第38卷 第15期
Vol.38 No.15

计 算 机 工 程
Computer Engineering

2012年8月
August 2012

·软件技术与数据库·

文章编号：1000—3428(2012)15—0039—04 文献标识码：A 中图分类号：TP393.07

RGPS过程层元模型正确性验证

袁开银[1]，郭 瑞[2]，陆翔升[3]，吴尽昭[3]

(1. 河南财经政法大学现代教育技术中心，郑州 450002；2. 郑州轻工业学院计算机与通信工程学院，郑州 450002；
3. 中国科学院成都计算机应用研究所，成都 610041)

摘 要：利用Web服务本体描述语言对RGPS过程层元模型进行描述，建立Promela模型。基于线性时序逻辑，以及Spin检测工具的偏序规约和on-the-fly等优化技术对Promela模型进行正确性验证，设计并实现RGPS过程层元模型正确性验证平台。通过城市交通系统实例证明该验证方法的正确性和有效性。

关键词：RGPS框架；Promela建模；Spin模型检测工具；过程层元模型；线性时序逻辑

Correctness Verification of RGPS Process Level Meta-model

YUAN Kai-yin[1], GUO Rui[2], LU Xiang-sheng[3], WU Jin-zhao[3]

(1. Modern Education Technology Center, Henan University of Economics and Law, Zhengzhou 450002, China;
2. College of Computer and Communication Engineering, Zhengzhou University of Light Industry, Zhengzhou 450002, China;
3. Chengdu Institute of Computer Application, Chinese Academy of Sciences, Chengdu 610041, China)

【Abstract】This paper establishes the Promela model of the Web Ontology Language for Service(OWL-S) process model for Role Goal Process Service(RGPS) process level meta-model, uses Linear Temporal Logic(LTL) to describe the properties of models, and uses the partial order reduction and on-the-fly optimization techniques of model checking tools Spin to verify the properties. It designs and implements RGPS process level meta-model correctness verification platform. The effectiveness of this verification framework is demonstrated by a case study in urban transportation system.

【Key words】Role Goal Process Service(RGPS) framework; Promela modeling; Spin model checking tool; process level meta-model; Linear Temporal Logic(LTL)

DOI: 10.3969/j.issn.1000-3428.2012.15.011

1 概述

随着云计算、对等计算和普适计算等新兴计算模式的不断涌现，软件产业引发了一场变革——软件网络化，其推动了软件工程快速进入网络化时代。

网络式软件系统的核心是需求工程，为了更有针对性的指导网络式软件的需求建模，武汉大学软件工程国家重点实验室提出了RGPS(Role Goal Process Service)需求元建模框架，现已通过ISO认证，成为国际标准[1]。然而，RGPS只是一个需求建模框架，并未提供需求验证方案，对最终结果刻画也缺乏正确性支撑。为了提高验证效率和可靠性，目前国际上正在推行形式化方法。

模型检测[2]是一种常见的形式化分析验证方法，它通过状态空间搜索方法检测一个给定的模型是否满足某种形式化表示的特定性质，包含系统建模、规约建立和系统验证3个主要步骤。模型检测的优点在于可以安全自动地进行验证，这一方法的成功在很大程度上应归功于有效的软件工具支持，如Spin、SMV、CWB等。

Petri网[3]和Promela是目前系统建模常采用的验证模型。Petri网形式化基础良好，但在构造可达树或可达图时，随着系统规模扩大会造成状态空间爆炸。而Promela利用Spin工具，使用on-the-fly和偏序规约等技术，避免状态空间爆炸及减小状态空间，并支持对验证进行优化。为此，本文将Promela模型作为正确性验证模型。

