软件开发 软件规范化 软件工程

计算机科学 1996 Vol. 23 №. 4

56-60

软件规范方法比较

TP311.52

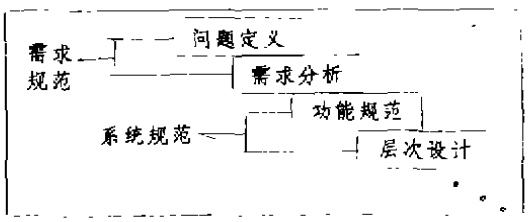
贾国平 郑国梁
(南京大学计算机科学系 南京210093)

摘 要 Most existing specification methods can be roughly classified into two styles: one is called logic-based approach. Its representative is temporal logic approach. The other is model-based approach. Its representative is state machine approach. In this paper, we compare these two styles of specification methods by the software engineering principles. At the end of the paper, we briefly introduce our fair transition system specification (FTSS) approach, which combines the best features of temporal logic and state machine methods and is easy to understand and to use.

关键词 Specification, Concurrent system, Logic-based specification, Model-based specification, Fair transition system specification.

1 引 言

软件工程是应用计算机科学、数学及管理科学等原理，借鉴工程化原则与方法，解决软件问题的工程。软件开发过程可划分为若干个从概念直到实现的阶段。一个典型的阶段划分序列包括：问题定义、需求分析、(功能)规范、(层次)设计、详细设计、实现、测试、维护和退役等。各阶段的划分并非固定不变，且其间无一个严格界限。称为"规范"的这个领域可以被划分为更加细致的一个层次，包括从问题定义到层次(总体)设计这几个阶段。一般按规范所包括的范围，将规范分为两种类型：一种是需求规范，粗略地对应于上述阶段划分中的问题定义和需求分析两个阶段；另一种是系统规范，粗略地对应于上述划分中的功能规范和层次设计两个阶段(如下图)。



在软件开发的最高级也是最初一级，我们须考虑需求的规范，它应该被看作是使用系统的用户和系统的实现者之间合同的基础。一般说，需求规范列出系统应该满足的性质(需求)集。在系统设计初级阶段，人们常常一开始是先写一个非形式的自然语言需求文档，列出所希望设计的系统中主要的性质及功能。一个需求规范可以被视为这个文档的一个形式表示，常常将一段段文档转换成适当逻辑语言中的公式。需求规范的一个很重要的特点是它主要集中于系统可观测行为的"做什么"上，应该忽略(或至少应避免)"如何做"这个问题。需求规范很少去考虑系统的组织和实现。

软件开发中，基于需求规范，在与用户制定了合同之后，开发小组开始一起设计系统规范，其设计应该确保满足用户需求。一般，系统规范应该提供系统的一个抽象模型。在每一个特定的状态下，这个模型应该能够完全预测出系统对将要发生的事件的可能反应。

在系统规范中，我们必须考虑在需求规范阶段两个有意忽略的问题：一是系统的高级组织和层次，这就等于将顶层系统分解为子系统或子行为。二是必须对不同需求之间的交互进行一系列的研究。

由上讨论，我们可以看出需求规范和系统规范之间的两个主要不同之处：(1)需求规范应该有一个合取特性，即需求规范应该包含一个需求的合取式。每当发现漏去了一个需求时，我们可以简单地将遗漏的需求加入到合取式中。(2)需求规范很少显式地说明并发性和非确定性，常常将它们视为实现问题

贾国平 博士生，郑国梁 教授、博士生导师。

而留给实现者在实现时自行裁决。因此，一个分布式系统的需求规范同一个集中式系统的需求规范并无什么不同，它们之间本质的区别在于其最终实现。然而对于系统规范情况却不相同。并发性和非确定性在系统规范中一般要显式说明。

软件开发的第一步是首先一个系统的规范可以看做是程序模块的实现者和这个模块的使用者之间的一个合同，其作用是：首先它允许实现者去设计满足规范所规定的需求的模块且据此对模块正确性进行验证。其次，它允许使用者根据此规范去书写调用此模块的程序且据此对程序的正确性进行验证。因此，一个规范很重要的一点就是不仅必须很容易地满足上述两个目的，而且要易于书写和易于理解。

