一种Web服务特征交互自动检测方法

骆翔宇[1,2] 谭 征[3] 董荣胜[3]
(华侨大学计算机科学与技术学院 厦门 361021)[1] (清华大学软件学院 北京 100084)[2]
(桂林电子科技大学计算机与控制学院 桂林 541004)[3]

摘 要 模型检测技术可有效验证Web服务组合的异常情况,如特征交互问题等,但是目前验证过程的自动化程度比较低。为了提高验证的自动化程度,需要将BPEL转化为模型检测工具的输入语言。在分析BPEL语言控制流程的基础上,提出BPEL活动执行的形式化模型,给出活动执行语义,进而分别提出将BPEL流程自动转换为七元组集合以及将这些七元组转化为MCTK(一种我们开发的符号化模型检测工具)输入语言的算法,这些七元组包含了智能体执行过程中有关状态变化的有效信息。实验表明,提出的算法可以有效验证Web服务中的特征交互问题,而且支持认知逻辑规范的验证。

关键词 模型检测,Web服务组合,特征交互,BPEL
中图法分类号 TP311 **文献标识码** A

Automated Detection Method for Web Services Feature Interaction

LUO Xiang-yu[1,2] TAN Zheng[3] DONG Rong-sheng[3]
(College of Computer Science & Technology, Huaqiao University, Xiamen 361021, China)[1]
(School of Software, Tsinghua University, Beijing 100084, China)[2]
(Department of Computer Science, Guilin University of Electronic Technology, Guilin 541004, China)[3]

Abstract Model checking techniques can be effectively applied to the verification of exceptions in Web services composition, such as feature interaction problems, but the verification process is not fully automatic. In order to improve the verification of intelligence level, we need to transform BPEL language into the input language of MCTK, a symbolic model checker developed by us. Based on detailed research on the BPEL control flow, we proposed a formal model for activities implementation and gave the semantics of the BPEL activities implementation. We then respectively developed an algorithm for automatically converting the BPEL process to seven-tuple collections and an algorithm for converting those seven-tuples to input language of MCTK, such seven-tuples included effective information about state changes in the business implementation process. The results show that the proposed algorithm can effectively verify feature interaction in Web Services, and support the verification of epistemic logic.

Keywords Model checking, Web service composition, Feature interaction, BPEL

Web服务的特征交互问题是当今模型检测领域的一个研究热点。目前,刻画Web服务组合的所有标准当中,应用最为广泛的流程描述语言便是BPEL4WS(Business Process Execution Language for Web Services, BPEL)语言[1]。但是,该标准主要用于刻画Web服务流程,并不能保证Web业务流程的正确性及质量。

目前,对于Web服务组合特征交互问题的验证,几乎所有的工作[2,3]都是将BPEL语言转化为Promela,然后将流程实例化,以验证全局消息交换队列的LTL属性,但其自动化程度并不高,并且只是基于线性时序逻辑,更不能验证认知逻辑规范。文献[4]给出了多智能体系统的模型检测方法,用于检测时态认知逻辑规范。文中定义了新的中间语言,但是用法比较繁琐,不能直接描述Web服务。因此,为了实现Web服务中特征交互的智能化验证,本文首先提出将BPEL语言转化为可以完整描述状态转换关系的一组七元组的算法,之后提出根据这些七元组自动生成模型检测工具MCTK[5,6](Model Checking Time and Knowledge)输入语言的算法,从而提高了Web服务验证过程的自动化程度,并可验证时态认知逻辑规范。MCTK是我们开发的时态认知逻辑符号化模型检测工具,时态部分是计算树逻辑CTL*。关于MCTK的详细说明请查阅相关文献[5,6]。

本文第1节提出BPEL活动的形式化模型并给出相应语义;第2节提出相关转换算法;第3节给出例子;第4节是实验及结果分析;最后是对本文的总结。

到稿日期:2010-01-27 返修日期:2010-04-13 本文受国家自然科学基金(60763004),中国博士后科学基金(20090450389),广西科学基金(桂科自0991242),广西青年科学基金(桂科青0728090),广西研究生教育创新计划项目(2008105950812M424)资助。
骆翔宇(1974-),男,博士,副教授,硕士生导师,主要研究方向为模型检测、网络安全等,E-mail:shiangyuluo@gmail.com;谭 征(1982-),男,硕士生,主要研究方向为模型检测、Web Services;董荣胜(1965-),男,教授,硕士生导师,主要研究方向为网络安全、形式化方法、协议工程等。

