稠密时间表示及冗余消除*

陈 靖+

(中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

Dense Time Representation and Redundancy Elimination

CHEN Jing+

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

+ Corresponding author: Phn: 86-10-62562796, Fax: 86-10-62563894, E-mail: chen@lifo.univ-orleans.fr

http://www.ios.ac.cn

Received 2002-09-03; Accepted 2003-04-14

Chen J. Dense time representation and redundancy elimination. *Journal of Software*, 2003,14(10):1681~1691.
http://www.jos.org.cn/1000-9825/14/1681.htm

Abstract: The efficiency of an algorithm depends greatly on the data structure adopted in practice, while the processing of useless data can cause not only much waste of memory but also much waste of time. Hence to eliminate redundant information is one of the main focuses in the research of algorithms, and the issue has received extensive attention in real-time field (especially those based on dense/continuous time semantics) for many years. After investigating existing problems in finite representation and operations, and by analyzing the dependence relation in information, a method is presented for eliminating redundant information in representation of dense time, which is developed based on an modification of the procedure of 'Normalization'. The correctness of the method is proved and the efficiency is tested by experiences.

Key words: real-time; timed automata; data structure; redundancy elimination; shortest path finding algorithm; optimization

摘 要: 算法的效率在很大程度上依赖于实际采用的数据结构.对无用数据的处理不仅会带来空间存储上的浪费,而且也会进一步造成时间上的浪费.因此,消除信息冗余一直是算法研究的一个重点.在当前实时领域(尤其是在基于稠密/连续时间语义)的算法研究中,该问题十分突出.从信息之间的依赖关系入手,分析了在对连续时间进行有穷表示和操作中存在的问题,通过改进"范式化"处理过程,给出了进行冗余信息消除的一种方法以及其正确性证明,并通过实验测试了改进的效率.

关键词: 实时;时间自动机;数据结构;冗余信息消除;最短路径算法;优化

中图法分类号: TP301 **文献标识码**: A

本文要解决的是当前实时领域中对实数域上连续时间的有效表示和操作的问题.该问题自时间自动机[1]提出后,随着互模拟判定算法和模型检测算法的研究深入而日显重要和迫切.国际上最早也最常用的数据结构

* Supported by the National Natural Science Foundation of China under Grant Nos.69833020, 60203028 (国家自然科学基金)

第一作者简介: 陈靖(1975－),男,湖南株洲人,博士,主要研究领域为进程代数,形式化方法.

DBM(difference bound matrix)[2]以及后来如CDD(clock difference diagrams)[3],NDD(numerical decision diagram)[4],RED(region-encoding diagram)[5]等,都在尝试提高数据结构的空间和时间效率,但同时也都存在着一些问题和困难需要解决.本文选择从信息依赖的角度入手,对数据操作中的"范式化"处理进行研究和改进.

范式通常是指表达式或数据结构的某种标准形式,而范式化则是指将一般表达式或数据结构转换成统一的范式的过程.范式化在不同场合和不同要求下的含义和作用是不同的,在算法中常用于将数据表示标准化、惟一化,从而保证相关操作的正确性.下面通过一个简单示例引出我们的问题.如图1所示,区域$D=\{(x_1,x_2)|x_1>1\wedge x_1\le 4\wedge x_2\ge 1\wedge x_2\le 3\}$,需要计算其"后继"$D\uparrow=\{(x_1+\delta,x_2+\delta)|\delta\in R^{\ge 0},(x_1,x_2)\in D\}$($R^{\ge 0}$为非负实数集).

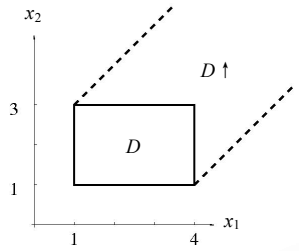

Fig.1 Example 1
图1 例1

不妨称$\Phi_D=x_1>1\wedge x_1\le 4\wedge x_2\ge 1\wedge x_2\le 3$为$D$的公式表示,要得到$D\uparrow$的公式表示,可通过对$\Phi_D$进行如下操作得到:

第1步先将Φ_D"范式化"得到$\mathrm{nf}(\Phi_D)=x_1>1\wedge x_1\le 4\wedge x_2\ge 1\wedge x_2\le 3\wedge x_1-x_2\le 3\wedge x_1-x_2>-2$.容易看到,类似经典逻辑中的合取范式,$\mathrm{nf}(\Phi_D)$和$\Phi_D$实际上是等价的公式,因为$\mathrm{nf}(\Phi_D)$中新增的合取项皆从$\Phi_D$导出(如$x_1\le 4\wedge x_2\ge 1\Rightarrow x_1-x_2\le 3$).第2步再删去$\mathrm{nf}(\Phi_D)$中的$x_1\le 4$和$x_2\le 3$(即去掉$x_i$的有限上界).容易验证最后所得到的不等式$x_1>1\wedge x_2\ge 1\wedge x_2\le 3\wedge x_1-x_2\le 3\wedge x_1-x_2>-2$即为$D\uparrow$的公式表示.

从此例可以得出:(1) 该"范式化"并未增加新信息,只将由原式可导出的信息完全推导出来,新生成的是一些冗余信息;(2) "范式化"步骤在这里是必要的,若跳过不做,直接删去项$x_1\le 4$和$x_2\le 3$后则会失去$x_1-x_2\le 3\wedge x_1-x_2>-2$的信息,从而最后所得到的不等式$x_1>1\wedge x_2\ge 1$并非$D\uparrow$的公式表示.

本文的贡献在于,通过对"范式化"的改进,给出了检查数据结构中冗余信息的方法,在保证正确性的前提下,在数据操作中避开了对冗余信息的不必要计算,解决了存在于DBM,CDD等数据结构中的类似问题,提高了实时领域中数据结构及相关算法的效率.

1 实时领域中的数据结构现状

这一节首先引入时间自动机定义并给出其操作语义,然后介绍用于表示时间的几种数据结构及其相关操作的语义和作用.最后分析其中存在的问题.

