第 8 卷第 2 期
1997 年 2 月

软 件 学 报
JOURNAL OF SOFTWARE

Vol. 8, No. 2
Feb. 1997

一个统一的程序验证框架*

贾国平　郑国梁

（南京大学计算机科学系　南京　210093）

（南京大学计算机软件新技术国家重点实验室　南京　210093）

摘要　本文提出了一个简单的方法，其中程序和其性质都由一个逻辑：时序逻辑中的公式表示. 文中给出了一个程序的转换模块的定义，提出了时序执行语义的概念. 它是一个时序公式，精确地说明了一个程序. 将时序逻辑作为规范语言，程序正确性就意味着说明程序的公式蕴含说明性质的公式，其中蕴含即为一般的逻辑蕴含. 因此，本文的方法为并发程序的规范及验证提供了一个统一的框架. 它允许充分利用现有的用于证明并发系统时序性质的各种完全证明系统. 一个缓冲系统的简单例子用来说明本文的方法. 此例子表明本文的方法是可行的.

关键词　程序验证，规范，程序正确性，时序逻辑，证明系统，缓冲系统.

程序验证是将一个程序与它的规范相比较. 现在主要有 2 种用于研究这种对比的框架. 一种我们称为单语言框架. 在这一框架中，同一语言既用于规范又用于程序设计（即实现）. 在此情况下，规范和程序设计在某种抽象级上都能被看作转换系统. 这些转换系统是由规范和程序设计语言的操作语义来定义的. 它们为它们与程序的比较提供了一个共同的框架. 这一对比常常通过在转换系统上定义一个等价关系或实现先序（Preorder）来进行. 抽象数据类型[1]、CCS[2]和 XYZ[3,4]等都是这一框架的例子.

另一对比框架称为双语言框架. 在此框架中，我们采用 2 个不同的语言，它们分别用于规范和程序设计. 一般地，规范语言 S 说明一个所期望完成系统的“what”部分. 与此对比，程序设计语言 P 应该说明一个实现系统的“how”部分. 在此情况下，语言 S 中的公式 F 作为规范，它说明了一个程序的行为. 此框架中的对比常常是通过定义一个关系 sat，它是 $P \times F$ 的一个子集. 一个断言 $p\ sat\ f$ 说明程序 p 满足由公式 f 所说明的性质，也即 p 是 f 的一个模型. 许多不同的程序逻辑，如时序逻辑[5,6]，都是这一框架中的例子.

本文给出一个简单的方法，其中系统和其性质都由同一逻辑来表示. 我们选用时序逻辑作为描述语言. 做此选择的主要目的在于时序逻辑不仅足以方便地用于说明程序的性质，而且它还提供许多用于程序推理的规则. 现在已有许多用于证明并发系统时序性质的完全证明系统. 文中首先简要介绍了公平转换系统模型及时序逻辑. 给出了一个程序的转换模块的

---

*　本文研究得到博士点基金资助.作者贾国平，1968 年生，博士，主要研究领域为软件工程，形式化方法及时序逻辑. 郑国梁，1937 年生，教授，博士导师，主要研究领域为软件工程，软件开发环境.

本文通讯联系人：贾国平，南京 210093，南京大学计算机科学系

本文 1996-01-26 收到修改稿

定义，提出了时序执行语义的概念. 它是一个时序公式，精确地说明了程序. 最后是结论. 由此方法，程序正确性问题能够归结成时序逻辑系统中的逻辑蕴含. 它为并发程序的规范及验证提供了一个统一的框架. 可以充分利用现有的许多完全证明系统于我们的程序验证问题.

1 方 法

1.1 公平转换系统模型

本节给出一般的公平转换系统模型. 此计算模型最初由文献[7]给出. 它包含了许多具有不同语法表示及通信机制的并发系统. 我们将它作为一种抽象的实现语言. 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，此语言中的公式称为断言.