1 BPEL 流程的形式化模型及活动执行语义

多智能体系统(Multi-agent System,MAS)是研究如何在一组自治的智能体(Agent)之间协调它们的行为,从而完成问题协作求解[7]。本文将每个 Web 服务提供者看成智能体,因此可以将组合 Web 服务流程看成是由多智能体组成的分布式软件系统。这样,研究目标就由对组合业务的特征交互验证转化为对多智能体软件系统的特征交互检测。为实现自动化验证,我们需要自动记录智能体在执行过程中的状态变化情况。为此,首先提出 BPEL 流程的形式化模型及活动执行的相关语义。

1.1 智能体执行的形式化模型

由于智能体需要与其他智能体及环境进行交互,本文将这种交互通信看成一个消息机制,为此引入通道的概念。智能体通过通道来发送和接收消息,从而使自己的状态发生迁移。本文假设所有通道名皆不相同,并且每个智能体皆有两个单向通道与其"相连",用于发送及接收消息。于是,一个智能体的执行可以抽象为一个五元组 BFM(BPEL Finite-state Machine):$BFM=\langle\Omega,O,\delta,P_0,F\rangle$。其中,

Ω:局部状态的有限集合;

O:智能体可观察变量集合,其中包括:

O_e:通道变量的有限集合;

O_l:链接变量的有限集合;

O_g:条件变量的有限集合;

δ:迁移关系 $\Omega\times\Delta\times\Omega$;

Δ:$O_e\times O_l\times O_g\times channel\times dir$;

$P_0\in\Omega$:初始状态;

$F\in\Omega$:终结状态集合。

$channel$ 表示与智能体相联系的通道,dir 表示通道中的信息是进入智能体还是离开智能体。因此,BPEL 每个活动的执行均产生一个相对应的迁移关系,即一个七元组 δ,本文称之为迁移七元组。其基本语法格式为($cur(state)$,O_e,O_l,O_g,$channel$,dir,$next(state)$)。为了便于研究,我们定义几个函数:$cur(state)$表示活动的初始状态,也就是活动刚开始执行时智能体的状态;$next(state)$表示活动执行后智能体的状态;$c(source_state,dest_state)$表示与智能体连接的通道名,通道中信息的传输可改变智能体状态;$value(variable)$表示通道中的信息;$dir(channel)$表示通道中信息传送的方向,即信息离开智能体还是到达智能体。

1.2 BPEL 活动执行语义

基本活动:

· receive

〈receive〉活动从流程的外部伙伴获取数据,并将其保存到流程变量中。其基本语法格式为

〈receive partnerLink=L portType=PT operation=Op variable=V〉〈/receive〉

Receive 活动如图 1 所示。

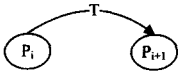

图 1 Receive 活动

Receive 迁移的语义:

$\delta_{receive}=\{T|P_i,P_{i+1}\in\Omega\wedge V\in O_e\wedge cur(state)=P_i\wedge next(state)=P_{i+1}\wedge c(P_i,P_{i+1})=PT\wedge value(PT)=V\wedge dir(PT)=IN\}$

产生七元组:

$T=(P_i,V,\varepsilon,\varepsilon,PT,IN,P_{i+1})$

· reply

〈reply〉活动发送消息给伙伴来应答通过 receive 活动所接收到的消息。其迁移图与图 1 一样。基本语法格式为

〈reply partnerlink=L portType=PT operation=Op variable=V〉〈/reply〉

Reply 迁移的语义:

$\delta_{reply}=\{T|P_i,P_{i+1}\in\Omega\wedge V\in O_e\wedge cur(state)=P_i\wedge next(state)=P_{i+1}\wedge c(P_i,P_{i+1})=PT\wedge value(PT)=V\wedge dir(PT)=OUT\}$

