中 国 科 学 院
博士学位研究生学位论文

LTLC: 面向实时与混成系统的连续时序逻辑

李 广 元

计算机科学开放实验室
中国科学院软件研究所，北京， 100080
Email: ligy@ios.ac.cn

指 导 教 师： 唐稚松 研究员
学科专业名称： 计算机软件与理论
论文提交日期： 2001 年 3 月
论文答辩日期： 2001 年 5 月
学位授予单位： 中国科学院软件研究所

摘 要

混成系统是一种既包含离散成分又包含连续成分的计算系统，数控系统等一些与其外部连续变化的物理环境不断交互的嵌入式系统就是其典型的代表。实时系统是一类特殊的混成系统，其连续成分是一组用来表示时间约束条件的时钟。由于这些系统在工业及国防领域有着重要而广泛的应用，它们的安全性和可靠性越来越引起人们的关注，因而对这些系统进行形式化分析以确保其安全性和可靠性也成为近年来的一个研究热点。

为了描述实时及混成系统的性质和行为，十多年来，各种不同的时序逻辑如： *Metric Interval Temporal Logic* ， *Real-Time Temporal Logic* ， *Integrator Computation Tree Logic* 和 *Hybrid Temporal Logic* 等相继提出，尽管这些时序逻辑作为规范语言用于描述实时及混成系统的性质时都还比较合适，但它们不适合用来表示实时及混成系统的实现，因为它们缺乏表示系统状态的动态变化的能力。

在现有文献中，实时和混成系统通常是用时间自动机、混成自动机、时间转换系统和相位转换系统等来表示的。但这些系统刻画语言却不适合作为规范语言来使用，因为它们不能表示实时和混成系统的一些重要性质（如安全性和活性等）。这样在基于逻辑方法的实时和混成系统的研究中，系统和它的性质通常是用两个不同的语言来表示的。

本文定义了一个具有连续语义的线性时序逻辑，记为 LTLC ，它是 Manna 和 Pnueli 所提出的线性时序逻辑 LTL 的一个推广。 LTLC 既能表示实时和混成系统的性质，又能很方便地表示实时和混成系统的实现，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。

本文还定义了 LTLC 的三个子语言： LTLC/B 、 LTLC/R 及 LTLC/H ，并证明了前两者的可满足性问题是可判定的。这三个子语言可分别用来表示有穷状态反应系统、有穷控制状态实时系统以及混成系统。本文所给的关于 LTLC/B- 公式的可满足性判定过程可用于检查有穷状态反应系统之间的一致性以及有穷状态反应系统与其规范之间的一致性；所给的关于 LTLC/R- 公式的可满足性判定过程可用于检查有穷控制状态实时系统之间的一致性以及有穷控制状态实时系统与其规范之间的一致性。

此外，本文还给出了在样本控制模式下（在此模式下，混成系统的跳跃转换只发生在整数时间点上）多速率混成系统关于 LTLC/H- 公式的一个模型检查过程。

关键词： 实时系统，混成系统，时序逻辑，实时逻辑，规范语言，系统刻画语言，可满足性，可判定性，模型检查。

LTLC: A Continuous-time Temporal Logic for Real-time and Hybrid Systems

Abstract

Hybrid systems are computational systems consisting of both continuous and discrete components. Typical examples are digital controllers that interact with continuously changing physical environments. Real-time systems are a subclass of hybrid systems where the continuous components of the system are clocks which are used to express the timing constraints on the system. Because of their extensive application in industry and national defence, real-time and hybrid systems mostly focus on the safety and reliability properties. So the formal analysis of these systems becomes an interesting topic.

In order to specify real-time and hybrid systems, many temporal logics, such as *Metric Interval Temporal Logic*, *Real-Time Temporal Logic*, *Integrator Computation Tree Logic* , and *Hybrid Temporal Logic* have been proposed. Though these logics are good at specifying properties of real-time and hybrid systems, they are not suited for describing the implementations of such systems. They lack the ability to describe the dynamic change in the state of real-time and hybrid systems.

In the literature, real-time and hybrid systems are usually described by *Timed Automata*, *Hybrid Automata*, *Timed Transition Systems* and *Phase Transition Systems*. However, these system description languages cannot be used as specification languages, because they lack the ability to express some important properties(such as safety properties and liveness properties) of real-time and hybrid systems. Thus, real-time and hybrid systems and their properties are usually described by different languages.

In this thesis, a new linear temporal logic with continuous semantics, 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 and hybrid 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.

Three sub-languages of LTLC, called LTLC/B, LTLC/R and LTLC/H, are defined and the satisfiability problems for the first two are proved to be decidable. They are used to represent finite-state reactive systems, finite-location real-time systems and

hybrid systems respectively. Our decision procedure for LTLC/B can be used to check whether a finite-state reactive system meets a required specification or whether a finite-state reactive system conforms with another system. Our procedure for LTLC/R can do the similar things for real-time systems with finite control locations.

In addition, a model-checking procedure for multirate hybrid systems is also given. It can decide whether a formula in LTLC/H is true with respect to a given multirate automaton whose jump transitions happen only at integer times.

Keywords: real-time system, hybrid system, temporal logic, real-time logic, specification language, system description language, satisfiability, decidability, model checking.

目　　录

第一章　　引　言

1.1　反应系统

在计算机使用的早期，计算机的主要任务是完成一定的数值计算（或数据处理），这种计算通常要在有限步内终止，并给出计算结果。对这种计算，人们关心的是计算的最终结果是否正确，而不太关心计算的中间过程，如果有两个程序对同样的输入有着同样的输出，则认为这两个程序是等价的。这样的程序可以看作是一个从输入到输出的函数，文献中通常用变换系统（ transformational system[76] ），或串型系统（ sequential system ）来称呼它们，它们的主要任务是完成特定的算法计算，并且这种计算可在有限步内终止。

随着计算技术的不断发展，数字化已深入到人们生活的各个角落，每天数以吉 (giga-byte) 计的各种程序代码运行在从数控家电、程控电话、移动通信到互联网等各种控制系统和通信系统中，它们中的许多程序已不再像串型系统那样在完成一定的计算后便终止，它们可能不再终止，它们要不断地接受外部的输入，并对这些输入作出反应，它们的任务是维持系统和环境之间的不断进行的交互过程。这种用于维持系统与环境之间不断进行的交互过程的计算系统称之为反应系统 (reactive system[76]) ，人们常说的数控系统、程控系统、嵌入式系统及网络服务器等大多属于这样的系统。由于反应系统的许多应用领域属于安全性至关重要 (safety-critical) 的领域，如核反应堆控制等，故如何保证反应系统的安全性和可靠性一直是人们关心的一个重要问题。在大多数的工程实践中，软件的可靠性主要是通过测试和反复试用来保证的，但由于反应系统的运行往往与外部环境有关，其执行存在着许多不确定因素，这对其测试带来了极大的困难，并且测试本身只能发现错误，而不能保证它无错误。故许多计算机科学家（如图灵奖得主 A. Pnueli[87] ）认为利用数学方法（也就是形式化方法）是解决反应系统的安全性的根本方法。

为了对反应系统进行形式分析和验证，首先需要对反应系统建立数学模型，通常根据不同的需要采用迁移系统 (transition system[61, 76, 78] ，也称变迁系统或转换系统等) 的不同变种作为反应系统的数学模型，迁移系统的基本成分是一个三元组 $< Q, \mathcal{T}, Q_0 >$, 其中 Q 是状态的集合， $\mathcal{T}$ 是变迁 (有时也称为转换或事件) 的集合，每一个变迁是笛卡尔集 $Q \times Q$ 的一个子集， Q_0 是 Q 的一个子集 (称为初始状态的集合) 。迁移系统的一个计算序

列是指这样的一个无穷序列 $s_0, s_1, s_2, s_3, \ldots$，其中 $s_0 \in Q_0$，且对于任意非负整数 k，存在 $\tau \in \mathcal{T}$ 使得 $(s_k, s_{k+1}) \in \tau$。迁移系统的行为是指它的计算序列的集合，迁移系统的性质是指这些 (无穷) 计算序列所满足的性质。在确定了反应系统的数学模型之后，还需要：

- *系统刻画语言* (system description language). 用于表示反应系统的数学模型 (系统的实现)。 CSP[58], CCS[81] ， Unity[33] 以及各种类似于程序设计语言的文本语言和类似于状态转换图的图表语言等常被用来作为反应系统的系统刻画语言。通常要求系统刻画语言是形式语言，要有严格定义的语法和语义。

- *规范语言* (specification language). 用于表示反应系统的性质 (系统的需求)，也称为性质描述语言。各种模态逻辑 [38] ，如线性时序逻辑 LTL[76, 68] 、分支时序逻辑 CTL[28] 、交替时间时序逻辑 ATL[20]) 和 μ- 验算 [65] 等常被用作为反应系统的规范语言。由于系统的性质是指它的计算序列的性质，而计算序列是一个无穷序列，无穷序列的性质通常可用模态逻辑来表示，故通常选用模态逻辑作为反应系统的规范语言 (这一做法首先是由 A. Pnueli 提出的，他在 [84] 中使用线性时序逻辑 LTL 作为反应系统的规范语言)。

在对反应系统进行形式化分析和验证时，采用的系统刻画语言和规范语言通常是两个不同的语言，如 Z. Manna 和 A. Pnueli 在 [76] 中分别采用线性时序逻辑 LTL 和一种称为 SPL(Simple Programming Language) 的并发程序设计语言来分别作为反应系统的规范语言和系统刻画语言。但采用同一种语言既作为反应系统的规范语言又作为系统刻画语言在许多情况下会带来很多方便之处，如有利于对反应系统进行逐步求精和对反应系统与其规范之间的一致性进行验证。唐稚松教授在其 XYZ 系统 [97, 98] 中就采用线性时序逻辑语言 XYZ/E 既作为规范语言又作为系统刻画语言用来表示系统的静态规范 (静态语义) 和动态行为 (动态语义)，为此 XYZ/E 被称为世界上第一个可执行的时序逻辑语言 [23] 。L. Lamport 也在其 TLA(The Temporal Logic of Actions[68]) 中使用线性时序逻辑语言既作为反应系统的规范语言又作为其系统刻画语言。

L. Lamport 在 [66] 中最早将反应系统的性质分为安全性 (safety[76]) 和活性 (liveness [76]) 两大类。Z. Manna 和 A. Pnueli 在 [75] 中给出了反应系统的 (使用时序逻辑 LTL 所表示的) 性质的一个更详细的分类，他们在 [72, 73, 74, 85] 中给出了反应系统的安全性、响应性 (response[74]) 和反应性 (reactivity[74]) 的证明规则，并在 [74] 中证明了这些规则的相对完备性，L. Lamport 在 [68] 中也给出了证明反应系统的安全性和公平性 (fairness[68, 76]) 的证明规则，至此利用演绎方法证明反应系统的性质的逻辑体系已基本建立。

E.M. Clarke 等人在 [90, 39, 28] 中提出了另外一种验证反应系统的性质的方法，称为模型检查 (model checking) 方法，它使用状态空间搜索的办法来全自动地检验一个 (有穷状态) 反应系统是否满足某个 (用时序逻辑公式表示的) 性质。模型检查方法自提出以来，发展很快，各种各样的模型检查技术不断提出，并已在通信协议验证和硬件系统验证等方

面的实际应用中取得了成功 [31, 32] ，所以该方法近年来得到广泛关注，随着计算机硬件速度的提高和模型检查算法的不断优化，模型检查方法的应用前景被普遍看好。

1.2 实时和混成系统

由于在反应系统的一般计算模型 (如公平转换模型 [76]) 中不考虑两次变迁发生的时间间隔, 故当所讨论的反应系统需要对变迁发生的时间间隔作出要求时, 利用反应系统的一般数学模型就不能刻画出人们关心的部分特性。实时系统 (real-time system)[17, 22, 56, 78] 就是这样一类需要考虑时间约束条件的反应系统。核反应堆、飞行控制以及铁路调度等方面的许多计算机控制系统都属于实时系统。这些系统的许多动作的完成是与时间有关的, 对外部事件的反应是有时间限制的，需要满足一定的时间约束，如某动作要在一秒钟内完成等.

实时系统常用的数学模型有时间转换系统 (timed transition system[56]) 和时钟转换系统 (clocked transition system[78]) 等。其系统刻画语言有时间自动机 (timed automata [10]) 、 TCCS(Timed CCS[93]) 和 TCSP(Timed CSP[35]) 等。由于基本的时序逻辑如 LTL 、CTL 等不能满足描述实时系统的性质的需要, 故各种扩充了时间表达能力的时序逻辑在 20 世纪 80 年代和 90 年代相继提出，这些能表示实时性质的时序逻辑通常称为实时逻辑，如 TCTL(Timed Computation Tree Logic[5]), MITL(Metric Interval Temporal Logic[12]), TPTL(Timed Propositional Temporal Logic[15]) 和 Real-Time Temporal Logic[83] 等， R. Alur 和 T.A. Henzinger 在 [13] 中给出了关于实时逻辑的一个综述，并提出了实时逻辑分类的几种原则，如：是线性时间的还是分支时间的，是离散时间的还是稠密时间的，时间是显式的 (即带时钟变量的) 还是隐含的，是基于点的还是基于区间的，是命题的还是一阶的。关于常见实时逻辑的一些基本情况可参见本文第六章的相关工作介绍。

实时系统研究中的一个热点问题是它的算法验证。 R. Alur 等人在 [9, 10] 中给出了时间自动机的非空性的判定算法，他们提出的划分 '域等价类' 的方法成为以后 (基于稠密时间的) 实时系统的模型检查的基础。随后 R. Alur 等人在 [5] 中给出了有限控制状态实时系统关于 TCTL 的模型检查算法，并证明了 TCTL 的可满足性是不可判定的。 T.A. Henzinger 等人 [57] 给出了有限控制状态实时系统关于 TCTL 的一个符号模型检查算法。R. Alur, T. Feder 和 T.A. Henzinger 在 [12] 中证明了 MITL 的可满足性是可判定的，此后一些针对实时系统的验证工具如 KRONOS[37, 36] 、 UPPAAL[70, 25] 等相继出现，并已在实际的工业协议验证中取得了成功 [24] 。

当反应系统作为嵌入式系统嵌入在一个物理环境之内并与外部的物理环境交互时, 则需要考虑既包含离散量 (这里离散量是指取值域为一个有限集的量, 如表示 '开''关' 状态的量, 其取值为 0 和 1) 又包含连续量 (这里连续量主要指物理量, 如连续变化的温度、时间、距离、

速度等) 的计算系统, 这类系统通常被称为混成系统 (hybrid system[71, 41, 6, 2, 3, 96]) 。近年来, 混成系统已成为计算机领域和控制论领域的科学家共同关注的一个热点问题, 国际上每年都有不少关于混成系统的专门会议 (如 'International Hybrid Systems Workshop' 、International Workshop on Hybrid and Real-Time Systems 、'Hybrid System: Computation and Control' 等, 它们的论文集均收于 Springer 出版的 'Lecture Notes in Computer Science' 系列中), 许多国际著名刊物如 'IEEE Transactions on automatic control' 、'Proccedings of IEEE' 等近年来也相继出专辑介绍混成系统的理论与应用现状 [1, 2, 3] 。

混成系统的一个比较早的数学模型是 O. Maler 等人提出的相位转换系统(phase transition systems[71, 78, 79]), 但现在使用较普遍的一个数学模型是混成自动机 (hybrid automata[8, 6, 46] , 它同时也作为混成系统的系统刻画语言来表示混成系统的实现。表示混成系统性质的规范语言仍然是各种模态和时序逻辑 (实时逻辑), 如 Temporal Logic of Actions TLA^+[67] 、Hybrid Temporal Logic (HTL)[60, 59] 、Integrator Computation Tree Logic (ICTL)[19] 和 Duration Calculus[96] 等。混成系统的算法验证主要集中在线性混成系统 (linear hybrid system[6]) 的模型检查, R. Alur 等人在 [6] 中讨论了线性混成系统的符号模型检查, 但所给出的计算过程是半可判定的, 对一些线性混成系统这个过程可能并不终止。为了确定哪些线性混成系统的模型检查问题是可判定的, T.A. Henzinger 等人在 [45, 51, 64, 21, 54, 55] 中讨论了线性混成系统的一些重要子类 (如多速率混成系统, 矩形混成系统等) 的可达性的判定问题, 给出了一些可判定性子类与不可判定性子类之间的界限, 如初始化多速率混成系统和初始化矩形混成系统的可达性是可判定的, 而非初始化的多速率混成系统和矩形混成系统是不可判定的。混成系统的模型检查工具有 KRONOS[37, 36] 和 HyTech[19, 48] 等。

1.3 本文的目标、贡献和组成

为了形式验证实时和混成系统, 我们需要系统刻画语言和规范语言来分别表示实时和混成系统的计算模型和性质。时间自动机 (TA) 和混成自动机 (HA) 是实时系统和混成系统使用最普遍的系统刻画语言, 但 TA 和 HA 不是逻辑语言, 它不能表示实时和混成系统的一些重要性质 (如安全性、可达性、公平性、有界响应性 [78] 等) 。实时和混成系统现有的规范语言 (如 TCTL 、MITL 、HTL 、ICTL 等) 大多是为了表示系统的性质和规范而设计的, 尽管它们作为规范语言还比较合适, 但却不适合作为系统刻画语言来表示实时和混成系统的动态执行过程, 它们缺乏表示系统状态的动态变化的能力, 如它们大多不能直接表示像赋值语句 $x := x + 1$ 这样的 (在点上发生的) 状态转换和像微分方程 $\dot{x} = 1 - x$ 这样的 (在非 0 时间段上的) 状态连续变化, 而表示这样的状态转换和状态连续变化对于表示混成系统是必需的。故在现有的基于逻辑方法的实时和混成系统的研究中, 系统和它的

性质通常是用不同的语言来表示的，性质是用时序逻辑语言或其它模态逻辑语言表示的，而系统则是用时间自动机、混成自动机或其它类程序设计语言表示的。这种作法既不利于系统性质的演绎证明（因性质和系统不是用同一语言写的，它们不在同一推演系统下，而不是同一语言写的系统和性质要想进行演绎证明，其推演系统将是很烦琐的）；也不利于系统从规范设计到系统实现的逐步求精过程中的平滑过渡（因规范和系统不是同一语言，中间必要从一个语言跳到另一语言）。显然若能实现将混成系统和它的性质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将是非常有益的 [68, 69, 98] 。对于非实时的计算系统（即普通的反应系统），将系统和性质用统一的时序逻辑公式来表示的思想已在唐稚松的 XYZ 系统 [97, 98] 和 Lamport 的 TLA[68] 中得到实现。

- 本文的目标之一就是想要建立这样的一个时序逻辑系统，它既能作为规范语言用来表示实时和混成系统的性质，又能作为系统刻画语言来表示实时和混成系统的实现，它能在统一的语义框架下表示出从静态的需求规范到动态的系统实现之间的不同抽象层次上的系统描述。在这样一个时序逻辑内，系统和它的性质都被表示为时序逻辑公式，系统满足性质以及两个系统之间的一致性都可表示为两个时序逻辑公式之间的蕴涵关系。

为了达到这一目标，本文提出了一个具有连续语义的线性时序逻辑 (a linear temporal logic with continuous semantics)，简记为 LTLC，它是 Manna 和 Pnueli 所提出的线性时序逻辑 LTL[76] 的一种推广，它将 LTL 的语义解释由无穷离散时间点上的状态转换改为在连续时间点上的状态改变。为了能表示混成系统，LTLC 中引入了两个新的算子：微分算子 ” 和新值算子 ‘′’，这使得微分方程和赋值语句都能在 LTLC 中得到表示。这样利用 LTLC 就可方便地表示出实时和混成系统，特别是，一个时间自动机或一个混成自动机在 LTLC 中就是一个公式，两个并发系统之间的并发关系变成了逻辑公式之间的合取 (conjunction) 关系，系统满足性质的检查变成了逻辑公式之间的蕴涵关系的检查，系统和系统之间的求精关系也变成了公式之间的蕴涵关系，这样系统与性质之间的一致性检查（即模型检查）和系统之间的‘求精’关系检查以及性质和性质之间的一致性检查均可转化为 LTLC 公式的可满足性检查。

此外，由于 LTLC 还能表示反应系统和它的性质，故 LTLC 既能作为反应系统、实时系统和混成系统三者统一的系统刻画语言，又能作为三者统一的规范语言，实现了三者的规范语言与系统刻画语言的大统一。

- 本文的目标之二是要发展基于 LTLC 的算法验证方法，包括模型检查、非空性检查等。由于 LTLC 的一个重要特征是它能支持实时和混成系统的逐步求精过程，故发展能检查两个不同抽象层次上的系统描述之间的一致性的算法方法是本文重要目标之一。

就这个目标而言，本文得到了如下结果：

1 。证明了 LTLC 的子语言 LTLC/R 的可满足性是可判定的。由于 LTLC/R 中的公式可表示不同抽象层次的有穷控制状态实时系统，故本文给出的判定过程既可以用来检查有穷控制状态实时系统的非空性，也可以用来检查有穷控制状态实时系统的逐步求精过程中不同层次系统描述之间的一致性。

2 。证明了在样本控制模式下（在此模式下，混成系统的跳跃转换只发生在整数时间点上）多速率混成系统关于 LTLC/H- 公式的模型检查是可判定的。

此外，本文还证明了 LTLC 的子语言 LTLC/B 的可满足性是可判定的。由于利用 LTLC/B 中的公式可表示有穷状态反应系统和它们的性质，故该判定过程可用于检查有穷状态反应系统之间的一致性以及有穷状态反应系统与其规范之间的一致性。

本文以下几章是这样组织的。

第二章属预备知识，共有两节。第 1 节介绍了线性时序逻辑 LTL[68, 76] 的基本概念，包括语法、语义及可满足性等。第 2 节介绍了关于 LTL 的可满足性的判定结果，为了后面（第四、五章）的需要，这一节还引入了 'n- 可满足性' 的概念，并证明了 LTL 的 n- 可满足性是可判定的。

第三章先给出了 LTLC 的语法和语义，然后定义了 LTLC 的一个子语言 LTLC/B ，并利用 LTLC/B 给出了有穷状态反应系统在 LTLC 中的表示，这一章还证明了 LTLC/B- 公式的可满足性是可判定的，这个结果可用于有穷状态反应系统的模型检查和有穷状态反应系统之间的一致性检查。

第四章是关于实时系统验证的。本章先定义了 LTLC 的一个子语言 LTLC/R 。然后在第 2 节中给出了实时系统的一种数学模型：时间模块 (这里的一个时间模块大致相当于一个时间自动机) ，定义了时间模块所对应的 LTLC 公式，并通过时间模块对应的 LTLC 公式的语义定义了时间模块的语义，这一节还给出了利用 LTLC 的语义来证明实时系统的性质和 '求精' 关系的示例。本章的主要部分是第 3 节，这一节证明了 LTLC/R- 公式的可满足性是可判定的。由于 LTLC/R 中的公式可以表示有限控制状态实时系统，于是利用该判定算法可以对有穷控制状态实时系统进行模型检查，并且还能对这样两个实时系统之间的一致性进行检查，而以前的实时系统模型检查算法一般只能用来检查系统和性质之间的一致性。

第五章是关于混成系统验证的，类似于第四章的结构，本章先定义了 LTLC 的一个子语言 LTLC/H ，然后给出了混成系统的一种数学模型：混成模块 (这里的一个混成模块大致相当于一个混成自动机) ，定义了混成模块所对应的 LTLC 公式，并通过混成模块所对应的 LTLC 公式的语义模型定义了该模块的语义模型。第 3 节给出了利用 LTLC 的语义定义来证明混成系统性质的示例，之所以能这样做是因为系统和性质是用同一语言写的。第 4 节给出了多速率混成系模块关于样本模型的模型检查算法，样本模型是混成系统在样

本控制 (sampling control[50]) 下的模型。 T.A. Henzinger 和 P. W. Kopke 在 [50] 中认为混成系统的样本控制是比稠密时间控制 (dense-time control[50]) 更现实和更自然的控制模式，故研究混成系统在样本模型下的模型检查很有实用价值。多速率混成系统是一类重要的混成系统 (cf. [21]) ，但它的可达性是不可判定的 (进而关于稠密时间的模型检查问题是不可判定的) ，本章证明了多速率混成系统的样本模型检查问题是可判定的。

第六章是相关工作的一个简单介绍。

第七章是本文的总结和进一步的工作。

第二章　线性时序逻辑 LTL 简介

2.1　线性时序逻辑 LTL

根据本文的具体需要，我们介绍线性时序逻辑 LTL[68, 76] 的一个子部分如下，但为方便起见，仍使用 LTL 来称呼原 LTL 的这个子部分.

LTL 的字母包括以下符号及括号：

1. 命题常量：　$first$, $false$;
2. 命题变量：　$p, q, p_1, p_2, \ldots$;
3. 整型变量：　$x, x_1, x_2, \ldots$;
4. 常元：　　$0, 1, -1, 2, -2, 3, -3, \ldots$(整数);
5. 关系符号：　$\leq$(小于或等于) ;
6. 函数符号：　$+$(加) ，　$-$(取负), $*$(乘);
7. 联结词：　　$\neg$(否定), $\wedge$(合取);
8. 时序符号：　$^+$(下一时刻的值) ，　$^-$(前一时刻的值) ，　$\Box$(必然算子), $\mathcal{U}$(直到算子).

定义 2.1 *LTL* 的表达式归纳定义如下：

$$e ::= x \,|\, x^+ \,|\, x^- \,|\, m \,|\, (-e) \,|\, (e_1 + e_2) \,|\, (e_1 * e_2)$$

这里 x 是整型变量，　m 是常元。

定义 2.2 LTL 的公式归纳定义如下：

$$\varphi ::= first \,|\, false \,|\, p \,|\, p^{+} \,|\, p^{-} \,|\, (e_1 \leq e_2) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\Box\varphi) \,|\, (\varphi_1 \mathcal{U} \varphi_2)$$

这里 p 是命题变量， e_1, e_2 为表达式。

其它常用联结词、时序算子和关系符如 $\vee$（析取），$\Rightarrow$（蕴涵），$\Diamond$（终于），$\geq, =, <, >$ 等可在 LTL 中定义出来。

注：在 LTL 中，时序符号 '+' 和 ' −' 具有特殊的含义，它们分别用于表示变量在下一时刻和前一时刻的值，称 p^{+} 、x^{+} 等为带有上标 '+' 的变量，p^{-} 、x^{-} 等为带有上标 '−' 的变量，它们本身并不是 LTL 中的变量，它们在当前时刻的值分别等于相应的（不带上标的）变量在下一时刻和前一时刻的值。另外注意与下一章提出的 LTLC 中的不同，符号 ' ' 在 LTL 中无特殊的含义，p' 、x' 等在 LTL 中可用来表示与变量 p, x 等无任何关系的变量 (但在 LTLC 中却不能，见下一章关于 LTLC 的语法及语义的介绍) 。

定义 2.3 设 V_d 是 LTL 中的变量（包括命题变量和整型变量）的一个有穷集合， V_d 上的一个状态 π 是指 V_d 上的这样一个指派：对于 V_d 中的命题变量 p 有 $\pi(p) \in \{0, 1\}$ ，对于 V_d 中的整型变量 x 有 $\pi(x) \in \mathcal{Z}$(这里 $\mathcal{Z}$ 表示所有整数的集合) 。 V_d 上的一个 LTL 模型 Π 是 V_d 上的状态的一个无穷序列。

称一个公式（或表达式）是 V_d 上的，若该公式（表达式）中出现的变量全在变量集 V_d 中。

定义 2.4 设 $\Pi ::=< \pi_0, \pi_1, \pi_2, \ldots >$ 是 V_d 上的一个 LTL 模型，对于 V_d 上的任一表达式 e 及 $i \in \mathcal{N}$, 可以归纳定义 e 在 Π 的第 i 个状态下的值 $\Pi(e, i) \in \mathcal{Z}$ 如下：

1. $\Pi(x, i) ::= \pi_i(x)$.
2. $\Pi(x^{+}, i) ::= \pi_{i+1}(x)$.
3. $\Pi(x^{-}, i) ::= \begin{cases} \pi_i(x) & \text{若 } i = 0, \\ \pi_{i-1}(x) & \text{若 } i > 0. \end{cases}$
4. $\Pi(m, i) ::= m$.
5. $\Pi(-e, i) ::= -\Pi(e, i)$.
6. $\Pi(e_1 + e_2, i) ::= \Pi(e_1, i) + \Pi(e_2, i)$.
7. $\Pi(e_1 * e_2, i) ::= \Pi(e_1, i) * \Pi(e_2, i)$.

这里 x 是一个整型变量， m 是一个整数常元， e, e_1, e_2 为表达式。

定义 2.5 设 $\Pi ::=< \pi_0, \pi_1, \pi_2, \ldots >$ 是 V_d 上的一个 LTL 模型，对于 V_d 上的任一 LTL 公式 φ 及 $i \in \mathcal{N}$，可以归纳定义 φ 在 Π 的第 i 个状态下的值 $\Pi(\varphi, i) \in \{0, 1\}$ 如下：

1. $\Pi(first, i) ::= \begin{cases} 1 & \text{若 } i = 0; \\ 0 & \text{若 } i > 0. \end{cases}$
2. $\Pi(false, i) ::= 0.$
3. $\Pi(p, i) ::= \pi_i(p).$
4. $\Pi(p^+, i) ::= \pi_{i+1}(p).$
5. $\Pi(p^-, i) ::= \begin{cases} \pi_i(p) & \text{若 } i = 0; \\ \pi_{i-1}(p) & \text{若 } i > 0. \end{cases}$
6. $\Pi(e_1 \leq e_2, i) ::= \begin{cases} 1 & \text{若 } \Pi(e_1, i) \leq \Pi(e_2, i); \\ 0 & \text{否则.} \end{cases}$
7. $\Pi(\neg\varphi, i) ::= 1 - \Pi(\varphi, i).$
8. $\Pi(\varphi_1 \wedge \varphi_2, i) ::= \Pi(\varphi_1, i) * \Pi(\varphi_2, i).$
9. $\Pi(\Box\varphi, i) ::= \begin{cases} 1 & \text{若对任意 } j \geq i \text{ 有：} \Pi(\varphi, j) = 1; \\ 0 & \text{否则.} \end{cases}$
10. $\Pi(\varphi_1 \mathcal{U} \varphi_2, i) ::= \begin{cases} 1 & \text{若存在一个} j \geq i \text{ 使得} \Pi(\varphi_2, j) = 1 \text{ 且} \\ & \text{对任意满足 } i \leq k < j \text{ 的非负整数 } k \text{ 有：} \Pi(\varphi_1, k) = 1; \\ 0 & \text{否则.} \end{cases}$

这里 p 是一个命题变量， e_1 、 e_2 为表达式， φ 、 φ_1 、 φ_2 为 LTL 公式。

如果 $\Pi(\varphi, i) = 1$, 则称 φ 在时刻 i 时为真。如果 φ 在时刻 0 时为真, 即有 $\Pi(\varphi, 0) = 1$, 则称 Π 是 φ 的一个 LTL 模型，记为 $\Pi \models_{ltl} \varphi$.

定义 2.6 设 φ 是变量集 V_d 上的一个 LTL 公式.

(1). 如果存在 V_d 上的一个 LTL 模型 Π 使得 Π 是 φ 的一个 LTL 模型，则称 φ 是 LTL- 可满足的；否则称它是 LTL- 不可满足的。

(2). 如果 $\neg\varphi$ 是 LTL- 不可满足的，则称 φ 是永真的，记为 $\models_{ltl} \varphi$

定义 2.7 设 φ 和 ψ 是变量集 V_d 上的二个 LTL 公式。如果 φ 的所有（在变量集 V_d 上的）模型都是 ψ 的 LTL 模型，则称 ψ 是 φ 的一个逻辑结论，记作 $\varphi \models_{ltl} \psi$ 。

2.2 LTL 公式的可满足性的判定

定义 2.8 一个命题 LTL 公式 φ 是指由如下规则生成的一个 LTL 公式.

$$\varphi ::= first \,|\, false \,|\, p \,|\, p^{+} \,|\, p^{-} \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\Box\varphi) \,|\, (\varphi_1 \mathcal{U} \varphi_2)$$

这里 p 是命题变量。

关于命题 LTL 的可满足性有如下结果。

定理 2.1 *[89, 62]* 设 φ 是一个命题 LTL 公式，则 φ 的 LTL- 可满足性是可判定的。

定义 2.9 设 n 是一个正整数， V_d 是一个有穷变量集合 *(*含命题变量和整型变量*)* ，φ 是 V_d 上的一个 LTL 公式；称 φ 是 n- 可满足的，若存在 V_d 上的一个 LTL 模型 Π 使得

1 。 Π 是 φ 的一个 LTL 模型；

2 。对于 V_d 中的任意整型变量 x 及任意 $i \in \mathcal{N}$ 有： $-n \leq \Pi(x, i) \leq n$.

现在想要证明的是 LTL 公式的 n- 可满足性是可判定的。在考虑公式的 n- 可满足性时，我们只需要考虑每个整型变量取值在 $-n$ 和 n 之间的情况；利用整数在计算机中的二进制表示法，我们可通过多个命题变量（每个命题变量取值 0 和 1 ）来表示一个整型变量。这样就可把一个 LTL 公式 φ 的 n- 可满足性判定转化为另一个命题 LTL 公式 $\widehat{\varphi}$ 的可满足性判定，而后一问题根据定理 2.1 是可判定的。下面我们着手给出 $\widehat{\varphi}$ 的构造过程（在本节以下的构造过程中我们将相对固定正整数 n 及变量集 V_d ）。

对于正整数 n , 令 $m ::= \lfloor \log n \rfloor + 2$, 这里 log 表示以 2 为底的对数，$\lfloor \log n \rfloor$ 表示 $\log n$ 的整数部分。利用整数在计算机中的二进制表示法， m 个比特可以表示从 $-(2^{m-1}-1)$ 到 2^{m-1} 之间的所有整数。由于 $n \leq 2^{m-1}-1$ ，于是在考虑 n- 可满足性时，用 m 个命题变量表示一个整型变量就可以了。

对于 V_d 中的任一整型变量 x_i ，我们用 m 个不在 V_d 中出现的命题变量 p_{i1}, p_{i2},..., p_{im} 来表示 x_i ，并称这 m 个变量为 x_i 的命题分量。当用 $p_{i1}, p_{i2}, \ldots, p_{im}$ 的值来表示 x_i 的值时，我们相应地用 $p^{+}_{i1}, p^{+}_{i2}, \ldots, p^{+}_{im}$ 的值和 $p^{-}_{i1}, p^{-}_{i2}, \ldots, p^{-}_{im}$ 的值来分别表示 x^{+}_{i} 和 x^{-}_{i} 的值。

记 $W ::= \{\, p \mid p$ 是 V_d 中的某个整型变量的命题分量 $\}$，则 W 是一个命题变量集合。令 $W^+ ::= \{\ p^+ \mid p \in W\ \}$，$W^- ::= \{\ p^- \mid p \in W\ \}$。我们称从集合 $W \bigcup W^+ \bigcup W^-$ 到集合 $\{0, 1\}$ 上的一个映射为 $W \bigcup W^+ \bigcup W^-$ 上的一个指派。对于 $W \bigcup W^+ \bigcup W^-$ 上的任一指派 μ，我们以 τ_μ 表示集合 $\{\, v \mid v \in W \bigcup W^+ \bigcup W^-$ 且 $\mu(v) = 1\ \}$ 与集合 $\{\neg v \mid v \in W \bigcup W^+ \bigcup W^-$ 且 $\mu(v) = 0\ \}$ 的并集，并称集合 τ_μ 为 $W \bigcup W^+ \bigcup W^-$ 上的一个规范子句。若集合 W 的基数为 $|W|$，则 $W \bigcup W^+ \bigcup W^-$ 上共有 $2^{3|W|}$ 个不同的指派，每个指派被其在 $W \bigcup W^+ \bigcup W^-$ 上的规范子句所唯一地表示。

设 μ 是 $W \bigcup W^+ \bigcup W^-$ 上的一个指派，对于 V_d 中的任一整型变量 x，设 p_{i1}，p_{i2}，$\ldots$，p_{im} 是 x 的所有命题分量，由于 p_{i1}，p_{i2}，$\ldots$，p_{im} 皆属于 W，故 $p_{i1}, p_{i2}, \ldots, p_{im}$，$p^+_{i1}, p^+_{i2}, \ldots,\ p^+_{im}$ 及 p^-_{i1}，p^-_{i2}，$\ldots$，p^-_{im} 在 μ 下都有其指派值，这样利用整数在计算机中的二进制表示法便可得到 x、x^+ 和 x^- 的三个值 (分别记作 $\mu(x)$、$\mu(x^+)$ 和 $\mu(x^-)$，它们都是介于 $-(2^{m-1}-1)$ 与 2^{m-1} 之间的整数)。于是我们可以归纳定义 V_d 上的表达式 e 在 μ 下的值 $\lambda_\mu(e)$ 如下：

1. $\lambda_\mu(x) ::= \mu(x)$.
2. $\lambda_\mu(x^+) ::= \mu(x^+)$.
3. $\lambda_\mu(x^-) ::= \mu(x^-)$.
4. $\lambda_\mu(m) ::= m$.
5. $\lambda_\mu(-e) ::= -\lambda_\mu(e)$.
6. $\lambda_\mu(e_1 + e_2) ::= \lambda_\mu(e_1) + \lambda_\mu(e_2)$.
7. $\lambda_\mu(e_1 * e_2) ::= \lambda_\mu(e_1) * \lambda_\mu(e_2)$.

这里 x 是 V_d 中的一个整型变量，m 是一个整数常元，e, e_1, e_2 为表达式。

设 e_1、e_2 是 V_d 上的任意两个表达式，我们以 $\mathcal{T}(e_1, e_2)$ 来表示 $W \bigcup W^+ \bigcup W^-$ 上的规范子句集 $\{\tau_\mu \mid \mu$ 是 $W \bigcup W^+ \bigcup W^-$ 上的一个指派，且 $\lambda_\mu(e_1) \le \lambda_\mu(e_2)\ \}$。若把 $\mathcal{T}(e_1, e_2)$ 中的子句用联结词 '$\vee$' 连接起来，并把每个子句中的元素用联结词 '$\wedge$' 连接起来，则我们得到了 W 上的一个命题 LTL 公式 (见定义 2.8)，并称这个命题 LTL 公式是子句集 $\mathcal{T}(e_1, e_2)$ 所表示的命题 LTL 公式，我们把规范子句集等同于它所表示的命题 LTL 公式，两者在记法上也不作区分。顺便说明一下，若 $\mathcal{T}(e_1, e_2)$ 是空集，则它所表示的命题 LTL 公式为 $false$。

定义 2.10 令 $W_d ::= \{v \mid v \in W$ 或 v 是 V_d 中的一个命题变量 $\}$。对于 V_d 上的任一 *LTL* 公式 φ，归纳定义命题变量集 W_d 上的命题 *LTL* 公式 $\overline{\varphi}$ 如下：

1. $\overline{\varphi} ::= \varphi$ ，若 φ 为 $first$, $false$, p, p^+, p^- 中之一.
2. $\overline{e_1 \leq e_2} ::= \mathcal{T}(e_1, e_2)$.
3. $\overline{\neg\varphi} ::= \neg\overline{\varphi}$.
4. $\overline{\varphi_1 \wedge \varphi_2} ::= \overline{\varphi_1} \wedge \overline{\varphi_2}$.
5. $\overline{\Box\varphi} ::= \Box\overline{\varphi}$.
6. $\overline{\varphi_1 \mathcal{U} \varphi_2} ::= \overline{\varphi_1}\mathcal{U}\overline{\varphi_2}$.

   这里 p 是一个命题变量， e_1, e_2 为表达式， $\varphi, \varphi_1, \varphi_2$ 为 LTL 公式。

设 $\Delta ::=< \delta_0, \delta_1, \delta_2, \ldots, >$ 是 W_d 上的一个 LTL 模型, 由 W_d 中的变量的值与 V_d 中的变量的值之间的对应关系, 我们可得 V_d 上的一个 LTL 模型 $\Pi_\Delta ::=< \pi_0, \pi_1, \pi_2, \ldots >$ 。若 x 是 V_d 中的一个整型变量， $p_{i1}, p_{i2}, \ldots, p_{im}$ 是 x 的所有命题分量, 则 $\pi_k(x)$ 的值取决于 $\delta_k(p_{i1}), \delta_k(p_{i2}), \ldots, \delta_k(p_{im})$ 的值，后面的 m 个值是前者的值的二进制表示分量。

引理 2.1 设 φ 是 V_d 上的一个 LTL 公式, Δ 是 W_d 上的一个 LTL 模型, 则 $\Pi_\Delta \models_{ltl} \varphi$ 当且仅当 $\Delta \models_{ltl} \overline{\varphi}$ 。

证明: 只需证明对于 $k \in \mathcal{N}$ 及 V_d 上的任意两个表达式 e_1 和 e_2 有 $\Pi_\Delta(e_1 \leq e_2, k) = \Delta(\overline{e_1 \leq e_2}, k)$.

1. 当 $k > 0$ 时。对于 $v \in W$ ，定义 $\mu(v) ::= \delta_k(v)$ ， $\mu(v^-) ::= \delta_{k-1}(v)$ ， $\mu(v^+) ::= \delta_{k+1}(v)$ ；则得到 $W \bigcup W^+ \bigcup W^-$ 上的一个指派 μ ，而且对于 V_d 中的任意整型变量 x 有 $\pi_k(x) = \lambda_\mu(x)$ ， $\pi_{k-1}(x) = \lambda_\mu(x^-)$ 且 $\pi_{k+1}(x) = \lambda_\mu(x^+)$ 。不难看到对于这样定义的 μ 有 $\Delta(\tau_\mu, k) = 1$ 成立 (注意在规范子句中元素之间是 '合取' 关系)。

   - 若 $\Pi_\Delta(e_1 \leq e_2, k) = 1$ ，则 $\Pi_\Delta(e_1, k) \leq \Pi_\Delta(e_1, k)$ ，即 $\lambda_\mu(e_1) \leq \lambda_\mu(e_2)$ ，于是 $\tau_\mu \in \mathcal{T}(e_1, e_2)$ ，由于 $\Delta(\tau_\mu, k) = 1$ ，于是 $\Delta(\mathcal{T}(e_1, e_2), k) = 1$ ，即 $\Delta(\overline{e_1 \leq e_2}, k) = 1$ 。
   - 若 $\Pi_\Delta(e_1 \leq e_2, k) = 0$ ，则 $\Pi_\Delta(e_1, k) > \Pi_\Delta(e_1, k)$ ，即 $\lambda_\mu(e_1) > \lambda_\mu(e_2)$ ，于是 $\tau_\mu \notin \mathcal{T}(e_1, e_2)$ ，由于 $\Delta(\tau_\mu, k) = 1$, 且 $\mathcal{T}(e_1, e_2)$ 中的任何规范子句都和 τ_μ 矛盾，于是 $\Delta(\mathcal{T}(e_1, e_2), k) = 0$ ，即 $\Delta(\overline{e_1 \leq e_2}, k) = 0$ 。

2. 当 $k = 0$ 时。对于 $v \in W$, 定义 $\mu(v) ::= \delta_0(v)$, $\mu(v^-) ::= \delta_0(v)$, $\mu(v^+) ::= \delta_1(v)$; 则得到 $W \bigcup W^+ \bigcup W^-$ 上的一个指派 μ ，而且对于 V_d 中的任意整型变量 x 有 $\pi_0(x) = \lambda_\mu(x) = \lambda_\mu(x^-)$ 且 $\pi_1(x) = \lambda_\mu(x^+)$ 。以下的证明过程与 $k > 0$ 时的证明过程类似，不再详细写出。

□.

定义 2.11 对于 V_d 上的任一 *LTL* 公式 φ , 定义 $\widehat{\varphi} ::= \overline{\varphi} \wedge \bigwedge_{x \in U} \Box(\overline{-n \leq x} \wedge \overline{x \leq n})$, 这里 U 是 V_d 中的所有整型变量构成的集合。

定理 2.2 设 φ 是 V_d 上的一个 *LTL* 公式, 则 φ 是 n- 可满足的当且仅当 W_d 上的命题 *LTL* 公式 $\widehat{\varphi}$ 是可满足的。

证明:

1. 若 φ 是 n- 可满足的, 则存在 φ 的一个 LTL 模型 $\mathfrak{A} ::=< \nu_0, \nu_1, \nu_2, \ldots >$ 使得对于 V_d 中的任意整型变量 x 及任意 $k \in \mathcal{N}$ 有: $-n \leq \nu_k(x) \leq n$. 由于 $-(2^{m-1}-1) \leq -n < n < 2^{m-1}$, 于是根据 V_d 中的整型变量的值与其在 W_d 中的分量的值之间的对应关系我们可得到 W_d 上的一个 LTL 模型 $\Delta ::=< \delta_0, \delta_1, \delta_2, \ldots, >$ 使得 $\Pi_\Delta = \mathfrak{A}$ 。

   由于 $\mathfrak{A}$ 是 $\varphi \wedge \bigwedge_{x \in U} \Box(-n \leq x \wedge x \leq n)$ 的一个 LTL 模型, 故由引理 2.1 可得 Δ 是 $\overline{\varphi} \wedge \bigwedge_{x \in U} \Box(\overline{-n \leq x} \wedge \overline{x \leq n})$ 的一个 LTL 模型, 即 $\widehat{\varphi}$ 是可满足的。

2. 若 $\widehat{\varphi}$ 是可满足的, 设 $\Delta ::=< \delta_0, \delta_1, \delta_2, \ldots, >$ 是 $\widehat{\varphi}$ 的一个 LTL 模型, 则由引理 2.1 知 Π_Δ 是 $\varphi \wedge \bigwedge_{x \in U} \Box(-n \leq x \wedge x \leq n)$ 的一个模型, 于是 Π_Δ 是 φ 的一个模型且对于 V_d 中的任一整型变量 x 有 $-n \leq x \wedge x \leq n$, 即 φ 是 n- 可满足的。

□.

