时序逻辑 TLA 简介

(Introduction to TLA)

Leslie Lamport

December, 1994

版权声明 (Copyright)

译者序

近代时序逻辑 (Temporal Logic) 是由著名的新西兰逻辑学家和哲学家 Arthur Prior(4 December 1914 –6 October 1969) 在 1953 年提出和发展而来。时序逻辑是把时间的变量引入到逻辑推理中，例如，“I am always hungry”，I will eventually be hungry”, or ”I will be hungry until I eat something”) 等等。

1977 年，以色列著名的计算机科学家 Amir Pnueli(April 22, 1941 –November 2, 2009) 将时序逻辑引入到计算机科学领域。1996 年，Pnueli 因为在时序逻辑和程序验证方面的开创性研究工作，获得了图灵奖，其图灵奖的获奖原因为：“for seminal work introducing temporal logic into computing science and for outstanding contributions to program and systems verification.”。

时序逻辑动作 (行为)（Temporal Logic of Actions）是 Leslie Lamport 在 1994 年正式提出和发表的一种时序逻辑，TLA 将时序逻辑与系统的状态动作相结合，用于描述并发系统的行为。Lamport 在分布式系统，并发系统方面的理论，算法和工程贡献都是非常巨大，并于 2013 年因为 “fundamental contributions to the theory and practice of distributed and concurrent systems, notably the invention of concepts such as causality and logical clocks, safety and liveness, replicated state machines, and sequential consistency” 获得了图灵奖。

在时序逻辑中，有几个重要的概念。
–safeness 是一个非常重要的概念，其目的是刻画一个系统的不变性，可以理解为 “坏事情绝对不能发生”。例如，系统绝对不能在生命周期里，进入某种状态，例如，“机器人不能伤害人类”。

–fairness/liveness-，表达的是“好事情最后一定会来到”，例如，“自动驾驶车一定要抵达目的地”。fairness 有弱公平性 (Weak Fairness) 和强公平性 (Strong Fairness)，分别刻画了对一个时序系统并发行为的公平性的强弱，例如，弱公平性表达了：只要一旦一直永远可能发生，事情就一定会频繁的发生；强公平性则表达了：只要不是永远不可能，事情就一定会频繁的发生。

整体而言，时序逻辑的思想和工具被广泛的用在了程序验证，模型验证领域。但在大规模的操作系统和应用软件领域，例如，互联网软件，并没有得到大规模的应用。这可能与通过一个形式化方法，数学逻辑的方法来描述一个大型应用软件系统的难度有很大的关系。不过，学习时序逻辑对于复杂系统能力的把握，抽象化能力的提高，可以帮助更加深刻的理解系统的 safeness 和 liveness 的行为，有非常大的帮助，特别是对现在各种大规模 AI 系统和应用在社会各个领域的普及，对系统可靠性，安全性的理论和算法理解，显得比任何时候，都更加重要。

1. 一个简单的例子

我们从指定一个以 x 等于 0 开始并永远将 x 递增 1 的系统开始。
在传统的编程语言中，这可以写成:

initially x = 0 ;

loop forever x := x + 1 end loop

一个 TLA 的规约可以定义如下。

$\Pi \triangleq$ (x = 0)　　最开始，x 的初值为 0。

$\wedge \Box[x' = x + 1]_x$

下一个状态 x' 的值永远（Always）等于当前值 x 加 1。

我们暂时不讨论那个下标 x 的含义。

$\wedge WF_x(x' = x + 1)$

暂时不讨论 WF 的含义。

随着规约变得越来越复杂，我们需要更好的规则的编写方法。我们使用带有 $\wedge$ 和 $\vee$ 的规则列表来表示“并”和“或”的关系。另外，我们使用缩进来消除括号。因此上述规约 Π 的定义可以写成:

$\Pi \triangleq \wedge$ (x = 0)

$\wedge \Box[x' = x + 1]_x$

$\wedge WF_x(x' = x + 1)$

2. TLA 范式 (Formula)

一个 TLA 的范式是一个关于一个系统（或者进程）行为（Behavior）的真（TRUE）或者假（FALSE）的判断命题。一个行为通过一个状态序列（States）来描述。对于上面的规则 Π 而言，对于一个行为，如果其第 i^{th} 个状态，对系统的变量 x 赋值为 $i-1$ 的行为 $(i = 1, 2, ...)$，Π 命题的值为真。

