1999 年 12 月
第 29 卷第 6 期
西北大学学报(自然科学版)
Journal of Northwest University(Natural Science Edition)
Dec. 1999
Vol. 29 No. 6



# 基于时态逻辑的抽象对象规约方法



宋 悦,郝克刚,葛 玮
(西北大学 计算机科学系,陕西 西安 710069)

**摘要**:提出了一种基于时态逻辑的抽象对象语义描述方法,采用这种方法,可以在说明对象的同时对其行为加以时态限制,从而在语义层次上规约了并行对象系统的行为。在此方法上,还可以进一步对系统进行形式化的验证。

**关 键 词**:时态逻辑;计算树逻辑;反应系统;形式化方法;规约 谓词逻辑, 抽象对象规约
**中图分类号**:TP301.2 **文献标识码**:A **文章编号**:1000-274X(1999)06-0499-04

近年来,随着全球信息网络化的迅速发展,各种面向对象的分布式系统越来越受到人们的重视。系统越来越复杂,并行程度越来越高,使得人们需要对反应系统(reactive system)[1]的模型和规约进行研究。通常此类系统比其他系统难以分析和规约,而且传统的测试也难以保证系统的正确性、完备性(completeness)。传统的数据模型[2,3](如 ER 或 ERA 模型)建立于实体、属性和关系上,缺乏数据抽象能力,也没有表示动态行为的机制。另一方面,关于系统动态特性的研究,如 petri net,进程代数 CCS 等,虽适于用来描述并行系统,却与系统的静态结构方面分开进行,且抽象层次较低。这样,人们往往使用两种不同的模型来表示系统的静态方面和动态方面,整个软件开发没有统一的逻辑基础[4]。

对象模型是两者的有机结合。对象是数据封装的载体,对象提供一组操作供外界观察对象属性或改变对象的内部状态。对象的内部属性与对象之间的关联是对象的静态结构,为了能完整地反映对象的行为特征,对象的语义模型还应给出对象的动态变化历史。Pnueli 提出用时态逻辑[1]来表达反应系统的性质,在谓词逻辑上扩充了时态算子,对系统的动态行为进行描述和规约。时态逻辑在较高的层次上提供了对并行系统进行描述的能力,适于对并行反应系统在语义层次进行描述和验证。本文提出了一种基于时态逻辑的抽象对象规约方法(Abstract object specification),用以形式地描述对象语义。

## 1 应用时态逻辑规约对象

### 1.1 所采用的模型和时态算子

本文提出的抽象对象规约是基于 CTL(Compute tree logic)[5]的 。CTL 的合适公式是由原子谓词、布尔运算符、时态算子及加在时态算子前的路径量词(path quntifier)构成。形式地给定一个原子谓词集 $P$,CTL 的公式定义[6]如下:① true,false 及原子谓词 $p\in P$ 都是 CTL 的公式;② 如果 $\varphi_1$ 和 $\varphi_2$ 是 CTL 的公式,那么 $\varphi_1$,$\varphi_1 \vee \varphi_2$,$AX\varphi_1$,$AG\varphi_1$,$AF\varphi_1$,$A\varphi_1 U\varphi_2$,$WX\varphi_1$,$EG\varphi_1$,$EF\varphi$ 也是 CTL 公式。

其中,$X\varphi$ 代表 next $\varphi$,$G\varphi$ 代表 forever $\varphi$,$F\varphi$ 代表 eventually $\varphi$,$\varphi_1 U\varphi_2$ 代表 $\varphi_1$ untill $\varphi_2$,这几个是线性时序算子[1,5]。A 代表对于所有的计算路径,E 代表对于某些计算路径,A 和 E 是路径量词。公式 $AX\varphi$ 意为 $\varphi$ 在当前状态的每一个可能的后继状态都成立。$AG\varphi$ 意味着从当前状态开始的所有可能计算路径上的状态都满足 $\varphi$。$AG\varphi$,$EX\varphi$ 和 $EF\varphi$ 的含义的示例如图 1 所示。

其他 CTL 的算子可以类似地定义,本文不再细述,详见文献[5]。另外,本文采用的时态算子还有 $P\varphi$ 代表 once $\varphi$(ie. sometimes in the past),符号 $x'$ 表示变量 $x$ 的下一状态的值。为使 CTL 的语义能用于对象的描述,本文具体化了 CTL 的谓词集合。在其中加入了对象符号、属性符号及与(operation list)中

