第 5 章　线性时序逻辑

本章介绍 (命题) 线性时序逻辑 (Linear Temporal Logic, LTL), 一种适合描述 LT 性质的逻辑形式化方法. 本章首先定义线性时序逻辑的语法和语义, 并通过一些例子说明如何用线性时序逻辑描述重要的系统性质. 然后聚焦于 LTL 的基于 Büchi 自动机的模型检验算法. 该算法可以用来回答以下问题: 给定一个迁移系统 TS 和 LTL 公式 φ, 如何检验 φ 是否在迁移系统 TS 中成立?

5.1　线性时序逻辑述要

对于反应系统, 正确性依赖于系统执行 (不仅依赖于计算的输入和输出) 以及公平性问题. 对于处理这些问题, 时序逻辑具有形式化方法的优越性. 时序逻辑用一种允许引用反应系统的无穷行为的模态扩充命题逻辑或谓词逻辑的概念. 为表达关于执行中的状态标记之间的关系的性质, 即 LT 性质, 时序逻辑提供了非常直观而又具有数学严谨性的记法. 古代的哲学等领域就曾研究时序逻辑及相关的模态逻辑. 20 世纪 70 年代末, Pnueli 提出把时序逻辑应用到复杂计算机系统验证上.

本章将集中讨论*命题时序逻辑*, 即命题逻辑的时序模态扩充. 这些逻辑应与那些在谓词逻辑之上施加时序模态的一阶 (或高阶) 时序逻辑区分. 本书假设读者在某种程度上熟悉命题逻辑的基本原理. 在附录 A 的 A.3 节中, 可以找到本书所用记号的简介和汇总. 大多数时序逻辑中的基本时序模态包括以下运算符:

$\Diamond$　终将 (未来终究要)

$\Box$　总是 (现在和未来总是)

在时序逻辑中, 时间的底层特性可以是*线性的*或*分支的*. 在线性时间观点中, 时间的每一时刻都有单一的后继时刻; 而在分支时间观点中, 有一个树状分支结构, 时间在其中会被分裂为可选的方向. 本章考虑 LTL, 它是一种基于线性时间观点的时序逻辑. 第 6 章介绍 CTL (计算树逻辑), 它是一种基于分支时间观点的逻辑. 一些模型检验工具将 LTL (或略有不同的 LTL) 用作性质准述语言. 模型检验器 SPIN 是这些自动验证工具中的典型. LTL 的优点之一是不需要任何新机制就能施加公平性假设 (像强公平性和弱公平性等): 典型的公平性假设都可在 LTL 中描述. 可用 LTL 的算法在公平性约束下验证 LTL 公式. 但这不适用于 CTL.

在详细介绍 LTL 之前, 为了避免读者产生任何可能的困惑, 先简要说明形容词 “时序的”. 虽然它使人想起反应系统中的实时行为, 但这只在抽象意义下是正确的. 时序逻辑允许描述事件的相对顺序. 例如 “司机一旦踩下刹车, 汽车就停下来”, 或 “消息在发出后收到”. 但是, 时序逻辑*不*支持任何涉及事件准确计时的方法. 不能描述事实: 在刹车和实际停车之间至少有 3μs 的最小延迟. 在迁移系统中, 时序逻辑的基本模态不能给出迁移的持续

时间或状态的逗留时间. 这些模态只是允许描述在执行期间状态标记出现的顺序, 或允许估计某些状态标记在一个 (或所有) 系统执行中无限经常地出现. 于是, 可以说时序逻辑中的模态是*时间抽象的*.

本章要讨论的 LTL 可为一类同步系统表示计时, 这类系统中所有组件都以步骤锁定方式行进. 在这种设置下, 一个迁移对应单个时间单位的推进. 现在的时刻对应当前状态, 而下一时刻对应直接后继状态, 因而底层时间域是*离散的*. 换言之, 就是设想在时间点 $0, 1, 2, \cdots$ 可观察到系统行为. 第 9 章将讨论如何利用*连续时间域*来处理异步系统中的实时约束, 将介绍 CTL 的时控版本, 称作时控 CTL. 表 5.1 总结了本书所讨论的主要时序逻辑的不同点.

表 5.1 本书中的时序逻辑分类

逻　辑	线性时间 (基于路径的)	分支时间 (基于状态的)	实时要求 (连续时间域)
LTL	√		
CTL		√	
时控 CTL		√	√

5.1.1 语法

本节描述语法规则, 根据这些规则可以构造 LTL 公式. LTL 公式的基本要素是原子命题 (状态标记 $a \in \mathrm{AP}$), 例如合取 $\wedge$ 和取非 $\neg$ 这样的布尔联结词, 以及两个基本时序模态 $\bigcirc$ (读作 "下一步") 和 U (读作 "直到"). 原子命题 $a \in \mathrm{AP}$ 在迁移系统中表示状态标记 a. 通常情况下, 原子 (命题) 就是关于控制变量之值 (如程序图中的位置) 或程序变量之值的断言, 如 $x > 5$ 或 $x \leqslant y$. 模态 $\bigcirc$ 是一元前缀运算符, 需要一个 LTL 公式作为操作数. 如果 φ 在下一 "步" 成立, 则公式 $\bigcirc\varphi$ 在当前时刻成立. 模态 U 是一个二元中缀运算符, 需要两个 LTL 公式作为操作数. 如果存在未来的某一刻 φ_2 成立, 而 φ_1 一直成立到未来这一时刻, 则公式 $\varphi_1 \,\mathsf{U}\, \varphi_2$ 在当前时刻成立.