产生七元组:

$T=(P_i,V,\varepsilon,\varepsilon,PT,OUT,P_{i+1})$

· 同步 invoke

〈invoke〉活动允许业务流程同步或异步调用由合作伙伴提供的服务,服务实现可以是单向或请求-响应操作,分别对应异步 invoke 与同步 invoke。其基本语法格式为

〈invoke partnerLink=L portType=PT operation=Op inputVariable=IV outputVariable=OV〉〈/invoke〉

异步 Invoke 活动如图 2 所示。

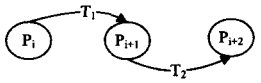

图 2 异步 Invoke 活动

同步 Invoke 迁移的语义:

$\delta_{invoke}=\{T_1|P_i,P_{i+1}\in\Omega\wedge IV\in O_e\wedge cur(state)=P_i\wedge next(state)=P_{i+1}\wedge c(P_i,P_{i+1})=PT\wedge value(PT)=IV\wedge dir(PT)=OUT\}\wedge\{T_2|P_{i+1},P_{i+2}\in\Omega\wedge OV\in O_e\wedge cur(state)=P_{i+1}\wedge next(state)=P_{i+2}\wedge c(P_{i+1},P_{i+2})=PT\_back\wedge value(PT\_back)=OV\wedge dir(PT)=IN\}$

产生七元组:

$T_1=(P_i,IV,\varepsilon,\varepsilon,PT,OUT,P_{i+1})$

$T_2=(P_{i+1},OV,\varepsilon,\varepsilon,PT\_BACK,IN,P_{i+2})$

· 异步 invoke

〈invoke partnerLink=L portType=PT operation=Op inputVariable=IV〉〈/invoke〉

其迁移图同图 1。

异步 Invoke 迁移的语义:

$\delta_{invoke}=\{T|P_i,P_{i+1}\in\Omega\wedge IV\in O_e\wedge cur(state)=P_i\wedge next(state)=P_{i+1}\wedge c(P_i,P_{i+1})=PT\wedge value(PT)=IV\wedge dir(PT)=OUT\}$

产生七元组:

$T=(P_i,IV,\varepsilon,\varepsilon,PT,OUT,P_{i+1})$

结构活动:

· switch

〈switch〉活动与传统的结构化语言的功能类似,从一组分支情况中选择一个特定的活动分支加以执行。其基本语法格式为

〈switch〉

〈case condition=C_1〉…〈/case〉

〈case condition=C_2〉…〈/case〉

…

〈otherwise〉…〈/otherwise〉

〈/switch〉

Switch 活动如图 3 所示。

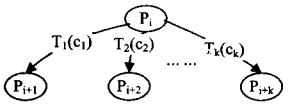

图 3 Switch 活动

Switch 迁移语义(假设其有 N 分支)：

$\delta_{switch}=\otimes_{1\leqslant k\leqslant n}\{T_k \mid P_i, P_{i+k}\in\Omega \wedge C_k\in O_g \wedge C_k=\text{true} \wedge cur(state)=P_i \wedge next(state)=P_{i+k}\}$

产生七元组：

$T_1=(P_i,\varepsilon,\varepsilon,C_1=\text{true},\varepsilon,\varepsilon,P_{i+1})$

$T_2=(P_i,\varepsilon,\varepsilon,C_2=\text{true},\varepsilon,\varepsilon,P_{i+2})$

…

$T_k=(P_i,\varepsilon,\varepsilon,C_k=\text{true},\varepsilon,\varepsilon,P_{i+k})$

⊗表示互斥，即在一个流程实例中，只有一个迁移被触发。

· pick

〈pick〉活动会等待一组相互排斥事件中的一个事件的发生，然后执行与发生的事件相关联的活动。其基本语法格式为：

〈pick〉

〈onmessage variable = m_1 porttype = PT〉…〈/onmessage〉

〈onmessage variable = m_2 porttype = PT〉…〈/onmessage〉

…

〈/pick〉

Pick 活动如图 4 所示。

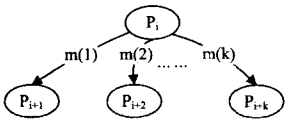

图 4 Pick 活动

Pick 迁移语义(假设存在 n 个〈onmessage〉活动)：

$\delta_{pick}=\otimes_{1\leqslant k\leqslant n}\{T_k \mid P_i, P_{i+k}\in\Omega \wedge m_i\in O_e \wedge cur(state)=P_i \wedge next(state)P_{i+k} \wedge c(P_i,P_{i+k})=PT \wedge dir(PT)=IN\}$

