带有时钟变量的线性时序逻辑与实时系统验证*

李广元, 唐稚松

(中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

E-mail: {ligy,cst}@ios.ac.cn

http://www.ios.ac.cn

摘要: 为了描述实时系统的性质和行为,10 多年来,各种不同的时序逻辑,如 Timed Computation Tree Logic,Metric Interval Temporal Logic 和 Real-Time Temporal Logic 等相继提出来.这些时序逻辑适于表示实时系统的性质和规范,但不适于表示实时系统的实现模型.这样,在基于时序逻辑的实时系统的研究中,系统的性质和实现通常是用两种不同的语言来表示的.定义了一个带有时钟变量的线性时序逻辑(linear temporal logic with clocks,简称 LTLC).它是由 Manna 和 Pnueli 提出的线性时序逻辑在实时情况下的一个推广.LTLC 既能表示实时系统的性质,又能很方便地表示实时系统的实现.它能在统一的语义框架中表示出从高级的需求规范到低级的实现模型之间的不同抽象层次上的系统描述,并且能用逻辑蕴涵来表示不同抽象层次的系统描述之间的语义一致性.LTLC 的这个特点将有助于实时系统的性质验证和实时系统的逐步求精.

关 键 词: 实时系统;时间自动机;线性时序逻辑;规范语言;系统描述语言;性质验证

中图法分类号: TP301 **文献标识码:** A

实时系统(real-time systems)[1~4]是一种带有时间约束的计算系统.核反应堆、飞行控制、铁路调度等方面的许多计算机控制系统都属于实时系统.这类系统的部分动作的完成是与时间有关的,即要满足一定的时间约束,如某动作要在 1 秒内完成等.对于这类系统,确保其正确性和可靠性是至关重要的.采用形式化方法对实时系统进行精确的描述与分析,是保证其正确性和可靠性的重要途径.

为了描述实时系统的性质和行为,10 多年来,各种不同的时序逻辑,如 Timed Computation Tree Logic[5], Metric Interval Temporal Logic[6],Timed Propositional Temporal Logic[7]和 Real-Time Temporal Logic[8]等相继提出来.它们作为实时系统的规范语言(specification languages),主要用于表示系统的性质(如安全性、可达性、公平性等),但不适合表示实时系统的实现模型.实时系统的实现模型通常是用时间转换系统(timed transition systems[3])、时钟转换系统(clocked transition systems[4])、时间自动机(timed automata[9])等系统描述语言(system description languages)来表示的,可是,这些系统描述语言又无法表示系统的一些重要性质(如安全性、可达性等).这样,在基于时序逻辑的实时系统的研究中,系统的性质和实现通常是用不同的语言来表示的.一方面,这种做法不利于系统性质的证明,因为性质和实现不是用同一语言表示的,它们不在同一推演系统下;另一方面,这种做法也不利于从规范设计到系统实现的逐步求精过程中的平滑过渡,因为从规范到实现不是在同一语言中进行的,其中必然要从一种语言跳到另一种语言.显然,若能将实时系统的性质和实现统一用一种逻辑语言表示出来,对于其性质证明和逐步求精则将是非常有益的.本文提出了一个带有时钟变量的线性时序逻辑 LTLC(linear temporal logic with clocks).它是由 Manna 和 Pnueli 提出的线性时序逻辑 LTL[10]在实时情况下的

* 收稿日期: 2000-07-20; 修改日期: 2001-06-20

基金项目: 国家自然科学基金资助项目(60073020);国家"九五"重点科技攻关项目(98-780-01-07-01);国家 863 高科技发展计划资助项目(863-306-ZT02-04-1)

作者简介: 李广元(1962—),男,陕西乾县人,博士,副研究员,主要研究领域为形式化方法,时序逻辑,实时系统;唐稚松(1925—),男,湖南长沙人,研究员,博士生导师,中国科学院院士,主要研究领域为时序逻辑,形式化方法,软件工程,软件体系结构.

一个推广.LTLC 既能作为规范语言用来表示实时系统的性质,也能作为系统描述语言来表示实时系统的实现模型.LTLC 的这个特点将有助于实时系统的性质验证和实时系统的逐步求精.

本文第 1 节给出了一些必要的概念和记号.第 2 节给出了 LTLC 的语法和语义.第 3 节给出了时间模块的概念(本文用时间模块来表示实时系统),并借助于 LTLC 的语义给出时间模块的语义.另外还给出了一个利用 LTLC 来证明实时系统性质的示例.第 4 节是一个简短的总结.

1 预备概念

在介绍时序逻辑 LTLC 之前,我们先给出一些必要的记号和概念.在本文中,$\Re$,$\Re^+$和 N 分别用于表示实数集、非负实数集和非负整数集.

定义 1.1. 称无穷序列$\{t_i\}_{i\in \mathrm{N}}$为一个时间序列,若

(1) $0=t_0<t_1<t_2<\ldots<t_n<\ldots$

(2) 序列$\{t_i\}_{i\in \mathrm{N}}$是一个发散序列,即 $\lim_{n\to\infty} t_n$ 无界.

