实时系统的模型检验中针对共享变量的优化技术*)

周修毅　赵建华　李宣东　郑国梁
(南京大学计算机科学与技术系　南京210093)

摘　要　实时系统可以使用由多个并发的时间自动机组成的时间自动机网络来建模。网络中的时间自动机通过共享变量和/或信道交互。带有不同共享变量取值的自动机网络的状态是截然不同的。因此，共享变量也是引起状态空间爆炸问题的原因之一。本文提出了在不同共享变量取值之间的兼容性关系的概念。使用这种兼容性关系，时间自动机网络的可达性分析算法就可以减少需要遍历的状态的个数。本文给出了检测符号化状态中共享变量的取值所能兼容的其它取值的算法以及进一步进行这种兼容性关系检测的增强算法。最后还给出了使用了这两种算法进行优化之后的可达性分析算法。实验结果显示经优化的可达性分析算法的空间效率得到了显著的提高。

关键词　模型检验，时间自动机，形式化方法

Optimization Technique for Shared Variables in Real-time System Model-Checking

ZHOU Xiu-Yi　ZHAO Jian-Hua　LI Xuan-Dong　ZHENG Guo-Liang
(Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　Real-time systems are often modeled as timed automaton networks, which are parallel compositions of timed automata. Those timed automata interact with each other through shared variables and/or communication channels. In the literate, symbolic states with different shared variable valuations in automaton networks are treated as distinct ones. Therefore, shared variables are also one of the causes of state-space explosion. We present the notion of the compatibility relationship between different shared variable valuations in this paper. Using this compatibility relationship can reduce the amount of states that need exploring in the reachability analysis algorithm of timed automaton networks. In this paper, we describe two algorithms: one that detects compact shared variable valuations and the other more powerful one that detects this compatibility relationship further. These two algorithms result in an enhanced reachability analysis algorithm. The experiment results show that our technique improves the space-efficiency of reachability analysis significantly.

Keywords　Model checking, Timed automata, Formal method

1　背景

模型检验是一种验证系统模型是否具有特定属性的形式化技术，其基本思想是穷尽式的状态空间遍历。模型检验算法通过遍历系统的所有状态来确定模型是否具有某个给定的属性。然而，当模型的规模增长时，状态空间爆炸式地增长，以至于模型检验算法不能够在适当的时间内使用适当的内存完成对整个状态空间的搜索。目前有很多技术来解决这个被称为"状态空间爆炸"的问题。只有有效地解决了这个问题，模型检验技术才可以被广泛应用。可达性分析是模型检验的一种，它可以分析一个系统能否到达某个特定的状态。在实际使用中，可达性分析可以被用来分析某个系统是否能够避免到达一个危险的状态，或者分析这个系统是否能够到达一个预设的状态。因此可达性分析是一个很有用的系统验证工具。

时间自动机[1]是一种被广泛应用于实时系统建模的理论工具。时间自动机可看作是加上了有限的时钟变量集合的传统的有限状态自动机。时间自动机的转换对应于实时系统里发生的事件，而时钟变量用来限定事件间的时间间隔。很多实时系统同时也是一个并发系统。人们可使用由并行的时间自动机组成的时间自动机网络来对这样的系统建模。每一个时间自动机对应于实时系统的一个子系统。时间自动机通过共享变量和/或信道交互，其中共享变量具有有限值域。

现有的工具 Kronos[2]、Uppaal[3]等，在状态空间遍历时使用具体的共享变量值作为状态的一部分。带有不同共享变量取值的符号化状态是截然不同的。共享变量取值组合的数目相当大，因此共享变量也是引起时间自动机网络模型检验时状态空间爆炸的原因。本文将给出一个专门针对共享变量的优化方法。该方法可以有效地解决由于共享变量的引入而产生的状态空间爆炸问题。

下面将给出关于时间自动机、时间自动机网络、可达性分析问题的简要介绍。

1.1　时间自动机

首先给出时间区域(time zone)和时间卫式(time guard)的定义。令 C 为时钟变量的有限集合。用 $B(C)$ 来表示一组形如 $x-y\sim n$ 的原子公式的合取的集合，其中 $x,y\in C\cup\{0\}$，$\sim\in\{\leqslant,<\}$，n 为整数。注意，$x\sim n$ 可以表示成 $x-0\leqslant(<)n$ 或 $0-x\leqslant(<)n$。用 D、D_1、D_2 表示 $B(C)$ 中的元素，这些元素称为在 C 上的时间区域。本文中，时间区域也被看作是满足所有原子公式的时钟变量取值集合。令 D_1 和 D_2 为两时间区域，如果 $D_2\Rightarrow D_1$，则称为 D_1 包含 D_2，表示成 $D_2\subseteq D_1$。用 $G(C)$ 来表示如 $x\sim n$ 的原子公式的合取的集合，原子公式形如 $x\sim n$，其中 $x\in C$，$\sim\in\{\leqslant,<,>,\geqslant\}$，$n$ 为整数。用 g、g_1、g_2

