计算机科学 1997 Vol. 24№. 6

并发系统的操作时序逻辑描述和验证*)

The Specification and Verification of the Temporal Logic of Action(TLA)for Concurrent Systems

蒋 慧 张兴元 王元元

(中国人民解放军通信工程学院计算机教研室 南京 210016)

摘 要 The Temporal Logic of Action(TLA)is a kind of temporal logic used to specify and to verify concurrent system, it is still in the tradition of assertional methods for reasoning about. In this method, the algorithm of the concurrent system and its properties is expressed by a TLA formula. So, the proof of the assertions which specify the concurrent system's specification and its properties are all in the system of a underlying logic system. TLA is a simply and a relatively compelete temporal logic. This paper introduces the TLA, and describes how it is used to specify and verify concurrent system.

关键词 Temporal logic, Temporal logic of action, Safety, Liveness, Fairness

1. 引言

近年来,为了在开发一个复杂系统的过程中尽量提高系统的正确性,减少开发过程中重复、繁琐的工作,国内外许多学者都从逻辑的角度进行研究,用逻辑系统对并发系统程序进行描述、验证,以期通过逻辑系统的简洁和严密来保证大型并发系统的正确性和精确性。

用时序逻辑来证明并发程序的思想最早是由 Pnueli 在 1977 年提出的,它是一种变迁公理方法(transition-axiom method)。在这种方法中,通过抽象程序和时序逻辑的组合来描述一个并发系统。随着将近二十多年的研究,这种方法经过不断的发展和演化,已经比较完善和成熟,并且逐步从理论的研究转为实践的应用。其中较为成熟和流行的时序逻辑系统有三个流派:Unify logic[Chandy and Misra 1988]、Manna & Pnueli[1991]的 MPTL 和 Lamport 的 TLA[1992]。Lamport 的 TLA 具有前两种系统所不具有的一些优点,本文就其在并发系统的描述和验证中的应用作一简单的讨论。

2. 操作的时序逻辑(TLA)

TLA(*Temporal Logic of Action*)是操作逻辑(logic of actions)和标准的时序逻辑的结合。TLA 的语义及部分规则见图 1 所示。

$\forall \overrightarrow{x_1}[p_1(\overrightarrow{t_1}) \leftarrow H_1]\cdots\forall \overrightarrow{x_n}[p_n(\overrightarrow{t_n}) \leftarrow H_n]$ $(n\geqslant 0)$

则由[45],可定义下述 DR(definitional reflection)规则,而系统的 CUT 消去定理仍成立:

$$\frac{\{\theta\Gamma,\theta H_i\rightarrow\theta C \mid \text{where } \theta \text{ is the mgu of A and } p_i(\overrightarrow{t_i})\}}{\Gamma,A\rightarrow C}\mathrm{DR}$$

Raymond McDowell 等指出[37],上述定义形式能够描述抽象状态变迁系统及 CCS 的线性逻辑规范,并且证明,在不使用 DR 规则的情况下,则状态 p 到状态 q 的可达性对应于矢列 $q\rightarrow p$(或其线性逻辑描述形式)的可推出性。假如使用 DR 规则,模拟和双模拟体现为可推出性。这些成果说明:合适的可算的线性逻辑子集不仅能用于描述抽象的状态变迁系统,而且能用于描述和判断如模拟和双模拟这样的元性质。

致谢:作者感谢 Dale Miller 教授提供的材料以及 P. Lincoln 所维护的线性逻辑 mailinglist,上海交通大学孙永强教授对我们在线性逻辑方面的研究工作给予很多支持。参考文献共 48 篇(略)。

---

*)国家自然科学基金资助项目,项目号:69572041。蒋 慧 硕士生,主要研究方向:协议验证、人工智能及其在网络中的应用。张兴元 副教授,主要研究方向:协议验证和网络。王元元 教授,主要研究方向:数理逻辑、人工智能。

2.1 操作的逻辑

算法是对变量进行操作，所以从某种角度来说，认为算法就是对变量赋值。我们简单地用无限集 Var 来表示一切变量名，用无限集 Val 来表示算法中可能遇到的一切值，如 1，－2…，字符串"ABC"…，以及集合如自然数集等等。逻辑则是一些公式以及对这些公式进行操作的规则，为了理解 TLA 的公式及其操作的意义，用状态(states)来描述操作逻辑的语义：