定义 1.2. 称$\Re^+$上的实函数 $f(t)$是一个步函数,若存在时间序列$\{t_i\}_{i\in \mathrm{N}}$,使得对任意 $i\in$N,函数 $f(t)$在区间(t_i,t_{i+1})上都是常函数(即取值为一个常数).若步函数 $f(t)$的取值域为集合$\{0,1\}$的子集,则称 $f(t)$是一个布尔值步函数.

定义 1.3. 称 $\Re^+$ 上的实函数 $f(t)$是一个时钟函数,若存在时间序列$\{t_i\}_{i\in \mathrm{N}}$,使得

$$f(t)=\begin{cases}0 & \text{若}t=0\\ t-t_i & \text{若}t\in(t_i,t_{i+1}]\end{cases},$$

或者存在有穷序列 $t_0,t_1,t_2,\ldots,t_n$,使得 $0=t_0<t_1<t_2<\ldots<t_n$,且

$$f(t)=\begin{cases}0 & \text{若}t=0\\ t-t_i & \text{若}t\in(t_i,t_{i+1}]\\ t-t_n & \text{若}t>t_n\end{cases}.$$

显然,步函数和时钟函数都是分段连续函数,它们的每个不连续点都是孤立点.在每个不连续点处,它们的左极限和右极限都存在(起始点 $t=0$ 除外,在此点处无法定义左极限),且左极限等于函数在此点的值(即这些函数是左连续的).

定义 1.4. 设 f 是$\Re^+$上的一个步函数或时钟函数.定义一个伴随于 f 的函数 f': $\Re^+$ α $\Re$ 如下:

对任意 $t_0\in\Re^+$,有 $f'(t_0)=\lim_{t\to t_0+} f(t)$.

不难看出,函数 f'是一个右连续函数.在 f 的连续点上,函数 f 和 f'具有相同的值.

2 LTLC 的语法和语义

下面,我们给出线性时序逻辑语言 LTLC 的语法和语义.

2.1 LTLC的语法

LTLC 的字母包括以下符号和括号:

(1) 全局时钟: t;

(2) 刚性变量: $u,u_0,u_1,u_2,\ldots$

(3) 布尔型变量: $p,p_0,p_1,p_2,\ldots$

(4) 时钟变量: $x,x_0,x_1,x_2,\ldots$

(5) 常元符号: $\{\overline{m}\mid m\in \mathrm{N}\}$;

(6) 函数符号: +,−,*;

(7) 关系符号: =,≤;

(8) 联结词: ¬,∧;

(9) 量词: ∃;

(10) 时序符: ′(撇,新值符号),[](总是),U(直到).

定义 2.1. LTLC 中的项归纳定义如下:

$$e ::= t \mid \overline{m} \mid u \mid x \mid x' \mid -e \mid (e_1 + e_2) \mid (e_1 * e_2).$$

这里,t 是全局时钟,$\overline{m}$ 是一个常元符号,u 是一个刚性变量,x 是一个时钟变量.

定义 2.2. LTLC 的公式归纳定义如下:

$$j ::= p \mid p' \mid (e_1 = e_2) \mid (e_1 \le e_2) \mid \neg j \mid (j_1 \wedge j_2) \mid (\exists u\, j) \mid []j \mid (j_1 U j_2).$$

这里,u 是一个刚性变量,x 是一个时钟变量,e_1 和 e_2 是项.

其他一些常见的联结词,如∨(析取),⇒(蕴涵)和⇔(等价)等可以通过∧和¬定义出来;时序算子 <> (最终)可以通过[]和¬定义出来.一些常用的函数和关系符号(如≥,<,>等)也可以作为已有公式的缩写在 LTLC 中定义出来.

为方便起见,我们常用 $var(j)$ 和 $var(e)$ 来分别表示公式 j 和项 e 中出现的所有变量所组成的集合.如果 $var(e) \subseteq V$,则称 e 是变量集 V 上的项;类似地,如果 $var(j) \subseteq V$,则称 j 是变量集 V 上的公式.

2.2 LTLC 的语义

在 LTLC 中,时间是用非负实数来表示的.布尔型变量被解释为时间域 $\Re^+$ 上的一个布尔值步函数,时钟变量被解释为时间域 $\Re^+$ 上的一个时钟函数,刚性变量被解释为一个实数.若布尔型变量 p 和时钟变量 x 分别被解释为布尔值步函数 f_p 和时钟函数 f_x,则 p' 和 x' 将分别被解释为与 f_p 和 f_x 相伴随定义的两个函数 f'_p 和 f'_x.

定义 2.3. 设 V 是一个有穷的变量集合.称 $\Im := \langle I, s\rangle$ 是变量集 V 上的一个 LTLC-模型(或解释),若

(1) 对 V 中的任一刚性变量 u,映射 I 指派实数域 $\Re$ 中的一个数作为 u 的解释,即 $I(u) \in \Re$;

(2) 对 V 中的任一布尔型变量 p,映射 $\acute{o}$ 指派 $\Re^+$ 上的一个布尔值步函数 f_p 作为 p 的解释;

(3) 对 V 中的任一时钟变量 x,映射 $\acute{o}$ 指派 $\Re^+$ 上的一个时钟函数 f_x 作为 x 的解释.