*)本项目得到国家自然科学基金(No. 60203009, No. 60273036，国家重点基础研究973计划(No. 2002CB31200001)和省自然科学基金(BK2003408, BK2002079)的资助。周修毅　硕士生，主要研究方向为模型检验；赵建华　副教授，硕导，主要研究方向为形式化方法，软件工程及程序设计语言；李宣东　教授，博导，主要研究方向为面向对象技术、形式化方法；郑国梁　教授，博导，主要研究方向为软件工程、软件开发环境及面向对象技术。

表示 $G(C)$ 中的元素，这些元素称为在 C 上的时间卫式。对任何时钟集合 C，$G(C)\subseteq B(C)$ 成立。

时间自动机 A 定义为五元组 (N,l_0,C,E,I)，其中 N 为有限的位置集合，$l_0\in N$ 是时间自动机的初始位置，C 为有限的时钟集合，$E\subseteq N\times G(C)\times 2^C\times N$ 为转换集合，I 赋予 N 中的每个位置一个 $G(C)$ 中的不变式。所有位置不变式中只包含形如 $x\leqslant(<)n$ 的原子公式。

时间自动机可以看作是传统的有限状态自动机加上一些时钟和时间卫式。时间自动机的一个具体状态由自动机的位置和时钟变量的取值所组成。时钟变量的实数型值随着时间的流逝而增长。只要满足位置不变式，自动机可以一直停留在该位置。令 $\langle l_1,g,r,l_2\rangle$ 为时间自动机的一个转换。只有当时间自动机停留在 l_1，且时钟变量取值满足时间卫式 g 时，转换才可能发生。当转换发生时，自动机的位置跳转到 l_2，在 r 中的所有时钟的值重置为0。

1.2 时间自动机网络

在很多例子中，实时系统同时也是并发的。这样的系统可以使用由有限多个时间自动机组成的时间自动机网络来建模。时间自动机网络中的每个时间自动机对应于一个子系统。网络中的时间自动机通过信道和共享变量来交互。它们之间的交互通过信道行为和与转换相关的共享变量约束来进行。两个不同自动机的转换可通过信道来同步。网络中时间自动机的转换在时间卫式以外也可以有共享变量卫式集合。令 V 为共享变量集合，用 $G_v(V)$ 表示形如 $v\sim n$ 的布尔表达式集合，其中 $v\in V$，$\sim\in\{(,),\leqslant,\geqslant,=,!=\}$，$n$ 为整数。用 $S_v(V)$ 表示形如 $v:=av'+b$ 的赋值式集合，其中 v、v' 是 V 中的共享变量，a 和 b 为整数。时间自动机网络的形式定义如下：

定义1 时间自动机网络为三元组 (C,V,A)，其中

1. $C=\{a_1,a_2,\cdots,a_k\}$ 为同步信道的有限集合。

2. $V=\{v_1,v_2,\cdots,v_m\}$ 为整型共享变量集合。每一共享变量 v_i 有有限的值域 $[L_{vi},U_{vi}]$。

3. A 为带有信道和共享变量的时间自动机集合 $\{A_1,A_2,\cdots,A_n\}$。每一时间自动机 $A_i=(N_i,l_{0i},C_i,E_i,I_i)$，除了 E_i 其余的元素都在1.1节中作了描述。$E_i\subseteq N_i\times G(C)\times 2^C\times M\times 2^{S_v(V)}\times 2^{G_v(V)}\times N_i$，其中 $M=\{!a\mid a\in C\}\cup\{?a\mid a\in C\}\cup\{\perp\}$。除了时间卫式和时钟重置集合，网络中时间自动机的转换可以附带有 M 中的发送/接收消息的行为、在 $G_v(V)$ 中的共享变量卫式集合，及在 $S_v(V)$ 中的共享变量赋值式集合。

网络 (C,V,A) 的运行过程实际上就是 A 中的时间自动机的运行过程。网络的(全局)状态是三元组 $(\bar l,\bar s,v)$，其中 $\bar l$ 是由局部位置组成的全局位置向量，$\bar s$ 代表 V 中共享变量的值，v 代表 $\bigcup_{i=1}^{n}C$ 中时钟的值。给定一个全局状态 $(\bar l,\bar s,v)$，用 $i[i]$ 表示 A_i 的局部位置，用 $\bar s(x)$ 表示在此状态上共享变量 x 的值，在此全局状态上的时钟变量 c 的值表示为 $v(c)$，网络运行如下。

1. 时间流逝：$(\bar l,\bar s,v)\xrightarrow{t}(\bar l,\bar s,v+t)$，如果对于每一个时间自动机 A_i，$v+t$ 满足 $\bar l[i]$ 的位置不变式，其中 $v+t$ 表示满足 $\forall\ x\in\bigcup_{i=1}^{n}C\cdot(v+t)(x)=v(x)+t$ 的时钟取值。

2. 单一转换：令 $e=(l,g,r,\perp,s_v,g_v,l')$ 为 A_i 的转换。$(\bar l,\bar s,,v)\xrightarrow{e}(\overline{l'},\overline{s'},v')$ 如果有