不同于顺序程序，并发程序一般并不能够由简单的输入/输出关系来说明描述，必须描述并发程序的行为。并发系统本质上是复杂的，我们也常常为其与时间有关的同步错误所困扰，只有严格的、形式化的推理才是减少并发系统错误的最好办法，因此对并发程序的规范方法的研究至关重要。现在已出现了许多用于说明并发程序的规范方法，如时序逻辑方法[1,2]、状态自动机[3]、通信系统演算(CCS)[4]，UNITY[5]，Statecharts[6]及 LOTOS[7]等。这些方法各有其特点。

现有的大多数用于并发系统的规范方法一般可分为两类：一类是基于逻辑的方法，通常给出系统应该满足的性质集合，通常说明程序应该有的性质来描述一个程序，是一种基于程序逻辑的方法，其中每一个性质都可以视为某一逻辑系统中的一个公理，所以本质上是一种公理化方法，其代表是时序逻辑方法。另一类是基于模型的方法，一般给出系统的一个抽象模型，说明系统的可能行为。它是一种基于抽象程序或自动机的方法，一般有一个类似于程序设计语言的语法。此方法本质上是一种构造性方法。CCS、UNITY、IO-自动机、Statecharts 及 LOTOS 等方法都属于这类方法。基于模型方法的代表是状态自动机方法。下面，我们从软件工程原理，对这两类规范方法进行比较和讨论。

2 基于逻辑规范与基于模型规范

我们称由基于逻辑方法得到的规范为基于逻辑规范，而称由基于模型方法得到的规范为基于模型规范，我们给出准则对这两类规范方法进行比较和讨论。准则可以有多种，下面我们从软件工程原理出发，找出几种比较准则，看看每一类规范方法是如何满足这些比较准则的。需要指出的是，我们对两类规范方法的对比，并不是想评价出哪一种方法更好，而只是想进一步总结出两类规范方法的不同，以便从中得出有用的结论。

2.1 增量式修改

增量式修改是指每当我们认识到一个需求被遗漏或需求被修改时，我们去改变规范的难易程度。在理想情况下，一个规范一经写成，其后，实现者和用户之间不再需要讨论。然而，实际中，如同每一个实际的合同，每当实现者认识到规范中有许多他无法实现的需求或用户认识到所说明的模块并不满足他的需要时，规范必须要经过再讨论，因此增量式修改是软件开发过程中一个很重要的性质，我们必须予以考虑。

首先看看基于逻辑规范方法。基于逻辑规范一般是一个性质的表或性质的合取式，因此当需要对现存的需求进行修改时，我们很容易找出与之相关的合取项；而当需要一个新的需求时，也很容易将它加入到合取式当中。

对比基于逻辑规范方法，基于模型规范进行这样的修改则常常比较困难，因为这种方法一般将所有需求集成为一个抽象程序，并根据进程或模块来结构化，而不是根据性质。对规范所需要的改变，常常并不能由整个规范局部化到单个性质上去。如一个用状态自动机说明的系统，当需要加入一个新的需求时，我们不能简单地将此新需求作为一个合取项加入，而必须要考虑原先需求同新加入需求之间的交互以重新构造整个规范。

2.2 相容性

相容性的含义是一规范定义了一个可能实现的非空集合，即至少存在一个系统满足规范。这是对一个规范方法的基本要求。

在基于逻辑规范方法中，不相容规范的可能性是存在的[8]。在此规范过程中每一个被形式化的需求之间几乎是相互独立的，因此很容易不小心加入两个相互矛盾的需求或者得到一个不能同时满足的更大的需求集。

