软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(9):2020−2041 [doi: 10.3724/SP.J.1001.2013.04366]

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

基于接口语义契约的 Web 服务测试数据生成*

侯可佳[1,2], 白晓颖[1,2], 陆 皓[1,2,3], 李树芳[1,2,4], 周立柱[1,2]

[1](清华大学 计算机科学与技术系,北京 100084)

[2](清华信息科学与技术国家实验室(筹),北京 100084)

[3](南京陆军指挥学院 作战实验中心,江苏 南京 210045)

[4](中国酒泉卫星发射中心,甘肃 酒泉 732750)

通讯作者: 侯可佳, E-mail: hkj09@mails.tsinghua.edu.cn

摘 要: 测试是 Web 服务质量保证的主要手段之一,测试自动化是降低测试代价的有效途径,测试数据生成是测试自动化研究的一个重要内容.提出采用本体及规则的知识描述语言,建立服务接口的语义契约模型(interface semantic contract,简称 ISC),并探讨了基于 ISC 的测试数据生成技术,给出了分区生成算法以及测试数据生成的模拟退火算法.实验结果表明,与随机测试相比,该方法能够采用 10%的测试用例数量达到同样的测试覆盖率;在同样的测试用例数量上,最高可提高 50%的测试覆盖率.

关键词: 服务测试;数据生成;本体模型;OWL-S

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 侯可佳,白晓颖,陆皓,李树芳,周立柱.基于接口语义契约的 Web 服务测试数据生成.软件学报,2013,24(9):2020−2041. http://www.jos.org.cn/1000-9825/4366.htm

英文引用格式: Hou KJ, Bai XY, Lu H, Li SF, Zhou LZ. Web service test data generation using interface semantic contract. Ruan Jian Xue Bao/Journal of Software, 2013,24(9):2020−2041 (in Chinese). http://www.jos.org.cn/1000-9825/4366.htm

Web Service Test Data Generation Using Interface Semantic Contract

HOU Ke-Jia[1,2], BAI Xiao-Ying[1,2], LU Hao[1,2,3], LI Shu-Fang[1,2,4], ZHOU Li-Zhu[1,2]

[1](Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

[2](Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China)

[3](Operation Research Center, Nanjing Army Command College, Nanjing 210045, China)

[4](Jiuquan Satellite Launch Center, Jiuquan 732750, China)

Corresponding author: HOU Ke-Jia, E-mail: hkj09@mails.tsinghua.edu.cn

Abstract: Testing is one of the primary methods for Web service quality control. Test automation is necessary to enhance test productivity and quality while reducing test effort. Test data generation is a critical issue of automated testing. The paper proposes a method called interface semantic contract (ISC) for modeling services' exposed functionalities using ontology and rule language. Algorithms are developed to generate input partitions and test data based on ISC. Case studies are exercised to illustrate the proposed approach. The results show that compared with conventional random testing, the proposed approach can enhance test coverage by 50% with the same number of test cases, and reduce test effort by 90% to reach the same test coverage.

Key words: service testing; date generation; ontology model; OWL-S

Web 服务是实现面向服务计算的主要技术,提供了一套基于标准的服务发布、发现和组合机制以及整合分

* 基金项目: 国家自然科学基金(61073003); 国家重点基础研究发展计划(973)(2001CB302505); 北京市自然科学基金(4132062); 清华大学自主科研计划

收稿时间: 2012-04-19; 修改时间: 2012-10-19; 定稿时间: 2012-12-27

布式服务组件的开放平台.Web 服务在提供灵活多变的应用架构的同时,也对测试提出了更高要求:首先,测试工作量大.在互联网环境下,相同功能的服务通常会存在多种不同版本,这些版本由不同的服务提供商提供,通常需要对这些服务进行测试和横向比较,对这些功能相同的不同服务进行逐个测试是一项重复而繁琐的工作,由测试人员手动完成不仅工作量大,而且极易出现错误;其次,服务在线演化.在 Web 服务发布后,服务提供商可以随时对其提供的服务进行修改完善,服务使用者需持续在线进行回归测试,以保证所需功能的可用性和正确性,由此对测试的时效性提出了更高的要求,人工测试难以满足;第三,Web 服务通常通过可编程服务接口的方式访问,缺少图形化用户操作界面,使得人工测试困难.因此,Web 服务自动化测试成为一个必然的发展趋势.Web 服务接口信息均遵照基于 XML 的标准规范进行描述,可以通过工具自动分析其相关文档获取数据结构、接口操作及其语义等必要的测试信息,使服务自动理解、测试自动生成以及基于 Web 服务标准的测试自动执行成为可能.通过测试自动化,可有效增强测试用例、测试脚本等测试资产的可重用性,提高测试效率和服务质量,降低测试代价.

本文提出了一种基于服务接口语义契约(interface semantic contract,简称 ISC)的测试方法以及基于本体的测试数据生成技术.服务接口描述,如 Web 服务的 WSDL(Web service definition language),从数据和功能两个方面定义软件所支持的操作,描述了服务运行环境、输入/输出参数等测试信息,可将其视为设计服务测试的功能需求,作为服务测试的依据.但是,基本服务规约的描述能力非常有限,尤其缺乏服务操作和集成的语义信息.借鉴语义 Web 服务及契约设计的思想,ISC 从两个方面增强和扩展服务操作的语义表达能力:利用本体模型,建立数据和服务统一的概念模型;借助语义规则定义,建立数据和服务的约束模型.通过本体和规则的知识描述语言,ISC 增强了服务的语义表达能力,为服务的理解、协同、验证与确认提供更丰富的语义信息.

测试数据生成是测试自动化的一个重要研究内容.传统的数据生成方法多局限于特定的数据类型,如数值型变量,或是依据需求的形式化规约如状态机模型.

基于 ISC,本文从本体模型和约束模型两个方面,探讨了测试数据的自动生成技术:本体模型基于集合操作,定义了概念之间的交、并、补、继承、等价、互斥等关系,为数据等价类划分提供了依据;约束模型基于谓词逻辑,定义了数据及其属性之间在基数、取值等方面的相互依赖和限制关系,是多参数数据生成时需要满足的条件.本文将基于 ISC 的测试数据生成转化为搜索问题,提出了基于搜索算法的数据分区和测试数据生成方法.图 1 显示了测试数据生成的主要过程:基于服务接口描述,定义接口的语义契约模型 ISC;基于 ISC 本体模型,生成测试分区;基于 ISC 约束模型,生成特定服务接口操作的一组测试数据.

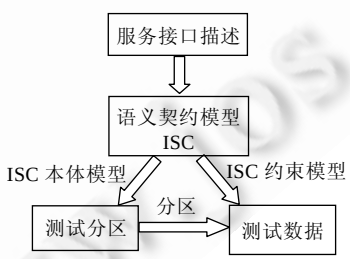

Fig.1 Test data generation using interface semantic contract

图 1 基于接口语义契约测试数据生成

1 相关工作

1.1 Web服务测试

近年来,SOA(service oriented architecture)软件测试得到了研究者的广泛关注.由于基于服务的系统强调服务自主计算、动态协同的能力,自动化成为基于服务的系统测试的必然趋势,需要在服务在线演化、动态重组重构的过程中,持续在线验证与确认服务在功能、性能各方面满足应用需求[1].

测试数据的自动生成是测试自动化的重要研究内容之一.在 Web 服务测试数据生成的研究中,先后提出了基于 WSDL 的数据生成、数据变异技术以及语义测试技术等相关研究.

文献[1]分析 Web 服务的 WSDL 描述,定义 3 种依赖关系:输入依赖、输出依赖以及输入/输出依赖;测试用例的生成以上述 3 种依赖关系为基础,通过测试数据生成、独立测试操作生成、操作流生成以及测试说明生成来完成.文献[2]为服务的输入数据定义形式化模型,并从服务的 WSDL 描述中提取该模型,根据提取的数据模型生成测试数据.

文献[3,4]探讨了 Web 服务变异测试技术.文献[3]提出一种数据变异的方法,改变服务请求信息中的数据取值,并对服务响应信息进行分析.提出了针对不同的数据类型的变异算子,例如最大值算子、最小值算子和零算子等,分别把原来的值改为相应类型的最大值、最小值和 0.文献[4]首先随机生成若干测试数据,然后应用变异算子生成契约的变异体,最后通过这些变异体完成对测试数据的选择.测试数据集可以通过契约变异分数 $D/(M_C–E_C)$来评价,其中,M_C 为变异总数,E_C 为等价变异数,D 为测试数据杀死的变异数.

文献[5−9]将语义技术应用于测试生成中.文献[5]定义测试本体模型,将其作为测试契约,描述测试领域内的相关知识.通过对 Web 服务语义模型中数据类属性以及类之间关系的分析生成数据分区,通过类计算和规则推理进行数据分区和数据值的完整性和一致性检查.文献[6]利用语义规则描述被测服务的预期行为,并通过规则推理产生被测服务的测试断言.文献[7]以 OWL-S(Web ontology language for services)描述的服务规约为基础,将服务的本体模型转换为 Petri 网模型,并为 Petri 网模型定义本体模型,丰富其语义信息,这些语义信息包括输入/输出数据、前置条件和执行效果.测试用例的生成包括两个方面:1) 遍历 Petri 网获取测试步骤;2) 利用服务本体的推理功能生成测试数据.文献[8]将语义技术应用于服务接口的领域知识建模,提出模型即服务方法,将常用的模型工具,例如决策表、状态机等,集成于一个标准服务接口,让用户可以按需调用,方便对特定属性的形式化建模.文献[9]提出 4 种容易出现在 Web 服务 OWL-S 语义描述中的错误:数据错误、状态错误、进程错误以及数据依赖错误,针对其中的数据错误提出两类变异算子:OWL-S 输入类型变异算子将 OWL-S 进程的输入参数类用其等价类、父类、子类等替换原数据类;OWL(Web ontology languge)本体变异算子通过更改类的属性、约束以及类之间的关系来改变类的定义.

本文在文献[5,8]的基础上,通过分析被测服务的接口描述为服务建立语义契约模型(ISC),模型包含被测服务的输入/输出参数、控制依赖、数据依赖等信息,有关输入/输出参数的数据本体模型包含了被测服务丰富的领域信息.在数据本体模型的基础上,利用约束规则定义多数据间相互依赖和限制关系,建立约束模型.依据数据本体模型中类关系结构以及属性取值为数据建立测试分区,在分区基础上,结合约束模型为被测服务生成测试数据.