定义 5.1 LTL *的语法*

原子命题集合 AP 上的 LTL 公式根据以下语法[①]构造:

$$\varphi ::= \mathrm{true} \mid a \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \bigcirc\varphi \mid \varphi_1 \,\mathsf{U}\, \varphi_2$$

其中 $a \in \mathrm{AP}$. ■

本书在大多数情况下不明确给出命题集合 AP, 因为它可从上下文得到或者可以定义为 LTL 公式中的原子命题的集合.

运算符的优先级如下. 一元运算符比二元运算符更优先. $\neg$ 和 $\bigcirc$ 具有同样的优先级. 时序运算符 U 优先于 $\wedge$、$\vee$ 和 $\rightarrow$. 适当的时候可以省略括号, 例如, 可以用 $\neg\varphi_1 \,\mathsf{U}\, \bigcirc\varphi_2$ 代替 $(\neg\varphi_1) \,\mathsf{U}\, (\bigcirc\varphi_2)$. 运算符 U 是右结合的, 例如, $\varphi_1 \,\mathsf{U}\, \varphi_2 \,\mathsf{U}\, \varphi_3$ 表示 $\varphi_1 \,\mathsf{U}\, (\varphi_2 \,\mathsf{U}\, \varphi_3)$.

① 就是使用更灵活的巴科斯范式 (Backus Naur Form, BNF). 具体地说, 把非终止符看作派生词 (公式) 和规则中的标记. 此外, 还使用在语法中未出现的括号, 例如在 $a \wedge (b \,\mathsf{U}\, c)$ 中. 这种简化的记法通常被称为*抽象语法*, 用于确定某些逻辑 (或其他演算项) 的形成语法.

使用布尔联结词 $\wedge$ 和 $\neg$, 就可实现命题逻辑的全部能力. 其他布尔联结词, 如析取 $\vee$、蕴涵 $\rightarrow$、等价 $\leftrightarrow$ 和奇偶运算符 (异或运算符) $\oplus$, 可按如下方法导出:

$$\begin{aligned}
\varphi_1 \vee \varphi_2 &\stackrel{\text{def}}{=} \neg(\neg\varphi_1 \wedge \neg\varphi_2) \\
\varphi_1 \rightarrow \varphi_2 &\stackrel{\text{def}}{=} \neg\varphi_1 \vee \varphi_2 \\
\varphi_1 \leftrightarrow \varphi_2 &\stackrel{\text{def}}{=} (\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1) \\
\varphi_1 \oplus \varphi_2 &\stackrel{\text{def}}{=} (\varphi_1 \wedge \neg\varphi_2) \vee (\varphi_2 \wedge \neg\varphi_1) \\
&\vdots
\end{aligned}$$

U 运算符也可表示时序模态 $\Diamond$ (“终将”, 将来某一时刻) 和 $\Box$ (“总是”, 从现在到永远总是), 方法如下:

$$\Diamond\varphi \stackrel{\text{def}}{=} \text{true}\ \mathsf{U}\ \varphi \qquad \Box\varphi \stackrel{\text{def}}{=} \neg\Diamond\neg\varphi$$

因此, 可以得到 $\Diamond$ 和 $\Box$ 如下的直观意义. $\Diamond\varphi$ 确保 φ 最终将取真. $\Box\varphi$ 成立当且仅当 $\neg\varphi$ 不会终将成立. 这相当于 φ 从现在开始总是成立.

