计算机科学 1996 Vol. 23 №. 6

基于模型的规格说明方法

魏峻 周莺 周顺 何克清

(武汉大学软件工程国家重点实验室 武汉 430072)

摘 要 这篇文章分析了基于模型(转换系统)的规格说明方法的特点,给出了形式方法的严格定义,并且通过实例,探讨了基于模型的规格说明方法在公平条件约束下,描述系统安全和活动性质的能力和特点。

关键词 模型,转换系统,规格说明,公平,行为,活动性质,安全性质

当今软件系统的发展具有一些显著的特征,如开放性、分布并发计算模式、基于部件(对象)的可组合性等。软件开发的方法很多,但都离不开对软件系统性质及行为规约的定义和描述。一般规格说明方法是形式化的,利用方法自身的验证或证明机制来保证它所描述的性质的可满足性及行为规约的一致、有效性。在表述软件系统的规格说明时,应提供相应的手段,即应是开放的语义;提供可组合的语法定义及语义说明;具备逐步求精的策略;重要的是有能力描述软件系统的一些性质,如安全性、活动性等。在此方法框架下,形式描述的软件系统不再是单一的整体,而应是分布的实体的组合,而且从抽象层次看,一个实体可视作一个反应式系统,与环境(由其它实体组成)交互,为环境提供服务。这类抽象计算模式已在不同领域进行了研究,包括软件工程[2],通讯协议[6],分布式算法设计[4],并发程序语义[1,7]等。

在上述的领域研究中,虽然表示系统组合及证明技术支持上各有差异,但大多数采用视单个局部化的单元(对象)为一个转换系统(transition system-TS)的观点。在具体表现上,有基于状态和基于动作的两种方法,前者[3,4]走逻辑的路,用断言表示转换关系(状态变化),用逻辑公式表示动作的语义;而基于动作的方法[1,5,7]是用操作语义来表示转换和动作。不管采用什么方法,在一定抽象层次上,用转换系统描述系统的行为以体现系统具有的性质,就是基于模型的。这里,我们采用带公平条件的转换系统作为基本模型,其基本语义是操作语义,与文[5]类似。

一、基于模型的规格说明

基于模型的规格说明应具有如下特点:

1. 开放性。一个系统是开放的,是指它与环境要发生交互。我们采用的模型,其语义是开放的,转换中包含了与环境的交互(输入、输出)。

2. 可组合性。一个软件系统是一个分布式系统,其规格说明是分布的各个部件的规格说明的组合。

3. 并发性。软件系统由独立部件组合而成,决定了模型描述的软件系统是分布并发计算系统,其计算模式是交替并发计算(interleaving computing),即不同部件的执行不是真正的并发执行,而是按照一个线序排列,交替进行的。

4. 可信性。只有转换系统的交替并发计算与它需描述的实际系统的重叠计算是一致的,此计算模型才是可信的[6]。

5. 可逐步求精。基于模型的规格说明 T_1 被 T_2 求精,即模型 T_1 表现的行为特性在模型 T_2 中都可体现。T_1 是高层次、粗粒度的规格说明,对其逐步求精为低层次、细粒度的 T_2—对应实现。此求精策略就是定义一种模拟关系,完成规格说明到实现的映射和证明。

1.1 安全性和活动性的描述

使用基于模型的方法描述系统的规格说明,实质上是表示系统应满足的一般形式地被地定义为行为集合的一些性质。系统典型的性质是某些安全性和活动性。下面我们就基于模型的规格说明方法与其它方法对比,说明其描述系统性质时的特点。

(1)安全性质,是指给出一个性质,对系统的所有执行应保持成立。典型的安全性质如相互排斥等,这些性质就是环境对系统的需求规格说明。许多研究工作采用时态逻辑表示这些性质,即用时态逻辑公式作为需求规格说明。安全性质通常表示为□P,P 是一个状态公式,表示性质;□是时态算子,表示"经常",安全性要求性质 P 经常满足。

