基于时态认知逻辑的Web服务模型检测

骆翔宇 陈 艳 古天龙 董荣胜
(桂林电子科技大学计算机与控制学院 桂林 541004)

摘 要 传统模型检测技术主要采用时态逻辑描述被验证的规范,人们较少注意多智能体认知逻辑的模型检测问题。而在分布式系统领域,系统和协议的规范很适合用认知逻辑来描述。Web服务是一个典型的分布式系统。把Web服务组合建模为多智能体系统,并成功采用我们实现的时态认知逻辑符号模型检测工具MCTK验证了SAS股票分析服务实例。同时采用WSAT,WS-Engineer和SPIN 3个模型检测工具在相同实验环境下验证了该实例,实验结果表明我们的Web服务模型检测方法不仅比这3个模型检测工具更高效,而且支持认知逻辑规范的验证,这是这3个模型检测工具所不具备的。

关键词 模型检测,时态认知逻辑,多智能体系统,Web服务
中图法分类号 TP301 **文献标识码** A

Model Checking Web Services Based on Temporal Logic of Knowledge

LUO Xiang-yu CHEN Yan GU Tian-long DONG Rong-sheng
(Department of Computer Science, Guilin University of Electronic Technology, Guilin 541004, China)

Abstract Model checking has being used mainly to check if a system satisfies the specifications expressed in temporal logic. People pay little attention to the model checking problem for multi-agent logics of knowledge. However, in the distributed systems community, the desirable specifications of systems and protocols have been expressed widely in logics of knowledge. A Web service is an abstract notion that must be implemented by a concrete agent. The agent is the concrete piece of software or hardware that sends and receives messages. Thus a Web services composition can be viewed as a multi-agent system. We applied our model checker MCTK for temporal logic of knowledge to verified the Web services example of the SAS protocol. We also verified the example with three model checkers WSAT, WS-Engineer and SPIN. The experimental results show that the performance of our model checking method for verifying Web services is higher than these three model checkers.

Keywords Model checking, Temporal logics of knowledge, Multi-agent system, Web service

1 引言

随着互联网技术的迅猛发展,面向服务的体系架构(SOA,Service-Oriented Architecture)将成为80%的开发项目的基础,成为占有绝对优势的软件工程实践方法。但由于互联网具有开放性,服务及服务协同的演化具有动态性,其运行环境也复杂多变,为基于互联网的开发带来了多种不确定因素,使得服务的正确性、安全性、可靠性、可用性、时效性等可信性质难以得到保证。需要提出面向Web服务的形式化建模和验证方法确保Web服务的可靠性、安全性等可信性质。

面向服务的验证与确认的研究尚处于起步阶段,已有工作主要针对Web服务的实现,从模型检测和测试两方面探讨对服务规约及服务动态特性的检测。测试并不能保证可靠性,因此本文将采用模型检测技术实现Web服务的形式化验证。

对于Web服务的模型检测,大部分的方法是将BPEL[1]等作为Web服务描述语言,用时态逻辑模型检测工具(如SPIN或NuSMV)验证安全性和行为属性。Pistore等[2]提出了一种基于"规划模型检测"的技术来设计Web服务组合的不确定性以及监视BPEL进程。Fu等[3]提出了一个框架,即把BPEL转换成SPIN的Promela语言,再用SPIN进行验证。Hai Huang等[4]提出的方法是通过使用OWL-S语言描述的BLAST模型检测工具自动生成测试用例,从而验证Web服务组合。

模型检测作为有限状态系统的自动形式化验证方法主要采用时态逻辑表达系统规范,比如模型检测器SPIN和FORSPEC采用线性时态逻辑LTL,而模型检测器SMV采用分支

到稿日期:2008-10-08 返修日期:2009-02-16 本文受国家自然科学基金(60763004),广西青年科学基金(桂科青0728090)和广西研究生教育创新计划项目(2007105950811m19)资助。

骆翔宇(1974—),男,博士,副教授,主要研究方向为模型检测、多智能体系统、知识推理等,E-mail: shiangyuluo@gmail.com;陈 艳(1984—),女,硕士生,主要研究方向为模型检测、时态逻辑、认知逻辑等;古天龙(1964—),男,教授,博士生导师,主要研究方向为软件规约、验证及测试、协议工程、知识工程与符号计算等;董荣胜(1965—),男,教授,主要研究方向为模型检测、安全协议验证、传感器网络等。