由定理 2.1 和由定理 2.2 立即有:

定理 2.3 设 n 是一个正整数, φ 是一个 *LTL* 公式, 则 φ 的 n- 可满足性是可判定的。

文献 [62] 中介绍了一个用来检查命题 LTL 公式的可满足性的算法, 该算法已经实现为一个可实际使用的验证工具, 作者用该工具对文献 [76] 中所有命题 LTL 公式进行了可满足性检查、并对 [76] 中的所有有限状态程序进行了模型检查。由于定义 2.8 定义的命题 LTL 公式是 [62] 中定义的命题 LTL 公式的一个子集, 所以我们可以利用 [62] 中的可满足性检查工具来检查本节定义的命题 LTL 公式的可满足性。此外对于一个给定的正整数 n , 我们也可以利用 [62] 中的工具来实现一个 LTL 公式的 n- 可满足性的检查工具, 并进而可利用该工具来对实时系统进行模型检查 (参见本文第四章的 4.3 节)。

第三章　连续语义线性时序逻辑 LTLC

3.1　预备概念

在本文中，$\mathcal{R}$ 、$\mathcal{R}^+$ 、$\mathcal{Q}$ 、$\mathcal{Z}$ 和 $\mathcal{N}$ 分别用来表示实数集、非负实数集、有理数集、整数集和非负整数集。

设 $f(t)$ 是一个实函数，我们以 $\lim_{t\to t_0-} f(t)$ 、$\lim_{t\to t_0+} f(t)$ 和 $\mathcal{D}f(t_0)$ 分别表示 $f(t)$ 在点 $t = t_0$ 处的左极限、右极限和（一阶）导数。若一个函数在某点的左（右）极限等于它在该点的函数值，则称函数在此点左（右）连续。

设 $f(t)$ 是一个在区间 (a, b) 上有定义的实函数。若对于任意 $t_1, t_2 \in (a, b)$ 有 $a < t_1 \leq t_2 < b$ 蕴涵 $f(t_1) \leq f(t_2)$ ，则称 f 在区间 (a, b) 上单调增加。同样，若对于任意 $t_1, t_2 \in (a, b)$ 有 $a < t_1 \leq t_2 < b$ 蕴涵 $f(t_1) \geq f(t_2)$ ，则称 f 在区间 (a, b) 上单调减少。称 f 在一个区间上单调若它在这个区间上单调增加或单调减少。

定义 3.1 称 $\mathcal{R}^+$ 上的实函数 $f(t)$ 是一个步函数，若存在无穷序列 $a_0, a_1, a_2, \ldots, a_n, \ldots$ 使得

1. $0 = a_0 < a_1 < a_2 < \cdots < a_n < \ldots$;

2. 对任意 $M \in \mathcal{N}$, 存在 $n \in \mathcal{N}$ 使得 $a_n > M$;

3. 对任意 $i \in \mathcal{N}$, 存在一常数 $d \in \mathcal{R}$ 使得对任意 $t \in (a_i, a_{i+1}]$ 有 $f(t) = d$ ；即 f 在区间 $(a_i, a_{i+1}]$ 上始终取一个固定的值.

由此定义可见，步函数在 $\mathcal{R}^+$ 上是左连续的，且它的（右）不连续点都是孤立点。

定义 3.2 设 f 是 $\mathcal{R}^+$ 上的一个步函数。

1. 称 f 是一个布尔值步函数，若 f 的值域为集合 $\{0, 1\}$.

2. 称 f 是一个整值步函数，若 f 的值域为整数集 $\mathcal{Z}$.

定义 3.3 称 $\mathcal{R}^+$ 上的实函数 $f(t)$ 是一个时钟函数, 若存在有穷序列 $a_0, a_1, a_2, \ldots, a_n$ 使得 $0 = a_0 < a_1 < a_2 < \cdots < a_n$ 且

$$f(t) = \begin{cases} 0 & \text{若 } t = 0 \\ t - a_i & \text{若 } t \in (a_i, a_{i+1}] \\ t - a_n & \text{若 } t > a_n \end{cases}$$

或者存在无穷序列 $a_0, a_1, a_2, \ldots, a_n, \ldots$ 使得

1. $0 = a_0 < a_1 < a_2 < \cdots < a_n < \ldots$;
2. 对任意$M \in \mathcal{N}$, 存在$n \in \mathcal{N}$ 使得$a_n > M$;
3. $f(t) = \begin{cases} 0 & \text{若 } t = 0 \\ t - a_i & \text{若 } t \in (a_i, a_{i+1}] \end{cases}$

与步函数类似, 时钟函数在 $\mathcal{R}^+$ 上也是左连续的, 且它的 (右) 不连续点都是孤立点。

定义 3.4 设 $f(t)$ 是 $\mathcal{R}^+$ 上的一个实函数, 称 $f(t)$ 是 $\mathcal{R}^+$ 上的一个正规函数, 若存在无穷序列 $a_0, a_1, a_2, \ldots, a_n, \ldots$ 使得

1. $0 = a_0 < a_1 < a_2 < \cdots < a_n < \ldots$;
2. 对任意 $M \in \mathcal{N}$, 存在 $n \in \mathcal{N}$ 使得 $a_n > M$;
3. f 在每个开区间 (a_i, a_{i+1}) 上是可微的, 且导函数 $\mathcal{D}f(t)$ 在 (a_i, a_{i+1}) 上连续;
4. 在每个开区间 (a_i, a_{i+1}) 的端点上, 导函数 $\mathcal{D}f(t)$ 的单侧极限存在, 即右极限 $\lim_{t \to a_i +} \mathcal{D}f(t)$ 和左极限 $\lim_{t \to a_{i+1} -} \mathcal{D}f(t)$ 存在。

定义 3.5 设 f 是 $\mathcal{R}^+$ 上的一个步函数或时钟函数。定义一个伴随于 f 的函数 $f' : \mathcal{R}^+ \mapsto \mathcal{R}$ 如下:

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

注: 在这里 f' 并不是 f 的导函数。

定义 3.6 设 f 是 $\mathcal{R}^+$ 上的一个正规函数。定义两个伴随于 f 的函数 $f' : \mathcal{R}^+ \mapsto \mathcal{R}$ 和 $\dot{f} : \mathcal{R}^+ \mapsto \mathcal{R}$ 如下:

1. 对任意 $t_0 \in \mathcal{R}^+$, $f'(t_0) ::= \lim_{t \to t_0 +} f(t)$.

2. $\dot{f}(t_0) ::= \begin{cases} \lim_{t\to 0+} \mathcal{D}f(t) & \text{若 } t_0 = 0, \\ \lim_{t\to t_0-} \mathcal{D}f(t) & \text{若 } t_0 > 0. \end{cases}$

注：上述定义中的 $\dot{f}$ 基本上可以当作 f 的导函数来看待，它和 f(真正) 的导函数 $\mathcal{D}f(t)$ 的差异只在 f 的不可微的点上，在这些点上 $\mathcal{D}f(t)$ 没有定义而 $\dot{f}$ 却是有定义的；在 f 的所有可微点上它们两者是一致的。但注意上述定义中的 f' 却与 f 的导函数无任何联系，它是一个与 f 相关的函数，在 f 的连续点上 f' 与 f 相等，在 f 的不连续点上 f' 等于 f 在该点的右极限。

3.2　LTLC 的语法

LTLC 的字母包括括号和以下的一些符号：

1. 全局时钟：　t ；
2. 刚性变量：　$u, u_1, u_2, \ldots$ ；
3. 布尔型变量：　$p, q, p_1, p_2, \ldots$ ；
4. 整型变量：　$z, z_1, z_2, \ldots$;
5. 时钟变量：　$c, c_1, c_2, \ldots$;
6. 柔性变量：　$x, y, x_1, x_2, \ldots$;
7. 常元符号：　$\{\bar{r} \,|r \in \mathcal{Q}\}$ ；
8. 函数符号：　$+$(加), $-$(取负), $*$（乘）,　Tr（取整）;
9. 关系符号：　$=$(等于),　$\leq$(小于等于) ；
10. 联结词：　　$\neg$(否定), $\wedge$(合取) ；
11. 量词：　　$\exists$(存在) ；
12. 时序算子：　$\Box$(必然算子), $\mathcal{U}$(直到算子) ；
13. 其它符号：　$\prime$ (撇，新值符号) , $\dot{}$ (点，微分符号).

说明：全局时钟 t 用于表示全局时间的流逝。为了处理方便起见，本文并不把全局时钟 t 看成是一个变量，而对它作特殊处理。

定义 3.7 *LTLC* 的项归纳定义如下：

$e ::= t \,|\, \bar{r} \,|\, u \,|\, z \,|\, z' \,|\, c \,|\, c' \,|\, x \,|\, x' \,|\, \dot{x} \,|\, (-e) \,|\, Tr(e) \,|\, (e_1 + e_2) \,|\, (e_1 * e_2)$

这里 t 是全局时钟，　$\bar{r}$ 是一个常元符号，　u 是一个刚性变量，　z 是一个整型变量，c 是一个时钟变量，　x 是一个柔性变量。

定义 3.8 *LTLC* 的公式归纳定义如下：

$\varphi ::= p \mid p' \mid (e_1 = e_2) \mid (e_1 \leq e_2) \mid (\neg\varphi) \mid (\varphi_1 \wedge \varphi_2) \mid (\Box\varphi) \mid (\varphi_1 \mathcal{U} \varphi_2) \mid (\exists u.\varphi)$

这里 p 是一个布尔型变量， u 是一个刚性变量， e_1 和 e_2 是项。

其它一些常见的联结词、量词和时序算子可通过上面的联结词、量词和时序算子定义出来，如：

1. 析取联结词：$\varphi_1 \vee \varphi_2 ::= \neg((\neg\varphi_1) \wedge (\neg\varphi_2))$

2. 蕴涵联结词：$\varphi_1 \Rightarrow \varphi_2 ::= (\neg\varphi_1) \vee \varphi_2$

3. 等价联结词：$\varphi_1 \Leftrightarrow \varphi_2 ::= (\varphi_1 \Rightarrow \varphi_2) \wedge (\varphi_2 \Rightarrow \varphi_1)$

4. 全称量词：$\forall u.\varphi ::= \neg(\exists u.(\neg\varphi)))$

5. 终于算子：$\Diamond\varphi ::= \neg(\Box(\neg\varphi)))$

此外，一些常用的函数和关系符号（如 $\geq, <, >$ 等）也可以作为已有公式的缩写在 LTLC 中定义出来。

注：在 LTLC 中， p' 、 c' 、 x' 等称为带撇的变量， $\dot{x}$ 、 $\dot{y}$ 等称为带点的变量，其实它们并不是 LTLC 中的变量，对它们的语义解释要依赖于其相应的（不带撇和点的）变量的解释。

为方便起见，我们常用 $var(\varphi)$ 和 $var(e)$ 来分别表示公式 φ 和项 e 中出现的所有变量所组成的集合。设 V 是一个变量集，如果 $var(e) \subseteq V$，则称 e 是变量集 V 上的项；类似地，如果 $var(\varphi) \subseteq V$，则称 φ 是变量集 V 上的公式。

3.3 LTLC 的语义

LTLC 中的项和公式都是在实数域 $\mathcal{R}$ 上解释的。常元符号 $\bar{r}$ 被解释为有理数 r ； '=' 解释为实数间的 '等于' 关系， '$\leq$' 解释为实数间的 '小于或等于' 关系，函数符号 $+$ 、 $-$ 、 $*$ 和 Tr 分别被解释为 '加法' 运算、 '取负' 运算、 '乘法' 运算以及 '取整' 运算。

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 $\mathcal{R}^+$ 上的一个布尔值步函数，整型变量被解释为时间域 $\mathcal{R}^+$ 上的一个整值步函数，时钟变量被解释为时间域 $\mathcal{R}^+$ 上的一个时钟函数，柔性变量被解释为时间域 $\mathcal{R}^+$ 上的一个左连续正规函数，刚性变量被解释为一个实数（可看作为时间域 $\mathcal{R}^+$ 上的一个常函数，它的值不随时间而变化）。若布尔型变量 q 和整型变量 z 被解释为布尔值步函数 f_q 和整值步函数 f_z，则 q' 和 z'

将分别被解释为与 f_q 和 f_z 相伴随的两函数 f_q' 和 f_z'(见定义 3.5)；类似地，若时钟变量 c 被解释为时钟函数 f_c, 则 c' 将被解释为与 f_c 相伴随的函数 f_c' 。若柔性变量被解释为左连续正规函数 f_x, 则 x' 和 $\dot{x}$ 将分别被解释为与 f_x 相伴随的两函数 f_x' 及 $\dot{f}_x$ (见定义 3.6)。

以下我们以 $\mathcal{F}_b$ 、 $\mathcal{F}_z$ 、 $\mathcal{F}_c$ 和 $\mathcal{F}_{lr}$ 来分别表示 $\mathcal{R}^+$ 上的所有布尔值步函数、所有整值步函数、所有时钟函数和所有左连续正规函数所组成的集合。

定义 3.9 设 V 是一个有穷的变量集合。称 $\mathcal{J} =< I, \sigma >$ 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 $\mathcal{R}$ 中的一个数作为 u 的解释，即 $I(u) \in \mathcal{R}$.
2. 对 V 中的任一布尔型变量 q, 映射 σ 指派 $\mathcal{R}^+$ 上的一个布尔值步函数 f_q 作为 q 的解释，即 $\sigma(q) = f_q \in \mathcal{F}_b$.
3. 对 V 中的任一整型变量 z, 映射 σ 指派 $\mathcal{R}^+$ 上的一个整值步函数 f_z 作为 z 的解释，即 $\sigma(z) = f_z \in \mathcal{F}_z$.
4. 对 V 中的任一时钟变量 c, 映射 σ 指派 $\mathcal{R}^+$ 上的一个时钟函数 f_c 作为 c 的解释，即 $\sigma(c) = f_c \in \mathcal{F}_c$ 。
5. 对 V 中的任一柔性变量 x, 映射 σ 指派 $\mathcal{R}^+$ 上的一个左连续正规函数 f_x 作为 x 的解释，即 $\sigma(x) = f_x \in \mathcal{F}_{lr}$ 。

定义 3.10 设 $\mathcal{J} =< I, \sigma >$ 是变量集 V 上的一个 $LTLC$- 模型， e 是 V 上的一个项。对于任意 $t_0 \in \mathcal{R}^+$, 时刻 $t = t_0$ 时 e 在模型 $\mathcal{J}$ 下的值 $\mathcal{J}(e, t_0)$ 归纳定义如下：

1. $\mathcal{J}(t, t_0) ::= t_0$.
2. $\mathcal{J}(\bar{r}, t_0) ::= r$.
3. 对任意刚性变量 u 有：$\mathcal{J}(u, t_0) ::= I(u)$.
4. 对任意整型变量 z 有：

   $\mathcal{J}(z, t_0) ::= f_z(t_0)$,

   $\mathcal{J}(z', t_0) ::= f_z'(t_0)$.
5. 对任意时钟变量 c 有：

   $\mathcal{J}(c, t_0) ::= f_c(t_0)$,

$\mathcal{J}(c', t_0) ::= f'_c(t_0)$.

6. 对任意柔性变量 x 有：

$\mathcal{J}(x, t_0) ::= f_x(t_0)$,

$\mathcal{J}(x', t_0) ::= f'_x(t_0)$,

$\mathcal{J}(\dot{x}, t_0) ::= \dot{f}_x(t_0)$.

7. $\mathcal{J}(-e, t_0) ::= -\mathcal{J}(e, t_0)$,

$\mathcal{J}(Tr(e), t_0) ::= \mathcal{J}(e, t_0)$ 的整数部分,

$\mathcal{J}(e_1 + e_2, t_0) ::= \mathcal{J}(e_1, t_0) + \mathcal{J}(e_2, t_0)$,

$\mathcal{J}(e_1 * e_2, t_0) ::= \mathcal{J}(e_1, t_0) * \mathcal{J}(e_2, t_0)$.

为方便起见，下面我们将常元符号 $\bar{r}$ 和它的解释 (有理数 r) 在书写时可互相代替，不作区分，如 '$2/3$' 既可以表示有理数 $2/3$, 也可以表示 $2/3$ 所对应的常元符号 $\overline{2/3}$.

定义 3.11 设 $\mathcal{J} =< I, \sigma >$ 是变量集 V 上的一个 *LTLC*- 模型， φ 是 V 上的一个公式。对于任意 $t_0 \in \mathcal{R}^+$, 时刻 $t = t_0$ 时 φ 在模型 $\mathcal{J}$ 下的值 $\mathcal{J}(\varphi, t_0)$ 归纳定义如下：

1. $\mathcal{J}(p, t_0) ::= f_p(t_0)$

2. $\mathcal{J}(p', t_0) ::= f'_p(t_0)$

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

4. $\mathcal{J}(e_1 \leq e_2, t_0) ::= \begin{cases} 1 & \text{若 } \mathcal{J}(e_1, t_0) \leq \mathcal{J}(e_2, t_0); \\ 0 & \text{否则.} \end{cases}$

5. $\mathcal{J}(\neg\varphi, t_0) ::= 1 - \mathcal{J}(\varphi, t_0)$.

6. $\mathcal{J}(\varphi_1 \wedge \varphi_2, t_0) ::= \mathcal{J}(\varphi_1, t_0) * \mathcal{J}(\varphi_2, t_0)$.

7. $\mathcal{J}(\Box\varphi,\ t_0) ::= \begin{cases} 1 & \text{若对任意 } t_1 \geq t_0 \text{ 有：} \mathcal{J}(\varphi, t_1) = 1; \\ 0 & \text{否则.} \end{cases}$

8. $\mathcal{J}(\varphi_1 \mathcal{U} \varphi_2, t_0) ::= \begin{cases} 1 & \text{若存在一个} t_1 \geq t_0 \text{ 使得} \mathcal{J}(\varphi_2,\ t_1) = 1 \text{ 且} \\ & \text{对任意 } t_2 \in [t_0, t_1) \text{ 有：} \mathcal{J}(\neg\varphi_1 \wedge \neg\varphi_2,\ t_2) = 0; \\ 0 & \text{否则.} \end{cases}$

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

这里 $\mathcal{J}[a/u] =< I[a/u], \sigma >$ 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 $\mathcal{J}(\varphi, t_0) = 1$，则称 φ 在时刻 $t = t_0$ 时为真。如果 φ 在时刻 $t = 0$ 时为真，则称 $\mathcal{J}$ 是 φ 的一个 (LTLC-) 模型。

定义 3.12 设 φ 是变量集 V 上的一个公式.

1. 如果存在 V 上的一个 $LTLC$ 模型 $\mathcal{J}$ 使得 $\mathcal{J}$ 是 φ 的一个 *(LTLC-)* 模型，则称 φ 是 *(LTLC-)* 可满足的；否则称它是 *(LTLC-)* 不可满足的。

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

定义 3.13 设 φ 和 ψ 是变量集 V 上的二个公式。如果 φ 的所有（在变量集 V 上的）模型都是 ψ 的模型，则称 ψ 是 φ 的一个逻辑结论，记作 $\varphi \models \psi$ 。

显然，ψ 是 φ 的一个逻辑结论当且仅当 $\varphi \wedge \neg\psi$ 是不可满足的。

定理 3.1 下面的各式在 $LTLC$ 中成立.

1. $\models \Box\Box\varphi \Leftrightarrow \Box\varphi \qquad \models \Diamond\Diamond\varphi \Leftrightarrow \Diamond\varphi$

2. $\models \neg\Box\varphi \Leftrightarrow \Diamond\neg\varphi \qquad \models \neg\Diamond\varphi \Leftrightarrow \Box\neg\varphi$

3. $\models \Box\varphi \Rightarrow \varphi \qquad \models \varphi \Rightarrow \Diamond\varphi$

4. $\models \Box(\varphi \wedge \psi) \Leftrightarrow (\Box\varphi) \wedge (\Box\psi)$

5. $\models \Diamond(\varphi \vee \psi) \Leftrightarrow (\Diamond\varphi) \vee (\Diamond\psi)$

6. $\models (\Box(\varphi \Rightarrow \psi) \wedge \Box\varphi) \Rightarrow \Box\psi$

7. 若 $\models \varphi \Rightarrow \psi$，且 $\theta \models \Box\varphi$，则 $\theta \models \Box\psi$

证明。由定义 3.12 容易证出，详细过程此处略。

利用带有时间界限的时序算子 (bounded-operator [56, 5]) 可以比较方便地表示出实时系统和混成系统的一些实时性质 (定量性质)，如有界响应性、有界不变量 [56, 78] 等。这些带有时间界限的时序算子 (简称有界算子) 可作为 LTLC 中的缩写公式在 LTLC 中定义出来，如：

定义 3.14

1. $\Box_{[a,b]}\varphi ::= \exists u.(t = u \wedge \Box((a + u \leq t \wedge t \leq b + u) \Rightarrow \varphi))$. 这里 a, b 是两个非负有理数且 $a \leq b$ ， u 是一个不在 φ 中出现的刚性变量。

2. $\Box_{\geq a}\varphi ::= \exists u.(t = u \wedge \Box(t \geq u + a \Rightarrow \varphi))$. 这里 a 是一个非负有理数， u 是一个不在 φ 中出现的刚性变量。

3. $\Diamond_{[a,b]}\varphi ::= \exists u.(t = u \wedge \Diamond(a + u \leq t \wedge t \leq b + u \wedge \varphi))$. 这里 a, b 是两个非负有理数且 $a \leq b$ ， u 是一个不在 φ 中出现的刚性变量。

4. $\Diamond_{\geq a}\varphi ::= \exists u.(t = u \wedge \Diamond(t \geq u + a \wedge \varphi))$. 这里 a 是一个非负有理数， u 是一个不在 φ 中出现的刚性变量。

其它的有界算子 (如 $\Box_{(a,b]}\varphi$, $\Box_{<a}\varphi$ ， $\Box_{>a}\varphi$, $\Diamond_{\leq a}\varphi$ 等) 也可类似地定义出来，详细定义此处从略。

说明：使用带有量词的公式 (如 $\exists u.\varphi$) 对于表示实时性质是不可缺少的，但使用存在量词在许多情况下会导致不可判定性；由于本文主要关心 (实时和混成系统的) 算法验证，故在以后的部分 (第四章和第五章) 我们基本上不考虑带量词的公式，这样变量集 V 中将不含刚性变量，这时 LTLC 模型 $\mathcal{J} =< I, \sigma >$(见定义 3.9) 中的第一部分 I 将为空，故在定义 V 上的模型 $\mathcal{J}$ 时只需定义第二部分 σ 即可。

LTLC 中最简单的一类公式是下面定义的 LTLC/B- 公式, 这些公式中不含整型变量、时钟变量、柔性变量及刚性变量。作为讨论 LTLC 公式的可满足性的开始，从简单的公式类开始是比较合适的。虽然 LTLC/B- 公式比较简单，但用它可以表示出有穷状态反应系统以及它们的时序性质。下面 3.4 节将讨论利用 LTLC/B- 公式表示有穷状态反应系统的问题， 3.5 节将讨论 LTLC/B- 公式的可满足性判定。

定义 3.15 一个 *LTLC/B-* 公式是指由下面规则生成的一个 *LTLC* 公式 φ.

$\varphi ::= p \,|\, p' \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\Box\varphi) \,|\, (\varphi_1 \mathcal{U} \varphi_2)$.

一个 *LTLC/B-* 状态公式是指由下面规则生成的一个 *LTLC* 公式 φ.

$\varphi ::= p \,|\, p' \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

一个 $LTLC/B$- 当前状态公式是指由下面规则生成的一个 $LTLC$ 公式 φ.

$\varphi ::= p \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

3.4　利用 LTLC/B- 公式表示有穷状态反应系统

按照公平转换模型 [76]，一个反应系统 [76] 可由这样的一个五元组 $P ::=< V, \Theta, \mathcal{T}, \mathfrak{J}, \mathcal{C} >$ 来描述。

- $V = \{v_1, v_2, \ldots, v_n\}$ 是一个有穷变量集合。V 上的一个状态是指 V 上的一个映射，它给 V 中每个变量指派一个值。
- Θ 是初始条件。它是一个可满足的断言，满足该断言的状态称为系统的初始状态。
- $\mathcal{T}$ 是转换的一个有穷集合。每一个 $\tau \in \mathcal{T}$ 是变量的当前值与变量的新值之间的一个转换关系，如 $(v_1 = 1 \wedge v_2 = v_3) \wedge (v_1' = 0 \wedge v_2' = v_2 \wedge v_3' = v_3 + 1)$，此式表示当此转换发生时，变量的当前值必须满足条件 $v_1 = 1 \wedge v_2 = v_3$，转换发生后，v_1 的新值为 0，v_2 的新值等于当前值，v_3 的新值等于当前值加 1。$\mathcal{T}$ 中的每个转换有一个卫式条件（如上面的 $v_1 = 1 \wedge v_2 = v_3$）和一个指派命令（如上面的 $v_1' = 0 \wedge v_2' = v_2 \wedge v_3' = v_3 + 1$），卫式条件是转换发生的一个必要条件（不是充分条件），只有当这个条件得到满足时（此时称该转换是使能的），转换才可能真正发生 (即指派命令被执行)。
- $\mathfrak{J}$ 是 $\mathcal{T}$ 的一个子集, 称为弱公平转换集, $\mathfrak{J}$ 中的转换必须满足弱公平性 (week fairness) 要求。一个转换 τ 满足弱公平性要求是指这样的情况不能发生：从某个时刻起 τ 一直是使能的，但从该时刻起 τ 却一直未能发生（即其指派命令未能被执行）。
- $\mathcal{C}$ 是 $\mathcal{T}$ 的一个子集，称为强公平转换集，$\mathcal{C}$ 中的转换必须满足强公平性 (strong fairness) 要求。一个转换 τ 满足强公平性要求是指这样的情况不能发生：从任意时刻起 τ 总有使能的时候，但在某时刻之后 τ 却一直未能发生。

为了表示反应系统与它的环境（另外的一个或数个反应系统）之间的交互，我们把每个反应系统的变量分为二类，一类叫控制变量 (controlled variable)，另一类叫外部变量（external variable）。前一类变量的值只能被系统自身所改变，不能被它的环境改变；而后一类变量的值只能被环境所改变，不能被系统自身改变。系统的外部变量对该系统而言只是可读的，只有控制变量才是可写的。由于本节只讨论有穷状态反应系统，故只需要考

虑布尔型变量就够用了，所以下面所说的变量（不论控制变量还是外部变量）一般都是指布尔型变量。

控制变量的值是通过转换的作用来改变的。每个反应系统包含有数个跳跃 (jump) 转换和一个叫做踏步的特殊转换。在 LTLC 的语义框架中，跳跃转换的发生不占用时间段，在其作用的瞬间 (零时段内) 它可改变所有控制变量的值。但踏步转换的每次发生具有一个正的时间段，在其作用的这个时段内，所有控制变量的值保持不变。在任意一个有限时间段中，跳跃转换只能发生有限次，相邻发生的两次跳跃转换之间为踏步转换的作用时段。

只含布尔型变量的有限状态反应系统的跳跃转换在 LTLC 中可表示为形如 '$guard \wedge assignment$' 的一个 LTLC/B- 公式, 其中 $guard$ 是一个(LTLC/B-)当前状态公式, 它是跳跃转换的使能条件(enabling condition), 它用于表示跳跃转换发生时系统的变量(包括控制变量以及外部变量)应满足的约束条件; $assignment$ 是一个(LTLC/B-)状态公式, 它是跳跃转换的命令部分, 相当于我们常见的赋值语句, 它给系统的每个控制变量赋予一个新值(当然新值可以等于原值), 通常假定每个跳跃转换在其作用时, 至少改变了系统的一个控制变量的值。为了直观和清楚起见，我们常把跳跃转换写成 '$\tau : guard \to assignment$' 的形式，其中 τ 是转换的名字， $guard$ 是转换的条件部分， $assignment$ 是转换的命令部分，注意这里的 '$\to$' 并不是一个逻辑联结词，它只是为了书写清楚起见而引入的一个分隔符，用来将转换的条件部分和命令部分分隔开来。

踏步转换不改变系统的所有控制变量的值，它可以表示成 $\bigwedge_{p \in V_c}(p' = p)$ 的形式，其中 V_c 是系统的所有控制变量的集合， $p' = p$ 是 LTLC/B- 公式 $(p' \wedge p) \vee (\neg p' \wedge \neg p))$ 的简写形式。在表示具体的反应系统时，通常将踏步转换省略不写。

一个有穷状态反应系统可写成如下的形状.

module	$module\_name$
external	$\{variable\_name\colon type\}^*$
controlled	$\{variable\_name\colon type\}^*$
init	$init\_cond$
jump	$\{trans\_name :\ guard \to assignment\}^*$
WF	$\{trans\_name\}^*$
SF	$\{trans\_name\}^*$

这里： $init\_cond$ 是反应系统的初始条件，它是一个当前状态公式，用于表示系统在初始时刻 $t = 0$ 时其控制变量所应满足的条件； external 和 controlled 后面分别是系统的外部变量声明和系统的控制变量声明， $type$ 表示变量的类型 (布尔型变量的类型记为 boolean)； WF 和 SF 后面分别是要求其满足弱公平性和强公平性的跳跃转换的列表。

现在我们利用语言 LTLC 给出有穷反应系统的语义。对于跳跃转换 '$\tau : guard \to assignment$' , 它所对应的 LTLC 公式为 '$guard \wedge assignment$', 简记为 TLF($\tau$) 。转换 τ

满足弱公平性在 LTLC 中可用时序公式 $(\Box\Diamond(guard \wedge assignment)) \vee (\Box\Diamond\neg guard)$ 来表示, 以下将此时序公式记为 $\mathrm{WF}(\tau)$; τ 满足强公平性在 LTLC 中可用时序公式 $(\Box\Diamond(guard \wedge assignment)) \vee (\Diamond\Box\neg guard)$ 来表示，以下将此时序公式记为 $\mathrm{SF}(\tau)$ 。

定义 3.16 设 M 是一个只含布尔型变量的反应系统，$\tau_0, \tau_1, \ldots, \tau_{n-1}$ 是 M 的所有跳跃转换, 我们称 $LTLC$- 公式 $init\_cond \wedge (\Box(V_c = V_c' \vee \bigvee_{j<n} TLF(\tau_j))) \wedge (\bigwedge_{\tau\in\mathfrak{J}} WF(\tau)) \wedge (\bigwedge_{\tau\in\mathcal{C}} SF(\tau))$ 为 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， $V_c = V_c'$ 是时序公式 $\bigwedge_{p\in V_c}(p' = p)$ 的缩写形式， $\mathfrak{J}$ 是 M 的弱公平转换集， $\mathcal{C}$ 是 M 的强公平转换集。以下以 $TLF(M)$ 来表示这个对应于反应系统 M 的时序逻辑公式。

我们将有穷状态反应系统 M 等同于它所对应的时序逻辑公式 $\mathrm{TLF}(M)$, 并将 $\mathrm{TLF}(M)$ 的语义模型作为 M 的语义模型。

定义 3.17 设 M 是一个有穷状态反应系统, φ 是一个 $LTLC/B$- 公式。如果 $TLF(M) \models \varphi$, 则称 φ 是 M 的一个性质，记作 $M \models \varphi$ 。

由上面这个定义和定义 3.13 可知，若想证明反应系统 M 具有性质 φ, 只需证明 φ 是 $\mathrm{TLF}(M)$ 的逻辑结论.

定义 3.18 设 M_1 和 M_2 是两个有穷状态反应系统， M_1 和 M_2 有相同的外部变量集, 且 M_1 的控制变量集是 M_2 的控制变量集的子集。如果 $TLF(M_2) \models TLF(M_1)$, 则称 M_2 是 M_1 的一个 '精化' (*refinement*)。

定义 3.19 M_1 和 M_2 是两个只含布尔型变量的有穷状态反应系统，如果 M_1 和 M_2 没有共同的控制变量, 则称它俩是相容的。称 n 个只含布尔型变量的有穷状态反应系统 M_1, M_2,..., M_n 是相容的若它们两两相容，并用 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 来表示这 n 个反应系统的并行复合。

定义 3.20 对于 $LTLC/B$- 公式 φ , 若 $TLF(M_1) \wedge TLF(M_2) \wedge \cdots \wedge TLF(M_n) \models \varphi$, 则称 φ 是 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 的一个性质。

下面通过一个简单的例子来看如何利用定义 3.20 来证明反应系统的性质。

例 3.1(互斥问题， mutual exclusion) 图 3.1 和 3.2 表示了两个共享同一资源的并发进程，每个进程有三种状态： outC 、 reqC 和 inC, 分别表示进程处在临界区 (critical section) 外、正在申请进入临界区和正处在临界区内。进程开始时都处在状态 outC, 随后

它们便按各自的步调进入状态 reqC 和 inC，此后便再次进入 outC 。进程在状态 reqC 和 inC 中每次可停留一个有限长的、不确定的时间段，在 outC 中可停留任意长（有穷或无穷）的时间段。 q 是一个来自两进程之外的调控信号，取值 0 或 1 。进程 1 在进入临界区时必须满足条件 '进程 2 此时在状态 outC 中，或者进程 2 此时虽在状态 reqC 中但 q 的值为 0'，进程 2 在进入临界区时必须满足条件 '进程 1 此时在状态 outC 中，或者进程 1 此时在状态 reqC 中但 q 的值为 1' 。

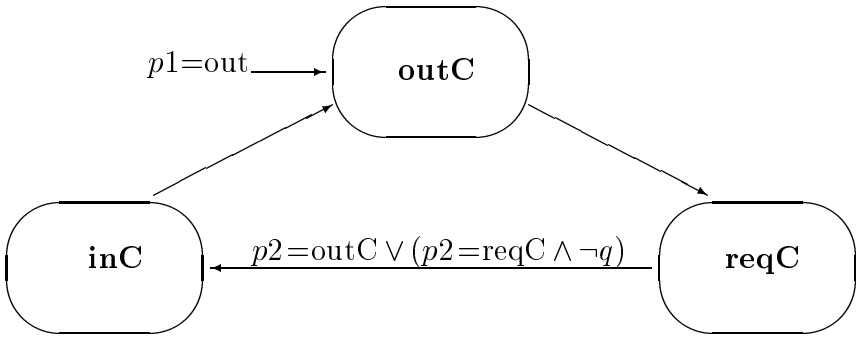

图 3.1. 进程 P_1

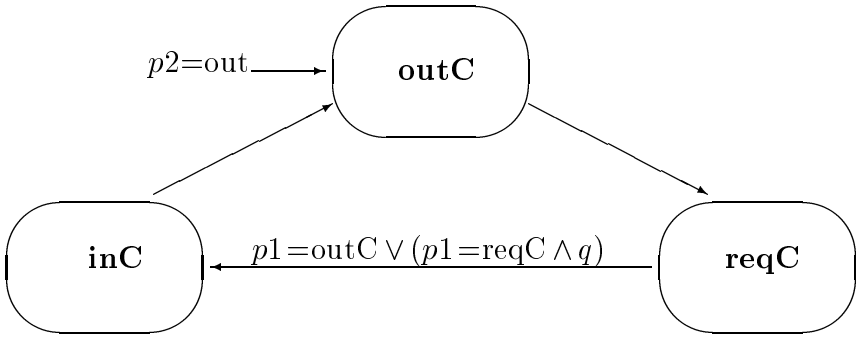

图 3.2. 进程 P_2

根据前面的讨论，这两个进程可分别表示为下面的反应系统 P_1 和 P_2.

module	P_1
external	$q : boolean$;
	$p2 : \{\text{outC,reqC, inC}\}$
controlled	$p1 : \{\text{outC,reqC, inC}\}$
init	$p1 = \text{outC}$
jump	$\alpha_1 :\ p1 = \text{outC} \to p1' = \text{reqC}$;
	$\alpha_2 :\ p1 = \text{reqC} \wedge (p2 = \text{outC} \vee (p2 = \text{reqC} \wedge \neg q)) \to p1' = \text{inC}$;
	$\alpha_3 :\ p1 = \text{inC} \to p1' = \text{outC}$
WF	$\{\alpha_3\}$
SF	$\{\alpha_2\}$

module	P_2
external	$q : boolean$;
	$p1 : \{\text{outC,reqC, inC}\}$
controlled	$p2 : \{\text{outC,reqC, inC}\}$
init	$p2 = \text{outC}$
jump	$\beta_1 :\ p2 = \text{outC} \to p2' = \text{reqC}$;
	$\beta_2 :\ p2 = \text{reqC} \wedge (p1 = \text{outC} \vee (p1 = \text{reqC} \wedge q)) \to p2' = \text{inC}$;
	$\beta_3 :\ p2 = \text{inC} \to p2' = \text{outC}$
WF	$\{\beta_3\}$
SF	$\{\beta_2\}$

在上面的进程中跳跃转换 α_3 和 β_3 要求满足弱公平性，α_2 和 β_2 要求满足强公平性。弱公平性保证了进程不会永远停留在 inC 状态，强公平性保证了每个进程的每次 '要求进入临界区' 的请求最终都能得到满足。

说明：在前面我们说到本节只考虑布尔型变量。但为了书写方便在本例中使用了有穷枚举型变量（上面的 $p1$ 和 $p2$ ），这里的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，本例中的两个进程完全可以被改换成只含布尔型变量的进程，改写过程此处略。

由定义 3.16 知：

$\text{TLF}(P_1) := (p1 = 0) \wedge (\Box((p1' = p1) \vee (p1 = 0 \wedge p1' = 1) \vee (p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2) \vee (p1 = 2 \wedge p1' = 0))) \wedge (\text{WF}(\alpha_3) \wedge \text{SF}(\alpha_2))$;

$\text{TLF}(P_2) := (p2 = 0) \wedge (\Box((p2' = p2) \vee (p2 = 0 \wedge p2' = 1) \vee (p2 = 1 \wedge (p1 = 0 \vee (p1 = 1 \wedge q)) \wedge p2' = 2) \vee (p2 = 2 \wedge p2' = 0))) \wedge (\text{WF}(\beta_3) \wedge \text{SF}(\beta_2))$

其中： $\mathrm{WF}(\alpha_3){:=}(\Box\Diamond(p1=2\wedge p1'=0))\vee(\Box\Diamond\neg(p1=2))$, $\mathrm{SF}(\alpha_2){:=}\ (\Box\Diamond(p1=1\wedge(p2=0\vee(p2=1\wedge\neg q))\wedge p1'=2))\vee(\Diamond\Box\neg(p1=1\wedge(p2=0\vee(p2=1\wedge\neg q))))$ 。 $\mathrm{WF}(\beta_3){:=}(\Box\Diamond(p2=2\wedge p2'=0))\vee(\Box\Diamond\neg(p2=2))$, $\mathrm{SF}(\beta_2){:=}\ (\Box\Diamond(p2=1\wedge(p1=0\vee(p1=1\wedge q))\wedge p2'=2))\vee(\Diamond\Box\neg(p2=1\wedge(p1=0\vee(p1=1\wedge q))))$ 。为了公式短一些，这里将 outC 、 reqC 和 inC 分别替换为 0 、 1 和 2 。

进程 1 和进程 2 的互斥性可表示为 LTLC- 公式 $\Box(\neg(p1=2\wedge p2=2))$ ；进程 1 和进程 2 对临界区的可进入性（ accessibility ）可表示为 LTLC- 公式 $\Box(p1=1\Rightarrow\Diamond(p1=2))\wedge\Box(p2=1\Rightarrow\Diamond(p2=2))$ 。要证明并发进程 $[P_1\parallel P_2]$ 满足这两个性质，只需要验证这两个公式是公式 $\mathrm{TLF}(P_1)\wedge\mathrm{TLF}(P_2)$ 的逻辑结论（ cf. 定义 3.13 ）就行，下面给出这两个性质的证明。

证明:

（ 1 ）. 互斥性. 设 $\mathcal{J}$ 是 $\mathrm{TLF}(P_1)\wedge\mathrm{TLF}(P_2)$ 的任意一个模型，且设 f_{p1} 、 f_{p2} 和 f_q 分别是变量 $p1$ 、 $p2$ 和 q 在模型 $\mathcal{J}$ 下的解释。

由 $\mathrm{TLF}(P_1)\models\Box((p1'=p1)\vee(p1=0\wedge p1'=1)\vee(p1=1\wedge p1'=2)\vee(p1=2\wedge p1'=0))$ 知 f_{p1} 只有三种不连续点，它们将 f_{p1} 的值分别由 0 、 1 和 2 改变为 1 、 2 和 0 。

同样地， f_{p2} 也只有三种不连续点，它们也是将 f_{p2} 的值分别由 0 、 1 和 2 改变为 1 、 2 和 0 。

假设 $\mathcal{J}$ 不是 $\Box(\neg(p1=2\wedge p2=2))$ 的一个模型，则存在 $a\in\mathcal{R}$ 使得 $f_{p1}(a)=2$ 且 $f_{p2}(a)=2$ 。由于 f_{p1} 在初始点的值为 0 且 f_{p1} 的不连续点都是孤立点，于是存在 $a_1\in(0,a)$ 使得 a_1 是 f_{p1} 的不连续点且 f_{p1} 在 (a_1,a) 上无不连续点。于是

$$f_{p1}(a_1)=1\text{ 且对于任意 }t\in(a_1,a]\text{ 有 }f_{p1}(t)=2. \tag{1}$$

又由于 $\mathrm{TLF}(P_1)\models\Box((p1=1\wedge p1'=2)\Rightarrow(p2=0\vee(p2=1\wedge\neg q)))$ ，于是由 $f_{p1}(a_1)=1\wedge f'_{p1}(a_1)=2$ 得：

$$f_{p2}(a_1)=0\vee(f_{p2}(a_1)=1\wedge f_q(a_1)=0) \tag{2}$$

同样地，存在 $a_2\in(0,a)$ 满足：

$$f_{p2}(a_2)=1\text{ 且对于任意 }t\in(a_2,a]\text{ 有 }f_{p2}(t)=2. \tag{3}$$

和

$$f_{p1}(a_2)=0\vee(f_{p1}(a_2)=1\wedge f_q(a_2)=1) \tag{4}$$

- 若 $a_1=a_2$ ，则由 (1)(4) 可得 $f_q(a_2)=1$, 由 (2)(3) 可得 $f_q(a_1)=0$, 矛盾。
- 若 $a_1>a_2$ ，则 $a_2\in(a_1,a]$ ，故由 (1) 得 $f_{p1}(a_2)=2$ ，这与 (4) 矛盾。

- 若 $a_1 < a_2$ ，则 $a_1 \in (a_2, a]$ ，故由 (3) 得 $f_{p2}(a_1) = 2$ ，这与 (2) 矛盾。

这样就证明了 $\mathcal{J}$ 一定是 $\Box(\neg(p1 = 2 \wedge p2 = 2))$ 的一个模型，于是 $\Box(\neg(p1 = 2 \wedge p2 = 2))$ 是 $\mathrm{TLF}(P_1) \wedge \mathrm{TLF}(P_2)$ 的逻辑结论。

（ 2 ）. 可进入性. 设 $\mathcal{J}$ 是 $\mathrm{TLF}(P_1) \wedge \mathrm{TLF}(P_2)$ 的任意一个模型。

假定 $\Box(p1 = 1 \Rightarrow \Diamond(p1 = 2))$ 在 $\mathcal{J}$ 下不成立。则必然有 $\mathcal{J} \models \Diamond\Box(p1 = 1)$, 即从某时刻起 P_1 永远停留在状态 $p1 = 1$ 中。

转换 α_2 满足强公平性，但它从某时刻起一直未能发生，于是由 $\mathcal{J} \models \mathrm{SF}(\alpha_2)$ 得 $\mathcal{J} \models \Diamond\Box\neg(p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q))))$ ，于是 $\mathcal{J} \models \Diamond\Box\neg(p2 = 0 \vee (p2 = 1 \wedge \neg q))$ ，即 $\mathcal{J} \models \Diamond\Box(p2 = 2 \vee (p2 = 1 \wedge q))$. 这样从某个时刻起 P_2 将不可能进入状态 $p2 = 0$ ，又由于 β_3 满足弱公平性，故 $p2 = 2$ 的每次出现必导致 $p2 = 0$ 在随后的出现，于是从某个时刻开始 $p2 = 2$ 也将永远不能出现，这样便有 $\mathcal{J} \models \Diamond\Box(p2 = 1 \wedge q)$ ，即从某个时刻开始 q 永远为真且 P_2 永远停留在状态 $p2 = 1$ 中。

同样地，由 $\mathcal{J} \models \Diamond\Box(p2 = 1)$ 可得到 $\mathcal{J} \models \Diamond\Box(p1 = \wedge\neg q)$ 。于是我们得到 $\mathcal{J} \models \Diamond\Box(q \wedge \neg q)$ ，这是个矛盾。于是 $\Box(p1 = 1 \Rightarrow \Diamond(p1 = 2))$ 是 $\mathcal{J}$ 的模型，同样 $\Box(p2 = 1 \Rightarrow \Diamond(p2 = 2))$ 也是 $\mathcal{J}$ 的模型。

因 $\mathcal{J}$ 是 $\mathrm{TLF}(P_1) \wedge \mathrm{TLF}(P_2)$ 的任意一个模型，故 $\Box(p1 = 1 \Rightarrow \Diamond(p1 = 2)) \wedge \Box(p2 = 1 \Rightarrow \Diamond(p2 = 2))$ 是 $\mathrm{TLF}(P_1) \wedge \mathrm{TLF}(P_2)$ 的性质。

□.