1.2 基于契约的测试

在传统的编程技术中,开发人员为了预防可能发生的各种问题,会在程序中编写大量判定语句以保证程序正常运行.但程序中盲目增加的判定语句会大大增加程序复杂度,降低程序性能.针对该问题,文献[10]提出了基于契约的设计思想(design by contract,简称 DbC).契约作为软件组件或合作者之间的协议,代替了程序中原有的判定语句,降低了程序的复杂度.基于契约的设计思想已被广泛应用于面向对象的软件设计以及基于组件的软件开发技术中.

基于契约的设计思想也被应用于软件测试中,文献[11]从可观察性和可诊断性两个方面探讨了契约对提高软件可测试性所提供的支持.可观察性是系统内部错误能够被发现的概率,可诊断性则用来衡量定位错误以及隔离错误的便捷性.该研究指出,契约可以代替传统的测试断言装载到被测软件中,在软件运行过程中探测错误并定位错误,降低了错误定位和隔离的代价.通过实验得出如下结论:

(1) 契约能够有效发现 80%植入错误,极大地降低了测试断言的开发成本;

(2) 与采用传统的测试断言相比,基于契约的测试方法将程序错误的可诊断性提高了近 8 倍.

契约作为软件描述的一部分,定义了函数输入和执行的标准,不仅有助于提高程序设计质量,还能提高软件测试的自动化程度.目前,软件描述语言已经被广泛应用于软件预期行为的描述,形式化软件描述包括 Z 语言、

有限状态机(finite state machine,简称 FSM)、谓词逻辑、线性时序逻辑以及代数描述等.1989 年文献[12]提出一种基于软件描述的测试数据自动生成技术,该研究将测试数据的选择技术分为两大类:基于错误的和基于故障的数据选择技术,在其定义的测试用例选择标准中包含了错误检测标准和故障检测标准.目前,基于软件描述的测试已受到越来越多研究人员的关注,开发出 UML(unified modeling language),ADL(architecture description language),SDL(specification description language)等更多形式化或半形式化的描述语言,从多个方面实现测试自动生成.

在 SOA 中,服务提供商和用户之间存在着一种基于契约的合作模式,因此,基于契约的测试技术也适用于 Web 服务测试.文献[13]将 UML2 中的协议状态机描述引入到 Web 服务描述 WSDL 中,增加对服务前置/后置条件以及数据/进程约束的描述,增强了 Web 服务的可验证性.文献[14]指出,在 SOA 中,契约描述存在多种表达方式:实现级、XML 级以及模型级契约描述.在实现级中,利用服务编程语言将契约定义为布尔表达式,用以描述服务的前置条件和后置条件.XML 级契约对 WSDL 等基于 XML 的 Web 服务协议进行了扩展,增加了服务操作方面的信息.模型级契约提供了服务契约的可视化形式,方便了开发人员之间的交流与合作.该研究将服务的行为信息加入到 Web 服务描述中,实现基于契约的 Web 服务测试,从服务提供商和用户角度分别定义了提供契约和请求契约,描述了提供商提供的服务行为和用户需要的服务行为.在互联网环境下,服务提供商通常不会将服务的修改或升级情况告知用户,但这些对服务的变更往往会改变服务的某些特性,包括功能特性和非功能特性.针对该问题,文献[15]提出将测试用例作为服务提供商和用户之间的契约,令用户能够利用提供商提供的测试用例对服务进行回归测试,验证 Web 服务新、老版本的一致性.

1.3 基于搜索算法的测试生成

测试数据生成问题通常可以通过建模转化为搜索问题,并利用目前已经成熟的搜索算法来求解和生成测试数据.常用的搜索算法包括爬山算法、模拟退火算法、遗传算法和蚁群算法等.

爬山算法[16]是一种局部搜索算法,算法简单、容易理解,但因易于找到局部最优解而无法继续搜索.模拟退火算法[16]是对爬山算法的一种改进,该算法不是一味地追求最优解,而是以一定概率接受较差的解,减少对初始解的依赖,有助于全局最优解的搜索.文献[17]针对面向路径的测试数据自动生成特点,结合模拟退火算法局部搜索能力和遗传算法全局搜索能力,利用模拟退火遗传算法实现测试数据自动生成.

遗传算法[16]源于生物进化论中的自然选择和遗传机制,通过模拟自然进化过程搜索最优解.该算法从一个初始解集开始,运用重组和变异等运算产生新一代解.文献[18,19]均利用遗传算法实现测试数据自动生成,文献[18]提出一种面向目标的测试数据自动生成技术,利用被测程序的控制依赖关系指导搜索过程,获取满足测试要求的测试数据.文献[19]基于被测软件的控制流图定义其适应度函数,应用遗传算法获取最优解.

蚁群算法[20]是一种随机搜索算法,以自然界中蚁群的集体行为为原型,通过由候选解组成的群体进化过程寻求最优解.文献[20]在基本蚁群模型的基础上确定最小挥发系数,并通过逐步减小挥发系数,在不影响收敛速度的前提下提高了算法的全局搜索能力.文献[21]应用蚁群算法实现测试数据自动生成,采用位串形式编码,为基本蚁群算法引入变异算子和自适应挥发系数,提高蚂蚁路径的多样性.

本文将基于 ISC 的测试数据生成转化为在限定区域的搜索问题.针对被测服务输入数据多、数据间存在复杂的相互依赖和限制关系,本文改进了现有的目标函数定义,使其能够表达多个输入数据之间复杂的约束关系,将语义模型引入模拟退火算法,通过搜索过程获得符合约束条件要求的测试数据.

2 服务接口语义契约

在基于服务的软件中,服务接口为用户提供标准的软件功能以及参数、返回值等信息的描述,是服务提供和服务使用之间互操作的基础,构建了开放的互联网环境中服务提供商和用户之间的一种基于契约的合作模式,双方对契约理解的一致性是服务能够正确使用的基础.然而,服务提供时通常有适合的应用范围和场景,例如参数的取值范围、操作的调用顺序等;服务使用也建立在对服务的某些隐含假设之上.现有的服务接口描述中缺少足够的信息,无法准确表达这些隐含的约束限制条件,常常造成双方对服务理解的不一致,从而导致运行

错误.

针对上述问题,本文从以下两个方面扩展服务接口描述:

(1) 基于契约设计的思想,建立接口的契约模型.采用基于谓词逻辑表达的规则语言,描述服务的数据及流程的约束限制条件.契约模型定义了原子服务的功能以及输入/输出参数,规定了服务之间的数据依赖关系和控制依赖关系,为服务调用以及服务之间的协同合作提供了所需的场景信息,并定义了在相应场景下的预期结果;

(2) 引入基于语义的知识表达技术,定义通用的领域概念模型,描述操作的输入/输出参数、与服务运行相关的环境变量以及系统状态参数等基本数据信息.本体模型定义了数据类型、数据之间关系、数据的约束限制条件等领域知识.约束条件分为两类:针对单独数据属性定义的简单约束条件,包括基数约束和值域约束以及针对多数据多属性之间关系的复杂约束条件,采用基于语义的规则描述.

服务接口契约以及服务相关的领域知识为 ISC 模型提供了丰富资源,下面从服务接口契约、数据语义描述和数据约束描述 3 个方面介绍 ISC 模型.

2.1 服务接口契约

用户调用服务时,需要为服务提供执行所必需的运行环境以及数据支持,当且仅当条件具备时,服务才能正常运行.服务接口契约给出了服务执行条件的描述和定义,是用户正确调用服务的基础.本文从以下几个方面定义接口的契约模型:

(1) 服务输入,服务在运行过程中,从外界所获取的数据;

(2) 服务输出,不同的输入会激发不同的运行过程,产生相应的执行结果.当服务运行异常时,输出结果还包括异常信息.输出是判断服务运行正常与否的重要指标之一;

(3) 服务之间的依赖关系,多个服务通过一定的协同满足用户更加复杂的需求.在协同过程中,服务之间存在各种约束依赖关系,如服务执行序列、运行时间约束、输入/输出参数之间的关联关系等.服务依赖关系准确完整的定义是实现服务协同合作的保证.

服务可形式化定义为五元组,包括服务基本信息、输入参数、输出参数、服务数据依赖以及控制依赖等,如下所示:

$$Service=\langle Spec,Inputs,Outputs,Control\text{-}Dependence,Data\text{-}Dependence\rangle.$$

其中,

1) $Spec:=\langle ID,Name,Description\rangle$为服务基本信息的定义,包括服务编号、服务名称及服务功能描述;

2) $Inputs:=\{data_i\}$为输入参数集合;

3) $Outputs:=\{data_i\}$为输出参数集合;

4) $Control\text{-}Dependence:=\{\langle ID,Dependence_i\rangle\}$定义服务之间的控制依赖关系;

5) $Data\text{-}Dependence:=\{\langle ID,Dependence_i\rangle\}$定义服务之间的数据依赖关系.

服务接口契约定义了两种服务依赖关系:控制依赖和数据依赖.在服务组合过程中,有些服务需要同时进行,有些服务只能顺序执行,对于服务执行序列的要求就是服务控制依赖,控制依赖定义了执行顺序的各种约束条件.例如,对于任意两个服务 S_1 和 S_2,控制依赖中的约束包括:

1) 顺序约束,例如 $Before(S_1,S_2)$,$After(S_1,S_2)$或 $Parallel(S_1,S_2)$,分别表示 S_1 在 S_2 之前执行、S_1 在 S_2 之后执行以及 S_1 和 S_2 并行执行等;

2) 时间约束,例如 $LessThan(Delay(S_1,S_2),T)$,表示 S_2 必须在 S_1 执行完之后的时间 T 之内执行.

服务之间的数据依赖包括输入依赖、输出依赖以及输入/输出依赖.对于顺序执行的两个服务 S_1 和 S_2,如果 S_2 的输入参数由 S_1 的输入参数决定,则 S_1 和 S_2 之间存在输入依赖;如果 S_2 的输出参数由 S_1 的输出参数决定,则 S_1 和 S_2 之间存在输出依赖;如果 S_2 的输入参数由 S_1 的输出参数决定,则 S_1 和 S_2 之间存在输入/输出依赖.服务数据依赖的形式化表示如下:

定义 1(数据依赖). $\exists d_1,d_2,S_1,S_2$,且 $d_1\in S_1.Inputs\cup S_1.Outputs,d_2\in S_2.Inputs\cup S_2.Outputs$.如果存在函数 F,使得

$d_2=F(d_1)$,则服务 S_1 和 S_2 存在数据依赖关系.

1) 如果 $d_1\in S_1.Inputs$ 并且 $d_2\in S_2.Inputs$,则服务 S_1 和 S_2 存在输入依赖关系,记为 $IND(S_1,S_2)$;
2) 如果 $d_1\in S_1.Outputs$ 并且 $d_2\in S_2.Outputs$,则服务 S_1 和 S_2 存在输出依赖关系,记为 $OUTD(S_1,S_2)$;
3) 如果 $d_1\in S_1.Outputs$ 并且 $d_2\in S_2.Inputs$,则服务 S_1 和 S_2 存在输入/输出依赖关系,记为 $INOUTD(S_1,S_2)$.