我们用提供的一个带公平约束的转换系统的模型给出这种需求,其所有执行正是所期望的满足性

质的要求。基于模型的规格说明的优点在于可以使用映射技术一"求精"、"模拟"等来表明系统实现的正确性。构造满足某些安全性质的系统规格说明(基于模型),对规格说明逐步求精,细化的结果看作系统实现(也基于模型),然后证明两者满足模拟关系,通过归纳,系统的执行对应规格说明的执行,则系统实现满足规格说明描述的某些安全性质。

(2)活动性质,指断言某事最终必发生。这意味着,在开放环境中,如果描述了部件能提供的服务,则必须知道这些服务保证响应请求的范围。如果规格说明只说明安全性,是不会令人满意的,因为一个不提供任何服务的系统可以满足任意安全性质。因此,在规格说明中,必须表示出活动条件,典型的活动性质包括,程序终止、服务响应和无饥饿等。

对于规格说明的活动性质表示,最常使用的方法是时态逻辑,一般形式为□(P→◇G),即无限经常 F 成立,导致最终 G 会满足。在文[6]中,用时态逻辑既表示了安全性又表示了活动性;还有一些混合方法[3,4],针对基于模型的安全规格说明,用时态逻辑表示其活动性,即规格说明包括一个模型和时态公式,在时态逻辑方法中,活动性验证遵从的是推理风格而不是映射风格—其有效地运用于安全性验证。

在基于模型的规格说明中,我们采用了同表示安全性一致的方法来表示活动性。首先,给出满足安全性质要求的基本转换系统模型作为规格说明,在此基础上,定义某种公平条件。公平条件是从模型的所有执行中,挑选出特殊子集。定义公平行为很重要,各种条件下的公平行为实质上对应着规格说明描述的各种活动性质。通常采用的公平定义是:在公平执行中,任意被连续使能的转换最终必发生。通过一致的映射技术,验证系统的所有公平执行是规格说明模型的公平执行,据此来说明系统满足某种活动性质。

二、形式方法

我们的形式方法中的基本模型称为 ETS,其中输入 In(T),输出 Out(T)和内部 Int(T)动作集合 Act(T)=In(T)∪Out(T)∪Int(T)。输入动作表示在环境控制之下进入 ETS 的离散事件;输出动作表示在 ETS 控制之下且会影响环境的事件;内部动作在 ETS 控制之下并不被环境所观察。我们只关心与环境相关的动作,即外部可观察动作,表示为 Ext(T)=In(T)∪Out(T)。处于局部控制(ETS 控制)的动作表示为 Local(T)=Int(T)∪Out(T)。

定义 2.1 (扩充转换系统) 一个扩充转换系统 T 是一个元组(S,S_0,A,t,P)。其中:

S 是状态集合。S 引用为 States(T);$S_0\subseteq S$ 是初始状态集合,S_0 引用为 Start(T);A 是动作集,引用为 Acts(T);$t\subseteq S\times Acts(T)\times S$,引用为 Steps(T),其元素$(s,a,s')\in t, s,s'\in S, a\in Acts(T)$;a 被使能,定义为在状态 s 下,动作 a 发生的条件满足且存在状态 s',使$(s,a,s')\in t$。为反映输入动作在环境控制之下,我们规定在某状态下的所有输入动作都被使能;P 是对 Local(T)的划分,引用为 Part(T)。

在引入公平执行的概念后,P 所起的作用会更清楚。划分局部动作集的主要思想是视一个系统为子部件的集合,各个划分部分的动作处于部件控制之下,根据此定义系统的公平性。

定义 2.2 (执行、迹和行为) T 的一个执行是有限序列 $s_0a_1s_1a_2s_2\cdots a_ns_n$ 或无限序列 $s_0a_1s_1a_2s_2\cdots a_ns_n\cdots$,$s_0$ 是初始状态,$(s_i,a_{i+1},s_{i+1})\in Steps(T)$,且不同状态和动作是交替的,表示为 Exec(T)。T 的迹是只关注动作的执行的子序列,表示为 Trace(T)。T 的行为是执行的子序列,即去掉执行中所有状态和内部动作,只由外部动作组成,Beh(T)=Exec(T)|Ext(T)。

定义 2.3 (非公平等价) A 和 B 为两个 ETS,称为非公平等价 $A\cong_U B$,当且仅当 A 和 B 有相同的外部动作 Ext(A)=Ext(B)和行为 Beh(A)=Beh(B)。