(a) $\bar l[i]=l,\overline{l'}[i]=l'$，且 $\bar l[j]=\overline{l'}[j]$ 当 $j\neq i$；

(b) v 满足时间卫式 g 且 $v'=r(v)$；

(c) $\bar s$ 满足 g_v 中的所有共享变量卫式，且 $\overline{s'}=f(\bar s)$。

这里用 $r(v)$ 表示满足如果 $c\in r$ 则有 $r(v)(c)=0$ 否则的话 $r(v)(c)=v(c)$ 的时钟取值；$f(\bar s)$ 表示满足如果 $x=ay+b$ 在 f 中则有 $f(\bar s)(x)=a*\bar s(y)+b$，否则有 $f(\bar s)(x)=\bar s(x)$ 的共享变量取值。

3. 同步转换：令 $e_1=(l_1,g_1,r_1,?a,f_1,g_{v1},l'_1)$ 为 A_i 的转换，$e_2=(l_2,g_2,r_2,!a,f_2,g_{v2},l'_2)$ 为 A_j 的转换 $(i\neq j)$。$(\bar l,\bar s,v)\xrightarrow{e_1|e_2}(\overline{l'},\overline{s'},v')$ 如果有

(a) $\bar l[i]=l_1,\bar l[j]=l_2,\overline{l'}[i]=l'_1,\overline{l'}[j]=l'_2$，且 $\overline{l'}[k]=\bar l[k]$ $(k\neq i,j)$；

(b) v 满足时间卫式 $g_1\wedge g_2$，且 $v'=r_1(r_2(v))$；

(c) s 满足在 $g_{v1}\cup g_{v2}$ 中所有的共享变量卫式，且 $\overline{s'}=(f_1\cup f_2)(\bar s)$。

共享变量的取值必须在它的值域中。如果单一转换或同步转换赋予了共享变量值域之外的值，系统出现异常停机。在本文剩下的部分里，我们使用 $\bar e$、$\bar e_1$、$\bar e_2$ 等表示时间自动机网络的全局转换。全局转换可以是上述的单一转换或者同步转换。

1.3 在时间自动机网络上的可达性分析

因为时钟变量为实数值，所以时间自动机和时间自动机网络的状态空间是无限的。因此大多数模型检验算法通过枚举符号化状态来遍历状态空间。一个符号化状态可以被看作是由一些具有同样位置的具体状态的一个集合。这个集合中的所有具体状态的时钟变量取值的集合可以使用时间区域来表示。

时间自动机网络 (C,V,A) 的符号化状态为三元组 $(\bar l,\bar s,D)$，其中 $\bar l$ 为全局位置，$\bar s$ 为共享变量取值，D 为在集合 $\bigcup_{i=1}^{n}C_i$ 上的时间区域，C_i 是 A 中 A_i 的时钟集合。如上所述，在时间自动机网络的符号化转换之间有两种转换关系：时间流逝和全局转换。这些转换关系定义如下。

-时间流逝。$(\bar l,\bar s,D)\xrightarrow{\delta}(\bar l,\bar s,D\uparrow\wedge I(\bar l))$，其中 $I(\bar l)=\wedge_{i=1}^{n}I_i(\bar l[i])$；

-全局转换。$(\bar l,\bar s,D)\xrightarrow{\bar e}(\overline{l'},f(\bar s),r(g\wedge D)\wedge I(\overline{l'}))$，这里 $\bar e$ 为全局转换，f、g、r 分别是共享变量赋值式集合、时钟卫式和 $\bar e$ 的重置时钟集合。有关在时间区域上的操作 $D\uparrow$、$r(D)$ 和 $D_1\wedge D_2$ 的细节在文[4]中可以查到。$sp\delta(\bar e)(\bar l,\bar s,D)$ 被称为 $(\bar l,\bar s,D)$ 对于的 $\bar e$ 的直接后继。从直觉上讲，$sp\delta(\bar e)(\bar l,\bar s,D)$ 是所有满足下面的条件的具体状态 s' 的集合：存在 $(\bar l,\bar s,D)$ 中的具体状态 s 和一个实数 t 使得 $s\xrightarrow{\bar e}s''\xrightarrow{t}s'$。

图1中不带下划线的伪代码部分描述的是基本的可达性分析算法。该算法可以检验是否能从符号化状态 $(\bar l_0,\bar s_0,D_0)$ 达到位置 $\bar l_1$。带有下划线的代码段是对基本算法的优化部分，第5节中有其详细的说明。这个算法在运行过程中维护了由它生成的符号化状态的前驱-后继关系。如果算法记录了状态 S 和 S' 之间对于全局转换 $\bar e$ 的前驱-后继关系，那么必然有：

