ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.4, April 2006, pp.720−728
DOI: 10.1360/jos170720

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

从基于迁移的扩展 Büchi 自动机到 Büchi 自动机*

易 锦[1,2+], 张文辉[1]

[1](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)

[2](中国科学院 研究生院 信息与工程学院,北京 100049)

Efficient Translation from Transition-Based Generalized Büchi Automata to Büchi Automata

YI Jin[1,2+], ZHANG Wen-Hui[1]

[1](Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100080, China)

[2](School of Information Science and Engineering, Graduate School, The Chinese Academy of Sciences, Beijing 100049, China)

+ Corresponding author: Phn: +86-10-62892796, Fax: +86-10-62563894, E-mail: yijin@ios.ac.cn, http://www.ios.ac.cn

Yi J, Zhang WH. Efficient translation from transition-based generalized Büchi automata to Büchi automata. *Journal of Software*, 2006,17(4):720−728. http://www.jos.org.cn/1000-9825/17/720.htm

Abstract: The automata-theoretic approach is one of the state-of-the-art model-checking methods, which consists of the following steps: use a Büchi automaton to represent the abstract system model; use an LTL formula to express the properties to be verified; translate the negation of the LTL formula to a Büchi automaton and check whether the intersection of sentences accepted by the two automata is non-empty. One type of methods for translating LTL formulas to Büchi automata has one step for calculating transition-based generalized Büchi automata (TGBA) and another step for translating TGBA to Büchi automata. This paper redefines the product operation of TGBA according to the characteristics of the accepting language of Büchi automata. This leads to the reduction of the number of states that need to be copied and therefore smaller Büchi automata. The experimental results given at the end of this paper demonstrate the advantage of the approach based on test cases with randomly generated formulas.

Key words: model checking; Büchi automata; LTL (linear temporal logic); TGBA (transition-based generalized Büchi automaton)

摘 要: 目前的模型检测方法中,有一种方法是基于自动机来实现的.具体做法是:将抽象出的系统模型用 Büchi 自动机来表示,将需要验证的性质用 LTL(linear temporal logic)公式来表达;然后将 LTL 公式取反后转化为 Büchi 自动机,并检查这两个自动机接受语言之间的包含关系.有一类 LTL 公式转化为 Büchi 自动机的算法是:在计算过程中,首先得到一个标注在迁移上的扩展 Büchi 自动机(transition-based generalized Büchi automaton,简称 TGBA),然后把这种扩展 Büchi 自动机转换成非扩展的 Büchi 自动机.针对这类转换算法,根据 Büchi 自动机接受语言的特点,重新定义了基于迁移的扩展 Büchi 自动机的求交运算,减少了需要复制的状态个数,使转换后的自

* Supported by the National Natural Science Foundation of China under Grant Nos.60223005, 60373050, 60421001, 60573012 (国家自然科学基金); the National Grand Fundamental Research 973 Program of China under Grant No.2002cb312200 (国家重点基础研究发展规划(973))

Received 2004-11-22; Accepted 2005-10-27

动机具有较少的状态.测试的结果表明:对随机产生的公式,新算法相对于以往的算法有明显的优势.

关键词: 模型检测;Büchi 自动机;LTL(linear temporal logic)公式;TGBA(transition-based generalized Büchi automaton)

中图法分类号: TP301 文献标识码: A

在基于自动机理论的模型检测方法[1]中,首先是将抽象出的系统模型用 Büchi 自动机来表示,然后将需要验证的性质用一个 LTL(linear temporal logic)公式来描述,并将该公式取反后转化为 Büchi 自动机,最后检查系统自动机的接受语言是否被包含在性质自动机的接受语言中.如果是,则说明系统具有 LTL 公式所描述的性质;反之则没有.通常,检查两个自动机接受语言的包含关系需要求交运算,这样所得自动机的状态个数就是这两个自动机状态个数的乘积.

由于模型检测方法所面临的最大问题是状态空间爆炸,所以我们要尽量减少求交后自动机的状态个数:一方面我们可以减少系统模型的状态个数;另一方面也可以减少 LTL 公式所生成的自动机的状态个数,从而减少求交后自动机的状态个数.一般来说,将 LTL 公式转换成 Büchi 自动机的算法需要经过 3 个步骤:第 1 步是将 LTL 公式按照一定的规则进行重写;第 2 步是将重写后的公式转换为 Büchi 自动机;第 3 步是将 Büchi 自动机尽可能地进行化简.我们可以根据 LTL 公式转换所得 Büchi 自动机的类型将转换算法分为 3 类:第 1 类是将公式转换为在自动机状态上进行标注的 Büchi 自动机,可接受集合是自动机状态的子集.这类算法主要是基于 tableau 规则[2−5];第 2 类是将公式转换为在迁移上进行标注的 Büchi 自动机,可接受集合由迁移构成,相关文献为文献[6–8];第 3 类仍是在迁移上进行标注,但是可接受集合由状态来构成,其具体算法可参考文献[9,10].

本文工作是基于第 2 类算法的.当该类算法得到一个标注在迁移上的扩展 Büchi 自动机时,为了能够应用目前比较成熟的 check emptiness 模型检测算法,往往需要把这种扩展 Büchi 自动机转换成一般的 Büchi 自动机.文献[7]的转换方法如下:设自动机 TGBA(transition-based generalized Büchi automaton,见后文定义 1.3)=$\langle S,\Sigma,\Delta,Q_0,F\rangle$,其可接受迁移集合 $F=\{f_1,f_2,\ldots,f_r\}$,转换后所得自动机 BA(见后文定义 1.2)=$\langle S\times\{0,\ldots,r\},\Sigma,\Delta',Q_0\times\{0\},S\times\{r\}\rangle$.这里,$\Delta'=\{((q,j),a,(q',j'))|(q,a,q')\in\Delta,j'=next(j,(q,a,q')),0\le j\le r\}$,其中的 *next* 函数定义如下:

$$next(j,t)=\begin{cases}\max\{i\,|\,0\le i\le r,\forall 0<k\le i.(t\in f_k)\}, & \text{if } j=r\\ \max\{i\,|\,j\le i\le r,\forall j<k\le i.(t\in f_k)\}, & \text{if } i\ne r\end{cases}.$$

文献[6]的方法是为需要转换的 TGBA 构造一个 degeneralizer(简称 DG).DG 是一个确定的 Büchi 自动机,其状态个数为 TGBA 接受集合中元素个数加 1,其迁移具有不同的优先级,迁移上的标识为可接受集元素 f_i,转换的过程即是将 TGBA 与对应的 DG 作同步求交运算.文献[8]的转换方法与文献[6]类似,也是需要构造一个自动机,然后令 TGBA 与构造的自动机求交.在上述 3 种转换方法中,都需要复制 TGBA 的状态.在最坏情况下,一个状态个数为 N 的 TGBA 转化成个数为 $N\times(|F|+1)$的一般 Büchi 自动机,其中的$|F|$为可接受集合里的元素个数,即 LTL 公式转换成 NNF(negation normal form)形式后“U”运算符的个数.根据 Büchi 自动机可接受语言的特点以及对现有转换方法的分析,我们重新定义了 TGBA 自动机的求交运算,记作⊗,使得 TGBA 在根据⊗运算转换为一般自动机时,状态个数会少于现有算法所得自动机的状态个数.