在基于模型规范方法中，相容性总是可以保证的，因为每一个抽象程序至少有一个执行，也即有一个非空语义。例如每一个非空的状态自动机都有一个类似进程的含义。

2.3 完全性

完全性的含义是系统的所有重要的性质都已经说明，没有性质被遗漏。在软件开发过程中，虽然一

个完全的规范在实际中不可能一次得到，一般要经过多次反复，但是得到完全的规范越早，软件开发中的花费就越少。因此完全性也是软件工程中一个很重要的准则。

基于逻辑规范方法，其完全性常常是困难的，因为此方法通常是通过每一时刻加入一个需求这样的过程来对软件进行规范的，且并无一个准则去判定何时这个过程终止。文[1]中通过将程序的性质进行安全性-活性分类(Safety-Progress)来获得用时序逻辑方法得到的规范的完全性。对一般情形完全性仍然是一个问题。

基于模型规范方法，其完全性并不是一个很大的问题。此方法中不完全性只会出现在当某一整个进程被遗漏或者当从某一个给定状态对可能活动集进行扩充时某些活动被遗漏的情形。

2.4 有效性

有效性是确保规范与设计者的意图相一致的过程。一般设计者内心的想法并无一个形式表示，因此有效性不是一个形式过程，但在软件开发中却至关重要，我们必须予以着重考虑。

在基于逻辑规范方法中，常常通过非形式的需求文档来表示设计者的意图，尔后设法形式化这个自然语言文档，因此我们能够以模块形式实现有效性过程。即使我们并无一个写好的需求文档，但也能对每一个需求分别确定有效性。我们可以确定是否设计者或用户同意形式需求的整个含义。

对比基于逻辑规范方法，基于模型规范方法之有效性的确定要困难得多。一般，一个基于模型规范是一个较大的抽象程序，我们必须去判定是否此抽象程序所产生的所有可能行为都与我们直观上系统应该有的行为相一致。这个过程非常类似于证明一个给定程序是正确的这个问题。显然，这个问题如果没有一个系统的方法是非常困难的。

3 公平转换系统规范

基于上述对两类规范方法的比较和讨论，我们看出，作为基于逻辑方法的代表，时序逻辑方法本质上只适用于需求规范，而作为基于模型方法的代表，状态自动机方法本质上只适用于系统规范。这就为两种规范方法的进一步应用带来了不利。这里，简要介绍我们的公平转换系统规范方法(FTSS 方法)。此规范方法集成了时序逻辑方法和状态自动机方法各自的优点，易于理解便于使用，我们已成功地将 FTSS 用于程序验证当中。关于 FTSS 及其应用的详细说明可参见文[9]。

3.1 基本模型

我们给出一般的公平转换系统模型，作为一种抽象的实现语言，并给出与此模型有关的一些基本概念。关于此基本模型的可靠性及其与许多具体程序设计语言之间的对应关系可参见文[10]和[1]。为了表达公平转换系统的语法，我们用了一个基本的一阶语言，其中的公式称为断言。

一个公平转换系统 S 是一个六元组(V，Σ，Θ，T，WF，SF)，其中：$V=\{u_1,\cdots,u_k\}$：状态变量的有穷集合；Σ：状态集合；Θ：初始条件；T：有穷转换集合；$WF\subseteq T$：弱公平性转换集；$SF\subseteq T$：强公平性转换集。

每一状态 $s\in\Sigma$ 是 V 的一个解释，它给每一变量 $u\in V$ 赋给其对应论域中的一个值，用 s[u]表示。对一个状态 $s\in\Sigma$，如果它满足 Θ，即 $s\models\Theta$，则我们称 s 为初始状态。每一转换 $\tau\in T$ 是一函数：$\Sigma\to 2^{\Sigma}$，它将每个状态 $s\in\Sigma$ 映入(可能空)τ-后继状态集合 $\tau(s)\subseteq\Sigma$。一个转换在状态 s 下是能行的当且仅当 $\tau(s)\neq\varnothing$，否则 τ 在此状态下是不能行的。我们在 T 中加入一个转换 τ_I，称为空转换，它是一个单位转换，即对每一状态 $s\in\Sigma$，$\tau_I(s)=\{s\}$。每一转换 τ 都由一个断言来表示，记为 $\rho_\tau(V,V')$，称为转换关系。它将状态 $s\in\Sigma$ 与它的 τ-后继状态 $s'\in\tau(s)$ 通过无上撇号和有上撇号的状态变量联系起来。无上撇号的状态变量引用 s 中的值，有上撇号的状态变量引用 s′中的值。我们说转换关系 $\rho_\tau(V,V')$ 标识出 s′是 s 的一个 τ-后继，如果 $(s,s')\models\rho_\tau(V,V')$，其中(s，s′)是联合解释，它将 $x\in V$ 解释为 s[x]，x′解释为 s′[x]。