1.1 时间自动机

令C为时钟变量的可数无穷集,其中的元素用$c,c_1,c_2\ldots$表示,d为任意自然数.时钟约束用BNF记号定义如下:

$$\Psi ::= c\sim d\,|\,c_1-c_2\sim d\,|\,\neg\Psi\,|\,\Psi\wedge\Psi,\quad \sim\in\{\le,<,\ge,>\}.$$

我们用*Constr*表示满足上述定义的时钟约束的集合$\{\Psi,\phi,\ldots\}$,若约束ϕ中不含任何逻辑运算符,则称其为原子约束.$R^{\ge 0}$表示非负实数集,计值ρ是C到$R^{\ge 0}$的全函数.对任意$\phi\in\mathrm{Constr}$,$\rho(\phi)\in\{\mathrm{true},\mathrm{false}\}$为将时钟变量用$\rho$计值后算出的$\phi$的真值.我们记$\rho(\phi)=\mathrm{true}$为$\rho\vdash\phi$.$\theta$是时钟置零操作,即形如$\bar{c}:=0$的赋值,其中$\bar{c}$是形如$\langle c_1,c_2,\ldots\rangle$的时钟向量,其语义为将$\bar{c}$中的所有时钟变量全部置0.时钟置零操作的集合记作*Reset*.规定$Cl(\bar{c}:=0)=\{\bar{c}\}$给出$\bar{c}$中时钟变量的集合.给定非负实数$\delta$,时钟置零操作$\theta$,我们定义$(\rho+\delta)$(时间流逝)和$\rho\theta$(时钟复位)如下:

$$\forall c\in C,\quad (\rho+\delta)(c):=\rho(c)+\delta;$$

$$\forall c\in C,\quad \rho\theta(c):=\begin{cases}0, & c\in Cl(\theta)\\ \rho(c), & c\notin Cl(\theta)\end{cases}.$$

定义1(时间自动机(timed automata)). 时间自动机A是一个五元组(Σ,N,n_0,E,Inv),其中:

- Σ为动作集,其中元素用$\alpha,\beta,\ldots$来表示.
- N为结点的有穷集.
- $n_0(\in N)$为初始结点.
- $E\subseteq N\times(\mathrm{Constr}\times\Sigma\times Reset)\times N$为边的有穷集.
- $Inv:N\to Constr$给出结点上的时间不变式.

给定时间自动机的一个结点 n 以及时钟变量的一个计值 ρ,得到的二元组 $n\rho$ 为系统的一个状态,时间自动机的行为通过状态转换得到体现.

简单地说,其状态转换可分为两种:一种为执行一条边上的迁移动作后转换至下一状态;另一种为选择留在本结点进行等待,让所有时钟变量的值在该结点按相同速率增加.后者称作延时动作,其状态转换只体现在 ρ 的变化上,但延时的长度通常受到限制,要求经过该延时后的时钟变量值必须满足该结点上的不变式.给定一个时间自动机,可根据一定的规则生成相应的状态集及该状态集上的状态迁移图.确定其迁移关系的规则如下:

$$\frac{n \xrightarrow{\varphi,\alpha,\theta} n'}{n_\rho \xrightarrow{\alpha} n'_{\rho\theta}}, \quad \rho \models \varphi \wedge Inv(n), \rho\theta \models Inv(n');$$

$$\frac{}{n_\rho \xrightarrow{\delta} n_{\rho+\delta}}, \quad \forall r.0 \le r \le \delta, \rho + r \models Inv(n).$$

其中 $\delta \in R^{\ge 0} - \{0\}$ 即为延时动作.根据上述规则,可对时间自动机的逐个状态 $n\rho$ 的行为进行解释.由于这里时钟变量的值域为非负实数,因而得到的状态空间是无限的.例如,最后这条语义是指,若计值不违背结点 n 上的不变式(时钟约束 $Inv(n)$),则系统在状态 $n\rho$ 可做 δ 个单位时间的延时后转换至状态 $n\rho+\delta$.

通常这样的 δ 有无穷多个,从而导致这种延时动作也有无穷多个,因此使简单进行状态搜索的算法不可行.时间自动机模型由于适当地限制了时钟约束的形式,使得在互模拟判定[6~8]和模型检测[9~11]算法中根据需要可将状态空间划分为在一定意义下等价的有限多个区域,使无限状态系统的问题可被转化到有限状态系统上解决,因此,在有穷化问题解决以后,目前实时领域的一个受到广泛关注的热点就是如何高效地表示和操作这些时间区域.本文开始部分提到的计算 $D\uparrow$ 的例 1,其语义即是计算从 D 中所有计值 ρ 进行任意长度延时后得到的所有计值 $\rho+\delta$ 的集合,注意到虽然由无穷个计值导致的状态无穷,但可以存在有穷的表示,而且可使计算步骤和结果形式都是有穷的.另外还有时间区域的"时钟置零"操作,如 $D\downarrow_{\{\bar{c}\}}=\{\rho[\bar{c}:=0] \mid \rho \in D\}$ 以及时间区域的"交"、"并"、"求补"等的高效实现也都是实时领域的迫切要求.文献[2~5]反映了近年国际实时领域在这方面的主要研究成果和状况,

本文主要通过对其中两种数据结构 DBM 和 CDD 进行分析来说明所需解决的问题和相应的解决方法.

1.2 从DBM到CDD

DBM[2]全称为 Difference Bounds Matrices,其核心思想是用矩阵来表示一块连续的时间区域,而矩阵中每个元素均为由一个不等运算符和一个整数组成的对偶,下面逐步引入 DBM 的定义.

令 $B=Z\times\{<,\le\} \cup \{(\infty,<),(-\infty,<)\}$(其中 Z 为整数集),且符号<和≤有全序关系:<严格小于≤.我们定义 B 中元素间的偏序 $(x,\sim) \sqsubseteq (x',\sim')$ 当 $x<x'$ 或当 $x=x'$ 且 $\sim \sqsubseteq \sim'$(即字典序),$\min(\sim,\sim')$ 给出 $\{\sim,\sim'\}$ 的最小元.令 $T=\{0,1,2,\ldots\}$ 为自然数的有限子集,则一个 DBM 为一个映射 $M:T^2 \to B$.其语义解释如下:设时钟变量依次排序为 $x_1,x_2,\ldots$,第 i 行 j 列矩阵元素 $M(i,j)=(d,\sim)$ 表示约束 $Cond(M(i,j))=x_i-x_j\sim d$,令 $R(M)=\{\rho \mid \rho \vdash \wedge_{(i,j)\in T^2} Cond(M(i,j))\}$ 为矩阵 M 表示的区域.注意到由于 B 的定义使这里的关系符~仅限于<和≤,而对约束 $x_i-x_j\sim d, \sim \in \{>,\ge\}$.我们先定义相应的求补操作 ^{c}:$>^c=<$,$\ge^c=\le$,然后通过 $M(j,i)=(-d,\sim^c)$ 即可表示.另外,在上述定义中,时钟变量的下标从 1 开始排列,原因是 x_0 有特殊用途,即如果要表示 $x_i\sim d$,则视作式 $x_i-x_0\sim d$ 进行处理即可.