说明: 本节用 LTLC/B- 公式表示有限状态反应系统的方法与文献 [68, 77] 中基本一致，但本文的语义模型是连续时间的，而 [68, 77] 中的语义模型是离散时间的。应该来说，在讨论异步并发进程时，采用连续语义模型比较自然一些，采用离散模型则验证起来可能较容易一些。但采用连续语义模型更便于与实时和混成系统的语义模型的一致和统一。

本节只讨论了有穷状态反应系统的表示，但类似于前面的做法，利用 LTLC 中的整型变量可以表示无穷状态反应系统及其性质，也就是说， LTLC 也可以作为 (以公平转换系统为数学模型的) 反应系统的系统刻画语言和规范语言。

3.5　LTLC/B- 公式的可满足性判定

本节讨论 LTLC/B- 公式的可满足性判定, 所给的判定过程可用来算法验证有穷状态反应系统的性质, 比如可检验例 3.1 中的公式 $\Box(p1 = 1 \Rightarrow \Diamond(p1 = 2))$ 是否是进程 $[P_1 \parallel P_2]$ 的一个性质。

对于布尔型变量集 V, 本节我们以 $Form_B(V)$ 来表示 LTLC/B- 公式集 $\{\varphi \mid var(\varphi) \subseteq V$ 且 φ 是一个 LTLC/B- 公式 $\}$.

设 V 是一个布尔型变量的非空集合，令 $V' ::= \{v' \mid v \in V\}$. 称 $V \cup V'$ 上的映射 σ 为变量集 V 上的一个 (LTLC/B-) 状态, 若对任意 $p \in V$ 有 $\sigma(p) \in \{0,1\}$, $\sigma(p') \in \{0,1\}$.

设 $\mathcal{J}$ 是布尔型变量集 V 上的一个 LTLC- 模型，对于任意 $t \in \mathcal{R}^+$, 我们用 $\mathcal{J}(t)$ 来表示 V 上如下定义的一个 (LTLC/B-) 状态：对任意的 $v \in V$ ， $\mathcal{J}(t)(v) ::= \mathcal{J}(v,t)$, $\mathcal{J}(t)(v') ::= \mathcal{J}(v',t)$ 。对于 $a \in \mathcal{R}^+$, 若存在 $p \in V$ 使得 a 是 f_p 的一个不连续点，则称 a 是 $\mathcal{J}$ 的一个不连续点，这里 f_p 是 p 在 $\mathcal{J}$ 下的解释。

定义 3.21 称无穷序列 $\{t_i\}_{i\in\mathcal{N}}$ 是一个时间序列，若

1. $0 = t_0 < t_1 < t_2 < \ldots < t_n < \ldots$.

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

引理 3.1 设 $\mathcal{J}$ 是布尔型变量集 V 上的一个 *LTLC*- 模型， $\mathcal{T} = \{t_i\}_{i\in\mathcal{N}}$ 是一个时间序列且 $\mathcal{J}$ 的不连续点均在序列 $\mathcal{T}$ 中。对于如下定义的映射 $\ell_{\mathcal{T}} : \mathcal{R}^+ \mapsto \mathcal{N}$ ，

$$\ell_{\mathcal{T}}(s) = \begin{cases} 2k & \text{若存在 } k \in \mathcal{N} \text{ 使 } s = t_k \\ 2k+1 & \text{若存在 } k \in \mathcal{N} \text{ 使 } t_k < s < t_{(k+1)} \end{cases}$$

我们有 $\ell_{\mathcal{T}}(s_1) = \ell_{\mathcal{T}}(s_2)$ 蕴涵 $\mathcal{J}(s_1) = \mathcal{J}(s_2)$.

证明:

设 $\ell_{\mathcal{T}}(s_1) = \ell_{\mathcal{T}}(s_2)$ ，若 $s_1 = s_2$ ，则结论显然；若 $s_1 \neq s_2$ ，不妨设 $s_1 < s_2$, 则存在 $k \in \mathcal{N}$ 使 $t_k < s_1 < s_2 < t_{k+1}$ 。对于任意 $p \in V$, 设 p 在 $\mathcal{J}$ 下的解释为布尔值函数 f_p, 则由于 $\mathcal{J}$ 在区间 (t_k, t_{k+1}) 中无不连续点，故 f_p 在区间 (t_k, t_{k+1}) 中连续，又由于 f_p 是布尔值函数，故 $f'_p(s_1) = f_p(s_1) = f_p(s_2) = f'_p(s_2)$ 。因 p 是 V 中的任一变量，所以 $\mathcal{J}(s_1) = \mathcal{J}(s_2)$ 。

□.

设 φ 是变量集 V 上的一个 LTLC/B- 公式，令 $V' ::= \{p' \mid p \in V\}$ ，将 $V \cup V'$ 作为 LTL 中的一个命题变量集合，则 φ 同时也是 $V \cup V'$ 上的一个 LTL 公式。那么 φ 在 LTLC 中的模型和 φ 在 LTL 中的模型有没有什么联系呢？

定义 3.22 设 $\mathcal{J}$ 是布尔型变量集 V 上的一个 *LTLC*- 模型， $\mathcal{T} = \{t_i\}_{i\in\mathcal{N}}$ 是一个包含 $\mathcal{J}$ 的所有不连续点的时间序列。我们用 $\overline{\mathcal{J}_{\mathcal{T}}}$ 来表示这样的一个定义在命题变量集 $V \cup V'$ 上的 *LTL* 模型：对于任意 $k \in \mathcal{N}$, 因 $\mathcal{T}$ 是一个无穷序列，故由 $\ell_{\mathcal{T}}$ 的定义知存在 $s \in \mathcal{R}^+$ 使 $\ell_{\mathcal{T}}(s) = k$ ；对于任意 $p \in V$ 定义 $\overline{\mathcal{J}_{\mathcal{T}}}(p, k) ::= \mathcal{J}(s)(p)$, $\overline{\mathcal{J}_{\mathcal{T}}}(p', k) ::= \mathcal{J}(s)(p')$ 。

引理 3.2 对于上述定义中的 $LTLC$- 模型 $\mathcal{J}$ 及 LTL- 模型 $\overline{\mathcal{J}_\mathcal{T}}$，我们有下面的结论：对于任意 $\varphi \in Form_B(V)$ 及任意 $s \in \mathcal{R}^+$ 有，$\mathcal{J}(\varphi, s) = \overline{\mathcal{J}_\mathcal{T}}(\varphi, \ell_\mathcal{T}(s))$。

证明：对 φ 进行归纳。

1. 由于 $\overline{\mathcal{J}_\mathcal{T}}(p, \ell_\mathcal{T}(s)) = \mathcal{J}(s)(p) = \mathcal{J}(p, s)$ 且 $\overline{\mathcal{J}_\mathcal{T}}(p', \ell_\mathcal{T}(s)) = \mathcal{J}(s)(p') = \mathcal{J}(p', s)$，故当 φ 为 p 或 p' 时结论成立。

2. φ 为 $\neg\phi$ 或 $\phi_1 \wedge \phi_2$ 时的证明此处略。

3. 当 φ 为 $\Box\phi$ 时。

   - 若 $\mathcal{J}(\varphi, s) = 1$，即 $\mathcal{J}(\Box\phi, s) = 1$，则对于任意 $t \geq s$ 有 $\mathcal{J}(\phi, t) = 1$，由归纳假设便知：对于任意 $t \geq s$ 有 $\overline{\mathcal{J}_\mathcal{T}}(\phi, \ell_\mathcal{T}(t)) = 1$, 于是对任意大于或等于 $\ell_\mathcal{T}(s)$ 的整数 k 有 $\overline{\mathcal{J}_\mathcal{T}}(\phi, k) = 1$, 即有 $\overline{\mathcal{J}_\mathcal{T}}(\Box\phi, \ell_\mathcal{T}(s)) = 1$.
   - 若 $\mathcal{J}(\varphi, s) = 0$, 即 $\mathcal{J}(\Box\phi, s) = 0$, 则存在 $t \geq s$ 使得 $\mathcal{J}(\phi, t) = 0$，由归纳假设得 $\overline{\mathcal{J}_\mathcal{T}}(\phi, \ell_\mathcal{T}(t)) = 0$，由于 $\ell_\mathcal{T}(t) \geq \ell_\mathcal{T}(s)$, 于是存在大于或等于 $\ell_\mathcal{T}(s)$ 的整数 k 使得 $\overline{\mathcal{J}_\mathcal{T}}(\phi, k) = 0$，于是 $\overline{\mathcal{J}_\mathcal{T}}(\Box\phi, \ell_\mathcal{T}(s)) = 0$.

   这样就证明了当 φ 为 $\Box\phi$ 时结论成立。

4. 当 φ 为 $(\varphi_1 \mathcal{U} \varphi_2)$ 时的证明此处略。

□.

引理 3.3 对于上述定义中的 $LTLC$- 模型 $\mathcal{J}$ 及 LTL 模型 $\overline{\mathcal{J}_\mathcal{T}}$，我们有下面的结论：

1. 对于任意 $p \in V$ 及任意 $i \in \mathcal{N}$ 有：$\overline{\mathcal{J}_\mathcal{T}}(p^+, i) = \overline{\mathcal{J}_\mathcal{T}}(p', i)$。

2. 对于任意 $p \in V$ 及任意 $k \in \mathcal{N}$ 有：$\overline{\mathcal{J}_\mathcal{T}}(p', 2k+1) = \overline{\mathcal{J}_\mathcal{T}}(p, 2k+1)$

证明：

1. 对于任意 $i \in \mathcal{N}$.

   (i). 当 i 是偶数时（设 $i = 2k$），则 $\ell_\mathcal{T}(t_k) = i$, 由于 $\mathcal{J}$ 在区间 (t_k, t_{k+1}) 中连续，故对于任意 $p \in V$，f_p 在区间 (t_k, t_{k+1}) 上恒为一常数（0 或 1），这里 f_p 是 p 在 $\mathcal{J}$ 下的解释。于是对于任意 $s \in (t_k, t_{k+1})$ 有 $f'_p(t_k) = f_p(s)$，于是 $\overline{\mathcal{J}_\mathcal{T}}(p^+, i) = \overline{\mathcal{J}_\mathcal{T}}(p, i+1) = \mathcal{J}(p, s) = f_p(s) = f'_p(t_k) = \overline{\mathcal{J}_\mathcal{T}}(p', 2k) = \overline{\mathcal{J}_\mathcal{T}}(p', i)$.

(ii). 当 i 是奇数时（设 $i = 2k+1$ ），取 $s \in (t_k, t_{k+1})$ ，则 $\ell_{\mathcal{T}}(s) = 2k+1 = i$, 对于任意 $p \in V$ ，由于 f_p 在区间 (t_k, t_{k+1}) 上连续且在 t_{k+1} 处是左连续的（参看定义 3.2 ），故 f_p 在区间 $(t_k, t_{k+1}]$ 上恒为一常数（ 0 或 1 ），于是 $f_p(t_{k+1}) = f_p(s) = f'_p(s)$ 。于是 $\overline{\mathcal{J}_{\mathcal{T}}}(p^+, i) = \overline{\mathcal{J}_{\mathcal{T}}}(p, i+1) = \mathcal{J}(p, t_{k+1}) = f_p(t_{k+1}) = f'_p(s) = \overline{\mathcal{J}_{\mathcal{T}}}(p', 2k+1) = \overline{\mathcal{J}_{\mathcal{T}}}(p', i)$.

2. 取 $s \in (t_k, t_{k+1})$, 则 $\ell_{\mathcal{T}}(s) = 2k+1$, 对于任意 $p \in V$ ，由于 f_p 在区间 (t_k, t_{k+1}) 上连续，故 $f_p(s) = f'_p(s)$, 即就是有 $\mathcal{J}(p, s) = \mathcal{J}(p', s)$ ，于是 $\overline{\mathcal{J}_{\mathcal{T}}}(p', 2k+1) = \mathcal{J}(p', s) = \mathcal{J}(p, s) = \overline{\mathcal{J}_{\mathcal{T}}}(p, 2k+1)$ 。

□.

设 V 是 LTLC 中的一个布尔型变量的集合，令 $V_d ::= V \cup V' \cup \{q\}$, 其中 q 是一个不在 $V \cup V'$ 中出现的符号，我们将 V_d 作为 LTL 中的一个命题变量集合 (即 V_d 中的变量都是命题变量，取值 0 或 1) 。对于任意 $\varphi \in Form_B(V)$, 定义 V_d 上的一个 LTL 公式 $\widehat{\varphi} ::= \varphi \wedge q \wedge \Box(V^+ = V') \wedge \Box((q \Rightarrow \neg q^+) \wedge (\neg q \Rightarrow (q^+ \wedge V' = V)))$ 。这里 $V^+ = V'$ 和 $V' = V$ 分别表示 LTL 公式 $\bigwedge_{p \in V}(p^+ = p')$ 和 $\bigwedge_{p \in V}(p' = p)$ 。

定理 3.2 对于 $\varphi \in Form_B(V)$ ，若 φ 是 *LTLC*- 可满足的，则 $\widehat{\varphi}$ 是 *LTL*- 可满足的。

证明: 设 $\mathcal{J}$ 是 φ 的一个 (在变量集 V 上的)LTLC- 模型， $\mathcal{T} = \{t_i\}_{i \in \mathcal{N}}$ 是一个包含 $\mathcal{J}$ 的所有不连续点的时间序列。定义 V_d 上的一个 LTL 模型 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 如下,

(1). 对于任意 $k \in \mathcal{N}$,

$$\widehat{\mathcal{J}_{\mathcal{T}}}(q, k) ::= \begin{cases} 1 & \text{若 } k \text{ 为偶数;} \\ 0 & \text{若 } k \text{ 为奇数。} \end{cases}$$

(2). 对于任意 $k \in \mathcal{N}$ 及任意 $v \in V \cup V'$ 定义 $\widehat{\mathcal{J}_{\mathcal{T}}}(v, k) ::= \overline{\mathcal{J}_{\mathcal{T}}}(v, k)$.

下面我们证明 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 是 $\widehat{\varphi}$ 的一个模型。因 $\mathcal{J}$ 是 φ 的一个 LTLC- 模型，故由引理 3.2 知 $\overline{\mathcal{J}_{\mathcal{T}}}$ 是 φ 的一个 LTL- 模型，又 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 与 $\overline{\mathcal{J}_{\mathcal{T}}}$ 在 $V \cup V'$ 上的解释是一致的，故 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 也是 φ 的一个 LTL- 模型。

由 (1) 中 $\widehat{\mathcal{J}_{\mathcal{T}}}(q, k)$ 的定义容易看到 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 是 $q \wedge \Box((q \Rightarrow \neg q^+) \wedge (\neg q \Rightarrow q^+))$ 的模型。

再由引理 3.3 的 (1) 和 (2) 分别知 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 也是 $\Box(V^+ = V')$ 和 $\Box(\neg q \Rightarrow V' = V))$ 的模型。于是 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 是 $\widehat{\varphi}$ 的一个 (LTL) 模型。

□.

定理 3.3 对于 $\varphi \in Form_B(V)$，若 $\widehat{\varphi}$ 是 LTL- 可满足的，则 φ 是 $LTLC$- 可满足的。

证明：设 $\Im$ 是 $\widehat{\varphi}$ 的一个 LTL- 模型，由于 $\Im \models_{ltl} q \wedge \Box((q \Rightarrow \neg q^+) \wedge (\neg q \Rightarrow q^+))$, 于是对于任意 $k \in \mathcal{N}$ 有：$\Im(q, 2k) = 1, \Im(q, 2k+1) = 0.$

对于任意 $p \in V$，定义实函数 $f_p : \mathcal{R}^+ \mapsto \{0, 1\}$ 如下：

对于任意 $s \in \mathcal{R}^+$，$f_p(s) ::= \begin{cases} \Im(p, 2\lfloor s \rfloor) & \text{若 } s = \lfloor s \rfloor ; \\ \Im(p, 2\lfloor s \rfloor + 1) & \text{若 } s \neq \lfloor s \rfloor 。 \end{cases}$

这里 $\lfloor s \rfloor$ 表示实数 s 的整数部分。

由于 $\Box(V^+ = V') \wedge \Box(\neg q \Rightarrow V' = V)$ 蕴涵 $\Box(\neg q \Rightarrow V^+ = V)$, 于是对任意 $k \in \mathcal{N}$ 有 $\Im(p, 2k+2) = \Im(p^+, 2k+1) = \Im(p, 2k+1)$，这表明函数 f_p 在区间 $(k, k+1]$ 上连续。于是 f_p 是一个布尔值步函数 (定义 3.2)。

于是对任意的 $p \in V$ 我们都构造出了一个布尔值步函数 f_p。以 f_p 作为布尔型变量 p 的解释，我们便得到了 V 上的一个 LTLC- 模型 $\mathcal{J}$。取时间序列 $\mathcal{T}$ 为非负整数序列 $< 0, 1, 2, 3, \ldots, k, \ldots >$, 则 $\mathcal{T}$ 包含了 $\mathcal{J}$ 的所有不连续点。下面我们证明 $\Im = \widehat{\mathcal{J}_{\mathcal{T}}}$（注：$\widehat{\mathcal{J}_{\mathcal{T}}}$ 的定义见定理 3.2 ）。对此只需要证明对于任意 $k \in \mathcal{N}$ 及任意 $p \in V$ 有 $\Im(p, k) = \overline{\mathcal{J}_{\mathcal{T}}}(p, k)$ 及 $\Im(p', k) = \overline{\mathcal{J}_{\mathcal{T}}}(p', k)$.

(1). 当 k 为偶数时, 设 $k = 2i$, 则由定义 3.22 知 $\overline{\mathcal{J}_{\mathcal{T}}}(p, k) = \mathcal{J}(p, i) = f_p(i) = \Im(p, 2i) = \Im(p, k)$, 且 $\overline{\mathcal{J}_{\mathcal{T}}}(p', k) = \mathcal{J}(p', i) = f'_p(i) = f_p(i{+}0.5) = \Im(p, 2i{+}1) = \Im(p, k{+}1) = \Im(p', k)$.

注：$\Im(p, k+1) = \Im(p', k)$ 是因为 $\Im$ 是 $\Box(V^+ = V')$ 的模型。

(2). 当 k 为奇数时，设 $k = 2i+1$, 则由定义 3.22 知 $\overline{\mathcal{J}_{\mathcal{T}}}(p, k) = \mathcal{J}(p, i+0.5) = f_p(i+0.5) = \Im(p, 2i+1) = \Im(p, k)$, 且 $\overline{\mathcal{J}_{\mathcal{T}}}(p', k) = \mathcal{J}(p', i+0.5) = f'_p(i+0.5) = f_p(i+0.5) = \Im(p, 2i+1) = \Im(p, k) = \Im(p', k)$ 。

注：$\Im(p, k) = \Im(p', k)$ 是因为 $\Im$ 是 $\Box(\neg q \Rightarrow V' = V)$ 的模型，而且当 k 为奇数时 $\Im(q, k) = 0$, 于是 $\Im(p', k) = \Im(p, k)$ 。

这样我们便证明了 $\Im = \widehat{\mathcal{J}_{\mathcal{T}}}$ 。

由于 $\Im$ 是 φ 的一个 LTL- 模型，故 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 也是 φ 的一个 LTL- 模型，又 $\widehat{\mathcal{J}_{\mathcal{T}}}$ 与 $\overline{\mathcal{J}_{\mathcal{T}}}$ 在 $V \cup V'$ 上一致，故 $\overline{\mathcal{J}_{\mathcal{T}}}$ 也是 φ 的一个 LTL- 模型。即 $\overline{\mathcal{J}_{\mathcal{T}}}(\varphi, 0) = 1$，应用引理 3.2 便得到 $\mathcal{J}(\varphi, 0) = 1$，于是 $\mathcal{J}$ 是 φ 的一个 LTLC- 模型，这样就证明了 φ 是 LTLC- 可满足的。

□.

至此我们就证明了 φ 是 LTLC- 可满足的当且仅当 $\widehat{\varphi}$ 是 LTL- 可满足的，而 $\widehat{\varphi}$ 的（ LTL- ）可满足性是可判定的 (见定理 2.1)，这样 LTLC/B 公式的可满足性是可判定的。即有：

定理 3.4 对于 $\varphi \in Form_B(V)$， φ 的 *(LTLC-)* 可满足性是可判定的。

说明：利用定理 3.2 和定理 3.3 和文献 [62] 中的工具，可以较容易地实现一个验证 LTLC/B 公式的可满足性的检验工具。

3.6 小结

本章给出了 LTLC 的语法和语义，定义了 LTLC 的一个子语言 LTLC/B，并证明了 LTLC/B- 公式的可满足性是可判定的。本章还讨论了如何利用 LTLC/B 来表示有穷状态反应系统的问题。由于 LTLC/B 可表示有穷状态反应系统以及它们的性质，故利用 LTLC/B 的可满足性判定过程可以对有限状态反应系统进行模型检查和‘求精’关系检查。

第四章　　实时系统

4.1　基本概念

实时系统 (real-time system)[17, 22, 56, 78] 是一种带有时间约束的计算系统，核反应堆、飞行控制以及铁路调度等方面的许多计算机控制系统都属于实时系统。这些系统的许多动作的完成是与时间相关的，即要满足一定的时间限制，如某动作要在一秒钟内完成等. 对这类系统，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。

下面是一个很简单的实时系统的例子。

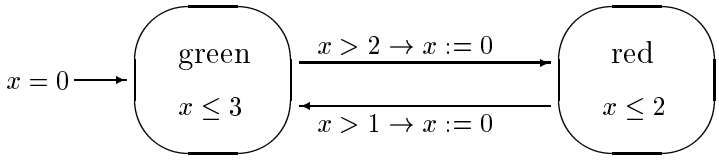

图 4.1 Traffic-light

例 4.1. 图 4.1 中的时间自动机表示的实时系统模拟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 个时间单元，在 '绿灯亮' 持续 2 个时间单元后可转为 '红灯亮' ； '红灯亮' 的持续时间不能超过 2 个时间单元，在 '红灯亮' 持续 1 个时间单元后可转为 '绿灯亮' 。

本章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) (上例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，如上例中的 $x \leq 3$ 和 $x \leq 2$) 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重置其值，表示时间约束条件的时钟变量的值既可维持不变，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，它的持续时间为 0 。

在本章中我们用 LTLC 的一个子语言 LTLC/R 来表示实时系统。 LTLC/R 中只包含两类变量：布尔型变量和时钟变量. 时钟变量用来表示系统中的时间约束条件，布尔型变量用来表示系统的控制状态（即所处的顶点，如可用 $p = 1$ 和 $p = 0$ 来分别表示例 4.1 中的控制状态 green 和 red).

一个 LTLC/R- 公式是指由下面规则生成的一个 LTLC 公式 φ:

$\varphi ::= p \,|\, p' \,|\, (c \leq m) \,|\, (c = m) \,|\, (c' = c) \,|\, (c' = 0) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\Box\varphi) \,|\, (\varphi_1 \mathcal{U} \varphi_2).$

其中 p 是一个布尔型变量， c 是一个时钟变量， m 是一个非负整数（常元符号）。

LTLC/R- 公式的语义模型的定义与 LTLC 公式的相同，只是去掉了在 LTLC/R 中不出现的成分的解释。为了叙述方便起见，下面定义几种特殊形式的 LTLC/R- 公式。

一个 LTLC/R- 状态公式是指由如下规则形成的一个 LTLC/R- 公式 φ ：

$\varphi ::= p \,|\, p' \,|\, (c \leq m) \,|\, (c = m) \,|\, (c' = c) \,|\, (c' = 0) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

LTLC/R- 当前状态公式是指由如下规则形成的 LTLC/R- 公式 φ ：

$\varphi ::= p \,|\, (c \leq m) \,|\, (c = m) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

顶点 (location) 公式是指由如下规则形成的 LTLC/R- 公式 φ ：

$\varphi ::= p \,|\, (\neg p) \,|\, (\varphi_1 \wedge \varphi_2)$

新顶点 (newlocation) 公式是指由如下规则形成的 LTLC/R- 公式 φ ：

$\varphi ::= p' \,|\, (\neg p') \,|\, (\varphi_1 \wedge \varphi_2)$

在以上的规则中 p 是一个布尔型变量， c 是一个时钟变量， m 是一个非负整数（常元符号）。

4.2 时间模块及其验证

我们以 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 (这里的变量可以是布尔型变量也可以是时钟变量) 。其中某些变量的值只能被模块自身所改变，不能被环境改变；而另一些变量的值只能被环境所改变，不能被模块改变。依此我们将每个模块的变量分为二类，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。

每个时间模块主要由有限个转换（ transition ）所组成。控制变量的值是通过转换的作用来改变的。每个时间模块有两种类型的转换：跳跃转换和延迟转换。跳跃转换的发生不占用时间段，在其作用的瞬间 (零时段内) 它可改变所有控制变量 (无论布尔型变量或时钟变量) 的值。但每个延迟转换的发生具有一个正的时间段，在其作用的这个时段内，布尔型变量的值保持不变，而时钟变量的值随时间流逝而同步增长，其增加量为所流逝的时间。在任意一个有限时间段中，跳跃转换只能发生有限次，相邻发生的两次跳跃转换之间为一个延迟转换的作用时段。

跳跃转换通常被写成形如 '$vertex \wedge guard \to new\_vertex \wedge assignment$' 的卫式命令的形式。这里 $vertex$ 是一个顶点公式, 用于表示跳跃转换发生时系统所处的顶点；$guard$ 是一个 LTLC/R- 状态公式, 它是跳跃转换的使能条件（ enabling condition ）, 它用于表示跳跃转换发生时系统的变量(包括控制变量以及外部变量)应满足的约束条件；new_vertex 是一个新顶点公式, 它用于表示跳跃转换发生后系统将位于的新顶点；$assignment$ 是跳跃转换的命令部分, 它相当于赋值语句, 它使模块中每个控制时钟变量 c 要么重置为 0(即 $c' = 0$), 要么维持原值不变 (即 $c' = c$), 它通常可表示为形如 '$c'_1 = e_1 \wedge c'_2 = e_2 \wedge \cdots \wedge c'_k = e_k$ ' 的一个 LTLC/R- 状态公式，这里 $c_1, c_2, \ldots, c_k$ 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 c_i 。为叙述方便起见，我们假定每个跳跃转换在其作用时，至少改变了一个控制变量 (无论布尔型或时钟型) 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下面的形式：'$vertex \to invariant$'。其中 '$vertex$' 是一个顶点公式, 它表示延迟转换作用时所处的顶点。$invariant$ 是延迟转换的不变量（ time invariant ）部分，它是一个 LTLC/R- 当前状态公式，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	$\{variable\_name\colon type\}^*$
controlled	$\{variable\_name\colon type\}^*$
init	$init\_cond$
jump	$\{vertex \wedge guard \to new\_vextex \wedge assignment\}^*$
delay	$\{vertex \to invariant\}^*$

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，它是一个 LTLC/R- 当前状态公式，用于表示模块在初始时刻 $t = 0$ 时其控制变量所应满足的条件。

注：时间模块中的不同的延迟转换所处的顶点应是互斥的, 即若 $vertex_1 \to invariant_1$ 和 $vertex_2 \to invariant_2$ 是某一个时间模块的两个不同延迟转换, 则 $vertex_1 \wedge vertex_2$ 是一个永假的公式。另外通常还要求延迟转换是完全的，即若 $\{vertex_k \to invariant_k\}_{k<\ell}$

是一个时间模块的所有延迟转换，则 $\bigvee_{k<\ell} vertex_k$ 应包括了时间模块可能位于的所有顶点。

说明：时间模块中的变量的类型有布尔型和时钟类型共两种。但为了书写方便我们还可使用有穷枚举型变量（如例 4.2 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被替换为不含枚举型变量的时间模块。此外为了方便起见，我们常把 $p=1, p=0, p'=1, p'=0, p'=p$ 分别作为公式 p, $\neg p$, p', $\neg p'$, $(p \wedge p') \vee (\neg p \wedge \neg p')$ 的另一种表示方式。若 lb 是一个取值域为 $\{0,1,2,3,4,5\}$ 的有穷枚举型变量，则 $lb=3, lb'=5$ 可分别看作为公式 $lb_0=1 \wedge lb_1=1 \wedge lb_2=0$ 和 $lb'_0=1 \wedge lb'_1=0 \wedge lb'_2=1$ 的缩写形式（其中 lb_0, lb_1, lb_2 是三个布尔型变量）。

例 4.1(续) 图 4.1 中的时间自动机可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 : {green, red};
	x : *clock*
init	$p = \text{green} \wedge x = 0$
jump	$p = \text{green} \wedge x > 2 \rightarrow p' = \text{red} \wedge x' = 0$;
	$p = \text{red} \wedge x > 1 \rightarrow p' = \text{green} \wedge x' = 0$
delay	$p = \text{green} \rightarrow x \leq 3$;
	$p = \text{red} \rightarrow x \leq 2$

注：上面转换中出现的符号 $\rightarrow$ 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，下面就给出时间模块对应的 LTLC 公式，并利用 LTLC 给出时间模块的语义。

对于跳跃转换 α : $vertex \wedge guard \rightarrow new\_vertex \wedge assignment$, 称 LTLC- 公式 $vertex \wedge guard \wedge new\_vertex \wedge assignment$ 为转换 α 所对应的时序逻辑公式，记作 $\text{TLF}(\alpha)$ 。对于延迟转换 β : $vertex \rightarrow invariant$, 称 LTLC- 公式 $vertex \wedge invariant$ 为延迟转换 β 所对应的时序逻辑公式，记作 $\text{TLF}(\beta)$ 。

定义 4.1 设 M 是一个时间模块， $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ 是 M 的所有跳跃转换，$\beta_0, \beta_1, \ldots, \beta_{l-1}$ 是 M 的所有延迟转换。称 *LTLC*- 公式 $init\_cond \wedge (\Box(V_c = V'_c \vee \bigvee_{j<n} TLF(\alpha_j))) \wedge \Box \bigvee_{k<l} TLF(\beta_k)$ 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 $V_c = ctr(M)$*)*， $V_c = V'_c$ 用作为时序公式 $\bigwedge_{v \in V_c}(v' = v)$ 的缩写形式。以下我们以 *TLF(M)* 来表示这个对应于模块 M 的时序逻辑公式。

例 4.1(续) 时间模块 Traffic-light 所对应的时序逻辑公式 TLF(Traffic-light)= $((p = 0 \wedge x = 0) \wedge \Box((p = 0 \wedge x > 2 \wedge p' = 1 \wedge x' = 0) \vee (p = 1 \wedge x > 1 \wedge p' = 0 \wedge x' = 0) \vee (p' = p \wedge x' = x)) \wedge \Box((p = 0 \wedge x \leq 3) \vee (p = 1 \wedge x \leq 2)))$.

我们将时间模块 M 等同于它所对应的时序逻辑公式 TLF(M), 并将 TLF(M) 的语义模型作为 M 的语义模型。

定义 4.2 设 M 是一个时间模块， φ 是一个 *LTLC*- 公式。如果 *TLF(M)*$\models \varphi$, 则称 φ 是 M 的一个性质，记作 $M \models \varphi$ 。

说明：定义 4.2 中的公式 φ 可以不局限于在 LTLC/R 中，它可以包含全局时钟 t 以及刚性变量等，但在本章的大多数时候，我们所考虑的性质仅限于它在 LTLC/R 中。

由上面这个定义和定义 3.13 可知，若想证明模块 M 具有性质 φ, 只需证明 φ 是 TLF(M) 的逻辑结论.

定义 4.3 设 M_1 和 M_2 是两个时间模块，且 $extl(M_1) = extl(M_2)$ ， $ctr(M_1) \subseteq ctr(M_2)$ 。如果 $TLF(M_2) \models TLF(M_1)$, 则称 M_2 是 M_1 的一个 '精化'(*refinement*)。

定义 4.4 称模块 M_1 和 M_2 是相容的，若 $ctr(M_1) \cap ctr(M_2) = \emptyset$ 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 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 来表示这 n 个模块的并行复合。并称公式 $\mathrm{TLF}(M_1) \wedge \mathrm{TLF}(M_2) \wedge \cdots \wedge \mathrm{TLF}(M_n)$ 为复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 所对应的时序逻辑公式。时序公式 $\mathrm{TLF}(M_1) \wedge \mathrm{TLF}(M_2) \wedge \cdots \wedge \mathrm{TLF}(M_n)$ 的语义模型被看作为是复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 的语义模型。

定义 4.5 对于公式 φ ，若 $TLF(M_1) \wedge TLF(M_2) \wedge \cdots \wedge TLF(M_n) \models \varphi$ ，则称 φ 是复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 的一个性质。

下面我们通过例子来说明如何利用定义 4.5(或定义 4.2) 来证明实时系统的性质.

例 4.2.[Railroad gate control] 图 4.2 中的时间自动机模拟一个环形铁路上的火车，该段铁路上有一个道口，图 4.3 中的时间自动机模拟这样的一个道口控制系统。这段铁路被分成三个区段： far, near 和 passing. 初始时，火车在 far 区段，当火车进入 near 区段时，它便发出进站信号 'in', 在 near 区段火车运行 3 到 5 分钟便进入 passing 区段，在 passing 区段火车运行 2 到 3 分钟将进入 far 区段，在进入 far 区段时火车将发出出站信

号 'out'. 火车每次在 far 区段中至少需要运行 3 分钟后才有可能进入 near 区段, 依此反复. 道口起初是打开的 (open 状态), 并每隔一分钟检查一次看是否有火车进站的信号, 若无则继续等待, 若有则开始下降道口上面的栏杆 (这一阶段道口处于 down 状态), 两分钟后道口彻底关闭, 这时道口处于关闭 (closed) 状态, 然后道口便开始每隔一分钟检查一次看火车是否仍然在车站内, 若是则继续等待, 否则便开始升起栏杆, 两分钟之后恢复到 open 状态并开始等待下一个 'in' 信号。图 4.2 和图 4.3 中的时钟变量 x 和 y 用来表示时间约束, 布尔型变量 sg 用于记录信号 'in' 和 'out' , sg=in 表示火车在站内, sg=out 表示火车在站外。

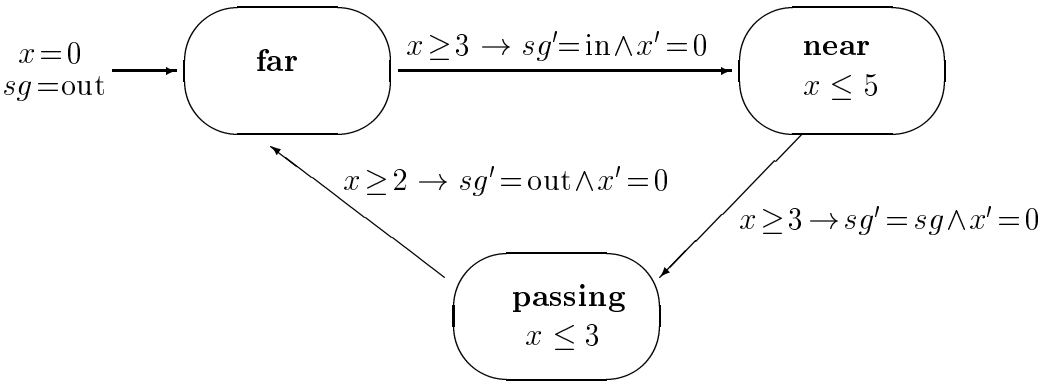

图 4.2: Train

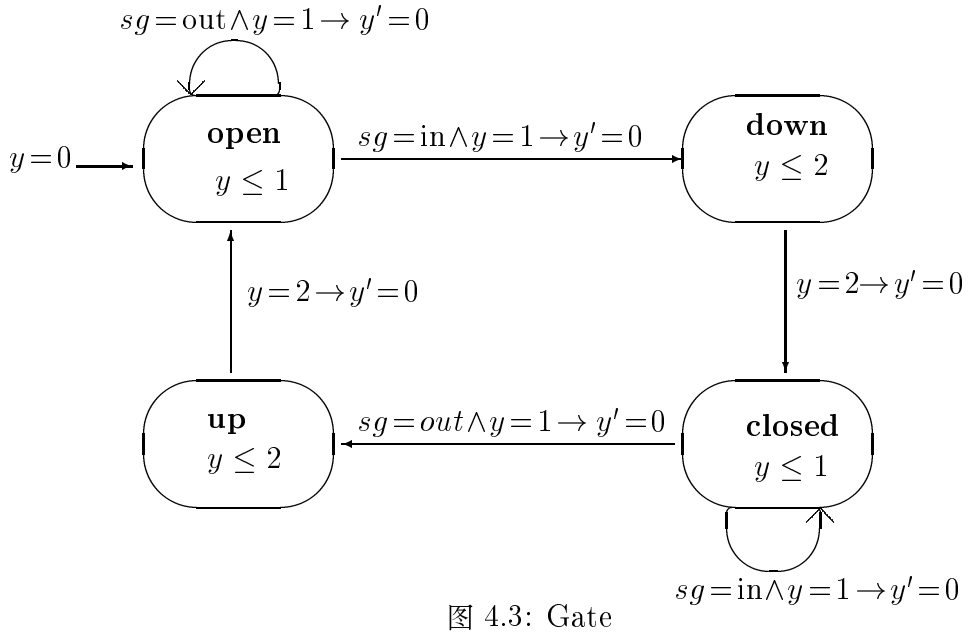

图 4.3: Gate

以上两个用时间自动机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　　$p = \text{far} \wedge sg = \text{out} \wedge x = 0$
jump
　$p = \text{far} \wedge x \geq 3 \rightarrow (p' = \text{near} \wedge sg' = \text{in} \wedge x' = 0)$;
　$p = \text{near} \wedge x \geq 3 \rightarrow (p' = \text{passing} \wedge sg' = sg \wedge x' = 0)$;
　$p = \text{passing} \wedge x \geq 2 \rightarrow (p' = \text{far} \wedge sg' = \text{out} \wedge x' = 0)$
delay
　$p = \text{far} \rightarrow true$;
　$p = \text{near} \rightarrow x \leq 5$;
　$p = \text{passing} \rightarrow x \leq 3$

module　　$Gate$
external　　sg : {in, out}
controlled
　q : {open,closed, up,down};
　y : $clock$
init　　$q = \text{open} \wedge y = 0$
jump
　$q = \text{open} \wedge sg = \text{out} \wedge y = 1 \rightarrow (q' = q \wedge y' = 0)$;
　$q = \text{open} \wedge sg = \text{in} \wedge y = 1 \rightarrow (q' = \text{down} \wedge y' = 0)$;
　$q = \text{down} \wedge y = 2 \rightarrow (q' = \text{closed} \wedge y' = 0)$;
　$q = \text{closed} \wedge sg = \text{in} \wedge y = 1 \rightarrow (q' = q \wedge y' = 0)$;
　$q = \text{closed} \wedge sg = \text{out} \wedge y = 1 \rightarrow (q' = \text{up} \wedge y' = 0)$;
　$q = \text{up} \wedge y = 2 \rightarrow (q' = \text{open} \wedge y' = 0)$
delay
　$q = \text{open} \rightarrow y \leq 1$;
　$q = \text{down} \rightarrow y \leq 2$;
　$q = \text{closed} \rightarrow y \leq 1$;
　$q = \text{up} \rightarrow y \leq 2$

由定义 4.1 知：

TLF(Train)=$(p=0\wedge sg=0\wedge x=0)\wedge\Box((p=0\wedge x\geq 3\wedge p'=1\wedge sg'=1\wedge x'=0)\vee(p=1\wedge x\geq 3\wedge p'=2\wedge sg'=sg\wedge x'=0)\vee(p=2\wedge x\geq 2\wedge p'=0\wedge sg'=0\wedge x'=0)\vee(p'=p\wedge sg'=sg\wedge x'=x))\wedge\Box(p=0\vee(p=1\wedge x\leq 5)\vee(p=2\wedge x\leq 3))$.

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

下面我们证明公式 $\Box(p=2\Rightarrow q=2)$(即： $\Box(p=\text{passing}\Rightarrow q=\text{closed})$) 是复合模块 [Train‖Gate] 的一个性质，这是一个安全性公式，它表示当火车处在 passing 状态时，道口一定是关闭的，类似地我们也可证明 $\forall u.(\Box((q=2\wedge t=u)\Rightarrow\Diamond(q=0\wedge t\leq u+9)))$ 是复合模块 [Train‖Gate] 的一个性质，它表示若一个要过道口的人发现道口是关闭的，那么不超过 9 分钟他就可以等到道口被打开。

设 $\mathcal{J}$ 是 TLF(Train) $\wedge$ TLF(Gate) 的一个模型，设 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 $\mathcal{J}$ 下的解释。

设 $a_1, a_2, \ldots, a_n, \ldots$ 是函数 f_p 的所有不连续点组成的序列且满足 $a_1<a_2<a_3<\cdots<a_n<\ldots$ (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)。

由于 TLF(Train)$\models p=0\wedge\Box((p=0\wedge p'=1)\vee(p=1\wedge p'=2)\vee(p=2\wedge p'=0)\vee(p'=p))\wedge\Box(p=0\vee p=1\vee p=2)$ ，故 f_p 的值域为 $\{0, 1, 2\}$ 且 f_p 的不连续点只可能有三种，它们分别将 f_p 的值从 0, 1, 2 改变为 1, 2, 0. 取 $a_0=0$ ，由于 f_p 的值由 0 变为 1 需要条件 $x\geq 3$ ，故 f_p 的第一个不连续点 a_1 不能是 0 ，于是 $a_1>a_0$ ，这样对任意 $i\in\mathcal{N}$ 就有：

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

由于 $f_x'(a_{3i+1})=0$, $3\leq f_x(a_{3i+2})\leq 5$ 且时钟函数 f_x 在 $(a_{3i+1}, a_{3i+2}]$ 连续，故 $3\leq a_{3i+2}-a_{3i+1}\leq 5$; 类似地有 $2\leq a_{3i+3}-a_{3i+2}\leq 3$.

由 TLF(Train)$\models sg=0\wedge\Box((p=0\wedge p'=1\wedge sg'=1)\vee(p=2\wedge p'=0\wedge sg'=0)\vee(p'=p\wedge sg'=sg))$ 知 f_{sg} 的不连续点出现在 f_p 的值由 0 变为 1 和由 2 变为 0 处，于是有：

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

设 $b_1,\,b_2,\,\ldots,b_n,\,\ldots$ 是函数 f_q 的所有不连续点组成的序列且满足 $b_1<b_2<b_3<\cdots<b_n<\ldots$ 。令 $b_0::=0$, 由 TLF(Gate) $\models q=0\wedge(\Box(q=0\wedge sg=1\wedge q'=1)\vee(q=1\wedge q'=2)\vee(q=2\wedge q'=3)\vee(q=3\wedge q'=0)\vee q'=q)\wedge\Box(q=0\vee q=1\vee q=2\vee q=3))$ 知 $b_1>b_0$ ，且对任意 $i\in\mathcal{N}$ 有：

$$f_q(t)=\begin{cases}0 & \text{若 } t\in(b_{4i},\,b_{4i+1}],\\ 1 & \text{若 } t\in(b_{4i+1},b_{4i+2}],\\ 2 & \text{若 } t\in(b_{4i+2},b_{4i+3}],\\ 3 & \text{若 } t\in(b_{4i+3},b_{4i+4}].\end{cases}\qquad(3)$$

由时钟函数 f_y 在区间 $[b_{4i+1},b_{4i+2}]$ 和 $[b_{4i+3},b_{4i+4}]$ 上的值的变化可得 $b_{4i+2}=b_{4i+1}+2$ 且 $b_{4i+4}=b_{4i+3}+2$ 。

下面我们归纳证明对序列 $\{a_i\}_{i\in\mathcal{N}}$ 和 $\{b_i\}_{i\in\mathcal{N}}$ 有如下关系成立。

$$b_{4i+1}-1\le a_{3i+1}<b_{4i+1}.\qquad(4)$$

步 1：当 $i=0$ 时, 由于 y 在 $(0,b_1]$ 上始终满足 $(y\le 1)\wedge((y=1\wedge y'=0)\vee(y'=y))$, 故 y 的值每隔一分钟置 0 一次，在点 b_1 前的每次置 0 因不改变 q 的值故只能使用跳跃转换 $q=0\wedge sg=0\wedge y=1\wedge q'=q\wedge y'=0$ ，于是在点 b_1-1 和它之前必有 $sg=0$, 在点 b_1 处因改变了 q 的值故只能使用跳跃转换 $q=0\wedge sg=1\wedge y=1\wedge q'=1\wedge y'=0$ 于是点 a_1 只能落入区间 $[b_1-1,b_1)$ 中，于是有 $b_1-1\le a_1<b_1$.

