并发系统模型 软件开发 数学模型

计算机科学1996Vol.23№.5

并发系统模型研究

贾国平 郑国梁
(南京大学计算机科学系 南京210093)

摘 要 In this paper, we present discussion and analysis of various existing models of concurrent systems, such as finite-state machine, hierarchy of functions, Statecharts, Petri-Net, graph model of computation and fair transition system model. As a result of discussion, we know that different properties are expressed by different models of requirements. Our work is very helpful for software developers, especially for systems analysts to choose proper models in their problems and it provides the foundation for further applications of various models.

关键词 Model, Concurrent system, Finite-state machine, Hierarchy of functions, Statecharts, Petri-Net, Graph model of computation, Fair transition system.

1. 引 言

一个模型的作用是对想要构造或分析的系统性质给出严格定义，同时也为验证这些性质提供一个基础。现有的用于软件开发的每一种形式化描述和分析方法其有效性范围均是相对于程序执行的某一数学模型而言，并不是实际的系统。尽管这些方法对于其模型是完全有效的，但它们对于程序实际的应用却只能由其对应模型表示实际程序执行的可信度来决定。通过一种形式化方法所得到的关于程序执行的结论，在某种程度上，只有当其基本的数学模型是可靠的，它才是可靠的。因此在开发形式化方法过程中，很重要的一点是它不仅应该包括构造新的模型和用于分析此模型的技术，而且还应该包括不断对模型之间以及它们与实际系统之间进行比较和评价。

我们对不同的并发系统模型进行了分析和讨论。不同于顺序系统模型，并发系统模型除了表示顺序程序的所有特征之外，很重要的一点是它还必须很好地表达并发性。文中我们就目前普遍用于说明并发系统性质的不同模型，如有穷状态机、函数层次模型、Statecharts、Petri-网、计算图模型及公平转换系统模型等的优点及不足进行了分析。由此分析和讨论，我们知道不同的性质可以由不同的需求模型表示。为了表达并发系统的特殊性质，我们必须要选择合适的模型加以表示。本文的工作有利于软件开发人员，特别是系统分析人员选择合适的模型以便于更好地对不同问题进行说明描述。它为不同模型的进一步应用提供了基础。

2. 并发系统模型

2.1 有穷状态机模型

有穷状态机模型(FSM)是计算机科学中的一个最基本的模型，已经广泛应用于通信协议的规范及验证之中，同时也是几乎所有证明技术及系统分析技术的核心[1]。

一般，一个 FSM 模型由这样几个部分组成：①Input：输入集合。②Output：输出集合。③S：状态集合。④s_0：初始状态。⑤T_{state}：状态转换函数：S×Input→S。⑥T_{transf}：变换函数：Input→Output。

这一模型非常适用于对数据处理问题的描述。数据处理的输入及输出分别对应于 FSM 中的 Input 和 Output。处理器内存及寄存器内容与 FSM 中状态相联，而执行代码对应 FSM 中的变换函数。

然而，FSM 用于说明并行处理时有两个主要的缺陷：其一，FSM 模型本质上将所有基本的并发进行了顺序化处理。模型显式地假设在下一输入到达之前，前一输入点上的所有处理都已完成。因此，当此模型用于并发系统的规范时，其关键问题是它忽略了冲突状态，即两个输入同时到达的状态。这也就导致了一个不合适模型上的正确性证明。在这方面，现在已出现了许多工作，它们主要通过采用多个并

贾国平 博士生。 郑国梁 教授、博士生导师。

行FSM对FSM模型进行扩充用以对并发性进行描述。其二，FSM模型本身并无一个机制用于处理复杂性。此模型是“平坦”的，并无一个层次，当所要描述的进程的状态个数过大时，如果没有一个辅助结构，FSM描述将很难理解。即使对于小型问题，FSM中对引起状态改变的状态转换函数及变换函数的描述也非常复杂。因此，一些机制需要引入FSM模型中以便将两个函数分解为更小、更易于理解的单元，易于实际构造。

2.2 函数层次模型

函数层次模型是数据处理中最常用的一种说明方法，在说明系统时，首先要说明一个系统函数，此函数有一个由所有可能输入数据组成的输入论域和一个由所有可能输出数据组成的输出论域。然后，此系统函数分解成一个相互交互的函数集合，它们之间具有输入/输出关系，而每一交互函数又可逐个被进一步分解为更低一级相互交互的函数。分解过程中，数据和函数都同时被分解。函数层次模型是许多软件规范技术(如SADT和PSL/PSA)的基础。