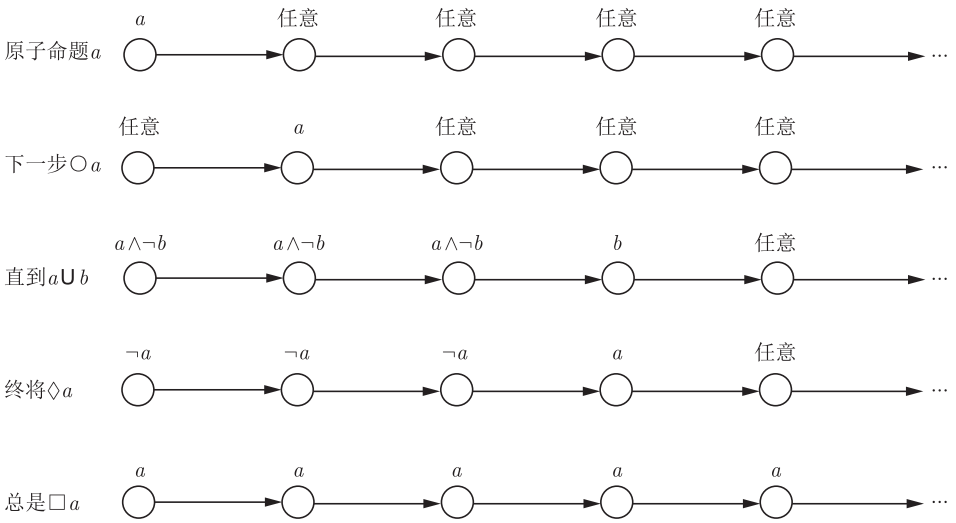


图 5.1　时序模态的直观语义

图 5.1 给出了当模态的操作数出自原子命题 $\{a, b\}$ 时, 时序模态的直观含义. 左边列出了一些 LTL 公式, 而右边描述了状态序列 (即路径).

通过组合时序模态 $\Diamond$ 和 $\Box$, 可得到新时序模态. 例如, $\Box\Diamond a$ (“总是终将 a”) 描述以下 (路径) 性质: 对于任何时刻 j, 存在时刻 $i \geqslant j$, 此刻访问的是 a 状态. 这相当于无限次访问 a 状态. 双模态 $\Diamond\Box a$ 表示从某一时刻 j 开始, 只访问 a 状态. 所以:

$\Box\Diamond\varphi$ 的含义是 “无限经常 φ”
$\Diamond\Box\varphi$ 的含义是 “终将总是 φ”

在给出 LTL 正式的语义之前, 先看一些例子.

例 5.1 互斥问题的性质

考虑两个并发进程 P_1 和 P_2 的互斥问题. 进程 P_i 由 3 个位置组成: ① 非关键节段; ② 当进程要进入关键节段时进入的等待节段; ③ 关键节段. 令命题 wait_i 和 crit_i 分别表示进程 P_i 处于等待节段和关键节段.

P_1 和 P_2 永不同时进入关键节段, 该安全性质可用 LTL 公式描述为

$$\Box(\neg\text{crit}_1 \vee \neg\text{crit}_2)$$

这个公式表明, 总是 ($\Box$) 至少有一个进程不在关键节段 ($\neg\text{crit}_i$).

每一进程 P_i 无限经常地处于关键节段, 这一活性要求可由 LTL 公式描述为

$$(\Box\Diamond\text{crit}_1) \wedge (\Box\Diamond\text{crit}_2)$$

弱化的形式, 即每一等待进程终将进入关键节段 (即无饥饿). 可以通过使用额外的命题 wait_i 描述为

$$(\Box\Diamond\text{wait}_1 \to \Box\Diamond\text{crit}_1) \wedge (\Box\Diamond\text{wait}_2 \to \Box\Diamond\text{crit}_2)$$

利用原子命题 wait_i 和 crit_i, 这些公式最终仅与位置 (即程序计数器的值) 相关. 然而命题也与程序变量相关. 例如, 可以用二进制信号 y 解决互斥问题, 公式

$$\Box((y = 0) \to \text{crit}_1 \vee \text{crit}_2)$$

表明只要信号 y 的值为 0, 就有一个进程处于关键节段. ■

例 5.2 哲学家就餐问题的性质

哲学家就餐问题 (见例 3.2) 的无死锁性质, 可用 LTL 公式描述为

$$\Box\neg(\bigwedge_{0\leqslant i<n} \text{wait}_i \wedge \bigwedge_{0\leqslant i<n} \text{occupied}_i)$$

此处, 假设有 n 位哲学家和 n 只筷子, 下标为 0 到 $n-1$. 原子命题 wait_i 意味着哲学家 i 已经手握一只筷子, 正在等待他左面或右面的筷子. 同样, occupied_i 表示第 i 只筷子正被占用. ■

例 5.3 交通灯问题的性质

对于有 “绿灯”“红灯” 和 “黄灯” 3 个阶段的交通灯, 活性性质 $\Box\Diamond\text{green}$ 表示交通灯可以无限次变成绿灯. 通过表示任一阶段的前驱阶段的 LTL 公式的合取, 可得到关于交通灯的周期及次序的描述. 例如, 规定 “一旦处于红灯阶段, 交通灯不能立即变成绿灯”, 可由 LTL 公式描述为

$$\Box(\text{red} \to \neg\bigcirc\text{green})$$

规定 “一旦处于红灯阶段, 总是在变为黄灯一段时间后终将变为绿灯” 可以表示为

$$\Box(\text{red} \to \bigcirc(\text{red}\ \mathsf{U}\ (\text{yellow} \wedge \bigcirc(\text{yellow}\ \mathsf{U}\ \text{green}))))$$

