基于抽象解释的服务间消息的数据约减

蒋曹清[1] 肖芳雄[1] 高 荣[1] 应 时[2] 文 静[2]
(广西财经学院计算机与信息管理系 南宁 530003)[1] (武汉大学软件工程国家重点实验室 武汉 430072)[2]

摘 要 面向服务软件中服务间消息的变量值可能存在无穷域的情况,从而导致模型检测时产生状态空间爆炸问题。为了使终止性验证在实践上可行,需要约减模型状态空间的大小,使得计算时间和空间需求合理。为此,基于抽象解释的区间抽象理论扩展了经典区间抽象域方法,并在统一的区间抽象域方法上借助异常控制流图对变量进行区间分析,在此基础上逆向分析得到服务间消息的变量区间集。变量区间上任意值相对于终止性验证是等价性,因此从每一个变量区间集中选取一个代表值,可组成服务间消息变量的约减值,从而为异常处理的终止性验证提供了约减的初始配置,有效避免了状态空间爆炸。

关键词 抽象解释,终止性验证,模型检测,数据约减
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2015.12.038

Data Reduction Analysis for Message between Services Based on Abstract Interpretation

JIANG Cao-qing[1] XIAO Fang-xiong[1] GAO Rong[1] YING Shi[2] WEN Jing[2]
(Department of Computer and Information Management, Guangxi University of Financial and Economics, Nanning 530003, China)[1]
(State Key Lab of Software Engineering, Wuhan University, Wuhan 430072, China)[2]

Abstract Infinite values of variable range of message between services may exist in service-oriented software, which results in the state space explosion situation when checking model. In order to make termination rerification feasible in practice, the size of the state space must be reduced which makes computing time and space demand reasonable. In this paper, based on interval abstract theory of abstract interpretation, we extended the classic interval abstract domain methods, carried out variable interval analysis on exception control flow graph with abstract domain methods of the unification interval. On this basis, interval set of message variables between services is obtained by the reverse analysis. An arbitrary value on a variable interval is equivalence relative to the termination of the verification, so reduction values of message variables between services can be gotten through selecting a representative value from every variable interval sets, which provides the reduced initial configuration for termination verification of exception handling, effectively avoides the state space explosion.

Keywords Abstract interpretation, Termination verification, Model check, Data reduction

1 引言

验证BPEL流程性质如终止性时,需要考虑服务间消息的数据变量的所有可能取值情况,然而BPEL流程服务间消息的变量取值范围往往很大,甚至是无穷域,这将导致采用模型检测等验证方法失效。因为模型的状态数量随着变量值数量的增加会呈现指数级增长。为了使验证BPEL流程性质在实践上可行,需要约减这些状态空间模型的大小,使得计算的时间和空间需求合理;同时,要求约减后的状态空间模型能够保持足够的信息,使得对可终止性属性检测时得到与原来模型相同的结果。当创建约减模型时,需要平衡这两个需求,即当从具体模型生成抽象模型时需要同时考虑模型约减时的精度和速度。

为了达到上述目标,本文根据抽象解释理论,首先在异常控制流图上分析变量值范围,然后针对服务间消息变量采用前向遍历找出该变量影响的变量值范围最小的区间划分集,然后对这些区间集中每一个区间采用逆向遍历,对区间值发生变化的地方进行逆运算(区间合并计算除外),直到分析服务间消息变量的输入节点,这样就获得服务间消息变量初始配置的区间集,即区间划分,显然求得的区间划分中所有变量值都有相同的执行路径,即都沿着相同的路径到达程序的出口。因而,本文可使用一个变量值代替这个区间,使服务

到稿日期:2015-02-11 返修日期:2015-03-25 本文受国家自然科学基础重点项目(91118003,61272113,61272108),国家自然科学基金项目(61070012,61170022,6126200),广西高校科学技术研究项目(YB2014349)资助。
蒋曹清(1973—),男,博士,副教授,主要研究方向为形式化方法、程序分析、软件体系结构、云计算、数据挖掘,E-mail:jcqng@163.com;肖芳雄(1971—),男,博士,教授,主要研究方向为软件工程、数据工程、云计算、电子商务;应 时(1965—),男,博士,教授,博士生导师,主要研究方向为面向服务和面向对象的软件开发、语义Web技术、软件体系结构和模式、软件的可重用性与互操作性等;文 静(1982—),女,博士后,主要研究方向为软件体系结构、面向方面软件开发、人机交互。

间消息变量值得到约减，从而实现异常处理可终止性的有效验证。

2 抽象解释基本理论

抽象解释理论是 1977 年 Cousot P 和 Cousot R 提出的一种能够处理计算机科学中不可判定问题和非常复杂问题的有效方法[1]。抽象解释理论的主要思想是对给定程序设计语言给出具体和抽象两种语义，并建立它们二者之间的映射关系，可通过对抽象语义的求解来实现保守计算具体语义的目的，使得程序抽象执行的结果能够部分反映程序的真实运行情况。