下面给出一个只涉及两个时钟变量的例子.如图 1 所示的例 1 给出了一块区域 $D=\{(x_1,x_2) \mid x_1>1 \wedge x_1\le 4 \wedge x_2\ge 1 \wedge x_2\le 3\}$,其 DBM 表示如下:

	0	1	2
0	(0,≤)	(−1,<)	(−1,≤)
1	(4,≤)	(0,≤)	(∞,<)
2	(3,≤)	(∞,<)	(0,≤)

通过在 B 上定义相应算子,可以得到如下的一个正则代数(regular algebra)[12]:

$$\mathbf{n}=(\infty,<),$$
$$\mathbf{e}=(0,\leq),$$
$$(x,\sim)+(x',\sim')=(x+x',\min(\sim,\sim')),$$
$$(x,\sim)\sqcap(x',\sim')=\begin{cases}(x,\sim), & \text{若}(x,\sim)\sqsubseteq(x',\sim')\\(x',\sim'), & \text{否则}\end{cases},$$
$$(x,\sim)^*=\begin{cases}(0,\leq), & \text{若}(0,\leq)\sqsubseteq(x,\sim)\\(-\infty,<), & \text{否则}\end{cases}.$$

其中,该代数的“乘”为“+”,“加”为“⊓”,“*”仍为 Kleene 星号,**n** 和 **e** 为相应常量“零元”和“单位元”.在该正则代数上定义的 $n\times n$ 矩阵(DBM)又可构成一个正则代数,其中只需令“+”为矩阵“乘”,“⊓”为矩阵“加”(以上述标量算子⊓和+的定义为基础).该代数的零元 **N** 为一个所有元素均为$(\infty,<)$的矩阵,而单位元 **E** 则形如 $\mathbf{E}(i,i)=(0,\leq)$且$\forall i\neq j.\mathbf{E}(i,j)=(\infty,<)$.$M^*$定义为 $M^0\sqcap M^1\sqcap\ldots$,而 $M^0=\mathbf{E}$.

这些矩阵上的偏序可定义为 $M\sqsubseteq M'$ 当且仅当 $\forall i,j.M(i,j)\sqsubseteq M'(i,j)$. 由区域 $R(M)=\{\rho|\rho\models\wedge_{(i,j)\in T^2}Cond(M(i,j))\}=\{\rho|\forall i,j\in T.\rho\models Cond(M(i,j))\}$,显然可以得到 $M\sqsubseteq M'$蕴含 $R(M)\subseteq R(M')$,并且 $R(M\sqcap M')=R(M)\cap R(M')$.

由于空区域都是相同的,而表示它们的 DBM 却各种各样,我们选择其中一个作为代表,即 $M_\varnothing$,为满足$M_\varnothing(0,0)=(-\infty,<)$的一个矩阵.对任意矩阵 M,其范式 nf(M)定义如下:

$$\mathrm{nf}(M)=\begin{cases}M^*, & R(M)\neq\varnothing\\M_\varnothing, & \text{否则}\end{cases}.$$

求解范式 nf(M)的过程即 M 的范式化过程.容易证明,对任意两个同阶 DBM 矩阵,若表示的区域相同,则一定会有相同的范式,或换句话说,表示任何区域的范式(矩阵)是惟一的,其详细证明以及其他性质的证明参见文献[2].

下面,我们从另一角度讨论范式中元素值的生成及各元素值之间的信息依赖问题.

我们称 T 中的 $i_1,i_2,\ldots,i_M$ 序列为一路径,则矩阵 M 中该条路径的代价定义为 $M(i_1,i_2)+M(i_2,i_3)+\ldots+M(i_{m-1},i_M)$.若 M'=nf(M),则 $M'(i,j)$等于 M 中从 i 到 j 的所有路径中代价最小的路径的代价值.因此,求 DBM 范式的问题被转化为最短路径的求解问题[2,12].并且,根据经典的理论,若矩阵(方阵)的阶数为 n,求解 M^*的最坏情况时间复杂度为 $O(n^3)$.如前述例子,DBM 只需迭代一次就可算出其范式矩阵:

	M				M				M'		
	0	1	2		0	1	2		0	1	2
0	(0,≤)	(−1,<)	(−1,≤)	0	(0,≤)	(−1,<)	(−1,≤)	0	(0,≤)	(−1,<)	(−1,≤)
1	(4,≤)	(0,≤)	(∞,<)	+ 1	(4,≤)	(0,≤)	(∞,<)	⇒ 1	(4,≤)	(0,≤)	(3,≤)
2	(3,≤)	(∞,<)	(0,≤)	2	(3,≤)	(∞,<)	(0,≤)	2	(3,≤)	(2,<)	(0,≤)

其中 M'即等于 M 的范式.注意,M'中的 $M'(2,1)=M(2,0)+M(0,1)$,即 M 中从 2 到 1 的最小代价路径是从 2 到 0 再到 1,而 $M'(1,2)$也可以同理算出.也即,$M'(2,1)$和 $M'(1,2)$是依赖其他数据计算来的,存储的是冗余信息,而其余几项如 $M'(1,0)$,$M'(0,2)$则存储的是必要信息.从该例可以看到,如果不进行“范式化”就不会生成这些冗余信息,但许多操作若预先不进行“范式化”则会导致后续操作的结果错误,如文章开始部分给出的例 1.另外,许多操作在完成后也要进行“范式化”,例如两个区域 D_1,D_2 的 DBM 在进行⊓运算后,结果必须进行范式化处理,否则,如果$D_1\cap D_2=\varnothing$,则无法查知其 DBM 结果表示中为空的区域.

以上提到的都是“范式化”的必要性,下面我们对其引入的冗余信息的危害进行分析,最后还将举例说明,此类冗余信息不仅仅由“范式化”引入,由其他操作也可以产生.

上述冗余信息危害最大之处在于对区域的求补运算.容易看到,一个 DBM 中的每个元素都对应着一个原子约束,整个 DBM 表示的区域即所有这些约束合取后表示的区域,要得到上述区域 D 的“补”区域,由笛·摩根律,只需将每个元素表示的约束取反后再进行析取即可.

