分类号：_____

UDC：_____ 编号：_____

工学硕士学位论文

时间自动机及其应用研究

硕士研究生：孙全勇

指导教师：李健利 副教授

学科、专业：计算机软件与理论

学位论文主审人：马光胜 教授

哈尔滨工程大学

2006年12月

分类号：_____

UDC：_____ 编号：_____

工学硕士学位论文

时间自动机及其应用研究

硕士研究生 ：孙全勇

指导教师 ：李健利 副教授

学位级别 ：工学硕士

学科、专业 ：计算机软件与理论

所在单位 ：计算机科学与技术学院

论文提交日期：2006 年 12 月

论文答辩日期：2007 年 1 月

学位授予单位：哈尔滨工程大学

Classified Index:

U.D.C:

A Dissertation for the Degree of M. Eng

Research on Timed Automata and Its Application

Candidate: Sun Quanyong
Supervisor: Associate Prof. Li Jianli
Academic Degree Applied for: Master of Engineering
Speciality: Computer Software and Theory
Date of Submission: December, 2006
Date of Oral Examination: January, 2007
University: Harbin Engineering University

哈尔滨工程大学
学位论文原创性声明

本人郑重声明：本论文的所有工作，是在导师的指导下，由作者本人独立完成的。有关观点、方法、数据和文献的引用已在文中指出，并与参考文献相对应。除文中已注明引用的内容外，本论文不包含任何其他个人或集体已经公开发表的作品成果。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律结果由本人承担。

作者（签字）：孙金鹏

日期：2007 年 1 月

摘 要

实时系统在军事和民用领域都有着广泛的应用，如何保证这类系统的安全性和可靠性是研究人员广泛关注的问题。模型检测方法是分析和验证实时系统的一种形式化验证方法，而时间自动机是模型检测方法的一个有效工具。时间自动机是对自动机理论的扩展，提供了形式化的方法来建立和分析实时系统的行为，在模型检测方面有着重要的应用。时间自动机已经成为计算机科学理论研究的一个热点。

带自动机是经典时间自动机的一种有穷状态构造，它的状态是由时钟带和位置构成的偶对，因此如何有效的表示和操作时钟带是实现带自动机的一个关键问题。目前已经提出了几种有效的表示方法并应用于实践，如 DBM、CDD 等，DBM 是应用比较广泛的一种数据结构。本文研究时间自动机理论及时钟带的表示和范式化，主要工作如下：

首先，系统的分析时间自动机理论及其相关性质，介绍区域自动机和带自动机。

其次，通过对时钟带 DBM 表示方式的分析，本文分离出一类时钟带，针对这类时钟带的特点，提出一种 DBM 的范式化算法。这种范式化算法相比经典的 Floyd 算法在时间性能上有较大的提高。基于这类时钟带范式化速度较快的特点，其存储空间也可以进行压缩，减少其空间消耗。理论证明及实验结果表明了这种方法的有效性。

最后，本文在理论研究的基础上对一个铁路交叉口控制系统建模并验证其安全性和可靠性。

关键词：模型检测；时间自动机；时钟带；DBM；UPPAAL

ABSTRACT

Real-time systems have been widely applied in military and civilian fields. How to ensure the safety and reliability of such systems is widely concerned by researchers. The method of model checking is a formal verification method that analyzes and verifies real-time systems. Timed automata is an effective tool for the verification of model checking. It is the expansion of automata theory, provides formal method to establish and analyze the behavior of real-time systems. Timed automata is an important application in model checking, and has become the research focus of computer science theory.

Zone automata is the finite construction of classical timed automata. Its state is a pair composed by clock zone and state, so how to represent and operate clock zone is the key realizing zone automata. There have been several effective representations in practice, such as DBM, CDD, etc. DBM is the data structure widely used. This paper researches timed automata theory, the representation and fomalization of clock zone. The main works are as follows:

Firstly, this paper analyzes timed automata theory and related properties systematically, introduces region automata and zone automata.

Secondly, by the analysis of the DBM representations, this paper sorts out a class of clock zone. In view of this kind of clock zone, an algorithm is presented for "normalization" of DBM, which improves time performance compared to the classical "floyd" algorithm. Based on this characteristic, the storage space also can be compressed. Theoretical and experimental results show the effectiveness of this method.

Finally, a control system of railway intersection is modeled, the safety and reliability are verified.

KeyWords: Model Checking; Timed Automata; Clock Zone; DBM; UPPAAL

目　录

第1章 绪论

1.1论文研究的背景和意义

实时系统(Real-time System)是指能对来自所控制的外部环境的交互作用做出及时响应以达到预定目的的计算机系统，是一种定量的反应式系统[1]。过程控制、铁路调度、军事防御、核反应堆等许多计算机控制系统都属于实时系统。这类系统的任何一个错误都会带来重大的经济损失、环境破坏，甚至威胁到生命安全。

如何保证这类系统的安全性和可靠性是研究工作者广泛关注的问题。模型检测方法可以在构建系统前对系统的安全性和可靠性进行验证[3]，以尽早地发现错误。R.Alur 等人在文献[4]中提出的时间自动机理论是一种模型检测的形式化验证方法，为模型检测和实时系统验证以及自动验证工具的开发提供了重要的理论基础。目前已经提出了许多类型的时间自动机用于模型验证，研究时间自动机理论对建立和分析实时系统模型有重要的意义[2,4,5]。

基于时间自动机模型验证的基本思想是穷举它的状态空间[10,11]，而状态空间与时间有紧密联系。因此，如何有效地表示时间是利用时间自动机进行模型验证的一个关键问题。由于时间的取值范围是非负实数集 R^+，因此时间自动机中的状态为无穷多。人们通过划分等价类的方法对时间进行合并，构建出有穷状态的时间自动机模型，如区域自动机和带自动机。带自动机中的状态数目相对较少，因此应用比较多。带自动机中的关键问题之一是要有效的表示时钟带，并方便在其上的相关操作。目前，已经提出了几种时钟带的表示方法并应用于实践，文献[10]提出的 DBM 数据结构可以有效的表示时钟带，但它对于凹集的表示十分复杂，文献[28]中提出的 CDD 数据结构虽比 DBM 多消耗一些时间，但可以对凹集较好表示，节省大量的空间。

本文在对时间自动机及其性质分析的基础上，对目前采用的时钟带 DBM 表示及其相应操作深入研究，分析时钟带的表现形式。把时钟带分类，尝试对某一类时钟带找出效率更高的表示及操作算法，在一定范围内提高模型检

测的效率。本文最后使用基于时间自动机模型的 UPPAAL 工具对一个实时系统建模并验证其安全性和可靠性[14,15]，作为研究时间自动机的一个实例。

1.2 相关领域的研究动态

1.2.1 时间自动机的研究现状

自动机理论 (Automata Theory)主要研究离散数字系统的功能、结构及其关系[16,23]。随着微电子技术和信息科学的发展，自动机理论向信息技术的各个应用领域渗透，为它们提供理论模型、设计技术和运行算法。自动机理论的研究领域主要包括：有穷自动机理论、无穷自动机理论、概率自动机理论、复自动机理论(如细胞自动机和图自动机等)以及抽象自动机理论等[33]。

时间自动机 (Timed Automata,TA)是为解决建模和验证实时系统而对自动机理论的扩展，它是为了适应实时系统的验证需要，由 R.Alur 等人提出的理论，直到今天，这一理论为模型检测和实时系统验证以及自动验证工具的开发提供了重要的基础。时间自动机理论提出十年来，有大量研究人员的研究工作集中在这一领域，主要表现在以下几个方面：

(1)讨论各种接受条件下时间自动机的识别能力与性质，时间正则表达式，时间正则语言与时间自动机的关系等。

(2)对基于时间自动机实时验证和模型检测的理论研究。

(3)引入新的时钟约束和时钟操作，使时间自动机具有不同的性质。

(4)提出各种扩展的时间自动机模型，如离散时间自动机[17]、概率时间自动机[18]、随机时间自动机[19]、时间 I/O 自动机[20]、时间树自动机等[21]。

(5)开发基于时间自动机理论的验证工具。近年来，时间自动机在系统验证领域取得了一系列的成功应用。目前已经开发出的验证工具主要有：UPPAAL、COSPAN、KRONOS 等。

1.2.2 基于时间自动机的模型验证

确保实时系统安全性和可靠性的方法主要有两类：一类是测试方法，另

一类是形式化验证方法。形式化验证方法又分为两类：一类是以逻辑推理为基础的演绎验证方法，另一类是以穷尽搜索为基础的模型检测方法[3]。演绎验证的优点是可以使用归纳的方法来处理无穷状态的问题，缺点是不能做到完全自动化，需要和用户交互，而且推理证明容易出错并且非常耗时。模型检测方法是目前研究的热点之一，它的优点是验证过程可以完全自动化，并且在系统性质未被满足时可以给出反例，帮助用户发现错误，但它也有难以解决的困难，即状态空间爆炸问题。模型检测实质上是状态空间的穷尽搜索，搜索的可穷尽性依赖于为系统建立的有穷状态模型或可归结为有穷状态的模型[9]，时间自动机是模型检测的一个有效工具。

利用时间自动机对实时系统进行规范验证，在近年来得到了广泛发展。在时间自动机中，每一个动作行为都与其发生的时间密切相关，这些发生时间又被状态转换所允许的时间间隔所限制。也就是说，一个动作行为是否能在一定的时间间隔内发生，完全由所有与系统动作行为相关的时间值是否满足每一个与系统状态转换相伴随的时钟约束[34]。

在时间的每点上，时间自动机的状态由当前位置和时钟值唯一确定。无穷多个时间导致状态空间有无穷多个状态，一个区域图是对状态进行合并所得到的状态空间的一个有穷表示。等价类的状态在某种意义上是等价的，但是在构建区域自动机的过程中，可能会出现状态空间组合爆炸问题；此外，构造出来的区域自动机模型可能会存在许多不可达或虽可达但无用的状态。为了消除状态空间组合爆炸问题，目前提出的方法有：(1)在区域自动机的基础上，构造带自动机 Z(A)以减少状态数目；(2)构造一个与带自动机 Z(A)相似但形式比其简单的带自动机 $Z^*(A)$，通过检查 $Z^*(A)$中是否存在状态环来验证 A 所识别的时间语言是否为空；(3)利用迭代算法，通过不断的迭代验证，最终将收敛于一个有穷的正整数，即 $\exists i \in Z^+$，$L(A)=L(A_i)$[41]。

现在已经存在许多使用时间自动机作为模型的工具来描述和验证实时系统。以时间自动机作为行为模型的自动验证工具 UPPAAL 可以有效地对实时系统进行建模、模拟和自动验证，从而确保实时系统在实际运行中的高度正确性。此外，UPPAAL 在算法分析和通讯协议验证方面的应用也十分广泛。UPPAAL 所使用的行为模型是由 R.Alur 和 Dill 提出的时间自动机模型，每一个时间自动机模型都有可能拥有一组整型变量和时钟变量，时间自动机之间

可以通过共享的整型变量和通道进行互相通信。重要的是，UPPAAL 采用限制技术和快速验证技术对解决建模验证过程中出现的状态空间爆炸问题具有良好的效用。UPPAAL 已经成功地应用在许多实时系统的用例研究中，典型的应用领域包括实时控制器性能研究和通讯协议的规范验证。时间自动机在通讯协议自动验证方面已显示了其极大的优点。

1.3 本文的主要工作和内容

本文主要研究时间自动机理论及验证方法，分析时钟带以及状态空间有效表示方法等理论[6]。主要研究工作为以下三个方面：

(1)研究时间自动机理论，分析其相关性质。研究如何将有无穷多状态的时间自动机构造为有穷状态的时间自动机。

(2)时间自动机实现的效率，关键在于如何对时间和状态进行有效的表示，它是时间自动机各种运算的基础，直接关系到整个验证过程所需时间和空间的大小。本文分析带自动机中时钟带的表示方法，重点分析时钟带的 DBM 表示方法，研究在其上的相关运算。本文分离出一类时钟带，研究发现这类时钟带的 DBM 表示方法在空间上可以进行压缩存储；给出这类时钟带范式化的一种新的计算方法，降低范式化过程的时间复杂度，并证明其正确性，最后通过实验数据对比其优越性。

(3)在理论研究的基础上，使用基于时间自动机模型的工具(UPPAAL)对一个铁路交叉口控制系统建模并验证系统的正确性和可靠性。

第2章 时间自动机模型

2.1 时间自动机概述

2.1.1 ω-有穷自动机

由于时间自动机的状态与某一事件发生时间相关，而时间是单调无限递增的，因此，时间自动机所接受的语言就是时间序列上的无限事件序列。首先简要介绍一下相关的 ω-有穷自动机理论。

ω-字是某个有穷字母表 Σ 中的字符组成的无穷序列，ω-有穷自动机是 Büchi 在 1962 年提出的一种识别 ω-字的抽象数学模型[22]。

用 Σ^{ω} 表示有穷字母表 Σ 上的无穷字的集合，则有穷字母表 Σ 上的 ω-语言是 Σ^{ω} 的子集，即 ω-语言是由无穷字的集合组成的。ω-有穷自动机提供了对特定类型的 ω-语言的有穷表示，它同非确定有穷自动机等价。为了处理无穷的输入字，对其接受条件作相应的改变。这里讨论两种类型的 ω-自动机：ω-Büchi 自动机与 ω-Muller 自动机。这些理论的提出为物理过程的自动验证奠定了基础。

定义 2.1(转换系统)[4,5] 一个转换系统 A 是一个四元组(Σ, S, S_0, E)，其中：

(1)Σ 是一个有穷字符集合；

(2)S 是一个有穷的状态集合；

(3)S_0 ($S_0 \subseteq S$)是一个初始状态集合；

(4)$E \subseteq S \times \Sigma \times S$ 是一个状态转移集合。

转换系统 A 从一个初始状态 s_0 ($s_0 \in S_0$)开始，E 中的一个状态转移<s, a, s'>表示转换系统 A 在输入字符 a 时，从状态 s 到状态 s'的一个转移。通常，对于定义在字符集 Σ 上的一个无穷字 $\sigma=\sigma_1\sigma_2\ldots$，称 $r: s_0 \xrightarrow{\sigma_1} s_1 \xrightarrow{\sigma_2} s_2 \xrightarrow{\sigma_3} \cdots$ 是转换系统 A 定义在字 σ 上的一个运行，其中 $s_0 \in S_0$，$<s_{i-1},\sigma_i,s_i>(\forall i \geq 1)$。对此运行，集合 inf(r)由状态 s 组成，$s \in S$，使得 $s=s_i$，其中 s 出现无穷多次，$i \geq 0$，即 $\inf(r)=\{s \in S | s=s_i$，对于无穷多的 $i \geq 0\}$。