抽象域是对程序具体语义进行抽象而获得一种抽象语义，由具体语义中的各计算状态组成，包括抽象属性和抽象操作。抽象域是一个完备格，具有有穷的元素或者不存在无穷递增链，从而保证抽象解释的终止。抽象解释理论中的基本概念是对象域抽象、伽罗瓦连接(Galois connection)以及完备格上的拓宽(widening)算子等，下面给出它们的形式化定义。

定义 1 设$\langle L,\sqsubseteq\rangle$是一个偏序，$\langle L,\sqsubseteq\rangle$称为一个完备格当且仅当：集合 L 中任何一个子集均存在最小上界和最大下界。完备格$\langle L,\sqsubseteq\rangle$通常表示为$\langle L,\sqsubseteq,\sqcup,\sqcap,\bot,\top\rangle$，其中，$\sqcup$表示最小上界算子，$\sqcap$表示最大下界算子，用$\bot=\sqcup\varnothing=\sqcap L$表示 L 中最小元素，用$\top=\sqcap\varnothing=\sqcup L$表示 L 中最大元素。

定义 2 设$\langle P,\leqslant\rangle$和$\langle Q,\sqsubseteq\rangle$是两个偏序集合，$\alpha:P\to Q$和$\gamma:Q\to P$是两个映射，序偶$\langle\alpha,\gamma\rangle$称为一个 Galois 连接当且仅当：$\forall x\in P,\forall y\in Q,\alpha(x)\sqsubseteq y$当且仅当$x\leqslant\gamma(y)$，其中，映射$\alpha$称为抽象算子，$\gamma$称为具体算子。

根据 Galois 连接的定义，容易验证抽象算子α和具体算子γ具有如下性质：

(1) $\forall x\in P,x\leqslant\gamma(\alpha(x))$；

(2) $\forall y\in Q,\alpha(\gamma(y))\sqsubseteq y$；

(3)α和γ单调递增。

具有上述性质的抽象算子α和具体算子γ符合 Galois 连接的定义。可以从上界和下界两个方向抽象逼近对象集合。抽象偏序集$\langle Q,\sqsubseteq\rangle$中的对象集合 q 可以上界抽象逼近具体偏序集$\langle P,\leqslant\rangle$中的对象集合 p 当且仅当$p\leqslant\gamma(q)$，$\gamma(q)$是 p 的上界保证抽象逼近的可靠性；理论上，p 存在最优上界抽象逼近$\bigcap\{q\mid p\leqslant\gamma(q)\}$。抽象算子$\alpha$刻画了具体对象域和抽象对象域之间的最优上界抽象逼近映射关系，即抽象分析时$\alpha(p)$作为 p 的抽象逼近。首先，$\alpha(p)$是 p 的一个上界抽象逼近($\forall x\in P,x\leqslant\gamma(\alpha(x))$)；其次，若 q 是 p 的任意一个上界抽象逼近，即 $p\leqslant\gamma(q)$。由 Galois 连接的定义，则有$\alpha(p)\sqsubseteq q$($\forall x\in P,\forall y\in Q,x\leqslant\gamma(y)$当且仅当$\alpha(x)\sqsubseteq y$)，表明$\alpha(p)$是 p 的最优上界抽象逼近。

Galois 连接运算具有如下性质：

(1)两个 Galois 连接的合成是 Galois 连接；

(2)Galois 连接的广义笛卡尔乘积是一个 Galois 连接。

Galois 连接的上述性质表明可以连续地抽象程序语义，直至抽象域上的抽象计算满足计算复杂度的要求。

定义 3 设$\langle L,\sqsubseteq,\sqcup,\sqcap,\bot,\top\rangle$是一个完备格，算子$\nabla:L\times L\to L$称为 Widening 算子当且仅当：

(1)∇是一个上界算子，即$\forall l_1,l_2\in L,l_1,l_2\sqsubseteq l_1\nabla l_2$；

(2)对于任意的递增链$(l_n)n$，递增链$(l_n^{\nabla})n$都收敛，其中，$(l_n^{\nabla})n$定义为：如果 $n=0$，则$(l_n^{\nabla})=l_n$；否则，$(l_n^{\nabla})=(l_{n-1}^{\nabla})\nabla l_n$。

偏序集可以扩展成为完备格，然后可以为此扩展后的偏序集定义 Widening 算子。例如，区间抽象域可以扩展成为完备格，对于扩展得到的完备格$\langle L,\sqsubseteq,\sqcup,\sqcap,\bot,\top\rangle$，定义下列 Widening 算子$\nabla$：