例如,若使用 D 的范式矩阵 M'求补,则得到的“补”区域 $\overline{D}$ 用公式表示为 $x_1\leq 1\vee x_2<1\vee x_1>4\vee x_2>3\vee x_1-x_2>3\vee$

$x_2-x_1\geq 2$.注意,其实该公式的最后两项分别是由 $Cond(M'(1,2))$和 $Cond(M'(2,1))$求反而来,由前述定义 $M'(1,2)$和 $M'(2,1)$存储的是冗余信息,因此对其求反显然也是多余的,造成了浪费.此其主要危害.另外,注意到DBM表示的区域为凸区域.而区域的"并"和"求补"则会引进凹区域,故对于凹区域来说,要么简单并置多个 DBM 来表示(如上述析取式可用 6 个 DBM 并置表示),要么引进新的数据结构.为了有效地表示凹区域,目前有多种数据结构,本文将对其中的 CDD[3]进行讨论,它是在 DBM 的基础上发展而来的,因篇幅所限,本节不给出其严格定义,只通过几个简单示例说明它与 DBM 的联系以及仍然存在的相同问题.

CDD 是一个有向图,它的结点分成两类:

- 终端结点.该结点要么为 1,要么为 0,不存在后继结点.
- 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).

图 2 给出了一个凹时间区域及其 CDD 表示.

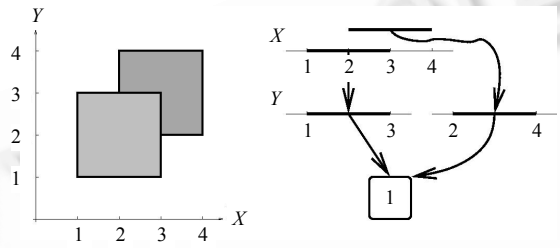

Fig.2 Example 2 (time regions and CDD representation)

图 2 例 2(时间区域及 CDD 表示)

容易观察到,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(注意,为了易读,省略了图中通向终端结点 0 的所有结点和路径).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 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.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,由于本文的侧重点及篇幅有所限制,详细步骤见文献[3],这里不再赘述.

CDD 为了消除冗余,保证操作的正确性,在范式化处理中也加入了许多措施,但下面通过对图 3 中两个 CDD 相交及其范式化结果做进一步的分析,可以说明我们所指的信息冗余及其危害仍然存在.

图 3 中将两 CDD(C_1,C_2)相交后(得到 $C_3=C_1\cap C_2$)再进行范式化得到的结果 $C_4(=nf(C_3))$的 x-y 结点存储的信息即为冗余信息,因为 $-1<x-y\leq 2$ 可由 $2<x\leq 3\wedge 1\leq y\leq 3$ 推导出来.因而在求补时对 x-y 结点的处理完全是不必要的.通过该例也同时可以看到,冗余信息不仅仅是由"范式化"带来的:这里即使不进行范式化,$C_1\cap C_2$ 也出现一个没有用处(即区间过于宽泛,非该区域真正的边界)的 x-y 结点,直接对 $C_1\cap C_2$ 求补也存在同样的信息冗余问题.

下面针对该问题给出我们的解决方法.该方法对原"范式化"进行了改进,加入了对冗余信息的检查,使得后续操作可以避开对冗余信息的多余计算.

2 改进后的"范式化"

下面我们先引入几个基本概念,再给出改进后的"范式化"方法,并以 DBM 为例介绍相应的实现方法.

定义 2(平凡约束). 平凡约束是指属于以下形式之一的时钟约束:

(1) $x-x=0$;

(2) $x\geq 0$;

(3) $x<\infty, x\leq\infty$ 或 $x-y<\infty, x-y\leq\infty$.

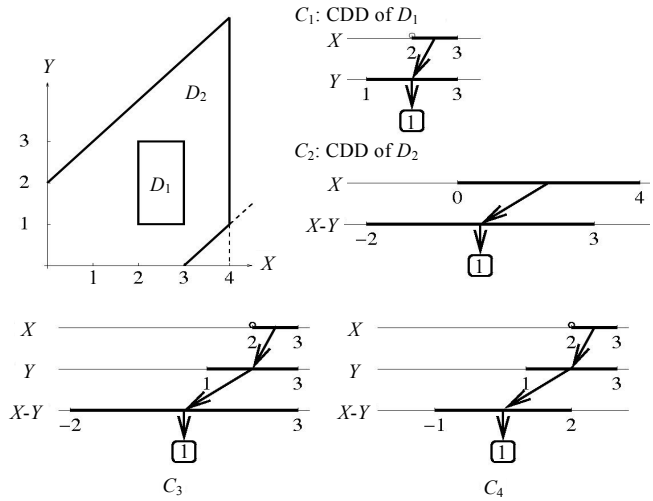

Fig.3 Example 3 (intersection and normalization of two CDDs)

图 3 例 3(两 CDD 的相交及其范式化)

定义 3(支撑集). 若时钟约束集$\{\phi_i|i\in I, I$ 为有限下标集$\}$以及约束$\phi\notin\{\phi_i\}$满足蕴含关系$\wedge i\phi_i\Rightarrow\phi$,则称约束集$\{\phi_i\}$为$\phi$的支撑集.

定义 4(基约束和导出约束). 给定一个合取式$\phi=\wedge i\phi_i, \phi_i\in Constr$ 为一个原子约束.称子集 $S\subseteq\{\phi_i|\phi=\wedge i\phi_i\}$为$\phi$的基约束集,若满足$\phi\Leftrightarrow\wedge_{\Psi\in S}\Psi$且 S 不含平凡约束.S 中的元素又称为公式ϕ的一个基约束.给定一个合取式,若其中一个约束存在只由某基约束集的元素和平凡约束组成的一个支撑集,则称该约束为导出约束.

定义 4 给出的导出约束是指可由基约束和平凡约束蕴含(导出)的约束.本文的目标是希望在存储时可删去导出约束,只保留(存储)蕴含它的其他约束,从而达到消除冗余、减小存储空间的效果.

直观上,对于给定合取式,基约束集是其合取项集的子集,基约束集中元素的合取表示的时间区域等价于原合取式表示的时间区域.一个公式的基约束集并不惟一,显然最坏情况下公式所有合取项均为基约束.我们希望基约束集尽可能地小,以利于提高处理的效率.

