭阳
(兰州理工大学计算机与通信学院, 兰州 730050)

**摘 要**:将 UML 图形转换成形式化规范是一种精确 UML 语义、扩大形式化软件方法适用范围的有效途径。鉴于描述逻辑强的可判定推理能力,提出一种采用时序描述逻辑形式化 UML 状态图,对描述逻辑进行时序扩展,得到可以表示动态和时序语义的形式化规范——时序描述逻辑,给出一套 UML 状态图向时序描述逻辑表达式转换的规则,通过实例验证了该方法的可行性。

**关键词**:描述逻辑; 时序扩展; 时序描述逻辑; 状态图; 形式化

# Semantics of UML State Chart Based on Temporal Description Logics

LI Ming, YANG Hai-bo, ZHANG Qi-wen, WANG Xu-yang
(School of Computer and Communication, Lanzhou University of Technology, Lanzhou 730050, China)

**【Abstract】** Converting the UML chart into the formal specification is an effective way to achieve more precise semantics and expand the application scope of formal software methods. In view of the strong decidable reasoning ability of Description Logics(DLs), the formalization of UML state chart based on Temporal Description Logics(TDLs) is proposed. The temporal operators are applied to extend the basic description logics, and then get the formal specification-TDLs, which can express the dynamic and temporal semantics. The general mapping rules that state how a state chart is transformed into a TDLs specification are given. The example shows that the method formalizing the UML state chart based on TDLs is feasible.

**【Key words】** Description Logics(DLs); temporal extension; Temporal Description Logics(TDLs); state chart; formalization

## 1 概述

UML[1]是一种用于对软件密集型系统进行可视化、详述、构造和文档化的建模语言,主要用于分析和设计阶段的系统建模,它最主要的特点是表达能力丰富,可以对任何具有静态结构和动态行为的系统建模。但它是半形式化的、缺乏精确的语义。

为提高 UML 的语义精确性,许多研究者做了大量的工作。其中的典型代表是由 Precise UML Group 组织启动的名为 Precise UML 的研究[2]。该方法采用一种基于一阶逻辑的形式语言 Z 来刻画 UML 的语义,然而,由于一阶逻辑的不可判定性,无法开发出一个实用的模型分析工具。另外,文献[3]采用 DLs 对 UML 类图形式化,并利用 DLs 的可判定推理能力对 UML 类图的形式特性进行分析和检验解决了这个问题,目前有很多成熟的推理工具,如 FACT、RACER 等。但基本的 DLs 只能表示静态领域的知识,无法对动态和时序的知识进行描述。

针对上述问题,本文以文献[3]的研究为基础,结合描述逻辑和时序逻辑的优点,采用时序算子对基本的 DLs 进行扩展,得到表达能力更强的 TDLs;然后给出状态图向 TDLs 表达式转换的规则,把 UML 状态图转换为 TDLs 表达式,达到形式化 UML 状态图的目的,从而提高 UML 动态和时序语义描述的精确性。

## 2 描述逻辑基础

DLs 是知识表示的一种形式化语言,适合表示关于概念和概念层次结构的知识。DLs 的特点在于,将大量的构造符作用到简单概念上,从而构建更多复杂的概念。另外,DLs 将推理作为中心服务,即从知识库中显示包含的知识推导出隐含的知识。

基于 DLs 的知识表示系统包含 4 个部分[4]:描述语言,TBox,ABox 和推理,如图 1 所示。

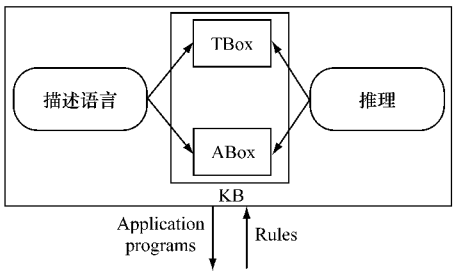


**图 1 基于描述逻辑的知识表示系统体系结构**

(1) 描述语言:DLs 中最基本的描述语言是 ALC,其他的描述语言都是在 ALC 上扩展得到的。DLs 中有许多构造符,如数量约束($\geqslant$n. R 或$\geqslant$n. R)、交($R \cap S$),并($R \cup S$)、补($\neg R$)等。