时态逻辑 CTL。然而,时态认知逻辑的模型检测问题却没得到足够的重视。时态认知逻辑的模型检验技术非常适用于多智能体通信协议这一当前的研究热点问题。模型检测知识最先由 Halpern 和 Vardi 提出[5]。文献[6,7]给出了一些模型检测知识规范的算法及其复杂度的讨论,但是没有给出一个实用的关于知识和时间的模型检测方法。Benerecetti 等[8]给出了一种基于特殊 Kripke 结构的时态模态逻辑的模型检测技术。Meyden 和 Su[9]给出了一种带有知识的匿名规范的模型检测方法,他们假设智能体具有完美记忆,但是没有考虑认知模态词的嵌套。Hoek 和 Wooldridge[10]提出的方法是把 CKLn 时态认知逻辑模型检测问题转化为线性时态逻辑 LTL 的模型检测问题,但该方法的处理过程需要人工输入,而且没有相关工具加以实现。

在 W3C 中有这样的定义:Web 服务是一个抽象的概念,由一个具体的智能体实现,这个智能体是具体的软件或硬件部分,可用来发送和接收消息。从这一观点出发,可以把 Web 服务看作是智能体,而 Web 服务的组合可看作是一个多智能体系统。当前,基于时态认知逻辑的多智能体系统模型检测工具主要有 MCK,MCMAS 和 DEMO 等。在文献[11]中给出了一种基于带有局部变量解释系统语义的符号化 CKLn 模型检测方法,并基于 NuSMV 实现了 CKLn 模型检测工具 MCTK。本文将采用 MCTK 验证 Web 服务组合的可信性质。实验结果表明该方法在性能上优于其他基于时态逻辑模型检测工具的方法,而且可以验证传统模型检测工具无法处理的认知规范。

本文第 2 节简介时态认知逻辑及其符号模型检测工具 MCTK;第 3 节以 SAS 股票分析服务实例为例阐述 Web 服务在 MCTK 中的建模方法,并给出与其他模型检测工具进行比较的实验结果;最后总结全文并展望未来相关工作。

2 模型检测时态认知逻辑

2.1 多智能体解释系统

在多智能体系统中,每个智能体在任意时间点都处于某种状态,称之为智能体的局部状态(和系统的全局状态相对应)。令 L_i 为智能体 i 的可能局部状态集合,则 $G\subseteq L_1\times\cdots\times L_n$ 为系统的可达全局状态集合。G 上的一个"运行"是从时间域到 G 上的一个函数,而运行 r 和时间 m 组成一个"点" (r,m),于是 $r_i(m)$ 则表示智能体 i 在点 (r,m) 的局部状态。

令 AP 为原子命题集合,则解释系统 $I=(R,\pi)$,其中 R 为全局状态上的运行的集合,而 π 为赋值函数,它赋予 R 中每个点上值为 true 的原子命题集合。

我们定义智能体在点集合上的等价关系 $\sim_i:(r,u)\sim_i(r',v)$ 当且仅当 $r_i(u)=r_i'(v)$。如果 $(r,u)\sim_i(r',v)$,则称 (r,u) 和 (r',v) 对于智能体 i 来说是不可区分的,即智能体 i 在 (r,u) 和 (r',v) 具有相同的信息。

2.2 时态知识逻辑 CKLn

CKLn 是 Halpern 和 Vardi 在 1989 年提出的一种时态知识逻辑[9]。CKLn 语言是由线性时态逻辑 LTL 语言加入知识算子 K_i(对于每个智能体 i),以及公共知识算子 C_Γ(对于一组智能体 Γ)。CKLn 的语法定义如下:

$$\langle f\rangle ::= true\mid p\mid\neg\langle f\rangle\mid\langle f\rangle\wedge\langle f\rangle\mid X\langle f\rangle\mid\langle f\rangle U\langle f\rangle\mid K_i\langle f\rangle\mid C_\Gamma\langle f\rangle$$

CKLn 的语义则通过可满足关系"$\models$"给出。给定原子命题集合 AP,解释系统 $I=(R,\pi)$ 和点 (r,u),有:

- $(I,r,u)\models p$ 当且仅当 $p\in\pi(r,u)$,其中 $p\in AP$;
- $(I,r,u)\models\neg\phi$ 当且仅当 $(I,r,u)\not\models\varphi$;
- $(I,r,u)\models\phi\wedge\varphi$ 当且仅当 $(I,r,u)\models\phi$ 并且 $(I,r,u)\models\varphi$;
- $(I,r,u)\models X\phi$ 当且仅当 $(I,r,u+1)\models\phi$;
- $(I,r,u)\models\phi U\varphi$ 当且仅当存在 $u'\geqslant u$ 使得 $(I,r,u')\models\varphi$,且对所有 $u\leqslant u''\leqslant u'$,有 $(I,r,u'')\models\phi$;
- $(I,r,u)\models K_i\phi$ 当且仅当对所有满足 $(r,u)\sim_i(r',v)$ 的 (r',v),有 $(I,r',v)\models\phi$;
- $(I,r,u)\models C_\Gamma\phi$ 当且仅当对所有满足 $(r,u)\sim_\Gamma^C(r',v)$ 的 (r',v),有 $(I,r',v)\models\phi$,其中 $\sim_\Gamma^C$ 为 $\bigcup_{i\in\Gamma}\sim_i$ 的传递闭包。

2.3 知识模态词的符号模型检测算法

本文采用在文献[11]中提出的基于局部命题的符号化模型检测知识的算法。给定一个 Kripke 模型 $M=\langle S,V,\sim_1,\cdots,\sim_n\rangle$,可用一个带 n 个智能体的有限状态程序 $P_M=\langle X,\theta(X),\tau(X,X'),O_1,\cdots,O_n\rangle$ 符号化地表示 M,其中 $X=\{x_1,\cdots,x_k\}$ 是一个命题变量集合,用于编码 M 中的状态,即一个状态是 X 或其子集的一个真值赋值。这样,可用 X 上的命题公式表示 S 及其子集。初始条件 θ 是 X 上的布尔公式,对系统可能的初始状态集合进行编码。迁移关系 τ 是 $X\cup X'$ 上的布尔公式,其中 $X'=\{x'_1,\cdots,x'_k\}$ 是另一个命题变量集合。如果 $\tau(X\cup X')$ 在 X 和 X' 的赋值(即状态)s 和 s' 下为 true,则意味着系统从当前状态 s 迁移到下一状态 s'。这样,从满足 $\theta(X)$ 的初始状态开始不断依照满足 τ 的迁移关系展开获得新的状态,得到的可达状态集合就是 M 中的 S。对于任一 $i=1\cdots n$,$O_i\subseteq X$ 是智能体 i 的可观察(局部)变量集合。给定一个状态 s,那么 $s\cap O_i$ 就是智能体 i 在状态 s 中的局部状态。假设对于某一命题变量 x_j,如果 $s\in V_{x_j}$,则有 $s(x_j)=$true。我们在 P_M 中忽略 V。

给定一个带 n 个智能体的有限状态程序 $P_M=\langle X,\theta(X),\tau(X,X'),O_1,\cdots,O_n\rangle$,构造表示 M 中 S 的全局可达状态集合的量化布尔公式:

$$G(P_M)=\mathrm{lfp}\,Z\left[\theta(X)\vee(\exists X(Z\wedge\tau(X,X')))\left(\frac{X'}{X}\right)\right]$$

其中 $\mathrm{lfp}\,Z\xi(Z)$ 是从 X 上的布尔公式到 X 上的布尔公式的运算 ξ 的最小不动点。令 ψ,ψ_0 是 X 上的布尔公式,如果 $\xi(\psi)\Leftrightarrow\psi$,称 ψ 是 ξ 的一个不动点。如果 ψ_0 是 ξ 的一个不动点,并且对任一 ξ 的不动点 ψ 有 $\psi_0\Rightarrow\psi$,则称 ψ_0 是 ξ 的最小不动点。$\varphi\left(\frac{X'}{X}\right)$ 是将公式 φ 中的 X' 变量置换为对应的 X 变量后得到的布尔公式。最后,给定一个公式 φ,根据文献[11]的命题 3,有:

$M\models K_i\varphi$ 当且仅当 $\forall(X-O_i)(G(P_M)\Rightarrow\varphi)$

这就是符号化模型检测知识的算法。由于有序二叉判定图 OBDD 支持布尔函数的与、或、非运算以及全称和存在量化运算,而最小不动点 $\mathrm{lfp}\,Z\xi(Z)$ 可从一个表示 false 的 OBDD 开始不断迭代计算 ξ 得到,因此很容易将 CKLn 公式在 M 中的可满足问题转化为一系列 OBDD 运算。

2.4 模型检测工具 MCTK

根据上述模型检测理论，在 LINUX 系统环境下用 C 语言实现一个 CKLn 符号模型检验工具 MCTK，可用于多智能体系统知识与时间的模型检测。MCTK 是在著名的时态逻辑模型检测工具 NuSMV 2.1.2 的基础上扩展实现的。事实上，MCTK 的规范还支持路径量词 E(存在路径)和 A(所有路径)。因此，MCTK 不仅可以验证 CTL 和 LTL，还可以验证包含 CTL * 在内的时态知识逻辑 CKLn。MCTK 的符号化算法采用科罗拉多大学的 Fabio Somenzi 开发的 CUDD OBDD 软件包。MCTK 中的智能体定义为一个模块(module)，其中的形式参数允许用户指定其对于该智能体是否是可观察的(observable)。智能体可观察形式参数名的前缀必须是“ObsPrm_”，用以标明该形式参数是可观察的。例如，形式参数 ObsPrim_x 是可观察的。一旦声明了一个智能体 i 模块实例，则该智能体的可观察变量集合 O_i 包含智能体 i 模块中的局部变量及其可观察实际参数。在该框架中，智能体的局部状态由其可观察变量表示。如果定义在该智能体模块中的局部变量是另一个模块实例，则这一模块实例中的局部变量(不包括模块实例变量)将递归地插入该智能体的局部变量集合中。

3 实例研究

本节将以文献[12]中的股票分析服务 SAS 协议为例，阐明如何用所开发的模型检测工具 MCTK 对 Web 服务进行形式化建模和验证。

SAS 协议涉及到 3 个参与者：投资者、股票经纪人和研究部门。首先，投资者向股票经纪人发送一个注册消息，股票经纪人可接受或拒绝这个注册消息。如果这个注册消息被接受，股票经纪人就发送分析请求给研究部门，研究部门发送一份分析结果报告给投资者。接收到这份报告后，投资者可向股票经纪人发送确认消息，也可取消这次服务。最后，股票经纪人可向投资者发送账单，也可继续处理其他请求。

3.1 系统建模

建模过程中涉及到投资者、股票经纪人和研究部门这 3 个 Web 服务，把它们看成 3 个智能体来进行建模。这样，就可以把 Web 服务组合看成是一个多智能体系统(Multiagent Systems，简称 MAS)，一个 MAS 由一个环境和若干个智能体组成。考虑到每一智能体只能观察环境和其他智能体的一部分状态，智能体可通过自身行为改变自身和环境的状态，也可与其他智能体交互通信。在这里，把这种交互通信看成是一个消息通信机制。为此，引入了通道的概念。这些智能体通过通道发送或接收消息，从而使得自己的知识得以更新。因此，智能体通过发送消息和接收消息，可以获取知识并可以通过观察到的数据，推理其他智能体的状态和知识。

然而，模型检测工具 MCTK 不能像 SPIN 那样直接支持通道，本文通过建立消息模块 message 来间接地模拟通道这一实体。同时令一个智能体直接使用的消息通道(包括发送和接收通道)对它自身而言是可观察的。消息通道模块定义如下：

```
MODULE message
VAR
mtype:{none, register, ack, cancel, accept, reject, bill, request, terminate, report};
source: { none, InvestorAgent, StockBrokerAgent, ResearchDepartmentAgent};
dest:{none, InvestorAgent, StockBrokerAgent, ResearchDepartmentAgent};
data1:{none, investorID, accountNum, stockID0, stockID1, stockID2, terminate};
data2:{none, orderID, creaditCard, stockID_occ};
```