数据依赖可以通过数据之间的函数 F 来定义,例如 $Assertion(d_1,d_2)$可以比较两个整数的大小,或者判定两个字符串是否相等.假设在被测服务中,S_1 为注册服务 reg,其输入参数 d_{11},d_{12},d_{13},分别为用户名、密码以及用户电子邮件地址;S_2 为更新服务 update,其输入参数 d_{21},d_{22},d_{23},分别为用户名、密码和需要更新的航班信息.那么,

1) S_1 和 S_2 之间存在输入依赖关系 $IND(S_1,S_2)$,也就是 $Equals(S_1.d_{11},S_2.d_{21})$以及 $Equals(S_1.d_{12},S_2.d_{22})$;
2) S_1 和 S_2 之间存在控制依赖关系 $Before(S_1,S_2)$,也就是说,执行服务 update 之前必须首先执行服务 reg,只有当用户注册成功成为合法用户才能完成更新任务.

2.2 数据语义描述

软件服务的输入/输出参数、环境变量以及系统状态参数等数据中含有大量的领域知识信息.例如在航班查询服务中,飞机的航班号、出发/抵达城市等参数,都必须是取自特定集合中的元素,而非简单的字符串定义.正确使用这些数据,不仅需要引用格式正确,更需要对数据语义的正确理解.因此,需要建立一个统一的数据概念模型,以便于服务提供方与使用方能形成一致的认识和理解.

本文采用本体技术构建服务的领域概念模型.本体起源于哲学,作为一种知识表达技术,是“对共享概念的明确的形式化描述”[22],已广泛应用于计算机科学和信息科学[23].本体为描述特定领域的关键概念、概念间的语义互联以及规则推理等提供了基础[24].本体模型将领域知识和操作知识区分开,并使得隐含的领域假设明确化,促进了对组织结构良好的数据形成共识以及领域知识在不同实现中的重用[24].本体通常应用类、属性、关系、约束和实例(个体)来对领域概念进行建模,ISC 中的数据语义模型定义如下:

$$Data:=\langle\{DataTypeClasses\},\{Relations\}\rangle.$$

其中,

1) $DataTypeClasses$ 为数据的类型定义;
2) $Relations:=\langle DataTypeClass1,DataTypeClass2,Rel\rangle$为数据类之间的关系定义.

在数据分类方面,可以分为基本数据类型和领域数据类型.基本数据类型包括字节、字符串、整型数据等简单数据类型以及由简单数据构成的复杂数据类型如结构、枚举等.在基本数据类型定义的基础上,领域数据类型可捕获领域内的基本概念及其相互关系.例如,图 2 显示了航班查询系统的数据模型.其中:TimeType,NameType 为 String 类型数据;SeatCount 为 Integer 类型数据;CityName 和 AirlineName 为 NameType 的子类,其继承 NameType 所有特性,并且比 NameType 含有更加丰富的语义信息.City,Airline,CabinType 等为枚举类型数据,参数值只能取自预定义的枚举元素集合.FlightInfo 和 TicketInfo 等为结构类型数据,包含多个数据作为其结构元素,这些元素可以是基本数据类型也可以是领域数据类型,通过数据属性定义结构数据与结构元素之间的关系.

基于本体定义,数据类型间可定义多种关系.例如 OWL 本体描述中,基于集合操作,定义的数据类型关系有:

1) 继承关系:子类继承父类所有的属性并且可以进行必要的扩展;
2) 等价关系:两个等价类拥有完全一样的实例,并且在使用时可以互换;
3) 互斥关系:两个互斥类不含有相同的实例,这说明两个互斥类之间没有重叠;
4) 交集关系:通过对类进行交集运算得到交集类;
5) 并集关系:通过对类进行并集运算得到并集类;
6) 补集关系:如果 O_i 是 O_j 的补集类,那么所有不属于 O_j 的实例均为 O_i 的实例.

数据之间的关系还可以通过数据属性定义,属性从多个方面描述数据的特性,丰富数据的语义信息.常用的数据属性,如:

1) 针对数值型数据,可定义属性如 hasValue,hasMaximum 和 hasMinimum,分别记录数值型数据类与相

应的数据实例或者数值之间的关系,表达数据取值、数据最大/最小值等含义;

2) 针对字符串数据,可定义属性如 hasStringLength,hasMaxLength 和 hasMinLength,定义字符串型数据类与相应数据实例或数值之间的关系,以指定字符串长度或定义字符串最大、最小长度;

3) 复杂数据类型的属性,定义了复杂数据类及其所包含数据元素之间的关系.例如,为 FlightInfo 定义属性 departureCity,该属性值域为数据类 City,用来说明 FlightInfo 的出发城市,departureCity 就是数据类 FlightInfo 和数据类 City 之间的关系定义.

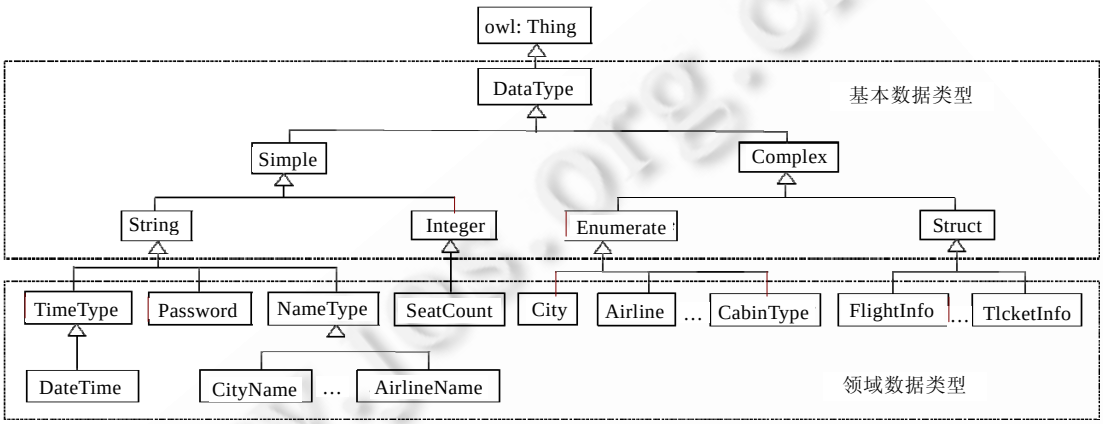

Fig.2 Instance of data model

图 2 数据模型实例

2.3 数据约束描述

数据之间存在着相互依赖和制约关系,在不同的约束条件下,服务将会产生不同的执行结果.例如,在 FlightService 服务中,定义数据类 FlightInfo 表达航班信息,FlightInfo 包含属性出发城市 departureCity 和到达城市 arrivalCity,如果 departureCity 和 arrivalCity 取值相同(即为相同的城市),则服务 FlightService 将会返回报错信息而不执行任何操作.这些关系需明确地表达,以便于检测数据的有效性和服务功能的正确性.本文基于上述服务接口契约以及数据语义描述,建立约束模型及其形式化表达,以准确捕获数据之间的约束关系.约束模型中包括 3 种约束条件:基数约束、值域约束以及规则约束.

$$Constrain:=\langle Cardinality,ValueRange,Rules\rangle.$$

其中,*Cardinality* 和 *ValueRange* 定义了数据自身属性的约束关系:

1) *Cardinality* 基数约束,是指被约束对象的取值数量的限制,包括最大、最小和固定的基数约束;

2) *ValueRange* 值域约束,是指被约束对象的取值范围的限制,例如所有取值、部分取值或是至少有一个取值来自于指定的类的实例.

例如,类 HotelInfo 含有名字属性 hasName,其取值约束 allValuesFrom:HotelName 规定了 HotelInfo 类的所有实例的名字属性 hasName 的取值均取自类 HotelName 的实例;其基数约束 cardinality:1 则表达了每个宾馆只能有一个名称的含义.

约束模型中的 Rules 能够定义多个数据以及多个属性之间比较复杂的约束关系,典型的有:

1) 同一数据不同属性之间的约束关系.例如,宾馆的剩余客房数不能大于其总客房数,即对于 HotelInfo 类的任意一个实例,其属性 availableRoom 的取值必须小于或等于其属性 totalRoom 的取值;

2) 不同数据的属性之间的约束关系.例如,在同一座城市两家宾馆的名字不能相同,即任意两个 HotelInfo 类的实例,如果其 inCity 属性取值相同,则 hasName 属性取值不能相同.

上述约束关系可以采用语义规则来定义和描述,目前,在本体定义中常用到的规则语言有 SWRL(semantic

Web rule language)[25],KIF(knowledge interchange format)[26]等.例如,采用 SWRL,上述规则可表述如下:

1) 宾馆剩余客房数不大于总客房数

$HotelInfo(?x)\wedge availableRoom(?x,?available)\wedge totalRoom(?x,?total)\rightarrow swrlb:lessThanOrEqual(?available,?total)$;

2) 在同一座城市的两家宾馆名字不能相同

$HotelInfo(?x)\wedge HotelInfo(?y)\wedge inCity(?x,?x\_city)\wedge inCity(?y,?y\_city)\wedge hasName(?x,?x\_name)\wedge hasName(?y,?y\_name)\wedge sameAs(?x\_city,?y\_city)\rightarrow differentFrom(?x\_name,?y\_name)$.

2.4 ISC模型实例

以服务 FlightService 的子服务 updateData 为例,为其建立 ISC 模型.该服务的功能是更新航班信息,输入参数包括:用户名、密码、航班信息,用户名和密码均为简单类型数据,用来验证执行更新任务的用户权限;航班信息为结构类型数据,包括了与航班相关的所有数据内容,该服务的执行情况通过枚举类型的输出参数来表示.该服务接口的具体定义如图 3 所示,其中,输入参数用户名、密码和航班信息的数据类型分别为 UserName, Password 和 FlightInfo,输出参数的数据类型为 ReturnCode.

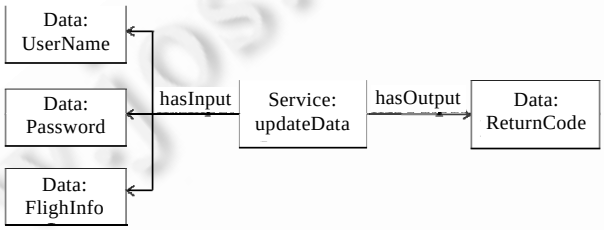

Fig.3 Definition of service updateData

图 3 服务 updateData 定义

服务参数 UserName 为 NameType 类的子类,数据类 NameType 和 Password 均为 String 类型,ReturnCode 数据为枚举类型,FlightInfo 为结构数据类型.服务 updateData 和数据 FlightInfo 的定义片段如图 4 所示,FlightInfo 中包含多个数据元素,其中,FlightNumber 为枚举类型,包含 FN-003 等 6 个枚举元素.