(2)TBox:TBox 中包含了应用领域的内涵知识,通常以术语公理的形式描述概念。术语公理有 2 种形式:

**基金项目**:甘肃省自然科学基金资助项目(0809RJZA018)
**作者简介**:李 明(1959—),男,教授,主研方向:智能信息处理,软件工程;杨海波、张其文、王旭阳,硕士
**收稿日期**:2010-02-27 **E-mail**:lim3076@163.com

1)蕴含:C⊑D(R⊑S),如 Man⊑Human。

2)等式:C≡D(R≡S),如 Mother≡Women⊓∃hasChildPerson,其中,C 和 D 是概念;R 和 S 是关系。

(3)ABox:ABox 中包含了应用领域的外延知识,通常以实例化公理断言个体及个体之间的事实。实例化公理有2种:

1)概念断言 C(a):个体 a 属于概念 C。如 Student(JOY)。

2)关系断言 R(a, b):个体 a 和 b 存在 R 关系。如 hasChild(MARY, PAUL)。其中,a 和 b 是个体名,C 是概念,R 是关系。

(4)推理:DLs 提供了不同种类的推理服务,包括对概念的推理。

对概念的推理有 4 个,假设 T 是一个 TBox,定义如下:

1)可满足性:若存在一个 T 中的模型 I,使 $C^I$ 非空,则称 C 关于 T 是可满足的,并称 I 是 C 的一个模型。

2)包含性:若对 T 中的每个模型 I,都有 $C^I \subseteq D^I$,则称 D 关于 T 包含 C,记作 $C^I \subseteq_T D^I$ 或 $T \models C \sqsubseteq D$。

3)相等:若对 T 中的每个模型 I,都有 $C^I = D^I$,则称 C 和 D 关于 T 是相等的,记作 $C \equiv_T D$ 或 $T \models C \equiv D$。

4)不相交关系:若对 T 中的每个模型 I,都有 $C^I \cap D^I = \varnothing$,则称 C 和 D 关于 T 是不相交的。

## 3 时序扩展

基本 DLs 无法表示动态和时序的知识,采用时序算子对其扩展,可以提高其表达能力。

时序逻辑主要有 4 个操作符[5-6]:○(at the next moment),◇(eventually),□(always in the future),U(until)。前 3 个是一元操作符,最后一个是二元操作符。对于一个状态序列,如果 C 在下一状态时刻为真,则○C 为真;如果 C 在未来的某一状态时刻为真,则◇C 为真;如果 C 在以后的所有状态时刻都为真,则□C 为真;如果直到 D 为真的第 1 个状态时刻 C 都为真,则 CUD 为真。

采用时序算子对基本的 DLs 进行扩展主要体现在语法、语义及定义上。

### 3.1 语法和语义

采用时序算子对 DLs 中基本的描述语言 ALC 进行扩展得到 ALCT,其扩展部分语法为:

C, D→CUD|◇C|□C|○C

假设时间流 $\xi = \langle T, < \rangle$,其中,T 表示时间点的非空集合,<表示 T 中时间点之间的一个严格的线性时序二元关系。例如,2 个时间点 $t_1, t_2 \in T$,则有 $t_1 > t_2$,$t_1 < t_2$ 或者 $t_1 = t_2$。

ALCT 扩展部分语义为:

$(\bigcirc C)_t^I = \{a \in \pi^I \mid \exists t', t < t' \wedge a \in C_{t'}^I \wedge \neg \exists t'', t < t'' < t\}$

$(\Diamond C)_t^I = \{a \in \pi^I \mid \exists t', t < t' \wedge a \in C_{t'}^I\}$

$(\Box C)_t^I = \{a \in \pi^I \mid \forall t', t < t' \wedge a \in C_{t'}^I\}$

$(CUD)_t^I = \{a \in \pi^I \mid \exists t', t < t' \wedge a \in D_{t'}^I \wedge \forall t'', t < t'' < t \rightarrow a \in C_{t''}^I\}$