对转换系统加入不同的接受条件，就可以定义不同类型的 ω-自动机。

定义2.2(ω-Büchi自动机)[7] 一个ω-Büchi自动机是一个五元组A=(Σ, S, S_0, E,F)。其中Σ, S, S_0, E的含义与定义2.1相同，F⊆S是接受状态集合。 A在Σ上的字σ∈Σ^ω的运行r是可接受的，当且仅当inf (r)∩F≠Φ，换句话说，运行r是可接受的当且仅当F中的某些状态在r中重复无穷多次[40]。由ω-Büchi自动机 A 所接受的语言 L(A)是 A 接受的 Σ^ω 上的无穷字 σ 的集合。即：L(A)={σ∈Σ^ω|存在A在σ上的运行r，使得inf(r)∩F≠Φ}。

例2.1 图2.1所示为ω-Büchi自动机，其中Σ={a,b}，S={s_0,s_1}, S_0={s_0}，E如图所示，F={s_1}。

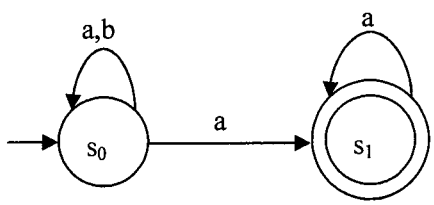


图2.1 ω-Büchi自动机

它的运行 $r=s_0 \xrightarrow{\sigma_1} s_0 \xrightarrow{\sigma_2} s_0 \cdots \xrightarrow{\sigma_n} s_0 \xrightarrow{a} s_1 \xrightarrow{a} s_1 \xrightarrow{a} \cdots$，其中$\sigma_i \in \{a,b\}$，$1 \le i \le n$，它所识别的语言为 $L_0=\{a+b\}^* a^\omega$，即自动机接受所有只含有穷多个b的无穷串。

定义 2.3(ω-正则语言)[7] 一个 ω-语言是 ω-正则语言当且仅当它被某个ω-Büchi自动机接受。

ω-正则语言有如下的两个性质：

(1)ω-正则语言类在布尔运算下封闭。

(2)对任意两个给定的ω-正则语言L_1和L_2，如下问题可判定：L_1为空、有穷或无穷；$L_1=L_2$；$L_1 \subseteq L_2$；$L_1 \cap L_2=\Phi$。

定义2.4(ω-Muller自动机)[7] 一个ω-Muller自动机是一个五元组A=(Σ, S, S_0, E,F)。其中Σ, S, S_0, E的含义与定义2.1相同，F⊆2^S是接受状态集族。 A在Σ上的字σ∈Σ^ω的运行r是可接受的，当且仅当inf(r)∈F，换句话说，运行r是可接受的当且仅当r中重复无穷多次的状态集合是接受状态集族F中的一个元素。由ω-Muller自动机A所接受的语言为L(A)={σ∈Σ^ω|存在A在σ上的运行r，使得inf(r)∈F}。

例2.2 图2.2所示为ω-Muller自动机，其中Σ={a,b}，S={s_0,s_1}, S_0={s_0}，

E 如图所示，定义 F={{s_1}}。

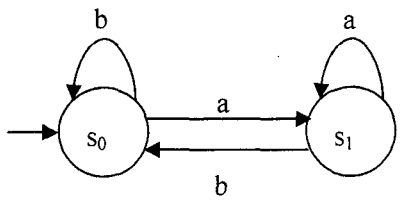


图 2.2 ω-Muller 自动机

2.1.2 时间语言

下面讨论带有时间特性的系统，用时间字表示实时系统的一个动作行为，即定义在事件字符集上的一个时间字与系统的一个动作行为相对应。在稠密时间自动机模型下，通常用非负实数集 R^+作为系统时钟变量的取值范围。

定义 2.5(时间序列)[4] 时间序列 $\tau=\tau_1\tau_2\ldots\tau_i\ldots$是时间值 τ_i 的无穷序列，$\tau_i\in R^+$且 $\tau_i>0$，满足以下条件：

(1)单调性：τ 是严格单调递增的，即 $\tau_i<\tau_{i+1}$，$i\geq 1$；

(2)无界性：对$\forall t\in R^+$，存在某个 $i\geq 1$，使得 $\tau_i>t$。

有穷字符集 Σ 上的时间字是一个偶对(σ,τ)，$\sigma=\sigma_1\sigma_2\ldots$是字符集 Σ 上的无穷字。

定义 2.6(时间语言)[4] Σ 上的时间语言是 Σ 上的时间字集合。

可以将时间字(σ,τ)看作时间自动机的一个输入，它表示自动机在 τ 时刻输入字符 σ。如果用时间自动机来模拟实时系统的行为，那么时间字($\sigma_1\sigma_2$..., $\tau_1\tau_2$...)中的一个符号分量 σ_i 表示系统发生的一个事件，则相应的分量 τ_i 表示该事件发生的时间。在某些特定情况下，时间字序列中许多连续发生的事件可能有相同的时间值。为了有效地处理这种情况，可以对时间字的形式化定义作一些改动，即不要求时间序列中的时间值是严格单调递增的，而只要求时间序列是单调递增的即可(也就是说，对所有 $i\geq 1$，有 $\tau_i\leq\tau_{i+1}$)。

例 2.3 令 Σ={a,b}，定义时间语言 L 为满足以下条件的所有时间字(σ,τ)组成的集合：在每一个时间字序列中，字母 a 和 b 交替出现，并且每一对连续出现的字母 a 和 b，它们之间的时间差满足递增关系。那么时间语言 L 就

可以表示为：$L=\{((ab)^{\omega},\tau)|\forall i((\tau_{2i}-\tau_{2i-1})<(\tau_{2i+2}-\tau_{2i+1}))\}$。

2.1.3 时钟约束和时钟解释

为了表示带有时间约束的系统行为，考虑被有穷“时钟”集合(实数值)扩大的有穷图。图的顶点称为“位置”(locations)，边称为“转换”(switches)，转换是瞬时的，时间可以在位置流逝。时钟可以随任何转换被同时复位为零。在任一瞬间，时钟值等于从上次它被复位后流逝的时间。对每个转换关联一个时钟约束，称为“转换条件”，并且要求仅当当前时钟值满足这个约束时转换才可能发生。对每一个位置关联一个称为该位置的“不变式”的时钟约束，并且要求仅当该位置的不变式为真时时间才可以在该位置流逝。在正式定义时间自动机之前，考虑一个简单的例子。

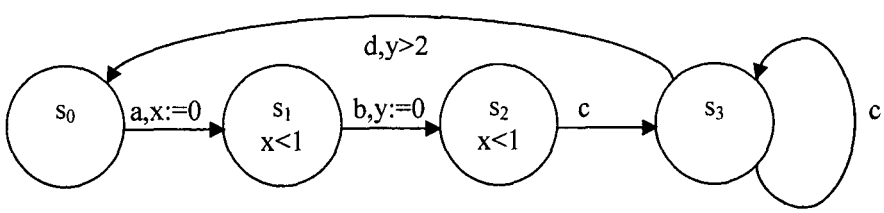


图 2.3 带有两个时钟的时间自动机

考虑图 2.3 中带有两个时钟的时间自动机。每次在标记 a 下系统从 s_0 转换到 s_1 时，时钟 x 置为 0。关联在位置 s_1 和 s_2 上的不变式(x<1)确保从 s_2 到 s_3 的标记为 c 的转换发生在先前的 a 发生后一个时间单位内发生。在从 s_1 到 s_2 的标记为 b 的转换时另一个独立的时钟 y 复位，并且从 s_3 到 s_0 标记为 d 的转换时检查 y 的值，确保在 b 和接下来的 d 之间的时间延迟总是大于 2。注意，为了约束 a 与 c 之间的延迟和 b 与 d 之间的延迟，系统没有在 a 和接下来的 b 之间增加任何明确的边界，这是有多时钟的一个重要的优点，它可以彼此独立的被设置。

为了形式化的定义时间自动机，需要表示什么类型的时间约束被允许作为不变式和转换条件，下面给出时钟解释和时钟约束的概念。

定义 2.7(时钟解释)[5] 考虑一个有穷的变量集合 X，称为时钟集,一个在 X 上的时钟解释是一个映射 $v:X\rightarrow T$，它给每个时钟分配一个时间值。X 上所有

有时钟解释的集合记为 T^X。令 $t\in T$，时钟解释 v+t 被定义为$(v+t)(x)=v(x)+t$，$\forall x\in X$。当$|X|=n$ 时，对时钟解释也可使用符号$(\alpha_i)_{1\le i\le n}$，表示 $v(x_i)=\alpha_i$，对于 X 的一个子集 λ，用$[\lambda\leftarrow 0]v$ 表示这样的时钟解释，对于每一个 $x\in\lambda$，$([\lambda\leftarrow 0]v)(x)=0$，并且对于每个 $x\in X/\lambda$，$([\lambda\leftarrow 0]v)(x)=v(x)$。

定义 2.8(时钟约束)[5] 给定一个时钟集合 X，在 X 上的时钟约束集合按如下的文法定义：

$\Phi::=x\sim c|x-y\sim c|\varphi_1\wedge\varphi_2|true$，其中 $x，y\in X，c\in Z，\sim\in\{<,\le,=,\ge,>\}$

当时钟解释 v 满足时钟约束 φ 时，记 $v|=\varphi$。

2.1.4 时间自动机的语法和语义

定义 2.9(时间自动机)[4,5] 时间自动机 A 是一个六元组$<\Sigma,S,S_0,X,I,E>$，其中：

(1)Σ 是一个有穷标记集合；

(2)S 是一个有穷位置集合；

(3)$S_0\subseteq L$ 是一个初始位置集合；

(4)X 是一个有穷时钟集合；

(5)I 是一个映射，它给每个位置 s 指定 Φ(X)中一些时钟约束；

(6)$E\subseteq S\times\Sigma\times 2^X\times\Phi(X)\times S$ 是一个转换集合，一个转换$<s,a,\oint,\lambda,s'>$表示在符号 a 下从位置 s 到位置 s'的一条边。$\oint$ 是一个 X 上的时钟约束，它指定什么时候可以发生转换，集合 $\lambda\subseteq X$ 给出在这次转换中被复位的时钟。

时间自动机 A 的语义由一个与它相关的转换系统 S_A 定义。S_A 的一个状态是一个偶对(s,v)，s 是 A 的一个位置，v 是 X 的一个时钟解释，使得 v 满足不变式 I(s)。A 的所有状态的集合表示为 Q_A。如果 s 是 A 的一个初始位置并且 v(x)=0，那么状态(v,s)是一个初始状态。在 S_A 中有两类转换：

(1)时间流逝：对于一个状态(s,v)和一个实数值的时间增量 $\delta\ge 0$，如果对于所有的 $0\le\delta'\le\delta$，$v+\delta'$都满足不变式 I(s)，那么：

$$(s,v)\xrightarrow{\delta}(s,v+\delta)$$

(2)位置转换：对于一个状态(s,v)和一个转换$<s,a,\oint,\lambda,s'>$，使得 v 满足 $\oint$，那么$(s,v)\xrightarrow{a}(s',v[\lambda:=0])$。

这样，S_A 就是具有标记集 $\Sigma \cup R^+$的一个转换系统。例如，图 2.3 所示的时间自动机，与之相应的转换系统的状态空间是$\{s_0,s_1,s_2,s_3\}\times R^{2+}$，符号集是$\{a,b,c,d\} \cup R^+$，转换的例子如下：

$$(s_0,0,0) \xrightarrow{1.2} (s_0,1.2,1.2) \xrightarrow{a} (s_1,0,1.2) \xrightarrow{0.7} (s_1,0.7,1.9) \xrightarrow{b} (s_2,0.7,0)$$

如果 $q \xrightarrow{\delta} q'$并且 $q' \xrightarrow{\varepsilon} q''$，那么 $q \xrightarrow{\delta+\varepsilon} q''$。

一个比较复杂的系统往往由多个互相通信并同步的若干子系统组成，对这些子系统建模时间自动机后需要将它们合并为一个时间自动机，下面给出一个时间自动机积的构造方法[5]。令 $A_1=<\Sigma_1,S_1,S_1^0,X_1,I_1,E_1>$和 $A_2=<\Sigma_2,S_2,S_2^0,X_2,I_2,E_2>$是两个时间自动机。假设时钟集合 X_1 和 X_2 不相交。那么 A_1 和 A_2 的积自动机 $A_1 \| A_2$ 是 $<\Sigma_1 \cup \Sigma_2,S_1 \times S_2,S_1^0 \times S_2^0,X_1 \cup X_2,I,E>$。其中，$I(s_1,s_2)=I(s_1) \wedge I(s_2)$，转换定义如下：

(1)对于 $a \in \Sigma_1 \cap \Sigma_2$，$E_1$ 中每个转换$<s_1,a_1,\phi_1,\lambda_1,s'_1>$和 E_2 中每个转换$<s_2,a_2,\phi_2,\lambda_2,s'_2>$，E 有转换$<(s_1,s_2),a,\phi_1 \wedge \phi_2,\lambda_1 \cup \lambda_2,(s'_1,s'_2)>$。

(2)对于 $a \in \Sigma_1/\Sigma_2$，E_1 中每个转换$<s,a,\phi,\lambda,s'>$和 S_2 中的每个位置 t，E 有转换$<(s,t),a,\phi,\lambda,(s',t)>$。

(3)对于 $a \in \Sigma_2/\Sigma_1$，E_2 中每个转换$<s,a,\phi,\lambda,s'>$和 S_1 中的每个位置 t，E 有转换$<(t,s),a,\phi,\lambda,(t,s')>$。

这样，积的位置是成员位置组成的偶对，一个复合位置的不变式是成员位置不变式的联合，转换由同一个标记进行同步转换得到。

例 2.4 如图 2.4 对两个时间自动机求积。

在 2.1.1 中给出了 ω-Büchi 自动机和 ω-Muller 自动机的定义，在此基础上将其扩展为时间自动机就相应的得到时间 Büchi 自动机(简称为 TBA)和时间 Muller 自动机(简称为 TMA)。由时间 Büchi 自动机接受的语言类称为时间正则语言。

定理 2.1 时间语言 L 是时间正则语言当且仅当对某个 TBA A 有 L=L(A)。

证明过程见文献[4]。

定义 2.10(确定的时间自动机)[5] 一个时间自动机被称为确定的当且仅当：

(1)它只有一个初始位置；

(2)对所有的 $s \in L$，$a \in \Sigma$，对每一对形如$<s,a,\phi_1,\lambda_1,s'_1>$和$<s,a,\phi_2,\lambda_2,s'_2>$

