协议工程 模型 计算机网络 ⑤

计算机科学1993Vol.20No.5

21—26

协议工程行为模型的研究

TP393

李腊元 （武汉水运工程学院 武汉430063）

摘 要

This paper has made the study on behavioral model for protocol engineering. A behavioral model which is hierarchical is described. In terms of this model, main concepts and properties of protocol engineering can formally be defined, and semantical relations of different FDT's are also discussed.

近年，随着计算机网络和分布式系统的不断发展，各类新型通信技术和分布式应用已开始出现，其中主要包括高速光纤网、多介质通信、宽带综合业务数字网(B-ISDN)、智能网，以及综合语音、数据和图象服务等。它们已对计算机通信协议的设计和实现带来了很大影响。为了适应这种形势的发展，一门新兴的高科技学科—协议工程已应运而生。协议工程本质上是计算机硬件工程和软件工程的理论方法在通信协议设计和实现中的具体应用，但由于协议具有实时、互操作和同步性等特点，因而其复杂度又要比一般传统软件大得多。协议工程的主要研究目标是使协议软件的生产怎样更好地实现规范化、工程化和自动化。它所包含的基本内容是：协议及服务的形式描述，协议验证和证实，形式描述的自动生成，协议转换，性能分析，半自动实现以及一致性测试[3]。迄今为止，协议工程在开发形式描述技术(FDT)、研制适应于某种FDT的编译器，以及探讨其综合开发环境或产生其自动工具等方面已取得不少可喜的成果，但从整体来看，协议工程的研究仍处在不断发展和完善之中，许多问题尚待进一步研究和探讨。为此，本文将研讨适应于协议工程的行为模型，旨在为协议工程有关概念及性质的形式描述，以及探讨FDT之间的语义关系提供一种抽象的形式模型。

一、一种行为模型

为了形式地描述通信系统或协议的各种行为特性，我们可描述一种抽象层次模型，其结构如图1所示：

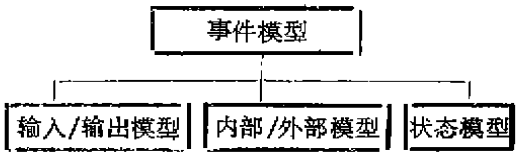

图 1 抽象层次模型

该模型分为两层：高层包括事件模型，它主要依据抽象事件来刻画系统的行为，并可用来描述和定义各种设计原理和相关的行为特性。低层主要包括输入/输出模型、内部/外部模型、状态模型，它们为表达事件提供了更为详细的形式途径，可进一步描述某些辅助设计原理和性质。此外，图1所示的模型还为各种FDT的综合分析（比较）提供了一种统一的基础。

1.1 事件模型 该模型是其它行为模型的基础，它所包含的基本概念是时间和事件。在下面的讨论中，我们用Time表示所

收到日期：92—12—05。李腊元 教授，IEEE INFOCOM、ICCC、ISDSRC等国际学术委员，国际论文评委，从事计算机网络和协议工程学的教学和科研工作。

概念，弥补了通常面向对象模型中的对象标识oid和关系模型中key的不足，为面向对象模型和面向值的模型的结合创造了条件。

（参考文献共19篇略）

有时间的集合，用Event表示所有事件的集合。实际上，时间值可表达事件发生的相对次序，也可表达实际时间。此外，Time可以是一个有限或无限的连续值集合，也可以是一个有限或无限的离散值集合。对于某些特定的规范，时间可以用一些简单的类型或结构化类型表示，如对应于原子事件的自然数或实数，以及对应于一个时间间隔的连续值集合等。并不是所有的场合都要求时间是全部有序的，时间值的部分有序可能更适于处理某些问题。我们用≤表示Time是部分有序的。类似地，对于某些特定的规范，Event也可有不同的表达形式，例如一个名字集合或一个状态转移集合等。一般而言，一个通信系统（或协议）可通过其可能的行为集合来建模。每个行为又可进一步由Time与Event之间的关系来建模。下面，采用Z对该模型进行形式描述(以下均同)[1,2]：

$$Behav == Time \leftrightarrow Event$$

$$System == PBehav$$

在该模型中，高层是非连接的，即一个系统将展示一种可能的行为。低层则是连接的，即一个系统必须为所选择的行为展示所有的时间-事件。

1.2 输入/输出模型 上述事件模型可借助输入/输出模型进一步求精，即把每个事件看成是与其环境的一次有向通信，它可以是输入，也可以是输出。采用Z可将该模型表征如下：

