一种动态消减时间自动机可达性搜索空间的方法*)

陈铭松　赵建华　李宣东　郑国梁

(南京大学计算机软件新技术国家重点实验室,南京大学计算机科学与技术系　南京 210093)

摘　要　时间自动机的可达性分析算法通常采用对符号状态的枚举来遍历其状态空间。符号状态由位置与时间区域组成,时间区域用形如 $x-y \leqslant (<)n$ 的原子公式的合取式来表示。在对时间自动机进行可达性分析的过程中,分析算法将生成大量的符号状态,往往导致对计算机内存的需求超出了可行的范围。本文给出了一个消减符号状态个数的方法。该方法通过对符号状态间的依赖关系进行分析,在不影响分析结果的前提下消去某些时间区域的原子公式,从而扩展符号状态。扩展后的符号状态包含有更加多的其它的状态,通过删除掉那些被包含的符号状态可以减少算法存储的状态个数,节省存储空间。本文最后给出了相关的案例分析,结果表明这个算法有效地减少了某些时间自动机可达性分析过程中所需的存储空间。

关键词　时间自动机,模型检验,符号状态,时间区域

An Algorithm to Dynamically Reduce the State Space of Timed Automata during the Reachability Analysis

CHEN Ming-Song　ZHAO Jian-Hua　LI Xuan-Dong　ZHENG Guo-Liang

(National Laboratory of Novel Software Technology, Department of Computer Science and Technology, Nanjing University, Nanjing 210093)

Abstract　The reachability analysis algorithm explores the state space of a timed automaton by enumeration of symbolic states. Each symbolic state consists of a location and a time zone which are conjunctions of automatic formulae in the form $x-y \leqslant (<)n$. Sometimes the amount of generated symbolic states is very large, the memory required to store the generated symbolic states is not feasible. In this paper, we present an approach to reduce the memory requirement of the reachability analysis algorithm. By analyzing the dependence relation between symbolic states, we can expand some of the symbolic states by removing specific kinds of atomic formulae without changing the reachability analysis result. The expanded states can contain more symbolic states. Removing these contained states can reduce the memory requirement of reachability analysis. The case studies presented in this paper show that our algorithm can save memory in the practical application efficiently.

Keywords　Timed automata, Model checking, Symbolic state, Time zone

1　引言

模型检验(model checking)[1]是一种被用来自动验证有穷状态系统的形式化技术。模型检验的基本思想是穷尽搜索系统的状态空间。当系统很复杂、规模较大的时候,系统状态的组合往往会导致"状态空间爆炸"。时间自动机(timed automaton)[2]在传统的自动机上引入了时间的概念,主要被用来对实时系统等关于时间的系统建模。因为时间自动机引入了时钟变量的概念,使得系统状态涉及到时钟变量的取值,从而大幅度增加了系统的状态数量,所以在检验过程出现的"状态空间爆炸"现象更加严重。要想获得好的模型检验效率,我们就必须采用各种方法约减状态空间。

时间自动机的具体状态(concrete state)中包含有时钟变量的取值,而时钟变量的定义域是非负实数 R^+,所以它的状态是无穷的。因此直接针对具体状态进行模型检验是不可行的。我们可以通过枚举时间自动机的符号状态将无穷化为有穷。一个具体状态由两个部分组成:时间自动机的位置与时钟变量取值。一个符号状态也由两个部分组成:时间自动机的位置与时间区域。时间区域是一组有关时钟变量的原子公式的合取。符号状态可以被看作是具有相同位置且时钟变量取值满足这些原子公式的所有具体状态的集合。如果消除一个符号状态中时间区域的某些原子公式,那么其对应的具体状态集合将会增大。在这样的情况下,我们称这个符号状态被扩展了。

通常时间自动机的可达性分析是基于对符号状态的枚举实现的。它的基本思想是首先从一个起始状态开始,可达性分析算法不停地产生已遍历状态的后继状态,直至到达预设的目标位置或者不能够再生成新的符号状态为止。目前时间自动机的可达性分析已经有了许多优化方法。其中有很多优化方法是通过扩展符号状态来实现优化的。例如,不活跃时钟约减技术[3]通过约减所有与不活跃时钟相关的原子公式扩展符号状态;文[4]给出在时间自动机只测试时钟 x_0 上界(或下界)的情况下,模型检验算法能够移除形如 $x_0-y \leqslant (<)n$ (或 $y-x_0 \leqslant (<)n$)的原子公式扩展符号状态。在文[5]中

*)本课题研究得到国家自然科学基金(No. 60573085)和国家重点基础研究 973 计划(No. 2002CB312001)的资助。陈铭松　硕士研究生,主要研究方向为模型检验、软件测试;赵建华　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。

提出了一个优化算法，该算法通过时钟的精确上界和下界信息来扩展生成的符号状态。以上这几种算法是在"状态空间搜索"之前利用静态分析获得的信息来扩展符号状态，约减状态空间。

本文针对时间自动机的可达性分析给出了一个基于动态扩展符号状态的优化方法。它在空间遍历的时候根据已生成符号状态的前驱后继的依赖关系删除一些无关的原子公式，以此来扩展可达性分析算法过程中的符号状态，减少可达性分析算法所使用的存储空间。