的转换，时钟约束 ϕ_1 和 ϕ_2 是互斥的(即 $\phi_1 \wedge \phi_2$ 是不满足的)。

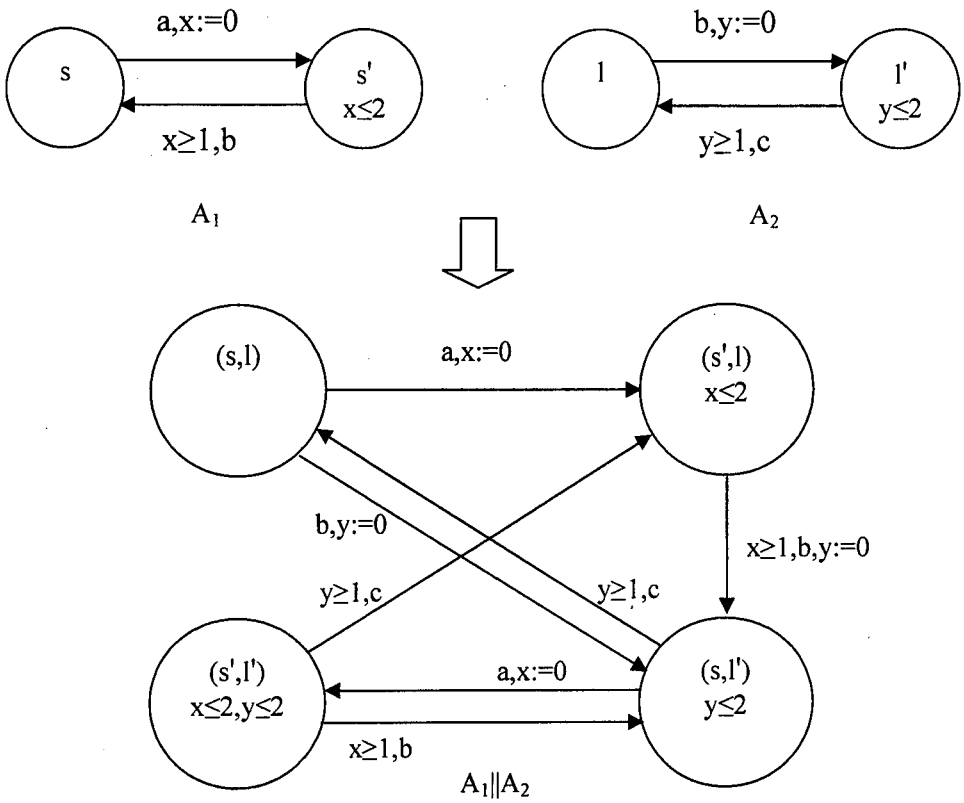


图 2.4 积时间自动机的构造

2.2 时间语言类性质

本文所研究的时间自动机模型执行的是无穷的事件序列，而不是有穷状态。但其与以有穷状态为基础的模型理论仅在细节上有些差异。在这些理论基础下，才能用一系列的时间与执行踪迹相比较，从而加入时间因素。下面对时间语言的性质做一简单说明。

2.2.1 判空复杂性

时间 Büchi 自动机判空算法的复杂性是时钟数和时间约束常量的几何级数，复杂性的爆炸式增长似乎是不可避免的。已经知道线性有界自动机的可

接受问题是多项式完全问题(P 类问题)。可以将线性有界自动机的可接受问题转化为时间 Büchi 自动机的判空问题，从而证明判空问题是 P 下界。同样可以通过论证算法可在多项式空间内实现，来证明该问题是多项式完全的。

定理 2.2 给定时间自动机 A 的判空问题是个多项式完全问题。

证明过程见文献[4]。

2.2.2 包含关系

Büchi 自动机的语言包含问题可以在多项式时间内解决。但是，却不存在确定过程来检查两个时间 Büchi 自动机的语言是否存在包含关系。这是使用时间自动机作为有穷实时系统的语言表述工具、进行自动验证的障碍。

对于普通意义的自动机(或者对于 Büchi 自动机)，检验一个自动机的语言是否包含于另一个，可以通过判断第一个自动机和第二个自动机的补的积是否为空，但这个方法不能用于 TBA，因为对于一般意义下的 TBA 求其补是不可能的。事实上，并不存在用于检验一个 TBA 语言是否包含于另一个的算法。

但对于确定的时间自动机可以很容易的对其求补，因为一个确定时间自动机在一个给定的时间字上至多有一个运行。判空的算法可以用于检验一个 TBA 语言是否包含于另一个确定的 TBA 语言中。

2.2.3 等价关系

自动机等价性的自然定义是使用其被接受语言的等价性。

下面的定义也说明了自动机的等价性。

定义 2.11(等价的时间自动机)[5]字母表 Σ 上的时间 Büchi 自动机 A_1、A_2，$A_1 \sim_1 A_2$ 当且仅当 $L(A_1)=L(A_2)$；$A_1 \sim_2 A_2$ 当且仅当 Σ 上的所有时间自动机 A，当 $L(A)\cap L(A_2)$为空时，$L(A)\cap L(A_1)$为空。

对在补运算下封闭的自动机类，上述二个定义一致。由于其对补操作的不封闭性，二个定义与时间正则语言类不同。事实上 $A_1 \sim_1 A_2$ 蕴涵着 $A_1 \sim_2 A_2$。反之不成立。

2.2.4 时钟约束的选择

时钟约束允许原子操作、时钟值与有理数常量比较、布尔连接。时间自动机仅表达关于转换间的常量的界限。扩展时钟约束的定义，允许公式包含两个时钟，例如 $x \leq y+c$，特别是时钟约束集合 $\Phi(X)$。对时钟 x，$y \in X$，$c \in Q$，允许使用公式 $x \leq y+c$ 和 $x+c \leq y$。非时间构造可以很容易的处理此扩展。仅需要重新定义关于时钟解释的等价关系。除去前述条件，需要两个等价的时钟解释，它们与在所有时钟约束出现的子公式一致。同样可以证明，时钟约束的扩展并不能对时间自动机的表达能力有所增加。

在时钟约束中允许加法存在，能够书写形如 $x+y \leq x'+y'$ 的时钟约束。这样，就可以允许自动机同不同的延迟比较，增加了形式化的表达能力(注意不是自动机的表达能力)。具有“+”的时钟约束太强，不能被有穷状态系统实现。即使要求所有事件都在整值时间内发生(即离散的时间模型)，为了检查两个符号的间隔与下两个符号间隔是否相同，自动机必须有一个无穷的存储空间。因此，使用有穷资源，自动机仅能比较而不能记录延迟。事实上，在时间约束中引入加法，使得时间自动机的空问题不可判定。

2.2.5 封闭属性

考虑确定时间自动机的封闭属性，被确定时间 Muller 自动机接受的语言类在布尔操作下封闭。

定理 2.3 确定的时间 Muller 自动机接受的语言类对并、交、补操作封闭。

考虑确定时间 Büchi 自动机的封闭性质，有无穷个 a 可以被确定 Büchi 自动机所表述，但其补，仅有有穷个 a 不能被确定 Büchi 自动机所表述。不能期望确定时间 Büchi 自动机类对补封闭，但是由于每个确定时间 Büchi 自动机可视为一个确定时间 Muller 自动机，确定时间 Büchi 自动机的补被一个确定时间 Muller 自动机接受。定理 2.4 描述了封闭性质。

定理 2.4 被确定时间 Büchi 自动机接受的时间语言类的并、交操作封闭，但其补操作不封闭。确定时间 Büchi 自动机的补语言，被某个确定时间 Muller 自动机所接受。证明过程见文献[4]。

2.2.6 表达能力

比较不同类型的时间自动机，仅仅通过重写接受条件，每一个确定时间 Büchi 自动机可以轻易地用确定时间 Muller 自动机表达。首先，每一个 ω-正则语言和确定 Muller 自动机的表达能力相同。因此，也同确定时间 Muller 自动机的表达能力相同。换句话说，确定的 Büchi 自动机比确定的 Muller 自动机的表达能力弱得多。某些 ω-正则语言不能被确定 Büchi 自动机表述。同样不能使用确定时间 Büchi 自动机表示此类语言。从而，确定时间 Büchi 自动机比确定时间 Muller 自动机的表达能力也要弱。事实上确定时间 Muller 自动机对补封闭，而确定时间 Büchi 自动机则不行。有结论： 对 ω-正则语言，时间语言$\{(\sigma,\tau)|\sigma \in L\}$可以被某个确定时间 Büchi 自动机接受当且仅当 L 可被某些确定 Büichi 自动机接受。表 2.1 显示了不同时间语言类之间的包含关系和封闭性。

表 2.1 时间自动机类之间的包含关系和封闭性

时间语言类	封闭操作
TMA=TBA	并、交
∪	
DTMA(确定时间 Muller 自动机)	并、交、补
∪	
DTBA(确定时间 Büchi 自动机)	并、交

2.3 可达性问题分析

在利用时间自动机模型对系统进行验证时，主要是安全可靠性验证，而安全可靠性验证可归约为时间自动机的可达性分析[22,23]，因此可达性的问题是时间自动机的主要研究内容之一。

2.3.1 可达性定义

时间自动机的可达性问题可以描述为对于给定时间自动机的两个状态是否存在从其中一个状态到达另外一个状态的执行序列。

下面给出基于一般点的可达性定义。

定义 2.12(点的可达性)[2] 假设 s_0，s_f属于位置集合 S 并且 v_0，v_f是时钟的赋值，如果状态(s_f, v_f)对于状态(s_0,v_0)是可达的，那么一定存在一个自然数 n 和一个执行序列从状态(s_0, v_0)出发到达(s_f, v_f)。即$(s_0,v_0) \xrightarrow{\tau_1} (s_1, v_1)\ldots (s_{n-1}, v_{n-1}) \xrightarrow{\tau_n} (s_f,v_f)$，并且在该式子中对于所有的 $\tau_i \in \Sigma \cup R^+$。

上面给出了一种时间自动机的时间符号转换图的最一般可达性定义。由于状态个数是无穷的，可以将性质相同的点组成一个集合。下面给出可达性的定义。

定义 2.13(时间区域)[2] 一个时间区域为满足 $\Phi(X)$中某个约束的时间值集合，给定 $\varphi \in \Phi(X)$，将满足时钟约束 φ 的所有时钟值的集合记为 D。$D=\{v| v|=\varphi\}$，称 φ 为时间区域 D 的特征约束。

定义 2.14(时间区域的可达性)[2] 假设 s_0，s_f属于位置集合 S 并且 D_0，D_f是时间区域，也就是点的集合。如果状态(s_f,D_f)对于状态(s_0,D_0)是可达的，那么一定存在一个自然数 n 和一个执行序列从状态(s_0,D_0)出发到达(s_f,D_f)，如下描述$(s_0, D_0) \xrightarrow{\tau_1} (s_1, D_1)\ldots(s_{n-1}, D_{n-1}) \xrightarrow{\tau_n} (s_f, D_f)$，并且在该式子中对于所有的 $\tau_i \in \Sigma \cup R^+$。

通过这个定义可以看到需要合并系统的点而组成时间区域即点的集合。合并的方法依赖于系统转换边上的时钟约束。

定理 2.5 给定元组(S,D)和(S',D')，设$(S,D)\rightarrow(S',D')$为从状态(S,D)到状态(S',D')的迁移，当且仅当$\forall v \in D, \exists v' \in D'$,有$(S,v)\rightarrow(S',v')$且$\exists v \in D, \forall v' \in D'$也有$(S,v)\rightarrow(S',v')$。证明过程见文献[6]。

可达性问题的判断，简单的说就是在一个系统中给定两个状态，判断它们之间是否存在一个执行。因此这种问题的判定可以被认为是在一个系统的所有状态中寻找特定的状态的方法。状态之间转换依赖于转换系统中的转换关系。转换关系描述了怎样从一个状态转移到另外一个状态。也就是说，系统的状态空间可以构建为一个图。因此，搜索可以分为两种：向前搜索和向

后搜索。向前搜索指的是遍历状态空间时通过访问状态的后继进行，而向后搜索指的是遍历状态空间时通过访问状态的前趋来进行。本文所研究搜索方式为向前搜索。

下面给出关于时间区域后继和时间区域前趋的定义。

定义 2.15(时间后继、前趋)[2] 给定时间自动机的两个元组(S,D)和(S',D')，如果存在一个执行满足(S,D)→…→(S',D')，则称(S',D')是(S,D)的时间后继，反过来称(S,D)为(S',D')的时间前趋。如果存在执行(S,D)→(S',D'),则称(S',D')是(S,D)的直接时间后继，反过来(S,D)为(S',D')的直接时间前趋。

2.3.2 可达性分析方法

时间自动机由于时钟值的取值范围为非负实数，所以它的状态空间个数是无穷的，如果直接利用这种无穷的状态进行可达性问题的判断是不可行的。因此，必须构造出只有有穷个状态的时间自动机。

目前可达性问题的判定方法主要有以下几种[35]。第一种方法是在基于时钟区域划分的区域自动机中进行可达性判断。这种方法首先要构建时间自动机对应的区域自动机，由于时钟区域的个数与自动机中时钟集合 X 中元素的个数呈指数关系，在规模较大的系统中，不可避免的会产生时钟区域的数目过大的问题，所以这种可达性判定方法不利于在大规模问题上实施。另一种方法是基于时钟区域代表的可达性判断方法，这种方法依然存在状态爆炸的问题。但由于它是纯粹离散的系统，它的优点是可以借助一些著名的离散化系统所常用的算法和数据结构简化问题。还有一种方法是先不构建所有的状态空间，而利用时间自动机上的时钟约束来进行可达性问题的判断。

2.4 本章小结

本章首先概述了 ω-有穷自动机理论，为了处理无穷的输入字，对 ω-有穷自动机的接受条件作相应改变，得到了 ω-Büchi 自动机与 ω-Muller 自动机，这些理论为处理带有时间特性的实时系统的时间自动机提供了理论基础。

为了处理带有时间特性的实时系统，定义了时间语言、时钟约束、时钟

解释等概念，给出了时间自动机的形式化定义，并说明了时间自动机的语法和语义。然后对时间语言类的性质做了简单说明，这些性质对时间自动机的深入研究有重要的指导意义。最后引入可达性问题，可达性问题有着重要的作用：首先，在验证时可以将系统的某个不安全性质构建为系统的一个状态，如果该状态不满足可达性，则说明系统在该性质下是安全的；相反，如果可达性问题满足，则说明系统存在该不安全性质的隐患。另一方面，在对其他类性质的验证问题上，也多是以可达性的分析为基础的。

第3章 有穷状态时间自动机的构造

3.1 时间抽象转换系统

在第二章已经介绍了时间自动机的基本概念，了解到一个时间自动机的转换系统 S_A 是无穷的，那么在使用时间自动机进行验证时就应该构造有穷商(quotients)[4,38]。