一个公平转换系统 S 是一个六元组 $\langle V,\Sigma,\Theta,T,WF,SF\rangle$，其中

- $V=\{u_1,\ldots,u_k\}$：状态变量的有穷集合.
- Σ：状态集合.
- Θ：初始条件断言. 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.
- 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 τ 在状态 s 下是能行的当且仅当 $\tau(s)\neq\Phi$，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 都由一个称为转换关系的断言来表示，记为 $\rho_\tau(V,V')$. 它表示由 τ 所执行的状态转换. 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 $\langle s,s'\rangle\models\rho_\tau(V,V')$，其中 $\langle s,s'\rangle$ 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：

- 初始化条件：s_0 是初始状态，即 $s_0\models\Theta$.
- 连续性：对 σ 中每一连续状态对 s_j,s_{j+1}，有转换 $\tau\in T$，使 $s_{j+1}\in\tau(s_j)$.
- 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.
- 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.

1.2 时序逻辑

本节简要介绍时序逻辑，将它作为我们的描述语言.

时序逻辑是一般经典逻辑语言的扩充. 我们所使用的时序逻辑语法上与其它的线性时间时序逻辑，如文献[6,8,9]类似. 它除了包含一般逻辑算子，如布尔联接词 $\sim,\wedge,\vee,\equiv,\to$；等式算子 $=$ 以及一阶量词 $\exists$，$\forall$ 等外，还包含 2 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.当作用于项时,记$Ot\equiv t^+$,称为t的下值.其它时序算子作为简写,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.

为了下面讨论简单起见,我们假设一个基本的断言语言L,它包含谓词演算以及用于表达某些具体论域上的标准算子及谓词的解释符.

我们首先介绍一些概念.

我们称出现在公式p中的所有变量集合为p的词汇.在词汇V上的一个结构是一个无穷状态序列$\sigma:s_0,s_1,s_2,\cdots$,其中每一状态$s_j$是对$V$中变量的一个解释.

令x是V中一变量,状态s'称为状态s的x-变体,如果状态s和状态s'除了x外对其它所有变量的解释都相同,而对x的解释可能不同.一个结构$\sigma':s_0',s_1',s_2',\cdots$是结构$\sigma:s_0,s_1,s_2,\cdots$的$x$-变体,如果对每一$j\geqslant 0$,$s_j'$是$s_j$的$x$-变体.

给定一结构σ,我们记项t在σ中位置$j\geqslant 0$的值为$val(\sigma,j,t)$.下面我们归纳定义一个时序公式p在结构σ中位置$j\geqslant 0$成立(记为$(\sigma,j)\models p$)如下:

· 对L中每一n-元谓词p和项$t_1,\cdots,t_n$,$(\sigma,j)\models p(t_1,\cdots,t_n)$当且仅当$p(val(\sigma,j,t_1),\cdots,val(\sigma,j,t_n))=true$.

· $(\sigma,j)\models\sim p$当且仅当$(\sigma,j)\models p$不成立.

· $(\sigma,j)\models p\vee q$当且仅当$(\sigma,j)\models p$或$(\sigma,j)\models q$.

· $(\sigma,j)\models Op$当且仅当$(\sigma,j+1)\models p$.

· $(\sigma,j)\models pUq$当且仅当对某些$k\geqslant j$,$(\sigma,k)\models q$,并且对所有$i:j\leqslant i<k$,有$(\sigma,i)\models p$.

· $(\sigma,j)\models\forall\ x.\ p$当且仅当对所有$\sigma$的$x$-变体$\sigma'$,有$(\sigma',j)\models p$.

· $(\sigma,j)\models\exists\ x.\ p$当且仅当对某些$\sigma$的$x$-变体$\sigma'$,有$(\sigma',j)\models p$.

在所有状态上都成立的断言称为断言有效的(Assertionally Valid).

如果$(\sigma,0)\models p$,则称公式p在结构σ上成立,记为$\sigma\models p$.一个公式p称为是可满足的(Satisfiable),如果它在某些结构上成立.一个公式p称为是时序有效的(Temporally Valid),如果它在所有的结构上都成立.