$sp\delta(e)(S')\subseteq S$。

```
PASSED:={};
WAITING:={(l0,s0,D0)};
repeat
    从 WAITING 得到一个符号化状态 S=(l,s,D);
    for 每一条从 l 出发的全局转换 e do
    begin
        if spδ(e)(S)为空 then
            尝试下一条全局转换;
        S'=spδ(e)(S),令 S'=(l',s',D');
        if l'=l1 then
            return YES;
        if 在 PASSED 中有一状态 S''=(l',s',D'') 使得 D'
            ⊆D'',且S'在 s''的被附加集合中 then
            记录在 S 和 S'' 之间的前驱-后继关系;
        else begin
                WAITING = WAITING∪S';
                设置 S' 的附加集合为{s'};
                记录在 S 和 S' 之间的前驱-后继关系;
            end
    end
```

```
把(l,s̄,D)加入 PASSED;
recursive-compute-alpha-set((l,s̄,D));
if 已生成符号化状态的个数首次超过某一阈值 θ
    further-expand-alpha-set();
until WAITING={};
return NO.
```

图1　基本的和经优化的可达性分析算法

2　共享变量取值之间的兼容性

在图1中描述的基本可达性分析算法(不带下划线部分)中,符号化状态使用了具体共享变量取值。当在 PASSED 中有状态$(l,\bar{s},D')$时,如果 $D\subseteq D'$,那么新生成的符号化状态$(l,\bar{s},D)$被忽略。然而,在某些例子中从符号化状态$(l,\bar{s}',D)$可达的所有位置也从符号化状态$(l,\bar{s},D)$可达($\bar{s}'\neq\bar{s}$)。如果这样的情况成立,而且$(l,\bar{s},D)$已经在 PASSED 中,则算法可以忽略状态$(l,\bar{s}',D)$和它的后继。此时 $\bar{s}$ 称为在二元组(l,D)上与$\bar{s}'$兼容。两个不同的共享变量取值的兼容性关系正式定义如下:

定义2　令 l 为时间自动机网络的全局位置,令 D 为该网络时钟集合上的时间区域,令 $\bar{s}$ 和$\bar{s}'$为两个不同的共享变量取值。$\bar{s}$ 称为在元组(l,D)上与$\bar{s}'$兼容,表示成$\bar{s}'\preceq_{(l,D)}\bar{s}$,如果对于每一个从 l 离开的全局转换 $\bar{e}$,如下的条件之一成立。

1. $D\wedge g=\varnothing$,其中 g 是 $\bar{e}$ 的时间卫式;

2. $\bar{s}$ 和$\bar{s}'$都不满足 $\bar{e}$ 的共享变量卫式;

3. $\bar{s}$ 的取值满足 $\bar{e}$ 的共享变量卫式并且以下条件之一成立:

(a)$f(\bar{s}')=f(\bar{s})$,其中 f 是 $\bar{e}$ 的共享变量赋值式集合;

(b)$f(\bar{s}')\preceq_{(\bar{e},D)}f(\bar{s})$,其中 f 是 $\bar{e}$ 的共享变量赋值式集合,且 $sp\delta(\bar{e})(l,\bar{s},D)=(l',f(\bar{s}),D')$。

定义3　令$(l,\bar{s},D)$和$(l,\bar{s}',D')$为时间自动机网络的两个符号化状态。如果有$\bar{s}'\preceq_{(l,D)}\bar{s}$ 且 $D'\subseteq D$,那么,$(l,\bar{s}',D')$称为在变量兼容性上被$(l,\bar{s},D)$包含,表示成$(l,\bar{s}',D')\subseteq_v(l,\bar{s},D)$。

从定义2中直接可引出下面两个引理。

引理1　令 $\bar{s}$ 和$\bar{s}'$为两个共享变量取值,l 为时间自动机网络的全局位置,D 和 D' 为在此网络时钟集合上满足 $D'\subseteq D$ 的两个时间区域。那么有:如果 $\bar{s}\preceq_{(l,D)}\bar{s}'$成立,则有 $\bar{s}\preceq_{(l,D')}\bar{s}'$。

引理2　令 $\bar{s}$,$\bar{s}'$和$\bar{s}''$为三个共享变量取值,l 为时间自动机网络的全局位置,D 为在此网络时钟集合上的时间区域。如果 $\bar{s}\preceq_{(l,D)}\bar{s}'$和$\bar{s}'\preceq_{(l,D)}\bar{s}''$成立,则有 $\bar{s}\preceq_{(l,D)}\bar{s}''$。

从定义2和定义3中可得出以下引理。

引理3　令$(l,\bar{s},D)$和$(l,\bar{s}',D')$为时间自动机网络的两个符号化状态。如果$(l,\bar{s}',D')\subseteq_v(l,\bar{s},D)$,则所有从$(l,\bar{s}',D')$可达的位置也从$(l,\bar{s},D)$可达。