$$Input, Output : P\ Event$$

$$Input = Event - Output$$

1.3 内部/外部模型 在某些场合，可将一部分事件看成是无输入和输出的，而将另一部分事件考虑成是有输入和输出的（如将一个低级交互顺序建模成一个高级原子事件）。对于前述事件模型，可借助将事件区别成内部和外部的来进一步求精。其中内部事件可视为不可观察和或不能由一个系统的环境来控制的事件。在本模型中，Event可分成集合Internal和External。

$$Internal, External : P\ Event$$

$$Internal = Event - External$$

1.4 状态模型 该行为模型与状态转移有关。状态转移可改变系统的状态。若用State表示所有状态的集合，则有：

$$pre, post : Event \rightarrow State$$

所有性质的集合和性质的等价关系可定义如下：

$$compat : Property \leftrightarrow Property$$

实际上，性质的兼容性将与一个特定FDT的语义有关。例如在一个基于逻辑和集合的语义框架中，当式子x=3和x=4时，公式p和¬p将是不兼容的。

对于任意给定的某个形式化系统，可进一步定义有关函数，这里假设与一特定状态或行为有关的任意两个性质都是兼容的。

$$sprops : State \rightarrow P\ property$$

$$bprops : Behav \rightarrow P\ property$$

$$\forall s : State \cdot \forall P_1, P_2 : sprops(S) \cdot P_1\ Compat\ P_2$$

$$\forall b : Behav \cdot \forall P_1, P_2 : bprops(S) \cdot P_1\ Compat\ P_2$$

如前所述，一个系统可由相关的事件和时间来建模。事件在各自对应的时间内发生，且一个事件的前一状态需与其后一状态相同。即有：

$$Behav == \{b : Time \leftrightarrow Event \mid \forall (t_1, ev_1), (t_2, ev_2) : b \cdot t_1 = t_2 \Rightarrow \forall P_1 : sprops(pre\ ev_1), P_2 : sprops(pre\ ev_2) \cdot P_1\ Compat\ P_2 \forall P_1 : sprops(post\ ev_1), P_2 : sprops(post\ ev_2) \cdot P_1\ Compat\ P_2 t_1 < t_2 \wedge \nexists t_3 : dom\ b \cdot t_1 < t_3 < t_2 \Rightarrow sprops(post\ ev_1) = sprops(pre\ ev_2)\}$$

下面将依据上述行为模型，对协议工程的有关概念和性质进行形式描述和定义。

二、有关概念和性质的形式描述

2.1 系统的等价性 根据事件模型，如

果两个系统具有相同的行为集，那么这两个系统就是等价的。

$\_\approx\_ : System \leftrightarrow System$

$\forall s_1, s_2 : system \bullet s_1 \approx s_2 \Leftrightarrow s_1 = s_2$

以上所定义的关系≈与等式=是等价的，为统一起见，本文将使用≈。需要指出的是，≈适用于任意一对系统，只要它们具有相同的Time和Event集，而不管其时间和事件的表达方式如何。

2.2 非确定性 根据事件模型，可部分地描述所谓非确定性，其必要但非充分条件是：一个非确定性系统必须具有多个可能的行为。

Non Det Sys : p System

$\forall s$: Non Det Sys $\bullet \#s > 1$

2.3 多阶段系统 一个多阶段系统的每个行为都包含有大量不同的阶段。例如许多通信系统都提供的所谓连接服务就包含有三个阶段：建立连接，传输数据和断开连接。现有的多阶段系统描述一般是通过结构化分解来实现的。这种方法的基本思想是将一个系统分解成若干个实体（如通信有限状态机），每个实体又对应一个特定的阶段。本文对多阶段系统的描述主要是基于一个系统的行为，而不是结构化分解。多阶段行为可根据每个阶段的有关子行为来定义。这些子行为的性质可借助某些辅助信息来描述。例如，起始事件，结束事件和其它在本阶段可能发生的有关事件，以及这些事件可能发生的时间间隔等。为简便起见，本描述还包括与该阶段相关联的全部事件集。