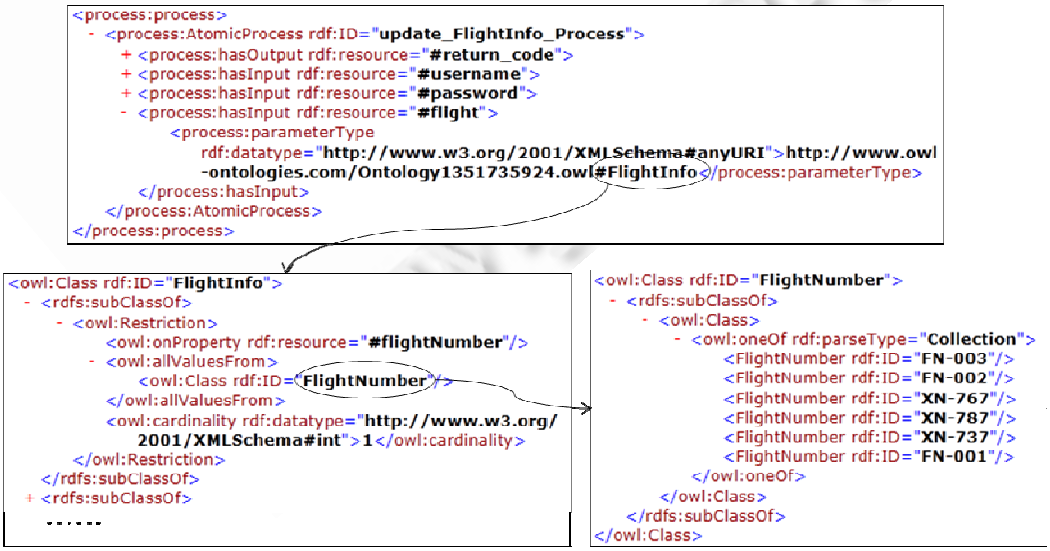

Fig.4 Definition of service and data

图 4 服务及数据的定义

图 4 中除了有服务、数据类层次关系以及属性关系的定义之外,还包含了约束模型中的基数约束和值域约束,多个数据之间的约束限制关系则需要通过规则表达.本研究利用 SWRL 规则语言定义数据间复杂的约束条件,SWRL 规则是基于语义的规则描述,该语言能够充分利用本体模型丰富的语义信息,并增强语义模型的知识推理能力.约束模型中的规则定义部分如图 5 所示.

Name	Expression
Rule-1	FlightInfo(?x_flight) ∧ FlightInfo(?y_flight) ∧ FlightNumber(?number) ∧ Airline(?x_comp) ∧ Airline(?y_comp) ∧ flightNumber(?x_flight, ?number) ∧ company(?x_flight, ?...
Rule-10	FlightInfo(?x) ∧ departureTime(?x, ?dep_time) ∧ arrivalTime(?x, ?arr_time) ∧ after(?dep_time, ?arr_time) → ReturnCode(ARR_BEFORE_DEP_TIME)
Rule-11	FlightInfo(?x) ∧ departureTime(?x, ?dep_time) ∧ hasTicketInfo(?x, ?ticket) ∧ departureDateTime(?ticket, ?ticket_date) ∧ swrlb:substring(?ticket_time, ?ticket_date, 11, 5...
Rule-12	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, BClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInBCabin(?ticket, ?free...
Rule-13	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, EClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInECabin(?ticket, ?free...
Rule-14	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, FClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInFCabin(?ticket, ?free...
Rule-2	FlightInfo(?x) ∧ departureCity(?x, ?departure) ∧ arrivalCity(?x, ?arrival) ∧ differentFrom(?departure, ?arrival) → ReturnCode(NO_ERROR)
Rule-3	FlightInfo(?x) ∧ departureTime(?x, ?dep_time) ∧ arrivalTime(?x, ?arr_time) ∧ after(?arr_time, ?dep_time) → ReturnCode(NO_ERROR)
Rule-4	FlightInfo(?x) ∧ departureTime(?x, ?dep_time) ∧ hasTicketInfo(?x, ?ticket) ∧ departureDateTime(?ticket, ?ticket_date) ∧ swrlb:substring(?ticket_time, ?ticket_date, 11, 5...
Rule-5	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, BClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInBCabin(?ticket, ?free...
Rule-6	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, EClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInECabin(?ticket, ?free...
Rule-7	FlightInfo(?x) ∧ hasCabinInfo(?x, ?cabin) ∧ cabinType(?cabin, FClassCabin) ∧ totalSeats(?cabin, ?total) ∧ hasTicketInfo(?x, ?ticket) ∧ freeSeatsInFCabin(?ticket, ?free...
Rule-8	FlightInfo(?x_flight) ∧ FlightInfo(?y_flight) ∧ FlightNumber(?number) ∧ Airline(?x_comp) ∧ Airline(?y_comp) ∧ flightNumber(?x_flight, ?number) ∧ company(?x_flight, ?...
Rule-9	FlightInfo(?x) ∧ departureCity(?x, ?departure) ∧ arrivalCity(?x, ?arrival) ∧ differentFrom(?departure, ?arrival) → ReturnCode(DEP_AND_ARR_SAME_CITY)

Fig.5 Rule definition in constraint model

图 5 约束模型中的规则定义

3 测试数据生成

本文基于被测服务的 ISC 本体模型为测试数据生成数据分区,在数据分区和被测服务 ISC 约束模型的基础上,采用模拟退火算法生成测试数据.由于被测服务有多个服务参数,并且参数之间存在约束依赖关系,因此需要对传统的模拟退火算法及其目标函数进行必要的修改,通过调整多个数据取值得到最优目标函数值,从而完成搜索过程.

3.1 数据分区生成

分区测试是一种广泛应用的测试数据设计方法,其基本思想是将输入数据和环境变量等服务参数划分为多个子域,这样的子域被称作数据分区[27,28].根据不同的覆盖率要求,应用不同的策略,从这些分区中选择测试数据进行测试.分区测试的有效性很大程度上取决于分区设计算法、分区及其数据是否能够有效发现缺陷,满足覆盖率的要求.传统方法中,分区往往根据测试人员的直觉和经验手动生成,自动生成技术也只能局限于语法信息,例如基于对某特定编程语言中数据类型的分析.本文引入服务的语义信息,通过语义分析,识别数据之间的关系,提高所生成数据分区的有效性.

基于 ISC 的本体模型,数据分区生成过程如下:

对于尚未定义分区的每个数据类 O_i:

1) 为 O_i 建立合法数据分区和非法数据分区;
2) 为 O_i 的合法数据分区建立与其子类相对应的合法数据子分区,并根据该数据类子类之间的关系定义相应的数据子分区之间的关系:
   a. 如果数据子类之间存在等价关系,则相对应的子分区之间也相互等价;
   b. 如果数据子类之间存在互斥关系,则相对应的子分区之间也为互斥关系;
   c. 如果某数据子类由其他类通过交集、并集或者补集运算得到,则与其对应的子分区也要通过相应的分区经过相同运算获得;
3) 根据关键属性的取值,为 O_i 定义合法子分区.例如,针对航班信息类 FlightInfo,可以根据其属性航空公司 company 的取值划分为不同的数据分区,每个分区的航班隶属于一个航空公司;
4) 根据属性约束条件为 O_i 的非法数据分区定义子分区.例如,根据航班信息的约束条件“座位不超过 200 个”为航班信息增加“座位超过 200 个”的非法数据子分区,可以根据单一约束划分分区,也可以综

合考虑多条约束来定义分区.

算法 1 给出了数据分区的生成和选择过程, $Par_{valid}^{O_i}$ 是为本体类 O_i 定义的合法分区, $Par_{invalid}^{O_i}$ 为非法分区, P^{O_i} 为本体类 O_i 的属性, C^{O_i} 是为本体类 O_i 定义的约束条件,其中,对于非法分区仅考虑违反一条数据属性约束的情况.

算法 1. 数据分区生成算法.

While (Exist Data Type O_i whose partitions are not defined) **do**

Generate partition $Par_{valid}^{O_i}$ and $Par_{invalid}^{O_i}$

If (O_i has son classes $\{S_0^{O_i},...,S_n^{O_i}\}$)

For (j=0, j<n, j++) **do**

Generate sub partition $Par_{valid,j}^{O_i}$ of $Par_{valid}^{O_i}$, which is equivalent to class $S_j^{O_i}$

If (exist O_i's son class $S_k^{O_i}$) and ($S_j^{O_i} \equiv S_k^{O_i}$),

Then $Par_{valid,j}^{O_i} \equiv Par_{valid,k}^{O_i}$

If (exist O_i's son class $S_k^{O_i}$) and ($S_j^{O_i}$ disjointWith $S_k^{O_i}$),

Then $Par_{valid,j}^{O_i}$ disjointWith $Par_{valid,k}^{O_i}$

If (exist O_i's son classes $S_m^{O_i}, S_n^{O_i}$) and ($S_j^{O_i} \equiv (S_m^{O_i}$ and $S_n^{O_i})$),

Then $Par_{valid,j}^{O_i} \equiv (Par_{valid,m}^{O_i}$ and $Par_{valid,n}^{O_i})$

If (exist O_i's son classes $S_m^{O_i}, S_n^{O_i}$) and ($S_j^{O_i} \equiv (S_m^{O_i}$ or $S_n^{O_i})$),

Then $Par_{valid,j}^{O_i} \equiv (Par_{valid,m}^{O_i}$ or $Par_{valid,n}^{O_i})$

If (exist O_i's son class $S_k^{O_i}$) and ($S_j^{O_i} \equiv$(not $S_k^{O_i}$)),

Then $Par_{valid,j}^{O_i} \equiv$(not $Par_{valid,k}^{O_i}$)

End For

End If

If (O_i has key properties $\{P_0^{O_i},...,P_m^{O_i}\}$)

For (j=0, j<m, j++) do

Generate sub partition $Par_{valid,j,k}^{O_i}$ for every value of property $P_j^{O_i}$

End For

End If

If (O_i has constraints $\{C_0^{O_i},...,C_n^{O_i}\}$)

For (j=0, j<n, j++) do

Generate sub partition $Par_{invalid,j}^{O_i}$ of $Par_{invalid}^{O_i}$ according to constraint $C_j^{O_i}$

End For

End If

End While

3.2 测试数据生成

考虑数据之间的多约束关系,本文将测试数据生成分为两部分:(1) 根据参数约束条件将被测服务的输入参数划分为若干个互不相交的参数集合;(2) 对不同参数集合分别采用随机算法和模拟退火搜索算法生成相应参数的取值.