我们称 Σ 中的无穷状态序列 $\sigma:s_0,s_1,\cdots$ 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。

对一个公平转换系统 S，我们记 Comp(S)为系统 S 的所有计算的集合。

3.2 公平转换系统规范

作为具体的规范描述语言，我们对变量集合 V，定义一对应的一阶时序逻辑，其句子解释为状态序列的性质。我们的时序逻辑语言在语法及语义上与其它的离散线性时间时序逻辑相类似(见文[1])。除

了包含一般逻辑算子，如布尔联接词～，∧，∨，≡，→；等式算子＝以及一阶量词∃，∀ 等外，它还包含两个时序算子：O(下一值)，U(直到)，其中算子 O 既可作用于公式，也可作用于项。作用于项时，我们记 $Ot \equiv t^+$，称为项 t 的下一值。其它时序算子我们作为简写，如：$\Diamond p \Leftrightarrow trueUp$，$\Box p \Leftrightarrow \sim\Diamond\sim p$，$pWq \Leftrightarrow \Box p \vee (pUq)$ 以及 $p \Rightarrow q \Leftrightarrow \Box(p \rightarrow q)$.

不包含任何时序算子(包括施用于项的下一值算子)的公式称为断言。

需要注意的是，变量集合 V 分为两部分：一部分为可变变量或程序变量，它不允许量词约束；另一部分为固定变量或逻辑变量，它只用于规范目的，可以由量词约束。

令一个公平转换系统 S 为 $(V, \Sigma, \Theta, T, WF, SF)$。首先我们引入一些公式，它们分别表达了 S 的计算的不同性质。

·$En(\tau)$：$(\exists V')\rho_\tau(V, V')$，其含义指转换关系为 $\rho_\tau(V, V')$ 的转换 τ 是能行的。

·$taken(\tau)$：$\rho_\tau(V, V^+)$，其中 $\rho_\tau(V, V^+)$ 通过将 $\rho_\tau(V, V')$ 中每一变量 y′ 由 y^+ 代替而得。假设 $taken(\tau)$ 在一个计算的 j 位置成立，则上述公式的含义指如果我们取 $y \in V$ 为状态 s_j 中的值，即 $s_j[y]$，y′ 为值 $val(\sigma, j, y^+) = val(\sigma, j+1, y) = s_{j+1}[y]$，那么关系 ρ_τ 成立，即 s_{j+1} 是 s_j 的 τ-后继。

·$wf(\tau)$：$\Diamond\Box En(\tau) \rightarrow \Box\Diamond taken(\tau)$。此公式说明，如果 τ 除了有限多个位置外均能行，则 τ 被执行无穷多次。因此，任何满足 $wf(\tau)$ 的计算序列对于转换 τ 是满足弱公平性的。

·$sf(\tau)$：$\Box\Diamond En(\tau) \rightarrow \Box\Diamond taken(\tau)$。此公式说明，如果 τ 无穷多次能行，则 τ 被执行无穷多次。因此，任何满足 $sf(\tau)$ 的计算序列对于转换 τ 是满足强公平性的。

对一给定的系统 S，我们定义其时序语义公式 Sem(S) 为：

$$Sem(S): \Theta \wedge \Box \bigvee_{\tau \in T} taken(\tau) \wedge \bigwedge_{\tau \in WF} wf(\tau) \wedge \bigwedge_{\tau \in SF} sf(\tau)$$