在消息模块中，mtype 表示消息类型，用以标明消息的含义；source 标明消息直接来源于哪一智能体；dest 标明消息发送的目标智能体；data1 和 data2 分别为各种类型消息的附加信息。

这样，消息模块的一个实例就是一个通道，而且假设每一通道都是单向传送消息的。因此整个 SAS 协议需要 4 个通道完成消息传递(如图 1 所示)：投资者(Inverstor)通过通道 1 向股票经纪人(Stock Broker Firm)发送消息；而股票经纪人通过通道 2 向投资者发送消息；股票经纪人通过通道 3 向研究部门(Research Department)发送消息；研究部门通过通道 4 向投资者发送消息。通道(带方向的直线表示)旁标注的文字表示该通道传送的消息类型。

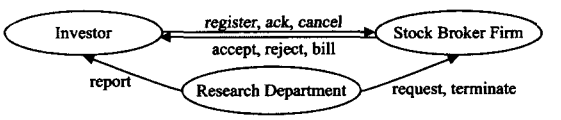

图 1 SAS 协议的通信模型

为了对上述通信模型进行建模，用 MCTK 的类 SMV 输入语言在 SAS 的环境(也就是 main 模块)中声明 4 个通道，如下所示：

```
MODULE main()
VAR
ch1_I2R:message;   ——通道 1
ch2_R2I:message;   ——通道 2
ch3_I2R:message;   ——通道 3
ch4_R2I:message;   ——通道 4
```

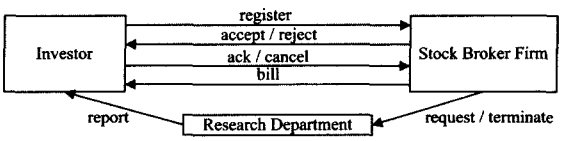

图 2 SAS 协议的处理流程

接下来要声明上述 3 个智能体：投资者、股票经纪人和研究部门。在 MCTK 的智能体声明中，用户需要根据实际情况指定该智能体可观察的环境变量和其他智能体局部变量，使得该智能体能够观察这一有限的变量集合中的变量取值，从而推断出其不能直接观察到的事实。因此，智能体可观察变量集合的选取非常重要。对于 SAS 协议的通信模型来说，每一智能体除了能观察到它自身的局部变量外，应该还能观察到它直接使用的通道(无论发送还是接收通道)中的所有变量，因为通道是智能体与外界(包括环境和其他智能体)交互的主要载体。当然，对于其他实例来说，不排除使用共享变量的方式进行交互的情况。在这种共享方式下，共享变量被所有访问它的智能体所观察。SAS 协议的智能体模块处理流程如图 2 所示。由于智能体模块内部的处理流程在 MCTK 中的描述机械冗长，篇幅所限，本文不罗列其中的具体细节，重点关注每一智能体可观察变量集合的定义。

首先投资者模块定义为：

```
MODULE InvestorAgent(ch1_I2R,ch2_R2I,ch4_R2I,
ObsPrm_ch1_I2R_mtype,ObsPrm_ch1_I2R_source,
ObsPrm_ch1_I2R_dest,ObsPrm_ch1_I2R_data1,
ObsPrm_ch1_I2R_data2,ObsPrm_ch2_R2I_mtype,
ObsPrm_ch2_R2I_source,ObsPrm_ch2_R2I_dest,
ObsPrm_ch2_R2I_data1,ObsPrm_ch2_R2I_data2,
ObsPrm_ch4_R2I_mtype,ObsPrm_ch4_R2I_source,
ObsPrm_ch4_R2I_dest,ObsPrm_ch4_R2I_data1,
ObsPrm_ch4_R2I_data2)
...
```

前 3 个形式参数意味着投资者将通过 ch1_I2R，ch2_R2I 和 ch4_R2I 3 个通道进行通信。剩余的以“ObsPrm_”起头的形式参数是这 3 个通道中的变量集合，这些变量对于投资者都是可观察的。这样做是由于 MCTK 目前的可观察变量获取算法只支持将可数或枚举变量描述为可观察的，不支持将模块类型的形式参数作为可观察变量的来源。这样，投资者的可观察变量集合就是这 3 个通道中的所有可数或枚举变量的布尔编码变量集合。