本文第2节分别介绍了时间自动机、符号状态，可达性分析算法，以及符号状态间的依赖关系等基本概念。第3节给出了针对我们算法的分析以及相关的定义。第4节给出了动态扩展状态空间的基本思想与相关实例，最后给出优化算法的完整描述。第5节给出了我们设计的检验工具对相关案例的分析以及实验数据结果。最后总结全文。

2 相关背景

在这一节，我们给出时间自动机及其符号状态的形式化定义以及依赖关系与可达性分析算法的描述。

2.1 时间自动机

我们用 $G(C)$ 表示基于时钟变量集合 C 上的时间卫式的集合。时间卫式是形如 $x \sim n$（$x \in C, n \in N, \sim \in \{\leqslant, <, >, \geqslant\}$）的原子公式的合取。一个时间自动机被定义为五元组 (N, l^0, C, E, I)：

(1) N 是一个有限的位置集合；

(2) $l^0 \in N$ 是起始位置，C 是有限的时钟变量集合；

(3) $E \subseteq N \times G(C) \times 2^C \times N$ 是一个转换的集合；

(4) I 给 N 中的每一个位置 l 分配了一个位置不变式 $I(l)$，$I(l) \in G(C)$，位置不变式中的所有原子公式都是形如 $x \leqslant (<) n$，n 为自然数。

一个时间自动机可以被看作是一个有穷自动机添加了时钟变量与时间卫式。假设 (l_1, g, r, l_2) 为时间自动机上的一个转换，当自动机的当前位置为 l_1，且时钟取值满足时间卫式 g，转化就可能发生。转换发生后，时间自动机的当前位置变为 l_2，集合 r 中的时钟变量的值被重置为0，而其它时钟变量的值保持不变。

时间自动机的具体状态可以用二元组 (l, v) 表示，l 代表时间自动机的一个位置，时钟变量取值 $v: C \to R^+$ 是一个映射函数。给定一个时钟变量 $x \in C$，$v(x) \in R^+$ 表示在状态 (l, v) 上 x 的值。随着时间的流逝，所有的时钟变量值都会以相同的速度增加。时间自动机的操作语义定义了由当前位置和当前时钟值所组成的状态之间的转换。在状态之间的转换可以归纳为两类。自动机可以延迟一定时间（时间流逝，也称作延迟转换），或者沿一条时钟约束得到满足的边进行状态跃迁（动作转换）。

定义1 具体状态转化关系的操作语义如下：

(1) 时间流逝：$d \in R^+$，如果 $v + d$ 满足 l 的位置不变式，则 $(l, v) \xrightarrow{d} (l, v + d)$，其中时钟变量取值 $v + d$ 表示对时钟变量 $x \in C$，$(v + d)(x) = v(x) + d$；

(2) 状态跃迁：如果 $e = (l_1, g, r, l_2)$，且 v 满足 g 中的所有原子公式，则 $(l_1, v) \xrightarrow{e} (l_2, v')$，其中 v' 表示满足下列条件的时钟变量取值：对于每一个时钟变量 c，如果 $c \in r$，则 $v'(c) = 0$，否则 $v'(c) = v(c)$。

2.2 符号状态

因为时钟的定义域为非负实数，时间自动机的状态空间是无穷的，而模型检验的对象是有穷的系统，所以研究人员一般采用符号状态将无穷的状态空间转化为有穷的。

设 C 为时钟集合，我们用 $B(C)$ 表示时钟集合 C 上的时间区域（time zone）的集合。每个时间区域是一组关于时钟变量的原子公式的合取式。时间区域的原子公式与时间卫式的原子公式不同，其原子公式都是形如 $x - y \sim n$（$x, y \in C \cup \{0\}$，$\sim \in \{\leqslant, <\}$，$n \in Z$）的针对时钟变量的约束。由于 $x \sim n (n \in N)$ 可以表示为 $x - 0 < (\leqslant) n$，或者 $0 - x < (\leqslant) - n$，因此对于任意的时钟集合 C，$G(C) \subseteq B(C)$。

设 D_1 与 D_2 是两个时间区域。如果 $D_2 \Rightarrow D_1$，则称 D_1 包含 D_2，记为 $D_2 \subseteq D_1$。如果 $D_2 \subseteq D_1$ 且 $D_1 \subseteq D_2$，则时间区域 D_1 与 D_2 等价。令 $d_1 (x - y \sim_1 c_1)$、$d_2 (y - z \sim_2 c_2)$ 为两个原子公式，这两个原子公式之间的连接 $x - z \sim_3 c_1 + c_2$ 记为 $d_1 \cdot d_2$，其中，如果 $\sim_1$ 和 $\sim_2$ 均为 $\leqslant$，则 $\sim_3$ 为 $\leqslant$，否则 $\sim_3$ 为 $<$。一个时间区域被称为是正则的，当且仅当对于任意两个它的原子公式 d_1、d_2，它必然有一个不弱于 $d_1 \cdot d_2$ 的原子公式。