系统是真实的；系统的行为是数学抽象化的对象。要判断一个系统 S 是否满足范式 Π，我们首先需要有一个方法，把系统 S 的执行

表达成为一个离散的状态行为。给定一个这样的基于状态的行为表达，我们说，一个系统 S 满足范式 Π(或者 S 实现了这个规约) 当且仅当 (iff) 范式 Π 对于系统 S 执行中的任何可能的状态行为都为真。

3. 另外一个例子

接下来，我们定义一个系统，其两个变量 x 和 y 最开始都为 0，然后持续的分别将 x 和 y 递增 1。系统的每一步（Step）是将 x 或 y 递增（但不同时递增）。两个变量会以任意顺序递增，但每个变量都会无限多次的 (Infinitely Often) 的递增。如果用一个常规的编程语言来表示，可以如下：

initially x = 0, y = 0 ;
cobegin
loop forever x := x+ 1 end loop k
loop forever y := y + 1 end loop
coend

对应于上面系统的 TLA 规约 Φ 可以定义如下。为简单起见，我们首先定义两个规范 χ 和 γ，然后基于这两个规范，定义 Φ。

$\chi \triangleq \wedge\ x' = x + 1$ x 递增
$\quad\ \wedge\ y' = y$ y 不变。
$\gamma \triangleq \wedge\ y' = y + 1$ y 递增
$\quad\ \wedge\ x' = x$ x 不变。
$\Phi \triangleq \wedge\ (x = 0) \wedge (y = 0)$ 最开始，x 和 y 都为 0
$\quad\ \wedge\ \Box[\chi \vee \gamma]_{<x,y>}$ 每一步是 χ 或者是 γ
$\quad\ \wedge$ WF<x,y>(χ) $\wedge$ WF<x,y>(γ) 这表示存在无限多步的 χ 或者 γ

规范 χ 和 γ 在 TLA 里被称作 “动作/行为”。一个 Action 是否会在一个由新旧状态组成的步骤之间发生，可以为真或者假。旧状

态通常由非 primed 变量（例如，x，y，z）来表示；新状态通常由 primed 变量（例如，x', y', z'）来表示。

4. 实现和 Stuttering[1]

我们说一个 TLA 规约 (Specification) 或者规范 (Formula) F 实现了另外一个规约/规范 G，当且仅当任何一个满足 F 的系统也同时满足 G。这个论断是正确的，因为如果所有满足 F 的行为也满足 G，这意味着 $F \implies G$。一个规范被称为是完全有效的 (Valid)，当且仅当其对应的一个系统的所有行为都被满足。所有的行为意味着这个系统的所有的状态序列，而非这个系统的某个特殊的执行部分。因此，如果 $F \implies G$ 是有效的，那么 F 实现了 G。实现是一种内嵌的行为。

一个可以持续的对变量 x 和 y 递增的系统包含可持续的增加 x。因此，规约 Φ 实现了规约 Π。这意味着每个满足 Φ 规约的行为也同时满足 Π 规约。满足 Φ 的行为允许只递增 y 的值而 x 的值不变。因此，Π 必须允许 x 变量不变的情况。这其实就是为什么我们在规范中引入对于变量的下标符号 $[N]_x$ 的原因。在 TLA 里，对于任意一个行为动作 Action(一个包含了常量，变量和表示变量下一个状态的 primed 变量的布尔表达式)$\mathcal{A}$ 和任意一个状态函数 (一个只包含了常量和变量的布尔表达式)f，我们定义：

$$[\mathcal{A}]_f \triangleq \mathcal{A} \vee (f' = f)$$

其中 f' 是把表达式 f 里的变量都做变换得到的[2]。因此，一个 TLA 逻辑里的步骤[3]满足一个规范 $\mathcal{A}$，当且仅当这一步满足 $\mathcal{A}$ 或者这个规范中的变量的值都保持原封不动。$\Box[\mathcal{A}]_e xpf$ 表达的是对于一个系统的任何/所有的行为步骤，每一步都要么满足 $[\mathcal{A}]$ 或者该 $[\mathcal{A}]$ 牵扯到的变量的值不被更新。因此，规约 Π 里的 $\Box[x' = x + 1]_e xpx$

[1]这个词在英文里是“口吃”的意思，用在 TLA 中是想表达重复，循环的语义。可以理解为 CSP 中的递归 recursive 的含义。即系统的一个动作发生后，状态不变，停留在之前的状态上。