步 2 ：假设当 $i=k$ 时 (4) 式成立，即 $b_{4k+1}-1\le a_{3k+1}<b_{4k+1}$.

下面看 $i=k+1$ 的情况.

由于 $b_{4k+2}>b_{4k+1}>a_{3k+1}$ 且 $b_{4k+2}=b_{4k+1}+2\le a_{3k+1}+3\le a_{3k+2}<a_{3k+3}$, 于是 $b_{4k+2}\in(a_{3k+1},a_{3k+3})$. 由 (2) 可知 sg 在点 b_{4k+2} 的值为 '1' ，而 sg 在点 b_{4k+3} 的值为 '0' ，类似于步 1 中的情况，时钟变量 y 的值在区间 $(b_{4k+2},b_{4k+3}]$ 中每隔一分钟便置 0 一次，根据置 0 时所使用的跳跃转换的条件我们可得到 $b_{4k+3}-1\le a_{3k+3}<b_{4k+3}$.

再由于 $a_{3k+4}\ge a_{3k+3}+3\ge b_{4k+3}+2=b_{4k+4}$ ，故 sg 在点 b_{4k+4} 处的值为 '0' ，但 sg 在点 b_{4k+5} 处的值为 '1' ，且 y 的值在区间 $(b_{4k+4},b_{4k+5}]$ 中每隔一分钟置 0 一次，再次类似于步 1 的证明知 $b_{4k+5}-1\le a_{3k+4}<b_{4k+5}$ 。于是当 $i=k+1$ 时 (4) 式仍然成立。

这样便证明了 (4) 式对任意的 $i \in \mathcal{N}$ 成立。

从上面的证明可得对于任意 $k \in \mathcal{N}$ 有 $b_{4k+2} \leq a_{3k+2} < a_{3k+3} < b_{4k+3}$，于是 $(a_{3k+2}, a_{3k+2}] \subset (b_{4k+2}, b_{4k+3}]$，再由 (1) 和 (3) 中 f_p 和 f_q 的值在各个区间上的分布情况可得 $\mathcal{J}$ 是 $\Box(p = 2 \Rightarrow q = 2)$ 的一个模型。由于 $\mathcal{J}$ 可以是 TLF(Train) $\wedge$ TLF(Gate) 的任意一个模型，故由定义 4.5 知 $\Box(p = 2 \Rightarrow q = 2)$ 是复合模块 [Train‖Gate] 的一个性质。

下面是刚才证明过程中的一个示意图。

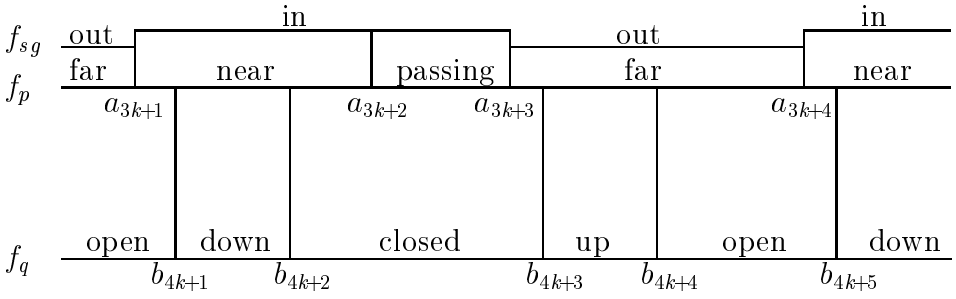

图 4.4: 例 4.2 证明的示意图

例 4.3(互斥问题， mutual exclusion) 图 4.5 和 4.6 表示了两个共享同一资源的并发进程，每个进程有三种控制状态： outC 、 reqC 和 inC，分别表示进程处在临界区 (critical section) 外、正在申请进入临界区和正处在临界区内。初始时，两个进程都处在状态 outC，在此至少停留一个时间单位后便可进入状态 reqC ，在状态 reqC 中，进程每隔一个时间单位便检查一次其进入临界区的条件是否成立，若成立则进入临界区，否则继续等待。进程 1 进入临界区的条件是 '进程 2 此时在状态 outC 中，或者进程 2 此时虽在状态 reqC 中但 q 的值为 0' ，进程 2 进入临界区的条件是 '进程 1 此时在状态 outC 中，或者进程 1 此时虽在状态 reqC 中但 q 的值为 1' ，这里 q 是一个来自两进程之外的调控信号，取值 0 或 1 。每个进程在临界区中停留 3 到 5 个时间单位后便回到 outC 状态，此后可再次重复上面的过程。图中的 x 和 y 是时钟变量，用来表示时间约束条件； $p1$ 和 $p2$ 是有穷枚举型变量，取值 0 、 1 和 2 ，分别表示进程在 outC 状态、 reqC 状态和 inC 状态。

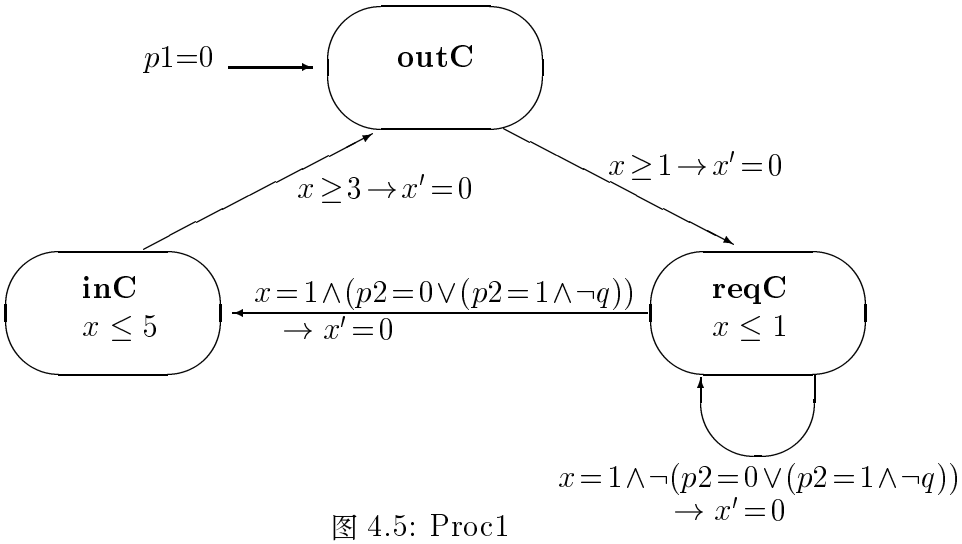

图 4.5: Proc1

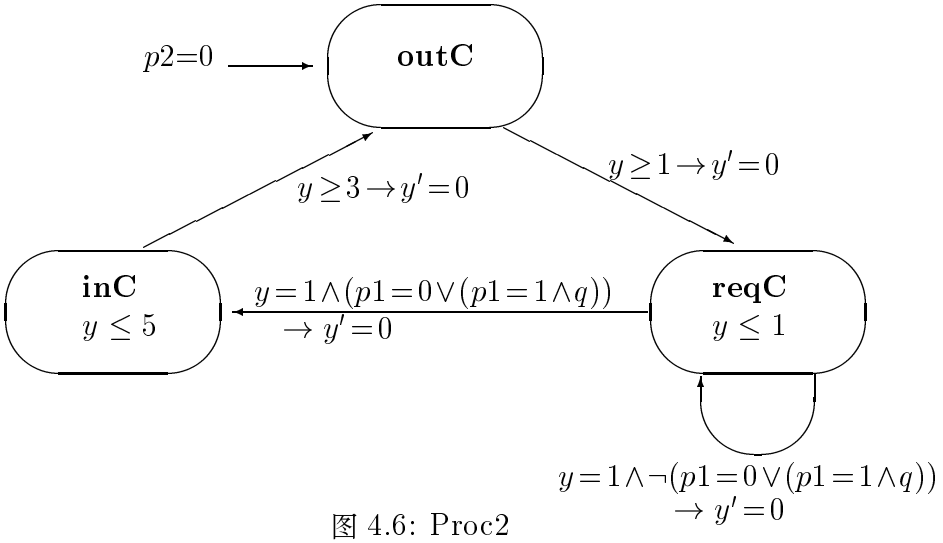

图 4.6: Proc2

以上两个进程可用时间模块表示如下：

module $Proc1$
external
$q : boolean$;
$p2 : \{0, 1, 2\}$
controlled
$p1 : \{0, 1, 2\}$;
$x : clock$
init $p1 = 0 \wedge x = 0$
jump
$p1 = 0 \wedge x \geq 1 \rightarrow (p1' = 1 \wedge x' = 0)$;
$p1 = 1 \wedge x = 1 \wedge \neg(p2 = 0 \vee (p2 = 1 \wedge \neg q)) \rightarrow (p1' = p1 \wedge x' = 0)$;
$p1 = 1 \wedge x = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \rightarrow (p1' = 2 \wedge x' = 0)$;
$p1 = 2 \wedge x \geq 3 \rightarrow (p1' = 0 \wedge x' = 0)$
delay
$p1 = 0 \rightarrow true$;
$p1 = 1 \rightarrow x \leq 1$;
$p1 = 2 \rightarrow x \leq 5$

module $Proc2$
external
$q : boolean$;
$p1 : \{0, 1, 2\}$
controlled
$p2 : \{0, 1, 2\}$;
$y : clock$
init $p2 = 0 \wedge y = 0$
jump
$p2 = 0 \wedge y \geq 1 \rightarrow (p2' = 1 \wedge y' = 0)$;
$p2 = 1 \wedge y = 1 \wedge \neg(p1 = 0 \vee (p1 = 1 \wedge q)) \rightarrow (p2' = p2 \wedge y' = 0)$;
$p2 = 1 \wedge y = 1 \wedge (p1 = 0 \vee (p1 = 1 \wedge q)) \rightarrow (p2' = 2 \wedge y' = 0)$;
$p2 = 2 \wedge y \geq 3 \rightarrow (p2' = 0 \wedge y' = 0)$
delay
$p2 = 0 \rightarrow true$;
$p2 = 1 \rightarrow y \leq 1$;
$p2 = 2 \rightarrow y \leq 5$

由定义 4.1 知：

$\mathrm{TLF}(Proc1) := p1 = 0 \wedge x = 0 \wedge (\Box((p1' = p1 \wedge x' = x) \vee (p1 = 0 \wedge x \geq 1 \wedge p1' = 1 \wedge x' = 0) \vee (p1 = 1 \wedge x = 1 \wedge \neg(p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = p1 \wedge x' = 0) \vee (p1 = 1 \wedge x = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2 \wedge x' = 0) \vee (p1 = 2 \wedge x \geq 3 \wedge p1' = 0 \wedge x' = 0))) \wedge (\Box((p1 = 0) \vee (p1 = 1 \wedge x \leq 1) \vee (p1 = 2 \wedge x \leq 5)));$

$\mathrm{TLF}(Proc2) := p2 = 0 \wedge y = 0 \wedge (\Box((p2' = p2 \wedge y' = y) \vee (p2 = 0 \wedge y \geq 1 \wedge p2' = 1 \wedge y' = 0) \vee (p2 = 1 \wedge y = 1 \wedge \neg(p1 = 0 \vee (p1 = 1 \wedge q)) \wedge p2' = p2 \wedge y' = 0) \vee (p2 = 1 \wedge y = 1 \wedge (p1 = 0 \vee (p1 = 1 \wedge q)) \wedge p2' = 2 \wedge y' = 0) \vee (p2 = 2 \wedge y \geq 3 \wedge p2' = 0 \wedge y' = 0))) \wedge (\Box((p2 = 0) \vee (p2 = 1 \wedge y \leq 1) \vee (p2 = 2 \wedge y \leq 5))).$

这里我们关心的是 [Proc1‖Proc2] 是否是例 3.1 中的 $[P_1 \parallel P_2]$ 的一个求精，即是否有 $\mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)} \models \mathrm{TLF}(P_1) \wedge \mathrm{TLF}(P_2)$，可惜此结论并不成立，我们可构造出一个 LTLC- 模型，它是前者的模型而不是后者的模型。但如果对外部变量 q 的变化作一定的限制，如限制 q 的每个状态（$p = 0$ 或 $p = 1$）的连续持续时间每次不少于 1 个时间单位（这个限制可表示为一个 LTLC/R- 公式 $\Box((q' = q \wedge z' = z) \vee (z \geq 1 \wedge z' = 0))$，其中 z 是一个时钟变量，下面我们以 H 记这个时序逻辑公式），则可证 [Proc1‖Proc2] 是 $[P_1 \parallel P_2]$ 的一个求精，即我们要证：

$$H \wedge \mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)} \models \mathrm{TLF}(P_2) \wedge \mathrm{TLF}(P_2). \qquad (1)$$

证明：

(i). 由 $(p1 = 1 \wedge x = 1 \wedge \neg(p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = p1 \wedge x' = 0)$ 蕴涵 $(p1' = p1)$，以及 $(p1 = 1 \wedge x = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2 \wedge x' = 0)$ 蕴涵 $(p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2)$ 等可得

$$\mathrm{TLF(Proc1)} \models (p1 = 0) \wedge (\Box((p1' = p1) \vee (p1 = 0 \wedge p1' = 1) \vee (p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2) \vee (p1 = 2 \wedge p1' = 0))). \qquad (2)$$

同样可得：

$$\mathrm{TLF(Proc2)} \models (p2 = 0) \wedge (\Box((p2' = p2) \vee (p2 = 0 \wedge p2' = 1) \vee (p2 = 1 \wedge (p1 = 0 \vee (p1 = 1 \wedge q)) \wedge p2' = 2) \vee (p2 = 2 \wedge p2' = 0))). \qquad (3)$$

于是，要证 (1) 只需证下面的 (4)(5) 即可。

$$H \wedge \mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)} \models \mathrm{WF}(\alpha_3) \wedge \mathrm{WF}(\beta_3). \qquad (4)$$

$$H \wedge \mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)} \models \mathrm{SF}(\alpha_2) \wedge \mathrm{SF}(\beta_2). \qquad (5)$$

(ii). 假设 (4) 不成立，不妨设 $H \wedge \mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)} \models \mathrm{WF}(\alpha_3)$ 不成立。则存在 $H \wedge \mathrm{TLF(Proc1)} \wedge \mathrm{TLF(Proc2)}$ 的模型 $\mathcal{J}$ 使得 $\mathcal{J} \models \neg\mathrm{WF}(\alpha_3)$，即 $\mathcal{J} \models \neg((\Box\Diamond(p1 =$

$2 \wedge p1' = 0)) \vee (\Box\Diamond\neg(p1 = 2)))$, 于是有 $\mathcal{J} \models \Diamond\Box(p1 = 2)$ 。

设 f_{p1} 是 $p1$ 在 $\mathcal{J}$ 下的解释, 则存在一个实数 $a \in \mathcal{R}^+$ 使得布尔值步函数 f_{p1} 在区间 $[a, \infty)$ 上恒为 2 , 于是在 $[a, \infty)$ 上恒有 $p1 = 2 \wedge p1' = 2$, 而 $p1 = 2 \wedge p1' = 2$ 与 $(p1 = 0) \vee (p1 = 1) \vee (p1 = 2 \wedge p1' = 0))$ 不相容, 于是 $(p1 = 0) \vee (p1 = 1) \vee (p1 = 2 \wedge p1' = 0))$ 在区间 $[a, \infty)$ 上的每个点上都不能成立, 但是由于 TLF(Proc1) 蕴涵 $\Box((p1' = p1 \wedge x' = x) \vee (p1 = 0) \vee (p1 = 1) \vee (p1 = 2 \wedge p1' = 0))$, 故在区间 $[a, \infty)$ 上必恒有 $p1' = p1 \wedge x' = x$, 即函数 f_x 在 $[a, \infty)$ 上连续, 这里 f_x 是 x 在 $\mathcal{J}$ 下的解释。由于 f_x 是一个时钟函数 (cf. 定义 3.3) 且它在 $[a, \infty)$ 上连续, 故必存在 $b \in [a, \infty)$ 使得 f_x 在区间 $[b, \infty)$ 上的值恒大于 5 , 于是在区间 $[b, \infty)$ 上 f_{p1} 的值恒为 2 而 f_x 的值恒大于 5 。这与 $\mathcal{J} \models \Box((p1 = 0) \vee (p1 = 1 \wedge x \leq 1) \vee (p1 = 2 \wedge x \leq 5))$ 矛盾, 于是 (4) 式成立。

(iii). 假设 (5) 不成立, 不妨设 $H \wedge \text{TLF(Proc1)} \wedge \text{TLF(Proc2)} \models \text{SF}(\alpha_2)$ 不成立。于是存在 $H \wedge \text{TLF(Proc1)} \wedge \text{TLF(Proc2)}$ 的模型 $\mathcal{J}$ 使得 $\mathcal{J} \models \neg\text{SF}(\alpha_2)$, 即

$$\mathcal{J} \models (\Diamond\Box\neg(p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2)) \wedge \Box\Diamond(p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q))). \quad (6)$$

设 f_{p1} 是 $p1$ 在 $\mathcal{J}$ 下的解释, 由 (6) 式知存在非负实数 a 使得在点 a 上有 $f_{p1}(a) = 1$, 但在区间 $[a, \infty)$ 上的每个点上 $(p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2)$ 都不能成立。 (7)

由 (2) 式知 $\mathcal{J} \models \Box((p1' = p1) \vee (p1 = 0 \wedge p1' = 1) \vee (p1 = 1 \wedge (p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge p1' = 2) \vee (p1 = 2 \wedge p1' = 0))$. (8)

由 (7)(8) 知在区间 $[a, \infty)$ 上恒有 $p1' = p1$, 于是 f_{p1} 在区间 $[a, \infty)$ 上连续, 这样它在该区间上的值就恒为 1 。 (9)

由 (9) 及 $\mathcal{J} \models TLF(Proc1)$ 还可知在区间 $[a, \infty)$ 中的每个点上都有:

$$x \leq 1 \text{ 且 } ((x' = x) \vee (x = 1 \wedge \neg(p2 = 0 \vee (p2 = 1 \wedge \neg q)) \wedge x' = 0)). \quad (10)$$

设 f_x 是 x 在 $\mathcal{J}$ 下的解释, 由 (9) 知 f_x 的值每隔一个时间单位便置 0 一次, 且在这些置 0 点上有 $\neg(p2 = 0 \vee (p2 = 1 \wedge \neg q))$, 即存在 $b \geq a$ 使得在点 $b + i$ $(i \in \mathcal{N})$ 上有 $p2 = 2 \vee (p2 = 1 \wedge q)$ 。 (11)

设 f_{p2} 是 $p2$ 在 $\mathcal{J}$ 下的解释, 下面证明对任意 $i \in \mathcal{N}$ 不可能有 $f_{p2}(b + i) = 2$ 。反之, 若存在 i 使得 $f_{p2}(b + i) = 2$, 则由 $\text{TLF}(Proc2) \models (\Box((p2' = p2 \wedge y' = y) \vee (p2 = 0) \vee (p2 = 1) \vee (p2 = 2 \wedge y \geq 3 \wedge p2' = 0 \wedge y' = 0))) \wedge \Box(p2 = 2 \Rightarrow y \leq 5)$ 知存在实数 $d \in (b + i, b + i + 5]$ 使得在点 d 上有 $(p2 = 2 \wedge p2' = 0 \wedge y' = 0)$, 再由

TLF$(Proc2)\models (p2=0\wedge y<1\Rightarrow p2'=p2)$ 知 f_{p2} 在 $(d,d+1)$ 上连续且值恒为 0，又 f_{p2} 是左连续的，故 f_{p2} 在 $(d,d+1]$ 上的值恒为 0，由于必然存在非负整数 j 使得 $b+j\in(d,d+1]$，这与 (11) 矛盾。

这样就得到了在所有的点 $b+i$ $(i\in\mathcal{N})$ 上都有 $(p2=1\wedge q)$。　(12)

设 f_q 是 q 在 $\mathcal{J}$ 下的解释，现在证明对于任意 $t\in[b,\infty)$ 有：　$f_q(t)=1$。　(13)

反之假设 (13) 不成立，则存在 $t_0\geq b$ 使得 $f_q(t_0)=1$，但同时 $f_q'(t_0)=0$；取非负整数 k 使 $b+k\leq t_0<b+k+1$，由于 f_q 在点 $b+k+1$ 上的值为 1，于是存在 $t_1\in(t_0,b+k+1)$ 使得 $f_q(t_1)=0$ 且 $f_q'(t_1)=1$，于是 t_0 和 t_1 是 f_q 的两个不连续点且 $0<t_1-t_0<1$。

设 f_z 是 z 在 $\mathcal{J}$ 下的解释，由 $\mathcal{J}\models H$ 知 f_z 在其不连续点满足 $z\geq 1\wedge z'=0$，因 z_0 和 z_1 是 f_z 的两个不连续点，故有 $f_z'(t_0)=0$ 和 $f_z(t_1)\geq 1$。但 f_z 是一时钟函数，由时钟函数的特点知 $t_1-t_0\geq f_z(t_1)-f_z'(t_0)$，于是 $t_1-t_0\geq 1$，这与 $0<t_1-t_0<1$ 矛盾，于是（ 13 ）式成立。

下面证明对于任意 $t\in[b,\infty)$ 有：　$f_{p2}(t)=1$。　(14)

反之假设 (14) 不成立，则存在 $s_0\geq b$ 使得 $f_{p2}(s_0)=1$ 且 $f_{p2}'(s_0)=2$，于是由 $\mathcal{J}\models$TLF(Proc2) 可得 $f_y'(s_0)=0$，这里 f_y 是 y 在 $\mathcal{J}$ 下的解释。于是在区间 (s_0,s_0+3) 上 f_y 的值就始终小于 3，这样由 $\mathcal{J}\models$TLF(Proc2) 便可知 f_{p2} 在区间 (s_0,s_0+3) 上连续，即 f_{p2} 在 (s_0,s_0+3) 上的值恒有 2。但由于存在 $i\in\mathcal{N}$ 使 $b+i\in(s_0,s_0+3)$，故这与 (12) 矛盾，于是 (14) 成立。

这样由 (9)(13)(14) 得，对于任意 $t\in[b,\infty)$ 有：　$f_{p1}(t)=1$ 、 $f_{p2}(t)=1$ 和 $f_q(t)=1$。　(15)

由于 TLF$(Proc2)\models\Box((p2'=p2\wedge y'=y)\vee(p2=0)\vee(p2=1\wedge\neg(p1=1\wedge q)\wedge p2'=p2)\vee(p2=1\wedge p2'=2)\vee(p2=2))$，但是由 (15) 知 $(p2=0)\vee(p2=1\wedge\neg(p1=1\wedge q)\wedge p2'=p2)\vee(p2=1\wedge p2'=2)\vee(p2=2)$ 在 $[b,\infty)$ 上的任一点上都不能成立，于是在 $[b,\infty)$ 上的任一点上都有 $(p2'=p2\wedge y'=y)$, 于是 f_y 在 $[b,\infty)$ 上连续。于是它在区间 $(b+1,\infty)$ 上的值始终大于 1。

但又有 TLF$(Proc2)\models\Box((p2=0)\vee(p2=1\wedge y\leq 1)\vee(p2=2))$，于是在区间 $[b,\infty)$ 上的任一点上都有 $(p2=1\wedge y\leq 1)$，于是 f_y 在区间 $[b,\infty)$ 上的值始终不大于 1，这与 f_y 在 $(b+1,\infty)$ 上的值始终大于 1 的结论是矛盾的。于是 (5) 式不成立的假设是错误的。

这样我们就证明了 (1) 式是正确的，即证明了在假设 '外部变量 q 的值在任意一个单位时段内最多只能改变一次' 下，进程 [Proc1||Proc2] 的确是 $[P_1\parallel P_2]$ 的一个精化。

□.

上面的两个例子虽然还比较简单，但直接使用定义（如定义 4.5 等）来手工进行证明也已经很烦琐了，而且也很难保证不出错，比较理想的方法是发展验证工具来自动检查一个实时系统是否具有某个我们所关心的性质。这在许多情况下是可能的。下一节给出的关于 LTLC/R- 公式的可满足性判断算法就可以用来自动检查一个时间模块是否满足一个可用 LTLC/R- 公式表示出来的性质。

4.3 时间模块的模型检查与 LTLC/R 公式的可满足性判定

时间模块的模型检查是指对于任一给定的时间模块 M 及任一给定的性质 φ（此处所称的性质是指一个 LTLC- 公式），判断是否有 $M \models \varphi$ 成立。由于 $M \models \varphi$ 等价于公式 $\mathrm{TLF}(M) \wedge \neg\varphi$ 是不可满足的，故时间模块的模型检查问题可化归为 LTLC 公式的可满足性问题。不幸的是，就一般情况而言，LTLC 公式的可满足性问题是不可判定的（我们还未着手去写出其证明来），但对相当一部分 LTLC 公式而言，它的可满足性是可判定的，本节将证明 LTLC/R- 公式的可满足性是可判定的；即存在一个算法，对于任给的一个 LTLC/R 公式 φ，经过有限步的计算该算法就可以回答 φ 是不是可满足的。利用这个算法就可以全自动验证一个时间模块是否满足一个用 LTLC/R- 公式表示出的性质，如可验证例 4.2 中的性质 $\Box(p = passing \Rightarrow q = closed)$ 和例 4.3 中的（1）式，我们也可用此算法来判定在逐步求精过程中两个不同抽象程度的时间模块是否是一致的（即求精后的时间模块的模型是否都是求精前的模块的模型），此外此算法还可用来判断一个时间模块是否是非空的等。

在本节以下的讨论中，n 是一个相对固定的正整数，V 是一个相对固定的有穷变量集（V 中只含布尔型变量和时钟变量，不含刚性变量、整型变量和柔性变量）。$Form_R(V, n)$ 用来表示 LTLC/R- 公式集 $\{\varphi \mid var(\varphi) \subseteq V$ 且 φ 中的常元皆不超过 $n\}$.

4.3.1 域等价

对于正整数 n, 定义函数 $h_n : \mathcal{R} \mapsto \{-2n-1, -2n, \ldots, -1, 0, 1, \ldots, 2n, 2n+1\}$ 如下：

$$h_n(a) = \begin{cases} 2a & \text{若 } a = \lfloor a \rfloor \text{ 且 } |a| \le n, \\ 2\lfloor a \rfloor + 1 & \text{若 } a \ne \lfloor a \rfloor \text{ 且 } |a| \le n, \\ 2n+1 & \text{若 } a > n, \\ -2n-1 & \text{若 } a < -n. \end{cases}$$

这里 $\lfloor a \rfloor, |a|$ 分别表示实数 a 的整数部分和 a 的绝对值.

不难验证 $h_n(a)$ 是单增的且对于不超过 n 的非负整数 m 有： $h_n(a) \leq 2m$ 当且仅当 $a \leq m$.

为方便起见下面常以 $[a]_n$ 记 $h_n(a)$.

设 $V = B \cup C$, 其中 B 是布尔型变量集， C 是时钟变量集。令 $V' = \{v'|v \in V\}$. 称 $V \cup V'$ 上的映射 σ 为变量集 V 上的一个 LTLC/R- 状态，若对任意 $p \in B$ 有 $\sigma(p) \in \{0,1\}$, $\sigma(p') \in \{0,1\}$, 且对任意 $c \in C$ 有 $\sigma(c) \in \mathcal{R}^+$ 、 $\sigma(c') = 0 \vee \sigma(c') = \sigma(c)$. 若 σ 是变量集 V 上的一个状态，对任意状态公式 φ, 我们可按通常的做法定义 φ 在 σ 下的真值 $\sigma(\varphi)$.

对于正整数 n, 定义 V 上的状态之间的等价关系 $\equiv_n$ 如下：

定义 4.6 设 σ 和 τ 是 V 上的任意两个状态， $\sigma \equiv_n \tau$ 当且仅当

1. 对任意 $p \in B$ 有： $\sigma(p) = \tau(p), \sigma(p') = \tau(p')$;

2. 对任意 $c \in C$ 有： $[\sigma(c)]_n = [\tau(c)]_n, [\sigma(c')]_n = [\tau(c')]_n$;

3. 对任意 $c, d \in C$ 有： $[\sigma(c) - \sigma(d)]_n = [\tau(c) - \tau(d)]_n$.

显然， '$\equiv_n$' 是一个等价关系，记 σ 所在的等价类为 $[\sigma]_n$.

引理 4.1 设 σ, τ 是 V 上的两个等价状态， $\varphi \in Form(V, n)$ 且 φ 是一个状态公式，则 $\sigma(\varphi) = \tau(\varphi)$.

证明：对公式 φ 进行归纳即可得，这里只给出情况 $\varphi := (c' = c)$ 时的证明作为示例。

假设 $\sigma(c' = c) = 1$, 而 $\tau(c' = c) = 0$ 。于是 $\sigma(c') = \sigma(c)$, 且 $\tau(c') \neq \tau(c)$ 。由状态的定义知 τ 满足 $\tau(c') = 0 \vee \tau(c') = \tau(c)$, 于是 $\tau(c') = 0$, 再由 $\sigma \equiv_n \tau$ 得出 $\sigma(c') = 0$, 进一步有 $\sigma(c) = 0$ 和 $\tau(c) = 0$, 但这又与 $\tau(c') \neq \tau(c)$ 矛盾.

□.

定义 4.7 设 σ 是 V 上的一个状态， $t \geq 0$ 。定义 $\sigma + t$ 是这样一个状态：对于任意 $p \in B$ 有 $(\sigma+t)(p) = \sigma(p')$ 且 $(\sigma+t)(p') = \sigma(p')$, 对于任意 $c \in C$ 有 $(\sigma+t)(c) = \sigma(c')+t$ 且 $(\sigma + t)(c') = \sigma(c') + t$.

注意根据上面的这个定义， σ 不一定就等于 $\sigma + 0$ ，但当 σ 是一个连续状态时 (见下面定义 4.8) ，则有 σ 一定等于 $\sigma + 0$ 。

定义 4.8 设 σ 是 V 上的一个状态。

1. 称 σ 是不连续状态，记为 *discontin(*σ*)* ，若存在 $p \in B$ 使得 $\sigma(p) \neq \sigma(p')$, 或存在 $c \in C$ 使得 $\sigma(c') = 0$; 否则称其为连续状态，记为 *contin(*σ*)*.

2. 称 σ 是边界状态，记为 *boundary(*σ*)* ，若存在 $c \in C$ 使得 $\sigma(c') \in \{0, 1, 2, \ldots, n\}$.

定义 4.9 设 σ 和 τ 是 V 上的任意两个状态. 称 $\sigma \rhd \tau$, 若 *contin(*σ*)* 且对于任意 $v \in V$ 有 $\tau(v) = \sigma(v)$ (注意 $\tau(v')$ 不一定等于 $\tau(v)$)。

定义 4.10 设 σ 和 τ 是 V 上的任意两个状态， $t > 0$ 。称 $\sigma \overset{t}{\rightsquigarrow} \tau$, 若 $((\sigma + t) \rhd \tau) \wedge \forall s((s > 0 \wedge s < t) \Rightarrow (\sigma + s \equiv_n \sigma \vee \sigma + s \equiv_n \tau))$ 成立。称 $\sigma \rightsquigarrow \tau$, 若存在正实数 t 使得 $\sigma \overset{t}{\rightsquigarrow} \tau$ 成立。

定义 4.11 设 $\mathcal{J}$ 是 V 上的一个 *LTLC-* 模型， $t \in \mathcal{R}^+$, 我们用 $\mathcal{J}(t)$ 来表示 V 上如下定义的一个状态：对任意的 $v \in V$ ， $\mathcal{J}(t)(v) ::= \mathcal{J}(v, t)$, $\mathcal{J}(t)(v') ::= \mathcal{J}(v', t)$ 。

1. 若 *discontin(*$\mathcal{J}(t)$*)*, 则称 t 是 $\mathcal{J}$ 的一个不连续点.

2. 若 *boundary(*$\mathcal{J}(t)$*)*, 则称 t 是 $\mathcal{J}$ 的一个边界点.

由定义 3.2 和定义 3.3 知 $\mathcal{J}$ 的不连续点和边界点均为 $\mathcal{R}^+$ 上的孤立点.

引理 4.2 设 $\mathcal{J}$ 是 V 上的一个 *LTLC-* 模型， $(a, b) \subset \mathcal{R}^+$;

1. 若 $\mathcal{J}$ 在 (a, b) 上无不连续点，则对任意 $s \in (a, b)$ 有： $\mathcal{J}(s) = \mathcal{J}(a) + (s - a)$.

2. 若 $\mathcal{J}$ 在 (a, b) 上无不连续点和边界点，则对任意 $s_1, s_2 \in (a, b)$ 有： $\mathcal{J}(s_1) \equiv_n \mathcal{J}(s_2)$.

证明:

1. **(i)** 对于任意 $p \in B$, 由于 $\mathcal{J}$ 在 (a, b) 上无不连续点，于是 f_p 在 (a, b) 上连续 (这里 f_p 是 p 在 $\mathcal{J}$ 下的解释) ，故由定义 3.2 知 f_p 在 (a, b) 上是一个常数 (0 或 1)，于是对任意 $s \in (a, b)$ 有 $f_p'(s) = f_p(s) = f_p'(a)$.

(ii) 对于任意 $c \in C$, 由于 $\mathcal{J}$ 在 (a, b) 上无不连续点，于是 f_c 在 (a, b) 上连续 (这里 f_c 是 c 在 $\mathcal{J}$ 下的解释) ，又由定义 3.3 知 f_c 在 (a, b) 上的导数恒为 1 。于是对任意 $s \in (a, b)$ 有 $f_c'(s) = f_c(s) = f_c'(a) + (s - a)$.

这样由 (i)(ii) 及定义 4.7 便得结论 (1).

2. **(i)** 对于任意 $p \in B$, 由结论 (1) 知对于任意 $s_1, s_2 \in (a, b)$ 有： $f_p'(s_1) = f_p(s_1) = f_p'(a) = f_p'(s_2) = f_p(s_2)$.

(ii) 对于任意 $c \in C$，由于 f_c 在 (a, b) 上连续且不取集合 $\{0, 1, 2, \ldots, n\}$ 中的值，于是复合函数 $h_n(f_c)$ 在 (a, b) 上仍然是一个连续函数。由于 (a, b) 是一个连通区间，故由连续函数的介值定理知 $h_n(f_c)$ 在 (a, b) 上的值域 $\{h_n(f_c(s)) \mid s \in (a, b)\}$ 仍然是一个连通的区间，但由于 h_n 的取值只能是离散的整数点，故 $\{h_n(f_c(s)) \mid s \in (a, b)\}$ 只可能是一个单点集，于是对任意 $s_1, s_2 \in (a, b)$ 有：$h_n(f_c(s_1))) = h_n(f_c(s_2)))$.

(iii) 对于任意 $c_1, c_2 \in C$ 及任意 $s_1, s_2 \in (a, b)$，由结论 (1) 知 $f_{c_1}(s_1) - f_{c_2}(s_1) = f_{c_1}(a)+s_1-a-(f_{c_2}(a)+s_1-a) = f_{c_1}(a)-f_{c_2}(a) = f_{c_1}(a)+s_2-a-(f_{c_2}(a)+s_2-a) = f_{c_1}(s_2) - f_{c_2}(s_2)$.

这样由 (i)(ii)(iii) 及定义 4.6 便得结论 (2).

□.

定义 4.12 设 $\mathcal{J}$ 是 V 上的一个 *LTLC-* 模型，$t_0, t_1, t_2, t_3 \ldots$ 是一个时间序列 *(*见定义 *3.21)*, 且其中包含了 $\mathcal{J}$ 的所有不连续点; 若对于任意 $i \in \mathcal{N}$ 有：$\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$, 则称 $\mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots$ 是 $\mathcal{J}$ 的一个执行序列。

引理 4.3 设 $\mathcal{J}$ 是 V 上的一个模型，则存在时间序列 $t_0, t_1, t_2, t_3, \ldots$ 使得 $\mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots$ 是 $\mathcal{J}$ 的一个执行序列.

证明：设 G 是 $\mathcal{J}$ 的所有边界点和不连续点构成的集合，则 G 是 $\mathcal{R}^+$ 上的一个孤立点集.

情况 1. 当 G 是有限集时，设 G 中的元素 (从小到大) 为 $s_0, s_1, s_2, \ldots, s_m$ 。对于大于 m 的整数 i 定义 $s_i ::= s_m + 2(i - m)$. 对于这个无穷序列 $\{s_i\}_{i\in\mathcal{N}}$ 定义 t_i 如下：

$$t_i = \begin{cases} s_{(i/2)} & \text{若 } i \text{ 是偶数} \\ (s_{((i-1)/2)} + s_{(((i+1)/2))})/2 & \text{若 } i \text{ 是奇数} \end{cases}$$

由于 $s_0 = 0$ 且 $\{s_i\}_{i\in\mathcal{N}}$ 发散，故显然 $\{t_i\}_{i\in\mathcal{N}}$ 是一个时间序列；下面证明对于任意 $i \in \mathcal{N}$ 有：$\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$ 。

(i). 当 i 为偶数时 (设 $i = 2k$)，由于 $\mathcal{J}$ 在 (s_k, s_{k+1}) 中无不连续点和边界点，故由引理 4.2 知对于任意 $s \in (t_i, t_{i+1}] \subset (s_k, s_{k+1})$ 有 $\mathcal{J}(s) = \mathcal{J}(t_i) + (s - t_i)$ 且 $\mathcal{J}(s) \equiv_n \mathcal{J}(t_{i+1})$. 根据定义 4.10 知 $\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$ 。

(ii). 当 i 为奇数时 (设 $i = 2k + 1$), 由于 $\mathcal{J}$ 在 (s_k, s_{k+1}) 中无不连续点和边界点，由引理 4.2 知对于任意 $s \in [t_i, t_{i+1}) \subset (s_k, s_{k+1})$ 有 $\mathcal{J}(s) = \mathcal{J}(t_i) + (s - t_i)$ 且 $\mathcal{J}(s) \equiv_n \mathcal{J}(t_i)$，再由 $\mathcal{J}$ 的左连续性知 $\mathcal{J}(t_i) + (t_{i+1} - t_i) \rhd \mathcal{J}(t_{i+1})$, 于是对照定义 4.10 知 $\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$ 。

由 (i)(ii) 及定义 4.12 便知 $\{t_i\}_{i\in\mathcal{N}}$ 满足要求。

情况 2. 当 G 是无限集时，由于 G 是孤立点集，故可设 G 中的元素可以从小到大排列为 $s_0, s_1, s_2,\ldots,s_k,\ldots$.

定义 t_i 如下：

$$t_i = \begin{cases} s_{(i/2)} & \text{若 } i \text{ 是偶数} \\ (s_{((i-1)/2)} + s_{((i+1)/2))})/2 & \text{若 } i \text{ 是奇数} \end{cases}$$

这样定义的无穷序列满足引理要求，证明过程同情况 1.

□.

引理 4.4 设 $\mathcal{J}$ 是 V 上的一个 *TLTC*- 模型， $t_0, t_1, t_2, t_3\ \ldots$ 是一个时间序列；若 $\mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots$ 是 $\mathcal{J}$ 的一个执行序列，则对任意 $i \in \mathcal{N}$ 有：

1. $\mathcal{J}$ 在区间 (t_i, t_{i+1}) 中没有不连续点和边界点。

2. $\{\mathcal{J}(s) \mid s \in (t_i, t_{i+1})\}$ 中的状态相互等价且等价于 $\mathcal{J}(t_i)$ 或 $\mathcal{J}(t_{i+1})$.

3. 若 t_i 是不连续点或边界点，则 t_{i+1} 必定是连续的非边界点。

证明:

(1). 对任意 $i \in \mathcal{N}$ ，由定义 4.12 知 (t_i, t_{i+1}) 中没有不连续点；下证 (t_i, t_{i+1}) 中没有边界点;

由引理 4.2 知对于任意的 $s \in (t_i, t_{i+1})$ 有 $\mathcal{J}(s) = \mathcal{J}(t_i) + (s - t_1)$. 假若 $s_0 \in (t_i, t_{i+1})$ 是 $\mathcal{J}$ 的一个边界点，则存在 $c \in C$ 使 $f_c(s_0) \in \{1, 2, 3, \ldots, n\}$(注意 $f_c(s_0) \neq 0$, 因为 s_0 是 $\mathcal{J}$ 的连续点) 。于是取 $s_1 \in (t_i, s_0)$ $s_2 \in (s_0, t_{k+1})$, 则 $f_c(s_1) = f'_c(t_i) + s_1 - t_i < f'_c(t_i) + s_0 - t_i = f_c(s_0) < f'_c(t_i) + s_2 - t_i = f_c(s_2)$, 由于 $f_c(s_0)$ 是不超过 n 的正整数，故 $h_n(f_c(s_0)), h_n(f_c(s_1)), h_n(f_c(s_2))$ 两两互异，于是 $\mathcal{J}(s_0)$, $\mathcal{J}(s_1)$, $\mathcal{J}(s_2)$ 三者两两互不等价，即存在 $t_0, t_1, t_2 \in (0, t_{i+1} - t_i)$ 使 $(\mathcal{J}(t_i) + t_0)\,(\mathcal{J}(t_i) + t_1)\,(\mathcal{J}(t_i) + t_2)$ 三者两两互不等价。这与 $\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$ 矛盾；于是 (t_i, t_{i+1}) 中没有边界点。

(2). 由 (1) 及引理 4.2 知 $\{\mathcal{J}(s) \mid s \in (t_i, t_{i+1})\}$ 中的状态相互等价; 再由 $\mathcal{J}(t_i) \overset{t_{i+1}-t_i}{\rightsquigarrow} \mathcal{J}(t_{i+1})$ 及定义 4.10 知 $\{\mathcal{J}(s) \mid s \in (t_i, t_{i+1})\}$ 中的状态等价于 $\mathcal{J}(t_i)$ 或 $\mathcal{J}(t_{i+1})$.

(3). 若 t_i 是不连续点或边界点，取 $s \in (t_i, t_{i+1})$ ，由于 $\mathcal{J}$ 在区间 (t_i, t_{i+1}) 中的点都是连续的非边界点；故 $\mathcal{J}(s)$ 不可能与 $\mathcal{J}(t_i)$ 等价，于是由 (2) 便得它只能和 $\mathcal{J}(t_{i+1})$ 等价，于是 t_{i+1} 必定也是连续的非边界点。

□.

定义 4.13 设 $\Im =< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是模型 $\mathcal{J}$ 的一个执行序列，定义函数 $\ell_\Im : \mathcal{R}^+ \mapsto \mathcal{N}$ 如下：

$$\ell_{\Im}(s)=\begin{cases} i & 若s=t_i\vee(s\in(t_i,t_{i+1})\wedge\neg(discontin(\mathcal{J}(t_i))\vee boundary(\mathcal{J}(t_i)))) \\ i+1 & 若s\in(t_i,t_{i+1})\wedge(discontin(\mathcal{J}(t_i))\vee boundary(\mathcal{J}(t_i))) \end{cases}$$

不难看出，函数 $\ell_{\Im}$ 是满的和单增的，且 (由引理 4.4) 对于任意 $s\in\mathcal{R}^+$ 有： $\mathcal{J}(s)\equiv_n \mathcal{J}(t_{\ell_{\Im}(s)})$.

4.3.2　构造 LTL 公式 $\widehat{\varphi}$

设有穷变量集合 $V=B\cup C$, 其中 B,C 分别为布尔型变量集和时钟命题变量集， n 是一个正整数。对于 $\varphi\in Form_R(V,n)$, 以下我们构造一个 LTL 公式 $\widehat{\varphi}$， $\widehat{\varphi}$ 中的变量属于集合 $V_d=B\cup C\cup D\cup B'\cup C'\cup D'$, 其中 $D=\{\Delta_{cd}\,|\,c,d\in C\}$, $B'=\{p'\,|\,p\in B\}$, $C'=\{c'\,|\,c\in C\}$, $D'=\{\Delta'_{cd}\,|\,c,d\in C\}$. $B\cup B'$ 中的变量是命题变量，取值 0 和 1; $C\cup C'$ 中的变量为整型变量，取值域为 $\{0,1,2,\ldots,2n+1\}$; $D\cup D'$ 中的变量也是整型变量，取值域为 $\{-2n-1,-2n,\ldots,-2,-1,0,1,2,\ldots,2n+1\}$ 。我们要证明对于构造的 $\widehat{\varphi}$ (见定义 4.19) 有 φ 是 LTLC 可满足的当且仅当 $\widehat{\varphi}$ 是 LTL 可满足的 (见引理 4.10 和引理 4.13) 。

定义 4.14 对于 V 上的一个状态 σ, 定义其在 V_d 上的投影状态 $\overline{\sigma}$ 如下：

1. 对任意 $p\in B$, $\overline{\sigma}(p)=\sigma(p)$, $\overline{\sigma}(p')=\sigma(p')$;

2. 对任意 $c\in C$, $\overline{\sigma}(c)=[\sigma(c)]_n$, $\overline{\sigma}(c')=[\sigma(c')]_n$;

3. 对任意 $c,d\in C$, $\overline{\sigma}(\Delta_{cd})=[\sigma(c)-\sigma(d)]_n$, $\overline{\sigma}(\Delta'_{cd})=[\sigma(c')-\sigma(d')]_n$.