股票经纪人模块定义如下：

```
MODULE StockBrokerAgent(ch1_I2R,ch2_R2I,ch3_I2R,
ObsPrm_ch1_I2R_mtype,ObsPrm_ch1_I2R_source,
ObsPrm_ch1_I2R_dest,ObsPrm_ch1_I2R_data1,
ObsPrm_ch1_I2R_data2,ObsPrm_ch2_R2I_mtype,
ObsPrm_ch2_R2I_source,ObsPrm_ch2_R2I_dest,
ObsPrm_ch2_R2I_data1,ObsPrm_ch2_R2I_data2,
ObsPrm_ch3_I2R_mtype,ObsPrm_ch3_I2R_source,
ObsPrm_ch3_I2R_dest,ObsPrm_ch3_I2R_data1,
ObsPrm_ch3_I2R_data2)
...
```

前 3 个形式参数表明股票经纪人通过 ch1_I2R，ch2_R2I 和 ch3_I2R 3 个通道进行通信。剩余的以“ObsPrm_”起头的形式参数是这 3 个通道中的变量集合，这些变量对于股票经纪人都是可观察的。这样，股票经纪人的可观察变量集合就是这 3 个通道中的所有可数或枚举变量的布尔编码变量集合。

研究部门模块定义如下：

```
MODULE ResearchDepartmentAgent(ch3_I2R,ch4_R2I,
ObsPrm_ch3_I2R_mtype,ObsPrm_ch3_I2R_source,
ObsPrm_ch3_I2R_dest,ObsPrm_ch3_I2R_data1,
ObsPrm_ch3_I2R_data2,ObsPrm_ch4_R2I_mtype,
ObsPrm_ch4_R2I_source,ObsPrm_ch4_R2I_dest,
ObsPrm_ch4_R2I_data1,ObsPrm_ch4_R2I_data2)
...
```

前两个形式参数表明研究部门通过 ch3_I2R 和 ch4_R2I 两个通道进行通信。剩余的以“ObsPrm_”起头的形式参数是这两个通道中的变量集合，这些变量对于研究部门都是可观察的。这样，研究部门的可观察变量集合就是这两个通道中所有可数或枚举变量的布尔编码变量集合。

我们在 main 模块中分别声明上述 3 个智能体模块的实例 agentInv、agentSB 和 agentRD，分别表示智能体投资者、股票经纪人和研究部门。例如 agentInv 实例声明如下：

```
agentInv:InvestorAgent(ch1_I2R,ch2_R2I,ch4_R2I,ch1_I2R.
mtype,ch1_I2R.source,ch1_I2R.dest,ch1_I2R.data1,
ch1_I2R.data2,ch2_R2I.mtype,ch2_R2I.source,ch2_R2I.dest,
ch2_R2I.data1,ch2_R2I.data2,ch4_R2I.mtype,ch4_R2I.source,
ch4_R2I.dest,ch4_R2I.data1,ch4_R2I.data2);
```

agentSB 和 agentRD 实例声明类似。

3.2 实验结果

在时态逻辑模型检测工具 NuSMV 基础上，采用 CUDD（一种高效的 OBDD 软件包）编程实现了本文的符号化模型检测算法，并根据上述方法对 SAS 协议进行建模和验证。实验平台是带 512M 内存的 IBM ThinkPad R50e 笔记本电脑，Ubuntu 5.04 Linux 系统。为了与其他模型检测工具进行比较，首先验证下述时态逻辑规范 1：

```
G((agentInv.state=register & ch1_I2R.data1=
   stockID0) ->
   F(agentSB.state=request & ch3_I2R.data1=stock-
   ID0))
```

其中，G 和 F 是时态算子，分别表示“总是(Globally)”和“最终(Finally)”，有如下两个等式：$F\varphi \equiv \text{True U } \varphi$ 和 $G\varphi \equiv \neg F \neg \varphi$。这个规范表明，若 agentInv 注册消息为 stockID0，则 agentRD 得到的 request 消息中的股票也为 stockID0。

分别在 WSAT，WS-Engineer 和 SPIN 3 个模型检测工具中对 SAS 协议进行建模并验证与规范 1 等价的性质，结果如表 1 所列。显然，该模型检测工具 MCTK 在性能上与其他工具相比有突出的优势。