定义 2.4. 设 $\Im = \langle I, s\rangle$ 是变量集 V 上的一个 LTLC-模型,e 是 V 上的一个项.对于任意 $t_0 \in \Re^+$,当时刻 $t = t_0$ 时,e 在模型 $\Im$ 下的值 $\Im(e, t_0)$ 归纳定义如下:

(1) $\Im(t, t_0) ::= t_0$;

(2) $\Im(\overline{m}, t_0) ::= m$;

(3) $\Im(u, t_0) ::= I(u)$;

(4) $\Im(x, t_0) ::= f_x(t_0)$,
$\Im(x', t_0) ::= f'_x(t_0)$;

(5) $\Im(-e, t_0) ::= -\Im(e, t_0)$;

(6) $\Im(e_1 + e_2, t_0) ::= \Im(e_1, t_0) + \Im(e_2, t_0)$;

(7) $\Im(e_1 * e_2, t_0) ::= \Im(e_1, t_0) * \Im(e_2, t_0)$.

定义 2.5. 设 $\Im = \langle I, s\rangle$ 是变量集 V 上的一个 LTLC-模型,j 是 V 上的一个公式.对于任意 $t_0 \in \Re^+$,当时刻 $t = t_0$ 时,j 在模型 $\Im$ 下的值 $\Im(j, t_0)$ 归纳定义如下:

(1) $\Im(p, t_0) ::= f_p(t_0)$;

(2) $\Im(p', t_0) ::= f'_p(t_0)$;

(3) $\Im(e_1 = e_2, t_0) := \begin{cases} 1 & \text{若} \Im(e_1, t_0) = \Im(e_2, t_0) \\ 0 & \text{否则} \end{cases}$;

(4) $\Im(e_1 \le e_2, t_0) := \begin{cases} 1 & \text{若} \Im(e_1, t_0) \le \Im(e_2, t_0) \\ 0 & \text{否则} \end{cases}$;

(5) $\Im(\neg j, t_0) := 1 - \Im(j, t_0)$;

(6) $\Im(j_1 \wedge j_2, t_0) := \Im(j_1, t_0) * \Im(j_2, t_0)$;

(7) $\Im([]j, t_0) ::= \begin{cases} 1 & \text{若对于任意 } t_1 \ge t_0 \text{ 有} \Im(j, t_1) = 1 \\ 0 & \text{否则} \end{cases}$;

(8) $\Im(j_1 U j_2, t_0) ::= \begin{cases} 1 & \text{若存在一个 } t_1 \ge t_0 \text{ 使得} \Im(j_2, t_1) = 1 \text{ 且对任意} t_2 \in [t_0, t_1) \text{ 有} \Im(\neg j_1 \wedge \neg j_2, t_2) = 0 \\ 0 & \text{否则} \end{cases}$;

(9) $\Im(\exists u.j, t_0) ::= \begin{cases} 1 & \text{若存在实数}a\text{使得}\Im[a/u](j, t_0)=1 \\ 0 & \text{否则} \end{cases}$.

这里, $\Im[a/u] ::= \langle I[a/u], s\rangle$ 且 $I[a/u]$ 是如下定义的一个映射:

$$I[a/u](w) ::= \begin{cases} I(w) & \text{若}w\text{是}V\text{中的一个刚性变量且}w\text{不是}u \\ a & \text{若}w\text{是}u \end{cases}.$$

如果 $\Im(j, t_0)=1$, 则称 j 在时刻 $t=t_0$ 时为真. 如果 j 在时刻 $t=0$ 时为真, 则称 $\Im$ 是 j 的一个 LTLC-模型.

定义 2.6. 设 j 是变量集 V 上的一个公式.

(1) 如果存在 V 上的一个 LTLC 模型 $\Im$ 使得 $\Im$ 是 j 的一个 LTLC-模型, 则称 j 是 LTLC-可满足的; 否则, 称它是 LTLC-不可满足的.

(2) 如果 $\neg j$ 是 LTLC-不可满足的, 则称 j 是永真的, 记为 $\models j$.

定义 2.7. 设 j 和 f 是变量集 V 上的两个公式. 如果 j 的所有(在变量集 V 上的)模型都是 f 的模型, 则称 f 是 j 的一个逻辑结论, 记作 $j \models f$.

定理 2.8. 以下各式在 LTLC 中成立:

(1) $\models []\Box j \Leftrightarrow \Box j$, $\quad \models \Diamond\Diamond j \Leftrightarrow \Diamond j$;

(2) $\models \neg\Box j \Leftrightarrow \Diamond\neg j$, $\quad \models \neg\Diamond j \Leftrightarrow \Box\neg j$;

(3) $\models \Box j \Rightarrow j$, $\quad \models j \Rightarrow \Diamond j$;

(4) $\models \Box(j \wedge f) \Leftrightarrow (\Box j) \wedge (\Box f)$;

(5) $\models \Diamond(j \vee f) \Leftrightarrow (\Diamond j) \vee (\Diamond f)$;

(6) $\models (\Box(j \Rightarrow f) \wedge \Box j) \Rightarrow \Box f$;

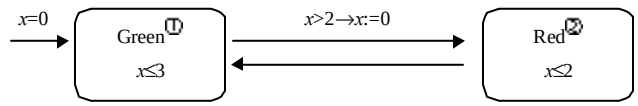
(7) 若 $\models j \Rightarrow f$, 且 $q \models \Box j$, 则 $q \models \Box f$.