(1) 划分输入参数

被测服务的输入参数集合 I,其中包含 n 个参数,这些参数之间存在约束条件 $C_1,C_2,...,C_m$,每个约束条件均

会包含一个或多个输入参数,分别组成 m 个参数集合 $P_1,P_2,\ldots,P_m$,其中,$P_i(1\leqslant i\leqslant m)$为输入参数集合的子集,包含约束条件 C_i 中的所有输入参数.$obj_1,obj_2,\ldots,obj_m$ 分别为上述约束条件的目标函数,通过目标函数能够判定搜索得到的数据是否满足要求,或者评价与预期目标的接近程度.

目标函数 obj_i 的定义见表 1,其中,$Expression(P_i)$为由约束条件 C_i 得到的表达式,包含 P_i 中的所有参数;a 为常数,通过约束条件的转换运算获得;K 为正整数;运算符 abs 为绝对值计算.此定义借鉴了 Tracey[29]提出的方法,并结合数据语义模型中关于属性约束的定义.

Table 1 Definition of objective function

表 1 目标函数定义

约束条件 C_i 的表达式	目标函数 obj_i
$Expression(P_i)=a$	如果 $Expression(P_i)=a$,则 $obj_i=0$;否则,$obj_i=abs(Expression(P_i)-a)+K$
$Expression(P_i)\neq a$	如果 $Expression(P_i)\neq a$,则 $obj_i=0$;否则,$obj_i=K$
$Expression(P_i)>a$	如果 $Expression(P_i)>a$,则 $obj_i=0$;否则,$obj_i=(a-Expression(P_i))+K$
$Expression(P_i)\geqslant a$	如果 $Expression(P_i)\geqslant a$,则 $obj_i=0$;否则,$obj_i=(a-Expression(P_i))+K$
$Expression(P_i)<a$	如果 $Expression(P_i)<a$,则 $obj_i=0$;否则,$obj_i=(Expression(P_i)-a)+K$
$Expression(P_i)\leqslant a$	如果 $Expression(P_i)\leqslant a$,则 $obj_i=0$;否则,$obj_i=(Expression(P_i)-a)+K$

表 1 中的运算根据本体类实例的具体类型情况进行相应的转换.如果 $P_i\cap P_j\neq\varnothing$,也就是约束条件 C_i 和 C_j 包含有相同的输入参数,则称这两个约束条件之间存在交互作用,将有交互作用的约束条件放在同一个约束集合中,得到约束集合 $G_1,G_2,\ldots,G_k$,这些约束集合满足 $G_1\cup G_2\cup\ldots\cup G_k=\{C_1,C_2,\ldots,C_m\}$,并且 $G_i\cap G_j=\varnothing$.参数集合 Q_i 包含 G_i 中与约束条件相关的所有参数,即 $G_i=\{\cup P_j|C_j\in G_i\}$.

参数集合具有属性 $Q_1\cup Q_2\cup\ldots\cup Q_k\subseteq I$ 以及 $Q_i\cap Q_j=\varnothing(1\leqslant i,j\leqslant k,i\neq j)$.

(2) 生成参数取值

由 Q_i 的定义可知,$I-Q_1\cup Q_2\cup\ldots\cup Q_k$ 中的参数没有受到任何约束条件的限制,因此只需随机生成这些参数取值即可.Q_i 中包含的参数受到一条或多条约束条件的限制,要找到符合这些约束条件的取值,本文运用模拟退火算法为集合 $Q_1,Q_2,\ldots,Q_k$ 中的参数搜索取值,从而生成满足所有约束条件的测试数据.

1) 对于 $I-Q_1\cup Q_2\cup\ldots\cup Q_k$ 中的每一个参数,随机生成所有参数取值;

2) 对于 $Q_1,Q_2,\ldots,Q_k$ 中的每一个参数集合 Q_i,循环执行以下操作:

   a. 与 Q_i 对应的约束集合 G_i,以随机生成法分别为 Q_i 中的所有参数赋值,为初始解 S,并计算初始解 S 的子目标函数,即 $obj_{G_i}=\sum_{C_j\in G_i}obj_j$;

   b. 如果目标函数不为 0,则搜索解 S 的相邻解空间,如果其相邻空间中存在新解 S'优于 S,也就是能够使目标函数值减小,则接受 S'作为新的当前解;否则,以一定的概率接受 S',此概率由模拟退火算法中的温度系数和两个目标函数之差决定;

   c. 如果目标函数为 0,则顺序选取下一个参数集合,并重复执行步骤 a~步骤 c.

算法 2 为测试数据生成的具体过程,根据参数影响的约束条件的个数将其分为两类:1) 不受任何约束条件限制的参数;2) 受到一条或多条约束条件限制的参数.对这两类参数分别应用随机方法和模拟退火算法生成参数取值.算法中,参数 t 为模拟退火算法中的温度系数,该系数按照预先制定的计划逐步减小,$num\_EachLevel$ 为每一档温度 t 所需进行的搜索次数.S 为参数集合 Q_i 的一种赋值情况,函数 $Neighbor(S)$为 S 的相邻解,与 S 仅有一个参数的取值不同.$Neighbor(S)$根据不同的参数类型会进行不同的操作:对于枚举类型的数据,在枚举元素集合中随机选取除当前枚举元素外的其他任意元素即可;对于数值类型的元素,选取比当前数值大 m(或者小 m)的数值,我们称 m 为步长,在搜索初期选择较大的步长,能够较快地收敛,随着搜索过程的进行,目标函数值逐渐减小,步长也要相应地减小.obj_{G_i} 为与集合 G_i 对应的目标函数,目标函数用来衡量所得数据是否满足约束条件,并指导下一步的搜索方向.该算法将生成相互之间存在约束关系的一组测试数据.

算法 2. 测试数据生成算法.

Let I be the parameter set contains all the service input parameters
Let $C_1,C_2,\ldots,C_m$ be the constraints for all the input parameters
Construct object functions $obj_1,obj_2,\ldots,obj_m$ for $C_1,C_2,\ldots,C_m$ respectively
Let P_i be a set contains parameters which appear in the constraint C_i
Construct constraints' sets $G_1,G_2,\ldots,G_k$, where $G_i=\{\cup C_j|\cap P_j\neq\varnothing,1\leqslant j\leqslant m\}$ and $obj_{G_i}=\sum_{C_j\in G_i} obj_j$

For (each set G_i in $\{G_1,G_2,\ldots,G_k\}$) **do**
    $Q_i=\{\cup P_j|C_j\in G_i\}$
End For
For (each parameter I_j in $I-Q_1\cup Q_2\cup\ldots\cup Q_k$) **do**
    Select a value λ_j for I_j at random
End For
For (each $Q_i(1\leqslant i\leqslant k)$) **do**
    Generate an initial solution S for Q_i
    While ($obj_{G_i}\neq 0$) **do**
        Select an initial temperature $t>0$
        While ($t>T$) **do**
            $it\leftarrow 0$
            While ($it<num\_EachLevel$) **do**
                Select $S'\in Neighbor(S)$ at random
                $\Delta e\leftarrow obj_{G_i}(S')-obj_{G_i}(S)$
                If ($\Delta e<0$) **Then** $S\leftarrow S'$
                Else
                    Generate random number r, $0\leqslant r<1$
                    If $\left(r<e^{-\frac{\Delta e}{t}}\right)$ **Then** $S\leftarrow S'$
                    End If
                End If
                $it\leftarrow it+1$
            End While
            Decrease t according to cooling schedule
        End While
    End While
End For

4 实 验

4.1 被测服务

本文以组合服务 TravelService 为被测服务,其包含 UserService,FlightService,HotelService 和 RestaService,分别完成用户注册、航班、宾馆以及餐馆相关数据的查询以及更新等服务,其中,FlightService,HotelService 和 RestaService 为主要的被测服务.涉及到的主要数据有 UserInfo,FlightInfo,HotelInfo 以及 RestaInfo,这 4 个数据均为结构型数据.

UserInfo 包含 3 方面信息:用户名、密码以及电子邮箱地址,其结构如图 6 所示.基数约束包括:一个用户只

能有一个用户名和一个密码,见表 2.

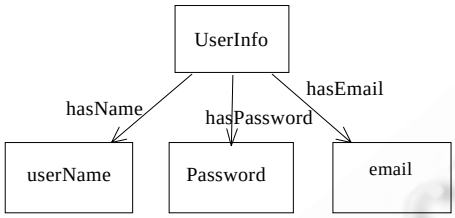

Fig.6 Ontology class of UserInfo

图 6 UserInfo 本体类

Table 2 Cardinality constraints for UserInfo

表 2 UserInfo 的基数约束

约束表达	约束描述
userName exactly 1	唯一的用户名
password exactly 1	唯一的密码

在用户注册时,要求不同用户不能选用相同的用户名,该约束利用 SWRL 规则表达如下:

$$UserInfo(?x) \wedge UserInfo(?y) \wedge hasName(?x,?x\_name) \wedge hasName(?y,?y\_name) \wedge sameAs(?x\_name,?y\_name) \wedge sameAs(?x,?y) \rightarrow ReturnCode(NO\_ERROR).$$

图 7 展示了数据类型 FlightInfo 的定义.