·状态：是对变量的赋值，也就是从变量名集 Var 到值集 Val 的一个映射。

·状态函数：一个用变量和常数符号构造的非布尔值表达式，如 x＋y－3，即，从状态集 St 到值集 Val 映射，即函数赋给状态的值。

·状态谓词：一个由变量和常数符号构造的布尔值表达式，如 x＋y＝3，即从状态集 St 到布尔值的映射。我们说一个谓词 P 给任一状态赋值为 true 或为 false，状态满足谓词 P 当且仅当谓词给状态的赋值为真。

TLA 语义：

变量的值(状态 s 给变量 x 赋的值)：$s[x]$
状态函数(函数 f 给状态 s 赋的值)：$s[f]\Leftrightarrow f(\forall\, v: s[v]/v)$
操作：$s[A]t\Leftrightarrow A(\forall\, v: s[v]/v, t[v]/v')$
$\delta[\neg F]\rightarrow\delta[F]$
$A\Leftrightarrow\forall\, s,t\in S: s[A]t$
$F\Leftrightarrow\forall\, \delta\in St^{\infty}: \delta[F]$
$s[\text{Enabled } A]\Leftrightarrow\exists\, t\in S: s[A]t$
$\langle s0,s1,s2,\cdots\rangle[\Box F]\Leftrightarrow\forall\, n\in Nat: \langle S_n,S_{n+1},K\rangle[F]$
$\langle s0,s1,s2,\cdots\rangle[A]\Leftrightarrow s0[A]s1$
注：$(\forall\, v: K/v, K/v')$ 表示替换所有变量 v

附加符号：

$F\leadsto G\Leftrightarrow\Box(F\Rightarrow\Diamond G)$
$\Diamond F\Leftrightarrow\neg\Box\neg F$
Unchanged $f: f'=f$
$[A]_f\Leftrightarrow A\vee(f'=f)$(读作方块 A 下 f)
$\langle A\rangle_f\Leftrightarrow A\wedge(f'\neq f)$(读作角 A 下 f)
$WF_f(A)\Leftrightarrow(\Box\Diamond\langle A\rangle_f)\vee(\Box\Diamond\neg\text{Enabled}\langle A\rangle f)$
$SF_f(A)\Leftrightarrow(\Box\Diamond\langle A\rangle_f)\vee(\Diamond\Box\neg\text{Enabled}\langle A\rangle f)$

TLA 的基本规则：

$$\text{TLA1}:\ \frac{P\wedge(f'=f)\Rightarrow P'}{\Box P\equiv P\wedge\Box[P\Rightarrow P']_f}$$

$$\text{TLA2}:\ \frac{P\wedge[A]_f\Rightarrow Q\wedge[B]_g}{\Box P\wedge\Box[A]_f\Rightarrow\Box Q\wedge\Box[N\wedge I\wedge I']_f}$$

TLA 的附加规则：

$$\text{INV1}:\ \frac{I\wedge[N]_f\Rightarrow I'}{I\wedge\Box[N]_f\Rightarrow\Box I}$$

$$\text{INV2}:\ \vdash\Box I\Rightarrow(\Box[N]_f\equiv\Box[N\wedge I\wedge I']_f)$$

$$\text{WF1}:\ \frac{\begin{array}{l}P\Rightarrow\text{Enabled}\langle A\rangle_f\\ P\wedge[N]_f\Rightarrow(P'\vee Q')\\ P\wedge(N\wedge A)_f\Rightarrow Q'\end{array}}{\Box[N]_f\wedge WF_f(A)\Rightarrow(P\leadsto Q)}$$

图 1 TLA 的部分语义及规则

·操作(action)：一个操作是一个由变量、初始化变量(x′)和常数符号构造而来的布尔值表达式。它是新旧状态之间的一个关系，即给一对状态 s，t 赋一个布尔值的函数。其中 s 为旧的状态，t 为新的状态。未初始化的变量代表旧的状态，初始化的变量代表新的状态。如 $x=y'-1$，意为 x 在旧的状态下的值等于 y 在新的状态下的值减 1。

·操作步(step)：状态对 s，t 称为一个"A 步"当且仅当 $s[A]t$ 为真。如果操作 A 代表一个程序的一个原子操作，那么 s，t 就是一个 A 步当且仅当在状态 s 下执行这个操作能够产生状态 t。我们也可以把谓词 P 看成一个没有初始化变量的操作。因此，$s[P]t$ 是一个布尔值，对任两个状态 s，t 等于 $s[P]$。一个状态对 s，t 是一个 P 步当且仅当 s 满足 P。