当下面考虑一个说明公平转换系统性质的公式p时,我们总是假设词汇V包含S的所有状态变量以及出现在p中的所有变量.另外,需要指出的是公平转换系统S中的计算就是本节中的结构.它们中的状态均是对V中变量的解释.然而,对任意一个结构,它并不一定是系统S中的计算.这是需要特别注意的一点.

V中的变量可以分为严格变量和可变变量2种.严格变量在计算的不同状态上有相同的值,而可变变量在计算的不同状态上值可能不同.转换系统中的状态变量就为可变变量.严格变量主要用于规范目的,它并不出现在系统之中.它主要用于连接计算序列中不同状态中的值.

给定一公平转换系统S,下面我们将注意力限制在计算集$Comp(S)$上,即由S的所有计算组成的集合上.如果公式p在S的所有计算上均成立,则称p在S上是有效的.显然,每一时序有效公式对任何系统S也是有效的.

1.3　公平转换系统的时序语义

本节对于一个给定的公平转换系统 S，构造一个时序公式 $Sem(S)$，它称为 S 的时序语义. $Sem(S)$在结构 σ 上成立当且仅当 σ 是 S 的一个计算. 这个结论最初由文献[5]给出.

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

- $En(\tau)$：$(\exists V')\rho_\tau(V,V')$. 此公式说明转换 τ 是能行的.
- $taken(\tau)$：$\rho_\tau(V,V^+)$. 此公式说明转换 τ 被执行.
- $wf(\tau)$：$\Diamond\Box En(\tau)\rightarrow\Box\Diamond taken(\tau)$. 此公式说明任何满足 $wf(\tau)$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，我们定义它的时序语义公式 $Sem(S)$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.
- $\Box\bigvee_{\tau\in T}taken(\tau)$保证每一转换步都是通过施用某一转换 $\tau\in T$ 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.

上述 4 项分别对应且保证一个序列是系统 S 的计算所必须满足的 4 个要求. 因此我们知道公式 $Sem(S)$精确地表示了系统 S 的计算. 此结果由下述命题给出：

命题 1. 一个结构 σ 满足公式 $Sem(S)$当且仅当 σ 是系统 S 的一个计算.

证明：此结果可以简单地通过将公式 $Sem(S)$中的每一项与系统 S 的计算的定义中的每一项相比较而得到.

1.4 方 法

本节我们首先给出一些定义，然后，给出主要的定理.

1.4.1 一些定义

定义 1. 一个转换模块 M 是一个系统 M：$\langle V,U,\Sigma,\Theta,T,WF,SF\rangle$，其中 S_M：$\langle V,\Sigma,\Theta,T,WF,SF\rangle$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 中变量的解释不同的任何结构.

定义 3. 一个转换模块 M 的时序执行语义，记为 $Run(M)$，定义为 $Run(M)$：$\exists U.Sem(S_M)$，其中 $Sem(S_M)$是公平转换系统 S_M 的时序语义.

显然，公式 $Run(M)$精确地表示了模块 M 的执行，也即一个结构 σ 满足公式 $Run(M)$当且仅当它是 M 的一个执行. 此结果由下面的命题给出：

命题 2. 一个结构 σ 满足公式 $Run(M)$当且仅当 σ 是 M 的一个执行.

需要特别注意的是对于满足相同性质的 2 个程序，它们可能并不具有相同的状态变量. 其中一个程序为了更有效地实现，可能采用了不同的辅助内部变量. 这些变量是如何实现的

并不重要,它们对于实现者是完全自由的.因此,我们将存在量词作用于这些可变变量.这就允许它们完全自由于实现.这正如程序设计语言中隐藏(Hiding)的概念.对可变变量上的存在量词精确含义的理解是理解我们上述定义的关键.