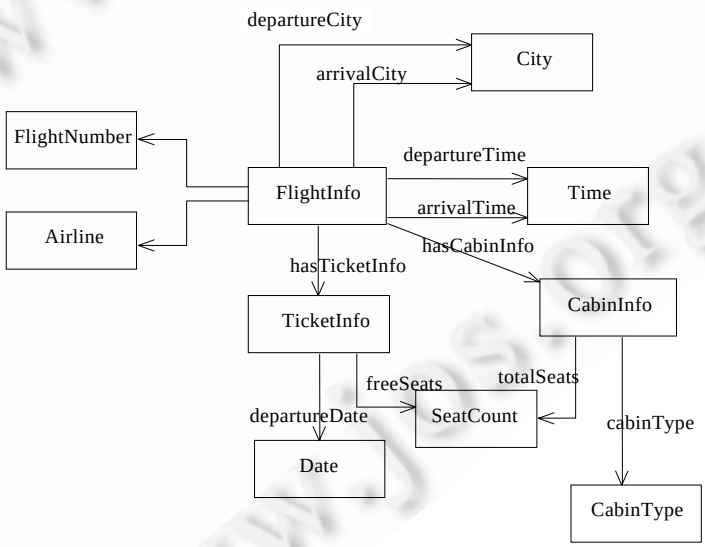

Fig.7 Ontology class of FlightInfo

图 7 FlightInfo 本体类

FlightInfo 所包含的元素可以分为 3 大类:(1) 航班信息;(2) 航空公司;(3) 机票信息.具体元素有航班号、航空公司、出发/抵达城市、起飞/抵达时间、座位信息以及机票信息,这些元素与结构数据类 FlightInfo 之间的关系均通过相应属性描述.例如,属性 departureCity 定义了 FlightInfo 和 City 之间的关系,即 City 为 FlightInfo 的出发城市.作为一条有效的航班信息,其航班号、航空公司、出发城市、抵达城市、起飞时间以及抵达时间的取值都应该是唯一的,如果这些元素存在两个或两个以上的取值,会造成数据之间的相互矛盾.也就是说,既从北京出发又从上海出发的航班信息是不合理的.由于每个航班至少会有一种等级的座位,因此数据类 FlightInfo 的

每个实例至少有一条座位信息.上述约束是对数据 FlightInfo 相关属性的基数约束,见表 3.除此之外,航班信息包含的元素之间也存在复杂的约束限制条件,例如同一个航班号不能隶属于两个航空公司、航班的抵达城市不能是出发城市以及航班的抵达时间一定要晚于出发时间等等.这些约束条件均涉及到两个或者更多的数据实例,需要利用基于语义的规则来描述.FlightInfo 包含的数据之间的取值约束见表 4,表中约束条件均为对合法数据的约束,合法数据对应的输出参数 ReturnCode 取值为 NO_ERROR.

Table 3 Cardinality constraints for FlightInfo

表 3 FlightInfo 的基数约束

约束表达	约束描述
flightNumber exactly 1	唯一的航班号
company exactly 1	唯一的航空公司
departureCity exactly 1	出发城市唯一
arrivalCity exactly 1	抵达城市唯一
departureTime exactly 1	起飞时间唯一
arrivalTime exactly 1	抵达时间唯一
hasCabinInfo min 1	至少一条座位信息

Table 4 Value constraints for FlightInfo

表 4 FlightInfo 的取值约束

序号	规则表达	约束描述
1	FlightInfo(?x_flight)∧FlightInfo(?y_flight)∧FlightNumber(?number)∧ Airline(?x_comp)∧Airline(?y_comp)∧flightNumber(?x_flight,?number)∧ company(?x_flight,?x_comp)∧flightNumber(?y_flight,?number)∧ company(?y_flight,?y_comp)∧sameAs(?x_comp,?y_comp)∧→ReturnCode(NO_ERROR)	每个航班号 只能隶属于 一个航空公司
2	FlightInfo(?x)∧departureCity(?x,?departure)∧arrivalCity(?x,?arrival)∧ differentFrom(?departure,?arrival)→ReturnCode(NO_ERROR)	抵达城市不能与出发 城市是同一座城市
3	FlightInfo(?x)∧departureTime(?x,?dep_time)∧arrivalTime(?x,?arr_time)∧ after(?arr_time,?dep_time)→ReturnCode(NO_ERROR)	抵达时间必须 晚于出发时间
4	FlightInfo(?x)∧departureTime(?x,?dep_time)∧hasTicketInfo(?x,?ticket)∧ departureDate(?ticket,?ticket_date)∧swrlb:substring(?ticket_time,?ticket_date,11,5)∧ sameAs(?dep_time,?ticket_time)→ReturnCode(NO_ERROR)	机票信息中的出发 时间要与航班信息的 出发时间一致
5	FlightInfo(?x)∧hasBClassCabin(?x,?bCabin)∧totalSeats(?bCabin,?total)∧ hasTicketInfo(?x,?ticket)∧freeSeatsInBCabin(?ticket,?free)∧ swrlb:greaterThanOrEqual(?total,?free)→ReturnCode(NO_ERROR)	对于每个等级的座位,剩余 座位数不能大于总座位数 (以商务舱为例,其他类似)

数据类 HotelInfo 包含宾馆 ID、宾馆名称、所在城市、总客房数、剩余客房数等,如图 8 所示,在一条有效的宾馆信息中,这些元素均只能有唯一取值,关于该数据类的基数约束见表 5.HotelInfo 还包含有表 6 所示的取值约束.

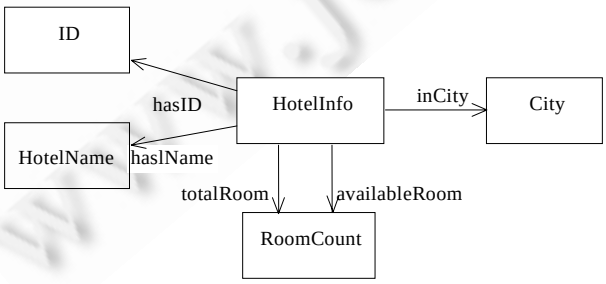

Fig.8 Ontology class of HotelInfo

图 8 HotelInfo 本体类

Table 5 Cardinality constraints for HotelInfo

表 5 HotelInfo 的基数约束

约束表达	约束描述
hasID exactly 1	宾馆有唯一的 ID
hasName exactly 1	宾馆有唯一的名称
inCity exactly 1	所在城市唯一
totalRoom exactly 1	唯一的客房总数
availableRoom exactly 1	唯一的剩余客房数

Table 6 Value constraints for HotelInfo

表 6 HotelInfo 的取值约束

序号	规则表达	约束描述
1	HotelInfo(?x)∧HotelInfo(?y)∧hasID(?x,?x_id)∧hasID(?y,?y_id)∧ sameAs(?x_id,?y_id)∧sameAs(?x,?y)→ReturnCode(NO_ERROR)	任意两家不同的宾馆 不能有相同的 ID 号
2	HotelInfo(?x)∧HotelInfo(?y)∧inCity(?x,?x_city)∧inCity(?y,?y_city)∧ hasName(?x,?x_name)∧hasName(?y,?y_name)∧sameAs(?x_city,?y_city)∧ differentFrom(?x_name,?y_name)→ReturnCode(NO_ERROR)	在同一座城市不存在 名字相同的宾馆
3	HotelInfo(?x)∧totalRoom(?x,?total)∧availableRoom(?x,?available)∧ Swrlb:lessThanOrEqual(?available,?total)→ReturnCode(NO_ERROR)	剩余客房数 不大于总客房数

有关餐馆的相关信息保存在数据类 RestaInfo 中,该数据类包含餐馆 ID、餐馆名称、所在城市、餐位总数以及剩余餐位数等,其结构如图 9 所示,与 RestaInfo 相关的基数约束定义在表 7 中.

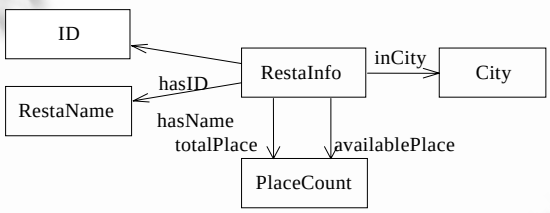

Fig.9 Ontology class of RestaInfo

图 9 RestaInfo 本体类

Table 7 Cardinality constraints for RestaInfo

表 7 RestaInfo 的基数约束

约束表达	约束描述
hasID exactly 1	餐馆有唯一的 ID
hasName exactly 1	餐馆有唯一的名称
inCity exactly 1	所在城市唯一
totalPlace exactly 1	唯一的餐位总数
availablePlace exactly 1	唯一的剩余餐位数

数据类 RestaInfo 对其属性取值也有若干约束条件,这些约束条件均利用 SWRL 规则描述,见表 8.

Table 8 Value constraints for RestaInfo

表 8 RestaInfo 的取值约束

序号	规则表达	约束描述
1	RestaInfo(?x)∧RestaInfo(?y)∧hasID(?x,?x_id)∧hasID(?y,?y_id)∧ sameAs(?x_id,?y_id)∧sameAs(?x,?y)→ReturnCode(NO_ERROR)	任意两家不同的餐馆 不能有相同的 ID 号
2	RestaInfo(?x)∧RestaInfo(?y)∧inCity(?x,?x_city)∧inCity(?y,?y_city)∧ hasName(?x,?x_name)∧hasName(?y,?y_name)∧sameAs(?x_city,?y_city)∧ differentFrom(?x_name,?y_name)→ReturnCode(NO_ERROR)	在同一座城市 不存在名字 相同的餐馆
3	HotelInfo(?x)∧totalPlace(?x,?total)∧availablePlace(?x,?available)∧ Swrlb:lessThanOrEqual(?available,?total)→ReturnCode(NO_ERROR)	餐馆的剩余餐位数 不大于总餐位数

被测服务接口包括用户注册、航班信息、宾馆信息以及餐馆信息的更新与查询等,这些被测服务接口的输入/输出信息见表 9.

Table 9 Information of the SUT

表 9 被测服务相关信息

服务分类	服务名称	服务描述	参数	参数类型	参数描述
用户管理	regUser	用户注册服务	username	String (in)	用户名称
			password	String (in)	用户密码
			email	String (in)	电子邮件地址
			star	int (out)	用户等级
航班信息管理	updateData	航班信息更新	username	String (in)	用户名称
			password	String (in)	用户密码
			flight	FlightInfo (in)	航班信息
			returnCode	int (out)	服务执行状态
	search	航班信息查询	username	String (in)	用户名称
			fromCity	String (in)	出发城市
			toCity	String (in)	到达城市
			startDate	String (in)	出发时间
			cabin	int (in)	座位等级
			numbers	int (in)	乘机人数
			flight	FlightInfo (out)	航班信息
宾馆信息管理	updateData	宾馆信息更新	username	String (in)	用户名称
			password	String (in)	用户密码
			hotelroom	HotelInfo (in)	宾馆信息
			returnCode	int (out)	服务执行状态
	search	宾馆信息查询	username	String (in)	用户名称
			hotelName	String (in)	宾馆名称
			inCity	String (in)	所在城市
			startDate	String (in)	入住时间
			numbers	int (in)	房间数
			hotelroom	HotelInfo (out)	宾馆信息
餐馆信息管理	updateData	餐馆信息更新	username	String (in)	用户名称
			password	String (in)	用户密码
			restatable	RestaInfo (in)	餐馆信息
			returnCode	int (out)	服务执行状态
	search	餐馆信息查询	username	String (in)	用户名称
			restaName	String (in)	餐馆名称
			inCity	String (in)	所在城市
			startDate	String (in)	用餐时间
			numbers	int (in)	餐位数
			restatable	RestaInfo (out)	餐馆信息