由于判定支撑集、减小基约束集的过程需要在求解范式的过程中进行,而如前所述,求解 DBM 范式的过程可对应至求解最短路径的过程,因此一般均通过逐步迭代的方式进行计算.若在迭代过程中不删除任何元素,则可不显式地使用支撑集;而若需在迭代过程中逐步删除某些元素、减小存储空间的占用,则原来的办法不能适用.下面通过一个例子说明支撑集在其中起到的作用.对基约束集 $B=\{x_1=0, x_2=0, x_1-x_2=0\}$,根据 $x_1=0\wedge x_1-x_2=0\Rightarrow x_2=0$,可从 B 中删去 $x_2=0$,得到 $B=\{x_1=0, x_1-x_2=0\}$,而 $x_2=0\wedge x_1-x_2=0\Rightarrow x_1=0$,是否也可以把 $x_1=0$ 从 B 中删去呢?显然不能.虽然 B 中所有约束的合取可蕴含 $x_2=0$ 进而蕴含 $x_1=0$,但该过程等价于 $x_1=0\wedge x_1-x_2=0\Rightarrow x_1=0$,因而据此进行删除是没有意义的.但在以增量方式进行的算法中,该问题并非显然,除非为每个约束都记录其当前的支撑集.例如,在计算 $x_1=0\wedge x_1-x_2=0\Rightarrow x_2=0$ 时得到$\{x_1=0, x_1-x_2=0\}$为 $x_2=0$ 当前的支撑集;再当计算 $x_2=0\wedge x_1-x_2=0\Rightarrow x_1=0$ 时,可根据 $x_2=0$ 当前的支撑集计算当前蕴含 $x_1=0$ 的约束集也为$\{x_1=0, x_1-x_2=0\}$,由于该集包含 $x_1=0$ 自身,因此不能作为 $x_1=0$ 的新支撑集,从而避免将 $x_1=0$ 从 B 中错误地删除.但是,对每个约束均存储一个支撑集的做法为算法带来了较大的存储上的开销,尤其是支撑集并非最终结果所需要.因此,下面给出我们解决此问题的一个算法,说明实际上不存储支撑集也可达到相同的目标.该算法以数据结构 DBM 为例(后面我们将说明该方法同样适用于 CDD 等其他数据结构),在原求解范式的基本算法上进行了一些扩充,不存储任何支撑集,并且时间复杂度与原基本算法相同.

我们希望,对给定公式的一个基约束集 B,在满足基约束集条件的前提下(保证其中所有元素合取表示的时间区域不变),存在一个过程将其中的元素个数减到最少,从而减小其中的信息冗余.下面先给出在该过程中可

能需要满足的一些性质:

性质 1. 集合 B 中的基约束个数一定不会增加,只可能减少.

性质 2. 在整个过程中,由 B 中所有约束合取表示的区域始终等于原公式表示的区域.

如前所述,DBM 范式求解的核心过程(除去空区域判定等过程)对应于一个所有点与点之间的最短路径求解过程.不失一般性,我们采用经典的 Floyd-Warshall[13]算法求解该问题的基本算法.给定一个阶数为 n 的 DBM 阵 M,Floyd-Warshall 算法在此问题上的递归定义如下:

$$W^{\langle k\rangle}(i,j) = \begin{cases} M(i,j), & 若k=0 \\ \min\{W^{\langle k-1\rangle}(i,j), W^{\langle k-1\rangle}(i,k-1)+W^{\langle k-1\rangle}(k-1,j)\}, & 若k>0 \end{cases}.$$

n 步迭代后所得到的 $W^{\langle n\rangle}$即为所求范式矩阵.针对上述信息冗余问题,给定矩阵 M,定义对偶(i,j)为约束 $Cond(M(i,j))$的类型,满足下述条件的类型集 $Base\subseteq T^2$ 为矩阵 M 的基类型集:

$$\wedge_{(i,j)\in Base} Cond(M(i,j)) \Leftrightarrow \wedge_{(i,j)\in T^2} Cond(M(i,j)). \quad (*)$$

容易看到,满足该条件的左式所有合取项的集合即为右式的一个基约束集.对应于基约束的概念,称$(i,j)\in Base$ 为 M 的一个基类型.

给定一个表示非空区域的矩阵 M 以及一个非空 Base 集(可取$\{(i,j)\in T^2 | Cond(M(i,j))$为非平凡约束$\}$),算法返回值 $W^{\langle n\rangle}$即为所求范式,且最终 Base 集为 $W^{\langle n\rangle}$的基类型集.算法伪代码如下:

fun S Path_Finding(M)=

```
1:   n=M 的阶数
2:   W⟨0⟩=M
3:   for k=1 to n
4:   {
5:       for i=0 to n−1
6:           for j=0 to n−1
7:           {
8:           if W⟨k−1⟩(i,(k−1))+W⟨k−1⟩((k−1),j)⊑W⟨k−1⟩(i,j) then
9:               {
10:              W⟨k⟩(i,j):=W⟨k−1⟩(i,(k−1))+W⟨k−1⟩((k−1),j);
11:              if k−1<min(i,j) and (i,j)∈Base then
12:              Base:=Base\{(i,j)};
13:              }
14:              else W⟨k⟩(i,j):=W⟨k−1⟩(i,j);
15:          };
16:  };
17:  Return W⟨n⟩;
```

我们称计算一个 $W^{\langle k\rangle}$的过程为算法第 k 个迭代步.

下面给出该算法的性质及正确性的证明.

定理 1(单调非增性). 算法满足:Base 集单调非增(对应性质 1).

证明:由迭代过程中只有代码行 12 对 Base 集进行修改,且只有集合差运算,因此该集合单调非增. □

针对算法,我们需要引入如下定义:

定义 5. $Dep(k,i,j)$为随算法执行,按如下方式生成的三元组集合:

(1) 对任何$(k,i,j)\in[1,n]\times[0,n-1]^2$,$Dep(k,i,j)$初始为空;

(2) 若代码第 12 行被执行,则$(k-1,i,k-1)$和$(k-1,k-1,j)$被加入到 $Dep(k,i,j)$中;

(3) 对当前 $p\in Dep(k,i,j)$,若 $Dep(p)\neq\varnothing$,则将 $Dep(p)$中的元素加入到 $Dep(k,i,j)$中.

$Dep(k,i,j)$即对应第 k 步约束 $Cond(W^{\langle k\rangle}(i,j))$的支撑集,其引入是为了正确性证明的需要,在算法中并不真正

存储.

引理 1. $\forall(k',i',j')\in Dep(k,i,j)$,一定有 $k'<k$.