若有 $m_1\leqslant m_2$，且 $n_1<n_2$，则有$[m_1,n_1]\nabla[m_2,n_2]=[m_1,\top]$；若有 $m_2<m_1$，且 $n_2\leqslant n_1$，则有$[m_1,n_1]\nabla[m_2,n_2]=[\bot,n_1]$；

若有 $m_1\leqslant m_2$，且 $n_2\leqslant n_1$，则有$[m_1,n_1]\nabla[m_2,n_2]=[m_1,n_1]$；若有 $m_2<m_1$，且 $n_1<n_2$，则有$[m_1,n_1]\nabla[m_2,n_2]=[\bot,\top]$。

定义 4 设$\langle L,\sqsubseteq,\sqcup,\sqcap,\bot,\top\rangle$是一个完备格，算子$\Delta:L\times L\to L$称为 Narrowing 算子当且仅当：

(1) $\forall l_1,l_2\in L$，若 $l_2\sqsubseteq l_1$，则 $l_2\sqsubseteq(l_2\triangle l_1)\sqsubseteq l_1$；

(2)对于所有的递降链$(l_n)n$，递降链$(l_n^{\triangle})n$收敛；其中，$(l_n^{\triangle})$定义为：如 $n=0$，则$(l_n^{\triangle})=l_n$；否则，$(l_n^{\triangle})=(l_{n-1}^{\triangle})\triangle l_n$。

在抽象解释理论中，使用抽象域上的计算来抽象逼近程序具体域上的迁移计算，当需要加快逼近时，使用 Widening 算子可以给出语义抽象的一个上界逼近，当计算精度不满足要求时，可以通过使用 Narrowing 算子给出该上界更精确的逼近。

3 数据类型的区间抽象域扩展

抽象解释的区间抽象理论可以有效地用于分析变量的取值范围，而已有基于抽象解释理论的变量区间分析方法一般仅针对数值型变量。为了获得 BPEL 流程的变量取值范围，我们提出了统一的区间抽象域定义方法。基于该方法可对 BPEL 流程及其伙伴服务的 WSDL 文档中的各种数据类型如字符串型、布尔型和异常类型定义区间抽象域；还可在 BPEL 流程的异常控制流图上，对扩展的各种数据类型变量采用相应的区间抽象域中定义的域操作，分析获得变量的取值范围。下面以异常类型为例说明给出的区间抽象域定义方法。

定义 5 异常类型区间代数抽象域为$\langle L_e,\sqsubseteq_e,\sqcup_e,\sqcap_e,\bot_e,\top_e\rangle$，其中 L_e 表示异常类型区间值的全集；$\bot_e$ 是异常类型区间集的最小值，表示区间集为空；$\top_e$ 是异常类型区间的最大值，指所有外部输入异常所构成的集合；$\sqcup_e$、$\sqcap_e$是最小上界算子和最大下界算子。

下面给出该抽象域的基本操作：

设 $E,E'\subseteq L_e$，Xe 为未赋值异常变量集。

(1)交(meet)与结合(join)

$E\sqcap_e E'=\{e\mid e\in E\wedge e\in E'\}$，特别地，$E\sqcap_e Xe=E$；

$E\sqcup_e E'=\{e\mid e\in E\vee e\in E'\}$，特别地，$E\sqcup_e Xe=E$。

(2)区间算术

设 $e\in E,e'\in E'$，则异常叠加运算$\oplus$定义为：

$e\oplus e'=e'\oplus e=ss'$，其中 ee'为异常 e 和异常 e'叠加后的组合异常。

$Xs\oplus s=s\oplus Xs=Xs$

对于任意的 e，判定它是否在异常类型区间 E 上：

$$e?\ E=\begin{cases}\text{true}, & \text{如果 } e\in E\\ \text{false}, & \text{否则}\end{cases}$$

(3)迁移函数

(a)赋值迁移函数:在抽象环境 $E^{\#}$ 下,赋值 $e=expre$ 的赋值迁移函数定义为:

$$[e:=expre]^{\#}E^{\#}\stackrel{\triangle}{=}E^{\#}[e\mapsto[expre]^{\#}E^{\#}]$$

其中,$[expre]^{\#}E^{\#}$ 表示在抽象环境 $E^{\#}$ 下采用上述异常类型区间算术操作来计算表达式 $expre$ 所得到的区间抽象值。

(b)测试迁移函数:设 $E^{\#}(e)=E\sqsubseteq L_e$,则有

$$[e=c]^{\#}E^{\#}\stackrel{\triangle}{=}\begin{cases}E^{\#}[e\mapsto\{c\}], & \text{如果 } e=c\\ \perp_e, & \text{否则}\end{cases}$$