下面我们考虑时序语义公式 Sem(S) 中的每一项：

·合取项 Θ 确保系统 S 的计算序列中的初始状态满足初始条件 Θ。

·$\Box \bigvee_{\tau \in T} taken(\tau)$ 保证对每一 $i = 0, 1, \cdots$，状态 s_{i+1} 是在状态 s_i 上通过施用某一转换 $\tau \in T$ 而得到的。

·$\bigwedge_{\tau \in WF} wf(\tau)$ 保证系统 S 的计算序列满足所有的弱公平性假设。

·$\bigwedge_{\tau \in SF} sf(\tau)$ 保证系统 S 的计算序列满足所有的强公平性假设。

对比公平转换系统 S 中计算的定义，我们知道公式 Sem(S) 中的上述四项分别对应且保证定义中的四个要求，因此公式 Sem(S) 精确地表示了系统 S 的计算。

在给出我们的公平转换系统规范之前，先给出几个定义。

定义1 一个转换模块 M 是一个系统 $M: (V, U, \Sigma, \Theta, T, WF, SF)$，其中 $S_M: (V, \Sigma, \Theta, T, WF, SF)$ 是一个公平转换系统，$U \subseteq V$ 是一个集合。我们称 S_M 是 M 的体，U 为 M 的隐藏变量集。

定义2 一个体为 S_M，隐藏变量集为 U 的转换模块 M，它的一个执行是 S_M 的一个计算的任何 U-变体，即与 S_M 的一个计算相比至多对 U 中变量解释不同的任何模型。

下面给出我们的公平转换系统规范形式：

定义3 我们称一个时序公式具有公平转换系统规范形式，如果它有形式：$\exists U. Sem(S)$，其中 S 是一公平转换系统，其状态变量集为 V，Sem(S) 为 S 的时序语义公式，U 为 V 的一个子集。

由此定义可知，对于某一转换模块 M，其中 $S_M: (V, \Sigma, \Theta, T, WF, SF)$ 是一公平转换系统，$U \subseteq V$ 为其隐藏变量集合，我们的公平转换系统规范具有下述形式：

$$\exists U. [\Theta \wedge \Box \bigvee_{\tau \in T} taken(\tau) \wedge \bigwedge_{\tau \in WF} wf(\tau) \wedge \bigwedge_{\tau \in SF} sf(\tau)] \qquad (*)$$

其中：

Θ 是一初始状态断言，说明了转换模块 M 的初始状态，即说明了变量的初始值。

$\Box \bigvee_{\tau \in T} taken(\tau)$ 是转换模块 M 的下一状态关系，即对每一 $i = 0, 1, \cdots$，状态 s_{i+1} 由状态 s_i 通过执行某些转换 $\tau \in T$ 而得到。转换 τ 的每一步对应转换模块 M 的相应程序中单个原子操作的执行。

$\bigwedge_{\tau \in WF} wf(\tau)$ 及 $\bigwedge_{\tau \in SF} sf(\tau)$ 确保转换模块 M 的执行序列分别满足所有弱公平性和强公平性假设。

U 是转换模块 M 的隐藏变量集合或称为内部变量集合。

显然，我们的公平转换系统规范(*)精确地表示了转换模块 M 的执行，即一个模型 σ 满足(*)当且仅当 σ 是 M 的一个执行。

记 $L = \bigwedge_{\tau \in WF} wf(\tau) \wedge \bigwedge_{\tau \in SF} sf(\tau)$，我们的公平转换系统规范(*)有如下解释：

存在某种选取隐藏变量集 U 的方式，使得：(1)谓词 Θ 在初始状态为真。(2)系统 M 的每一步执行或者是某一转换 $\tau\in(T\setminus\{\tau_l\})$ 的执行 taken(τ)或者是单位转换 τ_l 的执行 taken(τ_l)，此时系统 M 的所有变量保持不变。(3)系统 M 的整个行为满足 L，即满足所有弱公平性和强公平性假设。