■

类似于 “每个请求终将得到响应” 的进程性质可以用下面的公式描述:

$$\Box(\text{request} \to \Diamond\text{response})$$

注记 5.1　公式的长度

令 $|\varphi|$ 表示 LTL 公式 φ 的长度, 它等于 φ 中运算符的个数. 对 φ 的结构使用归纳法, 可以很容易地定义它. 例如, 公式 true 和 $a \in \mathrm{AP}$ 的长度为 0, 公式 $\bigcirc a \vee b$ 和 $a \vee \neg b$ 的长度为 2, 公式 $(\bigcirc a)\ \mathsf{U}\ (a \wedge \neg b)$ 的长度为 4. 在本书中, 多半需要渐近大小 $\Theta(|\varphi|)$. 因此, 无论在确定长度时是否考虑导出的布尔运算符 $\vee$、$\to$ 等, 以及是否考虑导出的时序模态 $\Diamond$ 和 $\Box$, 都没有关系. ■

5.1.2　语义

LTL 公式表示路径 (事实上是它们的迹) 的性质. 这意味着一个路径可以满足或不满足一个 LTL 公式. 为了精准地表述一个路径何时满足 LTL 公式, 按如下方法进行. 首先, LTL 公式 φ 的语义定义为语言 $\mathrm{Words}(\varphi)$, 它包含了字母表 2^{AP} 上所有满足 φ 的无限单词. 也就是说, 给每个 LTL 公式关联唯一一个 LT 性质. 然后, 把语义扩展为对迁移系统的路径和状态的解释.

定义 5.2　LTL 的语义 (单词上的解释)

令 φ 为 AP 上的一个 LTL 公式. φ 诱导的 LT 性质为

$$\mathrm{Words}(\varphi) = \left\{\sigma \in (2^{\mathrm{AP}})^{\omega} \mid \sigma \models \varphi\right\}$$

其中, 满足关系 $\models \subseteq (2^{\mathrm{AP}})^{\omega} \times$ LTL 是具有图 5.2 所示性质的最小关系. ■

$\sigma \models \text{true}$		
$\sigma \models a$	iff	$a \in A_0$ (即 $A_0 \models a$)
$\sigma \models \varphi_1 \wedge \varphi_2$	iff	$\sigma \models \varphi_1$ 且 $\sigma \models \varphi_2$
$\sigma \models \neg\varphi$	iff	$\sigma \not\models \varphi$
$\sigma \models \bigcirc\varphi$	iff	$\sigma[1..] = A_1A_2A_3\cdots \models \varphi$[①]
$\sigma \models \varphi_1\ \mathsf{U}\ \varphi_2$	iff	$\exists j \geqslant 0.\, \sigma[j..] \models \varphi_2$ 且 $\forall 0 \leqslant i < j.\, \sigma[i..] \models \varphi_1$

图 5.2　2^{AP} 上的无限单词的 LTL 语义 (可满足关系 $\models$)

此处, 对于 $\sigma = A_0A_1A_2\cdots \in (2^{\mathrm{AP}})^{\omega}$, $\sigma[j..] = A_jA_{j+1}A_{j+2}\cdots$ 是 σ 的从第 $j+1$ 个符号 A_j 开始的后缀.

注意, 在 LTL 公式语义的定义中, 单词片段 $\sigma[j..]$ 不能用 A_j 代替. 例如, 对于公式 $\bigcirc(a\ \mathsf{U}\ b)$, 为了能够在下一步中得到子式 $a\ \mathsf{U}\ b$ 的真值, 只能考虑后缀 $A_1A_2A_3\cdots$.

对于导出运算符 $\Diamond$ 和 $\Box$, 预期的结果为

$$\begin{aligned} \sigma \models \Diamond\varphi \quad &\text{iff} \quad \exists j \geqslant 0.\, \sigma[j..] \models \varphi \\ \sigma \models \Box\varphi \quad &\text{iff} \quad \forall j \geqslant 0.\, \sigma[j..] \models \varphi \end{aligned}$$

① 在形式语言中, [1..] 表示从 1 开始递增. $[i..]$ 和 $[j..]$ 等的含义与之类似.

从 $\Diamond$ 的定义和 U 的语义可直接得到关于 $\Diamond$ 的命题. 由

$$\begin{aligned}\sigma \models \Box\varphi = \neg\Diamond\neg\varphi \quad &\text{iff} \quad \neg\exists j \geqslant 0.\, \sigma[j..] \models \neg\varphi \\ &\text{iff} \quad \neg\exists j \geqslant 0.\, \sigma[j..] \not\models \varphi \\ &\text{iff} \quad \forall j \geqslant 0.\, \sigma[j..] \models \varphi\end{aligned}$$

可得关于 $\Box$ 的命题. 现在可导出 $\Diamond$ 和 $\Box$ 的组合的语义:

$$\begin{aligned}\sigma \models \Box\Diamond\varphi \quad &\text{iff} \quad \overset{\infty}{\exists} j.\, \sigma[j..] \models \varphi \\ \sigma \models \Diamond\Box\varphi \quad &\text{iff} \quad \overset{\infty}{\forall} j.\, \sigma[j..] \models \varphi\end{aligned}$$

