一种基于 Mealy! 机的语义程序验证方法*

胡罗凯[1,2], 柴 新[1], 许庆炜[1], 应 时[3]
(1. 湖北第二师范学院 计算机学院, 武汉 430205; 2. 华中科技大学 计算机学院, 武汉 430074; 3. 武汉大学 软件工程国家重点实验室, 武汉 430072)

摘 要: 语义验证是束缚语义软件和语义程序设计语言发展的问题之一,针对这一问题,在基于语义 Web 服务的语义程序设计语言 SPL 及其知识库业务领域本体(BDO)的基础上,提出了一种基于 Mealy! 机对 SPL 所编排的业务过程进行语义验证的方法,结合在线外汇交易平台的案例,详细描述了运用该方法进行语义验证的过程。通过案例证明,本方法有助于编写语义正确的语义程序。

关键词: Mealy 机; 本体; 语义程序设计; 语义验证
中图分类号: TP311 **文献标志码**: A **文章编号**: 1001-3695(2012)09-3320-04
doi:10.3969/j.issn.1001-3695.2012.09.032

Mealy! machine based approach for semantic program validation

HU Luo-kai[1,2], CAI Xin[1], XU Qing-wei[1], YING Shi[3]
(1. *School of Computer, Hubei University of Education, Wuhan* 430205, *China*; 2. *School of Computer Science & Technology, Huazhong University of Science & Technology, Wuhan* 430074, *China*; 3. *State Key Laboratory of Software Engineering, Wuhan University, Wuhan* 430072, *China*)

Abstract: Semantic validation is one of the shackles for the development of semantic software and semantic programming language. To solve this problem, this paper firstly introduces the semantic Web services based semantic programming language (SPL) and its knowledge base business domain ontology (BDO). It proposed a mealy! machine based approach for semantic validation of SPL language. Through a case study of an online foreign exchange trading platform, it described the approach in detail. The case proves that this approach is useful for orchestrating the semantic correct program.

Key words: Mealy machine; ontology; semantic programming; semantic validation

语义软件工程是将人工智能中较成熟的语义技术应用到软件工程中,试图能解决软件工程实践中的一些具体问题。语义程序设计语言是语义软件工程发展的必然要求。为了确保以业务流程为中心的程序能以设计者的预期方式正常执行,往往在执行程序之前要对其进行验证。与传统的程序需要进行语法验证相比,语义程序的验证除需要进行语法验证外,还需要从意义(meaning)层对程序进行验证,这里称之为语义验证(semantic validation),以验证程序是否违反业务逻辑的基本规则或规定。这些业务逻辑在本文中称之为业务知识(business knowledge)。目前关于知识的表示方法较多,其中本体是最常用的方法之一。本体最先出现在哲学领域,是一个哲学范畴。随着人工智能的发展,本体被引入到计算机领域,本体在软件开发中可以是某一领域的一组概念和概念之间关系的明确的规格说明[1]。在本文提出的基于 Mealy! 机的语义验证方法中,使用本体作为语义程序的知识库,以存储相关的业务知识,作为验证语义程序的语义基础。

1 相关工作

目前,国内外关于程序验证的方法归纳起来主要有三类:

a)基于模型检测的方法,其中面向 C 程序的模型检测工具有 Zing[2] 和 ComFoRT[3] 等,面向 Java 程序的模型检测工具有 Java PathFinder[4] 等。实际上,目前大部分应用系统的规模都超出了模型检测可处理的范围。

b)基于抽象解释的程序验证方法,抽象解释本质上是在计算效率和计算精度之间取得均衡、以损失计算精度求得计算可行性、再通过迭代计算增强计算精度的一种抽象逼近方法[5]。Cousot 等人[6~8] 以嵌入式实时软件静态分析工具 ASTREE 为主要成果,进行了一系列研究。

c)基于谓词抽象的程序验证方法,如微软研究院 Ball 等人[9] 设计的软件自动验证工具包 SLAM、加州大学伯克利分校 Henzinger 等人[10] 的软件自动验证工具 BLAST、卡耐基梅隆大学 Chaki 等人[11] 的软件自动验证工具 Magic 等。