文献[7,8]给出了对于一给定程序,如何构造其相应的公平转换系统的方法.这些系统中的状态变量包含数据变量和一个控制变量 π,它在程序位置上变化,指出下一步将要执行的程序语句.为了将此程序视为转换模块,我们只需标识出它的隐藏变量.从上述讨论,我们不难看出这些隐藏变量即为在程序中说明为局部的变量以及控制变量 π.因此,对每一给定程序,我们可用同样步骤构造出对应的转换模块.

1.4.2　主要结果

在前一节中,对体为 S_M,隐藏变量集为 U 的转换模块 M,我们定义了它的时序执行语义为 $\exists U.Sem(S_M)$.同时,对一给定程序,我们给出了对应转换模块的构造方法.值得注意的一点是,一个转换模块的时序执行语义既可以看作一个时序公式又可以视为一个带某些隐藏变量的公平转换系统.时序执行语义的这一性质是非常有用的.对一个给定的程序,我们能够构造出对应的转换模块 M 并且给出它的时序执行语义.它精确地说明了转换模块 M 的执行.将时序逻辑作为我们的规范语言,我们能够将程序及其性质在同一时序逻辑中表达出来.因此,一个程序满足它的规范的断言可以由逻辑蕴含表示.下面给出我们的主要定理.

定理. 一个程序 P 具有由时序公式说明的性质 ψ 当且仅当 $S_M \models \psi$,即公式 ψ 在系统 S_M 上是有效的,其中 M 是对应的转换模块,S_M 是 M 的体.

证明:令 M 是对应于程序 P 的转换模块,其体为 S_M,隐藏变量集为 U.从 M 的构造过程我们知道 U 包含所有说明为局部的变量以及控制变量.不失一般性,我们假设 U 中的隐藏变量在 ψ 中无自由出现.

程序 P 具有性质 ψ 当且仅当蕴含式:$\exists U.Sem(S_M) \rightarrow \psi$　(*)是有效的.(由命题 2)

应用标准的量词转换,得到蕴含式(*)是有效的当且仅当 $\forall U.(Sem(S_M) \rightarrow \psi)$(**)是有效的.

应用另一个用于量词的推理规则,我们知道蕴含式(**)是有效的当且仅当 $Sem(S_M) \rightarrow \psi$ 是有效的.这意味着任何满足 $Sem(S_M)$ 的序列都满足公式 ψ.应用命题 1,我们可推出 S_M 的任何计算均满足公式 ψ,即 $S_M \models \psi$. □

由上述定理我们知道一个程序 P 满足性质 ψ 等价于时序逻辑系统中蕴含式 $Sem(S_M) \rightarrow \psi$ 的有效性,其中 S_M 是对应于程序 P 的转换模块 M 的体.这一结果允许我们为了证明一个程序 P 具有性质 ψ 而利用任何用于证明并发程序时序性质的证明系统.[6,9~11]因此,我们得到结论可以在一个统一的时序逻辑框架中说明和验证并发程序.

限于篇幅,我们在这里并不给出完全的证明系统.详细的公理、证明规则以及它们的推理方法可参见文献[6,9~11].

2　例子:一个缓冲系统

为了说明我们的方法,我们给出了一个缓冲系统的例子.此系统通过 2 个异步通道 α 和 β 与它们的环境进行通信,其中 α 用于输入,β 用于输出.(见图 1)

图1 一个缓冲系统

令 *Buffer* 为一个缓冲器. 它从通道 α 接收消息,按消息到达的顺序将它们存放在内部存储器 q 中,然后,再以相同的顺序在通道 β 上将它们发送出去. 不失一般性,我们假设所有的消息都是自然数.

文献[12,13]已经对此例的时序性质进行过描述. 此处作为我们方法的说明,我们给出它的转换模块的构造以及它的时序执行语义.

假设有 2 个异步通信原语:发送原语 $\beta!\ e$,它在通道 β 上发送表达式 e 的值和接收原语 $\alpha?\ y$,它从通道 α 接收消息并且将它的值赋给变量 y.