Aux Phase Info

start-ev, end-ev : Event

other-evs, all-evs : P Event

start-tm, end-tm : Time

all-tms : P Time

---

start-ev $\notin$ other evs $\wedge$ end-ev $\notin$ other-evs

all-evs = {start-ev} $\cup$ {end-ev} $\cup$ other-evs

all-tms = {t : Time $\bullet$ start-tm $\leq$ t $\leq$ end-tm}

依据有关辅助阶段变量，可进一步定义一个阶段子行为的性质。

Phase

Aux Phase Info

sub-behav : Behav

start-tm $\in$ dom sub-behav $\wedge$ end-tm $\in$ dom sub-behav

$\forall (t, ev) : sub\text{-}behav \bullet$

$(t = start\text{-}tm \wedge ev = start\text{-}ev) \vee$

$(t = end\text{-}tm \wedge ev = end\text{-}ev) \vee$

$(start\text{-}tm < t < end\text{-}tm \wedge ev \in other\text{-}evs)$

现可依据阶段集定义一个多阶段行为。概括地说，该定义允许两个或多个阶段在时间上交错，并共用起始和结束事件。

Multi Phase Behav ==

{phaseset : P phase | $\forall P_1, P_2$: phaseset $\bullet$

$(P_1 \neq P_2 \wedge P_1.all\text{-}tms \cap P_2.all\text{-}tms \neq \phi) \Rightarrow$

$P_1.all\text{-}evs \cap P_2.other\text{-}evs = \phi$

$P_2.all\text{-}evs \cap P_1.other\text{-}evs = \phi$}

通过组合与各阶段相关联的子行为，我们可获得对应于（结构化）多阶段行为的（非结构化）行为。

behav-of : Multi Phase Behav $\rightarrow$ Behav

$\forall m$: Multi Phase Behav $\bullet$

behav-of(m) = $\bigcup$ {p : m $\bullet$ p.sub-behav}

2.4 输入前缀闭包 一个给定行为b的输入行为是对其输入事件行为的限制，即b$\triangleright$输入（其中$\triangleright$表示范围限制）。一个输入前缀闭包系统是一种特定的系统，即在这种系统中，每个输入行为的前缀均是其自身的一种输入行为。依据前述事件和输入/输出模型，可给出输入前缀闭包的形式定义，其中在时间t时行为的前缀可定义成受时间t_0所限制的一种行为，此时t_0小于t。

prefix : Behav $\times$ Time $\rightarrow$ Behav

$\forall b$: Behav, t : Time $\bullet$

prefix(b, t) = {t_0 : Time | $t_0 < t$} $\lhd$ b

Input Prefix Close Sys ==

{s : System | $\forall b_1$: s, t : Time $\bullet \exists b_2$: s $\bullet$

prefix($b_1 \triangleright$ Input, t) = $b_2 \triangleright$ Input}

2.5 因果关系 一个因果系统的输出事件与未来的输入事件无关。如果对于某个行为b_1，其在时刻t的输出行为不取决于在t过后所发生的输入，那么该系统将可以说成是因果的，即必定存在另外的某个行为b_2，其在时刻t具有与b_1相同的输出行为，且其全部输入行为与b_1输入行为的前缀等价。

Caus Sys==

{s: System| ∀b_1, s_1, t: Time·∃b_2:s·
prefix (b_1▷Input, t) =b_2▷Input∧
prefix(b_1▷Output, t)=prefix(b_2▷Output, t)}

2.6 运算兼容性 如果系统s_2的运算方式是系统s_1允许的，且导致的行为是s_1所期望的，那么则可以说s_2以一种运算兼容方式使s_1求精(记作$s_1 \sqsubseteq_{op} s_2$)。若上述断言成立，那么s_2必须接收s_1所接受的每个输入行为，并响应s_1所期望的输出行为(s_1的每个输出行为不必是s_2的一个可能的行为，亦即s_2可能比s_1更具有确定性)。

⊢⊑$_{op}$-:System⇔System

∀s_1, s_2:System·$s_1 \sqsubseteq_{op} s_2$⇔
∀ib_1:{b_1:s_1·b_1▷Input}·
∃b_2:s_2·i b_1=b_2▷Input
{b_1:s_1 |b_1▷Input=i b_1}⊇
{b_2:s_2| b_2▷Input=i b_1}

2.7 可达状态 在基于有限状态机(FSM)的协议形式化技术中，系统的可达状态能借助可达图或可达树来表征。一个系统的可达状态是该系统中某个行为能发生的状态。

Reachable:System→P State

∀s: System·Reachable(s)=
{ev:possev·pre ev}∪
{ev:possev·post ev}
其中possevs==(ran∪s)

2.8 安全性 安全性是协议工程中最重要的性质之一。我们说一个协议具有安全性是指它不会发生"坏"事情。一个安全状态也是指不会发生这类坏事情的状态。有些安全性可从系统的状态得到，即这些性质存在于该系统所有的可达状态之中。另外一些安全性则只能从某些历史状况的考察中获得。