在面向服务的程序验证方法中,因为 Petri 网适合于分布式系统的建模,许多研究工作都以 Petri 网作为 Web 服务组合建模的理论基础。Martens 等人[12] 为了验证 BPEL 的全局抽象流程与各参与者的局部可执行流程之间的一致性,即检验各局部流程能否在全局流程中正常交互,采用 Petri 网对 BPEL 进行建模,并利用 Wombat4WS 工具进行自动验证。与其类似,

收稿日期: 2012-03-04; **修回日期**: 2012-05-03 **基金项目**: 国家自然科学基金资助项目(61070012);湖北省教育厅重点科研项目(D20103004);湖北省科学技术研究计划优秀中青年人才项目(Q20113001)

作者简介: 胡罗凯(1981-),男,湖北武汉人,博士(后),主要研究方向为语义 Web 和智能软件工程(luokaihu@gmail.com); 柴新(1968-),男,湖北武汉人,副教授,硕士,主要研究方向为软件工程和知识工程;许庆炜(1976-),男,湖北武汉人,副教授,博士,主要研究方向为语义 Web 和知识工程; 应时(1965-),男,湖北武汉人,教授,博导,博士,主要研究方向为软件体系结构.

Hinz 等人[13]也提出了将 BPEL 转换成 Petri 网,然后利用 Petri 网的分析验证技术对 BPEL 程序进行分析验证。Bordeaux 等人[14]介绍了如何利用进程代数表达 Web 服务的编制和编排,即利用进程代数形式化描述 Web 服务以及 Web 服务组合,进而说明了该方法对于 Web 服务组合的验证和指导、Web 服务开发的作用和意义。Brogi 等人采用 CCS 对 Web 服务编排协议 WSCI[15]进行形式化建模,并对 WSCI 中参与交互的服务进行兼容性和可替换性分析;此外,对两个无法正常交互的服务,还提供了适配器(adaptor)机制使得两者能够实现通信[16]。Zhao 等人[17]提出了一种基于简化的进程代数形式化语言来刻画 WS-CDL 的方法,并采用模型检验工具对服务交互过程的正确性进行验证。Fu 等人[18]从服务之间的消息交互出发,将服务形式化描述为一个具备先进先出输入消息队列的非确定型 Buchi 自动机,将服务组合视为服务之间通过异步消息传递的全局会话协议(conversation protocol),同时提出了会话协议的可行性条件和异步消息的可同步化条件。然而,他们的工作主要都集中在对程序的语法验证上,目前由于语义程序设计语言的工作还不成熟,关于程序语义验证的研究还不多。

2　SPL 及其 Mealy! 机模型

针对目前仍缺乏基于语义信息将语义 Web 服务资源动态组合起来形成大规模复杂应用软件的方法、技术和支撑平台等问题,定义了一种面向语义 Web 服务资源的语义编程语言(semantic Web service based semantic programming language, SWS-SPL,简称 SPL)[19]。

2.1　SPL 的语法结构

SPL 是一种以语义信息处理为基础的、以语义 Web 服务为软件构成元素的编程语言。使用 SPL 在语义层进行程序设计,可以有效避免由于语法层的异构而造成的各种问题。SPL 中使用语义服务来表示 Web 服务的语义层抽象,使用语义数据表示程序中各种数据的语义层抽象。通过对语义服务、语义数据的操作和处理,完成语义程序设计,而不关心服务的接口和数据语法等问题,使得编程过程更加轻松,开发出的程序具有更大的灵活性。SPL 程序由定义和程序体两个部分组成。定义(definition)部分包括语义数据类型定义、一般数据类型定义和语义服务定义三个子部分组成;程序主体(program)部分包括语义变量定义、一般变量定义和过程体三个子部分组成。