这是关于 ETS 的一种观察等价的概念,它标识所有能执行同样外部动作序列的 ETS。

下面我们介绍三个基本操作:隐藏、换名和并行组合。可以看到,以上关系是关于这三个操作的同余,即在三个操作下保持等价关系。

定义 2.4 (隐藏) 给出一个 ETS,T=(S,S_0,A,t,P)和动作集 I,且 $I\cap In(T)=\varnothing$,我们称 $Hide_I(T)$是一个 ETS $Hide_I(T)=(S,S_0,A',t,P)$,A'与 A 在以下方面不同:①$Out(Hide_I(T))=Out(T)\backslash I$;②$Int(Hide_I(T))=Int(T)\cup(Acts(T)\cap I)$。

隐藏操作把外部动作转换为内部动作,即对外部环境隐藏了一些局部控制的动作,结果 ETS 与原始 ETS 不同在于外部接口的改变。Ext(T)=Ext($Hide_I(T)$),但 Beh(T)≠Beh($Hide_I(T)$)。条件 $I\cap In(T)=\varnothing$限制了:如果我们允许输入动作的隐藏,则操作 Hide 对 ETS 将不是封闭的,导致 Part(T)不再是 Local(T)的一个划分。此操作对改变一个 ETS 的开放性有很大的关系。

定义 2.5 (换名) 给出一个 ETS,T=(S,S_0,A,t,P)和一个单射 f,定义 f(T)为(S,S_0,A',t',P'),A',t',p 定义如下:①In(T')=f(In(T)),Out(T')=

f(Out(T)),Int(T')=f(Int(T));②t'={(s,f(a),s')|(s,a,s')∈Steps(T)};③P'={(f(a),f(a'))|(a,a')∈Part(T)}。

下面引入ETS的并行组合概念，定义ETS的并行组合需满足相容性，相容性条件要求内部动作不用于通信，且每个动作至多处于一个部件(进程)的控制之下。

定义2.6 (ETS的相容) (1)动作集的集合{A_i|i∈I}是相容的，当且仅当对所有i,j∈I,i≠j,(a)Out(T_i)∩Out(T_j)=∅;(b)Int(T_i)∩Acts(T_j)=∅。(2)ETS集合{Ti|i∈I}是相容的，当且仅当它们的动作集是相容的。

定义2.7 (ETS的组合) 相容ETS{T_i|i∈I}的组合T=‖_{i∈I}T_i，定义为一个ETS，其中：

①$States(T)=\prod_{i\in I}States(T_i)$。②$Start(T)=\prod_{i\in I}Start(T_i)$。③相容动作集的组合A=‖_{i∈I}A_i，定义为：(a)In(T)=∪_{i∈I}In(T_i)−∪_{i∈I}Out(T_i);(b)Out(T)=∪_{i∈I}Out(T_i);(c)Int(T)=∪_{i∈I}Int(T_i)。④Part(T)=∪_{i∈I}Part(T_i)、⑤Steps(T)={(s,a,s')|s,s'∈States(T)，如果a∈Acts(T_i)，则(s[i],a,s'[i])∈Steps(T_i);若a∉Acts(T_i)，则s[i]=s'[i],s,s'为状态向量。

此组合转换表示了所有进程必须在动作上同步。由于输入被使能以及局部控制等条件，此处的同步不是指通信的同步，而是指在一时刻只有一个进程能决定通信发生，其他进程以空转换步保持同步。

ETS的并行组合语义是：组合结果仍是一个ETS，完成并行T_i的计算。ETS描述的并发系统的计算模式是交替并发，要使其与描述的实际系统的真并发计算一致，必须说明此计算模式的可信性。

定义2.8 (ETS的可信) 给出一个ETS，T=(S,S_0,A,t,P)。我们称T是可信的，当且仅当T满足限制临界引用约束条件(LCR—Limited Critical Reference)。