其中, $\overset{\infty}{\exists} j$ 表示 "对无限多个 $j \in \mathbb{N}$", 即 $\forall i \geqslant 0.\, \exists j \geqslant i$; 而 $\overset{\infty}{\forall} j$ 表示 "对几乎所有 $j \in \mathbb{N}$", 即 $\exists i \geqslant 0.\, \forall j \geqslant i$. 下面证明第一个命题, 第二个命题的证明与之类似.

$$\begin{aligned}\sigma \models \Box\Diamond\varphi \quad &\text{iff} \quad \forall i \geqslant 0.\, \sigma[i..] \models \Diamond\varphi \\ &\text{iff} \quad \forall i \geqslant 0.\, \exists j \geqslant i.\, \sigma[j..] \models \varphi \\ &\text{iff} \quad \overset{\infty}{\exists} j.\, \sigma[j..] \models \varphi\end{aligned}$$

接下来, 确定 LTL 公式对于迁移系统的语义. 根据 LT 性质的可满足关系 (见定义 3.7), 如果所有从 s 开始的路径都满足 φ, 则 LTL 公式 φ 在状态 s 处成立. 如果迁移系统 TS 满足 LT 性质 Words(φ), 即, 如果 TS 的所有起始路径 (从初始状态 $s_0 \in I$ 开始的路径) 都满足 φ, 则称 TS 满足 φ.

不失一般性, 可假设迁移系统 TS 没有终止状态 (若有, 则可引入陷阱状态). 因此, 可假设所有路径和迹是无限的. 作此假设仅为简化问题, 对于有限路径也可定义 LTL 语义. 注意, 语义与 TS 是否有限无关. 只有对于 5.2 节的模型检验算法才要求 TS 是有限的.

像对待 LT 性质一样, 在对 AP' 上的迁移系统 TS 定义 $\mathrm{TS} \models \varphi$ 时, 假设 φ 是原子命题在 $\mathrm{AP} = \mathrm{AP}'$ 中的 LTL 公式 (此处可以更自由一些, 准许 $\mathrm{AP} \subseteq \mathrm{AP}'$).

定义 5.3 LTL 在路径和状态上的语义

令 $\mathrm{TS} = (S, \mathrm{Act}, \to, I, \mathrm{AP}, L)$ 是没有终止状态的迁移系统, 令 φ 是 AP 上的 LTL 公式.

- 对于 TS 的无限路径片段 π, 满足关系定义为

$$\pi \models \varphi \ \text{ iff } \ \mathrm{trace}(\pi) \models \varphi$$

- 对于状态 $s \in S$, 满足关系 $\models$ 定义为

$$s \models \varphi \ \text{ iff } \ \forall \pi \in \mathrm{Paths}(s).\, \pi \models \varphi$$

- 若 $\mathrm{Traces(TS)} \subseteq \mathrm{Words}(\varphi)$, 则称 TS 满足 φ, 记作 $\mathrm{TS} \models \varphi$. ■

由定义 5.3 可直接得出

$$\begin{array}{lll}\mathrm{TS} \models \varphi & \text{iff} & \mathrm{Traces(TS)} \subseteq \mathrm{Words}(\varphi) \qquad \text{(* 定义 5.3 *)} \\ & \text{iff} & \mathrm{TS} \models \mathrm{Words}(\varphi) \qquad \text{(* LT 性质中 } \models \text{ 的定义 *)} \\ & \text{iff} & \forall \pi \in \mathrm{Paths(TS)},\, \pi \models \varphi \qquad \text{(* Words}(\varphi) \text{ 的定义 *)} \\ & \text{iff} & \forall s_0 \in I,\, s_0 \models \varphi \qquad \text{(* 定义 5.3 中状态的 } \models \text{ *)}\end{array}$$

因此, $\mathrm{TS} = \varphi$ 当且仅当对于 TS 的所有初始状态 s_0 都有 $s_0 \models \varphi$.

例 5.4　LTL 的语义

考虑由图 5.3 描述的迁移系统, 命题集为 $\mathrm{AP} = \{a, b\}$.