[2]可以简单直接的理解为：如果 $f' \neq f$，就是变量的值不变。

[3]两个相邻的状态，我们称之为一步。

确实允许一个 x 的值不变的行为步骤。这样的步骤我们称之为 stuttering 步骤。

从数学的角度，方程 $x^2 = x + 1$ 不是说一个世界只含有 x；可以理解为含有任何可能描述这个世界的变量，例如，包括 x, y, z。方程 $x^2 = x + 1$ 只是没有涉及 y 和 z 而已[4]。同样的道理，范式 Π 是关于一个状态序列的断言，每一个状态是对所有的变量赋值，而非仅仅是针对 x。因此 Π 是对一个对变量 x 能够产生变化的系统的描述。但是一个系统的行为其实包括了还有这个系统的宇宙的历史，所以作为一个明智的规范，Π 要允许 stuttering 的步骤，从而使得宇宙或者系统的其他变量在变化，但 x 不变。

同样的道理，规范 Φ 允许在步骤之间，$< x, y >$ 的值不发生变化。换言之，x 和 y 的值都不变。因此如果我们只是观察系统中的 x 和 y，我们无法察觉这样的状态步骤发生过。

Stuttering 步骤使得认为一个系统是有限的生命周期变得没有必要。一个停下来的系统执行过程可以理解为是一个无限不停止的行为，只是描述这个系统的变量在经过有限的步骤后停止更新了。当一个系统停下来时，并不意味着整个宇宙到了尽头。因此，在 TLA 里，对于一个行为，我们认为总是无限的状态序列的。

5. 公平性 (Fairness)

公式 $\Box[x' = x + 1]_x$ 允许任意多的步骤保持 x 不变。事实上，这个规范可以满足 x 永远都不会改变的行为。但是，我们通常希望能够无限多次 (Infinitely Many) 地递增 x，因此我们的规范必须排除 x 仅递增有限次数的行为。在 TLA 里，这是通过 WF(Weak Fairness) 公式来完成的。

我们说，一个 TLA 的 Action 动作 $\mathcal{A}$ 在状态 s 下是“使能的”或者“可触发的” (Enabled)，当且仅当系统的行为序列中可能存在着一个状态 t 是 s 的一个下一个后续状态，即 $<$ 上一个状态

[4] y 和 z 的值在这个方程里不变。

s, 下一个状态 $t>$，并且这个二元组可以满足 $\mathcal{A}$ 这个动作规范[5]。$WF_f(\mathcal{A})$ 表示一个这样的系统行为，如果存在这样的一个 TLA 动作 $\mathcal{A} \wedge (f' \neq f)$，一旦变成是 “使能的”，“可触发的”，就保持着，并且永远是 “使能的”，“可触发的”，我们认为，那么这个系统就会出现无限多次（Infinitely Many) 的 $\mathcal{A} \wedge (f' \neq f)$ 步骤。换言之，如果一旦出现一个 Action 动作的可能性并且一直保持着这种系统会执行这个动作的可能性，那么就一定会出现和发生，或者说，被执行无限多次。

任何整数都可以加 1 以产生不同的整数。因此，TLA 动作 $(x' = x+1) \wedge (x' \neq x)$ 在 x 是整数的情况下，任何时候都是可能加 1 的，可以被触发的。规范 $(x = 0) \wedge \Box[x' = x + 1]_x$ 意味着 x 的初值为 0 而且每一步 x 要么递增或者不变，这意味着 x 确实是一个整数。因此，这个范式蕴含着 $(x' = x + 1) \wedge (x' \neq x)$ 是一直 (永远) 使能的。因此，第一节中的范式 Π 中的 $WF_x(x' = x + 1)$ 意味着可以出现无限多次的 $(x' = x + 1) \wedge (x' \neq x)$。因此，范式 Π 确保了一个系统的变量 x 是无限频繁的被递增的。

依此类推，$x = 0 \wedge \Box[\chi \wedge \gamma]_{<x,y>}$ 意味着 x 永远是一个整数，因此 $\chi \wedge (< x, y >' \neq < x, y >$ 是一直可能的，使能的。因此，Φ 规约蕴含着 x 可以被无限次频繁的递增。另外，因为 Φ 的行为也同时满足 Π，因此 $\Phi \implies \Pi$ 是成立的。