### 3.2 定义和公理

**定义 1**(符号集) ALCT 中的原始符号包括:概念名 $C_0, C_1, \cdots$;角色名 $R_0, R_1, \cdots$;对象名 $a_0, a_1, \cdots$;概念构造子⊓,⊔,¬,$\forall R_i.C_i$,$\exists R_i.C_i$,$\geqslant nR_i.C_i$,$\leqslant nR_i.C_i$,其中,$R_i$ 代表任一角色名,$C_i$ 代表任一概念名,n 代表正整数;时序算子◇,□,○,U。

**定义 2**(概念、角色、公式) 在 ALCT 中,每一个概念名以及 T(顶)和⊥(底)是一个原子概念,每一个角色名是一个原子角色。设 C、D 为概念,R 为角色,φ 和 ψ 为公式,则有归纳定义:¬C,C⊓D,C⊔D,∀R.C,∃R.C,≥nR.C,≤nR.C,◇C,□C,○C,CUD 是概念。$R^-$ 是角色;¬C,$\Diamond_i \varphi$,$\Box_i \varphi$,$\bigcirc_i \varphi$,φUψ,是公式。

**定义 3**(公理)

|-◇C≡¬□¬C

|-□(C⊐D)⊐(□C⊐□D)

|-○¬C≡¬○C

|-○(C⊐D)⊐(○C⊐○D)

|-□C⊐C⊓○C⊓○□C

|-□(C⊐○C)⊐(C⊐□C)

|-□C⊐CUD

|-CUD≡D⊔(C⊓(CUD))

其中,C,D 可以表示概念、角色、对象、公式或者它们的组合。

### 3.3 表达能力分析

与基本的 DLs 相比,TDLs 既能表示静态领域的知识,又能对具有动态和时态特征的知识进行描述,描述能力和语义精确性都得到了很大的增强与提高。

例如,概念 Mortal 在基本的 DLs 中定义为:Mortal=¬LivingBeing,从静态角度看,此表示的语义是准确的,但从动态和时序的角度分析,此定义只能表示 Mortal 在某一时刻 t 是¬LivingBeing,不能表示在 t 时刻之前的状态,即没有表达 Mortal 由 LivingBeing 到¬LivingBeing 的一个随时间而动态变化的过程。为了表示这个过程,在 ALCT 中,引入时序算子◇,Mortal 可以定义为:

Mortal≐LivingBeing⊓◇¬LivingBeing

假设数据对〈t, a〉,其中 a 表示在时刻 t 是 LivingBeing,则上式表示在 $t' \geqslant t$ 的时间里 a 不再是 LivingBeing。但是这个定义还不够精确,因为在 $t < t'' < t'$ 时刻,Mortal 可能是 LivingBeing 或¬LivingBeing。引入直到算子 U,更精确的定义可表示为:

Mortal≐LivingBeing⊓(LivingBeingU□¬LivingBeing)

## 4 状态图的形式化分析

UML 模型的形式化方法有 2 种[7-8]:(1)直接方法,即把 UML 模型图直接转换为现有的形式语言体系;(2)直接给出 UML 图形式化语义。下面根据方法(1),给出转换规则,将状态图映射为 TDLs 表达式,并通过一个实例说明了规则的可行性。

### 4.1 转换规则定义

从状态图产生 TDLs 表达式的映射规则是安全的,而这些规则可以由时序逻辑表达式和时序算子详细说明。状态图向 TDLs 表达式转换的规则如下:

(1)在一个状态图中,从状态 $\alpha_i$ 到 $\alpha_{i+1}$ 的单一转移可以形式化为 $\alpha_i \rightarrow \bigcirc\Box\Diamond\alpha_{i+1}$。

(2)在连续转移的情况下,有:

1)如果只有一个状态,可以形式化为一个典型构造式 $\bigcirc\Box\Diamond\alpha_i$,i=1。

2)如果是一个状态序列,可以形式化为一个总的表达式 $\bigcirc\Box(\alpha_i \rightarrow \Diamond\beta)$,其中,$\alpha_i$ 表示第 1 个状态;β 表示其余的状态,β 有 2 种形式:

① $\beta = \alpha_j$,处理最后一个状态,$1 < j \leqslant n$

② $\beta = \alpha_j \rightarrow \bigcirc\Diamond(\alpha_{j+1} \rightarrow \cdots \bigcirc\Diamond(\alpha_{n-1} \rightarrow \bigcirc\Diamond\alpha_n))$,$1 < j \leqslant n$