证明: 由定义 2.6 容易证出, 详细过程略. ☒

由于 LTLC 中带有全局时钟 t, 故利用 LTLC 可以表示有界反应性(bounded-response)及有界不变量(bounded-invariance)等实时性质, 其具体的表示形式与文献[3]中第 6.2 节的表示基本相同, 这里不再详细讨论.

3 使用 LTLC 表示实时系统

例 1: 如图 1 所示的实时系统模拟了一个简单的交通灯控制系统. 初始时"绿灯亮", 但"绿灯亮"的持续时间不能超过 3 个时间单位, 在"绿灯亮"持续 2 个时间单位后可转为"红灯亮"; "红灯亮"的持续时间不能超过 2 个时间单位, 在"红灯亮"持续 1 个时间单位后可转为"绿灯亮".



①绿色, ②红色.

Fig.1 Traffic_light

图 1 交通灯

本文讨论的实时系统具有有穷个变量(布尔型变量和时钟变量)和有穷个控制状态, 每个控制状态称为实时系统的一个顶点(vertex)(例 1 中的实时系统有两个顶点, 分别标为 Green 和 Red), 系统在每个顶点内可根据约束条件(称为时间不变量, 如例 1 中的 $x \le 3$ 和 $x \le 2$)的要求持续一段时间, 然后转换到另一个顶点. 在一个顶点内停留期间, 布尔型变量不改变其值, 但时钟变量的值随着时间的流逝而不断增加. 系统在一个顶点内的一段停留称为一个延迟(delay)转换. 延迟转换的持续时间是一个正实数(也可能是无穷). 当系统从一个顶点转换到另一个顶点时, 表示系统顶点的布尔型变量需要重置其值, 表示时间约束条件的时钟变量的值既可以维持不变, 也可以重置为 0. 系统从一个顶点到另一个顶点的转换过程称为一个跳跃(jump)转换. 跳跃转换的发生是瞬间完成的, 其持续时间为 0.

本文以时间模块(timed modules)来模拟实时系统. 一个时间模块 M 表示一个与其外部环境交互的实时系

统.每个模块中只能有有限个变量,其中某些变量的值只能被模块自身所改变,而不能被环境所改变;另一些变量的值只能被环境所改变,而不能被模块所改变.依此将模块的变量分为两类,前一类称为控制变量,后一类称为外部变量.不含外部变量的模块称为闭模块.以下以 $var(M)$,$ctr(M)$和 $extl(M)$分别表示模块 M 的所有变量的集合、所有控制变量的集合和所有外部变量的集合.

时间模块有跳跃转换和延迟转换两种类型的转换.跳跃转换通常被写成形如 $vertex\wedge guard\rightarrow new\_vertex\wedge assignment$ 的卫式命令的形式.这里,vertex 是由布尔型变量组成的表达式,用于表示跳跃转换发生时系统所处的顶点.new_vertex 表示跳跃转换发生后,系统将位于的新顶点.guard 是跳跃转换的使能条件(enabling condition),用于表示跳跃转换发生时系统的变量(包括控制变量和外部变量)所应满足的约束条件.assignment 是跳跃转换的命令部分,相当于赋值语句,它使模块中每个控制时钟变量 x 要么重置为 0(即 $x'=0$),要么维持原值不变(即 $x'=x$).它通常可以表示为形如 $x_1'=e_1\wedge x_2'=e_2\wedge\ldots\wedge x_k'=e_k$ 的公式,其中 $x_1,x_2,\ldots x_k$ 是该跳跃转换所在模块的所有控制时钟变量,对于任意 $1\le i\le k$ 有 e_i 或者是 0 或者是 x_i.为方便起见,假定每个跳跃转换在其作用时至少改变了一个控制变量(无论是布尔型还是时钟型)的值,即它不能使所有控制变量的值保持不变.延迟转换通常被写成 $vertex\rightarrow invariant$ 这种形式.其中 vertex 表示延迟转换作用时所处的顶点;invariant 是延迟转换的不变量部分,用于表示在该转换作用的时间段内,模块的时钟变量和外部变量必须满足的约束条件.

一般而言,一个时间模块具有如下的形式:

module module_name
external {*variable_name*:*type*}*
controlled {*variable_name*:*type*}*
init *init_cond*
jump {*vertex*∧*guard*→*new_vertex*∧*assignment*}*
delay {*vertex*→*invariant*}*

这里,type 表示变量的类型,变量的类型可以是 boolean(布尔型)或 clock(时钟类型);init_cond 是模块的初始条件,用于表示模块在初始时刻(即 $t=0$ 时)其控制变量所应满足的条件.