产生七元组：

$T_1=(P_i,m_1,\varepsilon,\varepsilon,PT,IN,P_{i+1})$

$T_2=(P_i,m_2,\varepsilon,\varepsilon,PT,IN,P_{i+2})$

…

$T_k=(P_i,m_k,\varepsilon,\varepsilon,PT,IN,P_{i+k})$

⊗表示互斥，即在一个流程实例中，只有一个迁移被触发。

2 转换算法

本节给出两个算法：算法 1 根据 BPEL 流程产生能够正确反映状态转换关系的七元组集合，以方便算法 2 利用这些迁移七元组自动产生 MCTK 代码，进而验证组合服务。以下首先提出 BPEL 业务流程到迁移七元组的转换算法 B2T：

算法中〈…Σ…〉表示任一活动的开始，∝〈…Σ…〉表示执行该活动并按本文 1.2 节介绍产生对应七元组。对于迁移元组($cur(state)$, O_e, O_L, O_g, $channel$, dir, $next(state)$)，我们标记元组 $tuple[j]$ 的第 i 项为 $tuple[j]^{(i)}$。

算法 B2T B2T(ACT(S), STAT(S))

```
输入：描述 Web 服务的 BPEL 源码
输出：迁移七元组集合
Init Act. Top=〈sequence〉; Init Stat. Top=sleep; cur(state)=sleep;
scan from current activity 〈…Σ…〉 to the last activity;
  for each element 〈…Σ…〉 do
  { /* 将活动及活动执行后的状态入栈 */
    Act. Top ++ & Act. Push 〈…Σ…〉;
    ∝〈…Σ…〉;
    Stat. Top ++ & Stat. Push(next(state));
    Cur(state)=next(state);
    If(〈…Σ…〉≡〈otherwise〉∪〈case〉)
    { /* 活动栈及状态栈同步弹栈直到 switch */
      do
      {
        Act. Pop & Act. Top--;
        Stat. Pop & Stat. Top--;
      }while(Act. Top ! ≡〈switch〉);
      Cur(state)=Stat. Top;
    }
    If(〈…Σ…〉≡〈onmessage〉)
    { /* 活动栈及状态栈同步弹栈直到 pick */
      do
      {
        Act. Pop & Act. Top--;
        Stat. Pop & Stat. Top--;
      }while(Act. Top ! ≡〈pick〉);
      Cur(state)=Stat. Top;
    }
}
```

其中，Act(s)表示活动栈，stat(s)表示状态栈，“≡”表示简单的字符串匹配，$A\equiv B$ 当且仅当 A，B 完全相同。cur(state)表示将要产生的迁移七元组的第一项。对于状态的命名，我们在交互的信息名前加上“send_”或者“get_”，表示此智能体的状态为已发送了某信息状态或者已接收了某信息状态。产生正确的分支结构非常重要，本文采用双栈算法保证了正确的七元组的产生。由于需要从上向下扫描所有活动，并在每个活动处做出相应处理，因此假设活动数为 n，则算法时间复杂度为 $O(n)$。

通过 B2T 算法，可以得到一组迁移七元组。由于编码的工作是枯燥并且机械的，为了实现进一步的自动验证，我们提出 T2M 算法。该算法以 B2T 算法生成的一组七元组作为输入，自动生成 MCTK 代码的主要部分。在算法开始以前，首先进行预处理，去掉不在本文所提活动之内的其他活动，将所有的〈variable〉标记中的标记名加入 mtype 集合。此集合初始化为 none。

