ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.18, No.6, June 2007, pp.1270−1281
DOI: 10.1360/jos181270

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

并发反应式系统的组合模型检验与组合精化检验*

文艳军[1], 王 戟[1,2+], 齐治昌[1]

[1](国防科学技术大学 计算机学院,湖南 长沙 410073)

[2](并行与分布处理国家重点实验室,湖南 长沙 410073)

Compositional Model Checking and Compositional Refinement Checking of Concurrent Reactive Systems

WEN Yan-Jun[1], WANG Ji[1,2+], QI Zhi-Chang[1]

[1](School of Computer, National University of Defense Technology, Changsha 410073, China)

[2](National Laboratory for Parallel and Distributed Processing, Changsha 410073, China)

+ Corresponding author: Phn: +86-731-4573637, E-mail: wj@nudt.edu.cn

Wen YJ, Wang J, Qi ZC. Compositional model checking and compositional refinement checking of concurrent reactive systems. *Journal of Software*, 2007,18(6):1270−1281. http://www.jos.org.cn/1000-9825/18/1270.htm

Abstract: Model checking and refinement checking are two approaches to formal verification, whose difficulties are due to the state explosion problem. As one of the proposed solutions to the problem, it is suggested to introduce compositionality in model checking and refinement checking based on the idea of divide-and-conquer, by which the verification task of the whole system is decomposed to several smaller subtasks on the subsystems. In a uniform framework, this paper surveys the approaches of compositional model checking and compositional refinement checking in a systematic way. From the perspective of module checking, the principle and verification strategies of the two compositional verification approaches are introduced. In addition, the complexities of various kinds of related problems are summarized and a comparison is made between compositional model checking and compositional refinement checking, which exposes the intrinsic relation between them. Finally, some trends are given for the future research.

Key words: model checking; refinement checking; compositional model checking; compositional refinement checking; state explosion problem; module checking

摘 要: 模型检验和精化检验是两种重要的形式验证方法,其应用的主要困难在于如何缓解状态爆炸问题.基于分而治之的思想进行组合模型检验和组合精化检验是应对这个问题的重要方法,它们利用系统的组合结构对问题

* Supported by the National Natural Science Foundation of China under Grant Nos.60233020, 60673118, 90612009 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2005AA113130, 2006AA01Z429 (国家高技术研究发展计划(863)); the National Basic Research Program of China under Grant No.2005CB321802 (国家重点基础研究发展计划(973)); the Program for New Century Excellent Talents in University of China under Grant No.NCET-04-0996 (新世纪优秀人才支持计划)

Received 2004-12-30; Accepted 2006-07-05

进行分解,通过对各子系统性质的检验和综合推理导出整个系统的性质.在一个统一的框架下对组合模型检验和组合精化检验作了系统的分析和归纳,从模块检验的角度阐述了上述两种组合验证方法的原理及其相应的组合验证策略.同时总结了各类问题的复杂性,并对上述两种方法作了比较分析,揭示了它们之间的内在联系.最后展望了组合模型检验与组合精化检验的发展方向.

关键词: 模型检验;精化检验;组合模型检验;组合精化检验;状态爆炸问题;模块检验
中图法分类号: TP301 文献标识码: A

反应式系统(reactive system)是指与环境有着持续不断交互的系统[1],典型的例子有嵌入式系统、操作系统和 Web 服务器等.它是与转换式系统(transformational system)相对照而言的,转换式系统的工作模式为:先接受输入,然后进行计算,最后将结果输出而终止.反应式系统则不是这样,它的行为是不终止的,这种不终止的行为一般用ω自动机(无穷字上的自动机)[2]或者时序逻辑来描述.

(逻辑)模型检验(logical model checking)是一种验证安全攸关反应式系统的关键性质的自动技术[3,4].其中,系统的行为用有穷状态的 Kripke 结构(可看作一种ω自动机)来描述,而系统的性质则用时序逻辑公式描述.这一方法已经在一些硬件设计和通信协议领域得到了成功的应用[5],目前的趋势是应用这一技术来分析软件系统.

精化检验(refinement checking)是另一种验证系统正确性的方法[6],与模型检验用逻辑公式来描述性质不同,在精化检验中,系统的行为及其应满足的性质都用类似ω自动机的某种形式来描述,通过比较行为自动机是否与性质自动机一致来判断系统的行为是否满足性质规约的要求.

模型检验的优点在于它可以完全自动化,不需要用户有高深的计算机理论专业知识.其难点在于,如何处理状态爆炸问题(state explosion problem).当系统数据结构比较复杂或者当系统中存在多个并发的子系统时,整个系统的状态空间常常出现指数级爆炸式的增长,导致计算所需资源(时间或空间)超出计算机的能力范围,从而使得该方法实际上不可行.组合模型检验是应对状态爆炸问题的一种方法,其基本思想是:利用系统的组合结构对问题进行分解,先检验各子系统的性质,然后综合推理导出整个系统的性质.这种分而治之的方法是降低模型检验复杂度的有效手段.同时,组合模型检验也是一个好的验证方法学的需要:好方法应该支持从子系统的性质导出系统整体的性质[7].类似地,精化检验也面临状态爆炸问题,也可以采用组合的方法来应对,称为其组合精化检验.

组合模型检验和组合精化检验的验证策略主要有两种:简单组合策略和假设/保证(assume/guarantee)组合策略[8,9](简称 A/G 组合策略),其基本模式分别如式(1)、式(2)所示.

$$\left.\begin{array}{c}\langle\text{true}\rangle M_1\langle\alpha\rangle\\ \langle\text{true}\rangle M_2\langle\beta\rangle\end{array}\right\}\Big/\ M_1\|M_2\ \text{sat}\ \alpha\ \text{and}\ \beta \tag{1}$$

$$\left.\begin{array}{c}\langle\text{true}\rangle M_1\langle\alpha\rangle\\ \langle\alpha\rangle M_2\langle\beta\rangle\end{array}\right\}\Big/\ M_1\|M_2\ \text{sat}\ \alpha\ \text{and}\ \beta \tag{2}$$

在使用简单组合策略时,先验证各子系统在任意环境下都满足的性质,然后从这些已验证的性质出发推导出系统的整体性质.例如在式(1)中,为了验证 M_1 和 M_2 的并发组合满足性质α和β,只需证明 M_1 在任意环境下都满足性质α(即$\langle\text{true}\rangle M_1\langle\beta\rangle$),$M_2$ 在任意环境下都满足性质β即可.这种验证一个模块是否在任意环境下都满足某一性质的问题称为模块检验(module checking)[10].使用简单组合策略时没有考虑模块之间的相互影响,因而许多的实际问题无法通过这种策略获得解决.