我们在这里说明一下.时间模块中的变量有布尔型和时钟类型两种类型.但是为了书写方便,还可以使用有穷枚举型变量(如例 4 中的 p 和 q),一个有穷枚举型变量的作用实际上相当于若干个布尔型变量的作用.后面出现的带有穷枚举型变量的时间模块完全可以替换为不含枚举型变量的时间模块.

例 2(续例 1):图 1 中的实时系统可以表示为如下的一个时间模块,其中的布尔型变量 p 用于表示交通灯的状态(红或绿),时钟变量 x 用于表示交通灯的某种状态的延续时间.

module Traffic-light
controlled *p*:*boolean*;
x:*clock*
init $p=0\wedge x=0$
jump $p=0\wedge x>2\rightarrow p'=1\wedge x'=0$;
$p=1\wedge x>1\rightarrow p'=0\wedge x'=0$
delay $p=0\rightarrow x\le 3$;
$p=1\rightarrow x\le 2$

在上面的转换中出现的符号→并不是 LTLC 中的逻辑联结词,所以,上面给出的时间模块现在还不是LTLC中的逻辑公式.下面给出时间模块所对应的 LTLC 公式,并利用 LTLC 给出时间模块的语义.

对于跳跃转换 a:$vertex\wedge guard\rightarrow new\_vertex\wedge assignment$,称 LTLC-公式 $vertex\wedge guard\wedge new\_vertex\wedge assignment$ 为转换a所对应的时序逻辑公式,记作 $TLF(a)$.对于延迟转换b:$vertex\rightarrow invariant$,称 LTLC-公式 $vertex\wedge invariant$ 为延迟转换b所对应的时序逻辑公式,记作 $TLF(b)$.

定义 3.1. 设 M 是一个时间模块,$a_0,a_1,\ldots,a_{n-1}$ 是 M 的所有跳跃转换,$b_0,b_1,\ldots,b_{k-1}$ 是 M 的所有延迟转换.称 LTLC-公式

$$init\_cond \wedge ([](V_c = V_c' \vee (\vee_{i<n} TLF(a_i)))) \wedge ([](\vee_{j<k} \mathrm{TLF}(b_j)))$$

为时间模块 M 所对应的时序逻辑公式.这里,$init\_cond$ 是 M 的初始条件,V_c 是 M 中所有控制变量的集合(即 $V_c=ctr(M)$),$V_c=V_c'$ 用作时序公式 $\wedge_{v\in V_c}(v=v')$的缩写形式.以下用 $TLF(M)$来表示这个对应于模块 M 的时序逻辑公式.

例 3(续例 1):时间模块 TLF(traffic-light)所对应的时序逻辑公式为

$$TLF=(p=0\wedge x=0)\wedge([]((p'=p\wedge x'=x)\vee(p=0\wedge x>2\wedge p'=1\wedge x'=0)\vee$$
$$(p=1\wedge x>1\wedge p'=0\wedge x'=0)))\wedge([]((p=0\wedge x\le 3)\vee(p=1\wedge x\le 2))).$$

我们将时间模块 M 等同于时序逻辑公式 $TLF(M)$,并将 $TLF(M)$的语义模型作为 M 的语义模型.

定义 3.2. 设 M 是一个时间模块,j 是一个 LTLC-公式.如果 $TLF(M)\models j$,则称 j 是 M 的一个性质,记作 $M\models j$.

由以上定义可知,若想证明模块 M 具有性质 j,只需证明 j 是 $TLF(M)$的逻辑结论.

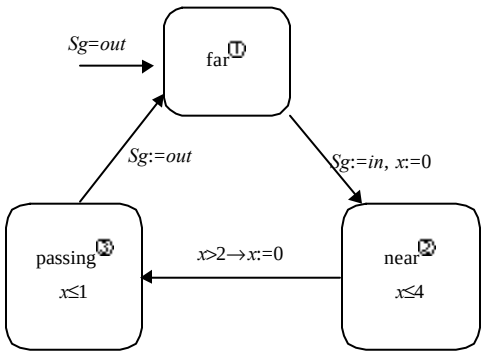
定义 3.3. 称模块 M_1 和 M_2 是相容的,若 $ctr(M_1)\cap ctr(M_2)=\varnothing$ 且对任意 $v\in var(M_1)\cap var(M_2), v$ 在 M_1 和 M_2 中有相同的类型声明.称 n 个模块 $M_1,M_2,\ldots,M_n$ 是相容的,若它们两两相容.

若模块 $M_1,M_2,\ldots,M_n$ 相容,我们用$[M_1\|M_2\|\ldots\|M_n]$来表示这 n 个模块的并行复合.并称公式 $TLF(M_1)\wedge TLF(M_2)\wedge\ldots\wedge TLF(M_n)$ 为复合模块 $[M_1\|M_2\|\ldots\|M_n]$ 所对应的时序逻辑公式.时序公式 $TLF(M_1)\wedge TLF(M_2)\wedge\ldots\wedge TLF(M_n)$的语义模型被看做是复合模块$[M_1\|M_2\|\ldots\|M_n]$的语义模型.