SPL 过程(process)定义一组有序的活动来实现特定的业务目标。业务过程指定了一组活动以及这些活动可能的执行顺序、活动之间共享的数据、参与业务过程的语义服务(业务伙伴服务)、一组共同的异常处理、一组共用的补偿处理等。过程分为主过程和子过程两种,采用同样的方式来定义。

SPL 中的活动(activity)被定义成基本活动(basicactivity)和结构化活动(structuredactivity),前者包括调用 Web 服务提供的操作(invoke),负责接收消息(receive)和发送响应消息(reply);后者包括顺序、选择、循环和并发在内的四种控制结构。活动的定义如下:

Activity:BasicActivity | StructuredActivity

BasicActivity:Invoke | Receive | Reply

StructuredActivity:Sequence | If-else | While | Parallel

其中:基本活动用 ServiceRole、Operation、Input、Output 分别说明执行该活动的服务角色、要调用的操作及其输入、输出消息。其定义为:

a) Invoke: y = S. O? [x],表示使用参数 x 调用语义服务 S 的操作 O,得到返回参数 y。

b) Receive: S. O? [x],表示接收语义服务 S 的操作 O 的参数 x。

c) Reply: S. O! [x],表示向语义服务 S 的操作 O 发送参数 x。

结构化活动的定义如下:

a) Sequence:S1;S2,表示活动 S1 与 S2 顺序执行。

b) If-else: S1 + S2,表示活动 S1 与 S2 选择执行。

c) While: S *,表示活动 S 循环执行。

d) Parallel: S1|S2,表示活动 S1 与 S2 并行执行。

下文将对以上七种活动分别刻画它们的 Mealy! 机模型。

2.2　SPL 的 Mealy! 机模型

本文将使用一种区分输入与输出的确定的有穷状态机模型,它是传统 Mealy 机的一种,不妨将其定义为 Mealy! 机。

定义 1　Mealy! 机。Mealy! 机 M 是一个七元组:$M = (Q,\Sigma,\Delta,\delta,\lambda,q0,F)$。其中,$Q$ 为状态的非空有穷集合,$\forall q \in Q$,q 称为 M 的一个状态(state);Σ 为输入字母表;Δ 为输出字母表;δ 为状态转换函数。$\delta: Q \times \Sigma \to Q$。对 $\forall (q,a) \in Q \times \Sigma$,$\delta(q,a) = p$表示 M 在状态 q 读入字符 a,将状态变成 p。在对 SWS-SPL 的建模过程中,可以理解为语义服务 SWS1 接受了一个输入 a,其状态由 q 变为 p;q_0 为 M 的开始状态,$q_0 \in Q$。

以上定义均与经典 Mealy 机模型一致。

λ 为状态转换函数。$\lambda: Q \times \Delta \to Q$,对 $\forall (q,a) \in \Delta \times \Sigma$,$\lambda(q, a) = p$表示 M 在状态 q 读入字符 a,将状态变成 p。在对 SWS-SPL 的建模中,可以理解为语义服务 SWS_1 产生了一个输出 a,其状态由 q 变为 p。为区分语义服务的输入与输出,将输入参数 a 记为? a,输出参数 a 记为! a。

F 为 M 的终止状态集合,$F \subseteq Q$。$\forall q \in F$,q 称为 M 的终止状态。

SPL 中定义的活动到 Mealy! 机模型的转换关系如表 1 所示。其中,S 表示服务调用方,P 表示服务提供方。

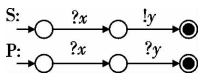
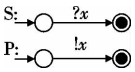
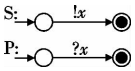
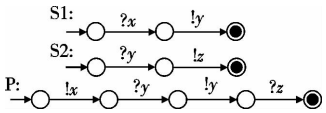
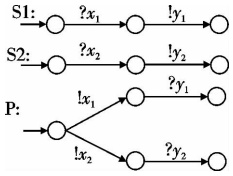
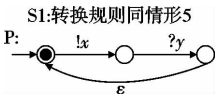
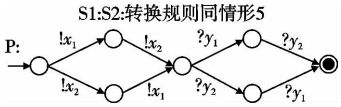
3　构建面向 Web 服务及其过程的业务领域本体