图2描述了兼容的共享变量取值的一个例子。令 $\bar{l}=(l_{11},l_{21})$为一全局位置,$D=(x>3\wedge y<10)$为一时间区域。从全局位置(l_{11},l_{21})出发有三个单一转换 e_{11}、e_{12}和 e_{21}。D 和 e_{12}的时间卫式的相交部分为空。不管共享变量取值是$(v_1=3,v_2=3)$还是$(v_1=2,v_2=3)$,转换 e_{11}设置共享变量为$(v_1=0,v_2=3)$,而 e_{21}设置共享变量为$(v_1=4,v_2=3)$。因此有$(v_1=3,v_2=3)\preceq_{(l,D)}(v_1=2,v_2=3)$。从$((l_{11},l_{21}),(v_1=3,v_2=3),D)$可达的所有位置亦从$((l_{11},l_{21}),(v_1=2,v_2=3),D)$可达。

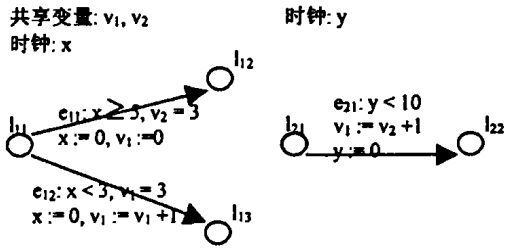

图2　兼容的共享变量取值的例子

3　寻找兼容的共享变量取值的算法

3.1　算法描述

此部分将提出用于寻找与生成的符号化状态的共享变量取值相兼容的变量取值的算法。在状态空间遍历时,新的模型检验算法给每一个生成的符号化状态$(l,\bar{s},D)$添加共享变量取值的集合 α。在进行空间遍历的时候,$\bar{s}$ 总是与 α 中的所有共享变量值在(l,D)兼容。当算法在状态空间遍历时新生成了一个符号化状态$(l,\bar{s},D)$,该状态对应的集合 α 仅包含 $\bar{s}$。然后,该集合 α 将根据$(l,\bar{s},D)$直接或间接后继的信息使用定义2描述的兼容性关系进行扩展。所有满足上述条件的共享变量取值可以加入$(l,\bar{s},D)$的附加集合 α。集合 α 中的值可以由图3描述的函数 compute-alpha-set 计算出来。一旦符号化状态的附加集合发生改变,其前驱状态的附加集合可以用函数 recursive-compute-alpha-set 重新计算。

```
compute-alpha-set(S)
begin
  令 S=(l,s̄,D);
  令 ē1,ē2,…,ēn 为所有离开 l 的全局转换;
  for i=1,2,…,n do
  begin
    if D∧g=∅(g 是 ēi 的时间卫式)then
      αi={s̄'|s̄'是一个共享变量取值};
    else if s̄ 不满足 ēi 的共享变量卫式 gv
      then begin
        找到在 gv 中满足 s̄(x) !~ n 的原子公式 x ~ n;
        令 αi={s̄'|s̄'(x) !~ n};
      end
      else
        αi={s̄|fi(s̄)在 α'i 中},其中 fi是 ēi 的共享变量赋值式且 α'i
        为生成状态 S' 的满足 spδ(ēi)(l,s̄,D)⊆vS' 的附加集合;
  end
  令 α=⋂_{i=1}^{n} αi;
  return α
end
recursive-compute-alpha-set(S)
begin
  令 α' 为添加到 S 的共享变量取值集合;
  令 α=compute-alpha-set(S);
  if α 与 α' 相同 then
    return;
  设置 S 的附加集合为 α;
  for 每一个满足 spδ(ē)(S')⊆vS 的生成状态 S'   do
      recursive-compute-alpha-set(S');
end
```

图3　寻找兼容的共享变量取值的算法

3.2　共享变量取值集合的表示

在本领域中,数据结构二元判定图 Binary Decision Diagrams(BDD)[5]广泛地用于各模型检验工具。然而,该数据结构并不易于表示整型的变量取值集合。这里给出一种特别为共享变量取值集合设计的紧凑数据结构。

令 $v_1,v_2,\cdots,v_n$ 为时间自动机网络的所有共享变量。我们首先说明,图1中带有单下划线的代码段生成的附加集合(图3中的共享变量取值集合 α)为各个共享变量 v_i 值域的子集的正交积。

首先,根据图1中带有单下划线的代码段,对于 WAIT-

ING 中的每一个新生成的符号化状态，添加的共享变量取值集合$\{\bar{s}\}$为正交积$\{\bar{s}(v_1)\}\times\{\bar{s}(v_2)\}\times\cdots\times\{\bar{s}(v_n)\}$。

其次，根据图3，如果调用函数 *compute-alpha-set*(S)来计算$S=(\bar{l},\bar{s},D)$的新添加的取值集合，对于每一条从$\bar{l}$出发的全局转换$\bar{e}_k$，集合α_k亦为正交积，原因如下所述。

1. 如果S的时间区域和$\bar{e}_k$的时间卫式的相交部分为空，α_k为$[L_{v1},U_{v1}]\times[L_{v2},U_{v2}]\times\cdots\times[L_{vn},U_{vn}]$，其中$[L_{vi},U_{vi}]$为$v_i$的值域。