2 Spin模型检测工具

Spin是美国贝尔实验室开发的模型检测工具，凭借其良好的算法设计和非凡的检测能力，成为迄今为止唯一获得ACM软件系统奖的模型检测工具。Spin验证流程见图1。

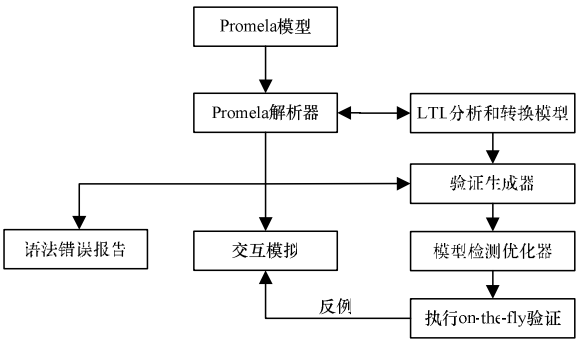

图1 Spin验证流程

基金项目：国家“863”计划基金资助项目“基于代数符号计算的新型软件形式化验证技术和支持工具”(2007AA01Z143)

作者简介：袁开银(1972－)，男，讲师，主研方向：形式化方法，软件工程；郭 瑞，硕士、CCF会员；陆翔升，硕士；吴尽昭，研究员、博士生导师

收稿日期：2011-10-08 **修回日期：**2011-12-09 **E-mail：**zzsh_zz_cn@163.com

用 Promela 建模待验证的模型后，进行语法正确性检查，若没有语法错误，则 Spin 通过交互方式模拟执行，生成一个优化的 on-the-fly 验证程序，该验证程序经模型检测器编译后运行得到验证结果。如果在执行过程中，发现了与正确性说明相背的反例，则返回交互模拟执行状态并确定产生反例的原因。

3 OWL-S 过程模型

3.1 OWL-S

Web 服务本体描述语言(Web Ontology Language for Service, OWL-S)[4]前身是 DAML-S。OWL-S 定义了 Web 服务的子类过程模型，根据不同的粒度层次，过程分为原子过程、组合过程和简单过程。其中，原子过程内部没有控制结构；组合过程包括通过控制流和数据流连接来实现的一系列子过程；简单过程主要以抽象的角度来描述过程模型，是一种不可调用的过程模型。

OWL-S 中定义的控制流有 Sequence、Split、Split-Join、Choice、If-Then-Else、Repeat-Until 等。其中，Sequence 定义一组顺序执行的过程；Split 定义一组同时执行的过程；Split-Join 定义部分同步执行的过程；Choice 定义为在 *m* 个过程中选出 *n* 个过程来执行；If-Then-Else 定义为根据条件选择相应的执行过程；Repeat-Until 定义为在一定条件下循环执行的过程。

每个过程包括 3 个部分：输入，执行条件和结果。结果，包含输出和执行效果。当满足过程执行条件时，则根据输入和当时运行环境的状态产生输出和执行效果，并且过程从执行前状态转移到执行后状态。

3.2 实例分析

交通需求的不断增长和交通网络的日益复杂对城市交通系统的开发提出了新的挑战，使得城市交通系统逐步呈现出典型的复杂软件系统特征。下面以城市交通系统的需求用例为出发点，在 RGPS 需求元建模框架的基础上，根据前述分析，构建城市交通系统过程层元模型，如图 2 所示。

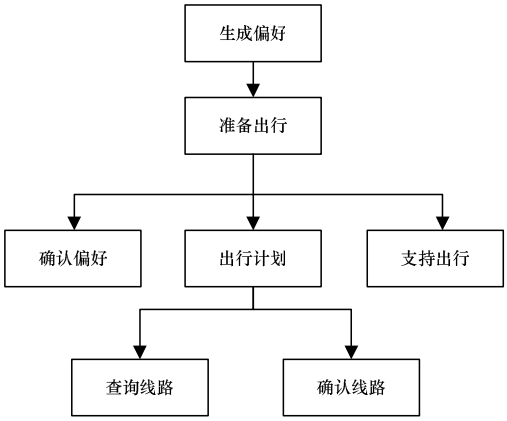

图 2 城市交通系统的过程层元模型

具体来说，实现用户出行计划可以分为原子过程生成偏好、组合过程准备出行、组合过程出行计划和原子过程支持出行 4 个过程，4 个过程间通过顺序进行关联。就组合过程出行计划而言，又由原子过程查询线路和原子过程确认线路组成。