算法 T2M(假设共产生 n 个七元组)

```
/* 步骤 1 产生描述状态迁移的代码片段 */
for each tuple[i](i≤n)
{ Next(state):=case
  (state=tuple[i](1)) &(ObsPrm_tuple[i](5). mtype= tuple[i](2)):
  tuple[i](7);
}
/* 步骤 2 假设满足 tuple[i](6)=OUT 的元组共有 m 个，则：*/
```

```
for each tuple[j]( j≠i∧j≤m≤n)
{
  if(tuple[i](1) == tuple[j](7)) then
  /*产生描述通道中信息变化的代码片段:*/
  {
    next(tuple[i](5).mtype):= case
      (state= tuple[i](1)) &
      (ObsPrm_tuple[j](5).mtype=tuple[j](2)):tuple[i](2);
      next(tuple[i](5).source):= case
      (state= tuple[i](1)) &
      (ObsPrm_tuple[j](5).mtype=tuple[j](2)):
        tuple[i](5).sourceAgent;
    next(tuple[i](5).dest):= case
      (state=tuple[i](1)) &
      (ObsPrm_tuple[j](5).mtype=tuple[j](2)):
        tuple[i](5).destAgent;
}
}
```

由于算法需要针对某一七元组扫描其余 $n-1$ 个七元组以找到满足 if 判断条件的元组集合，因此在最差情况下其时间复杂度为 $O(n!)$。

3 实验

本节应用我们开发的基于时态认知逻辑的模型检测工具 MCTK 来检测文献[8]中在线股票交易系统 STS 服务中可能存在的特征交互问题。我们以其中最为复杂的一个智能体——股票交易服务商为例。首先生成描述其运行的 BPEL 代码，将其预处理，之后将代码作为 B2T 算法的输入，运行完毕得到一系列迁移七元组，将其作为 T2M 算法的输入，得到 MCTK 代码，最后输入 MCTK，得到实验结果。下面简要介绍这两个算法的处理过程。

3.1 迁移元组的形成阶段：

根据算法 B2T，将自动生成的 BPEL 代码转换为如下七元组(限于篇幅，只列出一小部分代码片段)：

```
〈sleep,sStotraReqMsg,…,Tra_TCI,IN,get_sStotraReqMsg〉
〈get_sStotraReqMsg,sStockquoteReqMsg,…,Tra_TQO,OUT,send_
sStockquoteReqMsg〉
〈send_sStockquoteReqMsg,sStockofferMsg,…,Tra_TQI,IN,get_sS-
tockofferMsg〉
〈send_sStockquoteReqMsg,sStockNAMsg,…, Tra_TQI,IN,get_sS-
tockNAMsg〉
……
```

3.2 MCTK 代码的形成阶段

首先进行预处理，包括搜索本 Web 服务的 WSDL 代码，抽取代码中的〈portTypt name〉，找到所有与本智能体相联系的通道，包括入通道和出通道。其次，根据算法 T2M，将上述七元组转化为 MCTK 输入语言，得到 MCTK 描述状态迁移的代码(限于篇幅，下面只列出一小部分代码片段)：

```
VAR
state:{sleep,send_StockquoteReqMsg,
      get_StockofferMsg,send_StockACKMsg,…};
ASSIGN
next(state):= case
  ……
  (state=send_StockquoteReqMsg) &
  (ObsPrm_ch2.mtype=sStockofferMsg):get_StockofferMsg;
  ……
  1:state;esac;
```

上述代码说明，若智能体状态为发送了股票查询请求，并且智能体通过通道 ch2 接收到了相关股票信息，则进入得到股票信息的状态。

下面是描述通道中信息变化的代码片段：

```
Next(ch1.mytype):=case ……
  (state=get_BankofferMsg)&(ObsPrm_ch4.mytype=sBankoffer-
  Msg):{sStockACKMsg,sStockNACKMsg};
  Next(ch1.source):=case
  (state= get_sStotraReqMsg)&(ObsPrm_ch1.mytype= sStock-
  quoteReqMsg):Trading_PLT;
  (state=get_BankofferMsg)&(ObsPrm_ch4.mytype=sBankoffer-
  Msg):Trading_PLT;Next(ch1.dest):=case
  (state= get_sStotraReqMsg)&(ObsPrm_ch1.mytype= sStock-
  quoteReqMsg):Quote_PLT;
  ……
```