对应每一异步通道 γ,缓冲系统对应的转换系统中包含一个变量 γ,它在整数表上变化. 在计算的任一点,γ 包含那些已经发送给通道 γ 但还未从它接收到的消息表. 对应于语句 $\beta!\ e$ 的转换关系中包含合取项 $\beta'=\beta\cdot e$,它说明 β 的下一值是通过将 e 的当前值加到 β 的尾部而得到. 对应于语句 $\beta?\ y$ 的转换关系中包含合取项 $\beta=y'\cdot\beta'$,它说明 y 的下一值是当前包含在表 β 中的第 1 个元素,而 β 的下一值是通过移去这一元素而得到.

下面我们引入自然数表上的序的记号:

$\gamma\prec\delta\Leftrightarrow$或者对某一自然数 m,$\delta=\gamma\cdot m$ 或者 $\gamma=\delta$.

$\gamma\succ\delta\Leftrightarrow$或者对某一自然数 m,$\gamma=m\cdot\delta$ 或者 $\gamma=\delta$.

对 *Buffer*,我们定义转换模块为 $S_{buf}=\langle V,U,\Sigma,\Theta,T,WF,SF\rangle$,其中

$V:\{\alpha,\beta,q\}$,$U:\{q\}$,$\Theta:\alpha=\beta=q=(\ )$,

$T:\{\tau_I,\tau_1,\tau_2,\tau_3,\tau_4\}$,它们的转换关系如下:

$\rho_{\tau_I}:\alpha'=\alpha\wedge\beta'=\beta\wedge q'=q$.

$\rho_{\tau_1}:\alpha\prec\alpha'\wedge\beta'=\beta\wedge q'=q$.

$\rho_{\tau_2}:\alpha=\alpha'\wedge\beta'\succ\beta\wedge q'=q$.

$\rho_{\tau_3}:\exists\ d.(\alpha=d\cdot\alpha'\wedge\beta'=\beta\wedge q'=q\cdot d)$.

$\rho_{\tau_4}:\exists\ d.(\alpha=\alpha'\wedge\beta'=\beta\cdot d\wedge q=d\cdot q')$.

$WF:\{\tau_3,\tau_4\}$,$SF:\Phi$

它对应的时序执行语义为:$\exists\ q\in N^*.\ Sem(S_{buf})$,其中 N^* 是自然数序列集合,$Sem(S_{buf})$ 如下:

$$Sem(S_{buf}):\Theta\wedge\Box\bigvee_{\tau\in T}taken(\tau)\wedge\bigwedge_{\tau\in WF}wf(\tau).$$

值得指出的是,虽然我们的时序执行语义中用了一个表值变量 q 来表达期望的性质,但这并不意味着在系统实现时必须采用相同的数据结构. 这仅仅是一种使程序更易于表示的设施. q 是一内部变量,它对于实现者是完全自由的. 保证上述表示是抽象的且其实现是完全自由的,关键就是在可变变量 q 上的存在量词. 它等价于程序设计语言中隐藏(Hiding)的概念.

由上述定理知,*Buffer* 满足时序性质 ψ 等价于蕴含式 $Sem(S_{buf})\rightarrow\psi$ 在时序逻辑系统中是有效的. 我们可以利用现有的许多证明并发程序时序性质的完全证明系统来对我们的例子进行验证.

3 结 论

3.1 与相关工作的比较

在文献[3,4]中,唐稚松教授提出了一个用于时序逻辑程序的证明系统XYZ.在他的系统中,XYZ既是一个时序逻辑系统又是一程序设计语言.它作为系统形式语义、规范和验证的统一的逻辑基础.我们在本文中考虑了一个在某种抽象级上更广泛的程序设计语言,即带公平性假设的转换系统.它是一个抽象计算模型.现有的大多数系统都可以包含于其中.