在这一背景下,人们提出了 A/G 组合策略.如式(2)所示,在采用 A/G 组合策略时,为了验证 M_1 和 M_2 的并发组合满足性质α和β,只需先验证 M_1 在任意环境下都满足性质α,然后再验证 M_2 在任意满足性质α的环境下都满

足性质β(即$\langle\alpha\rangle M_2\langle\beta\rangle$)即可.这种验证一个模块是否在满足某一性质的所有环境下都满足另一性质的问题被称为模块化模型检验(modular model checking)[7].为统一起见,我们称其为假设/保证模块检验(简称 A/G 模块检验),而将前一种模块检验称为简单模块检验.

简而言之,简单模块检验是判断形如$\langle true\rangle M_2\langle\beta\rangle$的命题的真假;而 A/G 模块检验是判断形如$\langle\alpha\rangle M_2\langle\beta\rangle$的命题的真假.在本文第 2 节和第 3 节中将给出其形式定义.在上述两种策略中,性质的描述方式既可以是ω自动机也可以是时序逻辑,因此,这两种策略既可以用于组合模型检验,也可以用于组合精化检验.相应地,我们称组合模型检验中的模块检验为时序性质的模块检验;而称组合精化检验中的模块检验为精化关系的模块检验.

本文的目的是对组合模型检验和组合精化检验进行归纳和分析,比较系统地阐述简单模块检验和 A/G 模块检验的原理以及相应的组合验证策略,并展望相关研究的发展趋势.

在组合精化检验中,主要考虑两种精化关系:线性(linear)精化和分支(branching)精化.线性精化关系的实质是指两个模块的路径(trace)集合间的包含关系,而分支精化关系的实质是指两个模块的路径树(trace-tree)集合间的包含关系.这两类精化关系的模块检验(包括简单模块检验和 A/G 模块检验)与模型检验是一致的,不需要特别的转换.分支精化关系的模块检验比线性精化关系的模块检验要容易.

在组合模型检验中,主要考虑线性时序逻辑 LTL,分支时序逻辑∀CTL 和∀CTL*.其中,∀CTL 和∀CTL*分别是 CTL 和 CTL*的只包含全称路径模态算子的片断[7].这一方面的研究主要集中在∀CTL*的逻辑范围内.∀CTL*性质的简单模块检验与模型检验是一致的;当做了一定的约束以后,LTL 性质的 A/G 模块检验与模型检验也是一致的,不需要特别的处理.但是,∀CTL*性质的 A/G 模块检验需要计算极大模型,其复杂度是双指数时间的.即便限制在∀CTL 的范围内,其复杂度也是指数时间的(见第 3.3 节).

组合模型检验和组合精化检验两者有着紧密的联系.通过构造极大模型,很多∀CTL*范围内的时序性质的模块检验问题都可以转化为精化关系的模块检验问题.另外,当做了一定的条件限制以后,二者都支持循环 A/G 组合验证策略.总之,基于分支精化关系进行组合精化检验相对比较容易,基于∀CTL 和 LTL 进行组合模型检验的复杂度比较高,而基于∀CTL*进行组合模型检验的复杂度相当高.在∀CTL 和 LTL 两者中,使用后者进行组合模型检验可能更合适一些.

本文第 1 节介绍组合验证的基本概念与框架.第 2 节和第 3 节分别介绍组合精化检验和组合模型检验.第 4 节对二者作比较分析和讨论.最后是对本文的小结和对未来发展趋势的展望.

1 组合验证的概念与框架

组合模型检验和组合精化检验研究的对象是由多个开放模块组合而成的封闭系统,通过对各个模块的研究来推断系统整体的性质.开放模块可用形式模型公平模块(fair module)[7]来描述,所有公平模块构成的集合记为 $\boldsymbol{M}$.

定义 **1**(公平模块(**fair modules**)).公平模块(简称模块)是六元组 $M=\langle S,S_0,A,\Lambda,R,\Phi\rangle$,其中:

- S 为一个有穷状态集合.
- $S_0\subseteq S$,是一个初始状态集合.
- A 是一个有穷的原子命题集合.
- Λ是一个类型为 $S\to 2^A$ 的标记函数,它给出每个状态下为真的原子命题的集合.
- $R\subseteq S\times S$,是一个迁移关系.
- $\Phi\subseteq 2^{S\times S}$,是一组状态对构成的集合,称为 M 的 Streett 公平约束条件.

一个无穷状态序列$\pi=s_0s_1s_2\ldots\in S^\omega$称为是 M 的一次运行(run)当且仅当 $s_0\in S_0$并且对于所有的 $i\in N,(s_i,s_{i+1})\in R$.一个无穷序列$\gamma=\gamma_0\gamma_1\gamma_2\ldots\in(2^A)^\omega$称为是 M 的一条路径(trace)当且仅当存在 M 的一次运行$\pi=s_0s_1s_2\ldots$使得对于所有的 $i\in N,\gamma_i=\Lambda(s_i)$;在这种情况下,我们称运行$\pi$见证路径$\gamma$.记由$\pi$中无穷多次出现的状态构成的集合为 $\inf(\pi)$.运行π称为是公平的(fair)当且仅当它满足 Streett 公平约束条件,即对于所有的$(P,Q)\in\Phi$,如果 $\inf(\pi)\cap P\neq\emptyset$,则$\inf(\pi)\cap Q\neq\emptyset$;路径$\gamma$称为是公平的当且仅当存在一次公平的运行见证它.Streett 公平约束条件有两种特例[6]:空约

束条件和 Büchi 约束条件.前者认为所有的运行都是公平的;后者仅用一个集合 $\Phi_B \subseteq S$ 来指定公平的运行,其含义为:运行 π 是公平的当且仅当 $\inf(\pi)\cap\Phi_B\neq\varnothing$.

模块的原子命题集也可看作事件集,通过它,多个模块之间可以相互通信,通信的方式类似于 CSP 中进程之间的交互,所不同的是,前者采用同步语义,多个事件允许同时发生,而后者采用异步语义,同一时间只允许一个事件发生.

定义 2(公平模块的组合(**composition of fair modules**)). 设 M 和 M' 为两个公平模块,则它们的组合是如下定义的公平模块 M'',记为 $M||M'$:

- $S''=\{(s,s')|\Lambda(s)\cap A'=\Lambda(s')\cap A\}$.
- $S_0''=(S_0\times S_0')\cap S''$.
- $A''=A\cup A'$.
- $\Lambda''((s,s'))=\Lambda(s)\cup A'(s')$.
- $R''((s,s'),(t,t'))$当且仅当 $R(s,t)$和 $R'(s',t')$.
- $\Phi''=\{((P\times S')\cap S'',(Q\times S')\cap S'')|(P,Q)\in\Phi\}\cup\{((S\times P')\cap S'',(S\times Q')\cap S'')|(P',Q')\in\Phi'\}$.