一个时间自动机 A 的转换系统 S_A 有无穷多个状态和无穷多个标记。第一步，定义另外一个转换系统，被称为时间抽象转换系统，表示为 U_A，要求它的转换通过隐藏表示时间增量的符号，只用 Σ 中的标记来标注。U_A 的状态空间等于 S_A 的状态空间 Q_A。U_A 的初始状态集合等于 S_A 的初始状态集合。U_A 的标记集合等于 A 的标记集合 Σ。U_A 的转换关系是 $\Rightarrow$：对状态 q、q'和一个标记 a，$q \overset{a}{\Rightarrow} q'$ 当且仅当存在一个状态 q"和一个时间值 $\delta \in R^+$，使得在转换系统 S_A 中有 $q \xrightarrow{\delta} q'' \xrightarrow{a} q'$。在时间自动机的可达性问题中，希望确定目前位置的可达性。由此可得：为了解决可达性问题，可以考虑时间抽象转换系统 U_A 而不是 S_A。

3.2 稳定商

尽管时间抽象转换系统 U_A 只有有穷多个标记，但它仍然有无穷多个状态。为了处理这个问题，考虑在状态空间 Q_A 上的等价关系。一个在状态空间 Q_A 上的等价关系～称为是稳定的当且仅当只要有 q～u 和 $q \overset{a}{\Rightarrow} q'$，就存在一个状态 u'使得 $u \overset{a}{\Rightarrow} u'$ 并且 q'～u'。也就是说，一个稳定的等价是时间抽象转换系统的一个互模拟。

U_A 关于一个稳定划分～的商是一个转换系统$[U_A]_{\sim}$[4]：$[U_A]$的状态是～的等价类；如果一个等价类 π 包含 U_A 的初始状态，则 π 是$[U_A]_{\sim}$的初始状态；标记的集合是 Σ；如果对某个 $q \in \pi$ 和 $q' \in \pi'$，在 U_A 中有 $q \overset{a}{\Rightarrow} q'$，那么$[U_A]_{\sim}$包含一个从等价类 π 到 π' 的 a 标记的转换。为了将可达性问题(A, L^F)简化为一个关于～上的可达性问题，需要确保：除了稳定性之外，～不能使目标状

态与非目标状态等同。对于一个目标位置集合 $L^F \subseteq L$，如果无论何时当(s,v)～(s',v)，要么 s 和 s'都属于 L^F，要么 s 和 s'都不属于 L^F，则称等价关系是～L^F-敏感的。从而，为了解决可达性问题(A,L^F)，就需要搜索一个稳定的、L^F-敏感的并且只有有穷多等价类的等价关系～。

3.3 区域自动机

区域自动机是通过在状态空间上定义一个等价类来将无穷多状态划分为有穷的状态，其时钟区域的划分是要求时间的整数部分一致，并且所有时钟间的小数部分的变化顺序也一致。

3.3.1 时钟区域

定义 3.1(时间自动机的状态)[5] 设 $A=<\Sigma,S,S_0,X,I,E>$是一个时间自动机，如果有 $s \in S$，v 是时钟变量集合 X 上的一个时钟解释，称序偶(s,v)是 A 的一个状态。

时间自动机 A 的时钟变量的取值范围是非负实数集 R^+，如果 A 的状态是定义 3.1 所描述的形式，那么状态数目是不可数的，因此无法建立一个状态数目是不可变的自动机，即无法用时间自动机 A 的状态构建一个系统的自动机模型。

在自动机的状态空间上定义一个等价关系，它使得如果两个状态在所有时钟值的整数部分一致，并且所有时钟的小数部分的顺序也一致。时钟值的整数部分通常用于确定是否满足相应的时钟约束，而时钟值的小数部分的顺序则用于确定哪一个时钟变量将首先改变它的时钟值的整数部分。例如，在时间自动机的某一个状态，如果时钟变量 x 和 y 的时钟值满足$(0<x<1 \wedge 0<y<1)$，那么伴随有时钟约束(y=1)的位置转移是否可以跟在伴随有时钟约束(x=1)的位置转移之后，取决于这两个时钟变量目前的时钟值是否满足(x<y)。时钟值的整数部分可以任意大，但是如果一个时钟 x 从未和一个大于 c 的常数比较(即与 x 比较的最大常量为 c)，那么它的实际值一旦超过 c，就对决定允许的转移没有任何影响了。这里，假设所有的时钟约束都与整数

常量比较(如果时钟约束与有理数常量比较，可以用所有常量分母的最小公倍数来乘每一个常量)。

在给出时钟解释集合上等价关系的形式化定义之前，先对要用到的符号作一个简单介绍。对$\forall t\in R^+$，fr(t)表示 t 的小数部分，$\lfloor t\rfloor$表示 t 的整数部分，即 $t=\lfloor t\rfloor+fr(t)$。对于时间自动机 $A=<\Sigma,S,S_0,X,I,E>$的所有时钟变量 $x\in X$，设 c_x是出现在 E 中形如 $x\le c$ 或 $c\le x$ 的时钟约束子公式中的最大整数常量 c。

设～是定义在 X 上所有时钟解释组成的集合上的等价关系，即 v～v'当且仅当以下条件成立。

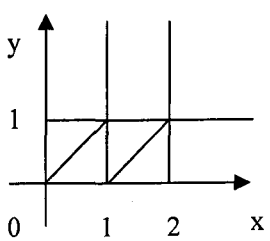
(1)对所有的 $x\in X$，或者$\lfloor v(x)\rfloor$与$\lfloor v'(x)\rfloor$相同，或者二者都大于 c_x；

(2)对所有的 $x,y\in X$，而且 $v(x)\le c_x$，$v(y)\le c_y$，$fr(v(x))\le fr(v(y))$当且仅当 $fr(v'(x))\le fr(v'(y))$成立；

(3)对所有 $x\in X$，而且 $v(x)<c_x$，那么 $fr(v(x))=0$ 当且仅当 $fr(v'(x))=0$。

定义 3.2(时钟区域)[5] 时间自动机 A 的时钟区域是由等价关系～所确立的时钟解释集合上的等价类，使用[v]来表示时钟解释 v 所属的等价类。

可以通过一个例子很好的理解等价类的实质。考虑一个带有两个时钟 x 和 y 且 $c_x=2$，$c_y=1$ 的时间自动机。在图 3.1 中显示了时钟区域。注意到区域的个数只是一个有穷数，最大为 $k!\cdot 2^k\cdot\prod_{x\in X}(2c_x+2)$，这里 k 是时钟的个数。这样，时钟区域的个数在时钟约束的编码上是指数级的。



6 个拐角点：例如[(0,1)]

14 个开集线段：例如[0<x=y<1]

8 个开区域：例如[0<x<y<1]

图 3.1 时钟区域

3.3.2 构造区域自动机

按定义 3.2 定义的时钟区域可以构造出只有有穷状态的区域自动机，下面先给出区域自动机时钟区域后继的定义。

定义 3.3(时钟区域的后继、前趋)[5] 时钟区域 α'是时钟区域 α 的时钟后继

当且仅当对每一个时钟解释 $v\in\alpha$ 存在实数 $t\in R^+$使得 $v+t\in\alpha'$。

要判定时间自动机 A 识别的时间语言是否为空，首先需要构造一个能够模拟 A 运行的转换系统。根据前面对时钟区域的定义，可以构造时间自动机 A 对应的区域自动机 R(A)。区域自动机 R(A)的每一个状态为(s,α)，(s,α)记录了时间自动机 A 的一个位置 s 和当前时钟解释所在的等价类 α，其中 s(s∈S)是时间自动机 A 的一个位置，α 是一个时钟区域，即如果 A 当前的状态是(s, v)，那么 R(A)的当前状态就是(s,[v]) ([v]表示时钟解释 v 所在的等价类)。区域自动机的初始状态是(s_0, [v_0])，其中 s_0 是 A 的起始位置，时钟解释 v_0 表示对时钟变量集合 X 中每一个时钟变量赋值为 0。在时间自动机 A 中，如果位置 s 在时钟解释为 v(v∈α)和输入字符是 a 的情况下可以转移到带有时钟解释 v'(v'∈α')的位置 s'，那么 R(A)中就存在一个标记有字符 a 的从(s,α)到(s',α')的状态转移。

定义 3.4(区域自动机)[5] 一个时间自动机 A=<Σ,S,S_0,X,I,E>所对应的区域自动机 R(A)是定义在字符集 Σ 上的一个如下所示的转换系统。

(1)R(A)的每一个状态都形如(s, α)，其中 s∈S，α 是一个时钟区域；

(2)R(A)的起始状态形如(s_0, [v_0])，其中 $s_0\in S$，并且$\forall x\in X$，$v_0(x)=0$；

(3)R(A)中有一个状态转移<(s,α),a,(s',α')>当且仅当时间自动机 A 中存在一个转移<s,a, $\oint$,λ,s'>∈E 并且存在一个时钟区域 α"满足以下三个条件：α"是 α 的时间后继；α"满足时钟约束 $\oint$ ；α'=[λ←0]α"。

例 3.1 图 3.2 为一个时间自动机 A，图 3.3 是与之对应的区域自动机 R(A)。

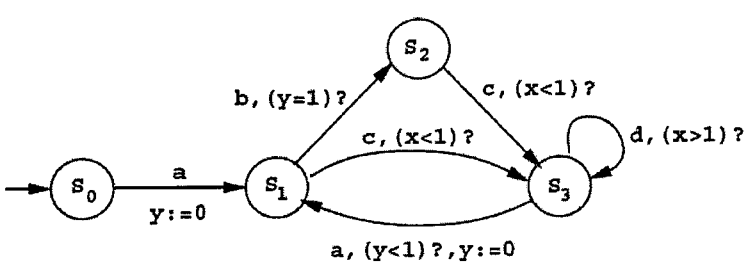


图 3.2 时间自动机 A

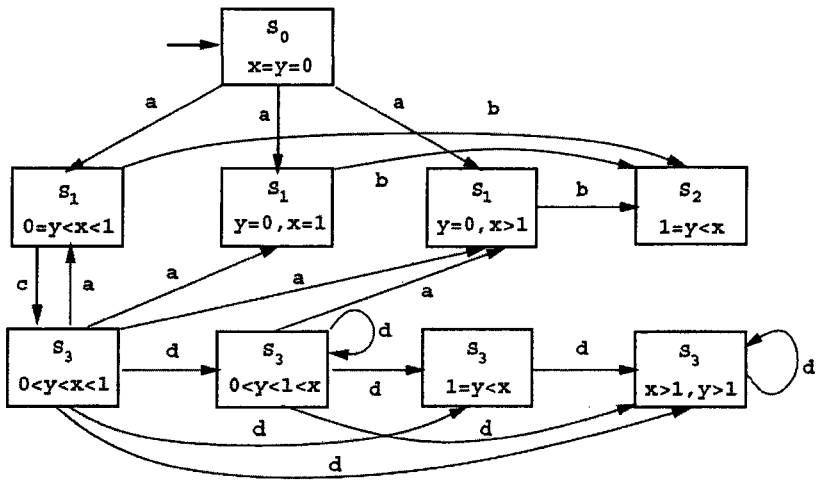


图 3.3 区域自动机 R(A)

3.3.3 利用区域自动机进行系统规范验证

对一个有穷状态的时间过程 P(A,L)，其中 A 是时间过程中所有事件组成的集合，L 是定义在集合 P(A)上的时间语言，可以用时间 Büchi 自动机 TBA A_P 模拟时间过程 P，即用时间 Büchi 自动机 TBA A_P 模拟时间过程 P 的过程转换图，用自动机 A_P 的时钟变量表示时间过程 P 的时间延迟，用自动机 A_P 的接受条件表示时间过程 P 的公平性条件[24]。因此，时间 Büchi 自动机 TBA A_P 所识别的时间语言 $L(A_P)$可以用来表示时间过程 P 的所有时间运行。一个时间过程 P 往往包含多个执行过程，因此相应的 TBA A_P 的构建是困难和繁琐的。通常情况下，将时间过程 P 的一个执行过程描述为该时间过程的一个子过程 $P_i=(A_i,L(A_i))$。

其中 A_i 表示子过程 P_i 所涉及到的事件组成的集合。这样就可以为每一个子过程构建其相应的 TBA A_{Pi}，然后再构造它们的积时间自动机 TBA A_I，表示所有子过程的积$[||_i A_{Pi}]$。时间过程 P 的规范说明用一个确定的 TBA A_S 表示，其中 A_S 识别的语言集合是定义在字符集 $P(A)(A=\cup_i A_i)$上的时间正则语言 S。时间过程 P 是正确的当且仅当 $L(A_I)\subseteq S$ 或者 $L(A_I)\cap S\neq\Phi$，即时间过程 P 能够满足自身的规范说明 S。

接下来根据区域自动机的构建方法，构建 TBA A_I 和 TBA A_S 的积时间自动机 $A_P=[A_I||A_S]$对应的区域自动机 $R(A_P)$。为了检验 TBA A_I 和 TBA A_S 所识

别语言的包含关系，可以在区域自动机 $R(A_P)$中查找是否存在一个满足如下条件的状态环：

(1)它包含 $R(A_P)$的初始状态$(s_0,[v_0])$；

(2)它的时钟变量满足递增条件：对所有的时钟变量 $x(x\in X)$，该状态环中至少存在一个时钟区域满足$[(x=0)\vee(x>c_x)]$；

(3)由于子过程 P_i 的定义要求只考虑那些所有自动机模型都无穷多次参与执行的无限运行，因此，对每一个 $1\le i\le n$，该状态环中应该包含一个过程 P 的所有子过程对应的自动机 TBA A_{Pi} 参与的状态转移；

(4)所有子过程对应的 TBA 必须满足公平性原则：即对每一个 $1\le i\le n$，状态环中必须包含一个状态，它的第 i 个状态分量属于 TBA A_{Pi} 的接受集合 F_i；

(5)规范说明对应的 TBA A_S 的公平性无法被满足；该状态环中不能包含这样的状态，它与 TBA A_S 对应的状态分量属于 TBA A_S 的接受集合 F_S。

如果区域自动机 $R(A_P)$中存在满足上述条件的状态环，就说明 $L(A_I)\not\subset S\wedge L(A_I)\neq S$，即 $L(A_I)\cap S=\Phi$。

3.4 带自动机

虽然区域自动机在实时系统的规范验证过程中起到了举足轻重的作用，但是，如 3.3.1 所述，区域自动机的状态数目会随着时钟集合中时钟变量数目的增加而呈指数级递增，并且状态个数的增加与时钟约束中的常量大小成正比。此外，构造出来的区域自动机模型可能会存在许多不可达或虽可达但无用的状态，因此研究人员就考虑利用其它方法来表示自动机的状态，其中带自动机可以较好的解决这些问题[4]。