本文第 1 节介绍一些基本的概念.第 2 节定义 TGBA 自动机的求交运算 ⊗,并证明利用运算⊗转换所得的自动机的接受语言与利用原定义转换所得的自动机的接受语言是一致的.第 3 节给出实验结果.最后一节是结论和进一步的工作方向.

1 基础知识

定义 **1.1**(**LTL** 公式及其语义). LTL 公式是由原子命题(atomic proposition)集合 *AP*、布尔运算符¬,∧,∨,时序运算符 *X*(*next*),*U*(*until*),*R*(*release*),*F*(*eventually*) 和 *G*(*always*) 按照下面的语法来构成,其中 $p\in AP,\varphi::=p|\neg\varphi|\varphi\vee\varphi|X\varphi|\varphi\mathrm{U}\varphi$.而∧,*R*,*F*,*G* 运算符可根据性质 $\varphi\wedge\psi=\neg(\neg\psi\vee\neg\varphi),\varphi R\psi=\neg(\neg\varphi\mathrm{U}\neg\varphi),F\varphi=\{TRUE\}\mathrm{U}\varphi,$

$G\varphi=\{FALSE\}R\varphi$来定义.令$S$是状态集合,我们用$\pi^k$来表示从$s_k$开始的一个无限长的状态序列$\pi^k=s_ks_{k+1}...$,设$L:S\to 2^{AP}$是从状态集合$S$到$AP$幂集的函数.LTL公式的语义定义如下:

- $L,\pi^k|=p$,如果$p\in L(s_k)$;
- $L,\pi^k|=\neg\varphi$,如果$L,\pi^k|\neq\varphi$;
- $L,\pi^k|=\varphi\vee\psi$,如果$L,\pi^k|=\varphi$,或者$L,\pi^k|=\psi$;
- $L,\pi^k|=X\varphi$,如果$L,\pi^{k+1}|=\varphi$;
- $L,\pi^k|=\varphi U\psi$,如果$\exists i\geq k,L,\pi^i|=\psi$且$\forall k\leq j<i,L,\pi^k|=\varphi$.

定义 **1.2**(**BA**,**Büchi automaton**). Büchi自动机是一个5元组:$BA=\langle S,\Sigma,\Delta,Q_0,F\rangle$,其中,$S$是有限的状态集合;$\Sigma$是有限的字母集合;$\Delta\subseteq S\times\Sigma\times S$是一个有字母标识的迁移关系;$Q_0\subseteq S$是初始状态集合;$F\subseteq S$是可接受状态集合.

Büchi自动机的一个有限路径是满足对所有$0\leq i<n$存在$a_i\in\Sigma$,使得$(s_i,a_i,s_{i+1})\in\Delta$的状态序列$\rho=s_0s_1s_2...s_n$.$BA$的一个无限路径是满足对所有$0\leq i$存在$a_i$使得$(s_i,a_i,s_{i+1})\in\Delta$的状态序列$\rho=s_0s_1s_2....$字母表$\Sigma$上的一个无限长字符串$\omega=a_0a_1a_2...$对$BA$来说是可接受的,当且仅当存在一个无限路径$\rho=s_0s_1s_2...$满足$s_0\in Q_0,(s_i,a_i,s_{i+1})\in\Delta$且$inf(\rho)\cap F\neq\varnothing$,其中的$inf(\rho)$表示在路径$\rho$中出现无限多次的状态的集合.

定义 **1.3**(**TGBA**,**transition-based generalized Büchi automaton**[6]). 基于迁移的扩展Büchi自动机是一个5元组:$TGBA=\langle S,\Sigma,\Delta,Q_0,F\rangle$,$S$是有限的状态空间,$\Sigma$是有限的字母标识集合,$\Delta\subseteq S\times\Sigma\times S$是一个有标识的迁移关系,$Q_0\subseteq S$是初始状态集合,$F\subseteq 2^{\Delta}$是可识别的状态转换集的集合.

TGBA的一个有限路径是满足对所有$0\leq i<n$存在$a_i\in\Sigma$,使得$(s_i,a_i,s_{i+1})\in\Delta$的状态转换序列$\pi=(s_0,a_0,s_1)(s_1,a_1,s_2)(s_2,a_2,s_3)...(s_{n-1},a_{n-1},s_n)$.TGBA的一个无限路径是满足对所有$0\leq i$存在$a_i\in\Sigma$,使得$(s_i,a_i,s_{i+1})\in\Delta$的状态转换序列$\pi=(s_0,a_0,s_1)(s_1,a_1,s_2)(s_2,a_2,s_3)\ldots$字母表$\Sigma$上的一个无限长字符串$s_2=a_0a_1a_2...$对TGBA来说是可接受的,当且仅当存在一个无限路径$\pi=(s_0,a_0,s_1)(s_1,a_1,s_2)(s_2,a_2,s_3)...$满足$s_0\in Q_0,(s_i,a_i,s_{i+1})\in\Delta$且对所有$f\in F$有$inf(\pi)\cap f\neq\varnothing$.其中的$inf(\pi)$表示在路径π中出现无限多次的状态转换的集合.

2 TGBA求交运算的重新定义和证明

我们用自动机的求交运算来形式化定义文献[7]中的方法.定义2.1说明了如何根据TGBA构造一个确定的Büchi自动机BA;定义2.2通过定义TGBA与BA的求交运算$\cap$得到了一个与TGBA具有相同接受语言的一般Büchi自动机BA.