语义软件开发需要定义一个可共同理解的知识库作为语义软件设计、开发的语义基础。针对基于语义 Web 服务的语义软件开发的特点,本文将构建一个以服务为中心的、以服务过程为主要内容的业务领域本体,以作为构建 SPL 程序的语义基础。利用 OWL 及其形式化基础—描述逻辑(description logic,DL)[20],本体技术可以很好地用于语义 Web 服务的描述、发现、组合和调用,以及基于语义 Web 服务的语义程序的部分正确性验证。选择描述逻辑而不是一阶逻辑、高阶逻辑或命题逻辑的原因在于:a)一阶逻辑或高阶逻辑是不可判定的;b)命题逻辑是可判定的,但表达能力却不强;c)DL 的表达能力较强且是可判定的。

业务领域本体(business domain ontology, BDO)按照业务服务、数据、功能、过程四个部分来组织本体中的概念及概念之

间的关系。

表 1 SPL 程序到 Mealy! 机的转换关系

序号	SPL 活动	Mealy! 机模型
1	Invoke:y = S. O? [x]	
2	Receive:S. O? [x]	
3	Reply:S. O! [x]	
4	Sequence: S1[? x,! y] S2[? y,! z]	
5	if-else: S1 + S2	
6	While:S1 *	
7	Parallel: S1 | S2	

3.1 业务服务部分

业务服务部分用于表示特定业务领域中常用的业务服务的概念。由于特定领域中的业务服务较多,如果仅仅是将所有的业务服务进行罗列,而没有分类的话,非常不利于检索、编辑和管理。因此需要对领域中的业务服务进行分类,以便于进行业务服务的管理和检索。业务服务的分类法有很多种,BDO 可以支持一种或多种分类方法。

业务服务部分包括两种元概念即服务分类(TN)与业务服务(BS)和两种元属性即 hasFunction 与 hasService。

$TN \sqsubseteq hasService.BS$

$BS \equiv \exists hasFunction.RFN \sqcap \forall hasFunction.RFN \sqcap \geqslant 1 hasFunction$

3.2 数据部分

数据部分用于表示特定业务领域中常用的业务数据的概念及概念之间的关系。

数据部分只定义一种元概念即业务数据(DN)。所有的作为服务输入输出参数的领域概念都是业务数据的子概念。数据部分中的属性为领域相关的属性,无须建立元属性。

3.3 功能部分

功能部分用于表示业务领域中业务功能的概念及这些业务功能所需要的输入输出数据集。

功能部分包括五种元概念:功能(FN)、数据集(DS)、输入数据集(IDS)、输出数据集(ODS)、错误信息(ERR)以及四种元属性 hasInputDataSet、hasOutputDataSet、hasError、hasDataParameter。其中功能(FN)又分为可完成实际工作的功能(记为 RFN)和空功能(记为 EFN)。

$RFN \sqsubseteq \exists hasInputDataSet.IDS \sqcap \forall hasInputDataSet.IDS \sqcap \geqslant 1 hasInputDataSet \sqcap \leqslant 1 hasInputDataSet$

$RFN \sqsubseteq \exists hasOutputDataSet.ODS \sqcap \forall hasOutputDataSet.ODS \sqcap \geqslant 1 hasOutputDataSet \sqcap \leqslant 1 hasOutputDataSet$

$RFN \sqsubseteq \exists hasError.ERR \sqcap \forall hasError.ERR \sqcap \geqslant 1 hasError$

$DS \sqsubseteq \exists IDS \sqcap ODS \sqcap ERR$

$DS \sqsubseteq \exists hasDataParameter.DN \sqcap hasDataParameter.DN \sqcap \geqslant 1 hasDataParameter$