WF 表示弱公平性。TLA 规约另外也定义了强公平性 (Strong Fairness)$SF_f(\mathcal{A})$，其中 f 是一个状态函数，$\mathcal{A}$ 是一个 TLA 动作范式。强公平性范式表示，如果 $\mathcal{A} \wedge (f' \neq f)$ 在一个无限的系统行为里存在着无限频繁多次的可能性[6]，那么 $\mathcal{A} \wedge (f' \neq f)$ 就一定会出现无限多次。如果一个动作范式在某个时间点变得永远可能，那么也满足 “无限经常” 的条件。因此，强公平性 $SF_f(\mathcal{A})$ 中蕴含了 $WF_f(\mathcal{A})$[7]。

[5] 可以理解为是 $\mathcal{A}$ 使得系统的状态从 s 转换到了 t 状态。

[6] 换一个角度，可以理解为：或者只要不是完全不可能。

[7] 关于强公平性和弱公平性的解释，可以参阅 Hillel Wayne 关于公平性的文章。

WF 和 SF 范式的下标（包括 $\Box[N_f]$）使得从研发上无法分清楚算法一个范式发生了 stuttering 步骤。在实践中，每当我们书写 $WF_f(\mathcal{A})$ 或者 $SF_f(\mathcal{A})$ 时，我们约定 $\mathcal{A})$ 蕴含着 $f' \neq f$，因此 $\mathcal{A})$ 默认为会改变状态函数 f 的值。

6. 隐藏 (Hiding)

如果一个行为序列，存在一系列具体的 y 值被赋值给变量 y，然后产生的与 y 无关的行为满足范式 Φ，我们说这个行为满足范式 $\exists y : \Phi$。反之亦然[8](这个定义是一个近似正确的说法；关于准确的定义，请参阅文献 [2]。) 时序存在量词 $\exists y$ 是在 TLA 中关于“隐含” 变量 y 的形式表达方法。如果我们在一个 x 和 y 重复递增的范式中隐藏变量 y，我们其实就得到了一个只是 x 重复递增的 T 范式。因此，通过隐含范式 Φ 中变量 y 得到的范式与 Π 应该是等价的。确实，$\exists y : \Phi$ 与 Π 是等价的。换言之，$(\exists y : \Phi) \equiv \Pi$ 是成立的。

7. 组合 (Composition)

如第 3 节所定义的两个规范 χ 和 γ，我们定义：

$\Pi_x \triangleq (x = 0) \wedge \Box[\chi]_x \wedge WF_{<x,y>}(\chi)$

$\Pi_y \triangleq (y = 0) \wedge \Box[\gamma]_y \wedge WF_{<x,y>}(\gamma)$

通过简单的计算可以得出，如果 x 和 y 都是整数，那么 $[\chi]_x \wedge [\gamma]_y$ 与 $[\chi \vee \gamma]_{<x,y>}$ 是等价的。从而我们得出，时序逻辑 $\Pi_x \wedge \Pi_y$ 与 Φ 是等价的。我们可以将 Π_x 和 Π_y 解释为在一个有两个变量 x 和 y 的程序里，关于两个进程的规范说明，一个是重复增加变量 x，另一个重复增加变量 y。这两个进程的组合成为一个含有两个变量的程序，持续的递增 x 和 y。因此，这个程序符合规范 Φ。

通常而言，系统 S 的规范 F 描述了 S 在其中正确运行的宇宙的行为（代表运行的历史）。系统 T 的规范 G 描述了 T 在其中正确运行的宇宙的行为。把 S 和 T 合并意味着确保 S 和 T 都正确运行。

[8] 可以理解为把变量 y 具体实例化了，y 已经变成一个不变量了。

两个系统都运行的宇宙的行为可以通过范式 $F \wedge G$ 来表示。组合是一种并 (conjunction) 的关系。

8. 确保 (Guarantee)

一个“假定/确保范式”是表示一种情形–如果一个系统所处的环境[9]在正确的运行，这个系统就一定正确。假设 M 是我们要考察的系统的规约范式，E 是环境的规约范式。我们可能会认为“假定/确保范式”的语义是 $E \implies M$，即要么是环境 E 没有被满足 (环境没有按照规范的要求正确运行)，或者是 M 本身被满足了 (系统正确的按照规范运行了)。TLA 里，“假定/确保范式”的语义要更加强一些，其表示为：$E \stackrel{+}{\rightarrow} M$，其语义包含了 E 蕴含了 M，并且除非 E 不被满足，否则 M 不会自己出错[10]。关于 $E \stackrel{+}{\rightarrow} M$ 的严格定义请参阅文献 [1]。