证明:对 Dep 的生成方式进行归纳直接可以得到. □

引理 2. $\forall p\in[1,n]\times[0,n-1]^2.\wedge_{(k',i',j')\in Dep(k,i,j)}Cond(W^{\langle k'\rangle}(i',j'))\Rightarrow Cond(W^{\langle k\rangle}(i,j))$.

引理 3. $\forall k_1,k_2\in[1,n]$.若 $k_1<k_2$,则$\forall i,j\in[0,n-1].Cond(W^{\langle k_2\rangle}(i,j))\Rightarrow Cond(W^{\langle k_1\rangle}(i,j))$.

以上几个引理的证明都只需对算法迭代过程进行归纳即可得到.由于篇幅所限,这里不再赘述.

引理 4. 若(i,j)在当前迭代步中被从 Base 中清除,则$\forall(k',i',j')\in Dep(W^{\langle k\rangle}(i,j))$一定有:如$(i',j')\in Base$在本次迭代前为真,则在当前步以及算法的后续迭代步中仍然保持为真.

证明:对 k 值进行归纳.

当 k=1 时,由 k–1=0 及 Dep 集合定义得到,$\forall(k',i',j')\in Dep(W^{\langle k\rangle}(i,j)).i'=0\vee j'=0$),由第 11 行的条件,$(i',j')$在当前步不会被从 Base 中删去.若$(i',j')$在将来某步被从 Base 中删去,由第 11 行的条件,必须存在某个 k'',使得 $k''-1<\min(i',j')$,而 $\min(i',j')=0$ 且由算法迭代中 k 值单调递增得 $k''>1$,故不存在这样的 k'',因此(i',j')不可能在将来的某步被从 Base 中删去,命题成立.

设命题在 $k=m-1$ 时成立,以下证明当 $k=m$ 时也成立.由 Dep 定义得到$(k-1,i,k-1),(k-1,k-1,j)$被加入到 $Dep(k,i,j)$中,再加上 $Dep(k-1,i,k-1)$和 $Dep(k-1,i,k-1)$中的元素即构成了当前的 $Dep(k,i,j)$.由归纳假设得到当前 $Dep(k-1,i,k-1)$和 $Dep(k-1,i,k-1)$中属于 Base 的元素不可能在当前步以及将来某步被从 Base 中删去,只需证明$(k-1,i,k-1)$和$(k-1,k-1,j)$若属于 Base 也不可能在将来某步被从 Base 中删去即可.由第 11 行条件,这两个元素显然不会在当前步被从 Base 中删去,若在将来某步被从 Base 中删去,则再由第 11 行的条件,必须存在某个 k'',使得 $k''-1<\min(i,k-1)$或 $k''-1<\min(k-1,j)$,而由当前 k 已经满足 $k-1<\min(i,j)$以及算法迭代中 k 值单调递增得到,也不存在这样的 k''.故命题成立.

综上所述,定理得证. □

定理 2(基约束表示不变性). 在每一个迭代步,算法满足(*)式条件(对应性质 2).

证明:记 $Base^{\langle m\rangle}$为算法迭代完第 m 步($k=m$)时的 Base 集合,则只需证明,在算法迭代的每一步,下式始终为真:

$$\wedge_{(i,j)\in Base\langle k\rangle}Cond(W^{\langle k\rangle}(i,j))\Leftrightarrow\wedge_{(i,j)\in T^2}Cond(W^{\langle k\rangle}(i,j)).$$

“⇐”由集合的包含关系显然成立,下面证明“⇒”.

当 k=0 时,$W^{\langle 0\rangle}=M$,由 Base 的定义得到命题成立.

设命题当 $k=m-1$ 时成立,以下证明当 $k=m$ 时也成立.下面分情况讨论.若 $Base^{\langle m\rangle}=Base^{\langle m-1\rangle}$,则由引理 3 得命题成立.否则,若当前某$(i_0,j_0)$被从 Base 中删去,由引理 2 得到

$$\wedge_{(k',i',j')\in Dep(m,i_0,j_0)}Cond(W^{\langle k'\rangle}(i',j'))\Rightarrow Cond(W^{\langle m\rangle}(i_0,j_0)).$$

而由引理 1,对$(k',i',j')\in Dep(m,i_0,j_0)$有 $k'<m$,因此 $k'\leq m-1$,再由引理 3 可得

$$\wedge_{(k',i',j')\in Dep(m,i_0,j_0)}Cond(W^{\langle m-1\rangle}(i',j'))\Rightarrow Cond(W^{\langle m\rangle}(i_0,j_0)). \quad (1)$$

令 $Base^{\langle m-1\rangle}$和 $Base^{\langle m\rangle}$分别为第 $m-1$ 和 m 步迭代完成后的 Base 集合,$Diff=Base^{\langle m-1\rangle}-Base^{\langle m\rangle}$,则一定有$(i_0,j_0)\in Diff$.由归纳假设得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle}Cond(W^{\langle m-1\rangle}(i,j))\Leftrightarrow\wedge_{(i,j)\in T^2}Cond(W^{\langle m-1\rangle}(i,j)).$$

两边同时去掉相同的合取项得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle\setminus Diff}Cond(W^{\langle m-1\rangle}(i,j))\Leftrightarrow\wedge_{(i,j)\in T^2\setminus Diff}Cond(W^{\langle m-1\rangle}(i,j)). \quad (2)$$

由引理 4 得到,对所有$(k',i',j')\in\cup_{(i,j)\in Diff}Dep(m,i,j)$一定满足$(i',j')\notin Diff$,故

$$\wedge_{(i,j)\in T^2\setminus Diff}Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in Diff}\wedge_{(k',i',j')\in Dep(m,i,j)}Cond(W^{\langle m-1\rangle}(i',j')). \quad (3)$$

由式(1)和式(3)得到

$$\wedge_{(i,j)\in T^2\setminus Diff}Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in Diff}Cond(W^{\langle m\rangle}(i,j)). \quad (4)$$

再由式(2)和式(4)可以得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle\setminus Diff}Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in Diff}Cond(W^{\langle m\rangle}(i,j)).$$

对右式应用引理 3 得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle\backslash Diff} Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in Diff} Cond(W^{\langle m-1\rangle}(i,j)).$$

在右式加入左式所有的合取项得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle\backslash Diff} Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in Base\langle m-1\rangle} Cond(W^{\langle m-1\rangle}(i,j)).$$