·Enabled 谓词：对任意操作 A，我们定义谓词 Enabled A 对某状态为真当且仅当从这个状态开始能够进行一个 A 步。Enabled A 的定义为如果操作 A 代表一个程序的一个原子操作，对那些能够执行这个操作的那些状态，Enabled A 的值为真。

在并发系统程序中，状态函数与普通程序语言中的表达式以及普通程序验证中的断言的子表达式对应，状态谓词与普通程序语言中的布尔值表达式及程序验证中的断言对应，而并发系统中的一个原子操作，在 TLA 中是由一个操作来表示的。形式化地，$s[A]t$ 是从 A 通过把每个未初始化的变量 v 用 $s[v]$ 代替，把每个初始化的变量 v′ 用 $t[v]$ 代替，所以 $s[x=y'-1]t$ 的值就等于布尔值 $s[x]=t[y]+1$。

·操作的有效性及其可证明性：一个操作是有效的，A 当且仅当每一步是一 A 步，好 $A\Leftrightarrow\forall\, s,t\in S: s[A]t$。

用 $\vdash F$ 表示 F 是可以由逻辑的推理规则证明的，逻辑的合理性意为每一个可证明的公式都是有效的，也就是说，$\vdash F$ 蕴含 F。TLA 在实践中的成功之处在于，它的公式验证依赖于它是怎样用一般的数学知识来说明的。

2.2 简单的时序逻辑

我们可以把一个算法的执行看成是一序列的操作步，每一步都通过改变某些变量的值来产生一个新的状态。因为一个执行是一个状态序列，那么算法的语义是它的可能执行的集。时序逻辑的语义是行为，对状态序列进行描述。

2.2.1 状态和公式的关系

·行为 $\delta(\langle s0,s1,s2,\cdots\rangle)$：一个无限的状态序列，其第一个状态为 s0。第二个状态为 s1。可以把行为看作是一个计算算法的设备在执行一个算法的过

程中的状态序列(因此,一个可终止的算法的执行是一个有穷的状态序列)。

·公式F:时序逻辑的公式解释成对行为的一个断言,是一个行为的布尔值的函数。

δ[F]:公式F赋给行为δ的一个布尔值。

⟨s0,s1,s2,…⟩[F]⇔s0[F]s1:行为δ满足公式F当且仅当公式F赋给行为δ的值为真。

如果把行为⟨s0,s1,s2,…⟩表达为一个可能世界的进化,那么sn是可能世界在时刻n的状态。⟨sn,sn+1,…⟩[F]断言F在时刻n为真。

2.2.2 几个有用的时序逻辑的算子

·总是(□ always):⟨s0,s1,s2,…⟩[□F]断言公式F在行为⟨s0,s1,s2,…⟩的所有时刻都是真的,即F总是真的:

⟨s0,s1,s2,…⟩[□F]⇔∀ n∈Nat:⟨sn,sn+1,sn+2,…⟩[F]

·最终(◇ eventually):⟨s0,s1,s2,…⟩[◇F]断言公式F在行为⟨s0,s1,s2,…⟩的所有时刻并不总是都是假的,即F最终会是真的。

·无限经常(□◇ infinitely often):一个行为满足公式□◇F当且仅当F在这个行为的无限多个时刻是真的,也就是说公式F是无限多次地为真:

⟨s0,s1,s2,…⟩[□◇F]⇔∀ n∈Nat:∃ m∈Nat:⟨sn+m,sn+m+1,sn+m+2,…⟩[F]

·最终总是(◇□ eventually always):公式□◇F断言最终公式F总是真的:一个行为满足公式◇□F当且仅当在这个行为中,有某一时刻,从这一时刻起公式F为真,说公式F是无限多次地为真。

·导致(leads to):我们定义公式FαG等价于□(F⇒◇G)。这个公式断言在任何时刻,只要F为真,G立刻或在一定时间以后为真;操作符α是传递的,也就是说任何满足(FαG)和(GαH)的行为也满足(FαH)。

2.3 原始的操作逻辑(RTLA)

原始的操作逻辑是通过用操作作为时序逻辑的公式而得到的。RTLA的语义是看一个操作在一个行为上是否为真。我们定义:

对一个行为δ的操作[A]是真⇔这个操作的最前面的状态对是一个A步,即⟨s0,s1,s2,…⟩[A]⇔s0[A]s1。

一个操作满足谓词P当且仅当操作的第一个状态满足谓词P。即⟨s0,s1,s2…⟩[P]⇔s0[P]。

3 用TLA描述并发系统

TLA通过描述并发算法的执行的外部可观察到的状态序列来描述并发系统。这个状态序列(即行为)中的每一个状态都是一个操作步,所以,TLA描述并发程序的实质是一种状态变迁的方法,需要指定每一状态的操作,即下一状态关系;另外要指定被操作所赋值的变量元组,并指明它们在程序中的初始状态。为了简化描述,我们定义以下的几种表示:

两个有序的状态对是相等的当且仅当它们的每一元素都是相等的,⟨x,y,z,…⟩=⟨x′,y′,z′,…⟩。

设A为任一操作,f为任一状态函数,有:

〔A〕f⇔A∨(f′=f)(读作方块A下f)

⟨A⟩f⇔A∧(f′≠f)(读作角A下f)

Unchanged f⇔(f′=f)

TLA在描述程序的公式时有着相同的形式,它们可以写成Φ,Φ是一个合取式Init∧□[N]f∧F。其中Init:指定变量初始值的谓词;N:程序的"下一状态关系",代表每一个原子操作的操作(actions);f:所有可变变量的n元组;F:是WFf(A)与/或SFf(A)的合取,其中A代表程序的原子操作的某一子集。

下面我们说明F的意义。

3.1 描述并发系统的三种属性:安全性、活性和公平性

时序逻辑通过描述系统的行为,即系统在和外界环境进行交互的时候产生的可观察到的操作(operations)序列来描述系统。精确地说,它描述了两种操作属性:安全和活性属性。安全性断言允许系统做什么,或等价地,不允许系统做什么。例如,部分正确性断言一个程序不会产生一个错误的答案是程序的一个安全性属性。活性属性断言一个系统必须做什么。例如,终止性断言一个程序必须最终产生一个答案就是一个活性属性的例子。形式化地,安全属性是一个断言,是一个断言某事永远不会发生的断言;活性属性是一个断言,是一个断言某事最终会发生的断言。

当用一个抽象的不确定性的程序作为一个算法的规范时,我们注意到它的安全属性。一个操作满足某种属性是指这个操作能够被这个程序产生。但是在程序的规范中的活性属性会给程序加上不必要的安全性,因此,我们用公平性来代替对安全性的描述。公平性限制了那些断言某些操作必须最终发生的断言,以保证必须终止的执行最终会结束的活性

属性。用公平性来代替活性属性的程序是机器关闭(machine closed)的，也就是说多余的安全属性不会附加给操作。

公平性断言指如果一个操作可能执行，那么程序必须最终执行它。并发程序的公平性可以用弱公平性和强公平性来表示。

弱公平性($WF_f(A)$)：若一个程序执行一个操作的可能性能保持足够长的时间，则它必须或者不可能再被执行或者最终被执行。其形式为：

$WF_f(A)\Leftrightarrow(\Box\Diamond executed)\vee(\Box\Diamond impossible)$。

强公平性($SF_f(A)$)：若一个程序执行一个操作的可能性足够经常，则它或者最终永远不可能被执行或者必须被执行。其形式为：

$SF_f(A)\Leftrightarrow(\Box\Diamond executed)\vee(\Diamond\Box impossible)$

"执行"就是 $Enabled\langle A\rangle_f$，"不可能"就是 $\neg Enabled\langle A\rangle_f$，从前面的定义，对任一操作 A 和状态 f 执行一个操作即进行一个 A 步，而当且仅当 $Enabled\langle A\rangle_f$ 为真才可能执行一个步。所以，不可能执行一个操作即不能进行一个 A 步，即 $\neg Enabled\langle A\rangle_f$，所以强公平性和弱公平性可以表示成：

$$WF_f(A)\Leftrightarrow(\Box\Diamond\langle A\rangle_f)\vee(\Box\Diamond\neg Enabled\langle A\rangle f)$$

$$SF_f(A)\Leftrightarrow(\Box\Diamond\langle A\rangle_f)\vee(\Diamond\Box\neg Enabled\langle A\rangle f)$$

3.2 对一个简单的并发程序的描述

我们对下面程序进行描述

```
var natural x,y=0;
do ⟨true→x ' =x+1⟩
   □
   ⟨true→y ' =y+1⟩od
```