上述代码说明，当 ch4 通道中的银行提供的查询信息进入股票交易智能体后，在下一时刻 ch1 中的信息为向股票查询智能体发送是否确认此股票的信息。此外，代码还说明了当股票交易服务商接收到交易请求后，ch1 通道中股票查询请求信息的源智能体为股票交易服务商，目标智能体为股票查询服务商。

4 实验结果分析

实验平台是带 1.25G 内存的清华同方电脑，Ubuntu 5.04 Linux 系统。AgentA 代表股票交易服务商，AgentB 代表股票更新服务商，AgentC 代表银行(AgentC2 代表另一间银行)，AgentD 代表用户。我们在自动生成的代码中手工写入以下规范验证：

性质 1 死锁(Deadlock)

规范：$AG(a \rightarrow \rightarrow EGa)$

其中，a 代表智能体状态迁移中的任一状态。此规范表示一旦智能体进入某种状态，未来它将不会进入终结状态。此规范验证结果为 true，说明不存在死锁现象。当 a 代表不同状态时，验证时间也不相同。本次实验选取 a 代表 send_sStockquoteReqMsg 状态，验证时间为 1.223s。

性质 2 调用错误(Invocation error)

规范：AG(AgentD.state=send_sCancelMsg →

→EF(AgentC.state=get_sBankapproveMsg))

此规范的含义是指任何情况下，如果用户终止股票交易操作，则银行资金服务将不能完成。此规范验证结果为 false，说明可能会出现这样的问题，即在用户终止操作前，资金操作已经执行完毕。验证时间为 2.725s。

性质 3 竞态条件(Race condition)

规范：AG(AgentC.state = get_BankreqMsg → →EF(AgentB.state=get_StockQuoteReqMsg))

此规范说明一旦银行交易被激活，它就必须在下一轮股票价格更新前完成，否则将出现实际交易价格与最初的股票报价不一致的问题。此规范验证结果为 false，说明会出现股票更新操作完毕而银行交易未完成的现象。验证时间为 2.247s。

性质 4 循环状态(loop situation)

(下转第 119 页)

参 考 文 献

[1] Allen J. Maintaining knowledge about temporal intervals[J]. Communications of the Association for Computing Machinery, 1983,26:832-843

[2] Lin T Y. Granular Computing: From Rough Sets and Neighborhood Systems to Information Granulation and Computing in Words[C]// European Congress on Intelligent Techniques and Soft Computing. September 1997:1602-1606

[3] Zadeh, et al. Granular Computing and Rough Set Theory[J]. Lecture notes in computer science, 2007, 4585:1

[4] Pfennigschmidt S, Voisard A. Handling Temporal Granularity in Mobile Services[C]// IEEE International Conference on Wireless and Mobile Computing, Networking and Communications 2009 (WIMOB 2009). Oct. 2009:295-300

[5] Zhang Xue-jie, Ng Kam-wing. A temporal partitioning approach based on reconfiguration granularity estimation for dynamically configurable systems[C]// IEEE International Conference on Field-Programmable Technology (FPT). 2003,12:344-347

[6] 万静,郝忠孝. 具有多时间粒度的强全序时态模式中多值依赖问题研究[J]. 计算机研究与发展,2008,6:1064-1071

[7] Egidi L, Terenziani P. A flexible approach to user-defined symbolic granularities in temporal databases[C]// Proceedings of the 2005 ACM symposium on Applied computing (SAC'05). Santa Fe, New Mexico, USA, March 2005:592-597

[8] Merlo I, Bertino E, Ferrari E, et al. Querying multiple temporal granularity data[C]// Seventh International Workshop on Temporal Representation and Reasoning (TIME 2000). July 2000: 103-114

[9] Bertino E, Ferrari E, Guerrini G, et al. Navigating through multiple temporal granularity objects[C]// Eighth International Symposium on Temporal Representation and Reasoning (TIME 2001). June 2001:147-155