定义 **2.1**(**FBA**:**TGBA** 到 **BA** 的映射). 设$A=\langle S,\Sigma,\Delta,I,F\rangle$为TGBA,其中$F=\{f_1,f_2,...,f_n\}$.令$f_{n+1}=\varnothing$,则$FBA(A)=\langle S',\Sigma',\Delta',I',F'\rangle$为BA.其中

- $S'=\{q_0,q_1,...,q_n\}$;
- $\Sigma'=\Delta$;
- $\Delta'=\{(q_i,u,q_k)|u\in f_{i+1}\cap...\cap f_k,u\notin f_{k+1},0\leq i\leq n-1,i+1\leq k\leq n\}$
  $\cup\{(q_i,u,q_i)|u\notin f_{i+1},0\leq i\leq n-1\}$
  $\cup\{(q_n,u,q_i)|u\in f_1\cap...\cap f_i,u\notin f_{i+1},0<i\leq n\}$
  $\cup\{(q_n,u,q_0)|u\notin f_1\}$;
- $I'=\{q_0\}$;
- $F'=\{q_n\}$.

定义 **2.2**($\cap$ 运算). 设$A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$为TGBA,$A_2=\langle S_2,\Sigma_2,\Delta_2,I_2,F_2\rangle$为BA,$\Sigma_2=\Delta_1$且$S_1\cap S_2=\varnothing$,则$A_1\cap A_2=\langle S,\Sigma,\Delta,I,F\rangle$其中

- $S=S_1\times S_2$;
- $\Sigma=\Sigma_1$;
- $\Delta=\{((s_1,s_2),a,(s_1',s_2'))|(s_1,a,s_1')\in\Delta_1,(s_2,(s_1,a,s_1'),s_2')\in\Delta_2\}$;
- $I=I_1\times I_2$;

- $F=S_1\times F_2$.

根据 A_1 构造 A_2 的具体做法是:首先将 F_1 的 n 个元素 $f_1,...,f_n$ 进行排序(按照任意顺序,排列顺序对状态个数的影响将在第 4 节中加以讨论),然后根据这个排列顺序生成相应的 n+1 个状态.若对状态 $q_i(0\le i\le n-1)$输入字符 u,则 q_i 经 u 可达其所能到达的最远的状态 q_k,即 $u\in f_{i+1}\cap...\cap f_k,u\notin f_{k+1}$,对 q_n 可将其看成初始状态 q_0 而得到相应的迁移.最后,标明 q_n 是可接受状态.由构造过程可知,该 BA 的可接受语言至少包含了每个 $f_i(0\le q_i\le q_n)$中的元素.这样,再根据定义 2.2,就可以把一个 TGBA 转换成非扩展的 Büchi 自动机了.

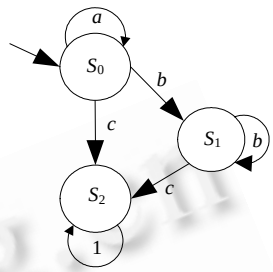

Fig.1 TGBA of the formula aU(bUc)

图 1 aU(bUc)公式的 TGBA

下面举一个例子来说明.由公式 aU(bUc)所得的 TGBA 如图 1 所示,其字母表由集合{a,b,c}的幂集构成,根据文献[7]中的算法可知:在 F_1 中每个 f_i 对应一个包含“U”算子的时序子公式,设 f_1 对应 bUc 和 f_2 对应 aU(bUc).令 1 代表集合{a,b,c}的任意子集,则

- $f_1=\{(s_0,\{a\},s_0),(s_1,\{c\},s_2),(s_0,\{c\},s_2),(s_2,1,s_2)\}$.
- $f_2=\{(s_0,\{b\},s_1),(s_1,\{b\},s_1),(s_1,\{c\},s_2),(s_0,\{c\},s_2),(s_2,1,s_2)\}$.

选择 F_1 中的元素的排序顺序为:f_1,f_2.根据 F_1 所生成的 BA 如图 2 所示(在图中直接用 f_i 表示所有属于集合 f_i 的字符).将 TGBA 与 BA 求交后所得的 Büchi 自动机如图 3 所示,图 3 中标识的 $q_{i\_}s_j$ 表示该状态是图 1 中的 s_j 状态与图 2 中的 q_i 状态求交所得到的.

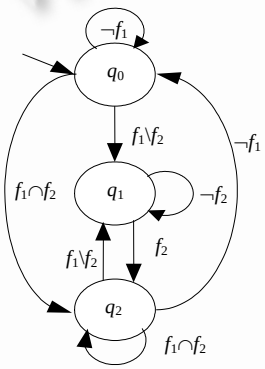

Fig.2 BA built by ordered F_1

图 2 根据排序的 F_1 生成的 BA

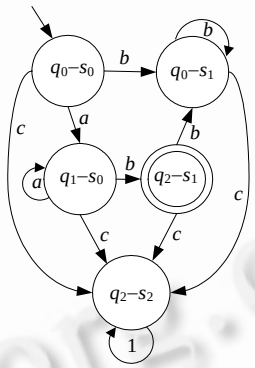

Fig.3 BA built by $\cap$ operator

图 3 根据 $\cap$ 运算得到的自动机

通过上面的介绍可知:现有方法在进行转换时,TGBA 的所有状态都需要和构造的 BA 进行求交运算.但根据自动机可接受语言的定义,我们仅需要语言中无限次循环的部分与 f_i 的交为非空即可.因此,我们重新定义 TGBA 和 BA 的求交运算,只将 TGBA 中的强连通部分(SCC)与 BA 进行求交运算,其余部分不变.这样,在一般情况下,需要复制的状态就会减少,因此最后转换所得的自动机的个数也会相应减少.下面给出上述想法的形式化定义.

定义 **2.3**. 设 $A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$ 为 TGBA,A_1 中强连通环 scc_i 是一个极大强连通环,当且仅当在 A_1 中不存在另一个强连通环 scc_j,使得 scc_i 中的每一个状态均属于 scc_j.

定义 **2.4**. 设 $A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$ 为 TGBA,若 A_1 中极大强连通环 scc_i 上的迁移集合与 F_1 中每个元素的交集不为空,则将由该强连通环的状态组成的集合记为 $\mathrm{SCC}(A_1)_i$.为方便叙述,我们用

- $\mathrm{SCC}(A_1)$表示所有 $\mathrm{SCC}(A_1)_i$ 的并集;
- $s\mathrm{SCC}(A_1)(a,b)$表示 $a,b\in\mathrm{SCC}(A_1)$且 a,b 在同一个极大强连通环中;
- $d\mathrm{SCC}(A_1)(a,b)$表示 $a,b\in\mathrm{SCC}(A_1)$且 a,b 在不同的极大强连通环中.