时间自动机中的符号状态定义为二元组 (l, D)，其中 l 是符号状态的位置信息，D 是符号状态的时间区域，$D \in B(C)$。(l, D) 可以看作是相同位置满足某种时间约束的状态的集合 $\{(l, v) \mid v$ 满足 D 中的所有原子公式$\}$。符号状态的后继 $SP_\delta (e, (l, D))$ 表示了具体状态的集合 $\{(l', v') \mid \exists e, d, (l, v)$ 使得 v 满足 $D \wedge (l, v) \xrightarrow{e} (l', v'') \xrightarrow{d} (l', v')\}$。对于转换 $e = (l, g, r, l')$，$SP_\delta (e, (l, D))$ 可由 $(l', (r(D \wedge g))^\uparrow \wedge I(l'))$ 计算得到，$I(l')$ 是位置 l' 的位置不变式，$r(D)$、$\uparrow$、$\wedge$ 都是针对于时间区域操作的运算符。对于时钟变量 $c \in r$，$r(D)$ 将 D 中的时钟变量 c 重置为零，并且取消 D 中所有与 c 相关的时间约束。$\uparrow$ 操作又称 up 操作，它得到的是 D 时间流逝后的最强约束的时间区域。$\wedge$ 是时间区域的与操作。有关这些运算符操作的具体细节请参考文[6]。

除初始符号状态外，其余符号状态的时间区域是由其前驱以及转换上的时间卫式通过运算符 $r(x)$、$\uparrow$、$\wedge$ 以及正则化等运算产生的。对于每个运算操作，运算结果中的所有的原子公式都可以表示为参加运算的时间区域中的单个或者多个原子公式的连接。具体来说：

(1) $r(D)$ 中的每个原子公式 d 要么是 D 中的原子公式，要么形如 $x \sim 0 (\sim \in \{=, \leqslant\})$。

(2) $D^\uparrow$ 中的每个原子公式 d 都是 D 中的原子公式。

(3) $D_1 \wedge D_2$ 中的每个原子公式要么是 D_1 中的原子公式，要么是 D_2 中的原子公式。

(4) 对于任意给定的时间区域 D，假设 D' 是 D 的正则形式，那么 D' 中的每个原子公式要么也是 D 中的原子公式，要么可以表示为 D 中的两个或者多个原子公式的连接。

2.3 时间自动机可达性分析的基本算法

可达性分析是模型检验研究的一个主要方向，它可以判定一个模型从一个起始状态能否到达某个给定的目标位置。时间自动机的可达性分析一般采用符号状态枚举的办法。分析算法从初始状态开始，不停地使用算子 SP_δ 来计算已经生成的状态的后继。这个过程一直到下面的两个条件之一成立才会终止：(1) 已经不能生成新的符号状态；(2) 算法生成了一个目标位置上的状态。

```
PASSED={};
WAITING={(l0,D0)};
```

```
repeat
从 WAITING 中取一个符号状态(l,D),取名为 S,WAITING=
WAITING-{S};
对于每条离开 l 的变换 e,
begin
  if SPδ(e,S)为空,then 尝试下一个变换;
  计算 S'=SPδ(e,S),令 S'=(l',D');
  if l' 是指定的目标位置 l1,then 返回 成功;
  if 有一个符号状态 S''=(l',D'')在 WAITING∪PASSED 中且 D'⊆
    D'',
  then 记录 S 到 S'' 的前趋后继关系;
   else WAITING=WAITING∪{S'},记录 S 到 S' 的前趋后继关
        系;
end
添加 S 到 PASSED;
until WAITING={}
返回 失败。
```

图 1　可达性分析的基本算法

图 1 中给出了基本的时间自动机可达性分析的算法。它的思想被广泛应用于不同的模型检验工具,具体请参见文[7~9]。算法给定了起始符号状态(l_0,D_0),以及要求判定是否能够到达的位置 l_1。在状态空间的遍历过程中,所有的符号状态被分为两组 WATING 和 PASSED。WATING 中保存了所有需要计算后继的符号状态;而 PASSED 中存放了所有后继已经被计算生成的符号状态。在分析开始时,PASSED 为空而 WAITING 中只有起始符号状态。算法在执行过程中记录了相邻符号状态间的前驱后继关系,利于对前驱符号状态的回溯。

在符号状态遍历的过程中,可达性分析算法生成了一个有向可达图,用来表示可达性分析当前的已遍历过的空间。有向可达图的节点是已经生成的符号状态,而图的每条边上标记了相邻符号状态之间的转换。由图 1 易见如果在有向可达图中有从 S 到 Se 的标记为 e 的边$(S\xrightarrow{e}Se)$,其中 $S=(l,D)$,$Se=(l',D')$,那么 $SP\delta(e,(l,D))\subseteq Se$ 必然成立。我们称 Se 为 S 在该图中的关于转换 e 的后继,S 为 Se 在该图中的关于转换 e 的前驱。

2.4　符号状态间的依赖关系

可达性分析算法的基本思想就是从起始符号状态节点开始,不断生成在 WAITING 中的符号状态的后继,形成相应的有向可达图,并判断该图中是否存在指定的位置。因为符号状态(l,D)关于 e 的后继 $SP\delta(e,(l,D))=(l',D')$的时间区域通过计算$(r(D\wedge g))^{\uparrow}\wedge I(l'))$获得,所以 $SP\delta(e,(l,D))$的每个原子公式都是由 D,g,以及集合$\{x\sim 0\mid x\in r,\sim\in\{=,\leqslant\}\}$中的原子公式通过连接得到的。因此说有向可达图符号状态节点间的原子公式间存在着某种依赖关系。下面给出这种依赖关系的定义。