4.2 测试执行

测试数据的生成以被测服务的 ISC 模型为基础,根据 ISC 提供的数据之间的关系以及数据约束的语义信息为数据建立数据分区.首先,对于功能测试而言,至少包含合法数据和非法数据两个数据分区.以结构类型数据航班信息 FlightInfo 为例,根据其是否有效将其划分为合法航班信息和非法航班信息两个分区.由于结构数据类型通过多个属性定义其包含的结构元素,其中任意一个属性取值不合法就会导致航班信息不合法,因此,可以根据航班信息包含的不同属性将非法航班信息分区进一步划分为非法航空公司信息、非法航班号信息、非法城市信息、非法座位信息、非法时间信息这 5 个子分区.针对本实验设计,非法航空公司、非法航班号、非法城市均表示相应属性取值为空,非法座位信息则说明座位数量超出了预定的数值范围,非法时间信息违反了时间格式.例如“2012-13-20-25:70”,其中,月份、小时数和分钟数等多处均违反了时间格式.这些分区中的数据均只有一个属性不符合其相应的取值要求,其余属性均为合法取值.由于时间属性又分为出发时间、到达时间以及机票时间,因此将非法时间信息进一步划分为非法出发时间信息、非法到达时间信息和非法机票时间信息这 3 个

子分区.以上分区均是根据单一属性取值情况进行定义的,除此之外,建立数据分区还应该考虑各属性取值之间的约束关系,根据这些约束关系建立新的数据分区.对于航班信息 FlightInfo 而言,航班到达时间早于航班出发时间、机票信息中的出发时间与航班出发时间不符、客舱剩余座位数大于总座位数以及航班到达城市与出发城市相同都是不合法的数据取值情况,应分别针对这些情况建立相应的数据分区.为被测服务 TravelService 其他相关数据建立数据分区的过程与 FlightInfo 相同,数据分区的具体内容见表 10,其分区含义为针对本实验设计定义.

Table 10 Data partition for TravelService

表 10 TravelService 的数据分区

服务	分区序号	数据分区	分区含义
UserService	1	ValidUserInfo	合法的用户信息
	2	SameUserName	用户名相同的用户信息
FlightService	3	ValidFlightInfo	合法的航空信息
	4	InvalidCompany	航空公司为空字符串
	5	InvalidFlightNumber	航班号为空字符串
	6	InvalidCity	城市为空字符串
	7	InvalidSeat	座位取值超出预定范围
	8	InvalidDepTime	出发时间格式错误
	9	InvalidArrTime	到达时间格式错误
	10	InvalidTicketTime	机票时间格式错误
	11	ArrBeforeDepTime	航班到达时间晚于出发时间
	12	TicTimeDiffDepTime	机票时间与航班出发时间不符
	13	AvaMoreThanTotSeat	剩余座位数大于总座位数
	14	ArrAndDepSameCity	到达城市与出发城市相同
HotelService	15	ValidHotelInfo	合法的宾馆信息
	16	InvalidHotelID	宾馆 ID 为空字符串
	17	InvalidName	宾馆名称为空字符串
	18	InvalidCity	所在城市为空字符串
	19	InvalidRoomCount	房间数量超出预定范围
	20	SameHotelID	不同宾馆的 ID 相同
	21	SameHotelNameInSameCity	在同一座城市存在名称相同的宾馆
	22	AvaMoreThanTotRoom	剩余房间数多于总房间数
RestaService	23	ValidRestaInfo	合法的餐馆信息
	24	InvalidRestaID	餐馆 ID 为空字符串
	25	InvalidName	餐馆名称为空字符串
	26	InvalidCity	所在城市为空字符串
	27	InvalidPlaceCount	餐位数量超出预定范围
	28	SameRestaID	不同餐馆的 ID 相同
	29	SameRestaNameInSameCity	在同一座城市存在名称相同的餐馆
	30	AvaMoreThanTolPlace	剩余餐位数多于总餐位数

实验共生成 30 个分区,其中用户信息数据分区 2 个,航班服务 FlightService 数据分区 12 个,宾馆服务 HotelService 数据分区 8 个,餐馆服务 RestaService 数据分区 8 个.由于航班服务、宾馆服务和餐馆服务为主要被测服务,用户注册服务仅为辅助功能,因此仅为用户信息生成一个合法数据和一个非法数据,为其余 3 个服务的每个分区分别生成 100 组测试数据,共 30 个数据集合作为基于 ISC 测试的输入数据.另外,分别为这 3 个被测服务随机生成 1 200 组测试数据,作为随机测试的输入数据.在生成的测试数据中分别为每个服务选取 50,100,200,300,400,500,600,700,800,900,1 000 个数据组成 11 个不同大小的测试集合,分别用每个测试集合对被测服务进行测试.由于测试数据分别取自不同的数据集合,在选取数据时采取随机策略,也就是首先随机选取数据集合,然后在选定的数据集合中再随机选取测试数据.数据量越大,越能够均匀覆盖不同的数据集合.

4.3 结果分析

4.3.1 覆盖率分析

本实验使用开源的 EMMA 工具监视和获取服务的代码覆盖情况,图 10 中包括了服务 FlightService, HotelService 以及 RestaService 的代码执行情况,综合考虑这 3 个服务,基于 ISC 测试和随机测试达到的代码覆盖率对比情况如图 11 所示.

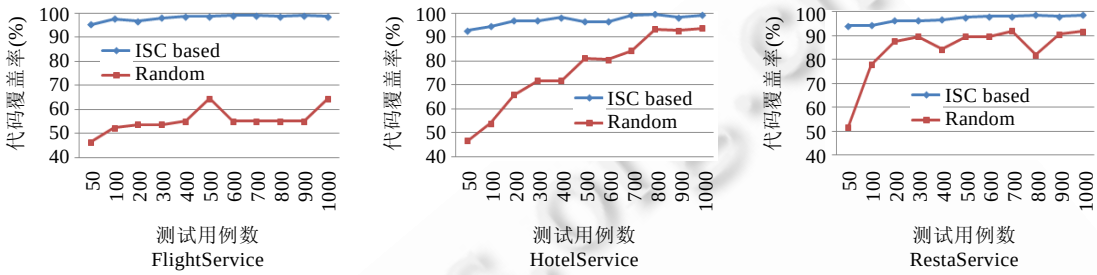

Fig.10 Code coverage of each service

图 10 代码覆盖率对比情况

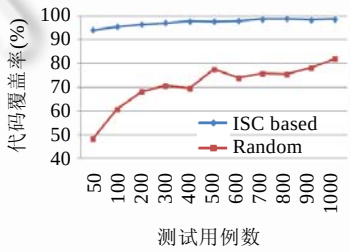

Fig.11 Code coverage

图 11 代码覆盖率

从图 11 可以看出:对于服务 HotelService,基于 ISC 测试能够以随机测试 12%的测试用例达到相同的代码覆盖率;对于服务 RestaService,基于 ISC 测试以随机测试 14%的测试用例达到相同的代码覆盖率;对于服务 FlightService,相同数量的测试用例,基于 ISC 测试和随机测试的代码覆盖率最多可以提高 50%,最少提高 20%.

图 12 包含了 FlightService,HotelService 和 RestaService 这 3 个服务的分支覆盖率情况,图 13 为综合 3 个服务的分支覆盖率对比情况.

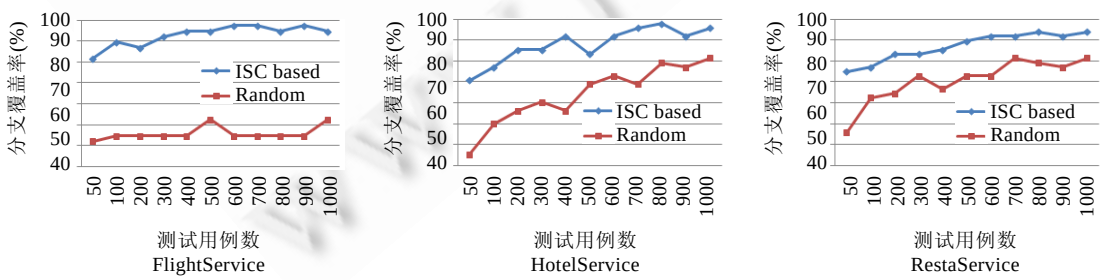

Fig.12 Branch coverage of each service

图 12 分支覆盖率对比情况

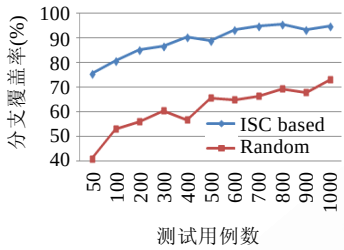

Fig.13 Branch coverage

图 13 分支覆盖率

对于服务 HotelService 和 RestaService,基于 ISC 测试分别用随机测试 17%和 33%的测试用例达到与其相同的分支覆盖率;对于服务 FlightService,基于 ISC 测试的分支覆盖率平均提高 40%.以上实验结果说明,对于输入数据多、数据间约束关系复杂的服务,基于 ISC 测试较随机测试具有明显的优越性.

4.3.2 植入错误

在被测服务中植入 10 个错误,错误类型主要分为以下 7 种:

1) 判定字符串是否为空错误;
2) 判定数值型数据取值范围错误;
3) 判定两个数值型数据之间大小关系错误;
4) 判定两字符串是否相等错误;
5) 判定时间先后顺序错误;
6) 赋值错误,例如在被测软件中有方法 setSTD 和 setSTA,由于名字类似,很容易在赋值时发生错误;
7) 且、或关系错误.

测试过程与上节类似,分别为服务 FlightService,HotelService 以及 RestaService 从基于 ISC 生成的测试数据和随机生成的测试数据中选取 50,100,200,300,400,500,600,700,800,900,1 000 个数据组成 11 个大小不同的测试集合,用这些测试集合对被测服务进行测试,并统计每个测试集合能够发现的植入错误的数量.

两种测试方法的对比结果如图 14 所示,以相同的测试用例数量,对于植入错误的发现率,与随机测试相比,基于 ISC 测试最高可以提高 40%.