3.4.1 带自动机的定义

带自动机是通过考虑凸时钟区域的合并来减少时钟区域。

定义 3.5(时钟带)[4,24] 一个时钟带 φ 是一个由时钟约束的并描述的时钟解释的集合，每个约束在一个时钟或两个不同的时钟上设置上界或下界。

如果时间自动机 A 有 k 个时钟，那么集合 φ 是一个在 k-维欧几里得空间

中的凸集。时钟带有如下性质：

(1)每个时钟区域是一个时钟带；

(2)对一个时钟区域 π 和一个时钟带 φ，π 或者完全包含在 φ 中，或者和 φ 不相交；

(3)两个时钟带的交是时钟带；

(4)位置上的不变式和转换的约束条件中所用的每一个时钟约束是一个时钟带；

(5)对两个时钟带 φ_1 和 φ_2，如果并集 $\varphi_1 \cup \varphi_2$ 是凸的，那么它是一个时钟带。由此可知时钟带的集合和时钟区域的凸并集的集合一致。

利用时钟带的可达性分析使用下面的三个操作：

(1)对于两个时钟带 φ 和 ψ，$\varphi \wedge \psi$ 表示两个带的交集；

(2)对于一个时钟带 φ，$\varphi\Uparrow$ 表示时钟解释集 $v+\delta$，其中 $v \in \varphi$，$\delta \in R^+$；

(3)对于一个时钟集合 λ 和一个时钟带 φ，φ[λ:=0]表示一个时钟解释集合 v[λ:=0]，其中 $v \in \varphi$。

时钟带的一个关键属性是在以上三个操作下的封闭性。

定义 3.6(带)[24] 一个带是一个位置 s 和一个时钟带 φ 的偶对(s, φ)。

定义 3.7(带的后继)[24] 对于一个带(s,φ)和一个时间自动机 A 的转换 e=<s,a,ψ,λ,s'>，令 succ(φ,e)为一个时钟解释 v'的集合，使得对于某些 $v \in \varphi$，状态(s',v')可以通过时间流逝和执行转换由状态(s,v)到达。即，集合(s',succ(φ,e))描述了在转换 e 下带(s, φ)的后继。集合 succ(φ,e)可以用时钟带上的三个操作按如下计算：

$$succ(\varphi,e)=(((\varphi \wedge I(s))\Uparrow) \wedge I(s) \wedge \psi)[\lambda:=0]$$

这样，时钟带在转换操作求后继时就被有效的封闭。一个带自动机有带(s, φ)与(s',succ(φ,e))之间的边。

对于一个时间自动机 A，带自动机 Z(A)是一个转换系统：Z(A)的状态是 A 的带，对 A 的每一个初始位置 s，带(s,[X:=0])是 Z(A)的一个初始位置，并且对于 A 每一个转换 e=<s,a,ψ,λ,s'>和每一个时钟带 φ，存在一个转换 <(s,φ),a,(s',succ(φ,e))>。

定义 3.8(带自动机)[24] 对时间自动机 A，相应的带自动机 Z(A)是一个转换系统：

(1)Z (A)的状态是带，即偶对(s, φ)；

(2)对 A 的初始状态 $s_0 \in S_0$，带(s,[v_0])是 Z (A)的初始状态，其中 $v_0(x)=0$，对 $\forall x \in X$；

(3)Z (A)中的状态转移为<(s, φ),a,(s', φ')>，其中 φ'包含所有满足 A 中状态转移<(s,v),a,(s',v')>的时钟解释 v'，$v \in \varphi$。

3.4.2 关于 k-近似

k-近似也是为加快状态空间搜索而采取的一种方法。若在可达集的计算中，确保计算所得可达集是实际可达集的扩大，则此类验证方法为过近似,也叫保守近似。对于安全性验证问题，过近似方法从初始状态集合出发，前向计算可达集，最后检查可达集是否在给定安全状态集合内。过近似方法沿着系统动态的演化计算状态值，一般都将连续变量状态空间划分为有穷数量的子空间，然后由过近似方法计算并判断出所有可达的子空间，最后得出可能发生的离散迁移，并检查系统安全规范是否满足。

令 k 为一个常量，一个 k-有界带是一个由 k-有界时钟约束定义的带(k-有界时钟约束是一个只包括在-k 和 k 之间的常量的时钟约束)。令 Z 为一个带，包含 Z 的 k-有界带的集合是有限的并且是非空的，这些 k-有界带的交集是一个包含 Z 的 k-有界带，并且是最小的一个(k-有界带)，它仍然包含 Z。这个最小的 k-有界带被称为 Z 的 k-近似 (k-approximation)[25,26]，记为 $Approx_k(Z)$。

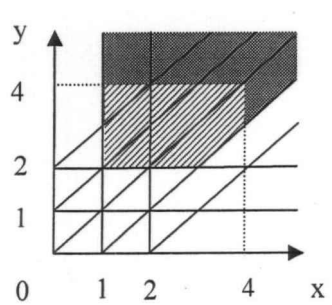


图 3.4 时钟带及其 $Approx_k(Z)$

例 3.2 考虑在上图中用▨画的带 Z：Z 由下面的时钟约束定义：

$1<x<4 \wedge 2<y<4 \wedge x-y<1$

取 k=2，Z 的 k-近似可以通过加上[redacted]获得。它由如下的时钟约束定义：

$1<x\wedge 2<y\wedge x-y<1$

引入 k-近似的概念是为了提高带自动机搜索状态的效率。在带自动机 Z(A)上可由 k-有界带构成新的时间自动机，令其为 $Z^*(A)$，这时，Z(A)的接受条件可以转换为 $Z^*(A)$的接受条件，即可以通过判断 $Z^*(A)$识别的语言为空，得出 Z(A)所识别的语言也为空的结论，从而可以判断出 A 识别的语言为空。但若 $Z^*(A)$所识别的语言非空，却得不出 Z(A)识别的语言也非空的结论。这时或者缩小 k-近似的范围，或者直接判断 Z(A)所识别的语言是否为空。

3.4.3 前向分析算法

前向分析算法是由初始状态逐步地求其后继。经典时间自动机前向分析算法中的一种是使用 k-近似作为扩展操作符。它的描述算法如下(q_0 是自动机的初始位置，而 Z_0 表示初始带，它通常包含所有时钟都被置为 0 的时钟解释)。

```
TA-Algorithm(A:TA){
  Define k as the maximal constant appearing in A;
  Visited:=Φ;          //Visited 用于存储已访问过的状态
  Waiting:={(q0,Approxk(Z0))};
  Repeat
    Get and Remove(q,Z) from Waiting;
    If q is final
      Then{Return "Yes";}
      Else{
        If there is no(q,Z')∈Visited such that Z⊆Z'
        Then{
          Visited:=Visited∪{(q,Z)};
          Successor:={(q',Approxk(Post(Z,e)))|e transition from q to q'};
          Waiting:=Waiting∪Successor;
        }
      }
```

```
    Until(Waiting=Φ);
    Return "No";
}
```

首先指出算法可以终止，因为有有限多个 k-有界带，从而有有限多个带的 k-近似，它们可由每个自动机的控制状态被计算出来。这个算法逐步计算可达状态集合的一个近似，并且测试这个近似与终止状态集合是否相交(是否有交集)。这样，如果算法返回“No”，那么没有可以达到的终止状态。但是，如果算法返回“Yes”，那么可先验地推理得出这样一种情况：此过近似与终止状态的集合相交，但是精确的可达状态的集合却与这个集合(终止状态的集合)不相交。

3.5 区域自动机与带自动机必须处理的问题

区域自动机构造出的状态数目会随着时钟变量数目的增加而呈指数级递增，并且状态个数的增加与时钟约束中的常量大小成正比。虽然可以采用一些优化算法去掉无用的状态[37]，但它的验证速度还是较难提高；带自动机也存在状态空间爆炸问题，但在一定程度上减少了状态空间的生成，在实际应用中较多。

从分析中可以发现，它们的共同点在于都将时间自动机中的无穷多状态构造为有穷个状态，构造的基本思想都是对时间进行合并归类。因此，时间的有效表示是它们都必须处理的问题，也是实现时间自动机的关键问题之一，在第 4 章本文将对时钟带的有效表示做深入研究。

3.6 本章小结

本章介绍了时间抽象转换系、区域自动机、带自动机的概念，分析了区域自动机的构造方法及其生成状态空间的方式，由于区域自动机状态个数对于时钟个数的增长为指数级，引出时钟带的概念，并引入了使用带作为状态的状态空间搜索的前向分析算法，最后指出构造出有穷状态的时间自动机都必须处理的问题。

第4章 时钟带表示方法

4.1 时钟带的DBM表示

实现带自动机的关键问题之一是如何有效的表示时钟带，目前已经提出多种时钟带的表示方法，其中 DBM 是较早提出的一种有效表示方法[3,13]。

4.1.1 DBM 数据结构

为了实现前向分析算法，需要一个适合的数据结构来描述带，并且这个数据结构要求能方便测试带的包含关系，也应该容易在算法中进行不同操作的计算，即：两个带的交，带的未来(也即后继)，重置操作下带的映射和带的 k-近似计算。UPPAAL 和 KRONOS 这些工具使用了 Dill 提出的数据结构-DBM(Difference Bounded Matrix)。

对于 n 个时钟集合 $X=\{x_1,\ldots x_n\}$的差分有界矩阵(简称 DBM)是一个元素为偶对的(n+1)阶方阵，偶对形式如下：

$$(m;\prec)\in V=(Z\times\{<,\leq\})\cup\{(\infty;<)\}$$

一个DBM $M=(m_{ij};\prec_{ij})_{i,j=0\ldots n}$定义如下的$T^n$($T^n$是一个k-有界的带并且包含Z)的子集(时钟 x_0 被假设总等于 0，即对每个时钟解释 $v,v(x_0)=0$)：

$$\{v:X\to T|\forall\ 0\leq i,j\leq n,v(x_i)-v(x_j)\prec_{ij}m_{ij}\}$$

其中$\gamma\prec\infty$只是表示 γ 是一些没有边界的实数。

这个 T^n 的子集是一个带并且表示为[M]。n 个时钟上的每个 DBM 表示一个 T^n 的带。这里应注意到几个 DBM 可以定义同一个带。

例 4.1 时钟约束 $0\leq x_1<2\wedge 0<x_2<1\wedge x_1-x_2\geq 0$ 可以表示为如下的 DBM 矩阵 D，也可表示为 D'。

$$\begin{pmatrix} (0;\le) & (0;\le) & (0;<) \\ (2;<) & (0;\le) & (\infty;<) \\ (1;<) & (0;\le) & (0;\le) \end{pmatrix} \quad 和 \quad \begin{pmatrix} (0;\le) & (0;\le) & (0;<) \\ (2;<) & (0;\le) & (2;<) \\ (1;<) & (0;\le) & (0;\le) \end{pmatrix}$$

D　　D'

图 4.1 时钟带的两个 DBM 表示

例 4.2 时钟约束 $x_1>1 \wedge x_1\le 4 \wedge x_2\ge 1 \wedge x_2\le 3$ 可以由图 4.2 的两个 DBM 表示。

$$\begin{pmatrix} (0;\le) & (-1;<) & (-1;\le) \\ (4;\le) & (0;\le) & (\infty;<) \\ (3;\le) & (\infty;<) & (0;\le) \end{pmatrix} \quad 和 \quad \begin{pmatrix} (0;\le) & (-1;<) & (-1;\le) \\ (4;\le) & (0;\le) & (3;\le) \\ (3;\le) & (2;<) & (0;\le) \end{pmatrix}$$

D　　D'

图 4.2 时钟带的两个 DBM 表示

因此，一般的 DBM 不是带的一个规范的表示[36]。而且，不能够从句法上检验是否有$[M_1]=[M_2]$，这个问题给带的其它操作带来了麻烦。观察在例 4.1 和例 4.2 中的第一个矩阵 D 中有许多隐含的约束没有被反映出来。而通过“加紧”所有约束可以从矩阵 D 获得矩阵 D'。这样的一个“加紧”是通过观察到在时钟差 x_i-x_j 和 x_j-x_l 上的上界的和就是差 x_i-x_l 上的上界的和(出于这个目的，将“+”操作扩展到矩阵中的元素上，即偶对表示的边界上)。因此需要给带的表示定义一个标准的形式，称为范式。

如果 DBM D 表示一个非空的时钟带，就称 D 是可满足的。每一个可满足的 DBM 有一个等价的标准的 DBM 形式。用标准的 DBM 表示时钟带，容易证明，对任意两个同阶 DBM 矩阵，若表示的时钟带相同，则一定会有相同的范式，也就是说，表示任何区域的标准的 DBM 形式是唯一的。

DBM 范式的计算采用弗洛伊德算法和一些句法上的改写[3,10]。在下面，本文用 φ(M)表示 M 的范式。

首先，要求在 V 上有一个全序关系：“<”严格小于“≤”。V 中元素间的关系如下：

$(m;\prec),(m';\prec')\in V$，那么：

$$(m;\prec)\leq(m';\prec')<=>\begin{cases} m<m' \\ 或 \\ m=m' \text{ 且要么“}\prec\text{”=“}\prec'\text{”要么“}\prec'\text{”=“}\leq\text{”} \end{cases}$$

当然，对每个 $m\in Z$，有 $m<\infty$。用自然的方式定义>,≥和<,≤。这些次序用如下方式被扩展到 DBM $M=(m_{i,j};\prec_{i,j})_{i,j=0\ldots n}$ 和 $M'=(m'_{i,j};\prec'_{i,j})_{i,j=0\ldots n}$：

$M\leq M'<=>$对每个 $i,j=0\ldots n$,都有$(m_{i,j};\prec_{i,j})\leq(m'_{i,j};\prec'_{i,j})$

现在可以叙述范式的一些非常有用的性质。如果 M 和 M'是 DBM，那么：

(1)$[M]=[\varphi(M)]$并且 $\varphi(M)\leq M$；

(2)$[M]\subseteq[M']<=>\varphi(M)\leq M'<=>\varphi(M)\leq\varphi(M')$。

最后一点表达了这样的一事实：带的包含的测试可以从语法上用DBM(表示带)的范式进行检测。

DBM 的范式可以用一种自然的方式刻画。如果 $M=(m_{i,j};\prec_{i,j})_{i,j=0\ldots n}$ 是一个 DBM，并且$[M]\neq\varphi$，那么下面的两种性质是等价的：