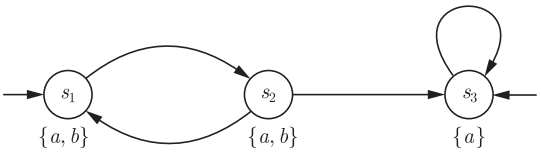


图 5.3　关于 LTL 语义的例子

例如, 因为所有状态都用 a 标记, 所以 TS 的迹是满足对所有 $i \geqslant 0$ 都有 $a \in A_i$ 的单词 $A_0A_1A_2\cdots$, 因此 $\mathrm{TS} \models \Box a$. 还可得 $s_i \models \Box a$, $i = 1, 2, 3$. 而且因为 $s_2 \models a \wedge b$ 且 s_2 是 s_1 的唯一后继, 所以 $s_1 \models \bigcirc(a \wedge b)$. 由 $s_3 \in \mathrm{Post}(s_2)$, $s_3 \in \mathrm{Post}(s_3)$, $s_3 \not\models a \wedge b$ 得 $s_2 \not\models \bigcirc(a \wedge b), s_3 \not\models \bigcirc(a \wedge b)$. 因为 s_3 是一个初始状态且 $s_3 \not\models \bigcirc(a \wedge b)$, 所以 $\mathrm{TS} \not\models \bigcirc(a \wedge b)$. 另一个例子是

$$\mathrm{TS} \models \Box(\neg b \to \Box(a \wedge \neg b))$$

它成立是因为 s_3 是唯一一个 $\neg b$ 状态, 而且到达 s_3 后再也不能离开 s_3, 同时 $a \wedge \neg b$ 在 s_3 处成立. 但是,

$$\mathrm{TS} \not\models b \,\mathsf{U}\, (a \wedge \neg b)$$

这是因为起始路径 $(s_1s_2)^\omega$ 不会访问 $a \wedge \neg b$ 成立的状态. 注意, 起始路径 $(s_1s_2)^*s_3^\omega$ 满足 $b \,\mathsf{U}\, (a \wedge \neg b)$. ■

注记 5.2　否定的语义

对于路径, $\pi \models \varphi$ 当且仅当 $\pi \not\models \neg\varphi$. 这是因为

$$\mathrm{Words}(\neg\varphi) = (2^{\mathrm{AP}})^\omega \setminus \mathrm{Words}(\varphi)$$

然而, 命题 $\mathrm{TS} \not\models \varphi$ 和 $\mathrm{TS} \models \neg\varphi$ 一般是不等价的. 而是 $\mathrm{TS} \models \neg\varphi$ 蕴涵 $\mathrm{TS} \not\models \varphi$. 注意,

$$\begin{array}{lll} \mathrm{TS} \not\models \varphi & \text{iff} & \mathrm{Traces}(\mathrm{TS}) \not\subseteq \mathrm{Words}(\varphi) \\ & \text{iff} & \mathrm{Traces}(\mathrm{TS}) \setminus \mathrm{Words}(\varphi) \neq \varnothing \\ & \text{iff} & \mathrm{Traces}(\mathrm{TS}) \cap \mathrm{Words}(\neg\varphi) \neq \varnothing \end{array}$$

因此, 一个迁移系统 (或是一个状态) 可能既不满足 φ 也不满足 $\neg\varphi$. 这是因为, 有可能存在路径 π_1 和 π_2 使得 $\pi_1 \models \varphi$ 和 $\pi_2 \models \neg\varphi$ (因此 $\pi_2 \not\models \varphi$). 此时, $\mathrm{TS} \not\models \varphi$ 且 $\mathrm{TS} \not\models \neg\varphi$ 成立.

为演示这种效果, 考虑图 5.4 中描绘的迁移系统. 令 $\mathrm{AP} = \{a\}$. 因为起始路径 $s_0(s_2)^\omega \not\models \Diamond a$, 所以 $\mathrm{TS} \not\models \Diamond a$; 另一方面, 因为起始路径 $s_0(s_1)^\omega \models \Diamond a$, 所以 $s_0(s_1)^\omega \not\models \neg\Diamond a$, 因此 $\mathrm{TS} \not\models \neg\Diamond a$. ■

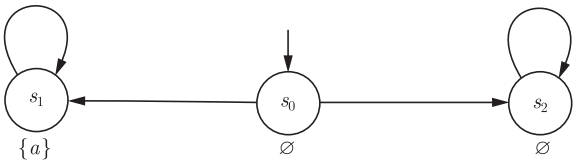


图 5.4　满足 TS $\not\models \Diamond a$ 和 TS $\not\models \neg\Diamond a$ 的迁移系统

5.1.3　准述性质

例 5.5　*重访基于信号的互斥*

考虑图 5.5 所描绘的迁移系统 $\mathrm{TS}_{\mathrm{Sem}}$, 它给出了互斥问题的一个基于信号的解决方案.