定义 2　如果不同符号状态时间区域中的原子公式之间符合以下 3 种情况之一,则称这些原子公式之间存在着依赖关系:

(1)假设符号状态(l,D)关于变换 e 的后继 $SP\delta(e,(l,D))$为(l',D'),D'中的某个原子公式 d'依赖于 D 中的某个原子公式 d 当且仅当 d'是由包含 d 在内的 D 中的原子公式,g 中的原子公式以及集合$\{x\sim 0\mid x\in r,\sim\in\{<,\leqslant\}\}$中的原子公式连接得到的。

(2)假设有向可达图中有两个符号状态 S 和 Se,且从 S 到 Se 有一条标记为 e 的边。Se 中的一个原子公式 $x-y\sim c$(其中 x,y 为时钟或者常数 0,$\sim\in\{<,\leqslant\}$)依赖于 S 的原子公式 d 当且仅当存在 $SP\delta(e,(l,D))$中的一个原子公式 $x-y\sim c'$使得 $c'\leqslant c$,且 $x-y\sim c'$依赖于 d 。

(3)假设 S 和 S'为有向可达图中的两个符号状态,S'中的一个原子公式 d'依赖于 S 中的原子公式 d 当且仅当存在一个可达图的路径 $S\to e_1\to S_1\to e_2\to S_2\to\cdots\to S_n(=S')$,并对所有的 $i(1\leqslant i\leqslant n-1)$,存在 S_i 的原子公式 d_i,使得 d'依赖于 d_{n-1},d_i 依赖于 $d_{i-1}(2\leqslant i\leqslant n-1)$且 d_1 依赖于 d 。

3　关于扩充符号状态的基本想法

由图 1 的算法,不难看出如果能将 PASSED 中符号状态进行扩展,那么将会有一些新生成的符号状态不会被添加到 WAITING 中,从而减少了可达性分析的时空开销。文[4]指出时间自动机在状态空间遍历过程中,符号状态中有一些特定的原子公式可以被删除,即扩展相应的符号状态,而可达性分析的结果却不会改变。然而上面的算法对原子公式的删除或减弱是基于对时间自动机的静态分析。在这一节,我们将给出一个在可达性分析算法执行期间通过分析原子公式之间的依赖关系来动态地扩展符号状态的优化技术。

假设在图 1 中的算法的运行过程中的某个时刻,集合 PASSED 中包含一符号状态(l,D)。那么,后面所有由算法生成的被(l,D)包含的符号状态都将被被弃用。所以如果能够进一步扩展 PASSED 中的符号状态,那么更多由算法生成的符号状态会被弃用。同时,算法也不会去生成这些被弃用的符号状态的后继,这样就有效地控制了状态空间。

当然,算法必须在保证分析结果不变的情况下对 PASSED 中的符号状态进行扩张。现在考虑动态扩展 PASSED 中的符号状态时会出现的问题。因为图 1 中的算法只对 WAITING 中的符号状态求后继,所以替换 PASSED 中的符号状态不会使得算法生成不能到达的符号状态。然而如果任意替换 PASSED 中的(l,D)为(l,D'),其中 $D\subseteq D'$,或许会使得算法遗漏一些可以到达的位置。例如,假设存在三个时间区域 D、D'、D''满足 $D\subseteq D'$,$D''\subseteq D'$,但是 $D''\not\subseteq D$;并且在可达性分析执行期间,我们强制地将 PASSED 中的状态(l,D)替换为(l,D');而此后,算法在 WAITING 中的某个符号状态又新生成了后继符号状态(l,D'')。这时因为(l,D')在 PASSED 中,所以(l,D'')将会被弃用。假设(l,D'')是状态空间中唯一能够到达目标位置的符号状态,那么弃用(l,D'')就会使得图 1 算法返回 FALSE。而实际上这个目标位置是可以达到的,也就是说这样的没有约束的扩展会使得分析算法把实际可达的位置当作不可达的。

定义 3　如果有向可达图中的两个符号状态 S 与 S'之间存在一个转换序列 $e_1,e_2,e_3,\cdots,e_{n-1},e_n$ 使得 $S\to e_1\to S_1\to e_2\to S_2\to\cdots e_n\to S_n=S'(n\geqslant 1)$,那么我们称这个序列为从符号状态 S 到 S'的一条路径。令路径 $P=e_1,e_2,e_3,\cdots,e_{n-1},e_n(n\geqslant 1)$,我们扩展 $SP\delta$ 操作符的语义,使得 $SP\delta(P,(l,D))=SP\delta(e_n,\cdots,SP\delta(e_2,SP\delta(e_1,(l,D))))$。

在本文中,我们将给出一种通过删除符号状态中的某些原子公式来扩展已经生成的符号状态的优化方法。显然,在删除这些原子公式的时候需要考虑避免出现上面的情况。假设在可达性分析的过程中,(l,D)是 PASSED 中的一个符号状态,并假设 D'是通过删除 D 中的某些原子公式得到的时间区域,易见 $D\subseteq D'$。那么当下面的条件 1 成立时,用(l,D')来替代(l,D)并不会改变可达性分析的结果。