[10] Belussi A, Combi C, Pozzani G. Formal and conceptual modeling of spatio-temporal granuzlarities[C]// Proceedings of the 2009 International Database Engineering & Applications Symposium (IDEAS '09). Cetraro, Calabria, Italy, September 2009:275-283

[11] Orgun M A, Liu Chuchang, Nayak A C. Representation and Integration of Knowledge based on Multiple Granularity of Time using Temporal Logic[C]// IEEE International Conference on Information Reuse and Integration. Sept. 2006:256-261

[12] Cotofrei P, Stoffel K. Temporal granular logic for temporal data mining[C]// IEEE International Conference on Granular Computing. 2005:417-422

[13] Chen Xiao-qing, Qiu Tao-rong, Liu Qing, et al. The Framework of Temporal Granular Logic Based on Information System[Z]. IEEE CGrC, 2006:604-606

[14] 王路帮,汤庸,余阳. 基于 BCDM 的双时态关系代数[J]. 计算机研究与发展,2004,11:1950-1953

[15] 叶小平,汤庸. 时态变量"Now"语义及相应时态关系运算[J]. 软件学报,2005,5:838-845

(上接第 109 页)

规范: AG((AgentC. state = sleep & AgentC2. state = sleep) → →EF (AgentC. state = send _ sFundReqMsg & AgentC2. state = send_sFundReqMsg))

此规范的含义是指不存在这样的情况,即 AgentC 当前状态为发送资金请求,而 AgentC2 当前状态也为发送资金请求,AgentC 和 AgentC2 相互为指定更换的服务。验证结果为 false,则说明会出现这样的情况。验证时间为 2.114s。

除了上述 4 个时态逻辑规范,我们还验证了如下时态认知逻辑规范: AG(ch5. mtype = sApproveMsg→AF(AgentD K (ch2. mtype = sStockapproveMsg & ch4. mtype = sBankapproveMsg)))

此规范表明,当用户得到了股票交易批准,则用户知道股票查询服务商发送给股票交易服务商以及银行发送给股票交易服务商的信息都是同意交易,即具有知识推理能力。规范验证结果为 true。验证时间为 4.138s。

结束语 由于 Web 服务组合的复杂性,其自动化验证的实现是一个困难、复杂的过程。相比目前国内外将 BPEL 转为模型检测工具 Spin 的输入语言 Promela 的方法而言,本文提出的验证方法的转化过程的自动化程度得以提高,并且可验证智能体所有可能执行的路径,比传统工作中只能验证全局消息队列的 LTL 性质的方法表达能力更强。我们的方法不仅可以验证时态逻辑,还可以验证认知逻辑规范,从而使其具有知识推理的能力。

但是,由于被验证规范需要手工书写,因此如何自动生成被验证规范便成为我们以后的工作之一。另外,由于 BPEL 语言的复杂性,目前我们较少考虑如何处理数据流,这对我们算法的适用范围产生了一定影响,这也将是我们后续工作的重点。

参 考 文 献

[1] Oasis S. Web service Business Process Execution Language (WSBPEL) TC[S]. Web service Business Process Execution Language Version 2.0, 11 April 2007:122-138

[2] Fu X, Bultan T, Su J. Analysis of Interacting BPEL Web Services[C]// Proceeding of the 13th International World Wide Web Conference. ACM Press, 2004:621-630

[3] Raman K. Formal Analysis of Web Service Composition[D]. University of Trento, March 2007

[4] Lomuscio A, Raimondi F. MCMAS: A model checker for multiagent systems[C]//TACAS'06. 2006, 3920:450-454

[5] Su K. Model checking temporal logics of knowledge in distributed systems[C]// Proceedings of the Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence. AAAI Press/The MIT Press, 2004:98-103

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

[7] Fagin R, Halpern J, Moses Y, et al. Reasoning about knowledge [C]// Proceedings of the 1986 Conference on Theoretical Aspects of Reasoning About Knowledge. Monterey, California, 1986:1-17

[8] Zhang J, Yang F, Su S. Detecting feature interactions in Web services with model checking techniques[J]. The Journal of China Universities of Posts and Telecommunications, 2007, 14(3): 108-112