定义 **2.5(⊗运算)**. 设 $A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$ 为 TGBA,$A_2=\langle S_2,\Sigma_2,\Delta_2,I_2,F_2\rangle$ 为 BA,$\Sigma_2=\Delta_1$ 且 $S_1\cap S_2=\varnothing$,则 $A_1\otimes A_2=$

$\langle S',A',\Delta',I',F'\rangle$.其中

- $S'=(\mathrm{SCC}(A_1)\times S_2)\cup(A_1\backslash\mathrm{SCC}(A_1))$;
- $\Sigma'=\Sigma_1$;
- $\Delta'=\{((s_1,s_2),a,(s_1',s_2'))|s\mathrm{SCC}(A_1)(s_1,s_1'),(s_1,a,s_1')\in\Delta_1,(s_2,(s_1,a,s_1'),s_2')\in\Delta_2\}$
  $\cup\{((s_1,s_2),a,(s_1',s_2'))|d\mathrm{SCC}(A_1)(s_1,s_1'),(s_1,a,s_1')\in\Delta_1,s_2\in S_2,s_2'\in I_2\}$
  $\cup\{((s_1,s_2),a,s_1')\}|s_1\in\mathrm{SCC}(A_1),s_1'\in S_1\backslash\mathrm{SCC}(A_1),(s_1,a,s_1')\in\Delta_1,s_2\in S_2\}$
  $\cup\{(s_1,a,(s_1',s_2'))|s_1\in S_1\backslash\mathrm{SCC}(A_1),s_1'\in\mathrm{SCC}(A_1),(s_1,a,s_1')\in\Delta_1,s_2'\in I_2\}$
  $\cup\{(s_1,a,s_1')\}|s_1,s_1'\in S_1\backslash\mathrm{SCC}(A_1),(s_1,a,s_1')\in\Delta_1\}$;
- $I'=((I_1\cap\mathrm{SCC}(A_1))\times I_2)\cup(I_1\backslash\mathrm{SCC}(A_1))$;
- $F'=\mathrm{SCC}(A_1)\times F_2$.

根据 ⊗ 的定义,我们将图 1 和图 2 中的自动机求交,由于图 1 中 $SCC(A_1)=\{s_2\}$,所以只需让 s_2 与图 2 中的 BA 求交,所得自动机如图 4 所示.

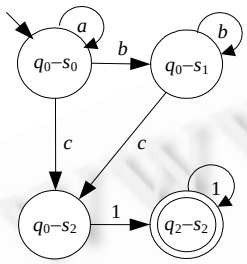

Fig.4 BA built by ⊗ operator

图 4 根据 ⊗ 运算得到的自动机

下面,我们将证明求交运算⊗定义的正确性.为方便叙述,在文章的以下部分,我们用:

- $S(A)$表示 A 的状态集;
- $\Delta(A)$表示 A 的迁移集;
- $I(A)$表示 A 的初始状态集合;
- $F(A)$表示 A 的可接受集;
- $R(A)$表示 A 的路径集合;
- $|r|$表示路径 r 的长度;
- $C(r)$表示路径 r 所对应的字符串集合;
- $L(A)$表示 A 的可接受语言.

定义 2.6. 设 $A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$ 为 TGBA,$A_2=\langle A_2,\Sigma_2,\Delta_2,I_2,F_2\rangle$ 为 BA,$\Sigma_2=\Delta_1$,$r_0\in S(A_1\cap A_2)$或 $r_0\in S(A_1\otimes A_2)$.

- 若 $r_0=(s,m)\in S(A_1)\times S(A_2)$,则 $red(r_0)=s$;
- 若 $r_0=s\in S(A_1)$,则 $red(r_0)=s$;
- 若 r,r'为 $A_1\cap A_2$ 或 $A_1\otimes A_2$ 上的路径且 $r=r_0r$,则 $red(r)=red(r_0)red(r')$.

引理 2.1. 设 $A_1=\langle S_1,\Sigma_1,\Delta_1,I_1,F_1\rangle$ 为 TGBA,$A_2=FBA(A_1)$且 $I(A_2)=\{q_0\}$,$r\in S(A_1\cap A_2)$,$t\in S(A_1\otimes A_2)$,$red(r)=red(t)$.对任意给定字符 $u\in\Sigma_1$,

- 若$\exists r'\in S(A_1\cap A_2)$使得$(r,u,r')\in\Delta(A_1\cap A_2)$,则$\exists t'\in S(A_1\otimes A_2)$使得$(t,u,t')\in\Delta(A_1\otimes A_2)$且 $red(t')=red(r')$.
- 若$\exists t'\in S(A_1\otimes A_2)$使得$(t,u,t')\in\Delta(A_1\otimes A_2)$,则$\exists r'\in S(A_1\cap A_2)$使得$(r,u,r')\in\Delta(A_1\cap A_2)$且 $red(r')=red(t')$.

证明:设 $r=(s_1,s_2)$,$t=(s_1,s_2^*)$或 $t=s_1$,$r'=(s_1',s_2')$.先证明第 1 种情况:已知$\exists r'$使得$(r,u,r')\in\Delta(A_1\cap A_2)$,即$((s_1,s_2),u,(s_1',s_2'))\in\Delta(A_1\cap A_2)$且$(s_1,u,s_1')\in\Delta(A_1)$,$(s_2,(s_1,u,s_1'),s_2')\in\Delta(A_2)$.根据定义 2.5 中 Δ'的定义,讨论以下几种情况:

(1) $s\mathrm{SCC}(A_1)(s_1,s_1')$

因为$(s_1,u,s_1')\in\Delta(A_1)$,根据 FBA 的定义可知:存在$(s_2^*,(s_1,u,s_1'),s_2'^*)\in\Delta(A_2)$.根据 ⊗ 的定义,存在状态 $t'=(s_1',s_2'^*)\in S(A_1\otimes A_2)$使得$((s_1,s_2^*),u,(s_1',s_2'^*))\in\Delta(A_1\otimes A_2)$成立,显然 $red(r')=red(t')=s_1'$.

(2) $d\mathrm{SCC}(A_1)(s_1,s_1')$

因为$(s_1,u,s_1')\in\Delta(A_1)$,根据 ⊗ 的定义,存在状态 $t'=(s_1',q_0)\in S(A_1\otimes A_2)$使得$((s_1,s_2^*),u,(s_1',q_0))\in\Delta(A_1\otimes A_2)$成立,显然,$red(r')=red(t')=s_1'$.