2. 如果S的变量取值不满足$\bar{e}_k$的变量卫式，那么总存在共享变量卫式中的公式$v_i\sim c$使得$\bar{s}(v_i)!\sim c$，其中$\sim\in\{\leqslant,<,>,\geqslant,!=,=\}$。$\alpha_k$为$[L_{v1},U_{v1}]\times\cdots\times[L_{vi-1},U_{vi-1}]\times S_i\times[L_{vi+1},U_{vi+1}]\times\cdots\times[L_{vn},U_{vn}]$，其中$S_i$为$\{x|x\in[L_{vi},U_{vi}]$且$x!\sim c\}$。

3. 如果$\bar{e}_k$的后继非空，那么$\alpha_k=\{\bar{s}|f_k(\bar{s})\in\alpha'\}$，其中$f_k$为$\bar{e}_k$的共享变量赋值式集合，$\alpha'$为后继的附加集合。令集合$\alpha'$为$S'_1\times S'_2\times\cdots\times S'_n$。则$\alpha_k=S_1\times S_2\times\cdots\times S_n$，其中$S_i=S_{i1}\cap S_{i2}$，且$S_{i1}=[L_{vi},U_{vi}]\bigcap_{v_j'=a*v_j+b\in f_k}\{v|a*v+b\in S'_j\}$；$S_{i2}=S'_i$，如果$\bar{e}_k$不赋值给$v_i$；$[L_{vi},U_{vi}]$，如果$\bar{e}_k$赋值给$v_i$。

因此如果所有后继的附加集合是正交积，函数 *compute-alpha-set*(S)的结果也是正交积。而在状态空间遍历的过程中，添加的共享变量取值集合或者是单一元素的集合，或者由函数 *compute-alpha-set*(S)生成。所以所有的附加集合都是正交积，可以表示为$\{\bar{s}|\bar{s}(v_1)\in S_1,\bar{s}(v_2)\in S_2,\cdots,\bar{s}(v_n)\in S_n\}$。

基于上述讨论，我们可以使用非常简单紧凑的数据结构来表示附加在各个已生成的状态上的共享变量取值集合。该数据结构逐个记录了正交积对应的每一个共享变量的取值集合。此外，如果共享变量的值域不大，可以使用32位整型来表示每一个共享变量的值集。因此对于一个具有n个共享变量的时间自动机，每一附加集合需要n个整数的存储空间。

4 进一步扩展兼容性关系的增强算法

在时间自动机网络的状态空间遍历时，经过第3节中的方法优化的算法已经能够用较小的空间代价来提高整体上可达性分析的效率。然而，模型检验对于存储空间的需求有时候是相当庞大的。是否能进一步降低可达性分析时所产生的节点和边的个数，往往能决定是否能在有限的内存、硬盘等条件制约下检验某些具体的工业案例。

下面将介绍进一步扩展兼容性关系的增强算法 further-expand-alpha-set。令$\alpha_1=\{\bar{s}_1\}$表示在寻找兼容的共享变量取值的算法中原有的附加集合，$\alpha_2=\{\bar{s}_2\}$表示新添加到符号化状态上的辅助的共享变量取值集合。该增强算法的思想是在某一特定时刻对已生成的所有状态的α_1进行扩充，使得α_1能够包含更多兼容的共享变量取值。如果一个状态$S=(\bar{l},\bar{s},D)$的附加集合$\{\bar{s}\}$更大，就可能有更多的状态$(\bar{l},\bar{s'},D')$满足$(\bar{l},\bar{s'},D')\subseteq_v(\bar{l},\bar{s},D)$。由于这样的$(\bar{l},\bar{s'},D')$不需要再加以遍历，状态空间就得到了进一步的压缩。*further-expand-alpha-set* 有一定的时间空间开销，是在有限的硬件条件限制之下可供选择的一项技术。该算法的做法是先使得所有已生成状态$(\bar{l},\bar{s},D)$的辅助集合α_2尽可能大，接着逐步地缩小这些辅助集合，直到最后$\bar{s}$兼容所有辅助集合中的共享变量取值。

首先，算法在作初始化时，对于每一个在 PASSED 中的状态$S=(\bar{l},\bar{s},D)$，如果$\bar{s}$满足所有离开$\bar{l}$的转换的共享变量卫式，那么α_2被设置为全集，即$[L_{v1},U_{v1}]\times[L_{v2},U_{v2}]\times\cdots\times[L_{vn},U_{vn}]$，其中$[L_{vi},U_{vi}]$为$v_i$的值域。否则，算法找出所有$\bar{s}$不满足的共享变量卫，并将$\alpha_2$设置为$\{|\bar{s'}|\bar{s'}(x)\ !\sim n,x\sim n$是$g_v$中的原子公式，而$g_v$为原有的$\bar{s}$不满足的共享变量卫式$\}$；对于每一个在 WAITING 中的状态$S$，设置$S$的辅助集合$\alpha_2=\alpha_1$。

然后，算法根据定义2的条件之三逐步缩小所有已生成状态的辅助集合。如果某个辅助集合能被缩小了，其前驱状态的辅助集合也重新计算并缩小。这个过程直到所有的辅助集合不可再缩小为止。当这个过程终止的时候，对于每个已生成状态$(\bar{l},\bar{s},D)$，$\bar{s}$兼容所有在相应的辅助集合中的共享变量取值。