(1)M 是一个 DBM 范式；

(2)对每个 $i,j=0\ldots n$，对每个实数$-m_{j,i}\prec_{j,i}r\prec_{i,j}m_{i,j}$，都存在一个时钟解释 $v\in[M]$，使得 $v(x_j)-v(x_i)=r$(仍然假设 $v(x_0)=0$)。

这个性质表达了这样一个事实：如果一个 DBM 是范式，那么使用弗洛伊德算法对这个 DBM 的约束不能再加紧。

为了计算 DBM 的范式，在集合 V 上定义一个加法：

$(m;\prec)+(m';\prec')=(m'';\prec'')$其中：

(1)$m''=m+m'$；

(2)如果$\prec$和$\prec'$都是≤，那么$\prec''$就是≤，否则$\prec''$就是<。

现在，如果 $M=((m_{i,j};\prec_{i,j})_{i,j})$是一个 DBM$[M]=\Phi$ 当且仅当在 M 中存在一个负循环，即指存在一个明确的下标序列$(i_1,i_2,\ldots i_{k-1},i_k=i_1)$使得：

$(m_{i1,i2};\prec_{i1,i2})+(m_{i2,i3};\prec_{i2,i3})+\ldots+(m_{ik-1,ik};\prec_{ik-1,ik})<(0;\leq)$

就像在本节开始时所讨论的那样，这个用于表示带的数据结构也必须适合于计算在算法 3.1 中使用的四个操作，即：未来、交、复位映射和 k-近似计算，下面是简单的描述：

交：令 $M=(m_{i,j};\prec_{i,j})_{i,j=1\ldots n}$ 并且 $M'=(m'_{i,j};\prec'_{i,j})_{i,j=1\ldots n}$ 是两个 DBM，按如下

的方式定义 $M''=(m''_{i,j};\prec''_{i,j})_{i,j=1\ldots n}$：

$(m_{i,j}'';\prec_{i,j}'')=\min((m_{i,j};\prec_{i,j}),(m'_{i,j};\prec'_{i,j}))\ i,j=1\ldots n$

则$[M'']=[M]\cap[M']$。注意到可能出现这种情况：即使 M 和 M'是范式，M'' 却不是范式。

未来：假设 $M=(m_{i,j};\prec_{i,j})_{i,j=1\ldots n}$ 是一个范式的 DBM。按如下方式定义 DBM $\vec{M}=(m'_{i,j};\prec'_{i,j})_{i,j=1\ldots n}$：

$$\begin{cases}(m'_{i,j};\prec'_{i,j})=(m_{i,j};\prec_{i,j}) & \text{如果 } j\neq 0\\ (m_i',_0;\prec'_{i,0})=(\infty;<)\end{cases}$$

则$[\vec{M}]=[M]\Uparrow$(M 的未来)并且 DBM $\vec{M}$ 是范式。

复位映射：假设 $M=(m_{i,j};\prec_{i,j})_{i,j=1\ldots n}$ 是一个范式的 DBM。按如下方式定义 DBM $M_{xk:=0}=(m'_{i,j};\prec'_{i,j})_{i,j=1\ldots n}$：

$$\begin{cases}(m'_{i,j};\prec'_{i,j})=(m_{i,j};\prec_{i,j}) & \text{如果 } i,j\neq k\\ (m'_{k,k};\prec'_{k,k})=(m'_{k,0};\prec'_{k,0})=(m'_{0,k};\prec'_{0,k})=(0;\leq) & \\ (m'_{i,k};\prec'_{i,k})=(m_{i,0};\prec_{i,0}) & \text{如果 } i\neq k\\ (m'_{k,i};\prec'_{k,i})=(m_{0,i};\prec_{0,i}) & \text{如果 } i\neq k\end{cases}$$

那么$[M_{xk:=0}]=[x_k\leftarrow 0][M]$，并且 DBM $M_{xk:=0}$ 是范式。

k-近似计算：假设 $M=(m_{i,j};\prec_{i,j})_{i,j=1\ldots n}$ 是一个标准形式的 DBM。按如下方式定义 DBM $M_k=(m'_{i,j};\prec'_{i,j})_{i,j=1\ldots n}$：

$$\begin{cases}(m'_{i,j};\prec'_{i,j})=(m_{i,j};\prec_{i,j}) & \text{如果 } -k\leq m_{i,j}\leq k\\ (m'_{i,j};\prec'_{i,j})=(\infty;<) & \text{如果 } m_{i,j}>k\\ (m'_{i,j};\prec'_{i,j})=(-k;<) & \text{如果 } m_{i,j}<-k\end{cases}$$

那么得到$[M_k]=\text{Approx}_k([M])$，但 M_k 不是范式。

结合所有这些构造，如果 e 是一个 $q\xrightarrow{g,a,C:=0}q'$形式的转换，并且 M 是一个表示带 Z 的 DBM，使用上面的性质，很容易的计算一个表示带 $\text{Approx}_k([C\leftarrow 0](g\cap\vec{Z}))=\text{Approx}_k(\text{Post}(Z,e))$。这个 DBM 可能不是范式。这样 DBM 数据结构就可以用来计算在前向分析算法中所需的所有操作。

4.1.2 DBM 的范式化算法

通过上面的分析可知对于一个 DBM，在其上的许多操作都要求首先对其进行范式化，即求出它的“最紧形式”，目前所采用的算法是 Floyd(弗洛伊德)算法，下面首先简单的介绍一下 Floyd 算法。

Floyd 算法所求解的问题是带权有向图中每对顶点之间的最短路径。Floyd 算法从图的邻接矩阵开始，按照顶点 $v_0,v_1,\ldots,v_{n-1}$ 的次序，分别以每个顶点 $v_k(0\le k\le n-1)$作为新考虑的中间点，在第 k-1 次运算得到的 $A^{(k-1)}$($A^{(-1)}$为图的邻接矩阵)的基础上，求出每对顶点 v_i 到 v_j 的目前最短路径长度 $A^{(k)}[i][j]$，计算公式为：

$$A^{(k)}[i][j]=\min(A^{(k-1)}[i][j],A^{(k-1)}[i][k]+A^{(k-1)}[k][j])(0\le k\le n-1,0\le i\le n-1,0\le j\le n-1)(4\text{-}1)$$

其中 min 函数表示取其参数表中的较小值，参数表中的前项表示在第 k-1 次运算后得到的从 v_i 到 v_j 的目前最短路径长度，后项表示考虑以 v_k 作为新的中间点所得到的从 v_i 到 v_j 的路径长度。若后项小于前项，则表明以 v_k 作为中间点(不排除已经以 $v_0,v_1,\ldots,v_{k-1}$ 中的一部分或全部作为其中间点)使得从 v_i 到 v_j 的路径长度变短，所以应把它的值赋给 $A^{(k)}[i][j]$，否则把 $A^{(k-1)}[i][j]$的值赋给 $A^{(k)}[i][j]$。总之，使 $A^{(k)}[i][j]$保存第 k 次运算后得到的从 v_i 到 v_j 的目前最短路径长度。当 k 从 0 取到 n-1 后，矩阵 $A^{(n-1)}$就是最后得到的结果，其中每个元素 $A^{(n-1)}[i][j]$就是从顶点 v_i 到 v_j 的最短路径长度。

上述算法描述如下：

```
void Floyd(int A[][MAXSIZE], int n)
{
int i,j,k;
for(k=0; k<n; k++)
    for(i=0; i<n; i++)
        for(j=0; j<n; j++)
        {
            if(A[i][k]+A[k][j]<A[i][j])
                A[i][j]=A[i][k]+A[k][j];
        }
```

}

容易看出，Floyd 算法的时间复杂度为 $O(n^3)$，下面通过一个例子说明用 Floyd 算法求解 DBM 范式的过程。以例 4.2 中的时钟带为例，时钟带的等式表示为：$x_1>1 \wedge x_1\leq 4 \wedge x_2\geq 1 \wedge x_2\leq 3$，其 DBM 表示如图 4.2 所示，下面来求解它的标准 DBM 形式。

$$\begin{bmatrix} (0;\leq) & (-1;<) & (-1;\leq) \\ (4;\leq) & (0;\leq) & (\infty;<) \\ (3;\leq) & (\infty;<) & (0;\leq) \end{bmatrix}$$

D

图 4.3 例 4. 2 的初始时钟带表示

此时钟带有两个时钟变量 x_1 和 x_2，x_0 的引入是为了便于表示和计算。通过分析发现，对于 x_1 与 x_2 之间的关系并没有准确的表示出来。实质上，通过对等式 $x_1\leq 4 \wedge x_2\geq 1$ 可以得出 $x_1-x_2\leq 3$ 的结果，而对于其初始的 DBM 表示却为 $x_1-x_2\leq\infty$，虽然在逻辑上是对的，但它并没有表达出 x_1 与 x_2 之间的准确关系。同理对于 $x_2-x_1<2$ 的关系也没有准确的表达。通过 Floyd 算法，进行一次迭代即可以求出其 DBM 的范式化形式，如图 4.4。

$$\begin{bmatrix} (0;\leq) & (-1;<) & (-1;\leq) \\ (4;\leq) & (0;\leq) & (\infty;<) \\ (3;\leq) & (\infty;<) & (0;\leq) \end{bmatrix} + \begin{bmatrix} (0;\leq) & (-1;<) & (-1;\leq) \\ (4;\leq) & (0;\leq) & (\infty;<) \\ (3;\leq) & (\infty;<) & (0;\leq) \end{bmatrix} = \begin{bmatrix} (0;\leq) & (-1;<) & (-1;\leq) \\ (4;\leq) & (0;\leq) & (3;\leq) \\ (3;\leq) & (2;<) & (0;\leq) \end{bmatrix}$$

D　　D　　D'

图 4.4 范式化 DBM 的求解过程

也即由 A[1][0]+A[0][2]=A[1][2]，A[2][0]+A[0][1]=A[2][1]。通过 4.1.1 节的分析已经知道这种 DBM 的范式化形式是唯一的。

4.1.3 对 DBM 范式化算法改进的尝试

以上为经典 Floyd 算法，本文曾尝试一些方法来减少范式化的执行时间。首先是通过如下分析对 Floyd 算法进行优化。

对于公式(4-1)，有几种情况不需要进行计算。

(1)当 i=j 时公式变为：

$$A^{(k)}[i][i]=\min(A^{(k-1)}[i][i], A^{(k-1)}[i][k]+A^{(k-1)}[k][i]) \quad (0\leq i\leq n-1)$$

若 k=0，则参数表中的前项 $A^{(-1)}[i][i]=A[i][i]=0$，后项 $A^{(-1)}[i][0]+A^{(-1)}[0][i]$ 必定大于等于 0，所以 $A^{(0)}$中的对角线元素同 $A^{(-1)}$中的对角线元素一样，均为 0。同理，当 k=1,2,…,n-1 时，$A^{(k)}$中的对角线元素也均为 0。

(2)当 i=k 或 j=k 时分别变为：

$$A^{(k)}[k][j]=\min(A^{(k-1)}[k][j], A^{(k-1)}[k][k]+A^{(k-1)}[k][j]) \quad (0\le j\le n-1)$$

$$A^{(k)}[i][k]=\min(A^{(k-1)}[i][k], A^{(k-1)}[i][k]+A^{(k-1)}[k][k]) \quad (0\le i\le n-1)$$

每个参数表中的后一项都由它的前一项加上 $A^{(k-1)}[k][k]$所组成，因为 $A^{(k-1)}[k][k]=0$，所以 $A^{(k)}[k][j]$和 $A^{(k)}[i][k]$分别取上一次的运算结果 $A^{(k-1)}[k][j]$和 $A^{(k-1)}[i][k]$的值，也就是说，矩阵 $A^{(k)}$中的第 k 行和第 k 列上的元素均取上一次运算的结果。因此可以通过增加一条判断语句去掉对无用元素之间的相加及判断大小运算，即，若“i= =j||i= =k||j= =k”时跳出进行下一次迭代。

这种想法表面上看起来可以减少 DBM 范式化的时间，然而本文通过分析发现它只适合时钟变量较少的情况。当时钟变量较多时，满足“i= =j||i= =k||j= =k”的元素相对于不满足条件的数量减少很快，而在 Floyd 算法的最内层增加了这条判断语句，使它的性能反而下降。

本文也曾尝试通过在 Floyd 算法中每次增加两个点以减少执行次数的方法来减少范式化所消耗的时间，推导出如下的公式：

$$\begin{aligned}A^{(k)}[i][j]=\min(&\\ &A^{(k-2)}[i][j],\\ &A^{(k-2)}[i][k-1]+A^{(k-2)}[k-1][j]),\\ &A^{(k-2)}[i][k]+A^{(k-2)}[k][j],\\ &A^{(k-2)}[i][k]+A^{(k-2)}[k][k-1]+A^{(k-2)}[k-1][j],\\ &A^{(k-2)}[i][k-1]+A^{(k-2)}[k-1][k]+A^{(k-2)}[k][j]) \qquad (4\text{-}2)\end{aligned}$$

这个想法同前面的想法存在同样的问题，虽然算法的执行次数减少为原来的一半，但由于最内层循环增加的语句太多，导致算法效率得不到提高，通过分析及实验结果表明也没有好的效果。

4.1.4 一类时钟带新的范式化算法

从 4.1.3 小节的分析中可以看出，降低范式化过程的时间复杂度存在一定

困难。而且其表示方法的空间复杂度为 $O(n^2)$，也不易压缩。

通过细致分析，可以发现给定时钟带的特点。以例 4.1 和例 4.2 给出的等式为例加以分析。

1. 问题描述

例 4.1 的时钟带等式为：$0\le x_1<2 \wedge 0<x_2<1 \wedge x_1-x_2\ge 0$，它可以表示为图 4.1 中 D 和 D'两个 DBM。对这 D 和 D'分别进行 DBM 范式化处理可以得到图 4.5 中的唯一范式：

$$\begin{pmatrix} (0;\le) & (0;\le) & (0;<) \\ (2;<) & (0;\le) & (2;<) \\ (1;<) & (0;\le) & (0;\le) \end{pmatrix}$$

$\Phi(D)$

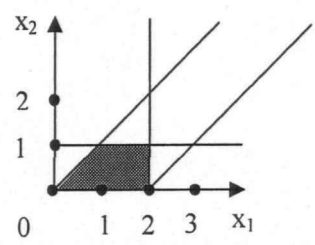


图 4.5 例 4.1 范式化结果及几何表示

例 4.2 的时钟带等式为 $x_1>1 \wedge x_1\le 4 \wedge x_2\ge 1 \wedge x_2\le 3$，它可以表示为图 4.2 中 D 和 D'两个 DBM。对这 D 和 D'分别进行 DBM 范式化处理可以得到图 4.6 中的唯一范式：

$$\begin{pmatrix} (0;\le) & (-1;<) & (-1;\le) \\ (4;\le) & (0;\le) & (3;\le) \\ (3;\le) & (2;<) & (0;\le) \end{pmatrix}$$