可信的ETS的交替计算与实际系统的重叠计算是一致的[6]。

前面谈论的执行，实质是基本TS的执行。在ETS中，由于输入被使能，产生了副作用。无限的输入动作序列，可以使系统处于不执行局部控制动作的状态，为了避免系统出现这类事件的发生，则要限制可观察的执行只为公平的。

定义2.9 (公平执行、迹) ETS的公平执行Fexec(T)是一个执行α，对所有X∈Part(T)：如果α是有限的，那么没有X的动作在α的最后状态被使能。如果α是无限的，那么或者来自X的动作无限经常地出现在α中，或者那些没有X的动作被使能的状态无限经常地出现在α中。一个公平迹是公平执行的迹，ETS的公平迹的集合表示为Ftrace(T)。

在ETS中，公平只涉及局部控制的动作，且公平是通过讨论动作集合而不是单个动作来定义的。局部控制动作划分的意义是：划分中的每个元素是在全局系统的一个组成部件的控制之下的动作集。在此框架下，公平机会的概念表示为，每个连续愿意执行其局部动作的部件，最终将执行其一。

我们可以定义一新的等价关系，它是ETS的一个弱同余关系。

定义2.10 (公平等价) A和B两个ETS是公平等价A≡_FB，当且仅当A和B有相同的外部动作Ext(A)=Ext(B)且Ftrace(A)=Ftrace(B)。

基于公平迹的概念，引入求精(或实现)的概念。

定义2.11 (求精) A和B是两个ETS，称A被B求精或B求精A，记作A≺B，当且仅当①In(A)⊆In(B)且Out(A)⊆Out(B);②Ftrace(A)⊆Ftrace(B)。

求精关系满足自反、传递性。如果A≺B，直觉上B是A更详细的规格说明。在基于模型的规格说明的逐步细化过程中，部件可根据这两个定义而替换。在我们方法中，用ETS同时表示规格说明SPEC和系统SYS。只要验证两者满足SYS≺SPEC，则规格说明到实现的映射中，保持了规格说明满足的性质，保证了实现的正确性。关于验证方法的形式证明不是本文讨论的焦点，在此省略。

三、实例分析

实例系统的描述为“非共享资源R由N个用户进程使用。R的存取由分配器A_R管理。用户进程U_i使用R之前，向A_R发出请求，当R变为可自由存取时，A_R作标志反应。当用户进程U_i使用完R时，向A_R发出信号释放R。在请求资源及获悉分配到资源间的任意时刻，用户进程可以取消请求。系统必须保证在任何时刻，R不会分配给多于一个的U_i(安全需求)”

给出系统的规格说明ETS的第一步是，标识且给系统与环境交互的动作命名，接着分别确定由系统控制的(输出)和由环境控制的动作(输入)。在此实例中，我们只关注资源分配器A_R的规格说明，选择其如下动作：

Request(i):U_i请求存取(A_R的输入);

Cancel(i):U_i取消未完成的请求(A_R的输入);

Grant(i):U_i被授权获得存取(A_R的输出);

Finish(i):U_i释放资源(A_R的输入)。

以上动作中，i 值不同，表示不同的动作。如 Request(1)与 Request(2)为不同动作。

状态空间 S 由下列变量的所有可能赋值决定：req 是一个布尔数组，req[i]＝True 表示 U_i 请求 R，但未满足或取消。grant 也是一个布尔数组，grant[i]＝True 表示 U_i 当前可存取 R。

动作类型的语法结构为如下形式：

〈动作名字〉WHEN〈条件〉DO〈表达式〉

转换形式上是三元组(s,a,s')，实质上转换由动作类型描述。a 对应动作〈名字〉；〈条件〉决定了所处的状态 s；s'是由〈表达式〉中对状态变量的赋值决定的。

下面是满足 A_R 的安全需求的规格说明，记为 T_{safe}：

①Request(i) WHEN True　DO req[i]：＝True

②Cancel(i) WHEN True　DO req[i]：＝False

③Finish(i) WHEN True　DO grant[i]：＝False

④Grant(i) WHEN(req[i] AND ¬∃ j,grant[j])

DO {grant[i]：＝True,req[i]：＝False}

我们以 ETS T_{safe} 为基础，探究公平行为。一般地，假设每个输出动作在等价关系划分的分离类中，则建立了 Part(T)。我们打算了解由此模型的公平执行描述的服务所体现的活动性质。