显然，对于 V 上的任意两个状态 σ 和 τ 有： $\overline{\sigma}=\overline{\tau}$ 当且仅当 $[\sigma]_n=[\tau]_n$(即当且仅当 $\sigma\equiv_n\tau$).

定义 4.15 对于 $\varphi\in Form_R(V,n)$, 归纳定义 V_d 上的 *LTL* 公式 $\overline{\varphi}$ 如下：

1. 对 $p\in B$， $\overline{p}::=p$, $\overline{p'}::=p'$;

2. 对 $c\in C$， $\overline{c\le m}::=c\le 2m$, $\overline{c=m}::=c=2m$, $\overline{c'=c}::=c'=c$, $\overline{c'=0}::=c'=0$;

3. $\overline{\neg\varphi}::=\neg\overline{\varphi}$, $\overline{\varphi_1\wedge\varphi_2}::=\overline{\varphi_1}\wedge\overline{\varphi_2}$, $\overline{\Box\varphi}::=\Box\overline{\varphi}$, $\overline{\varphi_1\mathcal{U}\varphi_2}::=\overline{\varphi_1}\mathcal{U}\overline{\varphi_2}$

引理 4.5 设 σ 是 V 上的一个状态， $\varphi\in Form_R(V,n)$ 且它是一个状态公式，则 $\sigma(\varphi)=\overline{\sigma}(\overline{\varphi})$ 。

证明：对 φ 用归纳即可，这里只给出情况 $\varphi := (c \leq m)$ 时的证明作为示例。

$\overline{\sigma}(\overline{\varphi}) = 1$ *iff* $\overline{\sigma}(c \leq 2m) = 1$ *iff* $\overline{\sigma}(c) \leq 2m$ *iff* $[\sigma(c)]_n \leq 2m$ *iff* $\sigma(c) \leq m$ *iff* $\sigma(\varphi) = 1$.

定义 4.16 设 $\Im =< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是 *LTLC*- 模型 $\mathcal{J}$ 的一个执行序列，称序列 $< \overline{\mathcal{J}(t_0)}, \overline{\mathcal{J}(t_1)}, \overline{\mathcal{J}(t_2)}, \ \ldots >$ *(*这里 $\overline{\mathcal{J}(t_i)}$ 是 $\mathcal{J}(t_i)$ 在 V_d 上的投影状态*)* 为序列 $\Im$ 在 V_d 上的投影序列，记为 $\overline{\Im}$ 。

引理 4.6 设 $\Im =< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是模型 $\mathcal{J}$ 的一个执行序列，$\overline{\Im}$ 是序列 $\Im$ 在 V_d 上的投影序列，对于 $\varphi \in Form_R(V, n)$ ，有： $\mathcal{J} \models \varphi$ 当且仅当 $\overline{\Im} \models_{ltl} \overline{\varphi}$.

证明：只需证明对任意 $s \in \mathcal{R}^+$ 有： $\mathcal{J}(\varphi, s) = \overline{\Im}(\overline{\varphi}, \ell_{\Im}(s))$ 即可.

对 φ 进行归纳，归纳基础可由引理 4.5 得到，下面给出 $\varphi := \Box\phi$ 情况下的证明，其余情况略.

1. 若 $\mathcal{J}(\Box\phi, s) = 1$, 则对于任意 $r \geq s$ 有 $\mathcal{J}(\phi, r) = 1$, 应用归纳假设可得，对任意 $r \geq s$ 都有 $\overline{\Im}(\overline{\phi}, \ell_{\Im}(r)) = 1$ 。考虑 $\ell_{\Im}$ 的特性，由于 $\ell_{\Im}$ 是满射，且是单增的，于是可得 $\overline{\Im}(\overline{\Box\phi}, \ell_{\Im}(s)) = 1$.

2. 若 $\mathcal{J}(\Box\phi, s) = 0$, 则存在 $r \geq s$ 使得 $\mathcal{J}(\phi, r) = 0$, 由归纳假设知 $\overline{\Im}(\overline{\phi}, \ell_{\Im}(r)) = 0$. 由于 $\ell_{\Im}(r) \geq \ell_{\Im}(s)$, 于是 $\overline{\Im}(\overline{\Box\phi}, \ell_{\Im}(s)) = 0$.

□.

定义 4.17 对于前面的变量集 V_d ， 我们引进一些 V_d 上的公式的缩写形式：

$B^+ = B' ::= \bigwedge_{p \in B}(p^+ = p')$

$(B')^+ = B' ::= \bigwedge_{p \in B}((p')^+ = p')$

$C^+ = C' ::= \bigwedge_{c \in C}(c^+ = c')$

$(C')^+ = C' ::= \bigwedge_{c \in C}((c')^+ = c')$

$(C')^+ = C^+ ::= \bigwedge_{c \in C}((c')^+ = c^+)$

$D^+ = D' ::= \bigwedge_{v \in D}(v^+ = v')$

$(D')^+ = D' ::= \bigwedge_{v \in D}((v')^+ = v')$

$discontin ::= (\bigvee_{c \in C}(c' = 0)) \vee \bigvee_{p \in B} \neg(p = p')$

$boudary(c) ::= (c' = 0 \vee c' = 2 \vee c' = 4 \vee \ldots \vee c' = 2n)$

$boundary ::= \bigvee_{c\in C} boudary(c)$

$critical(c) ::= (c' \leq 2n) \wedge \bigwedge_{d\in C}(d' \leq 2n \Rightarrow c' \leq d' + \Delta'_{cd})$

$next1 ::= (\bigwedge_{c\in C}((boundary(c) \Rightarrow c^+ = c' + 1) \wedge (\neg boundary(c) \Rightarrow c^+ = c'))) \wedge ((C')^+ = C^+)$

$next2 ::= \bigwedge_{c\in C}((critical(c) \Rightarrow c^+ = c' + 1) \wedge (\neg critical(c) \Rightarrow c^+ = c'))$

引理 4.7 设 $\Im =< \mathcal{J}(t_0),\ \mathcal{J}(t_1),\ \mathcal{J}(t_2),\ \ldots >$ 是模型 $\mathcal{J}$ 的一个执行序列，$\overline{\Im} = < \overline{\mathcal{J}(t_0)},\ \overline{\mathcal{J}(t_1)},\ \overline{\mathcal{J}(t_2)},\ \ldots >$ 是序列 $\Im$ 在 V_d 上的投影序列。对任意的 $i \in \mathcal{N}$,

(1). 当 t_i 是 $\mathcal{J}$ 的边界点时,

(1.1). 对于任意 $v \in B \cup D$ 有 $\overline{\mathcal{J}(t_{i+1})}(v') = \overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$ 。

(1.2). 对于任意 $c \in C$, 若 $\mathcal{J}(t_i)(c') \in \{0,\ 1,\ 2,\ \ldots,\ n\}$ ，则 $\overline{\mathcal{J}(t_{i+1})}(c') = \overline{\mathcal{J}(t_{i+1})}(c) = \overline{\mathcal{J}(t_i)}(c')+1$;若 $\mathcal{J}(t_i)(c') \notin \{0,\ 1,\ 2,\ \ldots,\ n\}$,则 $\overline{\mathcal{J}(t_{i+1})}(c') = \overline{\mathcal{J}(t_{i+1})}(c) = \overline{\mathcal{J}(t_i)}(c')$ 。

(2). 当 t_i 不是 $\mathcal{J}$ 的边界点但却是 $\mathcal{J}$ 的不连续点时，对于任意 $v \in B \cup C \cup D$ 有：$\overline{\mathcal{J}(t_{i+1})}(v') = \overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$ 。

(3). 当 t_i 是 $\mathcal{J}$ 的一个非边界的连续点时，设 $H_i = \{c \,|\, c \in C$且$\mathcal{J}(t_{i+1})(c) \in \{1, 2, 3, 4, \ldots, n\}\}$;

(3.1). 若 H_i 是空集，则对任意 $v \in B \cup C \cup D$ 有：$\overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$;

(3.2). 若 H_i 非空，则对任意 $c \in H_i$ 有 $\overline{\mathcal{J}(t_{i+1})}(c) = \overline{\mathcal{J}(t_i)}(c') + 1$ ；对任意 $v \in B \cup (C - H_i) \cup D$ 有 $\overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$ 。

证明:

(1). 当 t_i 是 $\mathcal{J}$ 的边界点时，由引理 4.4 知 $\mathcal{J}$ 在区间 $(t_i,\ t_{i+1}]$ 上是连续的，故对于任意 $v \in B \cup D$ 有 $\overline{\mathcal{J}(t_{i+1})}(v') = \overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$ 。取 δ 为 $t_{i+1} - t_i$ 和 Min$\{\ 1 - <\mathcal{J}(t_i)(c')> \,|\, c \in C,\ \mathcal{J}(t_i)(c') \leq n\ \}$ 之间的最小值 (这里 $<\mathcal{J}(t_i)(c')>$ 表示 $\mathcal{J}(t_i)(c')$ 的小数部分， Min 表示取集合中的最小元) ，令 $\sigma ::= \mathcal{J}(t_i) + \frac{1}{2}\delta$ ，由于 $0 < \frac{1}{2}\delta < t_{i+1} - t_i$, 故由引理 4.4 知 σ 等价于 $\mathcal{J}(t_{i+1})$ 。

于是对任意 $c \in C$, 当 $\mathcal{J}(t_i)(c') \in \{0, 1, 2, \ldots, n\}$ 时，由于 $0 < \frac{1}{2}\delta < 1$, 故 $\overline{\sigma}(c) = [\mathcal{J}(t_i)(c') + \frac{1}{2}\delta]_n = 2\mathcal{J}(t_i)(c') + 1 = \overline{\mathcal{J}(t_i)}(c') + 1$ ，于是 $\overline{\mathcal{J}(t_{i+1})}(c') =$

$\overline{\mathcal{J}(t_{i+1})}(c) = \overline{\sigma}(c) = \overline{\mathcal{J}(t_i)}(c')+1$；当 $\mathcal{J}(t_i)(c) \notin \{0,1,2,\ldots,n\}$ 时，由于 $0 < \frac{1}{2}\delta < 1- <\mathcal{J}(t_i)(c')>$，于是 $(\mathcal{J}(t_i)(c') + \frac{1}{2}\delta) \notin \{0,1,2,\ldots,n\}$ 且 $\lfloor \mathcal{J}(t_i)(c') + \frac{1}{2}\delta \rfloor = \lfloor \mathcal{J}(t_i)(c') \rfloor$，于是 $\overline{\sigma}(c) = [\mathcal{J}(t_i)(c') + \frac{1}{2}\delta]_n = 2\lfloor \mathcal{J}(t_i)(c') \rfloor + 1 = \overline{\mathcal{J}(t_i)}(c')$，于是 $\overline{\mathcal{J}(t_{i+1})}(c') = \overline{\mathcal{J}(t_{i+1})}(c) = \overline{\sigma}(c) = \overline{\mathcal{J}(t_i)}(c')$。

(2). 当 t_i 不是 $\mathcal{J}$ 的边界点但却是 $\mathcal{J}$ 的不连续点时，证明过程与 (1) 相似，在此从略。

(3). 当 t_i 是 $\mathcal{J}$ 的一个非边界的连续点时.

对于任意 $v \in B \cup D$，由引理 4.4 知 $\mathcal{J}$ 在区间 (t_i, t_{i+1}) 上是连续的，故有 $\overline{\mathcal{J}(t_{i+1})}(v) = \overline{\mathcal{J}(t_i)}(v')$。这样只要证明对于时钟变量所述结论成立即可。

由引理 4.4 还知对于任意 $s \in (0, t_{i+1} - t_i)$ 有：$\mathcal{J}(t_i) + s$ 等价于 $\mathcal{J}(t_i)$，于是 $t_{i+1} - t_i \leq \mathrm{Min}\{\, 1- <\mathcal{J}(t_i)(c')> \mid c \in C, \mathcal{J}(t_i)(c') \leq n \,\}$.

因 $\Im := < \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是解释 $\mathcal{J}$ 的一个执行序列，故由定义 4.12 和定义 4.10 知 $\mathcal{J}(t_i) + t_{i+1} - t_i \rhd \mathcal{J}(t_{i+1})$，于是对任意 $c \in C$ 有 $\mathcal{J}(t_{i+1})(c) = \mathcal{J}(t_i)(c') + t_{i+1} - t_i$。

(3.1). 若 H_i 是空集，则必有 $t_{i+1} - t_i < \mathrm{Min}\{1- <\mathcal{J}(t_i)(c')> \mid c \in C, \mathcal{J}(t_i)(c') \leq n \,\}$. 于是对任意 $c \in C$，我们有 $(\mathcal{J}(t_i)(c') + t_{i+1} - t_i) \notin \{0,1,2,\ldots,n\}$ 且 $\lfloor \mathcal{J}(t_i)(c') + t_{i+1} - t_i \rfloor = \lfloor \mathcal{J}(t_i)(c') \rfloor$，于是 $\overline{\mathcal{J}(t_{i+1})}(c) = [\mathcal{J}(t_i)(c') + t_{i+1} - t_i]_n = 2\lfloor \mathcal{J}(t_i)(c') \rfloor + 1 = \overline{\mathcal{J}(t_i)}(c')$。

(3.2). 若 H_i 非空，则必有 $t_{i+1} - t_i = \mathrm{Min}\{\, 1- <\mathcal{J}(t_i)(c')> \mid c \in C, \mathcal{J}(t_i)(c') \leq n \,\}$.

对任意 $c \in H_i$，我们有 $t_{i+1} - t_i = 1- <\mathcal{J}(t_i)(c')>$，于是 $\overline{\mathcal{J}(t_{i+1})}(c) = [\mathcal{J}(t_i)(c') + t_{i+1} - t_i]_n = [\lfloor \mathcal{J}(t_i)(c') \rfloor + 1]_n = 2\lfloor \mathcal{J}(t_i)(c') \rfloor + 2 = [\mathcal{J}(t_i)(c')]_n + 1 = \overline{\mathcal{J}(t_i)}(c') + 1$。

对任意 $c \notin H_i$，我们有 $t_{i+1} - t_i < 1- <\mathcal{J}(t_i)(c')>$，类似于 (3.1) 的证明可得 $\overline{\mathcal{J}(t_{i+1})}(c) = \overline{\mathcal{J}(t_i)}(c')$。

□.

引理 4.8 设 $\Im = < \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是 *LTLC*- 模型 $\mathcal{J}$ 的一个执行序列，$\overline{\Im} = < \overline{\mathcal{J}(t_0)}, \overline{\mathcal{J}(t_1)}, \overline{\mathcal{J}(t_2)}, \ldots >$ 是序列 $\Im$ 在 V_d 上的投影序列. 对任意的 $i \in \mathcal{N}$,

1. t_i 是 $\mathcal{J}$ 的边界点当且仅当 $\overline{\mathcal{J}(t_i)}(boundary) = 1$.

2. t_i 是 $\mathcal{J}$ 的不连续点当且仅当 $\overline{\mathcal{J}(t_i)}(discontin) = 1$

3. 当 t_i 是 $\mathcal{J}$ 的一个非边界的连续点且 H_i（见引理 *4.7*）非空时，$c \in H_i$ 当且仅当 $\overline{\mathcal{J}(t_i)}(critical(c)) = 1$

证明：(1) 和 (2) 比较显然，证明此处从略。

(3). 当 t_i 是 $\mathcal{J}$ 的一个非边界的连续点且 H_i 非空时，由引理 4.7 的证明知当 H_i 非空时有 $t_{i+1} - t_i = \mathrm{Min}\{\ 1- <\mathcal{J}(t_i)(c')> \mid c \in C,\ \mathcal{J}(t_i)(c') \leq n\ \}$；并且 $c \in H_i$ 当且仅当 $\mathcal{J}(t_i)(c') \leq n$ 且 $t_{i+1} - t_i = 1- <\mathcal{J}(t_i)(c')>$。于是 $c \in H_i$ 当且仅当 $\mathcal{J}(t_i)(c') \leq n$ 且对于任意 $d \in C$, 若 $\mathcal{J}(t_i)(d') \leq n$ 则 $1- <\mathcal{J}(t_i)(c')> \leq 1- <\mathcal{J}(t_i)(d')>$.

由于当 $\mathcal{J}(t_i)(c') \leq n$ 且 $\mathcal{J}(t_i)(d') \leq n$ 时有：$1- <\mathcal{J}(t_i)(c')> \leq 1- <\mathcal{J}(t_i)(d')>$ 当且仅当 $<\mathcal{J}(t_i)(c')> \geq <\mathcal{J}(t_i)(d')>$ 当且仅当 $[\mathcal{J}(t_i)(c')]_n \leq [\mathcal{J}(t_i)(d')]_n + [\mathcal{J}(t_i)(c') - \mathcal{J}(t_i)(d')]_n$.

于是 $c \in H_i$ 当且仅当 $\overline{\mathcal{J}(t_i)}(c') \leq 2n$ 且对任意 $d \in C$ 有：若 $\overline{\mathcal{J}(t_i)}(d') \leq 2n$ 则 $\overline{\mathcal{J}(t_i)}(c') \leq \overline{\mathcal{J}(t_i)}(d') + \overline{\mathcal{J}(t_i)}(\Delta'_{cd})$。即就是有 $c \in H_i$ 当且仅当 $\overline{\mathcal{J}(t_i)}(critical(c)) = 1$。

□.

定义 4.18 定义 V_d 上的公式 *Next*, *clock_constra* 和 *liveness* 如下：

$$
\begin{aligned}
Next \quad ::= & (B^+ = B') \wedge (D^+ = D') \wedge (\\
& (boundary \Rightarrow (((B')^+ = B') \wedge ((D')^+ = D') \wedge next1)) \wedge \\
& ((discontin \wedge \neg boundary) \Rightarrow (((B')^+ = B') \wedge (C^+ = C') \wedge \\
& \quad ((C')^+ = C') \wedge ((D')^+ = D'))) \wedge \\
& ((\neg discontin \wedge \neg boundary) \Rightarrow (next2 \vee (C^+ = C'))) \\
&)
\end{aligned}
$$

$$
\begin{aligned}
clock\_constra \quad ::= & (\textstyle\bigwedge_{c \in C}(c = 0 \wedge c' = 0)) \wedge (\bigwedge_{c,d \in C}(\Delta_{cd} = 0 \wedge \Delta'_{cd} = 0)) \wedge \\
& (\Box \textstyle\bigwedge_{c \in C}(c = c' \vee c' = 0)) \wedge \\
& \Box(\textstyle\bigwedge_{c,d \in C}((c' = c \wedge d' = d \Rightarrow \Delta'_{cd} = \Delta_{cd}) \wedge \\
& \quad (c' = 0 \Rightarrow \Delta'_{cd} = -d') \wedge (d' = 0 \Rightarrow \Delta'_{cd} = c')))
\end{aligned}
$$

$$
liveness \qquad ::= \textstyle\bigwedge_{c \in C} \Box\Diamond(c' = 0 \vee c > 2n)
$$

引理 4.9 设 $\Im =< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是解释 $\mathcal{J}$ 的一个执行序列，$\overline{\Im}$ 是序列 $\Im$ 在 V_d 上的投影序列. 则

(1). $\overline{\Im}$ 是 $\Box Next$ 的一个 *LTL* 模型;

(2). $\overline{\Im}$ 是 *clock_constra* 的一个 *LTL* 模型;

(3). $\overline{\Im}$ 是 *liveness* 的一个 *LTL* 模型。

证明:

(1). 由引理 4.7 和引理 4.8 立即可得。

(2). 由时钟变量的定义知 $\mathcal{J}$ 是 $(\bigwedge_{c\in C}(c=0\wedge c'=0))\wedge(\Box\bigwedge_{c\in C}(c=c'\vee c'=0))$ 的 LTLC- 模型, 于是显然有 $\overline{\Im}$ 是 $(\bigwedge_{c\in C}(c=0\wedge c'=0))\wedge(\bigwedge_{c,d\in C}(\Delta_{cd}=0\wedge\Delta'_{cd}=0))\wedge(\Box\bigwedge_{c\in C}(c=c'\vee c'=0))$ 的 LTL 模型。

对于任意 $c,d\in C$ 及任意 $i\in\mathcal{N}$.

(i). 若 $\overline{\mathcal{J}(t_i)}(c')=0$, 则 $\mathcal{J}(t_i)(c')=0$, 于是 $\overline{\mathcal{J}(t_i)}(\Delta'_{cd})=[\mathcal{J}(t_i)(c')-\mathcal{J}(t_i)(d')]_n=-[\mathcal{J}(t_i)(d')]_n=-\overline{\mathcal{J}(t_i)}(d')$.

(ii). 若 $\overline{\mathcal{J}(t_i)}(d')=0$, 类似地可得 $\overline{\mathcal{J}(t_i)}(\Delta'_{cd})=\overline{\mathcal{J}(t_i)}(c')$.

(iii). 若 $\overline{\mathcal{J}(t_i)}(c')=\overline{\mathcal{J}(t_i)}(c)$ 且 $\overline{\mathcal{J}(t_i)}(d')=\overline{\mathcal{J}(t_i)}(d)$, 则必有 $\mathcal{J}(t_i)(c')=\mathcal{J}(t_i)(c)$ 且 $\mathcal{J}(t_i)(d')=\mathcal{J}(t_i)(d)$ (否则若, 例如有 $\mathcal{J}(t_i)(c')\neq\mathcal{J}(t_i)(c)$, 则 $\mathcal{J}(t_i)(c')=0$, 于是由 $\overline{\mathcal{J}(t_i)}(c)=\overline{\mathcal{J}(t_i)}(c')=0$ 得 $\mathcal{J}(t_i)(c)=0=\mathcal{J}(t_i)(c')$, 这又与 $\mathcal{J}(t_i)(c')\neq\mathcal{J}(t_i)(c)$ 矛盾), 于是 $\overline{\mathcal{J}(t_i)}(\Delta'_{cd})=[\mathcal{J}(t_i)(c')-\mathcal{J}(t_i)(d')]_n=[\mathcal{J}(t_i)(c)-\mathcal{J}(t_i)(d)]_n=\overline{\mathcal{J}(t_i)}(\Delta_{cd})$.

于是 $\overline{\Im}$ 是 $\Box(\bigwedge_{c,d\in C}((c'=c\wedge d'=d\Rightarrow\Delta'_{cd}=\Delta_{cd})\wedge(c'=0\Rightarrow\Delta'_{cd}=-d')\wedge(d'=0\Rightarrow\Delta'_{cd}=c')))$ 的一个 LTL 模型。

(3). 对于 $c\in C$, 令 $Z_c::=\{\,i\mid i\in\mathcal{N}$ 且$\overline{\mathcal{J}(t_i)}(c')=0\,\}$ 。

(i). 若 Z_c 是无穷集合, 则 $\overline{\Im}$ 是 $\Box\Diamond(c'=0)$ 的 LTL 模型.

(ii). 若 Z_c 是有穷集合, 设 k 是 Z_c 中的最大元, 且设时钟变量 c 在 $\mathcal{J}$ 下的解释为时钟函数 f_c , 由于序列 $\{t_i\}_{i\in\mathcal{N}}$ 中包含了 f_c 的所有不连续点 (cf. 定义 4.12) , 故 f_c 在无穷区间 (t_k,∞) 上没有不连续点, 于是由时钟函数的定义 (定义 3.3) 知对于 $s\in(t_k,\infty)$ 有 $f_c(s)\geq s-t_k$, 又由于 $\{t_i\}_{i\in\mathcal{N}}$ 是一个发散序列, 于是存在正整数 N_0 使得当 $i\geq N_0$ 时有 $t_i\geq t_k+n+1$, 于是对于任意 $i\geq N_0$ 有 $\overline{\mathcal{J}(t_i)}(c)=[\mathcal{J}(t_i)(c)]_n=[f_c(t_i)]_n\geq[t_i-t_k]_n\geq[t_k+n+1-t_k]_n=[n+1]_n=2n+1>2n$, 于是 $\overline{\Im}$ 是 $\Box\Diamond(c>2n)$ 的 LTL 模型。

由以上的 (i)(ii) 便知 $\overline{\Im}$ 是 liveness 的一个 LTL 模型。

□.

定义 4.19 对于 $\varphi\in Form_R(V,n)$, 定义 φ 所对应的 *(V_d* 上的*)LTL* 公式 $\widehat{\varphi}::=\overline{\varphi}\wedge clock\_constra\wedge liveness\wedge\Box Next$.

引理 4.10 对于 $\varphi \in Form_R(V,n)$, 若 φ 是 $LTLC$ 可满足的, 则 $\widehat{\varphi}$ 是 LTL 可满足的。

证明: 设 $\mathcal{J}$ 是 φ 的一个 LTLC 模型, 并设 $\Im =< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是模型 $\mathcal{J}$ 的一个执行序列; 则由引理 4.6 知 $\overline{\Im}$ 是 $\overline{\varphi}$ 的一个 LTL 模型, 又由引理 4.9 知 $\overline{\Im}$ 是 clock_constra 、 liveness 和 $\Box Next$ 的一个 LTL 模型, 于是 $\overline{\Im}$ 是 $\widehat{\varphi}$ 的一个 LTL 模型。

□.

定义 4.20 对于固定的变量集 $V = B \cup C$ 和正整数 n, 设 $\Pi ::=< \pi_0, \pi_1, \pi_2, \ldots >$ 是 $clock\_constra \wedge liveness \wedge \Box Next$ 的一个 *(*定义在变量集 V_d 上的*)LTL* 模型; 归纳定义非负实数序列 $\{t_i\}_{i\in\mathcal{N}}$ 和 V 上的状态序列 $\{\sigma_i\}_{i\in\mathcal{N}}$ 如下:

- 步 *1.* 定义 $t_0 ::= 0$; 对于任意 $v \in B \cup B'$ 定义 $\sigma_0(v) ::= \pi_0(v)$; 对于任意 $v \in C \cup C'$ 定义 $\sigma_0(v) ::= 0$.

- 步 *2.* 假设 t_k 和 σ_k 已经定义。令 $A_k ::= \{1\} \cup \{1- <\sigma_k(c')> \mid c \in C, \sigma_k(c') \leq n$ 且 $\sigma_k(c') \notin \{0,1,2,\ldots,n\}\}$, $G_k ::= \{c \mid c \in C$ 且 $\pi_{k+1}(c) \in \{2,4,6,\ldots,2n\}\}$.

  这里 $<\sigma_k(c')>$ 表示非负实数 $\sigma_k(c')$ 的小数部分。 A_k 是一个非空的有穷正实数集合, 我们以 $MinA_k$ 记 A_k 中的最小元, 它是一个不大于 *1* 的正实数。

  若 G_k 是一个空集, 令 $\delta_k ::= \frac{1}{2} MinA_k$; 否则任取一 $c_1 \in G_k$, 令 $\delta_k ::= 1-<\sigma_k(c_1')>$ 。利用 σ_k, π_{k+1} 及 δ_k 可定义 t_{k+1} 和 σ_{k+1} 如下:

  (1). $t_{k+1} ::= t_k + \delta_k$;

  (2). 对于任意 $p \in B$, 定义 $\sigma_{k+1}(p) ::= \pi_{k+1}(p)$, $\sigma_{k+1}(p') ::= \pi_{k+1}(p')$;

  (3). 对于任意 $c \in C$, 定义 $\sigma_{k+1}(c) ::= \sigma_k(c') + \delta_k$; 当 $\pi_{k+1}(c') = 0$ 时定义 $\sigma_{k+1}(c') ::= 0$, 否则定义 $\sigma_{k+1}(c') ::= \sigma_k(c') + \delta_k$.

引理 4.11 定义 *4.20* 中定义的非负实数序列 $\{t_i\}_{i\in\mathcal{N}}$ 和 V 上的状态序列 $\{\sigma_i\}_{i\in\mathcal{N}}$ 具有下列性质.

(1). 对于任意 $i \in \mathcal{N}$ 有 $\overline{\sigma_i} = \pi_i$;

(2). 对于任意 $c \in C$ 及任意 $i > j$ 有 $0 \leq \sigma_i(c') \leq \sigma_i(c) \leq \sigma_j(c') + (t_i - t_j) \leq \sigma_j(c) + (t_i - t_j)$;

(3). $\{t_i\}_{i\in\mathcal{N}}$ 是一个时间序列。

证明:

(1). 对 i 进行归纳。当 $i=0$ 时显然。假设当 $i=k$ 时成立, 现看 $i=k+1$ 的情况。

情况 **1.** 当 $\pi_k(boundary)=1$ 时。

对任意 $c\in C$, *(i).* 若 $\pi_k(c')\in\{0,2,4,\ldots,2n\}$, 则 $\pi_k(boundary(c))=1$, 由于 Π 是 $\Box Next$ 的模型, 故由 $Next$ 和 $next1$ 的定义知此时有 $\pi_{k+1}(c)=\pi_k(c')+1$, 故有 $\pi_{k+1}(c)\notin\{0,2,4,\ldots,2n\}$; *(ii).* 若 $\pi_k(c')\notin\{0,2,4,\ldots,2n\}$, 则可得 $\pi_{k+1}(c)=\pi_k(c')$, 于是我们仍有 $\pi_{k+1}(c)\notin\{0,2,4,\ldots,2n\}$ 。这样我们得到了当 $\pi_k(boundary)=1$ 时 G_k 是一个空集。

由定义 *4.20* 知此时 $\delta_k=\frac{1}{2}MinA_k$, 下面证明对任意 $v\in V_d$ 有 $\overline{\sigma_{k+1}}(v)=\pi_{k+1}(v)$ (即 $\overline{\sigma_{k+1}}=\pi_{k+1}$)。

下面的 *(*甲*)(*乙*)* 分别给出了当 $v\in C$ 和 $v\in D$ 时的证明作为示例, 其它情况下的证明略。

(甲**).** 对于任意 $c\in C$.

(i). 若 $\pi_k(c')\in\{0,2,4,\ldots,2n\}$, 由前面证明知 $\pi_{k+1}(c)=\pi_k(c')+1$, 由归纳假设知 $\overline{\sigma_k}(c')=\pi_k(c')$, 于是 $\sigma_k(c')\in\{0,1,2,\ldots,n\}$, 再由 δ_k 是一个不大于 $\frac{1}{2}$ 的正实数知 $\overline{\sigma_{k+1}}(c)=[\sigma_{k+1}(c)]_n=[\sigma_k(c')+\delta_k]_n=2\sigma_k(c')+1=[\sigma_k(c')]_n+1=\overline{\sigma_k}(c')+1=\pi_k(c')+1=\pi_{k+1}(c)$.

(ii). 若 $\pi_k(c')\notin\{0,2,4,\ldots,2n\}$, 则由前面证明知 $\pi_{k+1}(c)=\pi_k(c')$, 且由归纳假设知 $\sigma_k(c')\notin\{0,1,2,\ldots,n\}$ 。若 $\sigma_k(c')>n$, 则 $\sigma_{k+1}(c)=\sigma_k(c')+\delta_k>n$, 于是 $\overline{\sigma_{k+1}}(c)=[\sigma_{k+1}(c)]_n=2n+1=\overline{\sigma_k}(c')=\pi_k(c')=\pi_{k+1}(c)$; 若 $\sigma_k(c')\le n$, 则 $1-<\sigma_k(c')>\in A_k$, 于是由 δ_k 定义知 $\delta_k<1-<\sigma_k(c')>$, 于是 $\lfloor\sigma_k(c')+\delta_k\rfloor=\lfloor\sigma_k(c')\rfloor$, 即 $\lfloor\sigma_{k+1}(c)\rfloor=\lfloor\sigma_k(c')\rfloor$, 于是 $\overline{\sigma_{k+1}}(c)=\overline{\sigma_k}(c')=\pi_k(c')=\pi_{k+1}(c)$.

(乙**).** 对于任意 $c,d\in C$, 由 $Next$ 的定义知 $\pi_{k+1}(\Delta_{cd})=\pi_k(\Delta'_{cd})$*(* 因为 $D^+=D'$ 成立*)* , 于是 $\overline{\sigma_{k+1}}(\Delta_{cd})=[\sigma_{k+1}(c)-\sigma_{k+1}(d)]_n=[\sigma_k(c')-\sigma_k(d')]_n=\overline{\sigma_k}(\Delta'_{cd})=\pi_k(\Delta'_{cd})=\pi_{k+1}(\Delta_{cd})$.

情况 **2.** 当 $\pi_k(discontin)=1$ 且 $\pi_k(boundary)=0$ 时。

对任意 $c\in C$, 由 $Next$ 的定义知 $\pi_{k+1}(c)=\pi_k(c')$, 再由 $\pi_k(boundary)=0$ 知 G_k 是一个空集。

由定义 *4.20* 知此时 $\delta_k=\frac{1}{2}MinA_k$, 下面证明对任意 $v\in V_d$ 有 $\overline{\sigma_{k+1}}(v)=\pi_{k+1}(v)$ (即 $\overline{\sigma_{k+1}}=\pi_{k+1}$)。

下面的 *(*丙*)(*丁*)* 给出了当 $v\in C$ 和 $v\in D'$ 时的证明作为示例, 其它情况下的证明略。

(丙). 对于任意 $c \in C$，由 $\pi_k(boundary) = 0$ 知 $\pi_k(c') \notin \{0, 2, 4, \ldots, 2n\}$, 此时的证明过程与上面的 *(甲 (ii))* 相同。

(丁). 对于任意 $c, d \in C$，由 *Next* 的定义知 $\pi_{k+1}(c') = \pi_k(c')$, $\pi_{k+1}(d') = \pi_k(d')$（由 $(C')^+ = C'$ 得到），于是 $\pi_{k+1}(c') \neq 0$, $\pi_{k+1}(d') \neq 0$，由定义 *4.20* 知 $\sigma_{k+1}(c') = \sigma_k(c') + \delta_k$, $\sigma_{k+1}(d') = \sigma_k(d') + \delta_k$, 于是 $\overline{\sigma_{k+1}}(\Delta'_{cd}) = [\sigma_{k+1}(c') - \sigma_{k+1}(d')]_n = [\sigma_k(c') - \sigma_k(d')]_n = \overline{\sigma_k}(\Delta'_{cd}) = \pi_k(\Delta'_{cd}) = \pi_{k+1}(\Delta'_{cd})$（最后一步由 $(D')^+ = D'$ 得到）。

情况 **3.** 当 $\pi_k(discontin) = 0$, $\pi_k(boundary) = 0$ 且 G_k 为空集时。

由于 G_k 为空集，故对于任意 $c \in C$ 有 $\pi_{k+1}(c) \notin \{2, 4, 6, \ldots, 2n\}$。由 $\pi_k(boundary) = 0$ 知 $\pi_k(c') \notin \{0, 2, 4, 6, \ldots, 2n\}$，又从 *Next* 的定义可得 $\pi_{k+1}(c) = \pi_k(c') + 1$ 或 $\pi_{k+1}(c) = \pi_k(c')$. 但 $\pi_{k+1}(c) = \pi_k(c') + 1$ 此时不可能成立，故只有 $\pi_{k+1}(c) = \pi_k(c')$ 才可能成立。以下的证明与情况 *2* 类似，在此从略。

情况 **4.** 当 $\pi_k(discontin) = 0$, $\pi_k(boundary) = 0$ 且 G_k 为非空集时。

任取一 $c_1 \in G_k$, 由于 $\pi_{k+1}(c_1) \in \{2, 4, 6, \ldots, 2n\}$, 但 $\pi_k(c'_1) \notin \{0, 2, 4, 6, \ldots, 2n\}$, 于是 $\pi_{k+1}(c_1) \neq \pi_k(c'_1)$, 于是由 *Next* 的定义可得 $\pi_k(critical(c_1)) = 1$。

下面证明对于任意 $c \in C$ 有：　$\pi_k(critical(c)) = 1$ 当且仅当 $\sigma_k(c') \leq n$ 且 $<\sigma_k(c')> = <\sigma_k(c'_1)>$.

- 当 $\pi_k(critical(c)) = 1$ 时，由 $critical(c)$ 的定义知 $\pi_k(c') \leq 2n$（即 $\sigma_k(c') \leq n$）且 $\pi_k(c') \leq \pi_k(c'_1) + \pi_k(\Delta'_{cc_1})$，由于 $\pi_k(critical(c_1)) = 1$，故还有 $\pi_k(c'_1) \leq \pi_k(c') + \pi_k(\Delta'_{c_1c})$. 而由归纳假设知 $\pi_k(\Delta'_{cc_1}) = \overline{\sigma_k}(\Delta'_{cc_1}) = [\sigma_k(c') - \sigma_k(c'_1)]_n = -[\sigma_k(c'_1) - \sigma_k(c')]_n = -\overline{\sigma_k}(\Delta'_{c_1c}) = -\pi_k(\Delta'_{c_1c})$, 于是 $\pi_k(c') = \pi_k(c'_1) + \pi_k(\Delta'_{cc_1})$，由于 $\pi_k(c') = \overline{\sigma_k}(c') = [\sigma_k(c')]_n = 2\lfloor\sigma_k(c')\rfloor + 1$ 且 $\pi_k(c'_1) = 2\lfloor\sigma_k(c'_1)\rfloor + 1$, 故 $2\lfloor\sigma_k(c')\rfloor = 2\lfloor\sigma_k(c'_1)\rfloor + [\sigma_k(c') - \sigma_k(c'_1)]_n$, 于是 $[\sigma_k(c') - \sigma_k(c'_1)]_n$ 是一个偶数，于是 $\sigma_k(c') - \sigma_k(c'_1)$ 是一个绝对值小于 n 的整数，这样就有 $<\sigma_k(c')> = <\sigma_k(c'_1)>$.
- 反之，当 $\sigma_k(c') \leq n$ 且 $<\sigma_k(c')> = <\sigma_k(c'_1)>$ 时，对任意的 $d \in C$，若 $\pi_k(d') \leq 2n$, 则由 $\pi_k(critical(c_1)) = 1$ 知 $\pi_k(c'_1) \leq \pi_k(d') + \pi_k(\Delta'_{c_1d})$，由于 $\pi_k(c'_1) = \overline{\sigma_k}(c'_1) = 2\lfloor\sigma_k(c'_1)\rfloor + 1 = 2\lfloor\sigma_k(c')\rfloor + 1 + [\sigma_k(c'_1) - \sigma_k(c')]_n = \overline{\sigma_k}(c') + \overline{\sigma_k}(\Delta'_{c_1c}) = \pi_k(c') + \pi_k(\Delta'_{c_1c})$ 且 $\pi_k(\Delta'_{c_1d}) = \overline{\sigma_k}(\Delta'_{c_1d}) = [\sigma_k(c'_1) - \sigma_k(d')]_n = [(\sigma_k(c') - \sigma_k(d')) + (\sigma_k(c'_1) - \sigma_k(c'))]_n = [\sigma_k(c') - \sigma_k(d')]_n + [\sigma_k(c'_1) - \sigma_k(c')]_n = \overline{\sigma_k}(\Delta'_{cd}) + \overline{\sigma_k}(\Delta'_{c_1c}) = \pi_k(\Delta'_{cd}) + \pi_k(\Delta'_{c_1c})$，故有 $\pi_k(c') \leq \pi_k(d') + \pi_k(\Delta'_{cd})$；又由于 $\sigma_k(c') \leq n$ 蕴涵 $\pi_k(c') \leq 2n$，故由 $critical(c)$ 的定义便知 $\pi_k(critical(c)) = 1$。

这样就证明了 $\pi_k(critical(c))=1$ 当且仅当 $\sigma_k(c') \leq n$ 且 $<\sigma_k(c')> = <\sigma_k(c'_1)>$ 。

由定义 *4.20* 知当情况 *4* 成立时， $\delta_k = 1 - <\sigma_k(c'_1)>$ 。下面证明对任意 $v \in V_d$ 有 $\overline{\sigma_{k+1}}(v) = \pi_{k+1}(v)$ 。

这里给出 $v \in C$ 和 $v \in C'$ 时的证明作为示例，其它情况下的证明略。

(戊). 对于任意 $c \in C$ 。

(i). 若 $\pi_k(critical(c)) = 1$, 则由 *Next* 的定义可得 $\pi_{k+1}(c) = \pi_k(c')+1$, 又由上面的结论知 $\sigma_k(c') \leq n$ 且 $<\sigma_k(c')> = <\sigma_k(c'_1)>$, 于是 $\overline{\sigma_{k+1}}(c) = [\sigma_{k+1}(c)]_n = [\sigma_k(c')+\delta_k]_n = [\sigma_k(c')+1-<\sigma_k(c'_1)>]_n = [\sigma_k(c')+1-<\sigma_k(c')>]_n = [\lfloor\sigma_k(c')\rfloor + 1]_n = 2(\lfloor\sigma_k(c')\rfloor + 1) = [\sigma_k(c')]_n + 1 = \overline{\sigma_k}(c') + 1 = \pi_k(c') + 1 = \pi_{k+1}(c)$ 。

(ii). 若 $\pi_k(critical(c)) = 0$ 且 $\sigma_k(c') \leq n$, 则由 *Next* 的定义可得 $\pi_{k+1}(c) = \pi_k(c')$ ，又由上面的结论知 $<\sigma_k(c')> \neq <\sigma_k(c'_1)>$ 。

但 $<\sigma_k(c')>$ 不可能大于 $<\sigma_k(c'_1)>$ ，因为若 $<\sigma_k(c')>$ 大于 $<\sigma_k(c'_1)>$ 则有 $\pi_k(c') + \pi_k(\Delta'_{c_1c}) = \overline{\sigma_k}(c') + \overline{\sigma_k}(\Delta'_{c_1c}) = [\sigma_k(c')]_n + [\sigma_k(c'_1) - \sigma_k(c')]_n = 2\lfloor\sigma_k(c')\rfloor+1+[\lfloor\sigma_k(c'_1)\rfloor + <\sigma_k(c'_1)> - \lfloor\sigma_k(c')\rfloor - <\sigma_k(c')>]_n = 2\lfloor\sigma_k(c')\rfloor+1+[\lfloor\sigma_k(c'_1)\rfloor - \lfloor\sigma_k(c')\rfloor - (<\sigma_k(c')> - <\sigma_k(c'_1)>)]_n = 2\lfloor\sigma_k(c')\rfloor+1+2(\lfloor\sigma_k(c'_1)\rfloor - \lfloor\sigma_k(c')\rfloor) - 1 = 2\lfloor\sigma_k(c'_1)\rfloor = [\sigma_k(c'_1)]_n - 1 = \overline{\sigma_k}(c'_1) - 1 = \pi_k(c'_1) - 1$ ，这与 $\pi_k(critical(c_1)) = 1$ 矛盾。

于是 $<\sigma_k(c')>$ 小于 $<\sigma_k(c'_1)>$ ，于是 $\overline{\sigma_{k+1}}(c) = [\sigma_{k+1}(c)]_n = [\sigma_k(c') + \delta_k]_n = [\sigma_k(c') + 1 - <\sigma_k(c'_1)>]_n = [\lfloor\sigma_k(c')\rfloor + <\sigma_k(c')> + 1 - <\sigma_k(c'_1)>]_n = [\lfloor\sigma_k(c')\rfloor + (1+<\sigma_k(c')> - <\sigma_k(c'_1)>)]_n = 2(\lfloor\sigma_k(c')\rfloor) + 1 = [\sigma_k(c')]_n = \overline{\sigma_k}(c') = \pi_k(c') = \pi_{k+1}(c)$

(iii). 若 $\pi_k(critical(c)) = 0$ 且 $\sigma_k(c') > n$, 则由 *Next* 的定义可得 $\pi_{k+1}(c) = \pi_k(c')$ ，于是 $\overline{\sigma_{k+1}}(c) = [\sigma_{k+1}(c)]_n = [\sigma_k(c') + \delta_k]_n = 2n + 1 = [\sigma_k(c')]_n = \overline{\sigma_k}(c') = \pi_k(c) = \pi_{k+1}(c)$.

(己). 对于任意 $c \in C$ 。

(i). 若 $\pi_{k+1}(c') = 0$, 则 $\sigma_{k+1}(c') = 0$, 于是 $\overline{\sigma_{k+1}}(c') = [\sigma_{k+1}(c')]_n = 0 = \pi_{k+1}(c')$.