定义 3.4. 对于公式 j,若 $TLF(M_1)\wedge TLF(M_2)\wedge\ldots\wedge TLF(M_n)\models j$,则称 j 是复合模块$[M_1\|M_2\|\ldots\|M_n]$的一个性质.

下面通过一个例子来说明如何利用定义 3.2(或定义 3.4)来证明实时系统的性质.

例 4(railroad gate control[11,12]):图 2 模拟一个环形铁路上的火车,该段铁路上有一个道口,图 3 模拟道口控制器.这段铁路被分成 far,near 和 passing 三个区段.初始时,火车在 far 区段.当火车进入 near 区段时,它发出 in 的信号.在 near 区段,火车需要运行 2~4 分钟进入 passing 区段.在 passing 区段,火车运行不超过 1 分钟将进入 far 区段,在进入 far 区段时,火车将发出 out 的信号.在 far 区段运行不确定的一个时间段后,火车再次进入 near 区段,依此反复.道口起初是打开的(open 状态),在 in 信号发出的瞬间,道口进入 down 状态,down 状态持续 1 到 2 分钟后,道口进入 closed 状态,道口保持 closed 状态不变,一直到 out 信号发出时进入 up 状态.若在进入 up 状态 1 分钟之内收到 in 信号,则进入 down 状态;否则,在 1 分钟后进入 open 状态.open 状态可持续不确定的一个时间段,直到 in 信号发出为止.图 2 和图 3 中的时钟变量 x 和 y 用于记录从某时刻起流逝的时间,布尔型变量 sg 用于表示发出的信号.



①远处,②近处,③正在通过.

Fig.2 Train

图 2 火车

图 2 和图 3 所表示的实时系统可用时间模块表示如下:

module Train

controlled p:{far,$near$,$passing$};
sg:{in,out};
x:$clock$

init $p=far\wedge sg=out\wedge x=0$

jump $p=far\rightarrow p'=near\wedge sg'=in\wedge x'=0$;
$p=near\wedge x\ge 2\rightarrow p'=passing\wedge sg'=sg\wedge x'=0$;
$p=passing\rightarrow p'=far\wedge sg'=out\wedge x'=x$

delay $p=far\rightarrow$true; //(这里 true 表示"真")
$p=near\rightarrow x\le 4$;

$p=passing\rightarrow x\leq 1$

module Gate

external sg:{in,out}

controlled q:{$open$,$closed$,up,$down$};

y:$clock$

init $q=open\wedge y=0$

jump $q=open\wedge sg'=in\rightarrow q'=down\wedge y'=0$;

$q=down\wedge y\geq 1\rightarrow q'=closed\wedge y'=y$;

$q=closed\wedge sg'=out\rightarrow q'=up\wedge y'=0$;

$q=up\wedge sg'=in\rightarrow q'=down\wedge y'=0$;

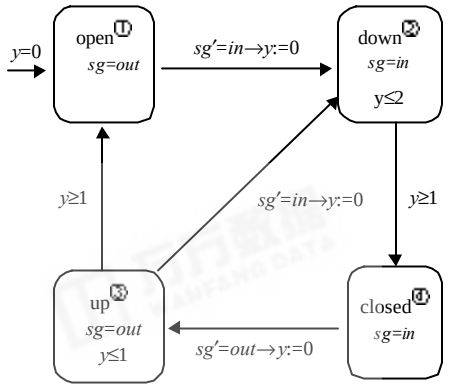
$q=up\wedge y=1\rightarrow q'=open\wedge y'=y$

delay $q=open\rightarrow sg=out$;

$q=down\rightarrow sg=in\wedge y\leq 2$;

$q=closed\rightarrow sg=in$;

$q=up\rightarrow sg=out\wedge y\leq 1$



①开,②下降,③关闭,④上升

Fig.3 Gate

图 3 道口

由定义 3.1 **可知**,

$$TLF(Train)=(p=0\wedge sg=0\wedge x=0)\wedge([]((p'=p\wedge sg'=sg\wedge x'=x)\vee(p=0\wedge p'=1\wedge sg'=1\wedge x'=0)\vee (p=1\wedge x\geq 2\wedge p'=2\wedge sg'=sg\wedge x'=0)\vee(p=2\wedge p'=0\wedge sg'=0\wedge x'=x)))\wedge ([](p=0\vee(p=1\wedge x\leq 4)\vee(p=2\wedge x\leq 1))),$$

且

$$TLF(Gate)=(q=0\wedge y=0)\wedge([]((q'=q\wedge y'=y)\vee(q=0\wedge sg'=1\wedge q'=1\wedge y'=0)\vee(q=1\wedge y\geq 1\wedge q'=2\wedge y'=y)\vee (q=2\wedge sg'=0\wedge q'=3\wedge y'=0)\vee(q=3\wedge sg'=1\wedge q'=1\wedge y'=0)\vee(q=3\wedge y=1\wedge q'=0\wedge y'=y)))\wedge ([]((q=0\wedge sg=0)\vee(q=1\wedge sg=1\wedge y\leq 2)\vee(q=2\wedge sg=1)\vee(q=3\wedge sg=0\wedge y\leq 1))).$$

下面证明公式[](*p*=2⇒*q*=2)(**即公式**[](*p*=*passing*⇒*q*=*closed*))**是复合模块**[*Train*||*Gate*]**的一个性质.设**$\Im$**是** $TLF(Train)\wedge TLF(Gate)$**的一个模型,且** f_p, f_{sg}, f_q, f_x **和** f_y **分别是变量** p,sg,q,x **和** y **在模型**$\Im$**下的解释**.