对于所有离开 l 的路径 p,如果 $SP\delta(p,(l,D))$为空,那么 $SP\delta(p,(l,D'))$必然也是空。　…… 条件 1

这个条件说明,如果算法用(l,D')来替换(l,D),那么由$(l,$

D')生成的所有后继节点保持(l, D)生成的所有后继节点的可达性的拓扑结构。在前面产生错误结果的例子里,我们可以看到如果(l,D')满足条件1,那么它肯定不可能包含(l, D''),因为(l,D'')的后继里有(l, D)到达不了的状态,与条件1矛盾。下面的定理最早在文[4]中给出。

定理1 如果符号状态(l ,D)关于变换 $e=(l,g,r,l')$ 的后继为空,那么必然存在 D 的某个原子公式 A 使得 A 和 g 相交为空。我们称 A 为关键原子公式。

删除 D 中某个关键原子公式可能会引起使得 $SP\delta(e,(l,D))$ 非空,可能使得得到的新状态违反上面的条件1。考虑到原子公式的依赖关系,如果 d 是一个有向可达图中的一个符号状态的原子公式,且某个关键公式依赖于 d,那么删除 d 后得到的新的符号状态也可能违反条件1。反之,如果我们保留所有这些被关键原子公式依赖的公式,那么在计算后继的时候,这些关键原子公式都会被保留。因此,只要不删除这些公式,条件1就不会被违反。我们把这一类原子公式,包括关键原子公式以及它们所依赖的其它原子公式都称为相关原子公式。直观地讲,这些公式是否被删除和最后的可达性分析的结果是相关的。当有向可达图全部生成完成之后,所有不是相关原子公式的原子公式都是可以被删除的。但是在模型检验过程中,我们只能够依据已经生成的部分有向可达图来判断一个原子公式是否相关。如果可达性分析算法还没有为某些状态计算出所有的后继,那么这些状态中的所有的原子公式是否会成为不相关的公式是无法确定的。我们把这些公式,以及它们所依赖的原子公式称为未知类型的原子公式。如果一个原子公式既不是相关公式,也不是未知类型的公式,那么删除这个公式是没有问题的,因此我们称它们为无关原子公式。这三类原子公式的正式定义如下:

定义4 在生成某个符号状态的时候,每个原子公式可以为三种类型之一:相关(relevant),不相关(irrelevant),未知(unknown)。

(1)相关(relevant):

a)如果某个符号状态(l, D)关于变换 e 的后继为空,且 A 是关键原子公式,那么 A 是相关原子公式。

b)如果某个符号状态(l', D')的原子公式 A 是相关的,那么在该符号状态的直接前驱(l, D)中,参与连接生成 A 的原子公式也是相关的。

c)只有以上定义的原子公式才是相关的。

(2)未知(unknown):

a)对于一个符号状态(l , D),e_1,e_2,…,e_n 是所有离开位置 l 的边。如果它的后继还没有全部生成,那么 D 中不是相关原子公式就是未知的原子公式。

b)如果一个符号状态(l', D')的原子公式 A 是未知的,那么该符号状态的直接前驱(l, D)中,参与连接生成 A 的原子公式如果不是相关的,那么必然是未知的。

(3)不相关(irrelevant):如果一个原子公式如果既不是相关的,也不是未知的,那么必然是无关的。

根据这个定义,图2给出了标记相关原子公式与未知原子公式操作的伪码。操作 TagDepAF(atomicF, type)的含义是递归标记原子公式 atomicF 所在符号状态的前驱状态中与 atomicF 有依赖关系的原子公式为 type 类型。TagUnknownAF(WATING)的含义是递归标记与集合 WATING 中原子公式有依赖关系的原子公式为 UNKNOWN 类型。TagReleventAF(af)的含义是递归标记与原子公式 af 有依赖关系的原子公式为 RELEVENT 类型,所以 TagReleventAF(af)= TagDepAF(af, RELEVENT)。RemoveIrrelevantAF()的含义是删除无关原子公式并且重置有向可达图及其符号状态的原子公式的类型,其具体操作为:删除没有被标记为 RELEVENT 或 UNKNOWN 类型(即 NULL)的原子公式;重新计算有向可达图的前趋后继关系;将所有符号状态中的未知原子公式都重置为 NULL 类型。

```
TagDepAF(atomicF,type)
Begin
  标记 atomicF 为 type 类型;
  for atomicF 所在符号状态的每个直接前驱状态 PreST
  begin
    if PreST 中存在原子公式 PreAF 与 atomicF 有依赖关系
    then TagDepAF(PreAF,type);
    else  return;
  end
end

TagUnknownAF(WATING)
begin
  for WATING 集合中的每个符号状态 S
  begin
    for 符号状态 S 中的每个原子公式 af
    begin
      TagDepAF(af,UNKNOWN);
    end
  end
end
```

图2 标记原子公式类型的操作

4 在可达性分析过程中动态扩展符号状态

4.1 动态扩展符号状态的基本思想

上文给出了3种原子公式类型的定义,以及递归标记这些原子公式类型的相关操作。动态扩展符号状态的过程就是在可达性分析中某个恰当的时间利用这些操作对有向可达图的所有原子公式进行标记,删除无关原子公式的过程。