State Safety Invariants,
Behav Safety Invariants:System→P property
∀s, System·State Safety Invariants(s)=
∩{st:Reachable(s)·sprops(st)}
∀s:System, P:property·
p∈Behav Safety Invariants(s)⇔
∀b:s, t:Time·p∈bprops(prefix(b, t))

2.9 活性 活性与安全性一样，也是协议工程中最重要的性质之一。目前大多数协议正确性验证技术，包括理论模型、方式或自动工具，仍只能限于验证协议的安全性，即只能检测出协议的某些错误，而并不能保证所设计的协议就一定正确。ISO和CCITT所标准化的三种FDT：Estelle、LOTOS和SDL也不能直接处理协议的活性性质。研究表明，时态逻辑可用于协议的活性验证，但它与其它的FDT如何组合使用才能取得最好的效果，目前仍不清楚。一个活性状态是指能使某些"好"事情有效地发生。形式化一点讲，活性性质是一种行为性质，对于每个完整的行为它必须为真，否则为假。活性性质实际上对于无限和有限系统均适用。关于这一点，目前国际上在认识上并不统一。有的意见认为它只适用于无限系统，并认为它包含有无限推理过程，不适用于实际系统的规范。也有的意见认为活性性质可为协议的某些特性提供高级的规范说明，减少实现错误。总的看来，活性验证比安全性验证更为困难，这主要表现在可用于活性验证的形式化技术要比可用于安全性验证的形式化技术少得多，活性验证的自动实现也比安全性验证困难得多。下面，依据前述行为模型对活性性质进行形式描述。

Liveness Invariants:System→P property

∀s:System;p:property·P∈Liveness Invariants(s)⇔
∀b:s·p∈bprops(b)
∃b:s;t:Time·p∉bprops(prefix (b,t))

以上，已利用前述行为模型对协议工程的有关概念和性质进行了形式描述和定义。下面将进一步利用前述行为模型研究某些典型FDT之间的语义关系。

三、FDT之间的语义关系

现有典型的FDT大多采用非形式或半形式化的语言来描述它们的语义，因而对于不同的协议工程开发者可能会带来不同的理解，难免造成歧义性[4]。FDT语义必须采用某种形式化方法进行描述和定义。前述行为模型可作为这种形式化方法的统一基础。下面着重讨论前述行为模型与某些典型FDT之间的语义关系。这里所考虑的FDT主要包括*Petri*网、*LOTOS*、关系表示法（*Relational Notation*）和Z。

3.1 Petri网 它可为一个系统的状态及其变化提供一种图形表达形式。这种表达形式实际上是由节点和弧所组成的一种有向图，其中节点又包含两种形式：圆圈和杠线。在圆圈中可含有标志（token），根据标志的标记（mark）方式，可将基本Petri网分成三类：一类是包含条件和事件的网；二类是带有单独标志的网；三类是包含位置和转移的网。除这三类基本Petri网外，还有若干改进型的变种，如分时Petri网、组合Petri网、标号Petri网、数字Petri网、随机Petri网以及评价网等。依据行为模型，其Event的概念可与Petri网中的转移相对应。Time则可表达成基于转移的时态（部分的）次序。这种次序可以通过Petri网的初始标记（标志的初始位置）和转移的输入/输出位置来定义。在Petri网中，Internal和External或Input和Output的概念没有明显的表达形式。State的概念可由Petri网的标记来表达。状态模型的行为集在形式上可与由Petri网语义所生成的可达树建立对应关系。由于Petri网的位置可导致多个转移（具有多条输入弧），所以它能描述非确定性系统，这种位置允许该系统展示多个可能的行为。