设 $a_1,a_2,\ldots,a_n,\ldots$**是函数** f_p **的所有不连续点所组成的序列,且满足** $a_1<a_2<\ldots<a_n<\ldots$**(这里不妨假定** f_p **有无穷个不连续点,只有有限个不连续点时的证明与无限时相仿**).

取 $a_0=0$,**则对任意** $n\in$N,**有** f_p **在区间**(a_n,a_{n+1})**上连续.由**[]((*p*′=*p*∧*sg*′=*sg*∧*x*′=*x*)∨(*p*=0∧*p*′=1∧*sg*′=1∧*x*′=0)∨(*p*=1∧*x*≥2∧*p*′=2∧*sg*′=*sg*∧*x*′=0)∨(*p*=2∧*p*′=0∧*sg*′=0∧*x*′=*x*))**可知,**f_p **的不连续点只可能有** 3 **种,它们分别将** f_p **的值从** 0,1,2 **改为** 1,2,0.**并且在第** 1 **种不连续点出现时,** f_{sg} **的值由** 0 **变为** 1;**在第** 3 **种不连续点出现时,** f_{sg} **的值由** 1 **变为** 0.**利用** f_p **的不连续点的特征,应用归纳法可得,对任意** $i\in$N,**有**

$$f_p(t)=\begin{cases}0 & \text{若 } t\in(a_{3i},a_{3i+1}]\\ 1 & \text{若 } t\in(a_{3i+1},a_{3i+2}]\\ 2 & \text{若 } t\in(a_{3i+2},a_{3i+3}]\end{cases}, \tag{1}$$

$$f_{sg}(t)=\begin{cases}0 & \text{若 } t\in(a_{3i},a_{3i+1}]\\ 1 & \text{若 } t\in(a_{3i+1},a_{3i+3}]\end{cases}. \tag{2}$$

由[]((*q*=0∧*sg*=0)∨(*q*=1∧*sg*=1)∨(*q*=2∧*sg*=1)∨(*q*=3∧*sg*=0))**和式**(2)**可得**

$$f_q(t)\in\begin{cases}\{0,3\} & \text{若 } t\in(a_{3i},a_{3i+1}]\\ \{1,2\} & \text{若 } t\in(a_{3i+1},a_{3i+3}]\end{cases}. \tag{3}$$

由 $f_p(a_{3i+2})=1$ **和** [](($p'=p\wedge sg'=sg\wedge x'=x$)∨($p=0\wedge p'=1\wedge sg'=1\wedge x'=0$)∨($p=1\wedge x\geq 2\wedge p'=2\wedge sg'=sg\wedge x'=0$)∨($p=2\wedge p'=0\wedge sg'=0\wedge x'=x$))**还可以得到** $f_x(a_{3i+2})\geq 2$,**再由定义** 1.3 **即知** $a_{3i+2}-a_{3i+1}\geq 2$.

对于任意的 $t_0\in(a_{3i+2},a_{3i+3}]$,**由式**(3)**可知** $f_q(t_0)=1\vee f_q(t_0)=2$.**假如** $f_q(t_0)=1$,**则由**[]((*q*=0∧*sg*=0)∨(*q*=1∧*sg*=1∧*y*≤2)∨(*q*=2∧*sg*=1)∨(*q*=3∧*sg*=0∧*y*≤1))**可知** $f_y(t_0)\leq 2$,**于是由时钟函数的定义可知** $t_0-a_{3i+1}\leq 2$,**这与** $t_0>a_{3i+2}\geq a_{3i+1}+2$ **矛盾.于是,当** $t_0\in(a_{3i+2},a_{3i+3}]$**时有** $f_q(t_0)=2$. **结合式**(1)**我们得到,对任意** $t_0\in\Re^+$,**若** $f_p(t_0)=2$,**则** $f_q(t_0)=2$.**这样就证明了**$\Im$**是**

[](*p*=2⇒*q*=2)的一个模型.由于$\Im$可以是 *TLF*(*Train*)∧*TLF*(*Gate*)的任意一个模型,故由定义 3.4 可知,[](*p*=2⇒*q*=2)是复合模块[*Train*‖*Gate*]的一个性质.

4 结 论

本文提出了一个带有时钟变量的线性时序逻辑语言 LTLC,与现有的一些实时逻辑,如 Timed Computation Tree Logic[5],Metric Interval Temporal Logic[6]和 TLA+[13]等相比,LTLC 的一个重要特色是,它既可以作为规范语言用来表示实时系统的性质,又可以作为系统刻画语言用来表示实时系统的实现.LTLC 的这个特点将有助于实时系统的性质验证和逐步求精(有关逐步求精的问题将另文加以讨论).