动态扩展符号状态基本思想如下:首先将起始符号状态 initState 时间区域的原子公式都标记为空(NULL)。在每个新生成的状态中,时间区域中所有的原子公式也标记为空。在生成某个符号状态 curState 的所有后继的时候如果某后继为空,那么标记 curState 的关键原子公式为相关的,并且递归标记与这个关键原子公式有依赖关系的原子公式为相关的。当在可达性分析算法检测到存储空间不够时,开始扩展 PASSED 中的符号状态。根据定义4将 WAITING 中的所有符号状态的时间区域中的原子公式标记为未知,然后由 WAITING 中的符号状态出发,递归标记 PASSED 中与 WAITING 中符号状态原子公式有依赖关系的原子公式为未知原子公式。标记结束,在 WAITING ∪ PASSED 中符号状态时间区域的原子公式有三种状态,空、未知、相关。易见被标记为空的原子公式就是无关公式,可以消除。由于消除了无关公式,PASSED 中符号状态之间可能出现新的包含关系,因此要删除那些被包含的符号状态,就需要重新计算前后继关系。最后,将所有符号状态中的未知原子公式都重置为空。

4.2 动态扩展符号状态实例

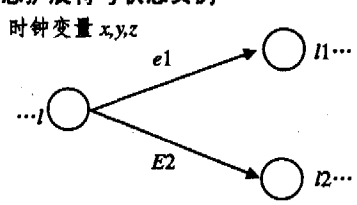


图3 动态扩展符号状态的实例

图3给出了一个在状态生成过程中动态扩展符号状态的例子。此图描述的是运行到某个阶段的时间自动机的一个局部片断，其基本数据如下：设 $e_1=(l,x<1,\emptyset,l_1)$，$e_2=(l,x>1,\{z\},l_2)$，假设模型检验中的某个时刻，在 PASSED 已经有了符号状态 $S_1=(l,x\geqslant 2\wedge y>1\wedge z>3)$、$S_2=(l,x\geqslant 2\wedge y>1\wedge z<2)$，符号状态 $S_3=(l_2,x\geqslant 2\wedge y>1)$在 WAITING 中。$S_1$ 的 e_1 后继与 S_2 的 e_1 后继符号状态均为 $\emptyset$，S_1 的 e_2 后继符号状态与 S_2 的 e_2 后继符号状态均为 S_3 在 WAITING 中。由图3中的算法可知，S_1、S_2 中的原子公式 $x\geqslant 2$ 是关键公式，从而也是相关公式；而 $y>1$ 为无关公式，S_1 中的 $z>3$ 与 S_2 中的 $z<2$ 均为无关公式可以删除。从 S_1、S_2 中删除了无关公式的状态是相同的符号状态，所以可以合并两个状态并重新计算前后继关系。虽然扩展了符号状态，但算法的验证结果不会改变。这种处理方法不仅消减了符号状态个数，降低了内存的需求，同时，由于扩展了 PASSED 符号状态，算法将合并更多后面生成的状态，因此减少了花费在生成后继上的 CPU 时间，加快了可达性分析的速度。

4.3 基于动态扩展符号状态优化时间自动机的可达性算法

在时间自动机的建模与检验过程中往往会使用很多时钟变量与状态，这样就会导致“状态空间爆炸”。对于某些检验，检验的空间是首要考虑的因素，如果没有足够的存储空间，检验的过程就无法实施。只有适当地减少空间消耗而不影响检验的最终结果，这次检验才是成功的。但是不可否认如果在检验过程中每一步都采用动态扩展 PASSED 中的符号状态将要进行大量的计算，对于某些对空间要求不大的时间自动机模型来说得不偿失。因此，我们的分析算法只有当空间需求超出一定阈值的时候才利用上一节中提出的方法进行空间消减。试验表明取得了不错的效果。图4给出了改进后的可达性分析算法，我们根据这个算法设计了相关的模型检验工具，并对相关的协议进行了检验。对于某些协议来说取得了较好的结果。

```
PASSED={};
WAITING={(l0,D0)},标记 D0 中的原子公式为空;
repeat
从 WAITING 中取一个符号状态(l,D),取名为 S,
WAITING=WAITING-{S};
for 每条离开 l 的变换 e,
begin
  if 当前 WAITING∪PASSED 的状态空间大于存储阈值 then
    begin
      TagUnknownAF(WAITING);
      RemoveIrrelevantAF();
      if 约减后的空间大于存储阈值,
      then 返回检验空间不够的信息;
    end
  if SPδ(e,S)为空,S 中的关键原子公式为 kaf
    then TagRelevantAF(kaf),尝试下一个变换;
  计算 S'=SPδ(e,S)=(l',D');
  if l' 是指定要到达的位置 l1,then 返回 成功;
  if 有一个符号状态 S''=(l',D'')在 WAITING∪PASSED 中且
    D'⊆D''
  then 记录 S 到 S''的前趋后继关系;
  else WAITING=WAITING∪{S'}
    并记录 S 到 S'的前趋后继关系;
end
添加 S 到 PASSED;
until WAITING={}
返回 失败。
```