用 OWL-S 描述该过程层模型的部分代码为：

```
<process: CompositeProcess  rdf: resource = "TravelPlan">
  <process: composed of>
    <process: sequence>
      <process: components  rdf: parseType = "collection">
        <process: AtomicProcess  rdf: resource = "Inquire Line">
        <process: AtomicProcess  rdf: resource = "Verify Line">
      </process: components>
    </process: sequnce>
  </process: composed of>
</process: CompositeProcess>
```

4 OWL-S 过程模型的 Promela 建模

Promela[5]是模型检测工具 Spin 提供的一种直观设计规约语言，是用来对有限状态系统进行建模的形式化描述语言。采用进程定义系统的行为，通过通道实现进程间的通信和数据的交换[6-7]。进程描述的基本要素包括赋值语句、条件语句、通信语句、非确定性选择和循环语句等。

4.1 数据类型和通道定义

由于在 Promela 语言中，支持所有传统的编程语言基本数据类型，如整型、字符型、布尔型、数组等。因此可以将 OWL-S 中的前置和后置条件中表示的某些变量定义为全局变量或者局部变量。同时，Promela 语言中支持 mtype 枚举类型，可以用 mtype 来定义消息类型。mtype = {nil, done}，表示某一进程是否执行，或者表示某一个变量的值是否被赋值。

消息之间的通信可以使用全局变量或通道实现。在 OWL-S 描述的过程模型中，主要涉及控制流和数据流。控制流通过在每个组合过程中定义 syncChan 通道来控制子过程的运行。进程控制通道 syncChan 为一二元组，如 chan syncChan = [1] of {int, mtype}，整型 int 表示发送进程的进程号，mtype 表示消息类型进程名。相应地，建立 dataChan 通道表示进程间的数据流。

4.2 过程模型建模

将 OWL-S 过程模型中的每个过程转化为 Promela 语言中的进程，通过 run 动作来实例化进程。

(1)原子过程/简单过程建模

原子过程和简单过程不可再分，在 Promela 模型中用 atomic{}实现：

```
proctype AtomicProcess(chan syncChan, dataChan){
    atomic{…}
    syncChan!_pid,done;}
```

(2)组合过程建模

组合过程是由一组过程(原子过程或组合过程)组合而成，在 Promela 模型中，控制一个组合过程中的多个子过程用通道方式实现，定义为通道 childSync 和 childData。

```
proctype CompositeProcess(chan syncChan, dataChan) {
    chan childSync = [1] of {int, mtype};
    chan childData = [1] of {int};
    …       //控制结构
    syncChan!_pid, done;}
```

(3)控制结构建模

控制结构包括顺序结构、分叉结构、选择结构等，具体建模方法如表 1 所示。

表 1　控制结构的 Promela 建模

控制结构	Promela 建模	控制结构	Promela 建模
顺序结构	proctype Sequence(chan syncChan, dataChan) {…　pid x1,x2; if ::childSync??eval(x1), run M(childSync, childData); -> if ::childSync??eval(x2), run N(childSync, childData);->skip; fi; fi; syncChan!_pid, done;}	选择结构	proctype Choices(chan syncChan, dataChan){ … pid x; if ::m>n->run M(childSync, dataChan); ::m==n-> run N(childSync, dataChan); ::else->skip fi; syncChan!_pid, done;}

分叉结构	proctype Split(chan syncChan, dataChan){
	chan childSync = [3] of {int, mtype};
	pid x1 = run L(childSync, dataChan);
	pid x2 = run M(childSync, dataChan);
	pid x3 = run N(childSync, dataChan);
	syncChan!_pid, done;}
		当循环控制结构	Proctype RepeatWhile(chan syncChan,dataChan){
			…
			do
			::m -> n;
			::else -> break;
			od}

分叉合并结构	proctype SplitJoin(chan syncChan, dataChan){ chan childSync = [2] of {int,mtype}; if ::childSync??eval(childM),runM(childSync, dataChan);-> if ::childSync??eval(childN), run N(childSync, dataChan); fi; fi; syncChan!_pid, done;}	条件选择结构	proctype IfThenElse(chan syncChan, dataChan){ … run condition()-> if :: (condition_return == false) ->goto endproc :: (condition_return == true) ->goto ifproc fi; … endproc; skip;}