其 TLA 公式为：

$Init_\Phi\Leftrightarrow(x=0)\wedge(y=0)$

$M_1\Leftrightarrow(x'=x+1)\wedge(y'=y)$

$M_2\Leftrightarrow(y'=y+1)\wedge(x'=x)$

$M\Leftrightarrow M_1\vee M_2$

$\Phi\Leftrightarrow Init_\Phi\wedge\Box M\ \wedge\ WF_{\langle x,y\rangle}(M_1)\wedge WF_{\langle x,y\rangle}(M_2)$

根据 Dijkstra 的语义，观察程序 1 的活性属性，它要求程序永远都不终止，即要求程序的操作必须有无限多的步骤。为了保证程序 1 中 x，y 的值都能够改变，我们把程序 1 的活性属性定义成为：$F\Leftrightarrow WF_{\langle x,y\rangle}(M_1)\wedge WF_{\langle x,y\rangle}(M_2)$，因为 $WF_{\langle x,y\rangle}(M_1)\wedge WF_{\langle x,y\rangle}(M_2)$和 $WF_{\langle x,y\rangle}(M_1)$都蕴含 M，所以 $WF_{\langle x,y\rangle}(M_1)\wedge WF_{\langle x,y\rangle}(M_2)$和 $WF_{\langle x,y\rangle}(M_1)$都不给 $Init_\Phi\wedge\Box[M]_{\langle x,y\rangle}$增加安全属性。

我们用 TLA 公式所描述的是一个并发系统的规范，它仅描述系统外部可见状态的组成元素。但是，如果用不可见的内部元素来描述操作的话将是非常有用的。系统从外部看可能是下面这种形式$\{e0,e1,e2,\cdots\}$，其中，ei 是外部可见元素的状态，如果存在内部元素的状态 yi，那么完全的行为将是形式$\{(e0,y0),(e1,y1),(e2,y2),\cdots\}$，一个规范必须能够允许那些仅有内部元素改变而外部元素不改变的步，例如序列$\{(e0,y0),(e1,y1),(e1,y1'),(e1,y1'),(e2,y2),\cdots\}$，由于内部的步不是外部可见的，因此这个序列的外部状态是$\{e0,e1,e1,e1,e2,\cdots\}$，而且应该等价于去掉"停滞步"的步骤$\{e0,e1,e2,\cdots\}$。因此一个正确的规范应该允许加上或是删除有穷的停滞步。这称为"规范在停滞条件下的不变性"。

因此程序 1 的规范我们还必须加上"停滞不前"条件。首先注意我们前面定义的一个表达式$(A)_f\Leftrightarrow A\vee(f'=f)$，所以程序 1 的规范到目前可以完整地表达为：

$\Phi\Leftrightarrow Init_\Phi\wedge\Box[M]_{\langle x,y\rangle}\wedge WF_{\langle x,y\rangle}(M_1)\wedge WF_{\langle x,y\rangle}(M_2)$

3.3 用 TLA 证明并发系统的性质

在 TLA 中，并发系统的性质是用一个 TLA 公式 F 来描述的，断言"程序 Φ 具有属性 F"是由公式 $\Phi\Rightarrow F$ 的有效性来表达的，它断言每一个满足 Φ 的行为都满足 F。

3.3.1 证明不变性 一个并发系统的不变性有诸如部分正确性，不死锁以及互斥性等。不变性由一个 TLA 公式$\Box P$表示，其中 P 是一个谓词，是用规则 INV1 来证明的。下面证明 Φ 满足类型正确性的性质。类型正确性表示为：

$T\Leftrightarrow(x\in Nat)\wedge(y\in Nat)$

下面证明 $\Phi\Rightarrow F$。

$\because\Phi\Rightarrow Init_\Phi\wedge\Box[M]_{\langle x,y\rangle}$

$\because Init_\Phi\Rightarrow T$，

$[M]\langle x,y\rangle\equiv M\vee[\langle x,y\rangle'=\langle x,y\rangle]$

$\equiv M_1\vee M_2\vee[\langle x,y\rangle'=\langle x,y\rangle]$

又 $T\wedge M_1=T'$

$T\wedge M_2=T'$

$T\wedge[\langle x,y\rangle'=\langle x,y\rangle]\Rightarrow T'$且

$T'\equiv((x\in Nat)\wedge(y\in Nat))'$

$\equiv(x'\in Nat)\wedge(y'\in Nat)$

$\therefore\ T\wedge M_1\Rightarrow(x\in Nat)\wedge(x'=x+1)\Rightarrow x'\in Nat$