一个模块可以通过它的事件集(即原子命题集)与环境交互,当系统中只有该模块而没有其他模块时,我们称该模块处于最大环境下.此时,该模块本身构成一个闭系统,可以将其看作一个带公平约束的 Kripke 结构[6],因此可在其上定义时序逻辑 LTL,CTL 和 CTL*的语义.该语义与在经典 Kripke 结构上定义的语义的不同之处在于:加上公平约束后,所有的路径表达式针对的都只是公平路径,具体细节可参见文献[7,11].如果一个模块 M 在最大环境下满足性质 φ,则称 M 为 φ 的一个模型,记为 $M|=\varphi$.判断一个模块是否为某一性质的模型的过程称为模型检验(model checking).

模块之间的精化关系刻画了模块在最大环境下的行为性质,它可以从两个角度来定义:线性精化和分支精化.线性精化对一个模块在最大环境下的公平路径的范围作了刻画,其定义为:设 M 和 M' 为两个模块且 $A\supseteq A'$,则 M 线性精化 M',记为 $M\leq_L M'$,当且仅当 M 的每一个公平路径在 A' 上的投影都是 M' 的公平路径.分支精化则对一个模块在最大环境下的整体行为特征进行了刻画,它是通过模块之间的模拟关系(simulation relations)[6,7,11]来定义的.针对公平模块的模拟关系有多种[6,12],这里,我们采用基于博弈(game)思想定义的模拟关系[6],因为它有着比较好的性质和较低的计算复杂性.

先介绍策略(strategy)的概念.模块 M' 针对模块 M 的一个策略 τ 是类型为 $(S\times S')^*\times S\mapsto S'$ 的一个部分函数,其直观含义为:如果到目前为止的博弈的轨迹为 $\lambda\in(S\times S')^*$,并且 M 的下一步迁移的目标状态为 s,那么在策略 τ 的指导下,M' 将迁移到状态 $s'=\tau(\lambda,s)$,从而使系统到达新的状态 (s,s').设 $\pi=s_0s_1s_2\ldots$ 为 M 的一次运行,则在策略 τ 的指导下,M' 对应于 π 的运行为 $\pi'=s_0's_1's_2'\ldots$,其中 $s_i'=\tau(s_0s_0's_1s_1'\ldots s_{i-1}s_{i-1}',s_i)$ $(i\in N)$.

定义 3(模拟关系(**simulation relations**)). 设 M 和 M' 为两个公平模块且 $A\supseteq A'$,设 $H\subseteq S\times S'$ 为一个二元关系,则 H 为一个从 M 到 M' 的模拟关系当且仅当下列条件成立:

1. 对于任意 $t_0\in S_0$,存在一个状态 $t_0'\in S_0'$ 使得 $H(t_0,t_0')$.
2. 存在一个模块 M' 针对模块 M 的策略 τ,使得对于任意 $(s,s')\in H$,下列命题成立:
   a) $\Lambda(s)\cap A'=\Lambda(s')$.
   b) 任给一个 M 的从 s 开始的公平运行 $\pi=s_0s_1s_2\ldots(s_0=s)$,设 $\pi'=s_0's_1's_2'\ldots$ 为 M' 在策略 τ 的指导下对应于 π 的运行,即 $\pi'=\tau(\pi)$,那么 π' 是 M' 的一次公平运行且 $s_0'=s'$,同时,对于每一个 $i\in N$ 都有 $H(s_i,s_i')$.

模块 M 分支精化模块 M',记为 $M\leq M'$,当且仅当存在一个从 M 到 M' 的模拟关系.

在本文中,我们采用一个统一的过程框架来分析组合精化检验与组合模型检验.这个框架分为两个部分:其一,解决组合验证的两类基本问题,即简单模块检验问题和 A/G 模块检验问题;其二,按某种组合验证策略进行验证.基本的验证策略已在上一节讨论,两类基本问题的模式定义如下:

$$\langle\text{true}\rangle M\langle\alpha\rangle=_{\text{df}}\forall E\in\boldsymbol{M}.M||E \text{ sat } \alpha,$$
$$\langle\alpha\rangle M\langle\beta\rangle=_{\text{df}}\forall E\in\boldsymbol{M}.(E \text{ sat } \alpha)\rightarrow(M||E \text{ sat } \beta).$$

其中,sat 表示模型与规范之间的一种满足关系:对于组合精化检验来说,它表示一种精化关系;对于组合模型检

验来说,它表示模型与逻辑性质之间的满足关系.在接下来的两节中,这一模式将被具体化.

2 组合精化检验

组合精化检验首先要解决两个问题:精化关系的简单模块检验问题和 A/G 模块检验问题.精化关系的简单模块检验问题指的是如何判定形如$\langle \text{true}\rangle M\langle M'\rangle$的命题的真假,该命题的含义为:模块 M 在任意环境下都精化模块 M';而精化关系的 A/G 模块检验问题指的是如何判定形如$\langle M'\rangle M\langle M''\rangle$的命题的真假,该命题的含义为:如果环境精化 M',那么在这个环境下,M 一定精化 M''.以分支精化的模块检验为例,其形式定义如下:

$$\langle \text{true}\rangle M\langle M'\rangle =_{\text{df}} \forall E\in \boldsymbol{M}.M\|E\le M',$$
$$\langle M'\rangle M\langle M''\rangle =_{\text{df}} \forall E\in \boldsymbol{M}.(E\le M')\to(M\|E\le M'').$$

2.1 分支精化

公平模块之间的分支精化关系具有如下定理所述的良好性质[12].

定理 **1**. 下列命题成立:

(1) $\le$是一个前序关系.

(2) 对于所有的模块 M 和 M',$M\|M'\le M$.

(3) 对于所有的模块 M,M'和 M'',如果 $M\le M'$,那么 $M\|M''\le M'\|M''$.

(4) 对于所有的模块 M,$M\le M\|M$.

因为有着上述性质,所以公平模块的分支精化关系的简单模块检验问题和 A/G 模块检验问题都容易解决,易证:

$$\langle \text{true}\rangle M\langle M'\rangle \text{ iff } M\le M',$$
$$\langle M'\rangle M\langle M''\rangle \text{ iff } M\|M'\le M''.$$

也就是说,上述两类问题都可以转化为简单的精化检验问题,并不会带来额外的负担.