4.3　建模算法

由前述内容，可得出 OWL-S 过程模型 Promela 建模算法的伪代码，其中 M 表示 OWL-S 过程模型：

```
OWLS2PROMELA(M)
  define_Type(M)
  define_NeverClaim(M.Prop)
  For each P in M{
      define_proctype(P)
      define_proctypeMain(P) }
  init{ define_Channel(M)
       run proctypeMain()}
```

5　正确性验证

正确性是指：不存在违背断言的情况、不存在死锁、不存在“坏”的循环、满足本文定义的线性时序逻辑(Linear Temporal Logic, LTL)公式，主要考虑 2 种性质：安全性(safety)和活性(liveness)。

5.1　线性时序逻辑

LTL 是一种重要的描述系统约束的形式化方法，由 Manna 和 Pnueli 首次开发出来用以描述系统的并发特性。它以路径(状态序列)作为命题的论断对象，在状态序列上解释其真值。线性时序逻辑可以方便准确地描述系统的重要性质，如安全性和活性。安全性用于说明“坏事情永远都不会发生”；“活性”用于说明“好事情最终会发生”。

Spin 支持所有可以用 LTL 表示的正确性验证要求，LTL 包括&&，||，->，! 等标准命题逻辑连接符以及◇，□，○ 3 种时序算子，例如，◇p：表示 p 在未来某个状态为真；□p：表示 p 在未来所有状态为真。

5.2　验证工具

该工具在 Eclipse 平台下采用 Java 语言编写实现，主要涉及到的辅助工具有：Spin 模型检测工具；OWL-S API[8]，利用 OWL-S API 解析规约的 OWL-S 文件。

用户首先需要对过程层元模型进行描述，形成规约的 OWL-S 文件，利用 OWL-S API 对其进行解析，然后进行 Promela 建模，生成.pml 文件，并利用 LTL 描述模型性质，最后进行 Spin 检测并返回结果。

本工具所包含的主要功能有：过程层元模型 OWL-S 描述；OWL-S 过程模型 Promela 建模；模型的正确性(安全性和活性)验证。

RGPS 过程层元模型正确性验证工具框架和平台初始界面分别如图 3、图 4 所示。

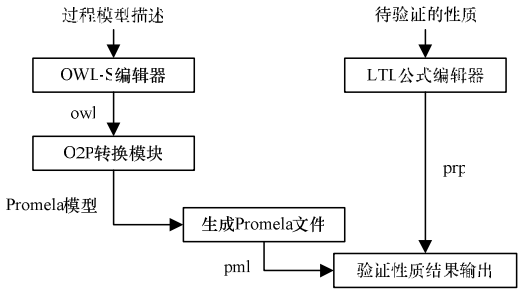

图 3　RGPS 过程层元模型正确性验证工具的框架

图 4　验证工具的平台初始界面

5.3　实例验证

在验证工具中，正确性验证主要考虑 2 个性质：安全性和活性。

以城市交通系统过程层元模型为例，根据前述，将 OWL-S 过程模型 Promela 建模，并验证其正确性，验证结果如表 2 所示。

表 2 正确性验证结果

性质	LTL 输入	验证结果
安全性		Full statespace search for: never claim – assertion violations +(if within scope of claim) cycle checks-(disabled by –DSAFETY) invalid end states +(disabled by never claim) State-vector 108 byte, depth reached 131, errors: 0
安全性反例	[]success	Full statespace search for: never claim – assertion violations +(if within scope of claim) cycle checks –(disabled by –DSAFETY) invalid end states –(disabled by never claim) State-vector 40 byte, depth reached 181, errors: 1
活性	[]<>price&&[]<>week&&[]<>travelticket	Full statespace search for: never claim – assertion violations +(if within scope of claim) acceptance cycles –(fairness enabled) invalid end states +(disabled by never claim) State-vector 52 byte, depth reached 131, errors: 0
活性反例	[]<>week&&[]<>line	Full statespace search for: never claim – assertion violations +(if within scope of claim) acceptance cycles –(fairness enabled) invalid end states –(disabled by never claim) State-vector 104 byte, depth reached 131, errors: 1