$\Phi(D)$

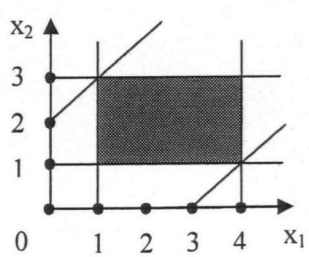


图 4.6 例 4.2 范式化结果及几何表示

通过观察发现，在例 4.2DBM 范式中的 $A[i][j]=A[i][0]+A[0][j](i,j\neq 0)$ (为方便叙述，本文暂不区别“≤”和“<”之间的关系)，但对于例 4.1 却得不到这个结论。

分析其原因在于例 4.1 和例 4.2 中时钟关系式的给出形式不同。例 4.1 的时钟带给出形式中即有 x_1、x_2 的形式，也有 x_1-x_2 的形式，而例 4.2 中只有 x_1、x_2 的形式。下面本文就按这种区别将时钟带分为两类：

(1)既含有 x_i、x_j 的形式，也有 x_i-x_j 的形式。在 DBM 的表示中体现为，

对于 DBM $M=(m_{ij};\prec_{ij})_{i,j=0\ldots n}$ 其中的元素 m_{ij}，当 i、j 都不为 0 时且不相等时，至少有一项 m_{ij} 不为∞，从几何角度看对应一个凸的非正多面体。本文称其为第一类型。

(2)只含有 x_i、x_j 的形式。在 DBM 的表示中体现为，对于 DBM $M=(m_{ij};\prec_{ij})_{i,j=0\ldots n}$ 其中的每个元素 m_{ij}，当 i、j 都不为 0 且不相等时，都有 $m_{ij}=\infty$，从几何角度看对应一个凸的正多面体。本文称其为第二类型。

以例 4.1 和例 4.2 为例，它们表示区域的几何表示形式为图 4.6 中的阴影部分，范式化之后是将这些区域限制在了与之对应的两个斜率为 1 的斜线之间。对于这两类时钟带类型，当然都可以通过 DBM 数据结构表示，并且用 Floyd 算法进行范式化。但对于第二类，可以找到一个更为高效的算法，并且在存储时也可以进行压缩，它是本文研究的主要内容。

2. 解决方法

在进行范化的过程中，是按公式 4-1 进行处理，即通过 n^3 的循环，判断任意两点之间的最短路径。下面再来看一下这个公式：

$A^{(k)}[i][j]=\min(A^{(k-1)}[i][j],A^{(k-1)}[i][k]+A^{(k-1)}[k][j])$ $(0\le k\le n-1,0\le i\le n-1,0\le j\le n-1)$

对于第二类型的时钟带，因为不存在 x_i-x_j 的形式，那么实际上并没有必要进行 n^3 循环。因为对于每个 A[i][j](i,j≠0)，它可以由 A[i][0]和 A[0][j]唯一确定，下面以定理的形式给出描述。

定理 4.1 对于第二类型的时钟带，对任意 i,j≠0，有下式成立：

$$A[i][j]=A[i][0]+A[0][j] \tag{4-3}$$

下面证明它的正确性。

证明：对于 Floyd 算法(它的原理在 4.1.2 中已经描述)，是通过每次加入一个新考虑的节点来求任意两点之间的最短距离。对于第二类型，初始时 A[i][j]为无穷大。它可能经过了 0,1…n-1 个节点后的路径最短，任取一个节点 k，有 A[i][j]=A[i][k]+A[k][j]，可以出现两种情况：

(1)k=0，则 A[i][j]=A[i][0]+A[0][j]

(2)k≠0，则 A[i][j]=A[i][k]+A[k][j]

而 A[i][k] 和 A[k][j]又需要按上述算法进行计算，又有：

若经过的中间节点是 0，则代入上式有

A[i][k]=A[i][0]+A[0][k]≥A[i][0]

A[k][j]=A[k][0]+A[0][j]≥A[0][j]

所以 A[i][j] =A[i][k]+A[k][j] ≥A[i][0]+A[0][j]

若经过的中间节点不为 0，则算法继续。

因为循环是从 0 开始，所以 k 一定可以取到 0，而由上面的不等式可知 A[i][j]=A[i][0]+A[0][j]必定为所有计算中最小的，也即为 i 到 j 之间的最短路径，证毕。

下面给出这种情况下求最短路径的算法，为方便叙述，本文称之为 Standard 算法：

```
void Standard(int A[][MAXSIZE], int n)
{
    int i,j;
        for (i = 1; i < n; i++)
            for (j = 1; j < n; j++)
            {
                if (A[i][0]!=INT_MAX&&A[0][j]!=INT_MAX&&i!=j)
                A[i][j]=A[i][0]+A[0][j];
            }
}
```

判断语句的作用是将不存在最短路径以及矩阵主对角线上的元素情况排除，因为它们不需要进行计算。

可以看出，在进行 DBM 范式化时，Standard 算法将 Floyd 算法时间复杂度由的 $O(n^3)$降低为 $O(n^2)$。

通过上面的分析还可以得出一个结论：A[i][j] (i,j≠0)可以由 A[i][0]和 A[0][j]唯一确定。这样对于 A[i][j] (i and j≠0)就没有必要存储其数据了，因为它们可以由 A[i][j] (i or j=0)产生，并且消耗的时间很少。那么可以只存储整个 DBM 的第 0 行和第 0 列，需要 2n+1 个存储单元，从而将 DBM 的空间复杂度从 $O(n^2)$降低为 O(n)。

3. 实验数据

下面通过实验数据对两种范式化方法进行比较(实际上在第二类型的时钟带表示中不可能出现负数)，对于给出的任意一个第二类型时钟带，共有

12 个时钟变量：

$1\le x_1\le 4$, $1\le x_2\le 3$, $x_3\le 6$, $8\le x_4$, $x_5=0$, $-2\le x_6\le 25$, $-12\le x_7\le 45$, $1\le x_8\le 35$, $3\le x_9\le 15$, $0\le x_{10}\le 5$, $-5\le x_{11}\le 10$, $1\le x_{12}\le 30$，它的 DBM 表示为图 4.7。

0,	-1,	-1,	∞,	-8,	0,	2,	12,	-1,	-3,	0,	5,	-1
4,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞
3,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞
6,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞
∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞
0,	∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞
25,	∞,	∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞
45,	∞,	∞,	∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞
35,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞
15,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	0,	∞,	∞,	∞
5,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	0,	∞,	∞
10,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	0,	∞
30,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	0

图 4.7 实验数据

对图 4.7 的矩阵分别用 Floyd 和 Standard 算法进行范式化，可以得到相同的结果，如图 4.8 所示。

0,	-1,	-1,	∞,	-8,	0,	2,	12,	-1,	-3,	0,	5,	-1
4,	0,	3,	∞,	-4,	4,	6,	16,	3,	1,	4,	9,	3
3,	2,	0,	∞,	-5,	3,	5,	15,	2,	0,	3,	8,	2
6,	5,	5,	0,	-2,	6,	8,	18,	5,	3,	6,	11,	5
∞,	∞,	∞,	∞,	0,	∞,	∞,	∞,	∞,	∞,	∞,	∞,	∞
0,	-1,	-1,	∞,	-8,	0,	2,	12,	-1,	-3,	0,	5,	-1
25,	24,	24,	∞,	17,	25,	0,	37,	24,	22,	25,	30,	24
45,	44,	44,	∞,	37,	45,	47,	0,	44,	42,	45,	50,	44
35,	34,	34,	∞,	27,	35,	37,	47,	0,	32,	35,	40,	34
15,	14,	14,	∞,	7,	15,	17,	27,	14,	0,	15,	20,	14
5,	4,	4,	∞,	-3,	5,	7,	17,	4,	2,	0,	10,	4
10,	9,	9,	∞,	2,	10,	12,	22,	9,	7,	10,	0,	9
30,	29,	29,	∞,	22,	30,	32,	42,	29,	27,	30,	35,	0

图 4.8 范式化结果

下面对从 3 阶到 12 阶矩阵分别使用 Floyd 和 Standard 算法迭代 100000 次，得出它们所消耗的时间，单位为秒，以表格的形式给出，结果如表 4.1 所示。运行平台为：Celeron(R)CPU2.4GHz，256M，Windows2000。

表 4.1 对第二类型时钟带使用 Floyd 与 Standard 算法的比较

	3	4	5	6	7	8	9	10	11	12
Floyd	0.203	0.344	0.484	0.937	1.250	1.781	2.172	3.157	3.859	4.719
Standard	0.078	0.078	0.125	0.156	0.203	0.265	0.344	0.469	0.531	0.687

从实验数据可以看出，由于 Standard 算法的执行次数减少为 n^2，它的时间效率有了明显的提高。Standard 算法执行速度很快，所以在存储时可以只存储第 0 行和第 0 列，而在需要其它元素时，只需进行一次 Standard 算法就可以得到，不需耗费太多时间。

第二类型的时钟带除了在给定时钟约束时能够出现，对于一些集合的交操作也可能出现，只要他的最后表示形式为第二类型就可以用上述方法进行存储和相关操作。

4.2 时钟带的其它表示方法

对于时钟带，除了 DBM 表示方法外，还有其它的一些表示方法，这些方法有 CDD (clock difference diagram)[28]，NDD (numerical decision diagram)[29]，RED (region-encoding diagram)[30]等，都是尝试提高数据结构的时间和空间效率。下面对 CDD 做一简单介绍。

CDD 是一个有向图，它的结点分成两类：

(1)终端结点。该结点要么为 1，要么为 0，不存在后继结点。

(2)内部结点。带类型(为时钟变量差或单个时钟变量)的结点，其值为实数域的子区间(对类型为单个时钟变量的结点，其区间限于非负实数域的子区间)。

CDD 的提出是为了克服 DBM 对于带的求并运算的复杂性以及 DBM 对凹区域表示的复杂性。图 4.9 给出了一个凹时间区域及其 CDD 表示。

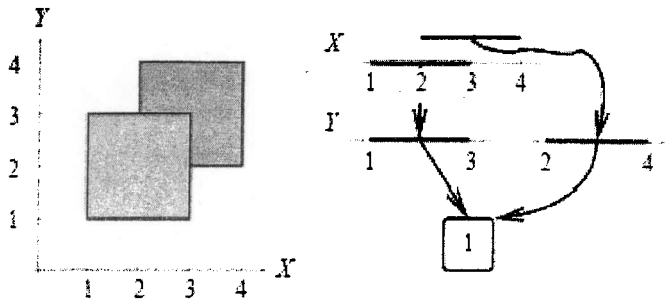


图 4.9 CDD 表示

CDD 从最顶层内部结点(无前趋的内部结点)到终端结点 1 的一条“路径”表示的即是一个凸区域。CDD 可视作多条“路径”组成的一个有向图,它所表示的区域即其所有路径表示的区域的并集。CDD 的求补操作十分简便，只需将所有直接后继为 1(或 0)的内结点调换成指向 0(或 1)即可。CDD 的“并”和“交”的原理可简述为：两 CDD 的“并”为两 CDD 里所有表示凸区域的路径的“并”，而两 CDD 的“交”则先将两 CDD 的各自所有表示凸区域的路径与对方所有表示凸区域的路径逐个求交，再取结果所有路径的“并”即可得到。但路径的“并”不是将路径简单并置于一起，而是要涉及到结点共享等处理，而路径的“交”则需要处理结点的区间包含关系。

CDD 在其“范式化”处理中对这每一条路径都要用最短路径算法计算其 DBM 范式，从而得到整个 CDD 的范式。若其时钟带为本文的提到的第二类型，仍可以按本文的方法进行范式化和存储。

4.3 本章小结

本章详细的介绍了带自动机中时钟带的 DBM 表示方法，分析时钟带的特点，并分离出一类时钟带，提出在这类时钟带上的范式化及存储方法。在理论上证明了这种方法的正确性，并通过实验验证了这种方法的有效性。本章最后对目前时钟带的其它表示方法做了简要介绍。

第5章 基于时间自动机的建模及验证

5.1 UPPAAL简介

UPPAAL 由 Aalborg 大学和 Uppsala 大学于 1995 年联合开发的一个基于时间自动机的工具。它适用于可以被描述为非确定的并行过程的积的系统[31,32]。每一个过程被描述为由有穷控制结构、实数时钟和实数变量组成的时间自动机，过程之间通过管道和(或者)共享变量来进行通讯，管道用于保证不同自动机中的两个转换同时执行。UPPAAL 主要通过快速搜索机制来验证时钟约束和可达性。它的主要优点是其高效性和使用的方便性。另外，它也可以用于验证更复杂的系统。

UPPAAL 的用户界面包括三个主要部分：系统编辑器(system editor)，模拟器(simulator)和验证器(verifier)。系统编辑器用于创建和编辑要分析的系统，一个系统被描述为一系列过程模板、一些全局声明、过程分配和一个系统定义。模拟器是一个确认工具，它用于检查所建系统模型可能的执行是否存在错误，以此在验证前发现这些错误。验证器通过快速搜索系统的状态空间来检查时钟约束和活性，它还为系统要求的规范和文件提供了一个需求规范编辑器。

UPPAAL 中使用的模型是时间自动机，它将时间自动机进行了一些扩展。首先用户可以声明一般值变量、全局时钟和用于同步的管道(channel)。变量的使用产生了数值约束，因此 UPPAAL 中存在两种形式的约束，如下式所示。

g::=g_clock|g_data|g,g

g_clock::=x<n|x≤n|x= =n|x≥n|x>n

g_data::=ex≤ex|ex<ex|ex= =ex|ex!=ex|ex≥ex|ex>ex

ex::=n|v[ex]|-ex|ex+ex|ex-ex|ex*ex|ex/ex|(g_data?ex:ex)

另外，UPPAAL 中增加了紧迫管道(urgent channel)，紧迫位置(urgent location)和托管位置(committed location)。所谓紧迫管道就是指当这个转换能发生时，它无延迟的发生，紧迫管道对应的转换上不能有时钟约束，但是可以有变量约束。在紧迫位置没有时间延迟，使用它可以减少模型中时钟的个

数，减少分析的复杂度。托管位置也没有时间延迟，下一个转换必须即刻离开所有的托管位置，托管位置的使用可以显著减少状态空间，但是要慎重使用它，因为它可能会导致死锁的出现。