在解决了上述精化关系的简单模块检验和 A/G 模块检验问题的基础上,可用简单组合规则和 A/G 组合规则进行精化检验,其基本形式分别见式(3)、式(4).

$$\left.\frac{M_1\le M_1'\quad M_2\le M_1'}{M_1\|M_2\le M_1'\|M_2'}\right\} \tag{3}$$

$$\left.\frac{M_1\le M_1'\quad M_1'\|M_2\le M_2'}{M_1\|M_2\le M_1'\|M_2'}\right\} \tag{4}$$

基于上述规则,可以将一个大的精化检验任务分解为几个小的任务,以降低复杂度.例如在使用式(4)时,因为 M_1' 一般要比 M_1 简单,所以计算 $M_1'\|M_2$ 的代价比计算 $M_1\|M_2$ 要小.

Moore 机[9,11,13]和反应式模块(reactive modules)[6,14]是两种采用同步语义的高层形式模型,其上定义的组合操作、线性精化和分支精化与在公平模块上对应的定义都是一致的,可以将它们看作公平模块的特例,因此,上述计算两类模块检验问题的方法和上述两种组合规则对 Moore 机和反应式模块都成立.此外,Rajamani 等人证明了当满足“非阻塞(nonblocking)”和“内部不确定性有穷(finite internal nondeterminism)”两个条件时,在上述两种高层模型的框架内还可以使用式(5)、式(6)所示的循环 A/G 组合规则[6,9].

$$\left.\frac{M_1\|Safe(M_2')\le M_1'\quad M_1'\|M_2\le M_2'}{M_1\|M_2\le M_1'\|M_2'}\right\} \tag{5}$$

$$\left.\begin{array}{c} M_1 \| M_2' \le M_1' \\ M_1' \| M_2 \le M_2' \\ \hline M_1 \| M_2 \le M_1' \| M_2' \end{array}\right\} \tag{6}$$

其中, $Safe(M_2')$ 是指将 M_2' 的 Streett 公平约束条件替换为空约束条件后得到的模块.当考虑的都是具有空公平约束条件的模块时,上述循环 A/G 组合规则(式(5))具有更简洁的形式,如式(6)所示.

2.2 线性精化

分支精化是对线性精化的增强,因此,如果两个模块具有分支精化关系,那么它们之间一定也具有线性精化关系.所以,可以推知定理 1 中(1),(2),(4)这 3 个命题对于线性精化同样成立.下一引理说明了定理 1 中的命题(3)对于线性精化也成立.

引理 **1**. 对于所有的模块 M,M'和 M'',如果 $M\le_L M'$,那么 $M\|M''\le_L M'\|M''$.

证明:设 $u=(s_0,s_0'')(s_1,s_1'')\ldots$ 为 $M\|M''$的一次公平运行,其见证的路径记为λ_u.因为 $M\le_L M'$,所以 $A\supseteq A'$,$(A\cup A'')\supseteq(A'\cup A'')$,故只需证明存在 $M'\|M''$的一次公平运行,其见证的路径等于λ_u 在 $M'\|M''$的原子命题集上的投影.令$\pi=s_0s_1\ldots$, $\pi''=s_0''s_0''\ldots$,由文献[11]中的引理 1 可知,π和π''分别是 M 和 M''的一次公平运行.因为 $(s_i,s_i'')(i\in N)$ 是 $M\|M''$的状态,所以 $\Lambda(s_i)\cap A''=\Lambda''(s_i'')\cap A$.因为 $M\le_L M'$,所以存在 M'的一次公平运行 $\pi'=s_0's_1'\ldots$ 使得 $\Lambda(s_i)\cap A'=\Lambda'(s_i')$.所以,

$$\Lambda'(s_i')\cap A''=\Lambda(s_i)\cap A'\cap A''=\Lambda''(s_i'')\cap A\cap A'=\Lambda''(s_i'')\cap A',$$

故 (s_i',s_i'') 是 $M'\|M''$的一个状态.又因为π'和π''分别是 M'和 M''的公平运行,所以根据文献[11]中的引理 1 知,$v=(s_0',s_0'')(s_1',s_1'')\ldots$ 是 $M'\|M''$的公平运行.考虑到

$$\begin{aligned}(\Lambda(s_i)\cup\Lambda''(s_i''))\cap(A'\cup A'') &= (\Lambda(s_i)\cap A')\cup(\Lambda(s_i)\cap A'')\cup(\Lambda''(s_i'')\cap A'')\\ &= \Lambda'(s_i')\cup(\Lambda''(s_i'')\cap A)\cup(\Lambda''(s_i'')\cap A'')\\ &= \Lambda'(s_i')\cup\Lambda''(s_i''),\end{aligned}$$

所以,v 见证的路径等于λ_u在 $M'\|M''$的原子命题集上的投影.得证. □

由此可见,对于线性精化来说,定理 1 依然成立.所以,相应地,式(3)、式(4)中的两种组合规则对于线性精化也成立.类似地,在 Moore 机和反应式模块上也可以定义线性精化关系[14−16],且它们与公平模块上定义的线性精化关系是一致的,因此,也可以使用与式(3)、式(4)中的组合精化检验规则.同时,当满足“非阻塞”和“内部不确定性有穷”两个条件时,在 Moore 机和反应式模块的框架内也可以使用式(5)、式(6)中的循环 A/G 组合规则[9,14,15].

2.3 计算复杂性

公平模块精化检验的计算复杂性见表 1,其中考虑的是模块 M 和 M'之间的精化关系,除了标注“PSPACE-complete”以外,都是关于时间复杂度的.

Table 1 Complexity of refinement checking of fair modules

表 **1** 公平模块精化检验的计算复杂性

	Empty constraint	Büchi constraint	Streett constraint
Linear refinement $M\le_L M'$	$O(\|M\|\times 2^{\|M'\|})$[17] PSPACE-Complete on $\|M'\|$	PSPACE-Complete[6]	PSPACE-Complete[6]
Branching refinement $M\le M'$[6]	$O(\|R\|\cdot\|S'\|+\|R'\|\cdot\|S\|)$	$O(\|R\|\cdot\|S'\|+\|R'\|\cdot\|S\|+(\|S\|\cdot\|S'\|)^3)$	$O(n^{(2f+1)}\cdot f!)$, $n=\|S\|\cdot\|S'\|\cdot(3^{\|\Phi\|}+\|\Phi'\|)$ $f=2\times\|\Phi\|+\|\Phi'\|$

由表 1 可知,线性精化检验的计算复杂性普遍较高,即便在空约束条件下,其在$|M'|$上的复杂性也是多项式空间完全的(PSPACE-complete).而分支精化检验的计算复杂性则较低:在空约束条件和 Büchi 约束条件下,其复杂性都是多项式时间的;在 Streett 约束条件下,其复杂性与模块的状态空间大小多项式时间相关,而与公平约束条件的大小指数时间相关,所以在 Streett 约束条件比较小时,其复杂性也是可以接受的.技术上,分支精化关系的检验是通过转换成树自动机(tree automata)之间的判空问题来解决的,而一个 Streett 树自动机的判空问题的时

间复杂性是 $O(n^{(2f+1)}\cdot f!)$,其中,n 为状态数,f 为公平约束条件中状态对的数目.