$\therefore\ T\wedge M_1\Rightarrow y'\in Nat$

如果 T 是一个操作$[M]_{\langle x,y\rangle}$的不变属性，则意味着$T\wedge[M]_{\langle x,y\rangle}\Rightarrow T'$。所以，一般来说，要证明并发程序 $\Phi=Init\wedge\Box[N]_f\wedge F$ 满足一个不变性 $\Phi\Rightarrow\Box T$ 比较简单。

3.3.2 证明最终性 一个并发系统的最终性断言某些事情最终要发生。一些传统的最终性如结

计算机科学1997 Vol.21 No.6

World Wide Web的索引与查询技术

The Index and Query Techniques of World Wide Web

阳小华　周龙骧

（中国科学院数学研究所　北京100080）　（中南工学院计算机系）

摘　要　With the explosive growth of World Wide Web, one of the most pressing issues is the so-called resource discovery problem. It leads to the development of information systems which index the Web documents and allow users to locate resources by specifying keywords. In this paper, we discuss the index and query techniques used in current WWW information systems and the future work of WWW search.

关键词　World Wide Web, information retrieve, index database

1　引言

WWW(World Wide Web)是一个由许多称为Web页的超媒体文档组成的集合，这些文档用HTML(Hyper Text Markup Language)书写，包含多种媒体对象和指向其它文档的指针(超级链接)。Web文档散布在世界各地的Web服务器上，每个服务器自主地管理自己的资源，没有统一的管理机制。由于WWW的迅速发展，其信息容量已超过Gopher和WAIS，成为全球最大的信息系统，因而要在

束、消息发送等，也是由TLA公式来表示的，最终性的证明一般都归约成 $P \leadsto Q$。因为并发系统的安全属性蕴含任何事情会发生，因此 $P \leadsto Q$ 的证明必须从程序的活性条件中导出。我们使用规则WF1来证明。只要做一些简单的替换就可以了：

$P \leftarrow n \in Nat \wedge x = n$　$N \leftarrow M$　$f \leftarrow \langle x, y \rangle$

$Q \leftarrow x = n+1$　$A \leftarrow M_1$

则该规则的假设变成

$(n \in Nat \wedge x = n) \wedge [M]_{\langle x,y \rangle} \Rightarrow ((n \in Nat \wedge x' = n) \vee (x' = n+1))$

$(n \in Nat \wedge x = n) \wedge \langle M_1 \rangle_{\langle x,y \rangle} \Rightarrow (x' = n+1)$

$(n \in Nat \wedge x = n) \Rightarrow Enabled\langle M_1 \rangle_{\langle x,y \rangle}$

所以规则的结论很容易得到

$\Box[M]_{\langle x,y \rangle} \wedge WF_{\langle x,y \rangle}(M_1) \Rightarrow ((n \in Nat \wedge x = n) \leadsto (x = n+1))$

在TLA中，一个并发程序和它的属性之间没有任何区别，对于 Φ 来说，我们与其把它看成是一个并发系统的描述，不如把它看成是一个要求程序所必须满足的性质。

3.4　用TLA描述低级程序实现高级程序

一个系统可以在许多抽象层次上进行描述，从高级的描述到低级的实现。如果S1允许的每一个外部可见的操作S2也允许，则我们说规范S1实现规范S2。为了证明S1实现S2，我们要证如果S1允许的操作序列$\langle (e0,z0),(e1,z1),(e2,z2),\cdots \rangle$，其中zi为内部状态，那么必然存在S2允许的内部状态 $yi \Leftrightarrow \langle (e0,y0),(e1,y1),(e2,y2),\cdots \rangle$。

只要找到一个函数f使$(ei,zi)=f(ei,yi)$，证明f映射S1的执行序列(可能有停滞不前步)到S2的执行序列，同时还保证了S2的活性属性。一个映射f是保序列和保活性属性的叫做一个精炼映射。

限于篇幅，本文不再讨论用TLA的公式和演泽规则来证明了。有关这方面的详细的内容，可参考本文的有关文献。

参考文献

[1] Leslie Lamport, The Temporal Logic of Actions, ACM Trans. on Programming Language and Systems, 16(3), 1994

[2] Martin Abadi and Leslie Lamport, The Existence of Refinement Mappings, Theoretical Computer Science 82, 1991

[3] 王元元，计算机科学中的逻辑学，1989