再由归纳假设得到

$$\wedge_{(i,j)\in Base\langle m-1\rangle\backslash Diff} Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in T^2} Cond(W^{\langle m-1\rangle}(i,j)),$$

即

$$\wedge_{(i,j)\in Base\langle m\rangle} Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in T^2} Cond(W^{\langle m-1\rangle}(i,j)).$$

而由算法不改变整个矩阵表示的区域[2],故得

$$\wedge_{(i,j)\in Base\langle m\rangle} Cond(W^{\langle m-1\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in T^2} Cond(W^{\langle m\rangle}(i,j)).$$

再对左式应用引理 3 得到

$$\wedge_{(i,j)\in Base\langle m\rangle} Cond(W^{\langle m\rangle}(i,j))\Rightarrow\wedge_{(i,j)\in T^2} Cond(W^{\langle m\rangle}(i,j)).$$

由此得到命题当 $k=m$ 时也成立.故命题对算法的所有迭代步均成立. □

定理 3(终止性). 算法一定在有限步终止.

证明:由算法的循环控制语句都由有界的 for 语句组成,且循环体不修改循环控制变量值可以直接得到. □

在该处理的前提下,我们将区域的求补运算由原来对合取范式包含的所有约束进行操作改进为只对其中的基约束集进行操作,因此,当合取范式所含约束的集合与基约束集大小相差甚远时,本方法有十分明显的优势.为了形式地刻画本方法适用的范围,我们定义合取式上的一个偏序$\sqsubseteq_c$:

定义 6. 任意给定合取式$\phi=\wedge_{i\in I}\phi_i$和$\Psi=\wedge_{i\in J}\phi_i$,$\phi\Leftrightarrow\Psi$,若$I\subseteq J$,则$\phi\sqsubseteq_c\Psi$.

在该偏序下,本算法可视作求解一个公式的合取范式的极小公式问题.从几何角度来看,合取范式的极小公式长度即为该公式所表示区域的边界个数.如例 1 中表示 D 的范式 $nf(\Phi_D)=x_1>1\wedge x_1\leq 4\wedge x_2\geq 1\wedge x_2\leq 3\wedge x_1-x_2\leq 3\wedge x_1-x_2>-2$,该范式存在一个极小公式$\phi_l=x_1>1\wedge x_1\leq 4\wedge x_2\geq 1\wedge x_2\leq 3$,有$|\phi_l|=4$,即 D 有 4 条边界线(为一个矩形).

因此,若一个公式ϕ的合取范式 $nf(\phi)$存在一个极小公式$\phi_l\sqsubseteq_c nf(\phi)$且公式长度$|\phi_l|<|nf(\phi)|$,则使用本方法能节省存储的空间以及相继操作的时间,且$|\phi_l|$与$|nf(\phi)|$差距越大,本方法在时空上的节省越显著.例如,刚刚提到例 1 中$|\phi_l|=4<6=|nf(\Phi_D)|$,因而求解 D 的补区域时使用本方法(即等价于通过使用ϕ_l而不是$nf(\phi)$进行求反)可以节省时间.同样,对于例 3(如图 3 所示),$D_1\cap D_2$ 实际上等于 D_1,其范式长度为 6,而该范式也存在一个长度为 4 的极小公式(因其表示的区域也为一矩形块),故利用该极小公式取反求解 $D_1\cap D_2$ 的补区域比利用其范式求解所花费的时间要少.

本方法对 CDD 同样适用,只需在 CDD 原范式化过程中对每条路径应用的 DBM 范式化过程进行上述改进即可.需要说明的是,CDD 的单个结点最多可对应于 DBM 相应的两个对称元素,如表示$-2\leq x_1-x_2\leq 3$的 CDD 结点对应于类型为(1,2),(2,1)的 DBM 元素($M(1,2)=(3,\leq)$,$M(2,1)=(2,\leq)$).例如,在例 3(如图 3 所示)中将变量排序为$x=x_1$,$y=x_2$,则相应的 $T=\{0,1,2\}$,对图中惟一一条终端结点为 1 的路径,可应用改进后的范式化处理过程:若令初始的 Base 集为 $T^2\backslash\{(i,i)|i\in T\}$,其结果 Base 将是{(1,0),(0,1),(2,0),(0,2)},即 x-y 对应的约束被排除在结果的基约束集之外,从而在对该区域求补时,只需对 x 和 y 结点进行处理即可.其求补结果对比情况如图 4 所示.其中,C_5 为对原范式 C_4 直接进行求补的结果,C_6 为对 C_3 应用改进后的范式化处理后再根据所得基约束集(不含 x-y 结点对应的约束)求补的结果.容易看到,由于范式化的改进,在随后的求补操作中避免了对导出约束的计算,不但节省了时间,还直接减小了结果数据占用的空间.

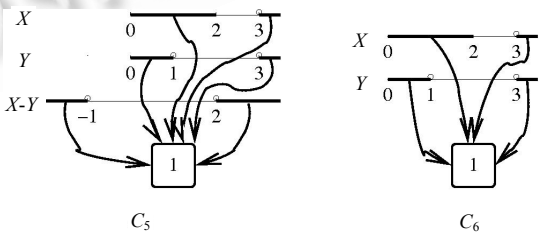

Fig.4 Comparison between the examples of before and after using the new normalization method

图 4 应用新的范式化方法前后的对比实例

3 结 论

本文给出了时间区域数据表示中冗余信息的一个检查与消除的方法,并证明了该方法的正确性.