3 组合模型检验

类似地,组合模型检验首先也要解决两个问题:时序性质的简单模块检验和 A/G 模块检验.时序性质的简单模块检验指的是如何判定形如$\langle true\rangle M\langle\varphi\rangle$的命题的真假,该命题的含义为:模块 M 在任意环境下都满足性质φ;而时序性质的 A/G 模块检验指的是如何判定形如$\langle\varphi\rangle M\langle\psi\rangle$的命题的真假,该命题的含义为:如果环境满足性质$\varphi$,那么在此环境下,$M$ 一定也满足性质ψ.它们的形式定义如下:

$$\langle true\rangle M\langle\varphi\rangle =_{df} \forall E\in \boldsymbol{M}.M\|E|=\varphi,$$

$$\langle\varphi\rangle M\langle\psi\rangle =_{df} \forall E\in \boldsymbol{M}.(E|=\varphi)\rightarrow(M\|E|=\psi).$$

3.1 ∀CTL*和∀CTL性质

∀CTL*和∀CTL 分别是 CTL*和 CTL 的只包含全称路径模态算子的片断[7],它的模型检验与前面介绍的分支精化检验有着紧密的联系,定理 2[11,12]说明了这一点.

定理 2. 设模块 M 和 M'满足 $M\leq M'$,则对于任意的∀CTL*公式φ(其原子命题都属于 A'),如果 $M'|=\varphi$,那么 $M|=\varphi$.

此定理可用公式的结构归纳法进行证明,它说明分支精化关系保持∀CTL*性质,又因为模块的组合具有定理 1 中的性质(2),所以,对于任意模块 M 和∀CTL*公式φ(其原子命题都属于 A)来说,下列命题成立:

$$\langle true\rangle M\langle\varphi\rangle \text{ iff } M|=\varphi.$$

所以,∀CTL*性质的简单模块检验问题可以转化为经典的模型检验问题.

∀CTL*性质的 A/G 模块检验问题要更复杂一些,需要引入极大模型(maximal model)[7,11,12]的概念.公平模块 M_φ是∀CTL*公式φ的对于分支精化来说的极大模型,当且仅当对于任意公平模块 M 来说,$M\leq M_\varphi$与 $M|=\varphi$是等价的.当从上下文来看“分支精化”比较清楚时,简称 M_φ是φ的极大模型.本节讨论的极大模型都是针对分支精化的,所以一般不强调这一点.下面先考虑∀CTL 性质的 A/G 模块检验.

每一个∀CTL 公式都有极大模型,定理 3[11,12]说明了这一点.

定理 3. 每一个∀CTL 公式φ都存在一个大小为 $2^{O(|\varphi|)}$的极大模型 M_φ.

该定理的证明要用到场景(tableau)[13]构造技术.对于任意∀CTL 公式φ和ψ,设 M_φ和 M_ψ分别为φ和ψ的极大模型,则由极大模型的性质易推知下列命题成立:

$$\langle true\rangle M\langle\psi\rangle \text{ iff } M|=\psi \text{ iff } \langle true\rangle M\langle M_\psi\rangle,$$

$$\langle\varphi\rangle M\langle\psi\rangle \text{ iff } M\|M_\psi|=\psi \text{ iff } \langle M_\varphi\rangle M\langle M_\psi\rangle.$$

由上述结论可知:∀CTL 性质的 A/G 模块检验问题也可以转化为经典的模型检验问题;∀CTL 性质的简单模块检验问题和 A/G 模块检验问题还可以分别转化为公平模块的分支精化关系的简单模块检验问题和 A/G 模块检验问题.

在解决了上述时序性质的简单模块检验和 A/G 模块检验问题的基础上,可用简单组合规则和 A/G 组合规则进行组合模型检验,其基本形式分别如式(7)、式(8)所示.

$$\left.\begin{array}{c} M_1|=\varphi \\ M_2|=\psi \\ \hline M_1\|M_2|=\varphi\wedge\psi \end{array}\right\} \tag{7}$$

$$\left.\begin{array}{c} M_1|=\varphi \\ M_2\|M_\varphi|=\psi \\ \hline M_1\|M_2|=\varphi\wedge\psi \end{array}\right\} \tag{8}$$

其中,式(7)中的φ和ψ可以是任意∀CTL*公式,而式(8)中的φ和ψ只能是∀CTL 公式,这是因为前面我们只论及

∀CTL 性质的 A/G 模块检验问题.至于∀CTL*性质的 A/G 模块检验问题,应该也可以按照类似的思路来解决.Kupferman 等人证明了文献[7]对于一种与前面的定义不同的分支精化关系来说,每一个∀CTL*公式都有一个极大模型,并提出了一种构造方法,但其构造的模型对于本文从博弈角度定义的分支精化关系来说是否也是极大模型尚需证实.

3.2 LTL性质

首先,我们针对线性精化定义 LTL 性质的极大模型.公平模块 M_φ是 LTL 公式φ的对于线性精化来说的极大模型,当且仅当对于任意公平模块 M,$M\leq_L M_\varphi$与 $M\models\varphi$是等价的.Vardi 等人证明了如下定理[18,19]:

定理 **4**. 任给一个 LTL 公式φ,存在一个状态数为 $2^{O(|\varphi|)}$的 Büchi 自动机 A_φ,其接受的语言$\Lambda(A_\varphi)$刚好是所有满足φ的计算的集合.

同时,他们还给出了一个 A_φ的构造算法.满足上述条件的 Büchi 自动机 A_φ可以简单地转换为一个带 Büchi 约束条件的公平模块 M_φ,且 M_φ的所有公平路径的集合等于$\Lambda(A_\varphi)$.易证,如此构造的 M_φ是公式φ的一个对于线性精化来说的极大模型.所以可知,每一个 LTL 公式都存在一个对于线性精化来说的极大模型.在此基础上,上一节中关于 A/G 模块检验的结论以及组合规则(式(7)、式(8))都可以平行推广到本节定义的极大模型上去.因为同一个模块的基于线性精化的极大模型一般要比它的基于分支精化的极大模型小,所以使用基于线性精化的极大模型可以更有效地进行组合模型检验.

另外,LTL 性质的某些特殊类型的 A/G 模块检验问题还有更简单的解决办法,下一定理[7]说明了这一点.