在本文中,实时系统是用时间模块来表示的,每个时间模块是一个 LTLC 公式,时间模块的性质也是用 LTLC 公式来表示的,这样,实时系统的性质验证便归结为验证两个 LTLC 公式之间的蕴涵关系.本文是利用 LTLC 公式的语义定义来检查 LTLC 公式之间的蕴涵关系,在以后的工作中,我们将给出 LTLC 的一些公理、证明规则以及其部分子类的判定性结果,以便发展出关于实时系统的更有效的性质验证方法.

模型检查(model checking)[12,14]是一种重要的基于算法的性质验证方法.我们提出了一种基于 LTLC 的实时系统模型检查方法,这将另文加以介绍.

致谢 英国 Leicester 大学的刘志明博士阅读过本文的初稿,并提出了很好的修改意见.在此我们向他表示感谢.

References:

[1] Alur, R., Henzinger, T.A. Real-Time system=discrete system+clock variables. Software Tools for Technology Transfer, 1997, 1(1/2): 86~109.

[2] de Bakker, J.W., Huizing, K., de Rover, W.-P, *et al*, eds. Proceedings of the REX Workshop 'Real-Time: Theory in Practice'. Lecture Notes in Computer Science 600, New York: Springer-Verlag, 1991.

[3] Henzinger, T.A., Manna, Z., Pnueli, A. Temporal proof methodologies for timed transition systems. Information and Computation 1994,112(2):273~337.

[4] Manna, Z., Pnueli, A. Clocked transition systems. In: Pnueli, A., Lin, H., eds. Logic and Software Engineering. Singapore: World Scientific, 1996. 3~42.

[5] Alur, R., Courcoubetis, C., Dill, D.L. Model-Checking in dense real-time. Information and Computation, 1993,104(1):2~34.

[6] Alur, R., Feder, T. Henzinger, T.A. The benefits of relaxing punctuality. Journal of the ACM, 1996,43(1):116~146.

[7] Alur, R., Henzinger, T.A. A really temporal logic. Journal of the ACM, 1994,41(1):181~204.

[8] Ostroff, J.S. Temporal logic for real-time systems. Taunton, England: Research Studies Press Ltd., 1989.

[9] Alur, R., Dill, D.L. A theory of timed automata. Theoretical Computer Science, 1994,126(2):183~235.

[10] Manna, Z., Pnueli, A. The temporal logic of reactive and concurrent systems: Specification. New York: Springer-Verlag, 1992.

[11] Bjφrner, N., Manna, Z., Sipma, H.B., *et al*. Deductive verification of real-time systems using STeP. In: Rus, T., Bertran, M., eds. Proceedings of the ARTS'97. Lecture Notes in Computer Science 1231, New York: Springer-Verlag, 1997. 22~43.

[12] Henzinger, T.A., Nicollin, X., Sifakis, J., *et al*. Symbolic model checking for real-time systems. Information and Computation, 1994,111(2):193~244.

[13] Lamport, L. Hybrid systems in TLA+. In: Rischel, H., Ravn, A.P., Nerode, A., Grossmann, R.L., eds. Hybrid Systems. Lecture Notes in Computer Science 736, New York: Springer-Verlag, 1993. 77~102.

[14] Clarke, E.M., Emerson, E.A., Sistla, A.P. Automatic verification of finite-state concurrent systems using temporal-logic specifications. ACM Transactions on Programming Languages and Systems, 1986,8(2):244~263.

A Linear Temporal Logic with Clocks for Verification of Real-Time Systems[1]

LI Guang-yuan, TANG Zhi-song

(*Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: {ligy,cst}@ios.ac.cn

http: //www.ios.ac.cn

Abstract: In order to specify real-time systems, many temporal logics such as Timed Computation Tree Logic, Metric Interval Temporal Logic and Real-Time Temporal Logic have been proposed. Although these logics are good at specifying properties of real-time systems, they are not suitable for describing the implementations of such systems. Thus, the specifications and the implementations are usually described by different languages for real-time systems. In this paper, a new linear temporal logic with clocks, called LTLC, is introduced. It is an extension of Manna and Pnueli's linear temporal logic. It can express both the properties and the implementations of real-time systems. With LTLC, systems can be described at many levels of abstraction, from high-level requirement specifications to low-level implementation models, and the conformance between different descriptions can be expressed by logical implication. This aspect of LTLC will be beneficial to the verification and the stepwise refinements of real-time systems.

Key words: real-time system; timed automaton; linear temporal logic; specification language; system description language; property verification

---

[1] Received July 20, 2000; accepted June 20, 2001

Supported by the National Natural Science Foundation of China under Grant No.60073020; the Key Sci-Tech Project of the National 'Ninth Five-Year-Plan' of China under Grant No.98-780-01-07-01; the National High Technology Development 863 Program of China under Grant No.863-306-ZT02-04-1