收稿日期:1999-01-28
基金项目:国家"863"计划资助项目(863-306-2T02-04-01)
作者简介:宋 悦(1971-)男,陕西蓝田人,西北大学硕士生,从事软件工程方面的研究。

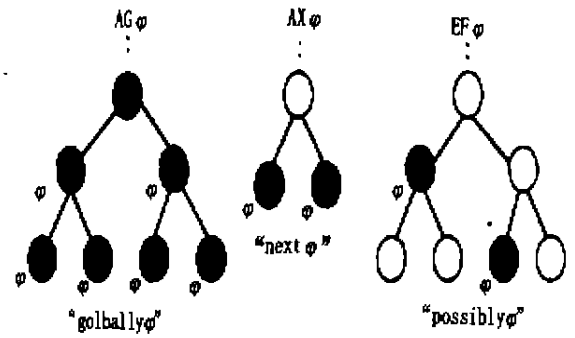


图 1 CTL 算子的含义

Fig. 1 The meaning of CTL

相应的谓词符号。新的谓词集合加入了形如 object. atrribute 和 object. operation 的谓词,其中 object 和 atrribute,operation 都是新加入的谓词符号。论域 U (univers)包括对象、对象的属性和操作。解释 I(interpretation)将符号集中的符号映射到论域中相应的值。为了表述对象行为间的时序关系,还加入了形如 enable(operation)的谓词,以表示在当前状态下该 operation 可以被对象执行,done(operation)表示在当前状态该操作已被对象完成。

为了形式化地解释时态公式,本文采用的模型是五元组 $M = (V, D, \Sigma, R, \Theta)$。其中 $V = \{u_1, u_2, \cdots\cdots, u_k\}$ 是状态变量的有穷集合,这些变量是对象的状态变量(由系统中各对象的属性组成)和控制变量。前者由规约中的语句显式的说明,后者指出系统运行过程中将要执行的对象及其操作的位置。$D = \{Du_1, Du_2, \cdots\cdots, Du_k\}$,其中 $Du_i$ 是变量 $u_i$ 的论域。$\Sigma$ 是状态集合。每一个状态 $s \in \Sigma$ 是 $V$ 上的一个映射,即给每一个变量 $u \in V$ 赋予其对应论域 $Du$ 中的一个值,用 $s[u]$ 表示。$R$ 是一个函数:$\Sigma \to 2^{\Sigma}$,它将每个状态 $s \in \Sigma$ 映设到(可能为空)后继状态集 $\tau_1(s) \subseteq \Sigma$,$\theta$ 是系统的初始条件。它是一个表示全体初始状态的断言,亦即程序开始执行的状态断言。反应系统的一个可能的计算可表示为一个 $\Sigma$ 中的无穷状态序列 $\sigma = s_0, s_1, s_2, s_3\cdots\cdots$,称为一条计算路径。其中 $\sigma$ 满足满足初始条件 $(s_0 \mid -\theta)$,且对于 $\sigma$ 中任意 $s_i, s_{i+1} \in S, s_{i+1}\tau_1(si)$。显然,由于后继状态的不确定性,即使对于相同的初始条件,计算也不定惟一。所有可能的计算构成一棵计算树。

### 1.2 应用时序逻辑方法规约对象类

对象类规约的形式框架采用如下形式,

(class _ specification) :: = ClassSpec (class _ name)

Inherit (inherit _ class)

Attribute (attribute _ list)

Operation (operation _ list)

Static constraint (state _ constraint)

Dynamic constraint (temporal _ constraints)