函数层次模型最大的优点在于它对大量数据的组织能力以及它对高级函数同它分解而成的众多低级交互函数之间相容性进行检查的能力。这一过程可以重复下去直到得到一个任意级的函数层次。这就解决了FSM模型中无法表达的并发及复杂性问题。

然而，函数层次模型也有其不足，主要是它只表示了数据流，而对控制流并未进行表示，特别是，执行顺序和条件信息并未在此模型中说明并且也不可能用此模型进行验证。这就限制了函数层次模型的作用，同时缺乏对并发性更细致的表示能力。

2.3 Statecharts模型

Statecharts模型是由D. Harel[1987]提出的用于描述复杂并发系统(及反应系统)的一种可视化形式。它通过状态、事件和条件对系统行为进行描述。而后两者的组合形成转换。从而导致状态之间的转换。

Statecharts模型本质上可看成是对传统的FSM模型及状态图(State Diagram)模型的推广，是Moore有穷状态自动机和Mealy有穷状态自动机的某种组合。此模型中的允许序列对应于由自动机所接受的语言。另外，Statecharts模型完全遵循了标准状态图的可视化表示。

具体地说，Statecharts模型从三个方面对传统的FSM模型及状态图模型进行了扩充。首先，此模型中采用“与/或”(AND/OR)状态分解/聚集机制。状态可以反复使用“与/或”状态聚集机制而被组合成高级状态。同样，高级状态可以通过反复使用“与/或”分解机制而被精化为低级状态。模型中的转换并不受状态级的限制，它可以从任何聚集级状态到其它任何聚集级状态。事实上，转换一般是从一个格局(Configuration)到另一个格局，因此模型获得了层次描述能力。其次，Statecharts模型采用“与”状态分解机制。如果系统处于某一状态，则通过使用“与”状态分解机制后，系统必须处于所有分解得到的“与”部件状态之中。此机制获得了并发系统中部件之间的并发性描述。最后，Statecharts模型中对并发部件之间采用广播通信机制，更加易于实际描述。这几方面的扩充使得Statecharts模型成为一种高度结构化、简洁且非常适用于复杂并发系统的模型，目前已经得到了较广泛的应用，尤其适用于复杂反应系统的规范及验证。

2.4 Petri-网模型

Petri-网模型[2,3]提供了一种形式表示顺序和并发以及标识和解决歧义性的方法。它是通信协议的规范及验证中的一个有效工具。

在Petri-网模型中，我们区分两类节点：位置节点和网转换节点。形式上，一个Petri-网模型是一个三元组：(P,T,F)，其中：P是位置节点集合，每一位置节点定义有向图中标记可以进驻的位置。T是网转换节点集合，每一网转换节点控制前一位置节点中的标记向下一位置节点的转换。如果网转换节点的条件满足，则此网转换就被执行。$F \subseteq (P \times T) \cup (T \times P)$是连接边集合。对一个简单的Petri-网中的网转换，当其前置条件满足时，此网转换被称为点火(Firing)，上一位置节点中的标记迁移至下一位置节点。

Petri-网模型能够用于精确地说明顺序、并发和同步等概念。另外，通过定义一个Petri-网处理器。我们可以利用Petri-网进行自动模拟。这个处理器检查所有网转换的状态，选择一满足输入条件的位置节点，然后根据后置条件移动标记。此步骤可以重复下去，直到不存在满足前置条件的网转换为止。系统分析人员通过观察标记的活动来对所描述进程的行为进行分析。并发系统中许多有趣的性质，如正确性终止、无死锁性、互斥等问题都可以用Petri-网得到定义和解决。

尽管Petri-网模型有上述优点，但用于描述并发系统存在两个缺陷。首先，它与FSM模型一样是平坦的，即Petri-网本质上只有一层，并无一个层次结构。许多研究工作中已经提出了Petri-子网的概念。子网层次的概念可以较好地处理大型复杂问题。这一方法使用得合适，并不会影响Petri-网本身的

语义。其次，Petri-网模型只说明系统的控制流而对数据流并未予以说明。即使网转换条件由数据值来表示，改变和存储数据变量值的语义仍然并不属于Petri-网内在的一部分。对此问题，目前也有不少的研究工作，如文[4]。