冗余信息消除是数据存储中的核心问题.各种数据结构一般都存在相应的消冗处理.如 CDD 本身通过结点排序、区间分割(消除区间重叠造成的冗余存储)、等价结点共享等方法来避免冗余信息的存储.另外,CDD 的范式化过程利用对其包含的各个凸区域进行相应的 DBM 范式化处理可检查和消除表示空区域的路径及其结点.但本文所指的这类冗余信息问题并未在 CDD 和 DBM 现有的操作中得到解决.这类冗余信息有些是由原范式化处理生成(如例 1),有些是由其他操作造成(如例 3 中由两 CDD 相交后生成的 CDD(C_3)的 x-y 结点),这些冗余信息会造成如区域求补等操作中不必要的计算.而本文针对此类情况给出的方法则在多数情况起到了很好的效果,尤其是当公式的合取范式的极小公式长度与范式长度相差越大,本方法(与原方法相比)效率越高.值得说明的是,由于本文对原算法的主要改进只是将矩阵各元素按 Base 和非 Base 集分类存储,Base 的修改过程也只是随迭代过程加入了对 k 和 i,j 值的简单比较,因而算法的复杂度仍和原算法相同,即使遇到当极小公式长度与范式长度十分接近甚至相等时的较坏情况也不会对原算法造成较多负面影响.我们对该方法进行了一定的测试,如对例 3 中两个区域求交后的结果,我们分别用改进前后的方法进行一次求补(结果分别为图 4 中的 C_5 和 C_6),得到改进前耗时为 0.001 089 秒,改进后为 0.000 684 秒.而对其进行连续两次求补得改进前耗时为 0.001 725 秒,改进后为 0.001 220 秒.我们又随机抽取了一些区域,其实验结果由表 1 给出,其中列出了算法对给定区域进行求补所消耗的时间及中间计算的 DBM 个数,将范式化方法改进前后的运行情况进行了对比(算法运行的硬件平台为 SUN SPARC Server 1000E,操作系统为 SUN Solaris 2.5).实验数据表明,由于可以避免相继的大量无用计算(相继的操作由于在结果上有依赖关系,故有一定的放大效应),上述方法对算法性能的提高是及其明显的,而且注意到随着时钟变量数的增多,算法消耗资源的增长速度相对缓慢,远远低于改进前的增长速度,这也从实践角度有力地证明了该方法所具有的价值.

Table 1 Experimental results before and after an improvement of normalization

表 1 范式化方法改进前后的实验结果

Time / No. of clocks	Negation once (s)		Negation twice continuous (s)	
	Before improved	After improved	Before improved	After improved
2	0.028	0.010	0.072	0.014
4	39.620	0.048	67.611	0.055
6	167.081	0.124	241.220	0.143
15	483.752	3.236	634.473	4.172
30	1 163.053	8.257	1 542.326	9.347
DBM / No. of clocks	Negation once (No. of DBM)		Negation twice continuous (No. of DBM)	
	Before improved	After improved	Before improved	After improved
2	16	9	46	23
4	35	12	1 044	21
6	51	12	1 060	21
15	164	45	1 683	83
30	371	86	2 456	127

另外,从实验数据也可以看出,算法的开销并不随时钟变量个数的增长而简单地递增,还取决于区域的边界形状.因此,在今后的工作中,我们还将考虑根据边界的不同情况作进一步的优化存储和处理,并根据实际需要对算法作进一步的改进.

致谢 张文辉博士对本文提出了许多宝贵的建设性意见,吴鹏博士帮助绘制了本文的插图,并在讨论中给予了作者许多启发,在此一并表示感谢.最后感谢林惠民老师长期以来对作者的悉心指导和教诲.

References:

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

[2] Dill DL. Timing assumptions and verification of finite-state concurrent systems. In: Sifakis J, ed. Proceedings of the Automatic Verification Methods for Finite State Systems. LNCS 407, Berlin: Springer-Verlag, 197~212.

[3] Behrmann G, Larsen KG, Pearson J, Weise C, Wang Y. Efficient timed reachability analysis using clock difference diagrams. In: Proceedings of the CAV'99. LNCS 1633, Trento, 1999. 341~353.

[4] Asarin E, Bozga M, Kerbrat A, Maler O, Pnueli A, Rasse A. Data-Structures for the verification of timed automata. In: Proceedings of the HART'97. LNCS 1201, Grenoble, 1997. 346~360.

[5] 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. 509~515.

[6] Cerans,K. Decidability of bisimulation equivalences for parallel timer processes. In: Proceedings of the CAV'92. LNCS 663, Montercal, 1992. 302~315.

[7] Weise C, Lenzkes D. Efficient scaling-invariant checking of timed bisimulation. In: Proceedings of the STACS'97. Hansestadt Lubeck, LNCS 1200, 1997. 177~188.

[8] Chen J, Lin HM. Timed bisimulation over timed symbolic transition graph. Chinese Journal of Computers, 2002,25(2):113~121 (in Chinese with English abstract)

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

[10] Henzinger TA, Nicollin X, Sifakis J, Yovine S. Symbolic model checking for real-time systems. Information and Computation, 1994,111(2):193~244.

[11] Wang Y, Pettersson P, Daniels, M. Automatic verification of real-time communicating systems by constraint solving. In: Hogrefe D, Leue S, eds. Proceedings of the 7th International Conference on Formal Description Techniques. Berne, Chapman & Hall, 1994. 223~238.

[12] Backhouse RC, Gasteren AJM. Calculating a path algorithm. In: Proceedings of the Mathematics of Program Construction, the 2nd International Conference (MPC'92). LNCS 669, Oxford, 1992. 32~44.

[13] Sedgewick, R. Algorithms. 2nd ed. Massachusetts: Addison-Wesley, 1988.

附中文参考文献:

[8] 陈靖,林惠民.时间符号迁移图及其互模拟判定.计算机学报,2002,25(2):113~121.

****

第 1 届全国人工智能教育学术研讨会

征 文 通 知

由中国人工智能学会主办，首都师范大学承办，北京航空航天大学、北京工商大学、天津师范大学协办的第一届全国人工智能教育学术研讨会将于 2003 年 12 月 13~15 日在北京召开。这次会议是中国人工智能学会主办的首届全国人工智能教育学术研讨会，也是我国人工智能教育工作者在新世纪的第一次盛会。它一定会对我国人工智能教育事业的发展起到积极的促进作用。

会议主题：信息时代的智能教育

主要内容：

1. 国内外人工智能教育的现状及我国人工智能教育改革的讨论；
2. 不同层次、不同专业的人工智能教学要求、教学计划和教学大纲的讨论；
3. 人工智能课程的教学经验和教材建设经验交流；
4. 智能教学软件和教学系统交流；
5. 关于试办人工智能本科专业的问题的讨论；
6. 其他一些与人工智能教育有关的问题；
7. 人工智能基础、理论、技术和应用研究的进展。

报到地点：北京航空航天大学招待所　　**报到时间**：12 月 13 日全天

时间安排：12 月 14 日上午开幕式、特邀报告,下午主题发言和分组交流
12 月 15 日上午分组交流,下午自由交流或参观

联系方式：

联系地址：100037 北京市西三环北路 105 首都师范大学信息工程学院

联 系 人：谢达　王万森　葛庆平

联系电话：010-68416830；68903443；68901041

E-mail：xieda89@263.net；wangwansen@263.net；wangws@mail.cnu.edu.cn