9. TLA 概貌 (All of TLA)

TLA 建立在一个关于行为动作逻辑 (Logic of Action) 的基础上，行为动作逻辑是一个逻辑语言，可以用来表达谓词 (Predicates)，状态函数 (State Functions) 和行为动作 (Actions)，以及对其进行推理的逻辑。谓词是包含常量和变量的布尔表达式；一个状态函数是一个包含常量和变量的非布尔表达式；一个动作是一个布尔表达式，包含常量，变量和 primed 变量。完整的规范语言 TLA+ 可以理解为是 TLA 的扩充[11]。

语法上，一个 TLA 范式有如下几种形式：

P	$\Box[\mathcal{A}]_f$	$\Box F$	$\exists x : F$	
$\neg F$	$F \wedge G$	$F \vee G$	$F \Rightarrow G$	$F \equiv G$
$\mathrm{WF}_f(\mathcal{A})$	$\mathrm{SF}_f(\mathcal{A})$	$F \stackrel{+}{\triangleright} G$	$\Diamond F$	$F \rightsquigarrow G$

其中 P 是谓词，f 是状态函数，$\mathcal{A}$ 是动作，x 是变量，F 和 G 是

[9] 环境本身也是一个系统。这个理解与 CSP 对环境的理解是一致的。

[10] 单纯的 $E \implies M$ 算子里，如果 E 是 TRUE，M 自己可以是 FALSE。

[11] 主要是增加了对一阶逻辑表达能力的扩充。

TLA 范式。范式可以通过其他的 TLA 逻辑的成分来表示（所有布尔运算符都可以由 $\neg$ 和 $\wedge$ 组成）。布尔运算符与通常的命题逻辑的布尔运算的含义是一致的；其他的一些 TLA 运算符的含义如下：

P
一个行为序列满足 P，当且仅当一个状态行为序列的初始状态的值满足 P。

$\Box[\mathcal{A}]_f$
当且仅当对于行为序列的任何一步，都满足 $[\mathcal{A}]$，或者 f 所涵盖的变量不发生更新变化。

$\Box F$
永远为真。当且仅当对于行为序列的任何时候，变量的最后状态，都满足 F。

$\exists x : F$
隐含变量。当且仅当一个行为序列的 x 变量的值被赋值后，能满足范式 F。

$WF_f(\mathcal{A})$
$\mathcal{A}$ 具备弱公平性，当且仅当一个行为满足 $\mathcal{A} \wedge (f' \neq f)$ 是无限频繁的变得不可能，或者 $\mathcal{A} \wedge (f' \neq f)$ 是无限多次的发生[12]。

$SF_f(\mathcal{A})$
$\mathcal{A}$ 具备强公平性，当且仅当一个行为满足 $\mathcal{A} \wedge (f' \neq f)$ 是有限次

[12]这个逻辑等价于：如果不是无限频繁的变得不可能，就得到无限多次的可能发生。“如果不是无限频繁的变得不可能”等价于存在着一旦可能，就永远保持着“使能”(Enabled) 的状态。因此，与前面小节定义弱公平性时的定义是一致的：如果一旦出现可能并且永远保持这种可能性，那么就一定会无限多次的发生。

的可能，或者 $\mathcal{A} \wedge (f' \neq f)$ 是无限多次的发生[13]。

$F \overset{+}{\rightarrow} G$
确保为真。只要 F 为真，G 会确保为真。

$\Diamond F$
最终 (Eventually) 会为真。可以被定义为：$\neg\Box\neg F$[14]

$F \rightsquigarrow G$
只要 F 是真，G 迟早会为真。可以被定位为：$\Box(F \implies \Diamond G)$

10. 文献 (References)

1. Martin Abadi and Leslie Lamport. Conjoining specifications. ACM Transactions on Programming Languages and Systems, 17(3):507-534,May 1995.

2. Leslie Lamport. The temporal logic of actions. ACM Transactions on Programming Languages and Systems, 16(3):872-923, May 1994.

[13]这个逻辑等价于：如果是无限频繁的有可能，就一定无限多次的可能发生。因此，与前面小节定义强公平性时的定义是一致的。

[14]可以理解为：不会永远都为假。