(3) $s_1\in\mathrm{SCC}(A_1)$且 $s_1'\in S_1\backslash\mathrm{SCC}(A_1)$

因为$(s_1,u,s_1')\in\Delta(A_1)$,根据 ⊗ 的定义,状态 $s_1'\in S(A_1\otimes A_2)$且$((s_1,s_2^*),u,s_1')\in\Delta(A_1\otimes A_2)$.令 $t'=s_1'$,显然,$red(r')=red(t')=s_1'$.

(4) $s_1\in S_1\backslash\mathrm{SCC}(A_1)$且 $s_1'\in\mathrm{SCC}(A_1)$

因为$(s_1,u,s_1')\in\Delta(A_1)$,根据 $\otimes$ 的定义,存在状态 $t'=(s_1',q_0)\in S(A_1\otimes A_2)$使得$(s_1,u,(s_1',q_0))\in\Delta(A_1\otimes A_2)$成立,显然,$red(r')=red(t')=s_1'$.

(5) $s_1\in S_1\backslash SCC(A_1)$且 $s_1'\in S_1\backslash SCC(A_1)$

因为$(s_1,u,s_1')\in\Delta(A_1)$,根据 $\otimes$ 的定义,状态 $s_1'\in S(A_1\otimes A_2)$且$(s_1,u,s_1')\in\Delta(A_1\otimes A_2)$.令 $t'=s_1'$,显然,$red(r')=red(t')=s_1'$.

在第 2 种情况时,由假设可知:$\exists t'\in S(A_1\otimes A_2)$使得$(t,u,t')\in\Delta(A_1\otimes A_2)$.因此,$(red(t),u,red(t'))\in\Delta(A_1)$,即$(s_1,u,red(t'))\in\Delta(A_1)$.根据 FBA 的定义可知:存在 s_2' 使得$(s_2,(s_1,u,red(t')),s_2')\in\Delta(A_2)$,根据 $\cap$ 的定义可知:存在状态 $r'=(red(t'),s_2')\in S(A_1\cap A_2)$使得$((s_1,s_2),u,r')\in\Delta(A_1\cap A_2)$,即$(r,u,r')\in\Delta(A_1\cap A_2)$成立,显然,$red(r')=red(t')$. □

引理 **2.2**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,$r\in R(A_1\cap A_2)$是以 r_0 为起点的任意路径,$w=u_0u_1...\in C(r)$为 r 上任意字符串,则存在以$(red(r_0),q_0)$或 $red(r_0)$为起点的路径 $r'\in R(A_1\otimes A_2)$使得 $red(r')=red(r)$且 $w\in C(r')$.

证明:用归纳法证明如下:

- 当 $r=r_0r_1$,$|r|=2$,$w=u_0$ 时

设 $r_0=(s_0,t_0)$,$r_1=(s_1,t_1)$,则$(s_0,u_0,s_1)\in\Delta(A_1)$.令 $r_0'=(s_0,q_0)$(当 $s_0\in SCC(A_1)$时)或 $r_0'=s_0$(当 $s_0\notin SCC(A_1)$时),显然,$red(r_0')=red(r_0)$.由引理 2.1 可知:存在 r_1' 使得$(r_0',u_0,r_1')\in\Delta(A_1\otimes A_2)$且 $red(r_1')=red(r_1)$.显然,$r'=r_0'\ r_1'\in R(A_1\otimes A_2)$,$red(r')=red(r)$.由于$(r_0',u_0,r_1')\in\Delta(A_1\otimes A_2)$且 $w=u_0$,所以,$w\in C(r')$.

- 当 $r=r_0r_1...r_k$,$|r|=k+1>2$,$w=u_0u_1...u_{k-1}$ 时

根据归纳假设,我们有 $r_0'\ r_1'...r_{k-1}'\in R(A_1\otimes A_2)$使得 $red(r_0'\ r_1'...r_{k-1}')=red(r_0r_1...r_{k-1})$,且 $u_0u_1...u_{k-2}\in C(r_0'\ r_1'...r_{k-1}')$.由于 $red(r_{k-1}')=red(r_{k-1})$且$(r_{k-1},u_{k-1},r_k)\in\Delta(A_1\cap A_2)$,由引理 2.1 可知:存在 r_k' 使得$(r_{k-1}',u_{k-1},r_k')\in\Delta(A_1\otimes A_2)$且 $red(r_k')=red(r_k)$.令 $r'=r_0'\ r_1'...r_{k-1}'\ r_k'$,则 $r'\in R(A_1\otimes A_2)$,$red(r')=red(r)$且 $w=u_0u_1...u_{k-1}\in C(r')$. □

引理 **2.3**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,$r\in R(A_1\otimes A_2)$为以 r_0 为起点的 $A_1\otimes A_2$ 上的任意路径,$w=u_0u_1...\in C(r)$为 r 上任意字符串,则对任意 $0\le i\le n$ 存在以$(red(r_0),q_i)$为起点的路径 $r'\in R(A_1\cap A_2)$使得 $red(r')=red(r)$且 $w\in C(r')$.

证明:(略)证明方法与引理 2.2 相同. □

引理 **2.4**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,字符串 $w=u_0...u_{k-1}$,$r=r_0...r_k\in R(A_2)$,$r'=r_0'...r_k'\in R(A_2)$.若 $w\in C(r)$且 $w\in C(r')$,$r_0=q_i$,$r_k=q_n$,$r_0'=q_j$,$0\le i\le j\le n$,则 r'中存在某个状态为 q_n.

证明:用归纳法证明如下:

- 当 $k=1$ 时,$r_1=q_n$,则 $r_0'=q_n$ 或 $r_1'=q_n$,结论成立.
- 当 $k>1$ 时,若 $r_0'=q_j=q_n$,则结论成立.若 $r_0'=q_j\ne q_n$,设 $r_1=q_x$,$r_1'=q_y$.由于 $w\in C(r')$且 $w\in C(r)$故$(r_0,u_0,r_1)\in\Delta(A_2)$且$(r_0',u_0,r_1')\in\Delta(A_2)$,根据 A_2 迁移特点可知 $x\le y$,根据归纳法原理可知 $r_1'...r_k'$ 中存在某个状态为 q_n,因此,r'中存在某个状态为 q_n. □

引理 **2.5**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,字符串 $w=u_0...u_{k-1}$,$r=r_0...r_k\in R(A_2)$,$r'=r_0'...r_k'\in R(A_2)$.若 $w\in C(r)$且 $w\in C(r')$,$r_0=r_k=q_n$,则 r'中存在某个状态为 q_n.