3.2 LOTOS 它是一种时态次序说明语言，该语言以通信演算系统（CCS）为基础，并引入了抽象数据类型(ADT)。它基本上由两部分组成：一部分是以进程代数方法为基础，可用于对进程行为和交互作用的描述；其二是以ADT为基础，适用于数据结构和数值的描述。在LOTOS中，一个分布式并发系统可看成是一个进程，该进程可包含多个子进程，子进程本身也是一个进程。由此可见，*LOTOS*描述一个系统的方法是一种由高层向低层逐级定义进程的方法。一个进程实际上是一个能完成内部不可见动作，并与其周围环境的其它进程进行交互作用的实体。进程间的交互作用以事件作为基本同步单元，复杂的交互作用则由若干原子交互作用（事件）组成。事件一般在进程的交互作用点(亦叫"门")上发生。在LOTOS中，进程也可看成是一个开有若干个门的黑盒，外界可通过这些门来观察该进程的对外行为，或对该进程进行操作。依据行为模型，Event的概念可与LOTOS中的门相对应。Time则可表达成基于门的时态次序。在LOTOS中，没有状态的明显概念，也正是由于这种高度抽象使得LOTOS特别适用于规范某些问题。它的运算语义为建立来自行为表达式的原子树，提供了一套公理和推理规则。一个原子树表达了一个特定进程的活动次序。这种次序在形式上与Internal/External和Input/Output组合模型的行为有一定对应关系。LOTOS还可用选择算子□来规范非确定性系统，该算子常与内部事件门i组合使用。LOTOS的并行组合算子 |[G]| （G是一个门表）允许事件不确定地交替发生，但不允许在同一Time值时发生。Internal的概念在LOTOS中可由事件门i表达。基本LOTOS中没有Input和Output的明显概念，但在改进的LOTOS中，则可由g? x:t和g!E的事件来表达这些概念。此时，g表示门标识符，x表示值标识符，t表示类标识符，E表示值表达式。

3.3 关系表示法 它是一种描述状态转

移系统的抽象形式方法，其主要基础是谓词逻辑和线性时间时态逻辑。前者可用来描述状态及其转移，后者可用来描述转移条件。在关系表示法中，系统的安全性与活性可通过抽象状态变量来描述，并以此为该系统建模。此外，系统的建模也可通过将系统分解成若干个实体来实现。每个实体又可通过一系列状态变量、事件及其初始条件来规范说明。依据行为模型，Event的概念与关系表示法中的运算相关联。Time可由基于运算的时态次序来表达，也可通过引入表达时间的变量来表达。关系表示法中没有内部和外部的明显概念。行为模型中输入和输出的概念与该表示法中的信道相关联。关系表示法也可用来规范非确定性系统，但它也存在与Petri网同样的限制，即无法区别外界选择和内部的非确定性。

3.4 Z表示法[2] 这种方法是一种基于集合论和一阶谓词逻辑的形式描述技术。与关系表示法相类似，Z表示法也可归于抽象模型法这一类。它与关系表示法的主要差别是：Z为了实现命名分组和建立各描述部分的关系，而使用了图解演算(schema calculus)技术。Z的数据类型以集合为基础，它提供了大量的标准类型(如自然数)。其中说明部分的n:N表示在一系列自然数中，n将有一个具体值。Z还允许引入新的类型，或根据原先的类型来定义新的数据类型。为了便于处理象关系、函数和顺序之类的类型，Z还提供了一些特殊的记法。其中关系可按集合的有序对来说明，函数可看成是一种特殊的关系，顺序又可作为一种特殊的函数。研究表明，Z非常适于描述状态转移系统，同时也可支持其它风格的形式描述。依据行为模型，Event概念与Z的运算有对应关系。Time可由基于运算的时态次序来表征，也可由表达时间的变量来表示。输入/输出的概念与运算图解的输入/输出参数相关联。非确定性系统的建模特征与上述Petri网和关系表示法相类似。Z经适当扩充还可用于描述协议的活性。

四、结束语

协议工程作为一门新兴的学科，仍处在不断发展和完善之中，有许多问题尚待进一步探讨。这主要表现在它的有关概念和性质往往采用非形式化的语言进行定义和描述，缺乏应有的形式化基础，难免造成歧义性；此外就现有的FDT而言，其中也包括被ISO和CCITT标准化的三种FDT，它们都各自有其语义基础，形式描述风格各异，不同FDT之间的语义关系难以建立或不清楚，FDT的评价与综合缺乏形式基础，难以形式化地确定哪一种FDT最适用于解决哪一类问题，当要扩充或组合现有的FDT，或希望开发新型FDT时，缺乏规范的指导方法，如此等等。本文已描述了一种适应于协议工程的行为模型，它采用抽象的层次式结构，可包含通信系统或协议的主要行为。研究表明，它可用于协议工程有关概念和性质的形式描述和定义，并可为研究典型FDT之间的语义关系提供形式化基础。

参考文献

[1] Abrial, J.R., The Specification Language Z, Tech. report, PRG, Oxford Univ., UK, 1980

[2] Spivey, J.M., The Z notation: a reference manual, Prentice Hall, 1989

[3] Li Layuan, A formal technique for communication protocol specification, Proc. IEEE INFOCOM, 1989

[4] 李腊元等，计算机局部网络，湖北科技出版社，1987