UPPAAL 使用的规范验证语言，形式如下。

Prop::='A[]'Expression|'E<>' Expression|'E[]'Expression|'A<>'Expression|

Expression→Expression

归纳为表 5.1(p 表示一个状态性质)：

表 5.1 UPPAAL 规范验证语言

类型	性质	等价性质
Possibly	E<>p	
Invariantly	A[]p	not E<>not p
Potentially always	E[]p	
Eventually	A<>p	not E[]not p
Lead to	p→q	A[](p imply A<>q)

(1)E<>p 表示 Possibly，E<>p 为真当且仅当在时间自动机中存在一个序列 $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$，使得 s_0 是起始状态，s_n 是 p；

(2)A[] p 表示 Invariantly，等价于 not E<> not p；

(3)E[] p 表示 Potentially always，在时间自动机中，E[] p 为真当且仅当存在一个序列：$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_i \rightarrow \ldots$使得 p 在所有状态 s_i 中得以满足，并且这个序列是无穷的或者在状态(l_n, v_n)终止；

(4)A<>p 表示 Eventually，等价于 not E[]not p；

(5)p→q 表示 Lead to，等价于 A[] (p imply A<>q)。

在建立完模型后就可以使用如上语法对系统进行验证。

5.2 铁路交叉口控制系统的建模及验证

5.2.1 系统描述

铁路与公路之间的交叉口应该有一个控制门，当火车将要通过时，要求这个控制门能够关闭此路口，阻止公路上的汽车或行人在火车通过此路口时接近火车。本文对此实时控制系统建立时间自动机模型，并通过 UPPAAL 进行相应性质验证。首先说明系统应满足的性质。

(1)在火车即将到达路口时，向控制器发出到达信号，控制器对控制门发出指令，要求自动门在规定时间内放下，当自动门处于放下状态时，由控制器向火车发出安全信号，告知火车可以通过；

(2)火车在通过路口时，自动门应处于放下状态；

(3)火车离开后，向控制器发出离开信号，控制器再对控制门发出指令，要求自动门打开；

(4)如果自动门在放下的过程中出现异常。例如机械故障，或有行车意外停在路口，使自动门没有放下，这时控制器应立刻向火车发出停车信号，等待故障排除后再向火车发出可以行驶的信号。

5.2.2 系统建模及验证

按照上面对系统功能需求的描述，分别建立火车、自动门和控制器三个子系统模型，如图 5.1，其中第一个图为自动门，第二个为火车，第三个为控制器。

一共定义了 8 个通道变量，用于三个子系统之间的通信。

appr,stop,go,leave,lower,raise,havedown,error;

火车有一个时钟变量 x，自动门有一个时钟变量 y，控制器无时钟变量，设时间单位为秒。如图，对于火车在发出接近信号 appr 后至少 120 个时间单位进入道口，最多运行 180 个时间单位，若自动门在收到放下信号后 100 个时间单位还没有放下将发出故障信号。下面对于系统的基本性质进行验证。

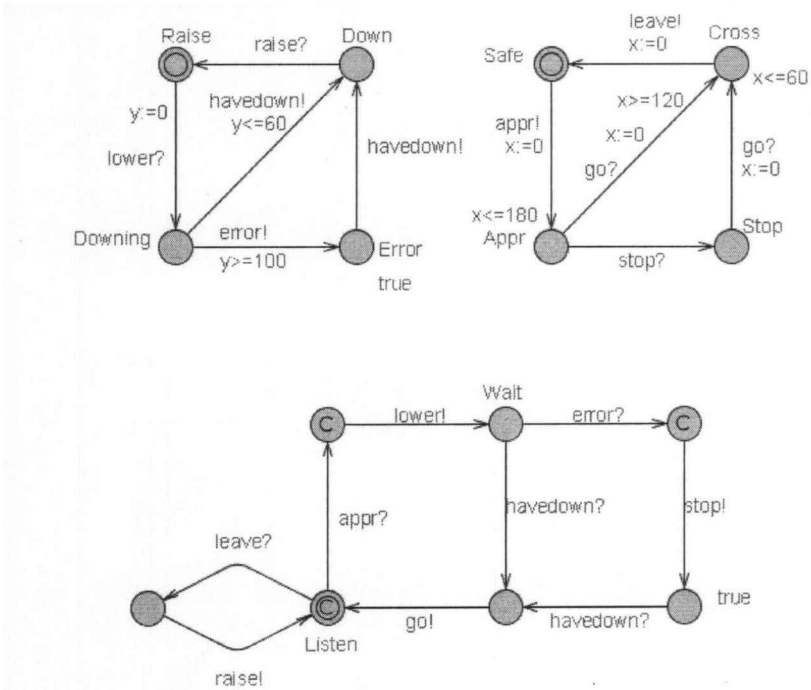


图 5.1 火车、自动门和控制器的时间自动机模型

(1)要求系统无死锁，按照 UPPAAL 语法，满足 A[] not deadlock 成立；

(2)当自动门处于 Error 状态时，火车应处于 Stop 状态，表示为：

Gate1.Error-->Train1.Stop

(3)火车通过路口时，自动门应处于放下状态，表示为：

Train1.Cross-->Gate1.Down

验证后可知，这三条都满足，对于其它性质也可以用类似方法验证。

5.3 本章小结

本章对目前应用比较广泛的 UPPAAL 工具的使用进行了说明，UPPAAL 是基于时间自动机模型的验证工具，在实时系统的模型验证中具有举足轻重的地位。本章利用此工具对一个铁路交叉口的控制系统进行建模并验证了它的安全性和可靠性。

结　论

实时系统的一个共同特点是它们必须在严格的时间限制下做出响应。这些系统设计原型是昂贵的，需要在执行前对时间性质做出详细的预测，并且估计设计的选择。模型检测是作为一种复杂交互系统的调试方法而出现的，传统的模型检测方法不能对时间进行建模。时间自动机作为模拟实时系统行为的模型被引入，已经成为广泛使用的对实时系统进行规范验证的形式化方法之一，在未来的模型检测中将会发挥出更大的优势。

本文在对时间自动机理论进行深入研究的基础上，介绍了有穷状态时间自动机的构造方法、时钟带的表示方法，分析了时间自动机在实时系统建模及验证中的应用，主要完成了以下工作：

(1)分析了时间自动机的研究背景，介绍了时间自动机的语法和语义及时间自动机的主要性质；讨论了构造有穷状态时间自动机的方法，分析了它们特点。

(2)由于时钟带的表示方法直接影响利用带自动机进行验证的效率，本文对时钟带的表示及相关操作进行分析，分离出一类时钟带，提出了这类时钟带 DBM 表示的一种范式化算法，证明了它的正确性，并通过实验验证了其优越性。

(3)通过对时间自动机的研究，学习了基于时间自动机模型的工具 UPPAAL 的使用，对一个铁路交叉口的自动控制系统进行建模，并验证了它的安全性和正确性，可以作为学习人员的参考。

本文所提出的 DBM 范式化算法目前仅适用于一类时钟带的表示，下一步工作将尝试是否可以把一般形式的时钟带转化为这类时钟带，并进一步研究时间自动机在其它领域的应用。

参考文献

[1] Z.Manna, A.Pnueli.Models for Reactivity. Acta Informatica. 1993, 30(7):609-678P

[2] R.Alur, C.Courcoubetis, and D.L.Dill.Model Checking for Real-time Systems. Proceedings of the fifth IEEE Symposium. Logic in Computer Science.IEEE Computer Society Press,1990:414-425P

[3] E.M.Clarke, O.Grumberg, D.Peled. Model Checking. MIT Press, 1999:1-24P

[4] R.Alur,D.L.Dill.A Theory of Timed Automata.Theoretical Computer Science.1994,126(2):183-235P

[5] R.Alur.Timed Automata.In Halbwachs N,Peled D,eds.Proceedings of the 11th Conference on Computer-Aided Verification.LNCS 1633, London:Springer-Verlag,1999:8-22P

[6] R.Alur, Courcoubetis.N.Halbwachs, D.L.Dill, and H.Wong-Toi. Minimization of Timed Transition Systems.Proceedings of the third Conference on Concurrency Theory.Lecture notes in Computer Science ,Springer-velag,1992,630:340-354P

[7] Rina S.Cohen and Arie Y.Gold,Theory of ω-Language.Journal of Computer and System Science.1977,15:168-184P

[8] Wang.F.Efficient Verification of Timed Automata with BDD-like Date-Structures.Lecture notes in Computer Science 2575.2003: 189-205P

[9] Zhao Jianhua, Dang V anhung, Checking Timed Automata for Linear Duration Properties. Journal of Computer Science&Technology. 2000,15(5):121-128P

[10] D.Dill.Timing Assumptions and Verification of Finite-State Concurrent Systems. Proceedings of the Automatic Verification Methods for Finite State Systems. LNCS 407, Berlin: Springer-

Verlag, 1980:197-212P

[11] M. Bozga, O. Maler, A. Pnueli, and S. Yovine. Some Progress in the Symbolic Verification of Timed Automata. Computer-Aided Verification, CAV' 97, Israel:Springer-Verlag, 1997:179-190P

[12] T. A. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic Model-Checking for Real-time Systems. Proceedings 7th Symposium. Logics in Computer Science, 1992:394-406P

[13] J. Bengtsson. Clocks, DBMs and States in Timed Systems, Ph. D. thesis. Department of Information Technology, Uppsala University, Uppsala, Sweden, 2002:1-24P

[14] Tobias Amnell, Gerd Behrmann, Johan Bengtsson, etc. UPPAAL-Now, Next, and Future. MOVEP2000, LNCS 2067. 2001:99-124P

[15] Http://www.uppaal.com

[16] Godefroid P. Using partial orders to improve automatic verification methods. Proceedings of the 2nd Workshop on Computer-Aided Verification, LNCS 531. Berlin: Springer, 1990:176-185P

[17] Dang. Z, Ibarra. O, Kemmerer. R. Generalized Discrete Timed Automata: Decidable Approximations for Safety Verification. Theoretical Computer Science. 2003, 296(1):59-74P

[18] Beauquier. D. On Probabilistic Timed Automata. Theoretical Computer Science. 2003, 292(1):65-84P

[19] Mateus. P, Morais. M, Nanes. C, etc. Categorical Foundations for Randomly Timed Automata. Theoretical Computer Science. 2003, 308 (1-3):393-427P

[20] Springintveld. J, Vandrager. F, Rdargenio. P. Testing Timed Automata. Theoretical Computer Science. 2001, 254:225-257P

[21] Salvatore. LT, Margherita. N. Timed Tree Automata with an Application to Temporal Logic. Acta informatica. 2001, 38:89-116P

[22] Buchi J R. On a Decision Problem in Restricted Second Order

Arithmetic. Proceedings Internat Congr Logic. Methodology and Philosophy of Science, Stanford: Stanford University Press, 1962: 1-120P

[23] Kupferman, O. and M. Y. Vardi, Model Checking of Safety Properties. Formal Methods in System Design. 2001, 19:291-314P

[24] 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, 1996, 12:55-82P

[25] F. Demichelis, W. Zielonka. Controlled timed automata. Proceedings 9th International Conference on Concurrency Theory. LNCS 1466, Springer, 1998:455-469P

[26] Bouyer. Forward Analysis of Updatable Timed Automata. Formal Methods in System Design. 2004, 34(3):281-320P

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

[28] Behrmann G, Larsen KG, Pearson J, Weise C, Wang Y. Efficient Timed Reachability Analysis Using Clock Difference Diagrams. Proceedings of the CAV' 99. LNCS 1633, Trento: 1999:341-353P

[29] Asarin E, Bozga M, Kerbrat A, Maler O, Pnueli A, Rasse A. Data-Structures for the Verification of Timed Automata. Proceedings of the HART' 97. LNCS 1201, Grenoble: 1997:346-360P

[30] Wang F. Region Encoding Diagram for Fully Symbolic Verification of Real-time Systems. In Martin R, ed. Proceedings of the 24th IEEE COMPSAC 2000(Computer Software and Applications Conference). Taipei: IEEE Computer Society, 2000:509-515P

[31] Beyer D, Lewerentz C, Noack A. Rabbit: A tool for BDD-based verification of real-time systems. CAV. LNCS 2725, London: Springer-Verlag, 2003:122-125P

[32] KimG. Larsen, Paul Pettersson, and Wang Yi. Compositional and Symbolic Model Checking of Real-time Systems. Proceedings of the 16th IEEE Real-time Systems Symposium, Pisa, Italy: IEEE Computer Society Press, 1995:76-87P

[33] Peter Linz. 形式语言与自动机导论. 孙家骕译. 第三版. 机械工业出版社, 2005:27-45 页

[34] 李广元, 唐稚松. 带有时钟变量的线性时序逻辑与实时系统验证. 软件学报. 2002, 13(1):33-41 页

[35] 晏荣杰, 李广元, 徐雨波等. 有限精度时间自动机的可达性检测. 软件学报. 2006, 17(1):1-10 页

[36] 陈靖. 稠密时间表示及冗余消除. 软件学报. 2003, 14(10):1681-1691 页

[37] 宋煌, 庄雷, 苏锦祥等. 一种改进的区域自动机构造方法. 计算机研究与发展. 2002, 39(5):607-611 页

[38] 支小莉, 童维勤, 戎璐. 时间自动机的自动抽象算法. 西南交通大学学报. 2004, 39(5):670-674 页

[39] 刘春明, 晏荣杰, 徐雨波. 有限精度时间自动机的时钟表示. 计算机应用研究. 2006, 7(21):23-31 页

[40] 周清雷, 汪国安. 关于 ω-有穷自动机的接受条件. 河南大学学报(自然科学版). 1999, 29(3):36-41 页

[41] 刘栋. 时间自动机可达性研究. 郑州大学硕士学位论文. 2004:12-25 页

攻读硕士学位期间发表的论文和取得的科研成果

[1] 孙全勇,李健利.SOAP技术在异构环境中的应用.信息技术(增刊).2005:64-66页

致　　谢

在我攻读硕士学位期间，自始至终得到了许多老师、同学的帮助，在此向他们表示深深的谢意。

衷心地感谢我的导师李健利副教授，在我两年半的学习、工作中，李老师给予了我悉心的指导和无微不至的关怀。衷心的感谢刘群教授，他对我的论文给予了极大的指导，他严谨求实的治学态度、一丝不苟的敬业精神、正直的处世原则和对我的严格要求对我有着深刻的影响，使我终生受益，在此表示衷心的感谢。

感谢实验室的所有老师及同学们。在他们的帮助下，开拓了我的视野，拓宽了我的知识面。在我论文完成期间同学们所给予我的帮助与支持，彼此间真挚的友谊使我终身难忘。

最后要感谢所有关心我、帮助我的人，特别是在实验室一起工作的同学们。在这样的一个温暖的集体中学习，我感到非常愉快。感谢教研室所有老师提供的帮助，感谢各位同学对我的关心，感谢父母及亲人对我的支持与鼓励。