2.5 计算图模型

计算图模型是一个带节点及弧的有向图，事实上，它是Petri-网模型的一种扩充，其中转换和位置被有效地组合成某一节点类型，模型中每一节点表示一个处理步。它可以有一个或多个输入弧，也可以有一个或多个输出弧。为了进行转换，一个输入函数用来定义说明是否所有或者只有一个输入弧是能行的。在转换完成点，一个结束函数说明是否所有或者只有一个输出弧是能行的。它包含类似于Petri-网模型中关于转换的说明。然而，不同于Petri-网模型的是在计算图模型中每一节点都有一个从某一论域中定义的输入和输出，它用于说明节点之间的数据流。因此，计算图模型不仅说明了控制流，同时它还说明了数据流，它为两者的相容性检测提供了基础。在计算图模型中，由于其控制流是用与Petri-网模型相同的形式表示的，因此许多性质，如正确性终止、无死锁性等也能够在此模型中进行分析。另外，Petri-网模型中子网的概念同样也可适用于计算图模型，它为计算图模型提供了一个层次分解机制。

计算图模型最大的缺陷就是：虽然它可以很好地用于描述数学函数的分解，但是本质上并不适用于对时间序列转换的描述，因为在此模型中并无状态的概念。因此计算图模型并不非常适用并发系统的描述。

2.6 公平转换系统模型

公平转换系统模型(FTS)最初由Z. Manna和A. Pnueli[1983]给出。它一方面可以作为并发系统的抽象计算模型，另一方面也可以作为一种抽象的实现语言。此模型包含了许多具有不同语法表示及通信机制的并发系统，它与许多具体程序设计语言之间的对应关系可参见文[5]。

一个公平转换系统S是一个六元组$(V,\Sigma,\Theta,T,WF,ST)$，其中，$V$：状态变量的有穷集合。$\Sigma$：状态集合。$\Theta$：初始条件。对一个状态$s\in\Sigma$，如果它满足$\Theta$，即$s\models\Theta$，则我们称$s$为初始状态。$T$：有穷转换集合。每一转换$\tau\in T$是一函数：$\Sigma\to 2^{\Sigma}$。$WF\subseteq T$：弱公平性转换集。$SF\subseteq T$：强公平性转换集。

一个转换在状态s下是能行的如果$\tau(s)\neq\varnothing$，否则τ在此状态下是不能行的。

Σ中的无穷状态序列$\sigma:s_0,s_1,\cdots$被称为是公平转换系统S的一个计算(Computation)，如果σ满足：①初始化条件：s_0是初始状态，即$s_0\models\Theta$。②连续性：对每一$j=0,1,\cdots$，状态s_{j+1}是状态s_j的τ-后继状态，即存在转换$\tau\in T$，使$s_{j+1}\in\tau(s_j)$。③弱公平性：对每一转换$\tau\in WF$，不会发生τ在σ中某一位置以后一直能行，但只有有限多次被执行。④强公平性：对每一转换$\tau\in SF$，不会发生τ在σ中无穷多次能行，但只有有限多次被执行。

由上可知，FTS模型本质上也是FSM模型的扩充，其中很重要的一点就是此模型中并发性是通过交替执行(Interleaving)来表示的。两个并行进程永远不会在同一时间执行它们的语句，只会交替执行其原子转换。形式上，当一个并行进程正在执行其原子转换时，其它进程中的原子转换都处于非激活状态。FTS模型中通过对转换进行不同的公平性限制而确保并发程序执行中，所有进程都参与了执行，避免了某些计算中只有一些进程被执行，而其它进程都没有机会被执行的不公平情形，从而解决了实际并发程序执行中进程间独立执行问题。在这样一个并发性交替表示模型中，并发系统规范及验证的理论就非常简单。另外很重要的一点是FTS模型为时序逻辑很好地应用于并发系统的规范及验证提供了基础。

然而，上述FTS模型中有一个很重要的干扰(Interference)问题，FTS模型本质上是一个交替模型，要求在一个转换执行过程中无干扰发生。为了使FTS模型能够完全描述并发系统的实际执行，必须要对FTS模型中原子转换及其粒度划分施加限制。文[6]中，通过对程序原子执行语句(即原子转换)进行限制临界引用约束，即LCR条件，而对干扰问题进行了处理。由此约束条件可以得到结论：FTS模型是可信的，它能够完全对并发系统的实际执行进行描述。

主要参考文献

[1] T. F. Piatkowski, The State of the Art in Protocol Engineering, In: Proc. ACM Sigcomm.' 86 Symp., Stowe, VT, 1986

[2] J. Peterson, Petri Net Theory and the Modeling of Systems, Prentice Hall, 1981

[3] N. Reisig, Petri Nets, EATCS Monographs, 1985

[4] S. Yau, M. Caglayan, Distributed software Design Representation Using Modified Petri Nets. Trans. on Software Engineering, 1983

[5] Z. Manna, A. Pnueli, The Temporal Logic of Reactive and Concurrent System: Specification, New York: Springer-Verlag, 1991

[6] 贾国平、郑国梁，论公平转换系统模型的可信性，《计算机科学》1996年，第23卷，第2期