End Spec

(inherit class) :: = class _ name

//可以为空

(atrribute list) :: = (name: type)*

//对象类属性列表

(operation list) :: = (operation _ name[precondition][postcondition])*

//带前/后置条件的操作列表

(state constraint) :: = predict _ formula

//状态约束

(temporal constrains) :: = temporal _ fomula

//对象的行为约束

其中(attribute list)用于说明对象的属性。(operation list)是对象可以执行的操作集合,对象状态的转换只通过其操作来实现的。在(operation list)中,create 和 destroy 是两个特殊的操作,create 操作是对象的初始操作,destroy 操作是终结操作,对象的状态转换总是以执行 create 操作开始,以执行 destroy 操作结束(如果对象能终结的话)。为简单起见,不考虑操作带参数的情况。(state constraint)是一组用一阶谓词逻辑公式表示的对象静态约束的集合,它规定了对象的动态约束、时序特性和对象间的同步。它规定了对象的合法状态转移之间满足的时序制约关系,即对象在当前状态所能(用 enable 谓词来表示)进行的操作。这样,对象类就被定义为一个四元组(A, Op, Cst, Cdy)。其中 A 是对象的属性集合。Op 是对象操作集合。Cst 是一组用一阶谓词逻辑公式表示的对象静态约束集合。对象实例通常是作为一个类的成员而存在,从对象类生成对象实例需经过命名和初始化,每一对象实例有一个全局惟一的标志名,对象实例的行为必须满足对象类所定义的各种约束。下面用一个累加器的例子作为说明。

Example 1: ClassSpec counter

Attribute i: int;

Operation create[pre condirtion]: ¬ Done(create)

[post condition]: allocate(i) ∧ i=0 ∧ Done(create)

increase[pre condition]: i = 0 ∧ Done (create)

Done(destroy)

[post condition]:i'=i+1

destroy[pre condition]:Done(create) ∧¬ Done(destroy)

[post condition]:free(i) ∧ Done(destroy)

Static Constrain i>=0

Dynamic Constraint

AG(Done(create) ∧¬ Done(destroy))⇒(Enable(increase) ∧ Enable(destroy))

AG(Done(create) ∧¬ Done(destroy)⇒A(Enable(increase)U Done(destroy)))

¬ EF((Enable(increase) ∨ Enable(destroy) ∧ ¬ Done(create))

¬ EF((Enable(increase) ∧ Done(destroy))

End Spec

累加器对象类 counter 没有父类,它的属性包括一个整数 i,可以执行的操作包括构造 create,累加 increase 和析构 destroy。其中,increase 的后置条件规定了 increase 操作只能每次对属性 i 加一。static constraint 中 i>=0 表示 counter 对象的合法状态空间为非负整数。在 dynamic constraints 中,AG(done(create)⇒AX(enable(increase) ∧ enable(destroy))表示对象被创建后可以累加或被销毁。

AG(done(create) ∧ done(destroy)⇒A(enable(increase)U done(destroy)))表示对象可以累加直至被销毁。EF(enable(increase) ∨ enable(destroy) ∧ done(create))则表示在创建前不存在对象可以累加或被销毁的情况。EF(enable(increase) ∧ done(destroy))说明对象销毁后不能再进行累加。这样就限定了 increase 和 destroy 只能在 create 之后出现,而 destroy 则只能在 increase 之后出现。

## 2　对象的继承和聚合

为了体现类之间的概括和特化关系,类之间可以通过继承关系构成类层次结构(类格)。子类继承其祖先类的所有内部结构包括属性和操作,从而支持代码复用。子类还继承其祖先的行为特性,包括静态约束和时序约束。当然,子类亦可添加属于自己的新的属性和操作,也可细化或重载父类的操作,但要求这些修改必须保证子类与父类的语义一致性,即:① 子类对象的行为要符合父类对象的行为约束;② 对于子类中的重载操作,其操作规范的前置条件不应强于父类中相应的前景条件,后置条件不应弱于父类中相应的后置条件。

聚合是提供由已有对象的描述构造复杂对象的描述手段。为描述聚合对象,必须为构成聚合对象的各成分对象提供名字。此外,还必须描述各成分对象间及成分对象和聚合对象间的关系(结构和行为关系)。在本文的对象描述中,成分对象是作为聚合对象的属性出现在聚合对象的 attribute 子句中,即本文的对象描述中的属性可以是数据属性也可以是对象属性。聚合对象将保有其各成分对象的特性(各种约束)。成分对象的属性只能通过成分对象所提供的动作加以修改,成分对象之间以及成分对象与聚合对象间的相互作用可以通过聚合对的动态约束和动作来描述。

## 3　描述对象的同步

在本节通过一个聚合对象的例子来说明如何进行对象的同步规范。首先,给出的是对象 sample 的规约,其中用到了前面说明的 counter 对象。

Example 2:ClassSpec sample

Attribute c1,c2:counter;

Operation create[pre condition]: Done(create)