图4 改进后的可达性分析算法

5 案例研究

我们用 C++ 编程语言开发了基于图4优化算法的实时系统模型检验工具，用户需要输入的是时间自动机的文本描述以及用户定义的存储空间阈值。模型检验工具预设了存储空间阈值为 50000，这个阈值可以根据用户的需求进行修改。因为如果在可达性分析算法的每一步都要动态扩展符号状态，将要消耗大量的计算时间，对于状态空间较小的时间自动机来说是没有必要的。只有在分析过程中当 PASSED 中的符号状态数超过用户所指定的阈值时，才调用动态扩展符号状态算法。在检验某个系统的时候，我们使用可行的最小阈值来表示算法需要的最小空间。如果说程序最终的符号状态数必定要超过这个阈值，那么程序将会及时停止，并且给出停止原因。

我们在内存为 3.62G 的惠普 ProLiant 服务器上检验了几个相关的工业案例，这些案例包括 Fischer 互斥协议（Fischer's mutual exclusion protocol），Bang & Olufsion 音频协议（the Bang & Olufsion audio protocol）[10] 和 CSMA/CD 协议（Carrier Sense Multiple Access with Collision Detection protocol）。

本文中的优化技术在检验 Fischer 互斥协议时效果并不好，阈值的设定并没有优化相关的存储空间。但当算法用来检验另外两个协议时会在状态空间到达阈值时自动压缩存储空间，在实验结果上表现为有向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，如果使用如图1中没有经过优化的可达算法将生成 125698 个状态节点与 136111 条边的可达图，耗时 39s；但是当我们将存储空间阈值设为 10000 时，利用图4中基于动态扩展符号状态的优化算法后，最终只生成 6044 个状态节点与 6590 条边的可达图，耗时 378s；当使用动态与静态结合的优化算法时，生成 8071 个状态节点与 8822 条边的可达图，耗时 268s。在文[3～5]中的两种静态优化方法都不能够对此协议的检验进行优化。

表1 CSMA/CD 协议的检验数据

protocol	Optimization Methods (nodes / runtime)			
	Basic	Static IAFR	Dynamic IAFR	Static+Dynamic IAFR
CSMA5	3580 / 2	664 /<1	550 / 1	550 /<1
CSMA6	25905 / 79	2057 / 1	1800 / 4	1800 /<1
CSMA7	N / A	6026 / 2	5000 / 24	5000 /<1
CSMA8	N / A	16907 / 5	13000 /164	13000 / 12
CSMA9	N / A	45836 / 21	33000 / 1270	33000 / 84
CSMA10	N / A	120845 / 96	90000 / 23328	90000 / 105
CSMA11	N / A	311310 / 325	N / A	250000 / 303
CSMA12	N / A	786447 / 1267	N / A	600000 / 1251

对于 CSMA/CD 协议的检验结果如表1所示。我们对比了四种不同的优化方法的结果：图1提出的基本算法 BASIC、静态消减算法 Static IAFR（Static Irrelevent Formulae Removal）[4]、图4提出的动态消减算法 Dynamic IAFR（Dynamic Irrelevent Formulae Removal）以及静态消减算法与动态消减算法的组合 Static+Dynamic IAFR。表中前两种优化方法的数据表示有向可达图的状态节点个数以及检验所花费的时间，后两种优化算法的数据表示了阈值和检验时间。对于某些项，由于状态空间很大，不能得到其实验数据，在表里用 N/

A 表示。

由表 1 可知，动态消减算法没有静态消减与动态消减组合算法的检验时间理想。因为在文[4]中提出的静态消减算法在模型检验的初期通过静态分析删除了大量的无关原子公式，使得可达性分析的速度加快，虽然空间消耗与前者接近，但是检验时间通常比前者快，所以在检验过程中我们偏向于采用两者结合的优化方法。同时，在验证的过程中，我们发现如果将阈值设置得比较大的时候，检验过程所需要的时间会有所下降。

结论 目前有许多针对扩展时间自动机的符号状态来减少状态空间的优化算法。这些方法的特点是基于静态的统计信息。本文提出的算法是动态的算法，它的核心思想是在检验过程中根据相邻符号状态的前驱与后继之间的依赖关系获得可以消去的无关原子公式，然后通过删除这些静态算法不能发现的无关原子公式来扩展符号状态，从而降低模型检验过程中对内存的需求。案例研究的实验数据表明，本文提出的算法能够在状态空间超过阈值时有效地压缩状态空间，但是因为要获得所有的符号状态原子公式间的依赖关系以及合并有包含关系的符号状态，需要一定的计算。一般来说这个算法对于当前的存储空间小于状态空间、必须进行状态空间约减的情况非常适用。

参 考 文 献

1 Clarke E M Jr, Grumberg O, Peled D A. Model Checking. Cambridge: MIT Press, 2000. 1～26

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

3 Daws C, Yovine S. Reducing the number of clock variables of timed automata. In: Burns A, ed. Proc. of the 17th IEEE Real Time Systems Symposiumb (RTSS' 96). Washington, DC, USA: IEEE Computer Society Press, 1996. 73～81