最后，如果某一状态$S=(\bar{l},\bar{s},D)$的辅助集合α_2还是要比原来的附加集合α_1要大，说明该$(\bar{l},\bar{s},D)$的$\bar{s}$可以兼容更多的$\bar{s'}$，则用辅助集合α_2替换原有的共享变量取值集合α_1。

进一步扩展兼容性关系的增强算法 further-expand-alpha-set() 如图4所示：

```
further-expand-alpha-set()
  令InitialList 为所有在 PASSED 和 WAITING 中的符号化状态;
  repeat
    从InitialList 中取出一状态 S=(l̄,s̄,D),它的辅助集合 α2为[Lv1,
      Uv1]×[Lv2,Uv2]×…×[Lvn,Uvn];
    if S 在 PASSED 中
    begin
      令ē1,ē2,…,ēn 为所有离开 l̄ 的全局转换;.
      for i=1,2,…,n do
      begin
        找到在 gv 中满足 s̄(x) !~n 的原子公式 x~n;
        令 α'i=(s̄'|s̄'(x) !~ n};
      end
      令 α'=⋂ⁿᵢ₌₁α'i;
      α2=α2⋂α';
    end
    if S 在 WAITING 中
      α2=α1;
    把 S 添入 BackwardsExpandingList;
  until InitialList ={};
  repeat
      从BackwardsExpandingList 中取出一状态 S=(l̄,s̄,D),该状
        态带有原有的共享变量取值集合 α1和辅助集合 α2;
      for 每一个满足 spδ(ē)(S')⊆vS 的生成状态 S' do
      begin
        α'=(s̄'|f(s̄')在 α 中},其中 f 是 ē 的共享变量赋值式且 α 为
          生成状态 S 的满足 spδ(ē)(l̄',s̄',D')⊆vS 的附加集合;
        α2=α2⋂α';
        if α2比原来的 α2要小 then 将 S' 加 BackwardsExpandingList
          中去;
    end
  until BackwardsExpandingList={};
  令 InitialList 为所有已生成的符号化状态;
  repeat
    if α1小于 α2
      用α2替换 α1,即用增强算法中辅助集合替换原来添加的共享变
        量取值集合;
    until InitialList={};
end
```

图4 进一步扩展兼容性关系的增强算法

5 优化可达性分析算法

5.1 用寻找兼容共享变量取值的算法优化可达性分析

如图1中带单下划线的代码段所示，可达性分析算法优化如下。

1. 对状态空间遍历中生成的每一个符号化状态，附加一共享变量取值集合。在向 WAITING 中加入一新生成的符号化状态$(\bar{l},\bar{s},D)$时，该状态的附加集合初始化为$\{\bar{s}\}$。

2. 当新的符号化状态$(\bar{l},\bar{s},D)$生成时，如果有另一符号化状态$(\bar{l},\bar{s'},D')$满足$D\subseteq D'$且$\bar{s}$在$(\bar{l},\bar{s'},D')$的附加集合中时，$(\bar{l},\bar{s},D)$将不加入 WAITING。

3. 每次当算法生成符号化状态的所有直接后继时，调用

（下转第213页）

fourth ACM Conference on Computer and Communication Security, 1997
6 Drossopoulou S. Towards an abstract model of Java dynamic linking and verification. In: R. Harper, ed. TIC'00-Third Workshop on types in Compilation (Selected Papers), volume 2071 of Lecture Notes in Computer Science, Springer, 2001. 53～84
7 Drossopoulou S, Lagorio G, Eisenbach S. Flexible models for dynamic linking. In Pierpaolo Degano, editor, European Symposium on Programming 2003, volume 2618 of Lecture Notes in Computer Science, Springer, 2003. 38～53
8 Jensen T, Metayer D L, Thorn T. Security and Dynamic Loading in Java: A Formalisation. In: Proc. of the 1998 IEEE Intl. Conf. on Computer Languages, Chicago, inois, May 1998. 4～15
9 Higuchi T, Ohori A. Java bytecode as a typed term calculus. In: proc. of the conf. on Principles and practice of declarative programming, 2002
10 Qian Z, GoldBerg A, Coglio A. A formal specification of Java TM class loading. In: Proc. ACM Conf on Object-Oriented Programming, Systems, Languages, and Applications. ACM Press, 2000
11 Fong P W L, Cameron R D. Proof Linking: Modular Verification of Mobile Programs in the Presence of Lazy, Dynamic Linking. ACM Transactions on Software Engineering and Methodology, 2000, 9(4): 379～409

(上接第196页)

函数 *recursive-compute-alpha-set*($(\vec{l},\vec{s},D)$)来计算该符号化状态和其前驱的所有附加集合。

经过这样的优化后得到的可达性分析算法所维护的前驱-后继关系与由基本算法生成的关系是不同的。如果在两个状态 S 和 S' 之间有关于全局转换的前驱和后继关系，则这两者的关系是 $sp\delta(\vec{e})(S)\subseteq_v S'$，而不是 $sp\delta(\vec{e})(S)\subseteq S'$。由附加集合的计算方法可知，对于任何符号化状态 $(\vec{l},\vec{s},D)$，$\vec{s}$ 在元组 $(\vec{l},D)$ 上与附加集合中所有共享变量取值相兼容。