[post condition]:Done(create)

print[pre condition]:Done(create) ∧ Done(destroy) ∧ (c 1.i=c2.i)

[post condition]:c1.i=c2.i

destroy[pre condition]:Done(create) ∧ Done(destroy)

[post ondition]:Done(destroy)

Static constraint

Dynamic constraint

AG(Enable(create)⇒AX(Enable(c1.create)) ∧ AX(Enable(c2.create)))

AG(Done(c1.increase)⇒AX(A( Enable(c1.increase)U Enable(print))))

AG(Done(c2.increase)⇒AX(A( Enable(c2.increase)U Enable(print))))

AG(Done(print)⇒AX(Enable(c1.increase)) ∧ (Enable(c2.increase)))

¬ EF(Enable(print) ∧ (c1.i≠c2.i))

End Spec

Sample 对象由两个 counter 对象聚合而成,由于除了在 dynamic constraint 中列出的约束外,不再有别的限制,故 c1,c2 被分别独立地进行计算。sam-

ple 对象在创建时首先创建两个 counter 对象，两个 counter 对象各自累加，当 c1，c2 分别累加一次后，由 sample 对象的 print 操作输出，然后继续这一过程。对其施加的限制为 c1，c2 完成一次累加后，直至 print 输为止出不能进行第二次累加。当输出后，应继续累加过程(liveness)。表示其可能计算路径的 Kriple 图(interleave 结构，略去对象的创建过程)表示如图 2。

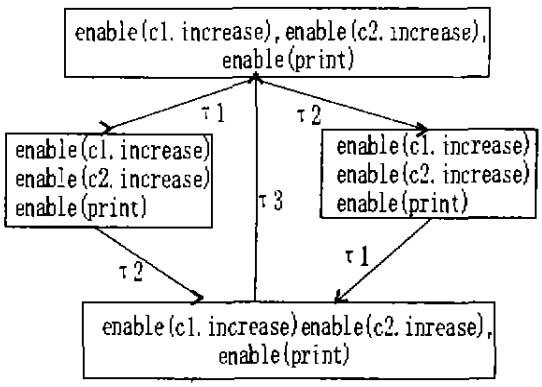


图 2 Sample 对象的状态转换图

Fig. 2 The chart of sample object state transaction

## 4 结束语

在形式化研究中，采用了许多方法来描述系统的动态特性和静态特性，通常的做法是用不同的模型加以分别表述。如利用 Z 语言来表示静态特性和 CCS 来表示动态特性。此外，还有许多 Z 的扩充语言，如 OOZE，OBJECTZ，Z＋＋等，但它们在对象的时序性质表述方面都不理想[7]。

本文试图采用时态逻辑来规约对象，将对象的静态方面和动态方面结合起来。未来的工作将集中在形式化的验证对象和对象的自动验证，尤其是对时态限制的验证，仍然需要进一步的研究。

## 参考文献：

[1] Manna Z，Pnueli A. The temporal logic of reactive and concurrent systems[M]. NewYork：Springer-verlag，1992.

[2] 王元元，计算机科学中的逻辑学[M]. 北京：科学出版社，1989.

[3] Rumbaugh J，Blaha M. Object-Oriented modeling and design[M]. NewYork：P-Hall Inc，1991.

[4] 黄 涛，冯玉琳. 对象描述语言及其指称描述[J]. 软件学报，1996，7(10)：577-586.

[5] Kupferman O，Grumberg O. Branching-time temporal logic and tree automata[J]. Information and computation，1996，125：62-69.

[6] Joanne M. State-Based model checking of event driven system requirement[J]. IEEE transaction on software engineering，1993，19(1)：24-39.

[7] 袁晓东，许 皓，胡德强，等. 现有 Z 面向对象扩充语言的比较[J]. 计算机科学，1997，24(3)：58-61.

(编 辑 曹大刚)

# Using temporal logic to specify abstract object

SONG Yue，HAO Ke-gang，GE Wei

(Department of Computer Science，Northwest University，Xi'an 710069，China)

**Abstract**：An abstract specification method of concurrent object family was presented. Temporal logic CTL was used as the base of language to specify dynamic behavior of object and synchronic between them. Using this method，object data and its behavior can be specified in one object model. Furthermore，object semantic can be checked formally and automally with this method.

**Key words**：temporal logic；CTL；reactive system，formal method；specification