$FN \sqsubseteq \exists EFN \sqcap RFN$

3.4 业务过程部分

业务过程部分用于表示业务领域中经常出现的业务过程,其相对于一般业务服务而言,完成的功能更为复杂。一个业务过程可以包括多个业务服务(或功能),将多个业务服务(或功能)按一定流程进行组合。业务过程部分中的元概念与元属性的关系如图 1 所示。

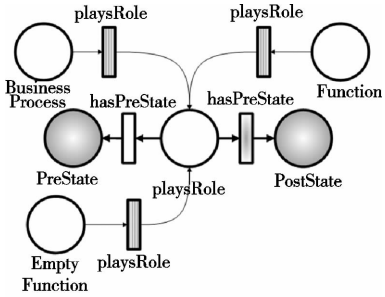


图1 业务过程部分

概念:状态(ST)、原子状态(AS)、组合状态(CS)、业务过程(BP)、过程角色(PR)

属性:playsRole、hasPreState、hasPostState

本体中业务过程部分与 Mealy! 机的转换关系与表 1 类似,不再赘述。

$ST \sqsubseteq AS \sqcap CS$

$CS \equiv (\forall hasORState.ST \sqcap \exists hasORState.ST \sqcap \geqslant 2 hasORState \sqcap 2 hasORState) \sqcup (\forall hasANDState.ST \sqcap \exists has\ ANDState.ST \sqcap \geqslant 2 hasANDState \sqcap \leqslant 2 hasANDState)$

$FN \sqsubseteq \exists playsRole.PR \sqcap playsRole.PR \sqcap \geqslant 1 playsRole$

$BP \equiv \forall playsRole.PR \sqcap \exists hasPreState.ST \sqcap \exists hasPostState.ST \sqcap \forall hasPreState.ST \sqcap \forall hasPostState.ST \sqcap \geqslant 1 hasPreState \sqcap \leqslant 1 hasPreState \sqcap \geqslant 1 hasPostState \sqcap \leqslant 1 hasPostState$

$PR \equiv \exists hasPreState.ST \sqcap \exists hasPostState.ST \sqcap \forall hasPreState.ST \sqcap \forall hasPostState.ST \sqcap \geqslant 1 hasPreState \sqcap \leqslant 1 hasPreState \sqcap \geqslant 1 hasPostState$

4 SPL 程序的语义验证

4.1 验证方法

定义 2 SPL 程序的语义部分正确性。若对每个使得输入断言 $I(i)$ 为真,并且程序计算终止的输入信息 i、输出断言 $O(i, SP(i))$ 为真且与程序的知识库没有语义冲突,称程序 SP 是关于 I、O 和知识库 $\mathcal{KB}$ 是语义正确的,记为

$$I(i) \wedge SP\downarrow \subseteq O(i, SP(i)) \wedge (SP \ltimes \mathcal{KB})$$

其中:$SP\downarrow$ 表示语义程序 SP 终止,$SP \ltimes \mathcal{KB}$ 表示语义程序 SP 语义与知识库 $\mathcal{KB}$ 没有语义冲突。

对 SPL 程序进行语义验证的方法如下:

a)将本体作为语义程序的知识库和专家系统,首先建立以服务为中心、以业务过程为主要内容的业务领域本体(BDO),进行转换为 Mealy! 机模型。

b)将待验证的 SPL 程序中每一个语义服务转换成相应的 Mealy! 机,进行转换成正则语言。

c)如果每一个语义服务所对应的正则语言都能够被 BDO 的 Mealy! 机接受,则该 SPL 程序为语义部分正确的。在进行与正则语言匹配的过程中,需要考虑语义等价和语义包含关系。只要有一个语义服务所对应的正则语言不能被接受,则该 SPL 程序为语义是不正确的。

4.2　案例应用

互联网和在线外汇交易的发展,打破了地域的局限,使得原来必须依赖本地经纪商才能参与外汇交易的个人和小型机构投资者,可以更加方便地进行外汇投资。在金融危机的大背景下,对外汇市场有兴趣的个人投资者希望能够更便捷地完成在线外汇交易,更快地收回资金,借助于 SOA 和 Web 服务技术的不断发展,A 公司试图建立一种一站式的在线外汇交易平台,自动完成与银行及外汇交易中心的交互,省去冗长的中间环节,以更快速度回收资金,最终为客户提供超一流的外汇交易工具。