最近,Lamport提出了活动时序逻辑TLA(temporal logic of actions)以及它的证明规则.[14]在他的方法中,系统和它们的性质也都能够由TLA逻辑中的公式统一表示.我们的方法基于Manna-Pnueli的时序逻辑框架.[8]这一时序逻辑理论目前已经得到了深入而广泛的研究.现在已有许多用于它的完全证明系统.[6,7,9~11]另外,我们的时序逻辑框架其时序语义比TLA的语义更加简单.我们的方法允许我们充分利用这些优点.

3.2 进一步的研究

本文提出的框架同样可以应用于并发系统的逐步求精:一个系统是另一个系统的实现的断言也能够由逻辑蕴含表示.利用现有的许多证明技术,我们可以证明一个系统精化了另一个系统.我们能够得到一个严格的系统开发方法.它从最高级规范开始直至精化到最低级规范,即实现.此工作正在我们的研究之中,其结果我们将另文给出.

参考文献

1 Goguen A, Thatcher J W, Wagner E G. An initial algebra approach to the specification, correctness, and implementation of abstract data types. In: Current Trends in Programming Methodology, Prentice-Hall, 1978, 4: 80~149.

2 Milner R. A calculus of communicating systems. In: Lec. Notes in Comp. Sci. 94, Springer-Verlag, 1980.

3 Tang C S. XYZ a program development environment based on temporal logic. In: Proc. Conf. on Prog. Lang. and Syst, 1983. 7~19.

4 Tang C S. Toward a unified logic basis for programming languages. In: Mason R E A ed. Information Processing 83, North-Holland, 1983. 425~429.

5 Pnueli A. The temporal semantics of concurrent programs. Theor. Comp. Sci., 1981, **13**: 1~20.

6 Manna Z, Pnueli A. Verification of concurrent programs: a temporal proof systems. In: de Bakker J W, van Leeuwen J eds. Foundations of Computer Science IV, Distributed Systems: Part 2, Amsterdam, 1983. 163~255.

7 Manna Z, Pnueli A. How to cook a temporal proof system for your pet language. In: Proc. 10th ACM Symp. Princ. of Prog. Lang., Austin, Texas, 1983. 141~154.

8 Manna Z, Pnueli A. The temporal logic of reactive and concurrent system: specification. New York: Spinger-Verlag, 1991.

9 Owicki S, Lamport L. Proving liveness properties of concurrent programs. ACM Trans. on Prog. Lang. and Sys., 1982, **4(3)**: 455~495.

10 Manna Z, Pnueli A. Adequate proof principles for invariance and liveness properties of concurrent systems. Sci. Comput. Prog., 1984, **32**: 257~289.

11 Manna Z, Pnueli A. Completing the temporal picture. Theor. Comp. Sci., 1991, **83(1)**: 97~130.

12 Lamport L. What good is temporal logic? In: Mason R E A ed. Information Processing 83, North-Holland, 1983. 657~668.

13 Pnueli A. Specification and verification of reactive systems. In: Kugler H J ed. Information Processing 86, North-Holland, 1986. 845～858.

14 Lamport L. The temporal logic of actions. ACM Trans. on Prog. Lang. and Sys., 1994, **16(3)**: 872～923.

A UNIFIED FRAMEWORK FOR PROGRAM VERIFICATION

JIA Guoping ZHENG Guoliang

(*Department of Computer Science Nanjing University Nanjing* 210093)

Abstract This paper presents a simpler approach in which both a program and its properties are specified by formulas in the same logic: the temporal logic. A transition module of a program is defined and the notion of temporal run semantics, which is a temporal formula precisely characterizing the program is presented. Taken temporal logic as specification language, the correctness of a program means that the formula specifying the program implies the formula specifying the property, where implies is ordinary logical implication. Therefore this approach gives a unified framework for specifying and verifying concurrent programs and allows a full utilization of the existing comprehensive proof systems developed for proving temporal properties of systems. A simple example of a buffer system is presented to illustrate this method and show that the approach is very promising.

Key words Program verification, specification, correctness of a program, temporal logic, proof system, a buffer system.