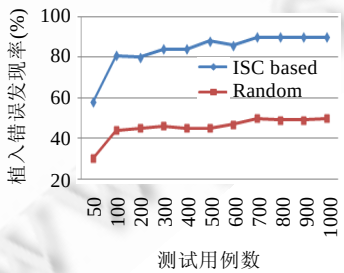

Fig.14 Discovery rate of implant fault

图 14 植入错误发现率

4.4 讨论

本文所提出的方法对于被测服务 FlightService,HotelService 及 RestaService 能够达到理想的覆盖率标准,但也具有一定的局限性:

(1) 本方法的一个前提条件是能够获取相应的领域知识,以建立完整的 ISC 模型,领域知识的完整性和正

确性是保证 ISC 模型质量的基础.然而,领域知识的获取和建模是较为困难的过程,虽然语义 Web 服务技术是近年来的研究热点和重要的发展趋势,并取得了较大的进展,但离实际应用尚存在距离.本研究中,借鉴语义 Web 的思想,与实际的服务测试需求相结合,力求进一步提升测试自动化的智能化程度;

(2) 相对于简单服务,本方法对于具有大量领域知识、复杂约束依赖关系的数据和服务的测试,具有更加明显的改善效果.从 FlightService,HotelService 和 RestaService 这 3 个服务的代码覆盖率对比统计图可以看出,对 HotelService 和 RestaService 两个服务,当测试数据达到一定数量时,随机生成的测试数据也能够达到对被测服务 90%以上的覆盖率,尤其是 RestaService,200 个测试数据就已经达到 87.6%的覆盖率.但同时我们也看到,在 800 个测试数据时,服务 RestaService 的随机测试却只有 82%的代码覆盖率.因此可以得到如下结论:对于输入数据少、数据约束简单的被测服务,随机生成数据也能达到满意的代码覆盖率,但不够稳定;对于服务 FlightService 而言,基于 ISC 测试方法则显示出较大的优势,因为 FlightService 的输入数据多、约束复杂,随机生成的数据很难同时满足这些数据约束,而基于 ISC 测试生成方法则能够根据服务的语义模型有目的的生成数据,因而能够达到较高的覆盖率并且保持相对稳定的覆盖率水平.由此可以看出,被测服务的复杂度越高,该方法的性能改进越明显;

(3) 目前,数据生成算法仅考虑到单一错误的情况,即仅违反一条数据约束条件的场景.因此,对于运行结果只受单一错误影响的服务而言,本方法能够达到理想的覆盖率标准.但多错误(即服务失效是由多个软件故障引发的)、错误之间存在相互影响等复杂的故障模式在软件中普遍存在.针对复杂故障模型的测试,本文中的算法还需要进一步地改进;

(4) 本方法根据 ISC 模型生成大量测试数据,使测试达到理想的覆盖率.但生成的测试数据集还有待进一步优化选择,消除数据冗余,降低测试数据集规模,从而进一步减少测试代价.

5 结 论

服务测试是提高服务质量、建立用户信任的重要手段,本文提出了基于 ISC 的建模与测试方法.ISC 模型借鉴语义 Web 服务和契约设计的思想,增强了 Web 服务的描述能力,以保证对服务的数据、功能、应用等领域知识理解的正确性.以 ISC 模型为基础,提出了基于本体的测试分区生成算法以及数据生成的模拟退火算法.实验表明,针对具有复杂约束依赖关系的服务,可从测试覆盖率和故障检测能力两个方面大幅度提升测试效率.

致谢 在此,我们向对本文的工作提出宝贵意见的陈光、李沐洋、黄骁飞等同学表示感谢.

References:

[1] Bai XY, Dong WL, Tsai WT, Chen YN. WSDL-Based automatic test case generation for Web services testing. In: Proc. of the Service-Oriented System Engineering Int'l Workshop. Washington: IEEE Computer Society, 2005. 207−212. [doi: 10.1109/SOSE.2005.43]

[2] Ma CY, Du CL, Zhang T, Hu F, Cai XB. WSDL-Based automated test data generation for Web service. In: Proc. of the Computer Science and Software Engineering. Washington: IEEE Computer Society, 2008. 731−737. [doi: 10.1109/CSSE.2008.790]

[3] Offutt J, Xu WZ. Generating test cases for Web services using data perturbation. SIGSOFT Software Engineering Notes, 2004, 29(5):1−10. [doi: 10.1145/1022494.1022529]

[4] Jiang Y, Li YN, Hou SS, Zhang L. Test-Data generation for Web services based on contract mutation. In: Proc. of the IEEE Int'l Conf. on Secure Software Integration and Reliability Improvement. Washington: IEEE Computer Society, 2009. 281−286. [doi: 10.1109/SSIRI.2009.49]

[5] Bai XY, Lee SF, Tsai WT, Chen YN. Ontology-Based test modeling and partition testing of Web services. In: Proc. of the Int'l Conf. on Web Services. Washington: IEEE Computer Society, 2008. 465−472. [doi: 10.1109/ICWS.2008.111]

[6] Bai XY, Hou KJ, Lu H, Zhang Y, Hu LP, Ye H. Semantic-Based test oracles. In: Proc. of the Computer Software and Applications Conf. Washington: IEEE Computer Society, 2011. 640−649. [doi: 10.1109/COMPSAC.2011.89]

[7] Wang YB, Bai XY, Li JZ, Huang RB. Ontology-Based test case generation for testing Web services. In: Proc. of the 8th Int'l Symp. on Autonomous Decentralized Systems (ISADS 2007). Washington: IEEE Computer Society, 2007. 43−50. [doi: 10.1109/ISADS.2007.54]

[8] Bai XY, Lu H, Zhang Y, Zhang RW, Hu LP, Ye H. Interface-Based automated testing for open software architecture. In: Proc. of the Computer and Applications Conf. Workshops. Washington: IEEE Computer Society, 2011. 149−154. [doi: 10.1109/COMPSACW.2011.34]

[9] Lee SF, Bai XY, Chen YN. Automatic mutation testing and simulation on OWL-S specified Web services. In: Proc. of the Simulation Symp. Washington: IEEE Computer Society, 2008. 149−156. [doi: 10.1109/ANSS-41.2008.13]

[10] Meyer B. Applying "design by contract". IEEE Computer, 1992,25(10):40−51. [doi: 10.1109/2.161279]

[11] Briand LC, Labiche Y, Sun H. Investigating the use of analysis contracts to support fault isolation in object oriented code. SIGSOFT Software Engineering Notes, 2002,27(4):70−80. [doi: 10.1145/566172.566183]

[12] Richardson DJ, O'Malley O, Tittle C. Approaches to specification-based testing. In: Proc. of the ACM 3rd Symp. on Software Testing, Analysis, and Verification (SIGSOFT'89). New York: ACM Press, 1989. 86−96. [doi: 10.1145/75308.75319]

[13] Bertolino A, Frantzen L, Polini A, Tretmans J. Audition of Web services for testing conformance to open specified protocols. In: Ralf HR, Judith AS, Clemens AS, eds. Proc. of the Architecting Systems with Trustworthy Components. LNCS 3938, Berlin/Heidelberg: Springer-Verlag, 2006. 1−25. [doi: 10.1007/11786160_1]

[14] Heckel R, Lohmann M. Towards contract-based testing of Web services. Electronic Notes in Theoretical Computer Science, 2005, 116(19):145−156. [doi: 10.1016/j.entcs.2004.02.073]

[15] Bruno M, Canfora G, Penta MD, Esposito G, Mazza V. Using test cases as contract to ensure service compliance across releases. In: Benatallah B, Casati F, Traverso P, eds. Proc. of the Service-Oriented Computing (ICSOS 2005). Berlin/Heidelberg: Springer-Verlag, 2005. 87−100. [doi: 10.1007/11596141_8]

[16] McMinn P. Search-Based software test data generation: A survey. Software Testing, Verification and Reliability, 2004,14:105−156. [doi: 10.1002/stvr.294]

[17] Fu B. Automated software test data generation based on simulated annealing genetic algorithms. Computer Engineering and Applications, 2005,41(12):82−84 (in Chinese with English abstract).

[18] Pargas RP, Harrold MJ, Peck RR. Test-Data generation using genetic algorithms. Journal of Software Testing, Verification and Reliability, 1999,9(4):263−282. [doi: 10.1002/(SICI)1099-1689(199912)9:4<263::AID-STVR190>3.0.CO;2-Y]

[19] Jones BF, Sthamer HH, Eyres DE. Automatic structural testing using genetic algorithms. Software Engineering Journal, 1996,11(5): 299−306.

[20] Wang Y, Xie JY. An adaptive ant colony optimization algorithm and simulation. Journal of System Simulation, 2002,14(1):31−33 (in Chinese with English abstract). [doi: CNKI:SUN:XTFZ.0.2002-01-009]

[21] Fu B. Automated software test data generation based on ant colony algorithm. Computer Engineering and Applications, 2007, 43(12):97−99 (in Chinese with English abstract).

[22] Gruber TR. Toward principles for the design of ontologies used for knowledge sharing. Int'l Journal Human-Computer Studies, 1995,43(5-6):907−928.

[23] Singh MP, Huhns MN. Service-Oriented Computing: Semantics, Processes, Agents. Chichester: John Wiley & Sons. Ltd., 2005.

[24] Gruber TR. A translation approach to portable ontology specifications. Knowledge Acquisition, 1993,5(2):199−220.

[25] Horrocks I, Patel-Schneider PF, Boley H, Tabet S, Grosof B, Dean M. SWRL: A semantic Web rule language combining Owl and RuleML. 2004. http://www.w3.org/Submission/SWRL/

[26] Michael RG, Richard EF. Knowledge interchange format. 1992. http://www-ksl.stanford.edu/knowledge-sharing/kif/#manual

[27] Ostrand TJ, Balcer MJ. The category-partition method for specifying and generating functional tests. Communications of the ACM, 1988,31(6):676−686. [doi: 10.1145/62959.62964]

[28] Chen TY, Poon P, Tse TH. A choice relation framework for supporting category-partition test case generation. IEEE Trans. on Software Engineering, 2003,29(7):577−593. [doi: 10.1109/TSE.2003.1214323]

[29] Tracey N, Clark J, Mander K, Mcdermid J, York H. An automated framework for structural test-data generation. In: Proc. of the Int'l Conf. on Automated Software Engineering. 1998. 285−288. [doi: 10.1109/ASE.1998.732680]

附中文参考文献:

[17] 傅博.基于模拟退火遗传算法的软件测试数据自动生成.计算机工程与应用,2005,41(12):82−84.

[20] 王颖,谢剑英.一种自适应蚁群算法及其仿真研究.系统仿真学报,2002,14(1):31−33. [doi: CNKI:SUN:XTFZ.0.2002-01-009]

[21] 傅博.基于蚁群算法的软件测试数据自动生成.计算机工程与应用,2007,43(12):97−99.

侯可佳(1983－),女,山西榆次人,博士生,主要研究领域为软件测试.
E-mail: hkj09@mails.tsinghua.edu.cn

李树芳(1978－),男,工程师,主要研究领域为软件架构,软件测试.
E-mail: shufang@ustc.deu

白晓颖(1973－),女,博士,副教授,CCF 会员,主要研究领域为软件测试,服务计算.
E-mail: baixy@tsinghua.edu.cn

周立柱(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为数据库系统,数字图书馆,海量信息处理.
E-mail: dcszlz@tsinghua.edu.cn

陆皓(1979－),男,讲师,主要研究领域为软件测试.
E-mail: superlhao@gmail.com