(ii). 若 $\pi_{k+1}(c') \neq 0$, 由 *clock_constra* 的定义（因 $c' = c \vee c' = 0$ 永远成立）知 $\pi_{k+1}(c') = \pi_{k+1}(c)$ ，又由 *(戊)* 知 $\overline{\sigma_{k+1}}(c) = \pi_{k+1}(c)$, 于是 $\overline{\sigma_{k+1}}(c') = [\sigma_{k+1}(c')]_n = [\sigma_k(c') + \delta_k]_n = [\sigma_{k+1}(c)]_n = \overline{\sigma_{k+1}}(c) = \pi_{k+1}(c) = \pi_{k+1}(c')$

至此就完成了 *(1)* 的证明。

(2). 由定义 *4.20* 知对于任意 $k \in \mathcal{N}$ 及任意 $c \in C$ 有： $0 \leq \sigma_{k+1}(c') \leq \sigma_{k+1}(c) = \sigma_k(c') + t_{k+1} - t_k$. 反复应用此结论便可得到所要的结论。

(3). 要证 $\{t_i\}_{i\in\mathcal{N}}$ 是一个时间序列我们只须证明它是一个发散的序列。

用反证法。假设 $\{t_i\}_{i\in\mathcal{N}}$ 是一个收敛序列, 则存在正整数 N_0 使得对于任意 $i \geq j \geq N_0$ 有 $t_i - t_j < 0.2$.

令 $C_0 = \{\, c \mid c \in C$ 且存在无穷多个 k 使 $\pi_k(c') = 0 \,\}$,

(i). 对于 $c \in C_0$, 必存在正整数 N_c 使得 $N_c \geq N_0$ 且 $\pi_{N_c}(c') = 0$. 由于 $\overline{\sigma_{N_c}} = \pi_{N_c}$, 于是 $\sigma_{N_c}(c') = 0$ ， 于是当 $i \geq N_c$ 时有 $\sigma_i(c') \leq \sigma_i(c) \leq t_i - t_{N_c} < 0.2$.

(ii). 对于 $c \in C - C_0$, 由于 Π 是 $\Box\Diamond(c' = 0 \vee c > 2n)$ 的模型, 故必然存在正整数 N_c 使得 $\pi_{N_c}(c) > 2n$ 且对于任意 $i \geq N_c$ 有 $\pi_i(c') \neq 0$, 于是 $\sigma_{N_c}(c) > n$ 且对任意 $i \geq N_c$ 有 $\sigma_i(c') = \sigma_i(c) = \sigma_{N_c}(c') + t_i - t_{N_c} = \sigma_{N_c}(c) + t_i - t_{N_c} \geq \sigma_{N_c}(c) > n$.

由于 C 是一个有限集, 取 N_1 为 $\{N_0\} \cup \{N_c \mid c \in C\}$ 中的最大值, 则对任意 $i \geq N_1$ 及任意 $c \in C$ 有: 若 $c \in C_0$, 则 $\sigma_i(c') \leq \sigma_i(c) < 0.2$; 若 $c \in C - C_0$, 则 $\sigma_i(c') = \sigma_i(c) > n$.

情况 **1.** 若 C_0 是空集; 则对任意 $i \geq N_1$ 及任意 $c \in C$ 有 $\sigma_i(c') = \sigma_i(c) > n$ ， 即对于任意 $i \geq N_1$ 及任意 $c \in C$ 有 $\pi_i(c') = \pi_i(c) > 2n$ 。任取一大于 N_1 的正整数 k , 则由定义 *4.20* 知此时 G_k 是空集， A_k 是单元素集 $\{1\}$, 于是 $\delta_k = \frac{1}{2}MinA_k = 0.5$, 即 $t_{k+1} - t_k = 0.5$ ， 这与前面的 "对于任意 $i \geq j \geq N_0$ 有 $t_i - t_j < 0.2$" 矛盾。

情况 **2.** 若 C_0 是非空集; 任取 $c_1 \in C_0$, 由 C_0 的定义知存在大于 N_1 的正整数 k 使得 $\pi_k(c') = 0$ ， 于是 $\pi_k(boundary) = 1$, 由 (*1*) 中的证明知此时 $\delta_k = \frac{1}{2}MinA_k$ ， 由于对于任意 $c \in C$ 有： '若 $\sigma_k(c') \leq n$, 则 $\sigma_k(c') < 0.2$' ， 于是 A_k 中的数都大于 0.8 ， 于是 $\delta_k > 0.4$, 即 $t_{k+1} - t_k > 0.4$ ， 这也与前面的 '对于任意 $i \geq j \geq N_0$ 有 $t_i - t_j < 0.2$' 矛盾。

这样就得到了 $\{t_i\}_{i\in\mathcal{N}}$ 是一个发散的序列, 因此它是一个时间序列。

□.

引理 4.12 对于固定的变量集 V 和正整数 n, 设 $\Pi = < \pi_0, \pi_1, \pi_2, \ldots >$ 是 $clock\_constra \wedge liveness \wedge \Box Next$ 的一个 *(*定义在变量集 V_d 上的*)LTL* 模型, 则存在 V 上的一个 $LTLC$ 模型 $\mathcal{J}$ 及 $\mathcal{J}$ 的一个执行序列 $\Im$ 使得 $\overline{\Im} = \Pi$.

证明: 由定义 4.20 和引理 4.11 我们知道有一个时间序列 $\{t_i\}_{i\in\mathcal{N}}$ 及 V 上的状态序列 $\{\sigma_i\}_{i\in\mathcal{N}}$ 满足对于任意 $i \in \mathcal{N}$ 有 $\overline{\sigma_i} = \pi_i$ 且 $(\sigma_i + t_{i+1} - t_i) \rhd \sigma_{i+1}$.

(i). 对于 $p \in B$, 定义布尔值步函数 $f_p : \mathcal{R}^+ \mapsto \{0,1\}$ 如下：

$$f_p(t) ::= \begin{cases} \sigma_0(p) & \text{若 } t = 0 \\ \sigma_i(p') & \text{若 } t \in (t_i, t_{i+1}] \end{cases}$$

(ii). 对于 $c \in C$, 定义函数 $f_c : \mathcal{R}^+ \mapsto \mathcal{R}^+$ 如下：

$$f_c(t) ::= \begin{cases} 0 & \text{若 } t = 0 \\ \sigma_i(c') + t - t_i & \text{若 } t \in (t_i, t_{i+1}] \end{cases}$$

不难验证 f_c 是一个时钟函数 (定义 3.3)。

对于 $p \in B$, 以 f_p 作为 p 的解释; 对于 $c \in C$, 以 f_c 作为 c 的解释。则得到变量集 V 上的一个 LTLC- 模型 $\mathcal{J}$ 。不难看到 $\Im ::=< \mathcal{J}(t_0), \mathcal{J}(t_1), \mathcal{J}(t_2), \ldots >$ 是 $\mathcal{J}$ 的一个执行序列且 $\overline{\Im} = \Pi$.

□.

引理 4.13 对于 $\varphi \in Form_R(V,n)$, 若 $\widehat{\varphi}$ 是 *LTL* 可满足的, 则 φ 是 *LTLC* 可满足的。

证明: 设 Π 是 $\widehat{\varphi}$ 的一个 (定义在变量集 V_d 上的)LTL 模型, 由上面的引理 4.12 知存在 V 上的一个 LTLC 模型 $\mathcal{J}$ 及 $\mathcal{J}$ 的一个执行序列 $\Im$ 使得 $\overline{\Im} = \Pi$ 。由于 $\Pi \models_{ltl} \overline{\varphi}$, 由引理 4.6 知 $\mathcal{J} \models \varphi$, 于是 φ 是 LTLC 可满足的。

□.

由引理 4.10 和引理 4.13 立即可得：

引理 4.14 对于 $\varphi \in Form_R(V,n)$, φ 是 *LTLC-* 可满足的当且仅当 $\widehat{\varphi}$ 是 *LTL-* 可满足的。

由于 V_d 中的所有变量的取值域都是有限的 (不超出域 $\{-2n-1, -2n, \ldots, -2, -1, 0, 1, 2, \ldots, 2n+1\}$) , 故由定理 2.3 便知 $\widehat{\varphi}$ 的 (LTL-) 可满足性是可判定的. 这样 LTLC/R 公式的可满足性是可判定的。即有：

定理 4.1 对于任意 *LTLC/R* 公式 φ , φ 的 *(LTLC-)* 可满足性是算法可判定的。

对于任一给定的时间模块 M 及任一给定的性质 φ ,由于 $M \models \varphi$ 当且仅当 $\mathrm{TLF}(M) \wedge \neg\varphi$ 是不可满足的, 而根据定理 4.1 , $\mathrm{TLF}(M) \wedge \neg\varphi$ 的可满足性是算法可判定的, 即有：

定理 4.2 对于时间模块 M 和 *LTLC/R* 公式 φ , M 关于 φ 的模型检查问题是算法可判定的。

利用引理 4.14 和文献 [62] 中关于命题 LTL 的可满足性检查算法我们可实现一个工具来检查 LTLC/R 公式的可满足性, 此工具还可用来对时间模块进行非空性检查和模型检查, 并可对两个时间模块进行一致性检查。

4.4 小结

本章先定义了 LTLC 的一个子部分 LTLC/R，然后在第二节给出了实时系统的一种数学模型：时间模块，我们对时间模块定义了其所对应的 LTLC 公式，并利用 LTLC 公式的语义定义了时间模块的语义。此外本节还给出了利用 LTLC 的语义定义证明实时系统的性质的示例，之所以能这样利用语义定义来证明时间模块的性质主要是由于时间模块和它的性质是用同一语言来表示的。第三节证明了 LTLC/R 中的公式的可满足性是可判定的，该结果对于利用时序逻辑 LTLC 来进行实时系统的模型检查具有重要理论价值，因为它不但能做模型检查，而且还能用于实时系统的非空性检查以及两实时系统之间的一致性检查等。

第五章　　混成系统

5.1　基本概念

混成系统 (hybrid system)[71, 77, 78, 96] 是一种既包含离散量又包含连续量的计算系统，数控系统等一些与其外部连续变化的物理环境不断交互的嵌入式系统就是其典型的代表。由于这些系统在工业及国防领域有着重要而广泛的应用，故这些系统的安全性和可靠性近年来成为人们关心的一个焦点，进而对这些系统进行形式化分析也成为人们非常感兴趣的课题。

下面是一个很简单的混成系统的例子。

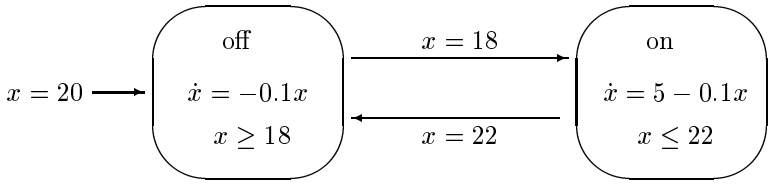

图 5.1 Thermostat

例 5.1. [温控器 [6, 46]] 图 5.1 中用混成自动机表示的混成系统模拟一个温控系统。当加热器的电源关闭时，房间温度 (记为 x) 按照微分方程 $\dot{x} = -0.1x$ 而降低；当电源打开时，温度按照方程 $\dot{x} = 5 - 0.1x$ 而升高。起初，温度是 20 度且电源是关的，当温度降到 18 度时电源将打开，但当温度升到 22 度时电源将关闭。如此这般，始终保持温度在 18 至 22 度之间。

本章中讨论的混成系统具有有穷个变量 (布尔型变量和柔性变量) 和有穷个控制状态，每个控制状态称为混成系统的一个顶点 (vertex) (上例中的混成系统由两个顶点，分别标为 on 和 off)，系统在每个顶点内可根据约束条件 (称为流不变量，如上例中的 $x \geq 18$ 和 $x \leq 22$) 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，但柔性变量的值可依照给定的微分方程而连续变化（如上例中的 x 在顶点 'off' 内按照方程 $\dot{x} = -0.1x$ 来改变）。系统在一个顶点内的一段连续变化称为一个流 (flow) 转换。流转换的持续时间是一个正实数（也可以是无穷）。当系统从一个顶点转换到另一个顶点时，布尔型变量和柔性变量均需重置其值，系统从一个顶点到另一个顶点的这个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，它的持续时间为 0 。

在本章中我们用 LTLC 的一个子语言 LTLC/H 来刻画和研究混成系统。LTLC/H 中只包含两类变量：布尔型变量和柔性变量。布尔型变量主要用来表示系统所处的不同控制状态（如可用 $p = 1$ 和 $p = 0$ 来分别表示例 5.1 中的控制状态 on 和 off ）；柔性变量几乎可用来表示系统中的任何量，它既可以表示连续量（这里连续量是指取值域为不可数集的分段连续变化的量，如上例中的房间温度以及上一章中的时钟变量等），也可以表示离散量（这里离散量是指取值域为可数集的变量，如整型变量等），

本章中所说的表达式是指由下面规则生成的一个 LTLC 项 (cf. 定义 3.7)

$e ::= r \,|\, x \,|\, (-e) \,|\, (e_1 + e_2) \,|\, (e_1 * e_2)$

这里 r 是一个有理数， x 是一个柔性变量。

一个 LTLC/H- 公式是指由下面规则生成的一个 LTLC 公式 φ:

$\varphi ::= p \,|\, p' \,|\, (x \le r) \,|\, (x = r) \,|\, (x' \le r) \,|\, (x' = r) \,|\, (x' = x) \,|\, (\dot{x} \le e) \,|\, (\dot{x} = e) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\Box\varphi) \,|\, (\varphi_1 \mathcal{U} \varphi_2)$.

这里 p 是一个布尔型变量， x 是一个柔性变量， r 是一个有理数， e 是一个表达式。

LTLC/H- 公式的语义模型的定义与 LTLC 公式的相同，只是去掉了在 LTLC/H 中不出现的成分的解释。为了叙述方便起见，我们定义几种特殊形式的 LTLC/H- 公式。

一个 LTLC/H- 状态公式是指由如下规则形成的一个 LTLC/H- 公式 φ ：

$\varphi ::= p \,|\, p' \,|\, (x \le r) \,|\, (x = r) \,|\, (x' \le r) \,|\, (x' = r) \,|\, (x' = x) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

LTLC/H- 当前状态公式是指由如下规则形成的一个 LTLC/H- 公式 φ ：

$\varphi ::= p \,|\, (x \le r) \,|\, (x = r) \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2)$

（本章所说的）微分公式是指由如下规则形成的一个 LTLC/H- 公式 φ ：

$\varphi ::= (\dot{x} \le e) \,|\, (\dot{x} = e) \,|\, (\dot{x} \ge e) \,|\, (\varphi_1 \wedge \varphi_2)$

在以上规则中 p 是一个布尔型变量， x 是一个柔性变量， r 是一个有理数， e 是一个表达式。

本章中的顶点 (vertex) 公式和新顶点 (new_vertex) 公式的含义与上章相同。

5.2 混成模块

我们以 混成模块(hybrid modules) 来模拟混成系统。一个混成模块 M 表示一个与其外部环境交互的混成系统。每个模块只能有有限个变量。其中某些变量的值只能被模块自身所改变，不能被环境改变；而另一些变量的值只能被环境所改变，不能被模块改变。依此将每个模块的变量分为二类，前一种称为控制变量，后一种称为外部变量。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。

每个混成模块主要由有限个转换（ transition ）所组成。控制变量的值是通过转换的作用来改变的。每个混成模块有两种类型的转换：跳跃转换和流转换。跳跃转换的发生不占用时间段，在其作用的瞬间（零时段内）它可改变所有控制变量（布尔型变量或柔性变量）的值。但每个流转换的发生具有一个正的时间段，在其作用的这个时段内，布尔型控制变量的值保持不变，柔性控制变量的值随时间流逝而连续改变。通常柔性控制变量的值的这种改变要满足一个称为流不变量（ flow invariant ）的 LTLC/H- 当前状态公式。在任意一个有限时间段中，跳跃转换只能发生有限次，相邻发生的两次跳跃转换之间为一个流转换的作用时段。

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 $vertex$ 是一个顶点公式，用于表示跳跃转换发生时系统所处的顶点；$guard$ 是一个 LTLC/H- 状态公式，它是跳跃转换的使能条件（ enabling condition ），它用于表示跳跃转换发生时系统的变量（包括控制变量以及外部变量）应满足的约束条件；new_vertex 是一个新顶点公式，它用于表示跳跃转换发生后系统将位于的新顶点；$assignment$ 是跳跃转换的命令部分，它相当于赋值语句，它给模块中每个柔性控制变量 x 赋一个新值 $(x' = r)$ 或维持其原值不变（ $x' = x$），它通常可表示为形如 '$x'_1 = e_1 \wedge x'_2 = e_2 \wedge \cdots \wedge x'_k = e_k$ ' 的一个 LTLC/H- 状态公式，这里 $x_1, x_2, \ldots, x_k$ 是该跳跃转换所在模块的所有柔性控制变量，对于任意 $1 \leq i \leq k$ 有 e_i 或者是一个常元 r 或者是变元 x_i(注：在 LTLC 中，跳跃转换的 guard 和 assignment 部分本来还可定义的更一般一些和更复杂一些，但上面的简单形式对于本文的讨论已经够用了，故为了简化定义过程，此处定义的 guard 和 assignment 的形式比较简单）。另外为讨论方便起见，我们假定每个跳跃转换在其作用时，至少改变了一个控制变量（无论布尔型变量或柔性变量）的值，即它不能使所有控制变量的值保持不变。

流转换通常被写成下面的形式： '$vertex \rightarrow diff \wedge invariant$' 。其中 '$vertex$' 是一个顶点公式，它表示流转换作用时所在的顶点；$invariant$ 是流转换的不变量部分，它是一个 LTLC/H- 当前状态公式，用于表示在该转换作用的时间段内模块的柔性变量和外部变量必须满足的约束条件；$diff$ 是一个微分公式，用于表示当系统位于该顶点时，系统中的所有柔性控制变量的改变所依照的微分方程，本来这个微分方程可以很复杂，但由于本章

只考虑比较简单的情况，故此处限制微分方程的形式为微分公式这样的简单形式（当然形式更复杂些的微分方程在 LTLC 中也是可以定义的）。

一般而言，一个混成模块具有如下的形式：

module	$module\_name$
external	$\{variable\_name\colon type\}^*$
controlled	$\{variable\_name\colon type\}^*$
init	$init\_cond$
jump	$\{vertex \wedge guard \to new\_vextex \wedge assignment\}^*$
flow	$\{vertex \to diff \wedge invariant\}^*$

这里 $type$ 表示变量的类型，变量的类型可以是 $boolean$（布尔型）或 $real$（实型）（此处规定所有柔性变量的类型为实型）；$init\_cond$ 是模块的初始条件，它是一个当前状态公式，用于表示模块在初始时刻 $t = 0$ 时其控制变量所应满足的条件。

注：混成模块的不同流转换所处的顶点应是互斥的，即如果 $vertex_1 \to diff_1 \wedge invariant_1$ 和 $vertex_2 \to diff_2 \wedge invariant_2$ 是某一个混成模块的两个不同流转换，则 $vertex_1 \wedge vertex_2$ 是一个永假的公式。另外我们通常还要求流转换部分是完全的，即若 $\{\, vertex_k \to diff_k \wedge invariant_k \,\}_{k\le\ell}$ 是一个混成模块的所有流转换，则 $\bigvee_{k\le\ell} vertex_k$ 应包括了混成模块可能位于的所有顶点。

说明：混成模块中的变量的类型有布尔型和实型共两种。但为了书写方便我们还可使用有穷枚举型变量（如下一节中的例 5.2 中的变量 pc 和例 5.3 中的变量 $pc2$），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。有关有穷枚举型的使用与上章规定相同。

例 5.1(续). 图 5.1 中的混成系统可表示为如下的一个混成模块，其中的变量 p 用来表示加热器电源的开关状态。

module	Thermostat
controlled	$p : boolean$;
	$x : real$
init	$p = \text{off} \wedge x = 20$
jump	$p = \text{off} \wedge x = 18 \to p' = \text{on} \wedge x' = x$;
	$p = \text{on} \wedge x = 22 \to p' = \text{off} \wedge x' = x$
flow	$p = \text{off} \to \dot{x} = -0.1x \wedge x \ge 18$;
	$p = \text{on} \to \dot{x} = 5 - 0.1x \wedge x \le 22$

注：上面转换中出现的符号 $\to$ 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，下面我们给出混成模块对应的 LTLC 公式。并利用语言 LTLC 给出混成模块的语义。

对于跳跃转换 $\alpha: vertex \wedge guard \rightarrow new\_vextex \wedge assignment$, 称 LTLC- 公式 $vertex \wedge guard \wedge new\_vextex \wedge assignment$ 为转换 α 所对应的时序逻辑公式，记作 $\mathrm{TLF}(\alpha)$ 。对于流转换 $\beta: vertex \rightarrow diff \wedge invariant$, 称 LTLC- 公式 $vertex \wedge diff \wedge invariant$ 为流转换 β 所对应的时序逻辑公式，记作 $\mathrm{TLF}(\beta)$ 。

定义 5.1 设 M 是一个混成模块， $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ 是 M 的所有跳跃转换， $\beta_0, \beta_1, \ldots, \beta_{l-1}$ 是 M 的所有流转换。我们称 *LTLC-* 公式 $init\_cond \wedge (\Box(V_c = V_c' \vee \bigvee_{j<n} TLF(\alpha_j))) \wedge \Box \bigvee_{k<l} TLF(\beta_k)$ 为混成模块 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 的所有控制变量的集合 *(*即集合 $ctr(M)$*)* ， $V_c = V_c'$ 是时序公式 $\bigwedge_{v \in V_c}(v' = v)$ 的缩写形式。以下我们以 $TLF(M)$ 来表示这个对应于模块 M 的时序逻辑公式。

例 5.1(续). 混成模块 Thermostat 所对应的时序逻辑公式 $\mathrm{TLF}(\mathrm{Thermostat}) = ((p = 0 \wedge x = 20) \wedge \Box((p = 0 \wedge x = 18 \wedge p' = 1 \wedge x' = x) \vee (p = 1 \wedge x = 22 \wedge p' = 0 \wedge x' = x) \vee (p' = p \wedge x' = x)) \wedge \Box((p = 0 \wedge \dot{x} = -0.1x \wedge x \geq 18) \vee (p = 1 \wedge \dot{x} = 5 - 0.1x \wedge x \leq 22)))$.

我们将混成模块 M 等同于它所对应的时序逻辑公式 $\mathrm{TLF}(M)$, 并将 $\mathrm{TLF}(M)$ 的语义模型作为 M 的语义模型。

定义 5.2 设 M 是一个混成模块， φ 是一个 *(LTLC-)* 公式。如果 $TLF(M) \models \varphi$, 则称 φ 是 M 的一个性质，记作 $M \models \varphi$ 。

由上面这个定义和定义 3.13 可知，要想证明模块 M 具有性质 φ，只需证明 φ 是 $\mathrm{TLF}(M)$ 的逻辑结论.

定义 5.3 称模块 M_1 和 M_2 是相容的，若 $ctr(M_1) \cap ctr(M_2) = \emptyset$ 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 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 来表示这 n 个模块的并行复合。并称公式 $\mathrm{TLF}(M_1) \wedge \mathrm{TLF}(M_2) \wedge \cdots \wedge \mathrm{TLF}(M_n)$ 为复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 所对应的时序逻辑公式。时序公式 $\mathrm{TLF}(M_1) \wedge \mathrm{TLF}(M_2) \wedge \cdots \wedge \mathrm{TLF}(M_n)$ 的语义模型被看作为是复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 的语义模型。

定义 5.4 对于 *LTLC-* 公式 φ ，若 $TLF(M_1) \wedge TLF(M_2) \wedge \cdots \wedge TLF(M_n) \models \varphi$ ，则称 φ 是复合模块 $[M_1 \parallel M_2 \parallel \cdots \parallel M_n]$ 的一个性质。

5.3　使用 LTLC 验证混成系统的性质

本节我们通过两个例子来说明如何利用定义 5.2 和定义 5.4 来证明混成系统的性质.

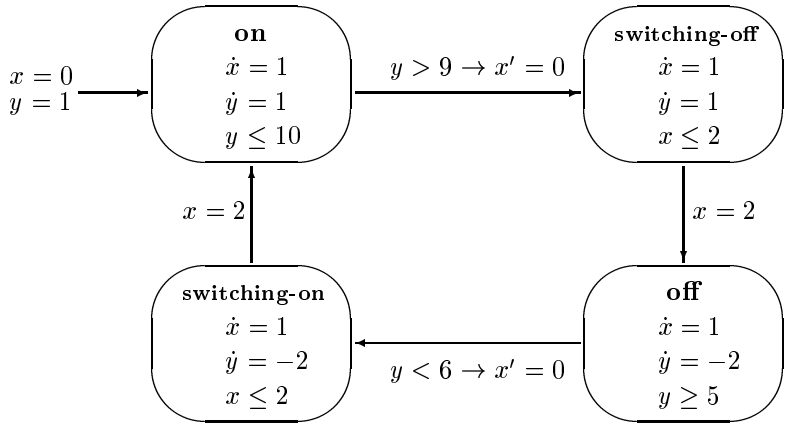

图 5.2: Water-level monitor

例 5.2. [水面监控系统 [6, 79]] 图 5.2 中用混成自动机表示的混成系统模拟一个水面监控系统。监控系统不断监测水箱中水面的高度，并将水泵打开或关闭。当水泵关闭时，水面高度（记为 y ）每秒下降 2 英寸；当水泵打开时，水面高度每秒上升 1 英寸。开始时，水面高度为 1 英寸且水泵是打开的。当水面高度超过 9 英寸时（ $y>9$ ），监控器就可以发出关闭水泵的信号，但此信号最晚也不能在水面高度达到 10 英寸之后才发出，也就是说最晚也必须在 $y=10$ 时发出关闭水泵的信号；类似地，当水面高度低于 6 英寸时，监控器就可以发出打开水泵的信号，但打开水泵信号最晚可在水面高度降到 5 英寸时才发出，即不能晚于 $y=5$ 时发出. 可是，从监控器发出 ‘打开水泵’ 或 ‘关闭水泵’ 的信号到水泵真正 ‘打开’ 或 ‘关闭’ 之间有一个 2 秒钟的延迟期。在延迟期，水泵仍保持原来的开或关状态。这样（如图 5.2 ），监控系统就有四个不同的控制状态： on, switching-off, off 及 switching-on 。在 on 和 witching-off 这两个阶段水泵是打开的，在 off 和 witching-on 两个阶段水泵是关闭的。图 5.2 中的变量 x 用来记录从某个时刻起所流逝的时间。这个混成系统可用下面的混成模块 *Monitor* 来表示。

module　*Monitor*
controlled
　pc : {on, off, switching-on, switching-off} ;
　x, y : *real*
init　$pc = \text{on} \wedge x = 0 \wedge y = 1$

jump

$pc = \text{on} \wedge y > 9 \rightarrow (pc' = \text{switching-off} \wedge x' = 0 \wedge y' = y)$;
$pc = \text{switching-off} \wedge x = 2 \rightarrow (pc' = \text{off} \wedge x' = x \wedge y' = y)$;
$pc = \text{off} \wedge y < 6 \rightarrow (pc' = \text{switching-on} \wedge x' = 0 \wedge y' = y)$;
$pc = \text{switching-on} \wedge x = 2 \rightarrow (pc' = \text{on} \wedge x' = x \wedge y' = y)$

flow

$pc = \text{on} \rightarrow \dot{x} = 1 \wedge \dot{y} = 1 \wedge y \leq 10$;
$pc = \text{switching-off} \rightarrow \dot{x} = 1 \wedge \dot{y} = 1 \wedge x \leq 2$;
$pc = \text{off} \rightarrow \dot{x} = 1 \wedge \dot{y} = -2 \wedge y \geq 5$;
$pc = \text{switching-on} \rightarrow (\dot{x} = 1 \wedge \dot{y} = -2 \wedge x \leq 2)$

由定义 5.1 知， $\text{TLF}(Monitor) = ((pc = 0 \wedge x = 0 \wedge y = 1) \wedge \Box((pc = 0 \wedge y > 9 \wedge pc' = 1 \wedge x' = 0 \wedge y' = y) \vee (pc = 1 \wedge x = 2 \wedge pc' = 2 \wedge x' = x \wedge y' = y) \vee (pc = 2 \wedge y < 6 \wedge pc' = 3 \wedge x' = 0 \wedge y' = y) \vee (pc = 3 \wedge x = 2 \wedge pc' = 0 \wedge x' = x \wedge y' = y) \vee (pc' = pc \wedge x' = x \wedge y' = y)) \wedge \Box((pc = 0 \wedge \dot{x} = 1 \wedge \dot{y} = 1 \wedge y \leq 10) \vee (pc = 1 \wedge \dot{x} = 1 \wedge \dot{y} = 1 \wedge x \leq 2) \vee (pc = 2 \wedge \dot{x} = 1 \wedge \dot{y} = -2 \wedge y \geq 5) \vee (pc = 3 \wedge \dot{x} = 1 \wedge \dot{y} = -2 \wedge x \leq 2)))$.

现在我们开始证明模块 *Monitor* 满足 $\Box(y \leq 12 \wedge y \geq 1)$ 。

设 $\mathcal{J}$ 是 $\text{TLF}(Monitor)$ 的一个模型，且 f_{pc} 、 f_x 和 f_y 分别是变量 pc 、 x 和 y 在模型 $\mathcal{J}$ 下的解释。

由于 $\Box((pc = 0 \wedge y > 9 \wedge pc' = 1 \wedge x' = 0 \wedge y' = y) \vee (pc = 1 \wedge x = 2 \wedge pc' = 2 \wedge x' = x \wedge y' = y) \vee (pc = 2 \wedge y < 6 \wedge pc' = 3 \wedge x' = 0 \wedge y' = y) \vee (pc = 3 \wedge x = 2 \wedge pc' = 0 \wedge x' = x \wedge y' = y) \vee (pc' = pc \wedge x' = x \wedge y' = y))$ 蕴涵 $\Box(y' = y)$ 。故 f_y 是 $\mathcal{R}^+$ 上的连续函数。

又 f_{pc} 在 $\mathcal{R}^+$ 上是正规的和左连续的，故存在无穷序列 $a_0, a_1, a_2, \ldots, a_n, \ldots$ 使得：

1. $0 = a_0 < a_1 < a_2 < \ldots < a_i < \ldots$;
2. 对任意 $N \in \mathcal{N}$, 存在 $i \in \mathcal{N}$ 使得 $a_i > N$;
3. 对任意 $i \in \mathcal{N}$ 函数 f_{pc} 在 $(a_i, a_{i+1}]$ 上连续。

不失一般性，我们可假定 $a_1, a_2, \ldots, a_n, \ldots$ 恰是函数 f_{pc} 的所有非零不连续点。

由于 $\text{TLF}(Monitor)$ 蕴涵 $pc = 0 \wedge \Box((pc = 0 \wedge pc' = 1) \vee (pc = 1 \wedge pc' = 2) \vee (pc = 2 \wedge pc' = 3) \vee (pc = 3 \wedge pc' = 0) \vee (pc' = pc))$, 而由后者可知函数 f_{pc} 只有四种不连续点。在这些不连续点处， f_{pc} 的值分别由 0, 1, 2, 3 改变为 1, 2, 3, 0.

施归纳法于非负整数 i 可得如下结论.

$$f_{pc}(t)=\begin{cases}0 & \text{若 } t\in(a_{4i},a_{4i+1}],\\ 1 & \text{若 } t\in(a_{4i+1},a_{4i+2}],\\ 2 & \text{若 } t\in(a_{4i+2},a_{4i+3}],\\ 3 & \text{若 } t\in(a_{4i+3},a_{4i+4}].\end{cases}$$

由于 f_x 在区间 $(a_{4i+1},a_{4i+2}]$ 和 $(a_{4i+3},a_{4i+4}]$ 的导数为 1，且值从 0 变为 2，故 $a_{4i+2}=a_{4i+1}+2$ 且 $a_{4i+4}=a_{4i+3}+2$。于是对任意 $t\in(a_{4i+1},a_{4i+2}]$ 有 $f_y(t)\le f_y(a_{4i+1})+2$；对任意 $t\in(a_{4i+3},a_{4i+4}]$ 有 $f_y(t)\ge f_y(a_{4i+3})-4$。

再次施归纳法于非负整数 i 可得：

$$f_y(t)\in\begin{cases}[1,10] & \text{若 } t\in(a_{4i},\ a_{4i+1}],\\ (9,12] & \text{若 } t\in(a_{4i+1},a_{4i+2}],\\ [5,12] & \text{若 } t\in(a_{4i+2},a_{4i+3}],\\ [1,6) & \text{若 } t\in(a_{4i+3},a_{4i+4}].\end{cases}$$

这样就得到 $\mathcal{J}$ 是 $\Box(y\le 12\wedge y\ge 1)$ 的一个模型，因 $\mathcal{J}$ 是模块 *Monitor* 的任一模型，故 $\Box(y\le 12\wedge y\ge 1)$ 是 *Monitor* 的一个性质。

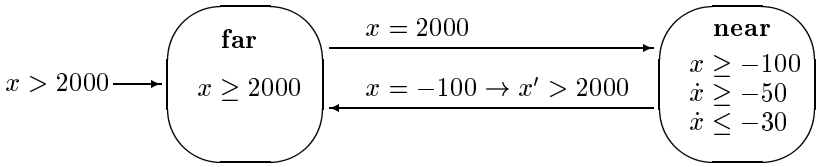

图 5.3 Train

例 5.3.[Railroad gate control [19, 46]] 图 5.3 中的混成自动机模拟一个环形铁路上的火车，该段铁路上有一个道口，图 5.4 中的混成自动机模拟这样的一个道口控制器。变量 x 表示火车距道口的距离，变量 y 表示道口横杆与地面的角度。初始时，火车距道口距离大于 2000 米 $(x>2000)$ 且道口是打开的 $(y=90)$。当火车行驶至距道口 1000 米处时，位于此处的一个传感器将探测出火车的到来，并向道口发出火车到来的信号。经过 5 秒钟的延迟后，道口横杆以每秒钟 9 度的速率由垂直角度改变为水平角度而将道口关闭。当火车过了道口到达距道口 100 米处时位于此处的另一个传感器将向道口发出火车离去的信号。又经过 5 秒钟的延迟后，道口横杆以每秒钟 9 度的速率由水平角度改变为垂直角度而将道口打开，如此这般。图 5.4 中的变量 z 用来记录从某时刻起所流逝的时间。

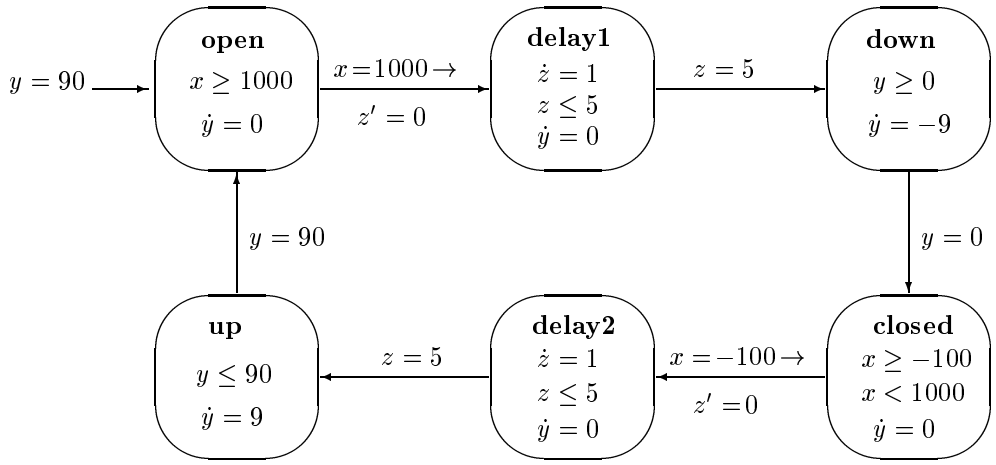

图 5.4 Gate

以上两个用混成自动机表示的混成系统可用混成模块表示如下：

module $Train$
controlled
$pc1 : \{\text{far, near}\}$;
$x : real$
init $pc1 = \text{far} \wedge x > 2000$
jump
$pc1 = \text{far} \wedge x = 2000 \rightarrow (pc1' = \text{near} \wedge x' = x)$;
$pc1 = \text{near} \wedge x = -100 \rightarrow (pc1' = \text{far} \wedge x' > 2000)$
flow
$pc1 = \text{far} \rightarrow x \geq 2000$;
$pc1 = \text{near} \rightarrow (x \geq -100 \wedge \dot{x} \geq -50 \wedge \dot{x} \leq -30)$

module $Gate$
external $x : real$
controlled
$pc2 : \{\text{open,closed, up,down,delay1,delay2}\}$;
$y, z : real$
init $pc2 = \text{open} \wedge y = 90$

jump

$pc2 = \text{open} \wedge x = 1000 \to (pc2' = \text{delay1} \wedge z' = 0 \wedge y' = y)$;
$pc2 = \text{delay1} \wedge z = 5 \to (pc2' = \text{down} \wedge z' = z \wedge y' = y)$;
$pc2 = \text{down} \wedge y = 0 \to (pc2' = \text{closed} \wedge z' = z \wedge y' = y)$;
$pc2 = \text{closed} \wedge x = -100 \to (pc2' = \text{delay2} \wedge z' = 0 \wedge y' = y)$;
$pc2 = \text{delay2} \wedge z = 5 \to (pc2' = \text{up} \wedge z' = z \wedge y' = y)$;
$pc2 = \text{up} \wedge y = 90 \to (pc2' = \text{open} \wedge z' = z \wedge y' = y)$

flow

$pc2 = \text{open} \to x \geq 1000 \wedge \dot{y} = 0$;
$pc2 = \text{delay1} \to z \leq 5 \wedge \dot{z} = 1 \wedge \dot{y} = 0$;
$pc2 = \text{down} \to y \geq 0 \wedge \dot{y} = -9$;
$pc2 = \text{closed} \to (x \geq -100 \wedge x < 1000 \wedge \dot{y} = 0)$;
$pc2 = \text{delay2} \to z \leq 5 \wedge \dot{z} = 1 \wedge \dot{y} = 0$;
$pc2 = \text{up} \to y \leq 90 \wedge \dot{y} = 9$

由定义 5.1, 我们知:

TLF(Train)=$((pc1 = 0 \wedge x > 2000) \wedge \Box((pc1 = 0 \wedge x = 2000 \wedge pc1' = 1 \wedge x' = x) \vee (pc1 = 1 \wedge x = -100 \wedge pc1' = 0 \wedge x' > 2000) \vee (pc1' = pc1 \wedge x' = x)) \wedge \Box((pc1 = 0 \wedge x \geq 2000) \vee (pc1 = 1 \wedge x \geq -100 \wedge \dot{x} \geq -50 \wedge \dot{x} \leq -30)))$,

TLF(Gate)= $((pc2 = 0 \wedge y = 90) \wedge \Box((pc2 = 0 \wedge x = 1000 \wedge pc2' = 1 \wedge z' = 0 \wedge y' = y) \vee (pc2 = 1 \wedge z = 5 \wedge pc2' = 2 \wedge z' = z \wedge y' = y) \vee (pc2 = 2 \wedge y = 0 \wedge pc2' = 3 \wedge z' = z \wedge y' = y) \vee (pc2 = 3 \wedge x = -100 \wedge pc2' = 4 \wedge z' = 0 \wedge y' = y) \vee (pc2 = 4 \wedge z = 5 \wedge pc2' = 5 \wedge z' = z \wedge y' = y) \vee (pc2 = 5 \wedge y = 90 \wedge pc2' = 0 \wedge z' = z \wedge y' = y) \vee (pc2' = pc2 \wedge z' = z \wedge y' = y)) \wedge \Box((pc2 = 0 \wedge x \geq 1000 \wedge \dot{y} = 0) \vee (pc2 = 1 \wedge \dot{z} = 1 \wedge \dot{y} = 0 \wedge z \leq 5) \vee (pc2 = 2 \wedge \dot{y} = -9 \wedge y \geq 0) \vee (pc2 = 3 \wedge x \geq -100 \wedge x < 1000 \wedge \dot{y} = 0) \vee (pc2 = 4 \wedge \dot{z} = 1 \wedge \dot{y} = 0 \wedge z \leq 5) \vee (pc2 = 5 \wedge \dot{y} = 9 \wedge y \leq 90)))$.

下面证明 $\Box(x \geq -100 \wedge x \leq 100 \Rightarrow pc2 = \text{closed})$ 是复合模块 [Train‖Gate] 的一个性质, 即当火车位于道口前后 100 米之内时, 道口是关闭的。

设 $\mathcal{J}$ 是 TLF(Train) $\wedge$ TLF(Gate) 的一个模型, 且 f_{pc2}, f_x, f_y 和 f_z 分别是变量 $pc2$, x, y 和 z 在模型 $\mathcal{J}$ 下的解释。

设 $a_1, a_2, \ldots, a_n, \ldots$ 是函数 $f_{pc2}(t)$ 的所有不连续点组成的序列且满足 $a_1 < a_2 < a_3 < \cdots < a_n < \ldots$ (这里我们不妨假定 f_{pc2} 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)。由于 $(x > 2000) \wedge (pc2 = 0) \wedge ((pc2 = 0 \wedge x = 1000 \wedge pc2' = 1) \vee (pc2 = 1 \wedge pc2' = 2) \vee (pc2 = 2 \wedge pc2' = 3) \vee (pc2 = 3 \wedge pc2' = 4) \vee (pc2 = 4 \wedge pc2' = 5) \vee (pc2 = 5 \wedge pc2' = 0) \vee (pc2' = pc2))$ 蕴涵 $(pc2' = pc2)$, 故 TLF(Train)$\wedge$TLF(Gate)

也蕴涵 $(pc2' = pc2)$，于是 f_{pc2} 在初始点 $t = 0$ 处是右连续的，于是 0 不是 f_{pc2} 的不连续点，于是有 $a_1 > 0$ 。

取 $a_0 = 0$, 则对任意 $n \in \mathcal{N}$ 有 f_{pc2} 在区间 $(a_n, a_{n+1}]$ 上连续。由 $\Box((pc2 = 0 \wedge x = 1000 \wedge pc2' = 1 \wedge z' = 0 \wedge y' = y) \vee (pc2 = 1 \wedge z = 5 \wedge pc2' = 2 \wedge z' = z \wedge y' = y) \vee (pc2 = 2 \wedge y = 0 \wedge pc2' = 3 \wedge z' = z \wedge y' = y) \vee (pc2 = 3 \wedge x = -100 \wedge pc2' = 4 \wedge z' = 0 \wedge y' = y) \vee (pc2 = 4 \wedge z = 5 \wedge pc2' = 5 \wedge z' = z \wedge y' = y) \vee (pc2 = 5 \wedge y = 90 \wedge pc2' = 0 \wedge z' = z \wedge y' = y) \vee (pc2' = pc2 \wedge z' = z \wedge y' = y))$ 知 f_{pc2} 最多只有六种不连续点，在这些不连续点处 f_{pc2} 的值分别由 0, 1, 2, 3, 4, 5 改变为 1, 2, 3, 4, 5, 6, 0.

通过对 i 进行归纳，可得对任意 $i \in \mathcal{N}$ 有：

$$f_{pc2}(t) = \begin{cases} 0 & \text{若 } t \in (a_{6i},\, a_{6i+1}], \\ 1 & \text{若 } t \in (a_{6i+1}, a_{6i+2}], \\ 2 & \text{若 } t \in (a_{6i+2}, a_{6i+3}], \\ 3 & \text{若 } t \in (a_{6i+3}, a_{6i+4}]. \\ 4 & \text{若 } t \in (a_{6i+4}, a_{6i+5}], \\ 5 & \text{若 } t \in (a_{6i+5}, a_{6i+6}]. \end{cases} \tag{1}$$

由 f_y 和 f_z 的值的变化可得： $a_{6i+2} = a_{6i+1} + 5$, $a_{6i+3} = a_{6i+2} + 10$, $a_{6i+5} = a_{6i+4} + 5$ 且 $a_{6i+6} = a_{6i+5} + 10$ 。又由于当 $(-100 < f_x(t) < 2000)$ 时 $f_x(t)$ 满足 $(-50 \leq \dot{f}_x(t) \leq -30)$ 于是由 $(-50 \leq \dot{f}_x(t))$ 可得 $f_x(a_{6i+2}) \geq f_x(a_{6i+1}) - 50 * 5 = 750$, $f_x(a_{6i+3}) \geq f_x(a_{6i+1}) - 50 * 15 = 250$, $f_x(a_{6i+5}) > 2000 - 50 * 5 = 1750$ 和 $f_x(a_{6i+6}) > 2000 - 50 * 15 = 1250$.