4 对比评价

对比其它规范方法，我们的方法有下述特性：

·规范的简明性。一个形式规范必须易于阅读及理解其含义。基于逻辑规范，尤其是时序逻辑规范是难于理解的，特别是对那些不熟悉逻辑的人更难。我们的方法给出了一种书写形式规范的途径，它集成了时序逻辑方法和状态自动机方法各自的优点，因此非常易于理解及使用。

·规范的抽象性。一个形式规范应该说明系统"做什么"，而不应该说明系统"如何做"，它不应对系统的具体实现进行约束。基于模型规范方法过多地考虑了系统的组织和层次，因此其规范非常难于修改。我们的规范方法采用了作用于内部变量上的存在量词。内部变量如何实现并不重要，它们对于实现者是完全自由的。存在量词作用于这些内部变量就保证了它们是完全自由于实现的。这正如程序设计语言中隐藏(hiding)的概念。

·语义的简明性。规范形式的简明性常常是表面的，一个形式规范真正的简明性要求其规范语言的形式语义简单，对其形式语义理解的难易程度是衡量一个规范复杂与否的关键。我们的方法基于 Manna-Pnueli 时序逻辑框架，在我们的公平转换系统规范中，规范本身是一个时序公式，它精确描述了系统的一个行为集合，由我们的方法得到的规范其时序语义简单、明了。

·证明方法的完备性。一个形式系统的完备性指每一个语义有效的断言都是可证明的。如前所述，我们的方法基于 Manna-Pnueli 框架，这一时序逻辑理论已经得到了广泛而深入的研究，许多现有的完备时序证明系统都可以重用于我们的方法，我们的方法是完备的。

·实用性。基于逻辑规范很可能由于遗漏了某些重要约束而不能完全说明系统。另外，由前述，此种规范方法也很容易得到不相容、不可实现的规范。实际中，由于这种方法并不给出从哪里开始及到何时终止的信息，因此很难实际应用。我们提出的方法给出了一个书写规范的系统方法，通过将程序性质分为安全性和活性两部分分别予以描述来获得规范的完全性，我们的规范过程是相容且完全的。我们的方法使用了两个很初级的概念：状态和转换，其描述是具体的，由此方法产生的规范是一抽象程序，非常易于阅读和理解，特别是对那些不熟悉逻辑的人很易于被他们所接受，因此是实用的。

结束语 我们的公平转换系统规范，既可看成是一转换模块，同时规范本身又是一时序公式，它精确描述了系统的允许行为集，当选用时序逻辑作为规范语言时，这种规范形式一方面为程序验证提供了一个统一框架，另一方面它也为并发系统自顶向下逐步求精提供了基础，我们可以充分利用现有的许多用于证明并发系统时序性质的完全证明系统。我们已将我们的方法成功地用于程序验证之中，我们的公平转换系统规范方法是可行的。

参考文献

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

[2] L. Lamport, Specifying Concurrent Program Modules, ACM Trans. on Prog. Lang. and Sys., 1983,

[3]R. Milner, A Calculas of Communicating System. Lec. Notes in Comp. Sci. 94, Springer-Verlag, 1980

[4]K. M. Chandy and J. Misra, Parallel Program Design, Addision-Wesley, 1988

[5]N. Lynch and M. Tuttle. An Introduction to Input/Output Automata, CWI-Quarterly, 1989, 2 (3)

[6]D. Harel, Statecharts: A Visual Formalism for Complex System, Sci. Comp. Prog., 1987

[7]T. Bolognesi and E. Brinksma, Introduction to the ISO Specification Language LOTOS, Comput. Net. and ISDN Sys., 1987, 14.

[8]M. Abadi 等, Realizable and Unrealizable Concurrent Program Specifications, Lec. Notes in Comp. Sci. 372, 1989, Springer-Verlag

[9]贾国平、郑国梁，公平转换系统规范及其应用，《软件学报》，1996年，第7卷，增刊

[10]贾国平、郑国梁，论公平转换系统模型的可信性，《计算机科学》，Vol. 23, No. 2, 1996