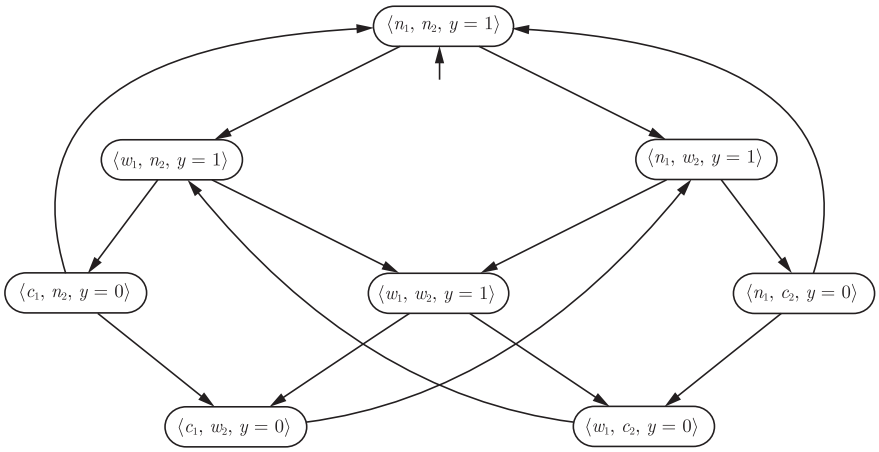


图 5.5　基于信号的互斥算法的迁移系统

每个形如 $\langle c_1, \cdot, \cdot\rangle$ 的状态都用 crit_1 标记, 而每个形如 $\langle \cdot, c_2, \cdot\rangle$ 的状态都用 crit_2 标记. 则有

$$\mathrm{TS}_{\mathrm{Sem}} \models \Box(\neg\mathrm{crit}_1 \vee \neg\mathrm{crit}_2) \quad \text{且} \quad \mathrm{TS}_{\mathrm{Sem}} \models \Box\Diamond\mathrm{crit}_1 \vee \Box\Diamond\mathrm{crit}_2$$

其中, 第一个 LTL 公式代表互斥性质, 第二个 LTL 公式表示两个进程中至少有一个无限次进入关键节段. 然而

$$\mathrm{TS}_{\mathrm{Sem}} \not\models \Box\Diamond\mathrm{crit}_1 \wedge \Box\Diamond\mathrm{crit}_2$$

因为缺少任何公平性假设, 不能保证进程 P_1 能够无限次进入关键节段. 它有可能一次也不能进入 (P_2 也有类似的情况). 同样的讨论可应用到证明

$$\mathrm{TS}_{\mathrm{Sem}} \not\models \Box\Diamond\mathrm{wait}_1 \rightarrow \Box\Diamond\mathrm{crit}_1$$

上. 这是因为, 原则上, 进程 P_1 一旦开始等待, 可能就会总也得不到它的机会. ■

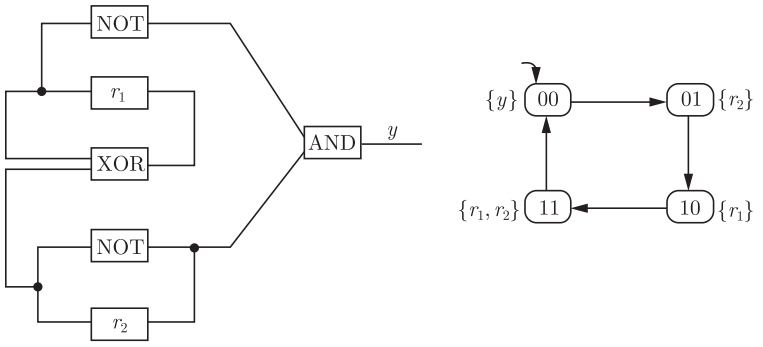
例 5.6　*模 4 计数器*

模 4 计数器可以由一个顺序电路 C 表示, 每当第 4 个周期时输出 1, 其余周期输出 0. C 没有输入位, 有一个输出位 y 和两个寄存器 r_1、r_2. 寄存器的赋值 $[r_1 = c_1, r_2 = c_2]$ 可用数值 $i = 2\cdot r_1 + r_2$ 确定. i 的值每一周期增加 1 (模 4). 用如下方法构造 C, 恰好当 $i = 0$

(因此, $r_1 = r_2 = 0$) 时输出位 y 置位. 迁移关系和输出函数由

$$\delta_{r_1} = r_1 \oplus r_2,\ \delta_{r_2} = \neg r_1,\ \lambda_y = \neg r_1 \wedge \neg r_2$$

给出. 图 5.6 给出了电路图和迁移系统 TS_C.



(a) 电路图　　(b) TS_C

图 5.6　模 4 计数器

令 $\mathrm{AP} = \{r_1, r_2, y\}$. 下面的命题可以从 TS_C 直接推出:

$$\begin{aligned} \mathrm{TS}_C &\models \Box(y \leftrightarrow \neg r_1 \wedge \neg r_2) \\ \mathrm{TS}_C &\models \Box(r_1 \rightarrow (\bigcirc y \vee \bigcirc\bigcirc y)) \\ \mathrm{TS}_C &\models \Box(y \rightarrow (\bigcirc\neg y \wedge \bigcirc\bigcirc\neg y)) \end{aligned}$$

如果假定只有输出变量 y (而不是寄存器的值) 可以由观察者感知, 那么 AP 的一个合适选择是 $\mathrm{AP} = \{y\}$. 至少每 4 个周期输出 1 的性质在 TS_C 中成立, 即, 有

$$\mathrm{TS}_C \models \Box\,(y \vee \bigcirc y \vee \bigcirc\bigcirc y \vee \bigcirc\bigcirc\bigcirc y)$$