表 1 SAS 协议的实验结果比较

规范	结果	模型检测工具	运行时间
规范 1	TRUE	MCTK	0.495s
规范 1	TRUE	WSAT	28 s
规范 1	TRUE	WS-Engineer	61s
规范 1	TRUE	SPIN	15 s

还验证了包含知识模态词的时态认知逻辑规范 2：

```
G((ch3_I2R.mtype=request & ch3_I2R.data1=stock-
   ID0) ->
   F(agentRD K (ch3_I2R.mtype=request & ch3_I2R.
   data1=stockID0)))
```

其中的形式为 $ag\ K\ \varphi$ 的公式表示智能体 ag 知道 φ 成立。规范 2 表明当研究部门 agentRD 得到股票为 stockID0，则研究部门可以知道投资者发送给股票经纪人的股票消息也为 stockID0，即具有知识推理能力。规范 2 的验证结果为 True，运行时间为 0.655s。由于 WSAT，WS-Engineer 和 SPIN 均不支持时态认知逻辑规范，因此这些模型检测工具不能验证规范 2。

从以上实验结果可以看出，该模型检测方法在性能上要优于其他方法，同时可以验证传统时态逻辑模型检测工具无法验证的时态认知逻辑规范。

结束语 Web 服务是一个典型的分布式系统，本文首先把 Web 服务组合看成是一个多智能体系统，其中的每一服务可由一个具体的智能体实现系统中的软件或硬件组件，这些组件可相互发送和接收消息。针对 SAS 股票分析服务实例，在 MCTK 输入语言中设计了基于通道的通信模型，从而方便有效地采用实现的时态认知逻辑模型检测工具 MCTK 符号化地描述该多智能体系统，并成功验证了一些时态和知识逻辑规范，使得 Web 服务形式化验证扩展到了知识推理领域。

今后将从 Web 服务形式化验证需求出发，系统地提出

Web服务的描述语言到模型检测所需的形式模型的转换方法，并考虑采用组合验证和符号化算法优化模型检测性能，从而推动Web服务形式化验证的实用化。

参考文献

[1] Curbera F, Goland Y, Klein J, et al. Business Process Execution Language for Web Services[OL]. ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.pdf, 2002

[2] Pistore M, Traverso P, Bertoli P. Automated Composition of Web Services by Planning in Asynchronous Domains[C]// Proc. ICAPS'05. 2005

[3] Fu X, Bultan T, Su J. Analysis of interacting BPEL web services [C]// WWW'04. ACM Press: 621-630

[4] Huang Hai, Tsai Wei-Tek, Paul R, et al. Automated model checking and testing for composite web services[C]// ISORC'05. IEEE Computer Society: 300-307

[5] Halpern J, Vardi M Y. Model checking vs. theorem proving: A manifesto[R]. IBM Almaden Research Center, 1991. An extended version of a paper in Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning, 1991

[6] van der Meyden R. Common knowledge and update in finite environments. I: extended abstract[C]// Proc. of the Conf. on Theoretical Aspects of Reasoning about Knowledge. 1994: 225-242

[7] van der Meyden R, Shilov N S. Model checking knowledge and time in systems with perfect recall[C]// Proc. Conf. on Software Technology and The oretical Computer Science, LNCS No 1738. Berlin: Springer, 1999: 262-273

[8] Benerecetti M, Giunchiglia F, Serafini L. A model checking algorithm for multi-agent systems[C]// J. P. Muller, M. P. Singh, A. S. Rao, eds. Intelligent Agents V, volume LNAI Vol. 1555. Berlin: Springer-Verlag, 1999

[9] van der Meyden R, Su Kaile. Symbolic model checking the knowledge of the dining cryptographers[C]// Proc of 17th IEEE Computer Security Foundations Workshop. June 2004: 280-291

[10] van der Hoek W, Wooldridge M. Model checking knowledge and time[C]// Proc. 19th Workshop on SPIN (Model Checking Software). Grenoble, April 2002

[11] Su Kaile, Sattar A, Luo Xiangyu. Model Checking Temporal Logics of Knowledge Via OBDDs[J]. The Computer Journal, 2007, 50(4): 403-420