一站式外汇交易平台的 UML 活动图表示如图 2 所示。一个典型的一站式外汇交易过程是由 Forex agent 接受用户的外汇交易请求,Forex Agent 将汇款请求发给银行(JPBankService),由后者将一种货币(如美元 USD)转入外汇交易系统 E-ForexService,如果 E-ForexService 不可用(如请求忙),则转入外汇交易系统 G-ForexService,如果 G-ForexService 可用,完成外汇交易后将另一种货币(如日元 JPY)转入银行(JPBankService),完成整个交易过程。

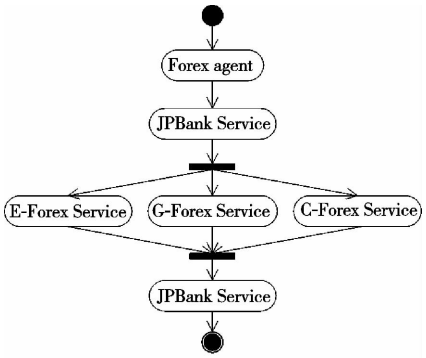


图2　一站式外汇交易

按照 2.2 节介绍的转换规则,对每个服务,将其转换成对应的 Mealy! 机模型,如图 3 所示,进而转换为 Mealy! 机所接受语言的正则表达式。

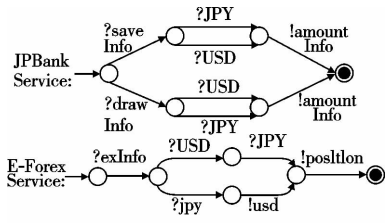


图3　语义服务对应的Mealy!机

语义服务 JPBankService 所对应语言的正则表达式为

(? saveInfo(? jpy + ? usd) + ? draw(! jpy + ! usd))! amount

语义服务 E-ForexService 所对应语言的正则表达式为

? exInfo(? usd ! jpy + ? jpy ! usd)! position

Web 服务 JPBankService 在 BDO 中对应的业务服务概念是 BankService,Web 服务 E-ForexService 在 BDO 中对应的业务服务概念是 ForeignExchangeService,将业务服务 BankService 和 ForeignExchangeService 转换成对应的 Mealy! 机,如图 4 所示。

可验证语义服务 JPBankService 对应的正则语言可以被 BDO 的 Mealy! 机接受,而语义服务 E-ForexService 对应的正则语言则不能被接受。所以该程序是语义不正确的。

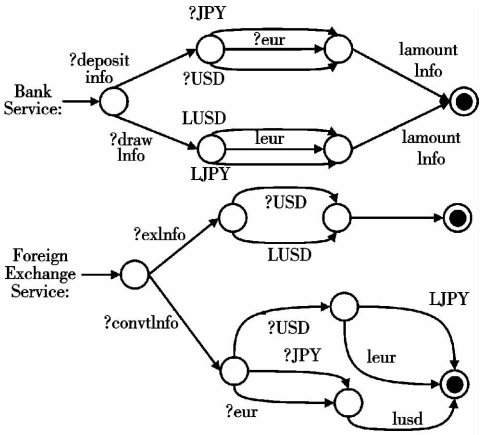


图4　业务服务对应的Mealy!机

5　结束语

针对语义程序的特点,本文首先对传统的 Mealy 机模型进行了扩充,提出了一种区分输入输出字符串的 Mealy! 机模型。利用人工智能中较成熟的本体技术作为语义程序的知识库,利用 Mealy! 机模型和 BDO 本体,在分析已有各种程序验证方法的基础上,提出了一种程序的语义验证方法,并结合一个在线外汇交易平台的案例,详细描述了该方法的验证过程。该方法能够很好地利用本体理论和自动机理论的特点,完成程序的语义验证工作。

该验证方法的基础是有一个正确和完整的本体作为专家系统,然而实际应用是纷繁复杂和动态变化的,所以本体也应该具有动态变化的能力来支持实际应用的快速变化,这是笔者下一步研究工作的重点。

参考文献:

[1] GRUBER T. Towards principles for the design of ontologies used for knowledge sharing[J]. International Journal of Human-Computer Studies, :1995,43(5-6):907-928.