这些输出是周期性产生的, 每当第 4 个周期时输出 1, 这个事实表示为

$$\mathrm{TS}_C \models \Box(y \rightarrow (\bigcirc\neg y \wedge \bigcirc\bigcirc\neg y \wedge \bigcirc\bigcirc\bigcirc\neg y))$$

■

例 5.7　通信通道

考虑两个通信进程间的一个单向通道、一个发送器 S 和一个接收器 R. 发送器 S 配置一个输出缓冲区 S.out, 接收器 R 配置一个输入缓冲区 R.in. 如果发送器 S 发送消息 m 给 R, 它就把消息插入它的输出缓冲区 S.out 中. 输出缓冲区和输入缓冲区是通过一个单向通道连接的. 接收器 R 接收消息, 并删除其输入缓冲区 R.in 中的消息. 在这里, 不考虑缓冲区的容量.

所考虑系统的示意图如下:

在下面的 LTL 描述中, 使用原子命题 $m \in S.\text{out}$ 和 $m \in R.\text{in}$, 其中 m 是任意消息. 利用 LTL 公式将下列非形式化要求形式化:

- 只要消息 m 在 S 的输出缓冲区中, m 终将被接收器用掉. 该要求可公式化为

$$\Box(m \in S.\text{out} \to \Diamond(m \in R.\text{in}))$$

路径 $s_1 s_2 s_3 \cdots$ 也满足上述性质, 其中 $s_1 \models m \in S.\text{out}$, $s_2 \models m \notin S.\text{out}$, $s_2 \models m \notin R.\text{in}$, 且 $s_3 \models m \in R.\text{in}$. 然而, 这样的路径表示一种诡异的行为, 即 S 的输出缓冲区中的消息 m (状态 s_1) 丢失 (状态 s_2) 后, 仍到达 R 的输入缓冲区 (状态 s_3). 事实上, 如果可靠 FIFO 通道满足以下 (更强的) 条件, 则这样的行为就是不可能的:

$$\Box(m \in S.\text{out} \to (m \in S.\text{out} \,\mathsf{U}\, m \in R.\text{in}))$$

该条件说明: 直到接收器 R 用掉 m, 消息 m 一直停留在 $S.\text{out}$ 中. 因为在一个 FIFO 通道中, 读和写不能同时发生, 所以可以将该要求公式化为

$$\Box(m \in S.\text{out} \to \bigcirc(m \in S.\text{out} \,\mathsf{U}\, m \in R.\text{in}))$$

- 如果假设没有消息在 $S.\text{out}$ 中出现两次, 那么 FIFO 通道的异步行为能确保性质 "消息不能同时出现在两个缓冲区中" 成立. 该要求可公式化为

$$\Box\neg(m \in S.\text{out} \wedge m \in R.\text{in})$$

- FIFO 通道的特点是: 根据先进先出的原则, 它们是保序的. 即, 如果 S 先把消息 m 送入其输出缓冲区 $S.\text{out}$, 然后送入 m', 则 R 将会在 m' 之前收到 m. 该要求可公式化为

$$\Box\Big(m \in S.\text{out} \wedge \neg m' \in S.\text{out} \wedge \Diamond(m' \in S.\text{out}) \\ \to \Diamond(m \in R.\text{in} \wedge \neg m' \in R.\text{in} \wedge \Diamond(m' \in R.\text{in}))\Big)$$

注意, 为确保在 m 之后把 m' 放入 $S.\text{out}$, 前提中需要 $\neg m' \in S.\text{out}$. 仅有 $\Diamond(m' \in S.\text{out})$, 则不能排除当 m 进入 $S.\text{out}$ 时 m' 已在发送缓冲区中的情形.

以上公式针对的是固定消息 m 和 m'. 为使上述性质能够针对所有消息, 要在所有消息 m 和 m' 上使用合取. 只要消息的字母是有限的, 就可以获得一个 LTL 公式. ■

例 5.8 *动态领袖选举*

(本例取自文献 [69].) 在当前的分布式系统中, 一些服务是由专用进程提供的, 例如地址的分配和登记、分布式数据库系统中的查询协调、时钟分布、令牌环网中令牌丢失后的再生、移动网络中拓扑更新后的初始化、负载平衡等. 系统中的许多进程通常都有提供这些服务的潜在可能. 然而, 为了确保相容性, 通常情况下, 任何时候都只允许一个进程实际提供指定的服务, 这个进程 (称为领袖) 实际上是选出的. 有时随便选个进程就可以了; 但是对于其他服务, 选出最有能力完成服务的进程是非常重要的. 在这里, 撇开特定能力, 并使用以进程 ID 为基础的排序, 意思是: 进程的 ID 越大, 它的能力越强.

假设以某种通信手段连接起来的进程的个数是一个有限数 $N > 0$. 像前面的例子一样, 进程之间的通信是异步的. 用图形描述为