6 结束语

由于模型检测具有高度自动化、简单快速、提供反例的特点，得到广泛关注。模型检测的复杂性主要依赖于系统状态空间大小，所以模型检测方法所面临的最大问题是状态空间爆炸和内存不足。Spin 提供了偏序规约和 on-the-fly 等多项优化技术，适合检测日益复杂的 Web 服务。本文介绍如何使用 Promela 语言建模 OWL-S 过程层元模型，并利用 Spin 工具设计开发了正确性验证平台，最后结合具体实例，验证了 RGPS 过程层元模型的正确性。

参考文献

[1] 何克清, 彭 蓉, 刘 玮, 等. 网络式软件[M]. 北京: 科学出版社, 2008.

[2] Clarke E M, Grumbergo P D A. Model Checking[M]. Cambridge, USA: MIT Press, 2001.

[3] 林 强, 胡 昊, 吕 建. 基于对象 Petri 网的 BPEL 建模技术[J]. 计算机工程, 2009, 35(6): 74-75.

[4] 何克清, 马于涛, 刘 婧. 软件网络[M]. 北京: 科学出版社, 2008.

[5] Spin-formal Verification[EB/OL]. [2011-08-20]. http://spinroot.com.

[6] Ankolekar A. Towards a Formal Verification of OWL-S Process Models[C]//Proc. of 2005 International Conference on Web Service. Galway, Ireland: Springer-Verlag, 2005: 37-51.

[7] 李景霞, 肖 政, 侯紫峰. 基于标签 Petri 网的 OWL-S 建模与分析[J]. 计算机工程, 2007, 33(7): 8-10.

[8] OWL-S API[EB/OL]. [2011-09-16]. http://www.mindswap.org/2004/owl-s/api/.

编辑 陆燕菲

(上接第 38 页)

参考文献

[1] Saraswat V. X10 Language Specification[EB/OL]. (2010-08-28). http://x10-lang.org.

[2] Raman R. Efficient Data Race Detection for Async-finish Parallelism[C]//Proc. of the 1st International Conference on Runtime Verification. Heidelberg, Germany: Springer-Verlag, 2010: 368-383.

[3] Christiaens M, de Bosschere K. TRaDe: A Topological Approach to On-the-fly Race Detection in Java Programs[C]//Proc. of 2001 Symposium on Java TM Virtual Machine Research and Technology. Berkeley, USA: USENIX Association, 2001: 105-116.

[4] Savage S. Eraser: A Dynamic Data Race Detector for Multi-threaded Programs[C]//Proc. of the 16th ACM Symposium on Operating Systems Principles. [S. l.]: ACM Press, 1997: 27-37.

[5] Dinning A, Schonberg E. Detecting Access Anomalies in Programs with Critical Sections[C]//Proc. of 1991 ACM/ONR Workshop on Parallel and Distributed Debugging. New York, USA: ACM Press, 1991: 85-96.

[6] Naik M, Aiken A, Whaley J. Effective Static Race Detection for Java[C]//Proc. of 2006 ACM SIGPLAN Conference on Programming Language Design and Implementation. Ottawa, Canada: ACM Press, 2006.

[7] Agarwal S. May-happen-in-parallel Analysis of X10 Programs[C]//Proc. of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming. San Jose, USA: ACM Press, 2007.

[8] Vasudevan N. Compile-time Analysis and Specialization of Clocks in Concurrent Programs[C]//Proc. of CC'09. [S. l.]: IEEE Press, 2009: 48-62.

[9] 桑春雷, 张兆庆. 别名集切片与并行化研究[J]. 计算机工程, 2011, 37(21): 6-10.

[10] Yelick K. Productivity and Performance Using Partitioned Global Address Space Languages[C]//Proc. of 2007 International Workshop on Parallel Symbolic Computation. New York, USA: ACM Press, 2007: 27-28.

编辑 陆燕菲