定理 **5**. 任给 LTL 公式φ,ψ和一个公平模块 M,如果φ和ψ中出现的所有原子命题都在 M 的原子命题集中,则$\langle\varphi\rangle M\langle\psi\rangle$当且仅当 $M\models\varphi\rightarrow\psi$.

由此定理可知,这类特殊的 A/G 模块检验问题可以直接转化为经典的模型检验问题(实际上是转化为一个蕴含式的模型检验),并不带来额外的负担.其对应的 A/G 组合规则的基本形式如下所示,Stark 研究了使用该规则进行推导的各种复杂模式[20].

$$\left.\begin{array}{c} M_1\models\varphi \\ \hline M_2\models\varphi\rightarrow\psi \\ \hline M_1\|M_2\models\varphi\wedge\psi \end{array}\right\} \tag{9}$$

在上述 LTL 性质的模块检验框架内,Assume 性质和 Guarantee 性质的关系是简单、直观的.因为对 Assume 性质不需要特殊的处理,所以,该组合验证过程实际上是一个逻辑推理的过程.但是,上述框架存在较大的缺陷:它无法表示 Assume 性质和 Guarantee 性质之间的循环依赖关系.举例来说,我们无法从 $M_1\models\psi\rightarrow\varphi$和 $M_2\models\varphi\rightarrow\psi$推导出 $M_1\|M_2\models\varphi\wedge\psi$.在 Stark 的研究中,为了避免这种错误的推导,禁止任何可能出现的循环依赖.然而,这种循环依赖的情况是比较普遍的.为此,需要改变 A/G 模块检验的定义,使用不同的组合模块检验框架.很多前期的组合模型检验研究都是关于这一方面的[21−24],它们大多采用这样一种思路来定义 A/G 模块检验问题$\langle\varphi\rangle M\langle\psi\rangle$:在系统运行的每个状态中,如果到前一状态为止,环境满足性质φ,那么到当前状态为止,M 一定满足性质ψ.另外,要求在初始状态下 M 满足性质ψ.在这个定义的基础上,可以将 Assume 性质和 Guarantee 性质之间的循环依赖关系转化为一个递归依赖关系,从而弥补前面提到的缺陷.典型的工作有:

Abadi 和 Lamport[22]通过给线性时序逻辑 TLA 新增一个时态算子来描述 Assume 性质和 Guarantee 性质间的递归依赖关系,并构造了相应的组合验证框架.类似地,为了建立一个一般的组合推理规则,Xu 等人[23]也引入了类似的时态算子来描述 Assume/Guarantee 规约.他们采用的都是对逻辑进行语义扩充的办法.与之不同的是,Jonsson 等人[24]从语法角度采用带过去算子的 LTL 逻辑描述 Assume/Guarantee 规约,并且建立了相应的组合验证规则.上述各种方案都可以对循环依赖进行建模和推理.Viswanathan 等人[25]对这类工作做了进一步的推广,从语义角度对不动点理论进行扩充,给出了一个更一般的框架,它既可以解释线性时序逻辑中的一些循环验证规则,也可以解释 Moore 机等模型上的循环 A/G 组合规则.Maier 等人[26]基于格理论提出了一个抽象框架来研究各种循环验证规则;Amla[27]等人也进行了类似的工作,其方法不仅是可靠的,而且也是语义上完全的.

3.3 计算复杂性

公平模块的时序性质的简单模块检验和 A/G 模块检验的计算复杂性见表 2.其中,简单模块检验考虑的问题是$\langle true\rangle M\langle\varphi\rangle$,A/G 模块检验考虑的问题是$\langle\varphi\rangle M\langle\psi\rangle$.同时也列出了经典的时序逻辑模型检验的复杂性,考虑的问题是 $M|=\varphi$.因为对于$\forall CTL^*$性质来说,其简单模块检验和经典的模型检验是一致的,所以在表中将它们列在同一行.具体来说,$\langle\forall CTL\rangle M\langle\forall CTL\rangle$指的是 Assume 性质和 Guarantee 性质都用∀CTL 逻辑表示时的 A/G 模块检验问题,其他式子的含义与此类似.需要指出的是,表中$\langle LTL\rangle M\langle LTL\rangle$这一行指的是满足定理 5 的约束条件的 A/G 模块检验问题的复杂度,而$\langle\forall CTL^*\rangle M\langle\forall CTL^*\rangle$这一行指的是当分支精化关系以一种不同的方式定义时该问题的复杂性[7](如第 3.1 节所述),这里一并列出.

从表中我们可以看出,除了$\langle true\rangle M\langle\forall CTL\rangle$型简单模块检验问题和 $M|=CTL$ 型模型检验问题的复杂度较低(多项式时间)以外,其他类型的问题都至少是指数时间复杂度的.特别地,$\langle\forall CTL^*\rangle M\langle\forall CTL^*\rangle$型模块检验问题的时间复杂性是双指数时间的,因此难以实用.

Table 2 Complexity of module checking of temporal logics

表 2 时序性质的模块检验的计算复杂性

	Time complexity	Space complexity
$\langle\forall CTL\rangle M\langle\forall CTL\rangle$	$\|M\|\times\|\psi\|\times 2^{O(\|\varphi\|)}$	$O(\|\psi\|\times(\log\|M\|+\|\varphi\|)^2)$ PSPACE-complete
$\langle\forall CTL^*\rangle M\langle\forall CTL^*\rangle$	$\|M\|\times 2^{O(\|\psi\|)+2^{O(\|\varphi\|)}}$	$O(\|\psi\|\times(\|\psi\|+\log\|M\|+2^{O(\|\varphi\|)})^2)$ EXPSPACE-complete
$\langle LTL\rangle M\langle LTL\rangle$	$\|M\|\times 2^{O(\|\varphi\|+\|\psi\|)}$	$O((\log\|M\|+\|\varphi\|+\|\psi\|)^2)$ PSPACE-complete
$M\|=LTL$ $\langle true\rangle M\langle LTL\rangle$	$\|M\|\times 2^{O(\|\varphi\|)}$	$O((\log\|M\|+\|\varphi\|)^2)$ PSPACE-complete
$M\|=CTL$ $\langle true\rangle M\langle\forall CTL\rangle$	$O(\|M\|\cdot\|\varphi\|)$ PTIME-complete	$O(\|\varphi\|\times\log^2\|M\|)$
$M\|=CTL^*$ $\langle true\rangle M\langle\forall CTL^*\rangle$	$\|M\|\times 2^{O(\|\varphi\|)}$	$O(\|\varphi\|\times(\log\|M\|+\|\varphi\|)^2)$ PSPACE-complete

4 比较和讨论

模型检验与精化检验之间有着密切的联系.首先,精化关系有着鲜明的逻辑特征:分支精化保持$\forall CTL^*$性质,而线性精化则保持 LTL 性质.同时我们不难发现,线性精化检验的复杂度与 LTL 性质的模型检验的复杂度相当,而分支精化检验的复杂度则与分支时序逻辑的模型检验的复杂度相当.具体来讲,对带空约束条件和 Büchi 约束条件的公平模块来说,其分支精化检验的复杂度与 CTL 的模型检验的复杂度相当;而对带 Streett 约束条件的公平模块来说,其分支精化检验的复杂度与 CTL^*的模型检验的复杂度相当.