施归纳于 i 可知对任意 $i \in \mathcal{N}$ 有：

$$\begin{aligned} &((t \in (a_{6i},\, a_{6i+1}] \Rightarrow f_x(t) \geq 1000) \wedge (t \in (a_{6i+1},\, a_{6i+2}] \Rightarrow f_x(t) \geq 750) \wedge \\ &(t \in (a_{6i+2},\, a_{6i+3}] \Rightarrow f_x(t) \geq 250) \wedge (t \in (a_{6i+3},\, a_{6i+4}] \Rightarrow f_x(t) \geq -100) \wedge \\ &(t \in (a_{6i+4},\, a_{6i+5}] \Rightarrow f_x(t) > 1750) \wedge (t \in (a_{6i+5},\, a_{6i+6}] \Rightarrow f_x(t) > 1250)) \end{aligned} \tag{2}$$

由（ 1 ）和（ 2 ）可得：

对任意$t \in \mathcal{R}^+$：

$$\begin{aligned} &((f_{pc2}(t) = 0 \wedge f_x(t) \geq 1000) \vee (f_{pc2}(t) = 1 \wedge f_x(t) \geq 750) \vee \\ &(f_{pc2}(t) = 2 \wedge f_x(t) \geq 250) \vee (f_{pc2}(t) = 3 \wedge f_x(t) \geq -100) \vee \\ &(f_{pc2}(t) = 4 \wedge f_x(t) > 1750) \vee (f_{pc2}(t) = 5 \wedge f_x(t) > 1250)) \end{aligned} \tag{3}$$

由 (3) 知 $\mathcal{J}$ 是 $\Box((pc2 = 0 \wedge x \geq 1000) \vee (pc2 = 1 \wedge x \geq 750) \vee (pc2 = 2 \wedge x \geq 250) \vee (pc2 = 3 \wedge x \geq -100) \vee (pc2 = 4 \wedge x \geq 1750) \vee (pc2 = 5 \wedge x \geq 1250))$ 的一个模型。但由于 $\mathcal{J}$ 是复合模块 [Train‖Gate] 的任一模型，故由定义 5.4 知 $\Box((pc2 = 0 \wedge x \geq$

$1000) \vee (pc2 = 1 \wedge x \geq 750) \vee (pc2 = 2 \wedge x \geq 250) \vee (pc2 = 3 \wedge x \geq -100) \vee (pc2 = 4 \wedge x \geq 1750) \vee (pc2 = 5 \wedge x \geq 1250))$ 是 [Train‖Gate] 的一个性质。

由于 $x \geq -100 \wedge x \leq 100$ 与 $x \geq 1000$ 、 $x \geq 750$ 、 $x \geq 250$ 、 $x \geq 1750$ 以及 $x \geq 1250$ 都不相容，于是由 $\Box((pc2 = 0 \wedge x \geq 1000) \vee (pc2 = 1 \wedge x \geq 750) \vee (pc2 = 2 \wedge x \geq 250) \vee (pc2 = 3 \wedge x \geq -100) \vee (pc2 = 4 \wedge x \geq 1750) \vee (pc2 = 5 \wedge x \geq 1250))$ 知当 $x \geq -100 \wedge x \leq 100$ 时必有 $pc2 = 3$ 。于是 $\Box(x \geq -100 \wedge x \leq 100 \Rightarrow pc2 = 3)$ 是复合模块 [Train‖Gate] 的一个性质。

5.4　多速率混成模块的样本模型检查

5.4.1　多速率混成模块

定义 5.5 称一个混成模块 M 是多速率的 *(multirate, 参考 [21] 中 multirate automata 的定义)* 若，

1. M 是闭的，即 M 中没有外部变量。
2. M 的初始条件 $init\_cond$ 是形如 $vertex \wedge (x_1 = r_1 \wedge x_2 = r_2 \wedge \ldots \wedge x_k = r_k)$ 的一个 $LTLC/H$ 公式，其中 $vertex$ 是一个顶点公式， $x_1, x_2, \ldots, x_k$ 是 M 中的所有柔性变量， $r_1, r_2, \ldots, r_k$ 是有理数；此条件限定了 M 中的任一柔性变量在初始状态的值都是有理数。
3. *Jump* 转换中的 *guard* 部分是一个当前状态公式。
4. *Flow* 转换中的 *diff* 部分是形如 $\dot{x}_1 = a_1 \wedge \dot{x}_2 = a_2 \wedge \ldots \wedge \dot{x}_k = a_k$ 的公式，这里 $x_1, x_2, \ldots, x_k$ 是 M 中的所有柔性变量， $a_1, a_2, \ldots, a_k$ 是有理数；即在每个顶点内，每个柔性变量的导数是一个有理数（称这个有理数为该变量在该顶点上的速率）；也就是说，每个柔性变量在每个顶点内依照一个固定的加速度变化，但在两个不同顶点内，同一个柔性变量的导数可以不同。

由上述定义不难看到例 5.2 中的混成模块 $Monitor$ 是一个多速率混成模块。。

在前面所讨论的混成系统的模型中，跳跃转换可发生在任何实数点上，混成系统的这种控制方式称为稠密时间控制 (dense-time control[50])，稠密时间控制是一种理想化的控制方式，在实际的控制系统中很难做到。相对而言，样本控制 (sampling control[50]) 是一种更现实和更自然的控制方式 [50] 。在这种控制方式下，控制器每隔一个固定的时间片段采样一次，并根据采样结果确定让哪一个跳跃转换发生。不失一般性，在数学上我们可假定这个时间片为 1 ，这样在样本控制下，混成系统的跳跃转换只能发生在整数点上。混成系

统在样本控制下的模型称为样本模型，下面我们讨论多速率混成模块在样本控制下的模型检查。多速率混成模块是一类常见的而且比较重要的混成系统，不幸的是，它（在稠密时间控制方式下）的模型检查问题是不可判定的 [21]，故研究它在样本控制下的模型检查就显得非常有必要。

本章下面的样本模型检查和文献 [50, 64] 中关于样本模型和离散模型下的模型检查不同。在他们的讨论中，模型已被抽象成由系统在整数点上的状态所形成的无穷序列（忽略掉系统在非整数点上的状态），并假定要检查的性质足够的大（'large enough'[50]）以保证可在抽样点上观察到它的每次发生，即该性质不能在两个相邻的抽样点上为假而在其之间为真。在我们下面的讨论中，混成系统的模型依然包括所有点（无论抽样点还是非抽样点）上的状态，它是系统状态随时间流逝而形成的一个自然的轨迹，包括了所有时刻的状态。我们也不要求所要检查的性质足够的大，它可以发生在相邻的两个抽样点之间的任一个点上（比如变量 x 在相邻抽样点 $t=1$ 和 $t=2$ 上的值分别为 1 和 3, 而验证的性质是 $x=2$ ）。

定义 5.6

1. 设 $\mathcal{J}$ 是变量集 V 上的一个 *LTLC*- 模型, 对于 $t \in \mathcal{R}^+$, 如果存在 V 中的变量 v 使得 $f_v(t) \neq f'_v(t)$*(*以下我们默认使用 f_v 来表示变量 v 在 *LTLC*- 模型 $\mathcal{J}$ 下的解释*)*，则称 t 是 $\mathcal{J}$ 的一个不连续点。若 $\mathcal{J}$ 的不连续点都是整点，则称 $\mathcal{J}$ 是 V 上的一个样本模型。

2. 设 M 是变量集 V 上的一个混成模块， φ 是变量集 V 上的一个不含有带点变量的 *LTLC/H*- 公式；若对于 M 的任一样本模型 $\mathcal{J}$ 有 $\mathcal{J} \models \varphi$，则称 φ 是 M 在样本模型下的一个性质，记为 $M \models_{sim} \varphi$.

3. 混成模块的样本模型检查问题是指对于任一给定的混成模块 M 及任一给定的不含有带点变量的 *LTLC/H*- 公式 φ（即要求 φ 中不含微分公式），判断是否有 $M \models_{sim} \varphi$.

设 M 是变量集 V 上的一个多速率混成模块, 若存在正整数 n 使得对于 M 的任一模型 $\mathcal{J}$ 及 V 中的任一柔性变元 v 有：函数 f_v 受囿于 n（即对于任意 $t \in \mathcal{R}^+$ 有 $-n \leq f_v \leq n$ ）或者它是分段单调增加的；则称模块 M 是有界或单调的，并称这样一个正整数 n 为模块 M 的一个上界。

声明： 本章以下所说的多速率混成模块总假定是有界或单调的。

定义 5.7 设 V 是一个有穷变量集 *(*含布尔型变量和柔性变量*)*， n 是一个正整数。

1. 以 $Form_H(V,n)$ 表示 *LTLC/H* 公式集 $\{\varphi \mid var(\varphi) \subseteq V$ ， φ 中不含带点变量和非整数常元，且 φ 中的（整数）常元的绝对值皆不超过 $n\}$.

2. 以 $\mathcal{H}(V,n)$ 表示变量集 V 上的所有满足下列条件的多速率混成模块 M 的集合。

- M 中出现的所有常元均是整数常元, 且绝对值不超过 n (此假定蕴涵 M 的所有柔性变量的初始值和其用公式 $x' = r$ 赋得的新值的绝对值都小于或等于 n).
- M 中的所有非零速率均整除 n (即若 $\dot{x} = a$ 在 M 中出现且 $a \neq 0$, 则 n 是 a 的整数倍).
- n 是 M 的一个上界, 即 M 中任一柔性变元要么其绝对值小于等于 n , 要么它是分段单调增加的。

通过对变量进行适当的替换 (如乘以适当的倍数), 多速率混成模块 (要求是有界或单调的, 但可以含有非整数的有理数常元) 的样本模型检查可化归为 $\mathcal{H}(V,n)$ 中的混成模块关于 $Form_H(V,n)$ 中的公式的样本模型检查 (需要选择适当的正整数 n) 。所以下面只考虑 $\mathcal{H}(V,n)$ 中的混成模块关于 $Form_H(V,n)$ 中的公式的样本模型检查即可。

5.4.2　状态等价

在以下的讨论中, n 是一个相对固定的正整数, V 是一个相对固定的有穷变量集, 且 $V = B \cup C$, 其中 B 是一个布尔型变量集, C 是一个柔性变量集。

对于上面固定的正整数 n, 定义函数 $h_n : \mathcal{R} \mapsto \{ -2n+1, -2n, \ldots, -1, 0, 1, \ldots, 2n, 2n+1 \}$ 如下:

$$h_n(a) = \begin{cases} 2a & \text{若 } a = \lfloor a \rfloor \text{ 且 } |a| \leq n, \\ 2\lfloor a \rfloor + 1 & \text{若 } a \neq \lfloor a \rfloor \text{ 且 } |a| \leq n, \\ 2n+1 & \text{若 } a > n, \\ -2n-1 & \text{若 } a < -n. \end{cases}$$

这里 $\lfloor a \rfloor, |a|$ 分别表示实数 a 的整数部分和 a 的绝对值.

不难验证 $h_n(a)$ 是单增的且对于不超过 n 的非负整数 m 有: $h_n(a) \leq 2m$ 当且仅当 $a \leq m$.

为方便起见下面常以 $[a]_n$ 记 $h_n(a)$.

设 $V = B \cup C$, 令 $V' = \{ v' \mid v \in V \}$, 称 $V \cup V'$ 上的映射 σ 为变量集 V 上的一个 (LTLC/H-) 状态, 若对任意 $p \in B$ 有 $\sigma(p) \in \{0,1\}$, $\sigma(p') \in \{0,1\}$, 且对任意 $x \in C$ 有 $\sigma(x) \in \mathcal{R}$, $\sigma(x') = \sigma(x) \vee \sigma(x') \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$. 若 σ 是变量集 V 上的一个状态, 对任意 (LTLC/H-) 状态公式 φ, 可按通常的做法定义 φ 在 σ 下的真值 $\sigma(\varphi)$.

对于正整数 n, 定义 V 上的状态之间的等价关系 $\equiv_n$ 如下：

定义 5.8 设 σ 和 τ 是 V 上的任意两个状态， $\sigma \equiv_n \tau$ 当且仅当

1. 对任意 $p \in B$ 有： $\sigma(p) = \tau(p),\ \sigma(p') = \tau(p')$;

2. 对任意 $x \in C$ 有： $[\sigma(x)]_n = [\tau(x)]_n,\ [\sigma(x')]_n = [\tau(x')]_n$;

显然， '$\equiv_n$' 是一个等价关系，记 σ 所在的等价类为 $[\sigma]_n$.

定义 5.9 设 σ 是 V 上的任意一个状态，定义 $V \cup V'$ 上的映射 $h_n(\sigma)$ 如下：

(1). 对任意 $p \in B$ 有： $h_n(\sigma)(p) ::= \sigma(p)$ 且 $h_n(\sigma)(p') ::= \sigma(p')$;

(2). 对任意 $x \in C$ 有： $h_n(\sigma)(x) ::= [\sigma(x)]_n$ 且 $h_n(\sigma)(x') ::= [\sigma(x')]_n$.

不难看到 V 上两状态 σ 和 τ 等价当且仅当 $h_n(\sigma) = h_n(\tau)$.

引理 5.1 设 σ, τ 是 V 上的两个等价状态， $\varphi \in Form_H(V, n)$ 且 φ 是一个状态公式，则 $\sigma(\varphi) = \tau(\varphi)$.

证明：对公式 φ 进行归纳即可得，下面只给出情况 $\varphi := (x' = x)$ 时的证明作为示例。

假设 $\sigma(x' = x) = 1$, 而 $\tau(x' = x) = 0$. 于是 $\sigma(x') = \sigma(x)$, 且 $\tau(x') \neq \tau(x)$. 由状态的定义知 τ 满足 $\tau(x') = \tau(x) \vee \tau(x') \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$, 于是 $\tau(x') \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$, 再由 $\sigma \equiv_n \tau$ 得出 $\sigma(x') = \tau(x')$ 且 $\sigma(x') \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$, 于是进一步可得 $\sigma(x) \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$ 且 $\tau(x) = \sigma(x)$. 这样就可得到 $\tau(x) = \sigma(x) = \sigma(x') = \tau(x')$, 但这与 $\tau(x') \neq \tau(x)$ 矛盾.

□.

定义 5.10 对于正整数 n 及变量集 V, 以 *Interp(V,n)* 表示 V 上所有满足如下条件的样本模型 $\mathcal{J}$ 的集合：

1. 对任意 $x \in C$, $f_x(0) \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$.

2. 对任意 $x \in C$ 及任意的 $k \in \mathcal{N}$, f_x 在 $(k, k+1)$ 上的导数恒为一个整数，且该整数整除 n 。

3. 对任意 $x \in C$ 及任意的 $k \in \mathcal{N}$, $f'_x(k) = f_x(k)$ 或 $f'_x(k) \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$.

引理 5.2 设 $M \in \mathcal{H}(V, n)$ ， $\mathcal{J}$ 是 M 的一个样本模型，证明 $\mathcal{J} \in$ *Interp(V,n)*.

证明:

1. 由定义 5.5 和定义 5.7 对 M 的假定知 M 的柔性变量的初始值为一个绝对值不超过 n 的整数，故对任意 $x \in C$ 有，$f_x(0) \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$.

2. 由于 $\mathcal{J}$ 是 M 的一个样本模型，故对任意的 $k \in \mathcal{N}$ 有 $\mathcal{J}$ 在 $(k, k+1)$ 上连续，于是在此期间模块所位于的顶点不发生变化；于是对任意 $x \in C$，由定义 5.5 和定义 5.7 对 M 的假定知存在 $a \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$ 使得 f_x 在 $(k, k+1)$ 上始终满足 $\dot{x} = a$，于是可得 f_x 在 $(k, k+1)$ 上的导数恒为整数 a，且 n 是 a 的一个倍数。

3. 对任意 $x \in C$ 及任意的 $k \in \mathcal{N}$，如果 $f_x(k) \neq f'_x(k)$，则 k 是 $\mathcal{J}$ 的一个不连续点，而 $\mathcal{J}$ 的不连续点是由 M 的某个跳跃转换的作用所引起的 (cf. 定义 5.1)，根据跳跃转换中 'assignment' 部分的假定 (见 70 页)，在跳跃转换作用时，柔性变量 x 要么保持原值 (若 'assignment' 中含公式 $x' = x$ 时) 要么被赋予一个新值 (若 'assignment' 中含公式 $x' = r$ 且 r 是一常元时)，但由定义 5.7 的假定，M 中只含绝对值不超过 n 的整数常元，于是当跳跃转换给变量 x 赋予一个新值时 (通过公式 $x' = r$)，这个新值只能是一个在集合 $\{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$ 中的值。于是对任意 $x \in C$ 及任意的 $k \in \mathcal{N}$ 有 $f'_x(k) = f_x(k)$ 或 $f'_x(k) \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$.

□.

引理 5.3 设 $\mathcal{J} \in$ *Interp(V,n)*，x 是 V 中的一个柔性变量，证明对任意 $k \in \mathcal{N}$ 有 $f_x(k)$ 和 $f'_x(k)$ 均为整数。

证明：对 k 进行归纳。

当 k=0 时，由定义 5.10 知 $f_x(0)$ 为整数，又由定义 5.10 知要么 $f_x(0) = f'_x(0)$ 要么 $f'_x(0) \in \{-n, -n+1, \ldots, 0, \ldots, n-1, n\}$. 于是总有 $f'_x(0)$ 是一个整数。

假设当 $k = i$ 时结论成立，则当 $k = i+1$ 时，由定义 5.10 知 f_x 在区间 $(i, i+1)$ 上可导且导函数在区间 $(i, i+1)$ 上是一个绝对值不超过 n 的整数，不妨设其为 a_i^x，则由中值定理得 $f_x(i+1) = f'_x(i) + a_i^x$，从归纳假设知 $f'_x(i)$ 是整数，于是 $f_x(i+1)$ 也是整数；由定义 5.10 还可知 $f'_x(k+1) = f_x(k+1)$ 或 $f'_x(k+1) \in \{-n, \ldots, -1, 0, 1, \ldots, n\}$，于是 $f'_x(i+1)$ 也是整数。

□.

设 $\mathcal{J} \in$ Interp(V,n)，对于任意 $t \in \mathcal{R}^+$, 我们用 $\mathcal{J}(t)$ 来表示 V 上如下定义的一个状态：对任意的 $v \in V$，变量 v 的值为 $f_v(t)$, v' 的值为 $f'_v(t)$ （由定义 5.10 知 $\mathcal{J}(t)$ 是 V 上的一个状态).

引理 5.4 . 设 $0 \leq b_1 < b_2 \leq n$, 且开区间 (b_1, b_2) 内不含任何整数，则对于任意 $s_1, s_2 \in (b_1, b_2)$ 有 $h_n(s_1) = h_n(s_2)$ 。

证明：由假设知 s_1 和 s_2 不是整数，但它们却有相同的整数部分（否则 s_1 和 s_2 之间必有一个整数），于是有 $h_n(s_1) = h_n(s_2)$ 。

□.

定义 5.11 定义映射 $\ell_n : \mathcal{R}^+ \mapsto \mathcal{N}$ 如下：

$$\text{对于 } s \in \mathcal{R}^+,\ \ell_n(s) = \begin{cases} 2k & \text{若存在 } k \in \mathcal{N} \text{ 使 } s = k/n \\ 2k+1 & \text{若存在 } k \in \mathcal{N} \text{ 使 } k/n < s < (k+1)/n \end{cases}$$

引理 5.5 设 $\mathcal{J} \in$ *Interp(V,n)* ，对于任意 $s \in \mathcal{R}^+$ 有 $\mathcal{J}(s) \equiv_n \mathcal{J}(\ell_n(s)/2n)$, 即状态 $\mathcal{J}(s)$ 和 $\mathcal{J}(\ell_n(s)/2n)$ 等价。

证明：

1. 若存在 $k \in \mathcal{N}$ 使 $s = k/n$ ，则 $s = \ell_n(s)/2n$, 结论显然。

2. 若存在 $k \in \mathcal{N}$ 使 $k/n < s < (k+1)/n$ 。设 k/n 的整数部分为 s_0, 则存在整数 k_1 使得 $k/n = s_0 + k_1/n$, 且 $0 \le k_1 < n$. 于是 $s_0 < s < s_0 + 1$.

   (i). 对于任意 $p \in B$, 由于 f_p 的不连续点只可能出现在整数点上，故 f_p 在开区间 (s_0, s_0+1) 上始终为 0 或始终为 1；　而 s 和 $\ell_n(s)/2n$ 均在 (s_0, s_0+1) 中，故 $f'_p(s) = f_p(s) = f_p(\ell_n(s)/2n) = f'_p(\ell_n(s)/2n)$.

   (ii). 对于任意 $x \in C$, 由于 f_x 的不连续点只可能出现在整数点上，故 f_x 在开区间 (s_0, s_0+1) 上连续，且 f_x 在 (s_0, s_0+1) 上的导数恒为一个整数，记为 a; 则 $f'_x(t) = f_x(s) = f'_x(s_0) + a(s - s_0)$; 同样有 $f'_x(\ell_n(s)/2n) = f_x(\ell_n(s)/2n) = f'_x(s_0) + a(\frac{2k+1}{2n} - s_0)$.

   若 $a = 0$ ，则 $f'_x(s) = f_x(s) = f'_x(s_0) = f_x(\ell_n(s)/2n) = f'_x(\ell_n(s)/2n)$ ，于是 $h_n(f_x(s) = h_n(f_x(\ell_n(s)/2n))$ 且 $h_n(f'_x(s) = h_n(f'_x(\ell_n(s)/2n))$.

   下面假定 $a \neq 0$ ，由定义 5.10 知 n 是 a 的倍数，于是存在正整数 n_1 使得 $n = |a|n_1$（这里 $|a|$ 表示 a 的绝对值）.

   令 $b_1 ::= k_1/n_1$ 且 $b_2 ::= (k_1+1)/n_1$ ，则不难验证 $a(s - s_0)$ 和 $a(\frac{2k+1}{2n} - s_0)$ 都位于开区间 (b_1, b_2) 内，且 (b_1, b_2) 内不含任何整数（否则存在正整数 d 使 $k_1 < n_1 d < k_1 + 1$, 但由于 n_1 和 d 都是整数，故这是不可能的）. 由前面的引理 5.4 便得到 $h_n(a(s - s_0)) = h_n(a(\frac{2k+1}{2n} - s_0))$; 又由于 $f'_x(s_0)$ 是整数，故 $h_n(f_x(s) = h_n(f_x(\ell_n(s)/2n))$ 且 $h_n(f'_x(s) = h_n(f'_x(\ell_n(s)/2n))$.

   这样就得到了结论 $h_n(\mathcal{J}(s)) = h_n(\mathcal{J}(\ell_n(s)/2n))$ ，即状态 $\mathcal{J}(s)$ 和 $\mathcal{J}(\ell_n(s)/2n)$ 等价。

□.

5.4.3　构造有限状态变换系统 $\widehat{M}$

对于正整数 n 及有穷变量集 $V = B \cup C$ 。设 $M \in \mathcal{H}(V,n)$ ，以下我们构造一个有限状态变换系统 $\widehat{M}$, $\widehat{M}$ 是一个 LTL 公式，它的变量集合为 $V_d = B \cup C \cup B' \cup C' \cup \{i\}$, 其中 $B' = \{p' \mid p \in B\}$, $C' = \{x' \mid x \in C\}$, 且 i 是一个不在 $B \cup C \cup B' \cup C'$ 中出现的变量. $B \cup B'$ 中的变量是命题变量，取值 0 和 1; $C \cup C'$ 中的变量为整型变量，取值域为 $\{-2n-1, -2n, \ldots, -1, 0, 1, \ldots, 2n, 2n+1\}$; i 也是整型变量,取值域为 $\{0, 1, 2, \ldots, 2n-1\}$.

定义 5.12 设 $\varphi \in Form_H(V,n)$, 归纳定义 φ 所对应的（变量集 V_d 上的） *LTL* 公式 $\overline{\varphi}$ 如下：

1. 对 $p \in B$ ，　$\overline{p} ::= p$,　$\overline{p'} ::= p'$;

2. 对 $x \in C$ ，　$\overline{x \leq r} ::= x \leq 2r$,　$\overline{x = r} ::= x = 2r$,　$\overline{x' \leq r} ::= x' \leq 2r$, $\overline{x' = r} ::= x' = 2r$,　$\overline{x' = x} ::= x' = x$;

3. $\overline{\neg\varphi} ::= \neg\overline{\varphi}$,　$\overline{\varphi_1 \wedge \varphi_2} ::= \overline{\varphi_1} \wedge \overline{\varphi_2}$,　$\overline{\Box\varphi} ::= \Box\overline{\varphi}$,　$\overline{\varphi_1 \mathcal{U} \varphi_2} ::= \overline{\varphi_1}\mathcal{U}\overline{\varphi_2}$.

引理 5.6 设 σ 是 V 上的一个状态，　$\varphi \in Form_H(V,n)$ 且它是一个状态公式，则 $\sigma(\varphi) = h_n(\sigma)(\overline{\varphi})$ 。

证明:

对 φ 用归纳即可，这里只给出情况 $\varphi := (x \leq r)$ 时的证明作为示例。

$h_n(\sigma)(\overline{\varphi}) = 1$ iff $h_n(\sigma)(x \leq 2r) = 1$ iff $h_n(\sigma)(x) \leq 2r$ iff $[\sigma(x)]_n \leq 2r$ iff $\sigma(x) \leq r$ iff $\sigma(\varphi) = 1$.

□.

定义 5.13 设 $\mathcal{J} \in$ *Interp(V,n)* ，定义 $\mathcal{J}$ 对应的一个（变量集 V_d 上的） *LTL-* 模型 $\widehat{\mathcal{J}}$ 如下：

1. 对于任意 $v \in V$ 及任意 $k \in \mathcal{N}$,

$\widehat{\mathcal{J}}(v,k) ::= h_n(\mathcal{J}(\frac{k}{2n}))(v)$,　$\widehat{\mathcal{J}}(v',k) ::= h_n(\mathcal{J}(\frac{k}{2n}))(v')$

2. 对于变量 i 及任意的 $k \in \mathcal{N}$, $\widehat{\mathcal{J}}(i,k) ::= k \ mod \ 2n$.

注：这里 'mod' 表示整数集上的 '模' 运算.

引理 5.7 设 $\mathcal{J} \in$ *Interp(V,n)* ，　$\widehat{\mathcal{J}}$ 是 $\mathcal{J}$ 所对应的 *LTL-* 模型；证明对于任意 $\varphi \in Form_H(V,n)$ 有，　$\mathcal{J} \models \varphi$ 当且仅当 $\widehat{\mathcal{J}} \models_{ltl} \overline{\varphi}$.

证明：只需证对任意 $s \in \mathcal{R}^+$ 有： $\mathcal{J}(\varphi, s) = \widehat{\mathcal{J}}(\overline{\varphi}, \ell_n(s))$ 即可.

对 φ 进行归纳，归纳基础可由引理 5.5 和 5.6 得到，下面给出 $\varphi := \Box\phi$ 情况下的证明，其余情况略.

1. 若 $\mathcal{J}(\Box\phi, s) = 1$, 则对于任意 $r \geq s$ 有 $\mathcal{J}(\phi, r) = 1$, 由归纳假设得，对任意的 $r \geq s$ 都有 $\widehat{\mathcal{J}}(\overline{\phi}, \ell_n(r)) = 1$ 成立。考虑 ℓ_n 的特性，由于 $\ell_{\mathbb{Z}}$ 是满射，且是单增的，于是可得 $\widehat{\mathcal{J}}(\overline{\Box\phi}, \ell_n(s)) = 1$.

2. 若 $\mathcal{J}(\Box\phi, s) = 0$, 则存在 $r \geq s$ 使得 $\mathcal{J}(\phi, r) = 0$, 由归纳假设得 $\widehat{\mathcal{J}}(\overline{\phi}, \ell_n(r)) = 0$. 由于 $\ell_n(r) \geq \ell_n(s)$, 于是 $\widehat{\mathcal{J}}(\overline{\Box\phi}, \ell_n(s)) = 0$.

□.

定义 5.14 对于上面固定的 n, 定义集合 $\{ -2n-1, -2n, \ldots, -1, 0, 1, \ldots, 2n, 2n+1 \}$ 上的两个运算如下：

$$succ(x) = \begin{cases} x+1 & \text{若 } x \leq 2n \\ x & \text{若 } x = 2n+1 \end{cases}$$

$$prec(x) = \begin{cases} x-1 & \text{若 } x \geq -2n \\ x & \text{若 } x = -2n-1 \end{cases}$$

为方便起见，在下面把 succ(x) 和 prec(x) 常写成 $x\dot{+}1$ 和 $x\dot{-}1$. 此外对于两个整数 a 和 b ，在下面的定义中以 $a|b$ 表示 a 整除 b, 即 b 是 a 的整数倍。

定义 5.15

1. 设 x 是 V 中的一个柔性变量， a 是一个绝对值不超过 n 的整数；定义微分公式 $\dot{x} = a$ 对应的（变量集 V_d 上的） *LTL* 公式 $\overline{\dot{x} = a}$ 为

   $\overline{\dot{x} = a} ::= first \vee ((a > 0 \wedge (\, 2n|i*a \vee 2n|(i-1)a\,) \Rightarrow x = (x')^- \dot{+} 1\,) \wedge (a < 0 \wedge (\, 2n|i*a \vee 2n|(i-1)a\,) \Rightarrow x = (x')^- \dot{-} 1\,) \wedge (a = 0 \vee \neg(\, 2n|i*a \vee 2n|(i-1)a\,) \Rightarrow x = (x')^-))$

2. 设 $x_1, x_2, \ldots, x_k$ 是 V 中的柔性变量， $a_1, a_2, \ldots, a_k$ 是整数且绝对值不超过 n ；则微分公式 $\dot{x}_1 = a_1 \wedge \dot{x}_2 = a_2 \wedge \ldots \wedge \dot{x}_k = a_k$ 所对应的 *LTL* 公式 $\overline{\dot{x}_1 = a_1 \wedge \dot{x}_2 = a_2 \wedge \ldots \wedge \dot{x}_k = a_k}$ 定义为 $\overline{\dot{x}_1 = a_1} \wedge \overline{\dot{x}_2 = a_2} \wedge \ldots \wedge \overline{\dot{x}_k = a_k}$.

注：上面的 $2n|i*a$ 可以看成是 LTL 公式 $\bigvee_{k \in \{-n,\ldots,-1,0,1,\ldots,n\}}(k * 2n = i * a)$ 的缩写； $2n|(i+1)a$ 可以看成是 LTL 公式 $\bigvee_{k \in \{-n,\ldots,-1,0,1,\ldots,n\}}(k * 2n = (i+1)a)$ 的缩写； $x = (x')^- \dot{+} 1$ 可以看成是 LTL 公式 $((x')^- \leq 2n \wedge x = (x')^- + 1) \vee ((x')^- =$

$2n+1 \wedge x = (x')^-$) 的缩写； $x = (x')^- \dot{-} 1$ 可以看成是 LTL 公式 $((x')^- \geq -2n \wedge x = (x')^- - 1) \vee ((x')^- = -2n-1 \wedge x = (x')^-)$ 的缩写. 采用缩写是因为在 LTL 中 (见定义 2.2) 没有包括这里的符号 $|, \dot{+}$ 及 $\dot{-}$ 。

定义 5.16

1. 设 $\alpha = vertex \wedge guard \rightarrow new\_vextex \wedge assignment$ 是 M 的一个跳跃转换, 定义 α 所对应的 (变量集 V_d 上的) LTL 公式 $\widehat{\alpha}$ 为 $i = 0 \wedge vertex \wedge \overline{guard} \wedge new\_vertex \wedge \overline{assignment}$.

2. 设 $\beta = vertex \rightarrow diff \wedge invariant$ 是 M 的一个流转换, 定义 β 所对应的 (变量集 V_d 上的) LTL 公式 $\widehat{\beta}$ 为 $vertex \wedge \overline{diff} \wedge \overline{invariant}$.

定义 5.17 对于多速率混成模块 $M \in \mathcal{H}(V, n)$ ，设 $init\_cond$ 是 M 的初始条件, $\alpha_0, \alpha_1, \ldots, \alpha_{m-1}$ 是 M 的所有跳跃转换， $\beta_0, \beta_1, \ldots, \beta_{l-1}$ 是 M 的所有流转换。我们称 (变量集 V_d 上的) LTL 公式 $\overline{init\_cond} \wedge (i = 0 \wedge \Box(i^+ = (i+1) mod\ 2n)) \wedge \Box(V' = V \vee \bigvee_{j<m} \widehat{\alpha_j}) \wedge \Box(B^+ = B') \wedge \Box(\bigvee_{k<l} \widehat{\beta_k})$ 为 M 所对应的 LTL 公式; 以下以 $\widehat{M}$ 来表示这个对应于模块 M 的 LTL 公式, 这个 LTL 公式表示了一个有穷状态变换系统。

注：在这里 $i = 0 \wedge \Box(i^+ = (i+1) \bmod 2n)$ 可以看成是 LTL 公式 $i = 0 \wedge \Box((i < 2n-1 \wedge i^+ = i+1) \vee (i = 2n-1 \wedge i^+ = 0))$ 的缩写。

引理 5.8 设 $\mathcal{J}$ 是多速率混成模块 $M \in \mathcal{H}(V, n)$ 的一个样本模型， $\widehat{\mathcal{J}}$ 是 $\mathcal{J}$ 所对应的 LTL- 模型; 证明 $\widehat{\mathcal{J}}$ 是 $\widehat{M}$ 的一个 LTL- 模型。

证明: 设 $\mathcal{J}$ 是多速率混成模块 M 的一个样本模型, 则 $\mathcal{J}$ 是 LTLC 公式 $TLF(M)$ 的一个模型且 $\mathcal{J}$ 的不连续点只可能是整数点。从 $\widehat{\mathcal{J}}$ 的定义的直观背景看 $\widehat{\mathcal{J}}$ 的状态序列是通过对样本模型 $\mathcal{J}$ 取抽样点 $\{k/2n\}_{k \in \mathcal{N}}$, 将抽样点 $k/2n$ 的状态经过映射 h_n 的变换后作为 $\widehat{\mathcal{J}}$ 的第 k 个状态 (但增加了对变量 i 的解释)。由于 $\mathcal{J}$ 的不连续点只可能是整数点, 而 $\widehat{\alpha_j}$ 的定义中的 $i = 0$ 反映了这样一个事实。于是由 $\mathcal{J} \models init\_cond \wedge \Box(V' = V \vee \bigvee_{j<m} \alpha_j)$ 、定义 5.13 及引理 5.7 知 $\widehat{\mathcal{J}} \models_{ltl} \overline{init\_cond} \wedge \Box(i+ = (i+1) \bmod 2n) \wedge \Box(V' = V \vee \bigvee_{j<m} \widehat{\alpha_j})$ 。

对于 $p \in B$, 记 p 在 $\mathcal{J}$ 下的解释为 f_p 。对于 $k \in \mathcal{N}$, 由于 f_p 在区间 $(k, k+1]$ 上的值始终为 1 或 0 。故 $\widehat{\mathcal{J}}$ 满足 $\Box(B^+ = B')$.

对于 $x \in C$, 对于任意 $k \in \mathcal{N}$, 设 $\mathcal{J}$ 在区间 $(k, k+1]$ 上满足 $\dot{x} = a$(a 为一个整数, 且整除 n) ，即对于任意 $t \in (k, k+1]$ 有 $\mathcal{J}(\dot{x} = a, t) = 1$ ，下面证明对于任意满足 $2nk < j \leq 2n(k+1)$ 的正整数 j 有 $\widehat{\mathcal{J}}(\overline{\dot{x} = a}, j) = 1$.

由于 f_x 在区间 $(k, k+1)$ 上的导数恒为 a ，于是对任意 $t \in (k, k+1]$ 有 $f_x(t) = f'_x(k) + a(t-k)$ 。

1. 若 $2nk < j < 2n(k+1)$ 。令 $i = j \bmod 2n$ ，则 $j = 2nk + i$ ，于是 $h_n(f_x(\frac{j}{2n})) = [f'_x(k) + \frac{i*a}{2n}]_n$ ，且 $h_n(f'_x(\frac{j-1}{2n})) = [f'_x(k) + \frac{(i-1)*a}{2n}]_n$ 。

   由于 $f'_x(k)$ 是整数且 a 可整除 n ，故只有当 $2n$ 整除 $i*a$ 或者整除 $(i-1)*a$ 时，$h_n(f_x(\frac{j}{2n}))$ 与 $h_n(f_x(\frac{j-1}{2n}))$ 才可能不等，严格地说，它们两者之间的关系为，

$$h_n(f_x(\frac{j}{2n})) = \begin{cases} h_n(f_x(\frac{j-1}{2n})) \dot{+} 1 & \text{若 } a>0 \wedge (\,2n|i*a \ \vee\ 2n|(i-1)a\,), \\ h_n(f_x(\frac{j-1}{2n})) \dot{-} 1 & \text{若 } a<0 \wedge (\,2n|i*a \ \vee\ 2n|(i-1)a\,), \\ h_n(f_x(\frac{j-1}{2n})) & \text{若 } a=0 \vee \neg(\,2n|i*a \ \vee\ 2n|(i-1)a). \end{cases} \quad (1)$$

   （此处应注意定义 5.7 中的 “ n 是 M 的一个上界” 对 (1) 式的成立是起了作用的，它保证了要么 f_x 的值始终介于 $-n$ 与 n 之间，要么 f_x 的值始终不小于 $-n$ 且在任意连续区间上都是单调增加的。这排除了 $f'_x(k) > n$ 且同时有 $(a < 0)$ 的情况的发生；因为在这种情况下 (1) 可能不成立，譬如取 $f'_x(k) = n+2$, $a = -1$ 且 $j = 2nk+1$ ，则 $h_n(f_x(\frac{j}{2n})) = h_n(n+2-1/2n) = 2n+1$ ，而 $h_n(f'_x(\frac{j-1}{2n})) \dot{-} 1 = h_n(n+2) \dot{-} 1 = (2n+1) \dot{-} 1 = 2n$ ，故此时 (1) 不成立）

2. 若 $j = 2n(k+1)$ 。令 $i = j \bmod 2n$ ，则 $i = 0$ ，于是 $h_n(f_x(\frac{j}{2n})) = [f'_x(k) + a]_n$ ，且 $h_n(f'_x(\frac{j-1}{2n})) = [f'_x(k) + a - \frac{a}{2n})]_n$ 。由于 $f'_x(k)$ 是整数且 a 可整除 n ，于是有，

$$h_n(f_x(\frac{j}{2n})) = \begin{cases} h_n(f_x(\frac{j-1}{2n})) \dot{+} 1 & \text{若 } a > 0, \\ h_n(f_x(\frac{j-1}{2n})) \dot{-} 1 & \text{若 } a < 0, \\ h_n(f_x(\frac{j-1}{2n})) & \text{若 } a = 0. \end{cases} \quad (2)$$

从 (1)(2) 便知当 $2nk < j \le 2n(k+1)$ 时 $\widehat{\mathcal{J}}$ 在时刻 j 满足公式 $\overline{\dot{x} = a}$ 。

至此应用定义 5.15 ，定义 5.15 及引理 5.7 我们不难由 $\mathcal{J} \models \Box(\bigvee_{k<l} \mathrm{TLF}(\beta_k)))$ 得到 $\widehat{\mathcal{J}} \models_{ltl} \Box(\bigvee_{k<l} \widehat{\beta_k})$ 。

于是 $\widehat{\mathcal{J}}$ 是 $\widehat{M}$ 的一个 LTL- 模型。

□.

引理 5.9 设 $\Im$ 是 $\widehat{M}$ 的一个 *LTL*- 模型， k, j 是非负整数，则

1. 对于 $p \in B$, 若 $2nk < j < 2n(k+1)$ 则 $\Im(p, j) = \Im(p', j) = \Im(p', 2nk)$ ，若 $j = 2n(k+1)$ 则 $\Im(p, j) = \Im(p', 2nk)$ 。

2. 对于 $x \in C$ 及非负整数 k ，存在绝对值不超过 n 的整数 a 使得对任意的满足 $2nk < j \le 2n(k+1)$ 的正整数 j 有 $\Im(\overline{\dot{x} = a}, j) = 1$ （这里 a 要么是 *0* 要么它能整除 n ）。

证明:

1. 对于 $p \in B$ ，令 $i = j \bmod 2n$ ；若 $2nk < j < 2n(k+1)$, 则 $i > 0$ ，由 $\Im \models_{ltl} \Box(V' = V \vee i = 0)$ 便得当 $2nk < j < 2n(k+1)$ 时有 $\Im(p', j) = \Im(p, j)$ ，再由 $\Im \models_{ltl} \Box(B^+ = B')$ 便可得 $\Im(p', 2nk) = \Im(p, 2nk{+}1) = \Im(p', 2nk{+}1) = \Im(p, 2nk{+}2) = \Im(p', 2nk+2) = \ldots = \Im(p, 2nk + (2n-1)) = \Im(p', 2nk + (2n-1)) = \Im(p, 2n(k+1))$.

2. 对于 $x \in C$ 及非负整数 k ，由于 $\Im \models_{ltl} \Box(\bigvee_{j<l} \widehat{\beta_j})$, 故 $\Im(\bigvee_{j<l} \widehat{\beta_j}, 2nk+1) = 1$, 于是 M 中有唯一的一个流转换 β_{j_0} 使得 $\Im(\widehat{\beta_{j_0}}, 2nk+1) = 1$ (唯一性可由 71 页关于流转换的互斥性假设得到), 利用本引理的结论 1 和流转换的互斥性假设还可进一步得知对于任意的满足 $2nk < j \leq 2n(k+1)$ 的正整数 j 均有 $\Im(\widehat{\beta_{j_0}}, j) = 1$ 。对于公式 $\widehat{\beta_{j_0}}$, 由定义 5.5 、定义 5.7 及定义 5.15 知存在一个整数 a 使得 $\widehat{\beta_{j_0}}$ 中包含公式 $\overline{\dot{x} = a}$(由定义 5.7 知若 a 非零则它必整除 n) 。于是由 $\widehat{\beta_{j_0}}$ 的定义 (见定义 5.16) 知对于任意的满足 $2nk < j \leq 2n(k+1)$ 的正整数 j 均有 $\Im(\overline{\dot{x} = a}, j) = 1$ 。

□.

引理 5.10 设 $\Im$ 是 $\widehat{M}$ 的一个 LTL- 模型, 则存在 M 的一个样本模型 $\mathcal{J}$ 使得 $\widehat{\mathcal{J}} = \Im$.

证明:

(I). 对于 $p \in B$, 定义布尔值步函数 $f_p : \mathcal{R}^+ \mapsto \{0, 1\}$ 如下:

$$f_p(s) = \begin{cases} \Im(p, 0) & \text{若 } s = 0; \\ \Im(p', 2nk) & \text{若存在 } k \in \mathcal{N} \text{ 使得 } k < s \leq k+1. \end{cases}$$

(II). 对于 $x \in C$, 定义 $\mathcal{R}^+$ 上的正规函数 f_x 如下:

- 当 $s = 0$ 时, 定义 $f_x(s) ::= \Im(x, 0)/2$;
- 当 $s \neq 0$ 时, 则存在 $k \in \mathcal{N}$ 使得 $k < s \leq k+1$, 由引理 5.9 知存在绝对值不超过 n 的整数 a 使得对任意的满足 $2nk < j \leq 2n(k+1)$ 的正整数 j 有 $\Im(\overline{\dot{x} = a}, j) = 1$ (这里 a 要么是 0 要么它能整除 n)。此时我们定义 $f_x(s) ::= \Im(x', 2nk)/2 + a(s-k)$.

对于任意 $v \in V$, 以上面定义的 f_v 作为 v 的解释, 则我们得到 V 上的一个 LTLC- 模型 $\mathcal{J}$, 由定义 5.10 不难验证 $\mathcal{J} \in Interp(V, n)$ ，下面需要验证 $\widehat{\mathcal{J}} = \Im$ 。

1. 对于 V_d 中的变量 i ，由定义 5.13 及 $\Im$ 满足 $i = 0 \wedge \Box(i^+ = ((i+1) \bmod 2n))$ 知对于任意 $j \in \mathcal{N}$ 有 $\widehat{\mathcal{J}}(i, j) = \Im(i, j)$.

2. 对于任意 $p \in B$ 及任意 $j \in \mathcal{N}$，设 $j = 2nk + i$，其中 $i \in \{0, 1, \ldots, 2n-1\}$。下面分情况证明 $\widehat{\mathcal{J}}(p, j) = \Im(p, j)$，且 $\widehat{\mathcal{J}}(p', j) = \Im(p', j)$.

   - 当 $k = 0$ 且 $i = 0$ 时，$\widehat{\mathcal{J}}(p, j) = \mathcal{J}(p, 0) = f_p(0) = \Im(p, 0) = \Im(p, j)$, 且 $\widehat{\mathcal{J}}(p', j) = \mathcal{J}(p', 0) = f'_p(0) = \Im(p', 0) = \Im(p', j)$。
   - 当 $k = 0$ 且 $i > 0$ 时，由 $\Im \models_{ltl} \Box(V' = V \vee i = 0)$ 及 $\Im \models_{ltl} \Box(B^+ = B')$ 知 $\Im(p, i) = \Im(p', i) = \Im(p', 0)$，于是 $\widehat{\mathcal{J}}(p, j) = \mathcal{J}(p, i/2n) = f_p(i/2n) = \Im(p', 0) = \Im(p, i) = \Im(p, j)$ 且 $\widehat{\mathcal{J}}(p', j) = \mathcal{J}(p', i/2n) = f'_p(i/2n) = f_p(i/2n) = \Im(p', 0) = \Im(p', i) = \Im(p', j)$。
   - 当 $k > 0$ 且 $i = 0$ 时，由 $\Im \models_{ltl} \Box(V' = V \vee i = 0)$ 及 $\Im \models_{ltl} \Box(B^+ = B')$ 知 $\Im(p', 2n(k-1)) = \Im(p, 2n(k-1)+1) = \Im(p', 2n(k-1)+1) = \Im(p, 2n(k-1)+2) = \Im(p', 2n(k-1)+2) = \ldots = \Im(p, 2nk-1) = \Im(p', 2nk-1) = \Im(p, 2nk)$，于是 $\widehat{\mathcal{J}}(p, j) = \mathcal{J}(p, k) = f_p(k) = \Im(p', 2n(k-1)) = \Im(p, 2nk) = \Im(p, j)$，且 $\widehat{\mathcal{J}}(p', j) = \mathcal{J}(p', k) = f'_p(k) = \Im(p', 2nk) = \Im(p', j)$。
   - 当 $k > 0$ 且 $i > 0$ 时，由 $\Im \models_{ltl} \Box(V' = V \vee i = 0)$ 及 $\Im \models_{ltl} \Box(B^+ = B')$ 知 $\Im(p, 2nk + i) = \Im(p', 2nk + i) = \Im(p', 2nk)$，于是 $\widehat{\mathcal{J}}(p, j) = \mathcal{J}(p, k + \frac{i}{2n}) = f_p(k + \frac{i}{2n}) = \Im(p', 2nk) = \Im(p, 2nk + i) = \Im(p, j)$, 且 $\widehat{\mathcal{J}}(p', j) = \mathcal{J}(p', k + \frac{i}{2n}) = f'_p(k + \frac{i}{2n}) = \Im(p', 2nk) = \Im(p', 2nk + i) = \Im(p', j)$,

3. 对于任意 $x \in C$ 及任意 $j \in \mathcal{J}$，设 $j = 2nk + i$，其中 $i \in \{0, 1, \ldots, 2n-1\}$。下面分情况证明 $\widehat{\mathcal{J}}(x, j) = \Im(x, j)$，且 $\widehat{\mathcal{J}}(x', j) = \Im(x', j)$.