(3)在有分支的情况下,将条件($condition_k$,$condition_{k+1}$,…,

$condition_{k+n}$也考虑进去，则有形式化表达式：($condition_k \wedge \alpha_i \rightarrow \bigcirc\Diamond\alpha_{i+1}$) $\vee$ ($condition_{k+1}$ ($\alpha_i \rightarrow \bigcirc\Diamond\alpha_{i+2}$)，…，$\vee$ ($condition_{k+n} \wedge \alpha_i \rightarrow \bigcirc\Diamond\alpha_{k+n+1}$)，$1 \leqslant i \leqslant n$，$1 \leqslant k \leqslant n$。

(4)在有叉的情况下，如果转移是从一个状态 $\alpha_i$ 到一组并发状态 $\gamma_k$，形式化表达式为 $\alpha_i \rightarrow \bigcirc\Diamond\gamma_k$，其中 $\gamma_k = \vee_{p=i+1}^{n} \alpha_p$ $\vee$表示所有状态 $\alpha_p$ 的析取。

(5)在有汇合的情况下，如果有一组并发状态 $\gamma_k$ 同时转移到状态 $\alpha_j$，则总的形式化表达式为 $\gamma_k \rightarrow \bigcirc\Diamond\alpha_j$，这里 $\gamma_k$ 同步骤(4)，$j>p$，$i+1<p<n$。

### 4.2 实例研究

以一个简化的客户取款为例来说明采用 TDLs 形式化 UML 状态图的过程及可行性。客户取款的需求分析如下：

“客户从 ATM 机中取款，首先要准确输入存储在卡中的 PIN，其次卡中的余额要大于或等于客户要取出的金额，最后，ATM 机要自动从客户的余额中减去客户取出的数额。如果客户输入的 PIN 有误，ATM 机提示 PIN 错误，如果错误输入 3 次，ATM 机将返回磁卡；如果客户需求的数额大于卡中的余额，ATM 机提示余额不足，要求重新输入取款金额。”根据需求分析，首先构造用例图，如图 2 所示。

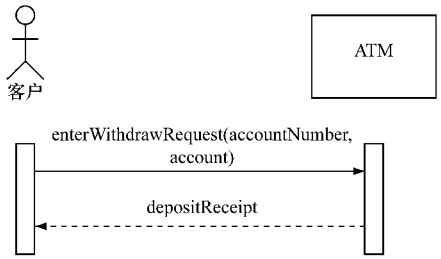


**图 2 客户取款用例图**

图中有 2 个角色：客户和 ATM 机。根据需求分析和用例图，构造客户取款的状态图，如图 3 所示。

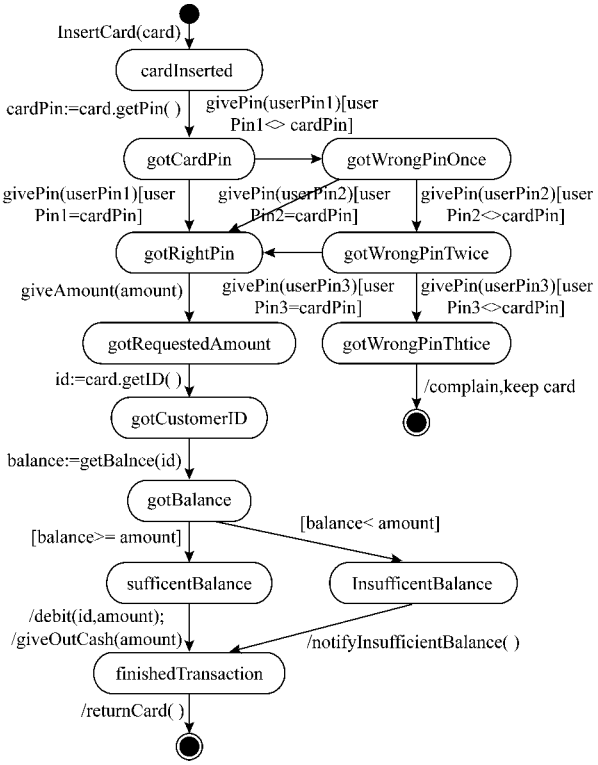


**图 3 客户取款状态图**

客户取款是以客户插入磁卡开始，以磁卡返回结束交易。在客户取款的状态图中，至少有 7 个控制流(实际可能会更多)，任何一个控制流都可以从初始伪装态到达终结状态，并且每个控制流的条件都是相互不冲突的。状态图中包含 12 个状态。根据转换规则，对客户取款的状态图形式化如下：