同时,模块检验与模型检验和精化检验之间也有着密切关系.实际上,所有已解的模块检验问题(包括精化关系的和时序性质的)都是通过转化为模型检验问题和精化检验问题而获得解决的.所不同的是,有的问题转换得比较直接,而有的则需要构造极大模型(如$\langle\forall CTL\rangle M\langle\forall CTL\rangle$型和$\langle\forall CTL^*\rangle M\langle\forall CTL^*\rangle$型问题).通过比较可以发现,精化关系的模块检验、$\forall CTL^*$性质的简单模块检验和 LTL 性质的模块检验这 3 类问题与它们各自所对应的精化检验(或模型检验)问题的复杂性是一样的;而∀CTL 和$\forall CTL^*$性质的 A/G 模块检验问题比它们对应的模型检验问题的复杂性要高.

另外,从前面的分析可知,时序性质的模块检验与精化关系的模块检验之间也存在着密切的联系.∀CTL 性质的模块检验问题都可以转化为分支精化关系的模块检验问题,这是因为每一个∀CTL 公式都有一个对于分支精化来说的极大模型.LTL 的模块检验问题都可以转化为线性精化关系的模块检验问题,这是因为每一个 LTL 公式都有一个对于线性精化来说的极大模型.严格地说,目前时序性质模块检验的研究都集中在$\forall CTL^*$的范围内,且所有已解的时序性质的模块检验问题都可以通过转化为精化关系的模块检验问题而获得解决.从这个意义上说,在组合验证的背景下用模型来描述性质比用时序逻辑来描述性质更加本质.

总之,如果综合考虑两类模块检验问题可以得出以下结论:分支精化关系的模块检验相对容易(对带空约束

条件或 Büchi 约束条件的公平模块来说),线性精化关系的模块检验、分支时序逻辑∀CTL 的模块检验和线性时序逻辑 LTL 的模块检验三者的复杂性相当,而分支时序逻辑∀CTL*的模块检验最难.虽然∀CTL 和 LTL 的表达能力是不可比较的,但实际应用中往往发现后者的能力更强[7],因此,虽然∀CTL 的模块检验与 LTL 的模块检验的复杂性相当,但后者可能更实用一些.

在相关的综述文章中,文献[8]分析了包括接口进程(interface process)方法和假设/保证方法在内的 4 种组合验证方法;文献[28]除了分析上述两种方法外,还分析了循环验证;文献[29]则主要介绍了接口进程方法.虽然前两篇文章都论及了假设/保证方法,但它们只讨论了∀CTL 性质的组合模型检验问题,没有考虑 LTL 性质和∀CTL*性质,也没有系统地论述组合精化检验问题.本文集中考察了模型检验和精化检验中的组合方法.此外,在形式验证的其他领域也广泛使用了组合方法,这方面的工作可参见文献[30].

5 结论与展望

本文以组合模型检验和组合精化检验的两类基本问题——简单模块检验和 A/G 模块检验为线索,对并发反应式系统的这两类组合验证方法进行了系统的分析和归纳,着重论述了模块检验的基本原理和相应的组合验证策略.通过对这两类方法的比较分析,揭示了它们之间存在的本质联系.另外,对各种情况下的计算复杂性作了归纳总结.

这方面的一些开问题主要有:基于从博弈角度定义的分支精化关系如何计算∀CTL*性质的极大模型,如何判定⟨true⟩M⟨CTL⟩型和⟨true⟩M⟨CTL*⟩型命题的真假,等等.

本文前面重点论述了模块检验的方法以及相应的组合验证策略,它们是组合模型检验与组合精化检验的理论基础.另外,还有一些其他因素影响组合方法的成功应用.比如,在组合模型检验中如何将系统的整体性质进行分解就是一个技巧性很强的问题,在性质比较复杂时更是如此.为此,可以考虑定理证明与模型检验的集成:使用定理证明的工具和方法辅助进行性质的分解和推理,然后使用模型检验方法验证各个子任务.同时,定理证明与模型检验的集成也有助于对无穷状态系统的验证.另外,也可以考虑组合设计与组合验证的集成:当我们在组合设计的过程中将一个高层模块精化为一组低层模块时,可以同时对高层模块应满足的性质进行组合设计,就是将这些性质进行拆分以落实到各个低层模块上去;这样一来,在组合验证的时候就可以直接利用这些性质进行验证,无须再考虑拆分的问题.反过来看,组合验证的难易从某种程度上也反映了组合设计的好坏.也就是说,组合验证应与组合设计统筹考虑.

组合验证的另一个困难是构造环境模块:精化关系的 A/G 模块检验的使用前提是有一个合适的环境模块;而时序性质的 A/G 模块检验的使用前提是有一个合适的环境性质,并且还要将该性质转换为相应的环境模块(即极大模型).无论哪一种情况,环境模块的构造和处理都是比较复杂的.为此,可以考虑使用那些能将环境信息显式地表达到模块中去的形式模型,如接口自动机(interface automata)[31].在使用这类模型建模时,由于各个模块中已经包含了一部分环境信息,所以在组合验证中,更多的时候只需要使用简单组合规则就行了,从而减少了构造环境模块的频度.

此外,从表达能力上看,因为 LTL 严格弱于ω自动机,只能表达具有 star-free 属性的ω正规式所能刻画的性质,所以它不能描述那些需要涉及模块与环境之间的交互位置的假设性质[32].也就是说,LTL 不足以描述模块对环境的所有假设.为此,可以考虑采用表达力更强的逻辑,如 ETL[32],来进行组合模型检验.

为了应对软件的组合验证所带来的新的挑战,当前的一个发展方向是采用博弈论的观点来定义软件的语义模型和相应的规范语言[33,34].总之,组合验证是一种有效的形式验证方法,尽管还有许多工作要做,它的逐步实用化依然具有较好的前景.

References:

[1] Browne A, de Alfaro L, Manna Z, Sipma HB, Uribe TE. Diagram-Based formalisms for the verification of reactive systems. In: Proc. of the CADE-13: Workshop on Visual Reasoning. Springer-Verlag, 1996.