证明:(略)证明方法与引理 2.4 相同. □

引理 **2.6**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,$r=r_0r_1...\in R(A_2)$,$r'=r_0'\ r_1'...\in R(A_2)$.若有无限长字符串 w 使得 $w\in C(r)$且 $w\in C(r')$,则 $q_n\in inf(r)$当且仅当 $q_n\in inf(r')$.

证明:若 $q_n\in inf(r)$,则在 r 上存在无穷多个 q_n,将 r 分为无穷多个子路径,每个子路径上的最后一个状态为 q_n,且 q_n 在每个子路径上出现的次数为一个偶数,则任意一个这样的子路径为 $r_s=q_i...q_n...q_n$,将 r'相应地也分为无穷个子路径.设与 r_s 对应的子路径为 $r_s'=q_j...q_m...q_l$,由于 $w\in C(r')$且 $w\in C(r)$,则存在 w 的片段 w',使得 $w'\in C(r_s)$且 $w'\in C(r_s')$.根据引理 2.5 可知,在任意 r_s' 中一定含有状态 q_n,所以在 r'中有无穷多个 q_n,即 $q_n\in inf(r')$.同理可证:若 $q_n\in inf(r')$,则 $q_n\in inf(r)$. □

引理 **2.7**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\}$,$r=r_0r_1...$是 $A_1\cap A_2$ 上的无穷路径,$w\in C(r)$为路径 r 上任意字符串,且存在 $l\in Nat$,使得对任意 $j\ge0$ 有 $red(r_j)\in SCC(A_1)_l$,那么存在以$(red(r_0),q_0)$为起点的 $A_1\otimes A_2$ 上的无

穷路径 $r'=r'_0\ r'_1...$使得 $w\in C(r')$且 $inf(r)\cap F(A_1\cap A_2)\neq\varnothing$ 当且仅当 $inf(r')\cap F(A_1\otimes A_2)\neq\varnothing$.

证明:由引理 2.2 可知:存在以$(red(r_0),q_0)$为起点的 $A_1\otimes A_2$ 上的无穷路径 $r'=r'_0\ r'_1...$使得 $red(r')=red(r)$且 $w\in C(r')$.我们先证明:若 $inf(r)\cap F(A_1\otimes A_2)\neq\varnothing$,则 $inf(r')\cap F(A_1\cap A_2)\neq\varnothing$.用反证法.假设 $inf(r)\cap F(A_1\otimes A_2)\neq\varnothing$ 时有 $inf(r')\cap F(A_1\cap A_2)=\varnothing$.

- 令字符串 $w=w_0w_1...$,由于 $red(r')\in \mathrm{SCC}(A_1)_l$,根据 $A_1\otimes A_2$ 的定义可知 r'上任意状态 $r_{i'}$ 为(s_j,q_k)形式,其中 $s_j\in S(A_1),q_k\in S(A_2)$,我们称 q_k 是状态 $r_{i'}$ 在 A_2 上所对应的状态.
- 令 r'上每个状态在 A_2 上对应的状态所形成的序列为 rt,它是以 q_0 为起点的无穷路径,根据 $A_1\otimes A_2$ 的定义可知 $w'=(red(r'_0),w_0,red(r'_1))(red(r'_1),w_1,red(r'_2))...\in C(rt)$.由于 $inf(r')\cap F(A_1\otimes A_2)\neq\varnothing$ 且 $F(A_1\otimes A_2)=\mathrm{SCC}(A_1)\times q_n\}$,可知 r'中存在无限多个形式为(s_j,q_n)的状态.(s_j,q_n)对应在 rt 上的状态为 q_n,所以在 rt 上有无限多个 q_n 出现,故 $inf(rt)\cap F(A_2)\neq\varnothing$.
- 令 r 上每个状态在 A_2 上对应的状态所形成的序列为 rs,根据 $A_1\cap A_2$ 的定义可知 $w''=(red(r_0),w_0,red(r_1))(red(r_1),w_1,red(r_2))...\in C(rs)$.由于 $inf(r)\cap F(A_1\cap A_2)=\varnothing$,所以 $inf(rs)\cap F(A_2)=\varnothing$.
- 由于 $red(r')=red(r)$,所以 $w'=w''$,因此有 $w'\in C(rs)$且 $w'\in C(rt)$,故根据引理 2.6 可知 $inf(rs)\cap F(A_2)=\varnothing$,则 $inf(rt)\cap F(A_2)=\varnothing$.推导出矛盾,故假设不成立.因此,若 $inf(r)\cap F(A_1\otimes A_2)\neq\varnothing$,则 $inf(r')\cap F(A_1\cap A_2)\neq\varnothing$.

同理可证明:若 $inf(r')\cap F(A_1\cap A_2)\neq\varnothing$,则 $inf(r)\cap F(A_1\otimes A_2)\neq\varnothing$. □

引理 **2.8**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$且 $S(A_2)=\{q_0,...,q_n\},r=r_0r_1...$是 $A_1\otimes A_2$ 上的无穷路径,$w\in C(r)$为路径 r 上任意字符串,且存在 $l\in$Nat,使得对任意 $j\geq 0$ 有 $red(r_j)\in \mathrm{SCC}(A_1)_l$,那么对任意 $0\leq i\leq n$,存在以$(red(r_0),q_i)$为起点的 $A_1\cap A_2$ 上的无穷路径 $r'=r'_0\ r'_1...$,使得 $w\in C(r')$且 $inf(r')\cap F(A_1\cap A_2)\neq\varnothing$,当且仅当 $inf(r')\cap F(A_1\otimes A_2)\neq\varnothing$.

证明:对任意给定满足 $0\leq i\leq n$ 的 i,由引理 2.3 可知:存在以$(red(r_0),q_i)$为起点的 $A_1\cap A_2$ 上的无穷路径 $r'=r'_0\ r'_1...$,使得 $red(r')=red(r)$且 $w\in C(r')$.剩下需要证明的是 $inf(r)\cap F(A_1\otimes A_2)\neq\varnothing$,当且仅当 $inf(r')\cap F(A_1\cap A_2)\neq\varnothing$.这部分的证明与引理 2.6 的相关部分的证明类似(略). □

定理 **2.1**. 设 A_1 为 TGBA,$A_2=FBA(A_1)$,则 $L(A_1\cap A_2)=L(A_1\otimes A_2)$.