可以发现，此模型满足如下活动性质，"如果任意用户提出请求且未取消它，则分配器最终必会授权某用户使用资源。"。然而，此模型允许任意用户饥饿，即存在公平执行——一个特定用户提出请求，却从未被授权或取消。虽然公平性会强使某用户允许存取，但不能确定就是此特殊用户将会被授权。假设一个等价类连续被使能，由于公平性，来自此等价类的一个动作将发生。无论何时，一用户获得存取权，所有其它 Grant[i]变为非使能。因此，即使特定 Grant[k]未发生，公平性也没有被破坏。

下面是一个公平执行 α_{starve}，用户 2 挨饿。

Request(2). Request(3). Grant(3). Finish(3). Request(1). Grant(1). Finish(1). Request(3). Grant(3). Finish(3). Request(1). Grant(1). Finish(1)…

α_{starve} 是公平的，因为每次资源 R 被分配，所有其他的 Grant(i)动作是非使能的，Grant(2)未连续被使能，所以，即使 Grant(2)在单独的等价类中，公平性也不能保证此动作发生。

如果我们只通过改变等价关系来改变规格说明——所有的输出动作在同一等价类中，那么会有什么变化呢？首先，此时的规格说明没有破坏安全需求；其二，它仍允许饥饿发生，因为当一用户被授权使用资源，则所有其它输出动作是非使能的。上例的执行，没有 Grant(2)发生，但在此时的规格说明下，仍是公平执行。然而，改变等价类，对公平执行集合 Fexec(T)是有影响的。例如下列一个执行在前面是公平的，但现在是不公平的。

Request(1). Request(3). Cancel(1). Request(1). Cancel(3). Request(3). Cancel(1). Request(1)…

此执行中，每个请求在后面取消。但在交替中，每个状态有某个用户的输出动作是使能的(请求者)，由于输出动作在单个等价类中，等价类被连续使能，但没有输出动作发生，它是不公平的执行。

上面划分改变局部控制动作的等价类，决定了不同的公平执行集，但未改变这些公平执行体现出的规格说明的活动需求。

我们想不仅仅简单地重定义局部控制动作的等价类。下面改变一下规格说明 T_{safe}，使之公平行为包含很多没有输出动作出现的序列。在 T_{safe} 中加入两个内部动作 stop 和 go 成为规格说明 T_{pause}，其状态变量是在 T_{safe} 中加入 status(布尔型，初值为 True)，其转换关系如下：

①Request(i) WHEN True　DO req[i]：＝True

②Cancel(i) WHEN True　DO req[i]：＝False

③Finish(i) WHEN True　DO grant[i]：＝False

④stop WHEN status　DO status：＝false

⑤go WHEN ¬status　DO status：＝True

⑥Grant(i) WHEN (req[i] AND ¬∃ j, grant[j] AND¬status) DO {grant[i]：＝True,req[i]：＝False}

在 T_{pause} 中，划分每个输出和内部动作为分开的等价类。忽略 T_{pause} 的状态中的 status 部分及内部动作，我们可以看出，它的任何行为是 T_{safe} 的行为。而 T_{safe} 的任何行为可成为 T_{pause} 的公平行为。对无限执行，我们把 T_{safe} 的执行的每个状态 s 替换为如下序列：起始状态为 T_{pause} 的一个状态，status＝True，其他同 s；接以内部动作 stop；接着是一个状态，其 status＝True，其他同 s；再为内部动作 go；最后是 status＝True，其他同 s 的状态。对有限执行，则改变结束状态 s；加入 status＝True，其他同 s，接以一个无终止的模式：内部动作 stop→状态(status＝False，其他同 s)→内部动作 go→状态(status＝True，其他同 s)。输出动作处于无限经常的非使能，每个内部动作无限经常地被执行，所以这些行为都是公平行为。

但容易看出，T_{pause} 的一个公平行为可不是 T_{safe} 的公平行为。如下面的公平行为不满足 T_{safe} 的活动性质：Request(1). Request(2). stop. go. stop. go….

(参考文献共 8 篇略)