[12] Fu Xiang, Bultan T, Su Jianwen. Model checking XML manipulating software[C]// Proceedings of the ACM/SIGSOFT International Symposium on Software Testing and Analysis, ISSTA 2004. Boston, Massachusetts, USA, July 2004: 252-262

(上接第125页)

能影响因素，比如说数据库连接池和线程池的调度等。

参考文献

[1] Smith C U, Williams L G. Performance Solutions[M]. Addison-Wesley, 2002

[2] Cortellessa V, Mirandola R. PRIMA-UML: a Performance Validation Incremental Methodology on Early UML Diagrams[J]. Science of Computer Programming, Elsevier Science, 2002, 44 (1): 101-129

[3] Petriu D B, Woodside M. Software Performance Models from System Scenarios in Use Case Maps[C]// Proceedings of International Conference on Modeling Techniques and Tools for Performance Evaluation. LNCS 2324. 2002: 141-158

[4] Verdickt T, Dhoedt B. Automatic Inclusion of Middleware Performance Attributes into Architectural UML Software Models [J]. IEEE Transactions on Software Engineering, 2005, 31(8): 695-711

[5] Petriu D C, Shen H. Applying UML Perfromance Profile: Graph Grammar-based Derivation of LQN Models form UML Specifications[C]// Proc. of International Conference on Modelling Techniques and Tools for Performance Evaluation. LNCS 2324. 2002: 159-177

[6] Woodside M, Petriu D C. Performance by Unified Model Analysis (PUMA)[C]// Proc. 5th Int. Workshop on Software and Performance. Palma de Mallora. July 2005: 1-12

[7] Woodside M. From Annotated Softwae Designs (UML SPT / MARTE) to Model Formalisms[C]// M. Bernardo and J. Hillston, eds. SFM. LNCS 4486. 2007: 429-467

[8] Lopez-Grao J P, Merseguer J, Campos J. From UML Activity Diagrams to Stochastic Petri Nets: Application to Software Performance Engineering[C]// ACM Proc. of Workshop on Software and Performance. 2004: 25036

[9] Woodside M, Frank G. The Future of Software Performance Engineering[C]// IEEE Future of Software Engineering (FOSE' 07). 2007: 171-187

[10] Jendrock E, Ball J, Carson D, et al. The Java EE 5 Tutorial [OL]. http://java.sun.com/javaee/5/docs/tutorial/doc/, Sun MicoSystems

[11] Object Management Group. UML Profile for Schedulability, Perfromance, and Time. 2005

[12] Roman E, Sriganesh R P. Mastering ejb3 or specification[M]. Addison-Wesley, 2006

[13] Object Management Group. The Common Object Request Broker: Architecture and Specification. 2002

[14] France R, Ray I. An Aspect-Oriented Approach to Early Design Modeling[C]// IEE Procceedings-software, Special Issue on Early Aspects: Aspect-oriented Requirements Engineering and Architecture Design. 2004: 173-185

[15] Miller J, Mukerji J. MDA Guide[M]. June 2003

[16] Petriu D C, Woodside M. Performance Analysis with UML[M] // B. Selic, L. Lavagno, G. Martin. UML for Real, 2003: 221-240

[17] Shen H, Petriu D C. Performance Analysis of UML Models using Aspect Oriented Modeling Techniques[C]// Model Driven Engineering Languages and Systems. LNCS 3713. 2005: 156-170

[18] Xu J, Woodside M. Template-Driven Performance Modeling of Enterprise Java Beans[C]// Proc. Workshop on Middleware for Web Services. Enschede Netherlands, 2005: 57-64

[19] Xu J, Oufimtsev A. Performance Modeling and Prediction of Enterprise JavaBeans with Layered Queuing Network Templates [C]// ACM Proceedings of Workshop on Specification and Verification of Component-Based Systems. 2005

[20] Franks R G. Performance Analysis of Distributed Server Systems[C]// Department of Systems and Computer Engineering. Carleton University, Ottawa, Ontario, Canada, December 1999

[21] Woodside M, Franks G. Tutorial Introduction to Layered Modeling of Software Perfromance[OL]. http://www.sce.carleton.ca/rads/lqns/lqn-documentation

[22] http://www.once.com.cn

[23] Wu X, Woodside. Performance Modeling from Software Components[C]// ACM Proc. of Workshop on Software and Performance. 2004: 290-301