证明:(1) 我们首先证明:对任意可接受字符串 $w\in L(A_1\cap A_2)$,则 $w\in L(A_1\otimes A_2)$.设 $w=w_0w_1...$,其对应的可接受路径为 $r=r_0r_1r_2....$由于 r 是可接受路径,存在 $x=r_k\in F(A_1\cap A_2)$,且 x 在 r 中出现无限多次.因此,存在 $l\in$Nat,使得对所有 $j\geq k$ 有 $red(r_j)\in \mathrm{SCC}(A_1)_l$,在 r 中,从状态 r_k 开始向左检查每一个状态 $r_i(0\leq i<k)$是否有 $red(r_i)\notin \mathrm{SCC}(A_1)_l$ 成立.下面分两种情况来讨论.

若在 r 上没有状态 r_i 使得 $red(r_i)\notin \mathrm{SCC}(A_1)_l$,则对任意状态 $s\in S(A_1)$都有 $s\in \mathrm{SCC}(A_1)_l$,那么 $A_1\cap A_2=A_1\otimes A_2$,显然,$w\in L(A_1\cap A_2)$,则 $w\in L(A_1\otimes A_2)$.

如果找到状态 r_{m-1} 使得 $red(r_{m-1})\notin \mathrm{SCC}(A_1)_l$,且 r_{m-1} 为第一个找到的状态.那么在 r 路径中,设以 r_m 为起点的路径为 $r^*=r_mr_{m+1}...$,则 $w^*=w_mw_{m+1}...\in C(r^*)$.由引理 2.7 可知:存在以$(red(r_m),q_0)$为起点的路径 $r'^*\in R(A_1\otimes A_2)$,使得 $w^*\in C(r'^*)$并且若 $inf(r^*)\cap F(A_1\cap A_2)\neq\varnothing$,则 $inf(r'^*)\cap F(A_1\otimes A_2)\neq\varnothing$.设以 r_{m-1} 为终点的路径为 $t=r_0r_1...r_{m-1}$,则 $w_0w_1...w_{m-2}\in C(t)$,由引理 2.2 可知,存在 $t'=t'_0\ t'_1...t'_{m-1}\in R(A_1\otimes A_2)$且 $t'_0\in I(A_1\otimes A_2)$使得 $w_0w_1...w_{m-2}\in C(t')$且 $red(r_{m-1})=red(t'_{m-1})$.由于 $r=t...r^*$上存在迁移$(r_{m-1},u_{m-1},r_m)\in \Delta(A_1\cap A_2)$,又因为 $red(r_{m-1})\notin SCC(A_1)_l,red(r_m)\in \mathrm{SCC}(A_1)_l$,由于 $red(r_{m-1})=red(t'_{m-1})$,根据 $A_1\otimes A_2$ 的定义,在 $A_1\otimes A_2$ 中存在迁移$(t'_{m-1},u_{m-1},(red(r_m),q_0))$,因此,我们可以将路径 t'和路径 r'^* 通过该迁移连接起来,得到路径 $r'=t'...r'^*\in R(A_1\otimes A_2)$.由于 $w_0w_1...w_{m-2}\in C(t'),w^*\in C(r'^*)$,且在 r'上存在迁移 $t'_{m-1},u_{m-1},(red(r_m),q_0))$,则 $w=w_0w_1...w_{m-2}.u_{m-1}.w^*\in C(r')$,又因为 $inf(r'^*)\cap F(A_1\otimes A_2)\neq\varnothing$,所以,$inf(r')\cap F(A_1\otimes A_2)\neq\varnothing$.即 r'是 $A_1\otimes A_2$ 上的一条可接受路径.故 $w\in L(A_1\otimes A_2)$.

(2) 由引理 2.3、引理 2.8 可证明:对任意可接受字符串 $w\in L(A_1\otimes A_2)$,则 $w\in L(A_1\cap A_2)$.证明思路与(1)的证明类似(略).综合(1)、(2),定理 2.1 成立. □

3 实验结果

我们用C语言实现了在第2节中定义的$A_1\otimes A_2$运算,并将该转换方法记为⊗算法.该算法中所需要的TGBA是由 LTL2BA[7]工具产生的,该工具的主要特点是在 LTL 公式转换为 Büchi 自动机的过程中引入了 Very Weak

Alternating Co-Büchi 自动机(VWAA)作为一个中间转换结果.具体过程是:首先,将 LTL 公式在线性时间内转换为 VWAA 自动机;然后,再将该 VWAA 转换为 TGBA;最后,由 TGBA 转化为 BA.同时,由于采用了 on-the-fly 的化简方法,因而可以节约大量的内存和计算时间.LTL2BA 与其他工具(例如 spin[11])作比较时,在计算时间以及所得自动机的状态和迁移个数上都具有明显的优势[7].我们的实验工作主要是与 LTL2BA 进行比较.测试环境为:PC 机,内存 512M,CPU2.0G.第 1 类测试是针对形式为 $\varphi_n=p_1\mathrm{U}p_2\mathrm{U}p_3\mathrm{U}\ldots\mathrm{U}p_{n+1}$ 的公式,由于在实际检测过程中常常需要验证这种类型的公式,所以我们选择了这种形式的公式.测试结果见表 1.测试结果表明:对这类公式,⊗ 算法相对于 LTL2BA 有明显的优势.

Table 1 Test formula φ_n

表 **1** 测试公式 φ_n

	LTL2BA (state)	⊗ (state)
φ_2	6	4
φ_3	15	8
φ_4	37	16
φ_5	89	32
φ_6	209	64
φ_7	481	129
φ_8	1 089	256

第 2 类测试是针对由工具 lbtt[12]产生的随机公式.我们根据公式的长度将这类测试分为 10,15,20,25,30 五组,每组测试 40 个公式,共计 200 个公式,测试的结果见表 2.测试的结果表明:对随机产生的公式,当公式较长时,⊗ 算法相对于 LTL2BA 有明显的优势,且其优势随着公式长度的增加而更加显著.