5.2 用增强算法进一步优化可达性分析

在图1中所示的带有双下划线的程序代码描述的是使用增强算法来进行优化的部分。如果在状态空间遍历的某一时刻已生成的所有状态个数超过阈值 θ 时，就可以尝试进一步扩充这些状态的附加集合使其能兼容更多的共享变量取值。

6 案例研究

以上寻找兼容的共享变量取值的算法及增强算法已用 C++语言实现并且在带有512M 主存 IBM thinkpad 笔记本上检验了几个工业案例。这些案例包括 Fischer 互斥协议(Fischer's mutual exclusion protocol)，带界限重新传输协议(the Bounded Retransmission Protocol[6])和 Bang&Olufsion 音频协议(the Bang&Olufsion audio protocol[7])。本文中的优化技术在检验 Fischer 互斥协议时并不产生效率的提高。但当算法用来检验另外两个例子时会节省很多存储空间，在实验结果上表现为可达性图中结点和边的个数的减少。

Bang&Olufsion 音频协议用来在单一总线上的音频/视频组件之间传输数据。协议能在必要时检测数据冲突并重新传输数据。当检验 Bang&Olufsion 音频协议时，经过初步优化的可达性分析算法(不含增强算法)生成了包括 36555 个节点和 40769 条边的可达性图。未经优化的算法产生包括 116064 个节点和 126599 条边的可达性图。

表1 带界限重新传输协议的结果

参数	未经优化的节点个数	未经优化的边的个数	优化后的节点个数	优化后的边的个数
MAX=3 N=3	709	917	539	732
MAX=3 N=2	519	681	410	551
MAX=2 N=3	570	738	444	601

带界限重新传输协议(the Bounded Retransmission Protocol)通过不稳定的信道传输文件。这个协议有两个参数：MAX 和 N。发送者最多可以重复传送数据 MAX 次；在一个传送序列中最多有 N 个数据段。表1显示了用不同参数检验该协议得到的结果数据(检验时不含增强算法)。而同时包含寻找兼容的共享变量取值的算法及增强算法的可达性分析在 $MAX=3$、$N=3$ 且 $\theta=200$ 时产生了包括 325 个节点和 448 条边的可达性图。也就是说，增强算法可以进一步优化可达性分析算法的空间效率。

结论 时间自动机网络在模型检验中可用于并行实时系统的建模。网络中的时间自动机通过同步信道或共享变量交互。如果两个状态的共享变量取值不同，那么这两个状态也不同。因此共享变量也是状态空间爆炸的原因之一。本文定义了在共享变量取值之间的兼容性关系，并提出了检测符号化状态中共享变量取值所能兼容的取值的算法以及进一步进行这种兼容性关系检测的增强算法。算法找到的共享变量取值以一种紧凑的数据结构来记录。案例研究表明这两种优化技术用在可达性分析上时能够显著地降低可达性分析对于存储空间的需求，因而这两者是其它处理时间假设和约束的优化技术的一个有用的补充。

参考文献

1 Alur R, Dill D L. A theory of timed automata. In Theoretical Computer Science, The preliminary version appears in Proc. 17th ICALP, 1990, LNCS 443, 1994, 126: 183～235
2 Daws C, Olivero A, Tripakis S, Yovine S. The tool Kronos. In DIMACS Workshop on Verification and Control of Hybrid Systems, LNCS 1066. Springer-Verlag, Oct. 1995
3 Behrmann G, David A, Larsen K G, et al. Uppaal - Present and Future. In: Proc. of the 40th IEEE Conf. on Decision and Control (CDC'2001), Orlando, Florida, USA, IEEE Computer Society Press, Dec. 2001
4 Bengtsson J, Wang Yi. Timed Automata: Semantics, Algorithms and Tools. In Lecture Notes on Concurrency and Petri Nets, Edited by W. Reisig and G. Rozenberg, LNCS 3098, Springer-Verlag, 2004
5 Burch J R, Clarke E M, McMillan K L. Symbolic model checking: 1020 states and beyond. In Information and Computation, 1992, 98: 142～170
6 D'Argenio P R, Katoen J-P, Ruys T, Tretmans J. Modeling and Verifying a Bounded Retransmission Protocol. In: Proc. of COST 247, Intl. Workshop on Applied Formal Methods in System Design, Maribor, Slovenia, June, 1996. Also appeared as Technical Report CTIT 96-22, University of Twente, July 1996
7 Havelund K, Skou A, Larsen K G, Lund K. Formal Modeling and Analysis of an Audio/Video Protocol: An Industrial Case Study Using Uppaal. In: Proc. of the 18th IEEE Real-time Systems Symposium, San Francisco, California, USA, Dec. 1997. 2～13