4 ZHAO Jianhua, LI Xuandong, ZHENG Tao, et al. Removing Irrelevant Atomic Formulas for Checking Timed Automata Efficiently. In: Niebert P, ed. Proc of First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS). LNCS 2791. Marseille, France: Springer-Verlag, 2003. 34～45

5 Behrmann G, Bouyer P, Larsen K G, et al. Lower and Upper Bounds in Zone Based Abstractions of Timed Automata. In: Jensen K, ed. Proc. of 10th Int Conf Tools and Algorithms for the Construction and Analysis of Systems (TACAS'04). LNCS 2988. Barcelona, Spain: Springer-Verlag, 2004. 312～326

6 Bengtsson J, Yi Wang. Timed Automata: Semantics, Algorithms and Tools. In: Reisig W, ed. Lecture Notes on Concurrency and Petri Nets 2003. LNCS 3098. Eichstätt, Germany: Springer-Verlag, 2004. 87～124

7 Behrmann G, David A, Larsen K G, et al. Uppaal - Present and Future. In: Proc. of the 40th IEEE Conference on Decision and Control (CDC'2001). Orlando, Florida, USA: IEEE Computer Society Press, 2001,3:2881～2886

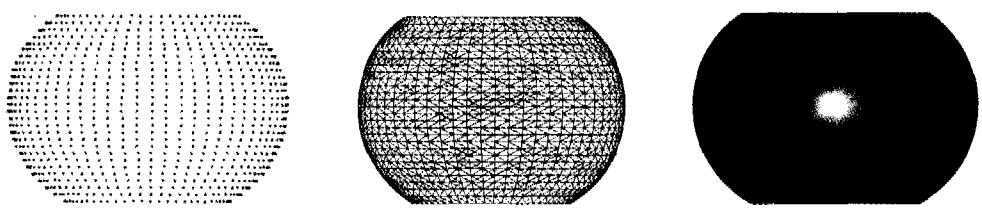
8 Daws C,Olivero A,Tripakis S, et al. The tool Kronos. In: Alur R, ed. DIMACS Workshop on Verification and Control of Hybrid Systems. Hybrid System III: Verification and Control, LNCS 1066. Berlin: Springer-Verlag, 1996. 208～219

9 Wang F. RED: Model-Checker for Timed Automata with Clock-Restriction Diagram. In: Proc. of Workshop on Real-Time Tools, Aalborg University, Denmark:[Technical Report]. 2001-014, ISSN 1404-3203, Department of Information Technology, Uppsala University

10 Havelund K, Skou A, Larsen K G, et al. Formal Modelling and Analysis of an Audio/Video Protocol: An Industrial Case Study Using Uppaal. In: Lin KJ, ed. Proc. of 18th IEEE Real-Time Systems Symposium. San Francisco, CA, USA: IEEE Computer Society Press, 1997. 2～13

(上接第 202 页)

图像处理、三角网格化、自动生成 STL(. ast)文件。图 3 为用本文软件对该断层图像各阶段的处理结果在反求软件 Imageware 中的显示效果，图 4 为自动生成的 STL 文件。



(a) 边界轮廓点　(b) 三角网格化的边界轮廓　(c) STL 文件

图 3

```
solid stl_1
facet normal -0.867126 -0.015804 0.497838
 outer loop
      vertex 0.000001 56.576290 -7.500054
      vertex 0.000001 52.221680 -10.000140
      vertex -3.588281 56.462379 -7.500054
 endloop
endfacet
.....
facet normal -0.174225 0.031245 0.984210
 outer loop
      vertex 5.360947 84.349602 35.000271
      vertex 0.000001 84.077232 32.500229
      vertex 0.000001 84.519791 35.000271
 endloop
endfacet
endsolid stl_1
```

图 4　生成的 STL 格式文件

结论 本文实现了由工业 CT 断层图像到 STL 文件的自动生成，实例验证了其正确性。与通过 CAD 软件重构出样件的实体模型，再将其转换成 STL 文件的方法相比，该方法缩短了开发时间，充分发挥了快速原型制造的优越性。

从理论上讲，构造的三角面片越多，生成的 STL 文件与样件的符合程度就会越高，但是，这会使运算量变大，对计算机的运算速度提出更高的要求。此外，生成的 STL 文件中三角面片过多，亦会给后继处理带来一定的困难，如何在保持精度和数据精简之间取得平衡，这将是下一步需要继续深入研究的问题。

参 考 文 献

1 Lee K H, Woo H. Direct integration of reverse engineering and rapid prototyping [J]. Computer & Industry Engineer, 2000, 38: 21～28

2 王霄. 逆向工程技术及应用[M]. 北京：化学工业出版社，2004. 103～111

3 Liu S, Ma W. Seed-growing segmentation of 3-Dsurface from CT-contourdata [J]. Computer- Aided Design, 1999, 31(8): 517～536

4 Park H, Kim K. Smooth surface approximation to serial cross-sections [J]. Computer-Aided Design, 1996,28(12): 995～1005

5 李发致. 模具先进制造技术[M]. 北京：机械工业出版社，2003. 208～242

6 纪凤欣，秦绪佳，等. 一种基于 CT 图像反求技术的实体几何造型方法[J]. 机械科学技术，2002, 21(1): 165～168

7 金涛，童水光，等. 逆向工程技术[M]. 北京：机械工业出版社，2003. 271～277