Table 2 Test random formula

表 **2** 测试随机公式

Size	LTL2BA (average state)	⊗ (average state)
10	5.375	5.15
15	15.125	13.425
20	49.075	37.275
25	86.85	57.1
30	270.45	157.15

4 结论和进一步的工作

针对模型检测中的空间爆炸问题,出现了各种缓解该问题的方法,例如符号化模型检测[13]、偏序规约(partial order)[14]、抽象[14]等.由于在基于自动机的 LTL 模型检测的方法中,从 LTL 公式到自动机的转换过程对缓解空间爆炸有很重要的意义,所以这方面有很多的相关工作[2−9].此外,直接利用 simulation[15,16]或是 bisimulation 方法来化简 Büchi 自动机也是一种重要的方法.

在本文中,我们根据自动机可接受语言的特点,重新定义了 TGBA 与 BA 的求交运算 ⊗,在求交过程中尽量减少状态的复制,因此得到一个从 TGBA 到 BA 的有效转换方法.根据 ⊗ 的定义以及对实验数据的分析可知:TGBA 自动机 A 中 SCC(A)状态个数与 A 总的状态个数之比越小,该方法的优点就越明显,所生成的自动机状态个数就越少.例如在表 1 中,公式 φ_n 的 TGBA 只含有一个强连通环,且该环仅有一个状态,所以我们可获得非常好的效果.反之,效果就不明显,特别是在最坏情况下,当所有 A 的状态均处在一个 SCC(A)之中时,⊗ 算法的结果就与 ∩ 算法的结果具有相同的状态个数.

在第 2 节介绍如何构造 $FBA(A)$时,我们提到了可接受集元素排序的问题,即在某些情况下,当我们对接受集合 F 使用不同排列顺序时,所得 Büchi 自动机有可能具有不同的状态个数.这主要是因为如果每一个 $SCC(A)_i$ 环里的迁移顺序与所构造的 Büchi 自动机的迁移顺序相同,则得到 BA 的状态个数就最少,否则就可能有冗余的状态产生.简单地讲,我们可以对 $F(A)$的 N 个元素进行 $N!$的全排列,然后按照每一种排列方式进行构造和转换,从而找到可获得最少状态个数的排列方式.但是,当 SCC(A)中的状态数目较多且 N 值较大时,程序会需要大量的计算时间.因此,如何在时间复杂度较低的情况下找到最佳排列顺序,还需要作进一步的研究.

References:

[1] Vardi MY, Wolper P. An automata-theoretic approach to automatic program verification. In: Proc. of the LICS'86. Cambridge: IEEE CS Press, 1986. 332−344.

[2] Gerth R, Peled D, Vardi MY, Wolper P. Simple on-the-fly automatic verification of linear temporal logic. In: Proc. of the 15th IFIP/WG6.1 Symp. on Protocol Specification Testing and Verification. Warsaw, 1995. 3−18.

[3] Daniele M, Giunchiglia F, Vardi MY. Improved automata generation for linear temporal logic. In: Halbwachs N, Peled D, eds. Proc. of the 11th Int'l Computer Aided Verification Conf. LNCS 1633, Heidelberg: Springer-Verlag, 1999. 249−260.

[4] Somenzi F, Bloem R. Efficient Büchi automata for LTL formulae. In: Leeuwen JV, ed. Proc. of the 12th Int'l Conf. on Computer Aided Verification. LNCS 855, Heidelberg: Springer-Verlag, 2000. 247−263.

[5] Sebastiani R, Tonetta S. More deterministic vs. smaller Büchi automata for efficient LTL model checking. In: Geist D, Tronci E, eds. Proc. of the 12th Advanced Research Working Conf. on Correct Hardware Design and Verification Methods. LNCS 2860, Heidelberg: Springer-Verlag, 2003. 126−140.

[6] Giannakopoulou D, Lerda F. From states to transitions: Improving translation of LTL formulae to Büchi automata. In: Budach L, ed. Proc. of the 22nd IFIP WG 6.1 Int'l Conf. on Formal Techniques for Networked and Distributed Systems. LNCS 529, Heidelberg: Springer-Verlag, 2002. 308−326.

[7] Gastin P, Oddoux D. Fast LTL to Büchi automata translation. In: Berry G, Comon H, Finkel A, eds. Proc. of the 13th Int'l Conf. on Computer Aided Verification. LNCS 2102, Heidelberg: Springer-Verlag, 2001. 53−65.

[8] Couvreur J. On-the-Fly verification of temporal logic. In: Wing JM, Woodcock J, Davies J, eds. Proc. of the World Congress on Formal Methods in the Development of Computing Systems. LNCS 1708, Heidelberg: Springer-Verlag, 1999. 253−271.

[9] Etessami K, Holzmann G. Optimizing Büchi automata. In: Palamidessi C, ed. Proc. of the 11th Int'l Conf. on Concurrency Theory. LNCS 1877, Heidelberg: Springer-Verlag, 2000. 153−167.

[10] Fritz C. Constructing Büchi automata from linear temporal logic using simulation relations for alternating Büchi automata. In: Ibarra OH, Dang Z, eds. Proc. of the 8th Int'l Conf. on Implementation and Application of Automata. LNCS 2759, Heidelberg: Springer-Verlag, 2003. 35−48.

[11] Holzmann GJ. The model checker SPIN. IEEE Trans. on Software Engineering, 1997,23(5):279−295.

[12] Tauriainen H. lbtt: Automated testing of Büchi automata translators for linear temporal logic. Research Report A66. Espoo: Helsinki University of Technology, Laboratory for Theoretical Computer Science, 2000.

[13] Burch J, Clarke E, McMillan K, Dill D, Hwang W. Symbolic model checking: 10^{20} states and beyond. In: Proc. of the LICS'90. Philadelphia: IEEE CS Press, 1990. 428−439.

[14] Clarke E, Grumberg O, Peled DA. Model Checking. The MIT Press, 1999.

[15] Henzinger T, Kupferman O, Rajamani S. Fair simulation. In: Antoni M, Jozef W, eds. Proc. of the 9th Int'l Conf. on Concurrency Theory (CONCUR'97). LNCS 1243, Heidelberg: Springer-Verlag, 1997. 273−287.

[16] Etessami K, Wilke T, Schuller A. Fair simulation relations, parity games, and state space reduction for Büchi automata. In: Orejas F, Spirakis PG, Leeuwen JV, eds. Proc. of the Automata, Languages and Programming: 28th Int'l Colloquium. LNCS 2076, Heidelberg: Springer-Verlag, 2001. 694−707.

易锦(1976－),女,四川都江堰人,博士生,主要研究领域为形式化方法,自动机,模型检测.

张文辉(1963－),男,博士,研究员,CCF 高级会员,主要研究领域为程序正确性,模型检测,逻辑推理,形式化方法.