   - 当 $k = 0$ 且 $i = 0$ 时，由于 $\Im(x, 0)$ 和 $\Im(x', 0)$ 均属于集合 $\{-2n, \ldots, -2, 0, 2, \ldots, 2n\}$，于是 $\widehat{\mathcal{J}}(x, j) = [\mathcal{J}(x, 0)]_n = [f_x(0)]_n = [\Im(x, 0)/2]_n = \Im(x, j)$, 且 $\widehat{\mathcal{J}}(x', j) = [\mathcal{J}(x', 0)]_n = [f'_p(0)]_n = [\Im(x', 0)/2]_n = \Im(x', 0) = \Im(x', j)$。
   - 当 $k = 0$ 且 $i > 0$ 时，

     (a). 设 f_x 在 $(0, 1)$ 上的导数为 a_0，则由前面 f_x 的定义及引理 5.8 的证明知对任意的满足 $0 < j < 2n$ 的正整数 j 有 $\widehat{\mathcal{J}}(\overline{\dot{x} = a_0}, j) = 1$，且 $\widehat{\mathcal{J}}(x, j) = \widehat{\mathcal{J}}(x', j)$。

     (b). 由前面 f_x 的定义及引理 5.9 知对任意的满足 $0 < j < 2n$ 的正整数 j 有 $\Im(\overline{\dot{x} = a_0}, j) = 1$。由 $\Im$ 是 $\Box(V' = V \vee \bigvee_{j<m} \widehat{\alpha_j})$ 的模型可得 $\Im$ 也是 $\Box(V' = V \vee i = 0)$ 的模型，于是对于 $0 < j < 2n$ 也有 $\Im(x, j) = \Im(x', j)$。

     分别对 $j = 1, 2, 3, \ldots, 2n-1$ 利用 (a) 和 (b) 中关于 $(x')^-$, x, x' 三者的关系便可得对于 $0 < j < 2n$ 有 $\widehat{\mathcal{J}}(x, j) = \Im(x, j)$，且 $\widehat{\mathcal{J}}(x', j) = \Im(x', j)$。

- 当 $k=1$ 且 $i=0$ 时，由于 $\Im(\overline{\dot{x}=a_0},\, j)=1$ 和 $\widehat{\mathcal{J}}(\overline{\dot{x}=a_0},\, j)=1$ 对 $j=2n$ 也成立，应用 $\widehat{\mathcal{J}}(x',2n-1)=\Im(x',2n-1)$ 可得 $\widehat{\mathcal{J}}(x,2n)=\Im(x,2n)$；此外由于 $[\Im(x',2n)/2]_n=\Im(x',2n)$，再利用 f_x 的定义便可得 $\widehat{\mathcal{J}}(x',2n)=[\Im(x',2n)/2]_n=\Im(x',2n)$。
- 当 $k=1$ 且 $i>0$ 时，类似于 $k=0$ 且 $i>0$ 时的情况可得 $\widehat{\mathcal{J}}(x,j)=\Im(x,j)$，且 $\widehat{\mathcal{J}}(x',j)=\Im(x',j)$。
- 类似地对 $k=2,3,\ldots$ 使用上面的步骤 (对 k 使用归纳法)，我们便可得到对任意 $j\in\mathcal{J}$ 有 $\widehat{\mathcal{J}}(x,j)=\Im(x,j)$，且 $\widehat{\mathcal{J}}(x',j)=\Im(x',j)$。

这样就证明了 $\widehat{\mathcal{J}}=\Im$，应用此结论及引理 5.7 、引理 5.9 还可得 $\mathcal{J}$ 是 M 的一个样本模型。

□.

定理 5.1 对于正整数 n, 设 $M\in\mathcal{H}(V,n)$ 且 $\varphi\in Form_H(V,n)$，则 $M\models_{sim}\varphi$ 当且仅当 $\widehat{M}\models_{ltl}\overline{\varphi}$

证明:

(1). 假设 $M\models_{sim}\varphi$。

设 Π 是 $\widehat{M}$ 的任意一个模型，由上面的引理 5.10 知存在 M 的一个样本模型 $\mathcal{J}$ 使得 $\widehat{\mathcal{J}}=\Pi$。由于 $\mathcal{J}$ 是 M 的一个样本模型，且 $M\models_{sim}\varphi$，于是 $\mathcal{J}\models\varphi$，应用引理 5.7 便可得 $\widehat{\mathcal{J}}\models_{ltl}\overline{\varphi}$，即 $\Pi\models_{ltl}\overline{\varphi}$. 于是 $\widehat{M}\models_{ltl}\overline{\varphi}$。

(2). 假设 $\widehat{M}\models_{ltl}\overline{\varphi}$。

设 $\mathcal{J}$ 是 M 的一个样本模型，由引理 5.8 得 $\widehat{\mathcal{J}}$ 是 $\widehat{M}$ 的一个 LTL- 模型；应用假设条件 $\widehat{M}\models_{ltl}\overline{\varphi}$ 便得到 $\widehat{\mathcal{J}}\models_{ltl}\overline{\varphi}$。这样由引理 5.7 便可推出 $\mathcal{J}\models\varphi$，于是 $M\models_{sim}\varphi$。

□.

由于 V_d 中的所有变量的取值域都是有限的 (不超出域 $\{-2n-1,\,-2n,\,\ldots,\,-2,-1,\,0,1,2,\,\ldots,\,2n+1\}$)，故由定理 2.3 知 $\widehat{M}\models_{ltl}\overline{\varphi}$ 的永真性在 LTL 中是可判定的，这样利用 $\widehat{M}\models_{ltl}\overline{\varphi}$ 的永真性判定算法就可判定 $M\models_{sim}\varphi$ 的永真性。

5.5　小结

本章先定义了 LTLC 的一个子语言 LTLC/H，然后利用 LTLC/H 的语法给出了混成系统的一种数学模型：混成模块，并利用利用 LTLC 公式的语义定义了混成模块的语义，第三节给出了利用 LTLC 的语义定义来证明混成系统性质的示例，第四节证明了多速率混成系统的样本模型检查问题是可判定的。

第六章　　相关工作

本章简单介绍一些与本文相关的工作。

6.1　实时逻辑

为了表示实时和混成系统的性质和规范，各种具有不同特征的实时逻辑相继提出，它们大多是由线性时序逻辑 LTL 、分支时序逻辑 CTL(Computation Tree Logic[39]) 和区间时序逻辑 ITL(Interval Temporal Logic[43, 82]) ，通过引入带有时间界限的时序算子（如 $\Diamond_{[0,3]}, \Box_{\leq 5}$ 等）或引入显式的时钟变量，发展而来的。就其表示的性质而言，实时逻辑可分为基于点的实时逻辑和基于区间的实时逻辑；就时间域和计算模型而言，实时逻辑又有离散时间与稠密时间以及分支时间与线性时间之分。下面是常见的一些实时逻辑的主要特点的简略介绍。

- TCTL(Timed Computation Tree Logic[5]) 是一种命题分支时序逻辑，它由 CTL 发展而来，通过引入带有时间界限的时序算子来表示实时性质。 TCTL 公式的语法形式为

  $\varphi ::= p \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, \exists(\varphi_1 \mathcal{U}_{\sim c} \varphi_2) \,|\, \forall(\varphi_1 \mathcal{U}_{\sim c} \varphi_2)$

  这里 p 是一个原子命题， c 是一个非负整数， $\sim$ 是 '$<, \leq, =, \geq, >$' 中的任意一个关系符。

  TCTL 的语义解释可见文献 [5] , 此处从略。TCTL 是一个基于点的时序逻辑, 其时间域为非负实数集 $\mathcal{R}^+$ ，它的可满足性问题是不可判定的。 TCTL 还有另外一种语法定义形式（见 [57, 52]), 是通过引入显式的时钟变量而不是用带有时间界限的时序算子来表示实时性质的, 但时序算子 $\mathcal{U}_{\sim c}$ 可通过时钟变量定义出来。文献 [5] 中证明了时间图（ timed graph, 一种简单的有限控制状态实时系统）关于 TCTL 的模型检查问题是 PSPACE- 完全的, 文献 [57] 给出了有限控制状态实时系统相对于 TCTL 公式的一个符号模型检查算法, 文献 [52] 提出了一种利用 CTL 的模型检查工具来对 TCTL 进行模型检查的方法。

- MITL(Metric Interval Temporal Logic[12]) 是一种命题线性时序逻辑, 它是由 LTL 发展而来的, 它也是通过引入带有时间界限的时序算子来表示实时性质的。MITL 公式的语法形式为

  $\varphi ::= p \,|\, (\neg\varphi) \,|\, (\varphi_1 \wedge \varphi_2) \,|\, (\varphi_1 \mathcal{U}_I \varphi_2)$

  这里 p 是一个原子命题， I 是一个区间 (如 $[1,2], (1,2), (1,2), (1,\infty), [2,\infty)$ 等), 但 I 不能是一个单点区间 (如 $[2,2]$ 等)。

  MITL 的语义解释可参见文献 [12, 91] ， MITL 是一个基于点的时序逻辑, 其时间域为非负实数集 $\mathcal{R}^+$ ， 它的可满足性问题是可判定的 (是 EXPSPACE- 完全的), 它的模型检查问题也是 EXPSPACE- 完全的。文献 [91] 给出了 MITL 的一个完备的公理系统。

- TPTL(Timed Propositional Temporal Logic[15]) 是一个命题线性时序逻辑, 它引入了 '冻结量词' (freeze quantification[15]) 来表示实时性质, 冻结量词 "$x.$" 将当前时间赋给变量 'x' , 对于带有冻结量词 "$x.$" 的 TPTL 公式 $x.\varphi(x)$ 有： $x.\varphi(x)$ 在时刻 t_0 为真当且仅当公式 $\varphi(t_0)$ 在时刻 t_0 为真, 其中 $\varphi(t_0)$ 是将 $\varphi(x)$ 中的 x 替换为时间值 t_0 后的结果。例如, 含有公式冻结量词的公式 $\Box x.(p \Rightarrow \Diamond y.(q \wedge y \leq x+3))$ 可表示这样一个事实： '若 p 在某个时刻为真, 则 q 必在其后不超过 3 个时间单位之内也为真' 。有界时序算子 $\Diamond_{\leq a}.\varphi$ 在 TPTL 中可定义为 $x.\Diamond y.(\varphi \wedge y \leq x+a))$.

  TPTL 的项 π 和公式 φ 的语法形式为：

  $\pi ::= x + c \,|\, c$

  $\varphi ::= p \,|\, (\pi_1 \leq \pi_2) \,|\, \neg\varphi \,|\, (\varphi_1 \wedge \varphi_2) \,|\, O\varphi \,|\, (\varphi_1 \mathcal{U} \varphi_2) \,|\, x.\varphi$

  这里 p 是一个原子命题， x 是一个变量， c 是一个整数常元， O 是 '下一时刻 (next)' 算子。

  TPTL 是一个基于点的时序逻辑, 其时间域为非负整数集 $\mathcal{N}$, 文献 [15] 中证明了时间状态图 (Timed state graph[15]) 关于 TPTL 公式的模型检查问题是 EXPSPACE- 完全的。

- RTTL(Real-Time Temporal Logic[83]) 是一个一阶线性时序逻辑, 它是基于离散时间模型的, 其计算模型是状态的无穷序列。RTTL 很接近于普通的一阶线性时序逻辑 [76] , 它包含时序算子 $\Box$ ， $\Diamond$ 和 $\mathcal{U}$, 但不包含 '下一时刻' 算子 O 。RTTL 中采用显式时钟变量 (注意在离散时间模型中, 时钟变量的值不是连续变化的, 这与我们在第三章中所说的时钟变量不一样) 来表示时间约束条件, 有界时序算子可在其中定义出来, 例如有界时序算子 $\Diamond_{[4,6]}.\varphi$ 在 RTTL 中可定义为 $\forall u.(t = u \Rightarrow \Diamond(\varphi \wedge (u + 4 \leq t \wedge t \leq u + 6)))$ ， 这里 t 是全局时钟， u 是刚性变量。

- TLA$^+$(Temporal Logic of Actions [68, 67]) 也是一个线性离散时间时序逻辑，它与 RTTL 类似，但 TLA$^+$ 中包含有一个使用范围受到限制的 '下一时刻' 算子。TLA$^+$ 也是通过时钟变量（在 TLA$^+$ 中有一个记为 'now' 的时钟变量）来表示实时性质的，此外 L.Lamport[67] 还利用积分的数学性质在 TLA$^+$ 中定义了一个积分算子 $\int$，并利用积分方程来表示混成系统的行为。另外 TLA$^+$ 中还有一组验证规则 (见文献 [68])，可用来（通过演绎）验证实时和混成系统的性质。

- ICTL(Integrator Computation Tree Logic [19]) 是一个连续时间分支时序逻辑，它是 TCTL 的一个推广，它增加了一种称为 'integrator' 的变量，这种变量像一个可以被暂时停止并可被重新启动的时钟，它可以用来记录状态的延迟和累积时间。LCTL 在一些文献中常用来作为混成系统的规范语言，混成系统验证工具 HyTech[19, 48] 就是采用它作为规范语言的，LCTL 是一种基于点的时序逻辑。

- 时段验算 DC(Duration Calculus [95, 96]) 是一种基于连续时间模型的区间时序逻辑，它是在离散时间区间时序逻辑 ITL(Interval Temporal Logic[43, 82]) 的基础上扩充发展而来的，DC 中有一个积分算子 '$\int$' 用来表示状态变量（或状态表达式）在一个区间上的积分（如有 $\int 0 = 0$, $\int 1 = \ell$, 这里 ℓ 表示区间的长度），利用 $\int$ 可以方便地表示出实时和混成系统的一些基于区间的性质，如 DC 公式 '$\Box(\ell \geq 60 \Rightarrow 6 \int Leak \leq \ell)$'（这里状态变量 Leak 表示一个从时间域 $\mathcal{R}^+$ 到集合 {0,1} 上的映射）可表示这样一个关于燃气炉的需求：'在任意一个不短于一分钟的时段上，燃气炉处于漏气状态 (注：对于任意时刻 t，若 $Leak(t) = 1$ 则表明是漏气状态，否则是非漏气状态) 的累积时间不能超过该时段的六分之一'。DC 适合于表示 '累积' 性质，主要用于表示系统较高层次上的规范和需求。DC 有一套相对完备的公理推演系统 [44]，基于 DC 的实时和混成系统验证主要是采用演绎方法，即先将系统和规范统一用 DC 公式表示出来，然后利用推演规则证明它们之间具有蕴涵关系。

- HTL(Hybrid Temporal Logic [60, 59]) 也是一种基于连续时间模型的区间时序逻辑，它与 DC 比较接近，但没有算子 '$\int$'。在 HTL 中，$\overleftarrow{x}$ 和 $\overrightarrow{x}$ 分别用于表示变量 x（变量解释为时间域 $\mathcal{R}^+$ 上的分段光滑实函数）在所讨论区间左端点上的右极限和右端点上的左极限。若以变量 T 表示全局时钟，L 表示燃气的泄漏量（即 $\dot{L}$ 为泄漏速率），则 HTL 公式 $\Box(fin \Rightarrow (\overrightarrow{T} - \overleftarrow{T} \geq 60 \Rightarrow 6(\overrightarrow{L} - \overleftarrow{L}) \leq \overrightarrow{T} - \overleftarrow{T}))$ 可以表示上面关于燃气炉的需求，这里 fin 表示 '有限区间'。文献 [60, 59] 给出了验证混成系统的安全性（以 HTL 作为规范语言）的验证规则。

6.2　实时和混成系统的验证

实时和混成系统的验证从大的方面可分为基于演绎的方法和基于算法的方法两大类。演绎法主要是利用一些公理或验证规则来证明系统具有某些性质，它既可用于有穷状态系统，也可用于无穷状态系统。但演绎法的缺点是验证过程往往需要人来提供验证所需要的一些中间断言，不能全自动进行，这就使得演绎法的效率较低，不太适合于大系统的验证。

TLA^+ 和 DC 是典型的基于演绎法的形式系统， Manna 和 Pnueli 等人也在实时和混成系统的演绎验证方面做了不少工作 [56, 60, 77, 78, 59, 26, 79] ，给出了关于实时和混成系统的一些验证规则和验证方法。现在已发展了不少能支持实时和混成系统演绎验证的工具（交互式定理证明器，如 STeP[26, 79] 、 PVS[88, 92] 等）。

基于算法的方法（模型检查）主要适用于有穷状态系统，它是利用状态空间搜索的方法来确定系统是否具有某些性质。基于算法的方法由于是全自动的，故能用于比较大的系统的形式验证，但却面临着这样两个问题。一是算法复杂性问题，许多验证问题具有很高的时间和 / 或空间复杂度，如何对付搜索空间的状态爆炸是人们必须设法解决的问题，符号模型检查 [80, 30, 19, 57, 54] 、 on-the-fly 算法 [53] 以及抽象与分解 [29, 42, 63] 等都是比较好的对付状态爆炸的办法；二是对于无穷状态系统，能否（或者如何）通过合适的抽象而将其验证问题化归为有穷状态系统下的验证问题。基于稠密时间模型的实时和混成系统由于状态空间是无穷的，故首先就面临如何将无穷状态空间转换为有穷状态空间， R. Alur 等人在 [9, 10] 中提出的利用 '域等价' （ region equivalence ）来将无穷状态空间化为有穷等价类的方法成为此后许多（基于稠密时间模型的）实时和混成系统算法验证工作的基础， R. Alur 等人 [5] 给出了有限控制状态实时系统关于 TCTL 的模型检查算法， T.A. Henzinger 等人 [57] 给出了有限控制状态实时系统关于 TCTL 的一个符号模型检查算法。R. Alur 等人在 [6] 中讨论了线性混成系统（ linear hybrid system ）的符号模型检查，但所给出的计算过程是半可判定的，对一些线性混成系统这个过程可能并不终止。为了确定哪些线性混成系统的模型检查问题是可判定的， T.A. Henzinger 等人在 [45, 51, 64] 等文中讨论了线性混成系统的一些重要子类（如多速率混成系统，矩形混成系统，初始化矩形混成系统等）的可判定性问题。有关实时和混成系统的算法验证和可判定性的较详细的介绍可参看文献 [4, 46, 21, 54, 55] 等。实时和混成系统的模型检查工具有： KRONOS[37, 36] 、 UPPAAL[70, 25] 和 HyTech[19, 48] 等。

第七章　　总　结

本文的主要工作有：

1. 提出了一个具有连续语义的线性时序逻辑 LTLC，定义了它的语法和语义。
2. 通过定义实时模块和混成模块的语法和语义，给出了在 LTLC 中表示实时和混成系统的方法。
3. 给出了利用 LTLC 的语义定义来证明实时和混成系统性质的方法。
4. 证明了 LTLC 的子语言 LTLC/B 和 LTLC/R 的可满足性问题是可判定的。
5. 证明了在样本控制模式下多速率混成系统关于 LTLC/H- 公式的模型检查是可判定的。

作为一种实时逻辑，与其它实时逻辑相比， LTLC 的主要特征是：它既可以作为规范语言用来表示实时和混成系统的性质，又可以作为系统刻画语言用来表示实时和混成系统的实现，而其它实时逻辑则主要是作为规范语言设计的，它们不适合用来描述实时和混成系统的实现。

LTLC 支持实时和混成系统的逐步求精过程，它能表示从静态的需求规范到动态的系统实现之间的不同抽象层次上的系统描述，并且使这些描述具有统一的语义基础，这就使得不同抽象层次的系统描述之间的语义一致性检查在数学上有了可靠和坚实的基础。

此外， LTLC 不但能作为实时和混成系统的规范语言和系统刻画语言，同时也能作为一般反应系统的规范语言和系统刻画语言，并且能支持从非实时系统到实时系统的逐步求精过程。

在 LTLC 中，有穷状态反应系统的逐步求精过程中的一致性检查可使用 LTLC/B 的可满足性判定过程来完成，有穷控制状态实时系统的逐步求精过程中的一致性检查可使用 LTLC/R 的可满足性判定过程来完成。

由于时间的因素，关于 LTLC 还有许多工作有待进一步来完成，如：

1. 建立 LTLC 的公理演绎系统，首先考虑能否对子语言 LTLC/B 和 LTLC/R 建立起一个相对完备的公理系统，由于这两个子语言中不含柔性变量和微分公式，性质较容易刻画和把握，故应当能建立起一组（关于实数系统）相对完备的公理系统。然后再尽可能扩大 LTLC 中可公理化部分的范围。此外还要讨论一些特殊的实时性质（如有界响应性等）在 LTLC 中的验证方法和验证规则，并利用现有验证工具（如 PVS 等）或发展新的辅助验证工具来支持 LTLC 的演绎验证过程。

2. 本文给出的关于子语言 LTLC/B 和 LTLC/R 的判定过程尽管可以直接实现为验证工具，但我们认为直接实现这些过程可能效率并不高，我们想寻求效率更高的算法来实现一个关于时序逻辑 LTLC 的模型检查工具（能检查系统的非空性，系统与系统之间的一致性以及系统与性质之间的一致性）。

3. 继续讨论混成系统的不同子类（如矩形自动机 [21] 等）在样本控制模式下（关于 LTLC 的）的模型检查问题，我们认为样本控制有重要的实际应用价值，应该进一步发展混成系统在样本控制模式下的模型检查方法。

4. 结合具体实例进一步讨论在 LTLC 中如何进行实时和混成系统的逐步求精，给出在 LTLC 中进行逐步求精的原则和方法。

5. 利用 LTLC 对一些实际应用问题（如实时协议和实时控制系统等）进行形式描述、分析和验证。

6. 对 LTLC 进行扩展使能够用它来对实时概率系统 (real-time probabilistic system[34]) 进行形式规范与验证。实时概率系统是一种其转换的发生要满足一定概率分布的实时系统，它可用来模拟现实世界中的一些随机过程。

7. 对 LTLC 进行扩展使得能用它来表示指针、信息隐藏（局部变量）等高级机制。

攻读博士学位期间发表和录用的文章

1. 李广元, 唐稚松, 基于时序逻辑 LTLC 的实时系统模型检查, 软件学报 (已录用) 。

2. 李广元, 唐稚松, 带有时钟变量的线性时序逻辑 LTLC 与实时系统验证, 软件学报 (已录用) 。

3. LI Guangyuan and Tang Zhisong, A Linear Temporal Logic with Continuous Semantics for Hybrid Systems. Proceedings of the International Conference on Software: Theory and Practice (ICS'2000)/ the 16th IFIP World Computer Congress, pp395–402, Beijing, August,2000.

4. LI Guangyuan and Tang Zhisong , Formalization and Verification of Pointers in the Temporal Logic Language XYZ/E Programs. 软件学报, 第 11 卷, 第 3 期, pp285–292, 2000 年 3 月.

5. LI Guangyuan and Tang Zhisong , The Composability Problem of the Semantics of XYZ/BE-communicating Processes. Systems Science and Mathematical Sciences, Vol.12 Suppl. 61–69, May, 1999.

致　谢

在攻读博士学位的三年期间，我得到了许多老师、同学以及朋友的帮助，值此论文完成之际，我衷心感谢所有曾经帮助过我的人们。

首先我想感谢我的导师唐稚松教授，自从九五年认识唐先生以来，我就一直受益于唐老师多方面的启发、关怀和指导。正是在他的指导下，我开始了在时序逻辑和实时系统方面的研究，我的每一点、每一滴的进步，都凝聚着唐老师的心血。从与他的无数次讨论和谈话中，我不但学到了知识，学到了思考问题的方法，而且更重要的是，感受到了老一辈科学家对科学事业永不疲倦的探索精神和对祖国民族的强烈责任感，这一切都将使我受惠终身。在这里请允许我表达我对唐老师的感谢和崇敬之情，并请允许我说一声：谢谢您，唐老师。

其次，我要感谢林惠民教授多年来在学术上对我的指导与帮助，林老师审阅了本文的初稿，并提出了不少宝贵的修正意见。从林老师主持的 '模型检查' 讨论班上，我得到了许多模型检查的知识，并能有机会交流本文最初的一些设想和结果。

在这三年期间，我主要是在开放实验室度过的，实验室的张健博士、柳欣欣博士、蒋颖博士、陈海明博士等都曾给予我许多指导和帮助，在此向他们表示感谢。此外我还要感谢实验室的郭菊卿老师、胡永千老师、庄丽华老师、屠晓平老师等给我提供了一个良好的工作和学习环境。同时我也要感谢研究生部的李彩丽老师三年来给我的帮助和照顾。

感谢赵琛博士、王春江博士、张广泉博士、郭亮博士等 XYZ 组的全体同事，与他们的讨论使我对时序逻辑、形式化方法和软件工程中的许多问题有了更深刻的理解和认识。王春江博士和张广泉博士对本文初稿提出了许多修改意见，使本文增色不少。在本文的写作过程中，小组的各位同事也都给予了各种各样的支持和帮助，使论文能得以顺利完成，在此向他们表示感谢。

我还要感谢我在软件所的同学对我的各种帮助，他们是詹乃军、王栩、陈靖、丁一强、钱军、黄涛、严涛、陈彦云、朱军、刘瑞虹、冯金辉、吴国华、郑新、阮彤、周桓、夏伟忠、詹敏、林鸿、栗阳、程成、刘海峰、王贵林、马恒太等。与他们在一起的日子是愉快的、充实的和令人难以忘怀的，我庆幸我能遇到这么多的好朋友。

最后，我要特别感谢我的妻子李莉女士多年来对我的理解、支持和帮助。三年来她独自一人承担了所有的家务，使我能安心学业并顺利完成这篇论文。

参 考 文 献

[1] Automatica, Special Issue on Hybrid Systems, 35(3), March 1999.

[2] IEEE Transactions on automatic control, Special Issue on Hybrid Systems, 43(4), April 1998.

[3] Proceedings of the IEEE, Special Issue on Hybrid Systems: Theory and Applications, 88(7), July 2000.

[4] R. Alur. Timed Automata. In 11th International Conference on Computer-Aided Verification, Lecture Notes in Computer Science 1633, pp. 8-22, Springer-Verlag, 1999.

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

[6] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. Theoretical Computer Science, 138(1): 3-34, 1995.

[7] R. Alur, C. Courcoubetis, T.A. Henzinger. The observational power of clocks. Proceedings of the Fifth Conference on Concurrency Theory, Lecture Notes in Computer Science 836, pp. 162-177, 1994.

[8] R. Alur, C. Courcoubetis, T.A. Henzinger, and P.-H. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In: Hybrid systems [Grossman et al. ed.], Lecture Notes in Computer Science 736, Springer-Verlag, 1993.

[9] R. Alur and D.L. Dill. Automata for modeling real-time systems. In Automata,Languages and Programming: Proceedings of the 17th ICALP, Lecture Notes in Computer Science 443, pp322-335, 1990.

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

[11] R. Alur and D.L. Dill. Automata-theoretic verification of real-time systems. In Formal Methods for Real-Time Computing, Trends in Software Series, John Wiley & Sons Publishers, pp. 55-82, 1996.

[12] R. Alur, T. Feder, T.A. Henzinger. The benefits of relaxing punctuality. Journal of the ACM 43:116-146, 1996

[13] R. Alur and T.A. Henzinger. Logics and models of real time: a survey. In： Real Time: Theory in Practice, Lecture Notes in Computer Science 600, Springer-Verlag,pp.74-106, 1992.

[14] R. Alur and T.A. Henzinger. Real-time logics: complexity and expressiveness. Information and Computation 104(1):35-77, 1993

[15] R. Alur and T.A. Henzinger. A really temporal logic. Journal of the ACM 41:181-204, 1994.

[16] R. Alur and T. A. Henzinger. Reactive modules. In: Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science(LICS 1996), pp. 207-218, 1996.

[17] R. Alur and T.A. Henzinger. Real-time system = discrete system + clock variables. Software Tools for Technology Transfer 1:86-109, 1997.

[18] R. Alur and T.A. Henzinger. Modularity for timed and hybrid systems. In: CONCUR'97, Lecture Notes in Computer Science 1243, Springer-Verlag, 1997.

[19] R. Alur, T.A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. IEEE Transactions on Software Engineering, 22(3): 181-201, 1996.

[20] R. Alur, T.A. Henzinger, and O.Kupferman. Alternating-time temporal logic (Abstract). Proceedings of the 38th Annual IEEE Symposium on Foundations of Computer Science, pp100-109, 1997.

[21] R.Alur, T. A. Henzinger, G. Lafferriere, and G.J. Pappas. Discrete abstractions of hybrid systems. Proceedings of the IEEE 88:971-984, 2000.

[22] J.W.de Bakker, K.Huizing, W.-P de Rover, and G. Rozenberg, editors. proc. of the REX Workshop "Real-Time: Theory in Practice", Lecture Notes in Computer Science 600, Springer-Verlag, 1991.

[23] H.R. Barringer and D.Gabbay, Executing Temporal Logic: Review and Prospect. Lecture Notes in Computer Science 335(ed.by F.H. Vogt), Springer-Verlag, 1988.

[24] J. Bengtsson, W.O. Griffioen, K.J. Kristoffersen, K.G. Larsen, F. Larson, P. Petterson, and Y. Wang. Automated analysis of an audio-control protocol using UPPAAL. In: CAV'96, Lecture Notes in Computer Science 1102, Springer-Verlag, pp.244-256, 1996.

[25] J. Bengtsson, K.G. Larsen, F. Larson, P. Petterson, and Y. Wang. UPPAAL: a tool suite for automatic verification of real-time systems. In: Hybrid systems III, Lecture Notes in Computer Science 1066, Springer-Verlag, pp.232-243, 1996.

[26] N.Bjϕrner, Z. Manna, H.B. Sipma and T.E. Uribe. Deductive Verification of Real-Time Systems Using STeP. In Proc. of ARTS'97, Lecture Notes in Computer Science 1231, pp. 22-43, Springer-Verlag, May 1997.

[27] A. Bouajjani and Y. Lakhneck. Logic vs. Automata: The hybrid case. In: Hybrid systems III [R. Alur et al. ed.] Lecture Notes in Computer Science 1066, Springer-Verlag, 1995.

[28] E.M. Clarke and E.A. Emerson and A.P. Sistla, Automatic verification of finite-state concurrent systems using temporal-logic specifications, ACM Transactions on Programming Languages and Systems, Vol 8, No.2, 244–263, 1986.

[29] E.Clarke, O. Grumberg, and D. Long. Model checking and abstraction. In Proceedings of 19th ACM Symposium on Principles of Programming Languages,1992.

[30] E.Clarke, O. Grumberg and D. Long. Verification tools for finite-state concurrent systems. In: A Decade of concurrency–Reflections and Perspectives . Lecture Notes in Computer Science, 803, 1994.

[31] E. Clarke and R. Kurshan, Computer-aided verification. IEEE Spectrum Vol.33,No.6,pp61-67, 1996.

[32] E. Clarke, J.M. Wing, et al. Formal Methods: State of the Art and Future Directions, ACM Computing Surveys, Vol. 28, No.4, 1996.

[33] K.M. Chandy and J. Misra, Parallel Program Design, Addison-Wesley, 1988.

[34] C. Courcoubetis and S. Tripakis, Probabilistic model checking: Formalisms and algorithms for discrete and real-time systems. In: Verification of Digital and Hybrid Systems(edited by M.K. Inan and R.P. Kurshan), pp.183-219, Springer, 2000.

[35] J. Davies and S. Schneider, An introduction to timed CSP. Technical Monograph PRG-75, Programming Reseach Group, Oxford University Computing Laboratory, 1989.

[36] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In: Hybrid systems III, Lecture Notes in Computer Science 1066, Springer-Verlag, pp.208-219, 1996.

[37] C. Daws and S. Yovine. Two examples of verification of multirate timed automata using Kronos. In Proceedings of the 16th Annual Real-time Systems Symposium, pp.65-75, IEEE Computer Society Press, 1995.

[38] E.A. Emerson, Temporal and Modal Logic, In 'Handbook of Theoretical Computer Science, Vol. B,(J. van Leenveen, ed.), Elasevier/North-Holland, 1991.

[39] E.A. Emerson and E.M. Clarke. Using branching-time temporal logic to synthesize synchronization skeletons. Science of Computer Programming, 2(3):241-266, 1982.

[40] R, Gerth, D.Peled, M.Vardi, and P.Wolper. Simple on-the-fly automatic verification for linear time temporal logic. In Protocal specification testing and verification, 3-18, Chapman & Hall, 1995.

[41] R.L. Grossman, A. Nerode, A.P. Raven, and H. Rischel,editors. Hybrid Systems, Lecture Notes in Computer Science 736,Springer-Verlag, 1993.

[42] O. Grumberg and D. Long. Model checking and modular verification. ACM Trans. on Prog. and Syst., 16(3):843-871, 1994.

[43] J. Halpern, B. Moszkowski, and Z. Manna. A hardware semantics based on temporal intervals. In Proceedings of the 10th ICALP. Automata, Languages and Programming, Lecture Notes in Computer Science 154, Springer-Verlag, pp278-291, 1983.

[44] M.R. Hansen and C. Zhou. Duration calculus: logical foundations, Formal Aspects of computing, 9(3):283-334, 1997.

[45] T. A. Henzinger. Hybrid automata with finite bisimulations. Proceedings of the 22nd International Colloquium on Automata, Languages, and Programming

(ICALP 1995), Lecture Notes in Computer Science 944,pp. 324-335, Springer-Verlag, 1995.

[46] T.A. Henzinger. The theory of hybrid automata. Proceedings of the 11th Annual IEEE Symposium on Logic in Computer Science (LICS 1996), pp. 278-292, 1996.

[47] T.A. Henzinger. It's about time: real-time logics reviewed. Proceedings of the Ninth International Conference on Concurrency Theory, Lecture Notes in Computer Science 1466, Springer-Verlag, pp. 439-454, 1998.

[48] T.A. Henzinger, P.-H. Ho, and H.Wong-Toi. HyTech: a model checker for hybrid systems. Software Tools for Technology Transfer 1:110-122, Springer, 1997.

[49] T.A. Henzinger, and P. Kopke. Verification methods for the divergent runs of clock systems. Proceedings of the Third International Symposium on Formal Techniques in Real-time and Fault-tolerant Systems (FTRTFT 1994), Lecture Notes in Computer Science 863, pp. 351-372 , Springer-Verlag, 1994.

[50] T.A. Henzinger and P.W. Kopke. Discrete-time control for rectangular hybrid automata. Theoretical Computer Science 221:369-392, 1999.

[51] T.A. Henzinger, P.W. Kopke, A. Puri, and P. Varaiya. What's decidable about hybrid automata? Journal of Computer and System Sciences 57:94–124, 1998.

[52] T.A. Henzinger and O. Kupferman. From quantity to quality. Proceedings of the First International Workshop on Hybrid and Real-time Systems (HART 1997), Lecture Notes in Computer Science 1201, pp. 48-62, Springer-Verlag, 1997.

[53] T. A. Henzinger, O. Kupferman, and M. Y. Vardi. A space-efficient on-the-fly algorithm for real-time model checking. Proceedings of the Seventh International Conference on Concurrency Theory (CONCUR 1996), Lecture Notes in Computer Science 1119, Springer-Verlag, pp.514-529, 1996.

[54] T.A. Henzinger and R.Majumdar. Symbolic model checking for rectangular hybrid systems(Abstract). Proceedings of the Sixth International Workshop on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2000), Lecture Notes in Computer Science 1785, Springer-Verlag, pp. 142-156, 2000,

[55] T.A. Henzinger and R.Majumdar. A classification of symbolic transition systems(Abstract). Proceedings of the 17th International Conference on Theoretical Aspects of Computer Science (STACS 2000), Lecture Notes in Computer Science 1770, Springer-Verlag, pp. 13-34, 2000.

[56] T.A. Henzinger, Z. Manna, and A.Pnueli. Temporal proof methodologies for timed transition systems. Information and Computation 112:273-337, 1994.

[57] T.A. Henzinger, X. Nicollin, J.Sifakis, and S. Yovine. Symbolic model checking for real-time systems. Information and Computation 111:193-244, 1994.

[58] C.A.R. Hoare, Communicating Sequential Processes, Prentice-Hall, London, 1984.

[59] A. Kapur. Interval and point-based approaches to hybrid systems verification, PhD thesis, Stanford University, 1997.

[60] A. Kapur, T.A. Henzinger, Z. Manna, and A. Pnueli. Proving safety properties of hybrid systems. In: Lecture Notes in Computer Science 736, Springer-Verlag, 1993.

[61] R.M. Keller, Formal verification of parallel programs. Comm. ACM, Vol. 19, No. 7, pp371-384, 1976.

[62] Y. Kesten, Z. Manna, H. McGuire, and A. Pnueli. A decision algorithm for full propositional temporal logic. In C. Courcoubetis, editor, 5th conference on computer aided verification, Lecture Notes in Computer Science 697, pp.97-109, Springer-Verlag, 1993.

[63] Y. Kesten and A. Pnueli. Modularization and abstraction: the keys to practical formal verification. In. 23rd Int. Symp. Mathematical Foundations of Computer Science, MFCS-98, Lecture Notes in Computer Science 1450, pp.54-71, Springer-Verlag, 1998.

[64] P.W. Kopke, The theory of rectangular hybrid automata, PhD thesis, Cornell University, 1996.

[65] D. Kozen, Results on the Propositional Mu-Culculus, Theor. Comp. Sci., pp333-354, Dec. 1983.

[66] L. Lamport. Proving the Correctness of multiprocess Programs. IEEE Trans. on Software Eng., SE-3(2):125-143 , 1977.

[67] L. Lamport. Hybrid systems in TLA+. In: Lecture Notes in Computer Science 736, pp77-102, Springer-Verlag, 1993.

[68] L. Lamport. The Temporal Logic of Actions. ACM Trans. Prog. Lang. and Syst.,16(3),pp872-923, 1994.

[69] N. A. Lynch, R. Segala, F.W. Vaandrager, and H.B. Weinberg. Hybrid I/O automata. In: Hybrid Systems III, Lecture Notes in Computer Science 1066, pp496-510, Springer-Verlag, 1996.

[70] K.G. Larsen, P. Petterson, and Y. Wang. Compositional and symbolic model checking for real-time systems. In Proceedings of the 16th Annual Real-time Systems Symposium, pp.76-89, IEEE Computer Society Press, 1995.

[71] O. Maler, Z. Manna, and A. Pnueli. From timed to Hybrid systems. In: Lecture Notes in Computer Science 600, pp447-484, Springer-Verlag, 1992.

[72] Z. Manna and A. Pnueli.How to cook a temporal proof system for your pet language. In Proc. 10th ACM Symp. Princ. of Prog. Lang., pp141-154, 1983.

[73] Z. Manna and A. Pnueli. Verification of concurrent programs: A temporal proof system. In J.W.de Bakker and J.Van Leuwen, editors, Foundations of Computer Science IV, Distributed Systems: Part 2, pp165-255, Mathematical Centre Tracts 159, Center for Mathematics and Computer Science, Amsterdam, 1983.

[74] Z. Manna and A. Pnueli. Completing the Temporal Picture. In Theoretical Computer Science Journal, Vol. 83, No. 1, 1991, pp. 97-130. Also in 16th International Colloquium on Automata, Languages, and Programming, Springer-Verlag, Berlin, pp. 534-558, 1989.

[75] Z. Manna and A. Pnueli.A Hierarchy of Temporal Properties. In 9th Symposium on Principles of Distributed Computing, Quebec, Canada, pp. 377-408, Aug. 1990.

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

[77] Z. Manna and A. Pnueli. Verifying hybrid systems. IN: Lecture Notes in Computer Science 736, pp.4-35 , Springer-Verlag, 1993.

[78] Z. Manna and A. Pnueli. Clocked transition systems. A tribute to Prof. C.S.Tang on his 70th birthday. Stanford CSD Technical Report STAN-CS-TR-96-1566.In Logic and Software Engineering, pp. 3-42, World Scientific Pub., 1996.

[79] Z. Manna and H.B. Simpma. Deductive verification of hybrid systems using SteP. In: Lecture Notes in Computer Science 1386, Springer-Verlag, 1998.

[80] K. L. McMillan. Symbolic model checking - an approach to the state explosion problem. PhD thesis, SCS, Carnegie Mellon University, 1992.

[81] R. Milner, Communication and Concurrency. Prentice-Hall, London,1989.

[82] B. Moszkowski, A temporal logic for multilevel reasoning about hardware. IEEE Transactions on Computers, 18(2):10-19,1985.

[83] J.S. Ostroff. Temporal logic for real-time systems. Research Studies Press Limited, England, 1989.

[84] A. Pnueli , The Temporal Logic of Programs. Proc. 18th IEEE Symp. on Found. of Comp. Sci.,46-57, 1977.

[85] A. Pnueli , Applications of temporal logic to the specification and verification of reactive systems: A survey of current trends. In J.W. de Bakker, W.P.de Roever, and G. Rozenberg, editors, Current Trends in Concurrency, pp510-584, Lecture Notes in Computer Science 224, Springer-Verlag,1986.

[86] A. Pnueli , Verification Engineering: A Future Profession. (A. M. Turing Award Lecture) Sixteenth Annual ACM Symposium on Principles of Distributed Computing, San Diego, August, 1997.

http:// www.wisdom.weizmann.ac.il/ ãmir/ invited-talks.html

[87] A.Pnueli. Formal Verification: All Questions and Some Answers. CS Leading Teachers Course, WIS, May 9, 1999.

http:// www.wisdom.weizmann.ac.il / ãmir/ invited-talks.html

[88] N. Shankar, Verification of real-time systems using PVS. Lecture Notes in Computer Science 697, Springer-Verlag,pp280-281, 1993.

[89] A.P. Sistla and E.M. Clarke. The complexity of propositional linear temporal logics. Journal of the association for computing machinary, Vol.32, No.3, 733-749, 1985.

[90] J. Queillin and J.Sifakis. Specification and verification of concurrent systems in CESAR. In M. Dezani-Ciancaglin and U. Montanari, editors, Fifth International Symposium on Programming, Lecture Notes in Computer Science 137, Springer-Verlag, Berlin, pp337-351, 1981.

[91] J.-F. Raskin, P.-Y. Schobbens, and T. A. Henzinger. Axioms for real-time logics(Abstract). Proceedings of the Ninth International Conference on Concurrency Theory, Lecture Notes in Computer Science 1466, Springer-Verlag,pp219-236, 1998.

[92] J. Vitt and J. Hooman. Assertional specification and verification using PVS of the steam boiler control system.In Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control, Lecture Notes in Computer Science 1165, pp453-472,Springer-Verlag, 1996.

[93] Y. Wang , Real-time behaviour of asynchronous agents. Lecture Notes in Computer Science 458, Springer-Verlag, 1990.

[94] S. Yovine. Model checking timed automata. In: Lecture Notes in Computer Science 1494, 114-152 , Springer-Verlag, 1998.

[95] C. Zhou(周巢尘), C.A.R. Hoare, and A.P. Ravn. A calculus of durations. Information Processing Letters, 40(5):269-276, 1991.

[96] C. Zhou(周巢尘), A.P. Ravn, and M.R. Hansen. An extended duration calculus for hybrid real-time systems. In: Lecture Notes in Computer Science 736, Springer-Verlag, 1993.

[97] C.S. Tang(唐稚松), Toward a Unified logic Base for Programming Language. Tech. Rep. No.: STA-CS-81-865, Dept. of Comp. Sci. Stanford, USA, 1981. A revised version In: Proc. IFIP Congress (ed. R.E.A. Mason). North Holland, 1983.

[98] 唐稚松. 时序逻辑程序设计与软件工程 (上册), 科学出版社, 北京, 1999.

[99] 陆汝钤. 计算机语言的形式语义, 科学出版社, 北京, 1992.