[2] QADEER S, RAJAMANI S. Deciding assertions in programs with references [R]. MSR Technical Report: MSR-TR-2005-08, 2005.

[3] CHAKI S, IVERS J, SHARYGINA N, *et al*. The ComFoRT reasoning framework[C] //LNCS, vol 3576. Berlin: Springer, 2005: 164-169.

[4] PASAREANU C, VISSER W. Verification of Java programs using symbolic execution and invariant generation[C]//LNCS, vol 2989, Berlin: Springer, 2004: 164-181.

[5] 李梦君, 李舟军, 陈火旺. 基于抽象解释理论的程序验证技术[J]. 软件学报, 2008, 19(1): 17-26.

[6] COUSOT P, COUSOT R, FERET J, *et al*. The ASTRéE analyser [C]//LNCS, vol 3444. Berlin: Springer, 2005: 21-30.

[7] RIVAL X. Abstract dependences for alarm diagnosis[C]//LNCS, vol 3780. Berlin: Springer, 2005: 347-363.

[8] MINÉA. Symbolic Methods to Enhance the Precision of Numerical Abstract Domains[C]//LNCS, vol 3855. Berlin: Springer, 2006: 348-363.

[9] BALL T, MAJUMDAR R, MILLSTEIN T, *et al*. Automatic predicate abstraction of C programs[C]//Proc of ACM SIGPLAN Conference on Programming Language Design and Implementation. New York: ACM Press, 2001: 203-213.

(下转第 3329 页)

(上接第 3323 页)

[10] HENZINGER T A, JHALA R, MAJUMDAR R, *et al*. Abstractions from proofs[C]//Proc of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. New York: ACM Press, 2004: 232-244.

[11] CHAKI S, CLARKE E, GROCE A, *et al*. Modular verification of software components in C[J]. IEEE Trans on Software Engineering, 2004, 30(6): 388-402.

[12] MARTENS A. Simulation and equivalence between BPEL process medels[C]//Proc of International Conference on DASD. 2005: 232-237.

[13] HINZ S, SCHMIDT K, STAHL C. Transforming BPEL to Petri nets [C]//Lecture Notes in Computer Science, vol 3649, Berlin: Springer, 2005: 220-235.

[14] BORDEAUX L, SALAUN G. Using process algebra for web services: early results and perspectives[C]//Lecture Notes in Computer Science, vol 3324. Berlin: Springer, 2005: 54-68.

[15] WSCI. Web Service Choreography Interface (WSCI) 1.0 [EB/OL]. (2002). http://www.w3.org/TR/wsci.

[16] BROGI A, CANAL C, PIMENTEL E. Formalizing Web service choreographies[C]// Proc of the 1st International Workshop on Web Services and Formal Methods. 2004: 73-94.

[17] ZHAO Xiang-peng, YANG Hong-li, QIU Zong-yan. Towards the formal model and verification of web services choreography description language [C]//Lecture Notes in Computer Science, vol 4184. Berlin: Springer, 2006: 273-287.

[18] FU X, BULTAN T, SU J. Analysis of Interacting BPEL Web services [C]//Proc of the 13th International Conference on World Wide Web. New York: ACM Press, 2004: 621-630.

[19] 曹虹华, 应时, 杜德慧, 等. 一种面向语义 Web 服务的软件设计语言和设计方法[J]. 电子学报, 2007. 35(z2): 129-135.

[20] BAADER F, HORRROCKS I, SATTLER U. Description logics as ontology languages for the semantic Web[C]//Lecture Notes in Computer Science, vol 2605. Berlin: Springer, 2005: 228-248.