[2] Thomas W. Languages, automata, and logic. In: Rozenberg G, Salomaa A, eds. Handbook of Formal Languages, Vol. 3: Beyond Words. New York: Springer-Verlag, 1997. 389−455.

[3] Lin H, Zhang W. Model checking: Theories, techniques and applications. Chinese Journal of Electronics, 2002,30(12A): 1907−1912 (in Chinese with English abstract).

[4] Clarke EM, Schlingloff H. Model checking. In: Robinson A, Voronkov A, eds. Handbook of Automated Reasoning. Elsevier Science, 2001. 1635−1790.

[5] Clarke EM, Kurshan RP. Computer-Aided verification. IEEE Spectrum, 1996,33(6):61−67.

[6] Rajamani SK, Henzinger TA. New directions in refinement checking [Ph.D. Thesis]. Berkeley: University of California, 1999.

[7] Kupferman O, Vardi MY. An automata-theoretic approach to modular model checking. ACM Trans. on Programming Languages and Systems, 2000, 22(1):87−128.

[8] Berezin S, Ser C, Clarke EM. Compositional reasoning in model checking. LNCS 1536, 1998. 81−102.

[9] Henzinger TA, Qadeer S, Rajamani SK, Tasiran S. An assume-guarantee rule for checking simulation. ACM Trans. on Programming Languages and Systems, 2002,24(1):51−64.

[10] Vardi MY. Verification of open systems. In: Ramesh S, Sivakumar G, eds. Proc. of the FSTTCS 1997. Springer-Verlag, 1997. 250−266.

[11] Grumberg O, Long DE. Model checking and modular verification. ACM Trans. on Programming Languages and Systems, 1994,16(3):843−871.

[12] Bustan D, Grumberg O. Applicability of fair simulations. In: Katoen JP, Stevens P, eds. Proc. of the Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2002). Heidelberg: Springer-Verlag, 2002. 401−414.

[13] Long DE. Model checking, abstraction, and compositional verification [Ph.D. Thesis]. Pittsburgh: Carnegie Mellon University, 1993.

[14] Alur R, Henzinger TA. Reactive modules. In: Clarke E, ed. Proc. of the 11th Annual Symp. on Logic in Computer Science. IEEE Computer Society Press, 1996. 207−218.

[15] McMillan KL. A compositional rule for hardware design refinement. In: Grumberg O, ed. Proc. of the 9th Int'l Conf. on Computer Aided Verification. London: Springer-Verlag, 1997. 24−35.

[16] Kurshan RP. Computer-Aided Verification of Coordinating Processes. Princeton: Princeton University Press, 1994.

[17] Stockmeyer LJ, Meyer AR. Word problems requiring exponential time. In: Proc. of the 5th Annual ACM Symp. on Theory of Computing. ACM Press, 1973. 1–9.

[18] Gerth R, Peled D, Vardi MY, Wolper P. Simple on-the-fly automatic verification of linear temporal logic. In: Dembinski P, Sredniawa M, eds. Proc. of the 15th IFIP WG6.1 Int'l Symp. on Protocol Specification, Testing and Verification XV. Chapman & Hall, Ltd., 1996. 3−18.

[19] Vardi MY, Wolper P. Reasoning about infinite computations. Information and Computation, 1994,115(1):1−37.

[20] Stark EW. Foundations of a theory of specification for distributed systems. Technical Report, MIT-LCS-TR-342, Massachusetts Institute of Technology, 1984.

[21] Abadi M, Lamport L. Composing specification. ACM Trans. on Programming Languages and Systems (TOPLAS), 1993,15(1): 73−132.

[22] Abadi M, Lamport L. Conjoining specifications. ACM Trans. on Programming Languages and Systems (TOPLAS), 1995,17(3): 507−535.

[23] Xu Q, Cau A, Collette P. On unifying assumption-commitment style proof rules for concurrency. In: Proc. of the Int'l Conf. on Concurrency Theory. London: Springer-Verlag, 1994. 267−282.

[24] Jonsson B, Tsay YK. Assumption/Guarantee specifications in linear-time temporal logic. Theoretical Computer Science, 1996,167: 47−72.

[25] Viswanathan M, Viswanathan R. Foundations for circular compositional reasoning. In: Orejas F, Spirakis PG, Leeuwen JV, eds. Proc. of the ICALP 2001. Springer-Verlag, 2001. 835−847.

[26] Maier P. A lattice-theoretic framework for circular assume-guarantee reasoning [Ph.D. Thesis]. Saarbrücken: Universität Saarbrücken, 2003.

[27] Amla N, Emerson EA, Namjoshi KS, Trefler RJ. Abstract patterns of compositional reasoning. In: Amadio RM, Lugiez D, eds. Proc. of the 14th Int'l Conf. on CONCUR 2003—Concurrency Theory. Springer-Verlag, 2003. 423−438.

[28] Peng H, Tahar S. A survey on compositional verification. Technical Report, Department of Electrical and Computer Engineering, Concordia University, 1998.

[29] Clarke E, Long D, McMillan K. Compositional model checking. In: Parikh R, ed. Proc. of the 4th Annual Symp. on Logic in Computer Science. IEEE Press, 1989. 353−362.

[30] Furia CA. A compositional world: A survey of recent works on compositionality in formal methods. Technical Report, 2005.22, Dipartimento di Elettronica e Informazione, Politecnico di Milano, 2005.

[31] Alfaro LD, Henzinger TA. Interface automata. In: Gruhn V, ed. Proc. of the 9th Symp. Foundations of Software Engineering. ACM Press, 2001. 109−120.

[32] Kupferman O, Piterman N, Vardi M. Extended temporal logic revisited. In: Kim GL, Mogens N, eds. Proc. of the 12th Int'l Conf. on Concurrency Theory. Springer-Verlag, 2001. 519−535.

[33] Ghica DR. A games-based foundation for compositional software model checking [Ph.D. Thesis]. Kingston: Queen's University, 2002.

[34] Abramsky S, Ghica DR, Murawski AS, Ong CHL. Applying game semantics to compositional software modeling and verifications. In: Kurt J, Andreas P, eds. Proc. of the 10th Int'l Conf. on Tools and Algorithms for the Construction and Analysis of System (TACAS 2004). Springer-Verlag, 2004. 421−435.

附中文参考文献:

[3] 林惠民,张文辉.模型检测:理论、方法与应用.电子学报,2002,30(12A):1907−1912.

文艳军(1975－),男,湖北天门人,博士,主要研究领域为软件体系结构,组合模型检验.

王戟(1969－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为形式化方法,软件测试,软件可靠性.

齐治昌(1942－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件工程.