$\bigcirc\Box$(cardInserted$\rightarrow\bigcirc\Diamond$(gotCardPin$\rightarrow$
$\bigcirc\Diamond$((gotRightPin$\vee$gotWrongOnce)$\rightarrow$
$\bigcirc\Diamond$((gotRequestAmount$\vee$gotWrongPinTwice)$\rightarrow$
$\bigcirc\Diamond$((gotCustomerID$\vee$gotWrongpinTwice)$\rightarrow$
$\bigcirc\Diamond$((gotBalance$\vee$closeTransaction)$\rightarrow$
$\bigcirc\Diamond$((sufficientBalance$\vee$InsufficientBalance$\rightarrow$
$\bigcirc\Diamond\Box$finishedTransaction))))))

用 TDLs 表达式来表示状态图的形式化语义，不仅严格而且直观，并且将状态图转换为 TDLs 表达式以后，就可以在逻辑的框架下讨论相关的性质。在本例中，“取钱成功”必须满足的性质是：只要控制流走到状态 finishedTransaction，输入的账号和密码必须是正确的，并且钱的数额是充足的。这是一个安全性质，用 TDLs 公式表示为：

$\bigcirc\Box$(cardInserted$\rightarrow\bigcirc\Diamond$gotCardPin$\rightarrow$
$\bigcirc\Diamond$gotRightPin$\rightarrow\bigcirc\Diamond$gotRequestAmount$\rightarrow$
$\bigcirc\Diamond$gotCustomerID$\rightarrow\bigcirc\Diamond$gotBalance$\rightarrow$
$\bigcirc\Diamond$sufficientBalance$\rightarrow\bigcirc\Diamond\Box$finishedTransaction$\rightarrow$)

通过对“取钱成功”状态的分析，说明转换是正确的。

## 5 结束语

本文研究采用 TDLs 形式化 UML 状态图的方法。通过时序扩展，将基本的 DLs 扩展成为既可以表示静态领域的知识，又可以对动态和时序知识进行描述的时序描述逻辑，增强了描述逻辑的表达能力。另外，文中给出 UML 状态图到 TDLs 表达式的转换规则，通过规则，将状态图转换为 TDLs 表达式，提高了状态图的语义精确性。

## 参考文献


[1] James R, Ivar J, Grady B. The Unified Modeling Language Reference Manual[M]. [S. l.]: Addison-Wesley, 1999.

[2] Evans A, Bruel J M, Franee R, et al. Making UML Precise[C]//Proc. of OOPSLA'98. [S. l.]: IEEE Press, 1998.

[3] Berardi D, Calvanese D, de Giacomo G. Reasoning on UML Class Diagrams[J]. Artificial Intelligence, 2005, 168(1): 70-118.

[4] Baader F, Calvanese D. The Description Logic Handbook: Theory, Implementation, and Applications[M]. Cambridge, UK: Cambridge University Press, 2003.

[5] Carsten L, Frank W. Temporal Description Logics: A Survey [C]//Proc. of the 15th International Symposium on Temporal Representation and Reasoning. [S. l.]: IEEE Press, 2008.

[6] Alessamdro A, Eenico F. Introducing Temporal Description Logics[D]. Manchester, UK: University of Manchester, 2005.

[7] Sabine K, Martin G, Ralf K. An Integrated Semantics for UML Class, Object and State Diagrams Based on Graph Transformation[C]//Proc. of the 3rd International Conference on Integrated Formal Methods. [S. l.]: IEEE Press, 2006.

[8] 张晓蒙，戎　玫，张广泉. 基于 rCOS 的 UML 状态图语义研究[J]. 计算机工程，2009，35(2)：21-23.


编辑：陈　文