注意:任意形式的测试条件约束均可以抽象成一个或多个形如 $e=c$ 的条件约束。

(4)Widening 与 Narrowing 算子

本文给出关于异常类型区间抽象域上的 Widening 算子 ∇_s 与 Narrowing 算子 $\triangle_s$ 的定义:

$$\perp_e\nabla_e E=E\nabla_e\perp_e\stackrel{\triangle}{=}E$$

$$E\nabla_e E'\stackrel{\triangle}{=}\{E\sqsubseteq E'\ ?\ Le:E\}$$

$$E\triangle_e E'\stackrel{\triangle}{=}\{E=Le\ ?\ E':E]$$

4 服务间消息的数据约减

本文采用类似基于抽象解释的传统变量值范围分析方法,在 BPEL 流程的异常控制流图上遍历分析得到各种数据类型的变量值范围。为了便于分析,本文首先给出 BPEL 流程异常控制流图的相关定义;然后再介绍基于抽象解释的值范围分析方法。分析时将主要考虑赋值活动(〈assign〉)、分支活动(〈if〉、〈flow〉)、循环活动(〈while〉、〈repeatUntil〉、〈forEach〉)、人消息活动(〈receive〉、〈invoke〉、〈pick〉)及异常抛出活动(〈throw〉、〈rethrow〉)情况下数据值的区间计算。从控制流图 entry 节点开始分析每一个节点处的变量值范围,并沿着控制流方向采用前向搜索算法,直到分析完 exit 节点为止。

4.1 BPEL 流程的异常控制流图的相关定义

BPEL 流程的异常控制流图是反映流程控制结构的有向图,一般可表示为〈$N,E,entry,exit$〉。其中,N 是活动节点的集合,E 是有向边的集合,$entry$ 为流程的唯一入口节点,$exit$ 为流程的唯一出口节点。

为了便于分析变量值范围,把节点集 N 定义为:$N=\{entry,exit\}\cup Assignments\cup Judgments\cup Junctions\cup Calls$。其中 $Assignments$ 为赋值活动节点集合,$Judgments$ 为条件判断节点集合,包括分支活动节点(if、switch)、循环活动节点(while、repeatUntil、forEach),$Junctions$ 为分支活动汇合点集合,汇合点可分为分支活动的简单汇合点和循环活动汇合点,即 $Junctions=SimpleJunctions\cup LoopJunctions$。$Calls$ 为服务调用点集合,包括 invoke、receive、pick 活动节点。

为了便于分析上述节点处的变量的值范围,需要首先给出如下概念。

定义 6 节点 n 的入边集为 $preEdges(n):N\to 2^E$,表示控制流图中以节点 n 为后继节点的有向边集;出边集为 $sucEdges(n):N\to 2^E$,表示以节点 n 为前驱节点的有向边集。

定义 7 有向边 e 的起始节点为 $originNode(e)$,表示有向边 e 的起始端节点;有向边 e 的终止节点为 $endNode(e)$,表示有向边 e 的终止端的节点。

定义 8 节点 n 的前驱节点集为 $preVertexes(n)$,表示控制流图中以节点 n 为终止节点的有向边的起始节点的集合;节点 n 的后继节点集为 $sucVertexes(n)$,表示控制流图中以节点 n 为起始节点的有向边的终止节点的集合。

定义 9 对于条件判断节点 $j\in Judgments$,当条件表达式取值为"真"和"假"时,分别对应后继边 $sucEdge-T(j)$ 和 $sucEdge-F(j)$,即 $sucEdges(j)=\{sucEdge-T(j),sucEdge-F(j)\}$。

定义 10 流程的各变量值范围由控制流图中各条边上的上下文元组组成,上下文关系定义为 $C\in Variables\times Intervals$,由各个变量的名称-区间值对组成,例如,$C_1=\{(x,[6,12]),(a,\{\text{TRUE}\})\}$。

定义 11 在上下文 C 中,变量 x 的取值区间为

$$C(x)=\begin{cases}interval, & \text{如果存在 } x\neq\perp\wedge(x,interval)\in C\\ \perp, & \text{否则}\end{cases}$$

对于上面例子中的 C_1,有 $C_1(x)=[6,12]$,$C_1(a)=\{\text{TRUE}\}$,$C_1(y)=\perp$。

定义 12 对于上下文 C 和 C',$C\sqsubseteq C'$ 当且仅当对于 $\forall x\in Variables$,$C(x)\sqsubseteq C'(x)$。

4.2 服务间消息变量的值范围分析

基于经典抽象解释理论,以获得的伙伴服务的服务间消息变量的值范围作为异常控制流图中 calls 节点变量值的初始区间,沿控制流的 entry 节点开始逐步分析流程中活动的各变量的取值范围。分析的基本思路是:沿着异常控制流图中有向边的方向进行分析,即每个节点 n 上的取值范围由指向 n 的有向边集 $preEdges(n)$ 的各条边上的各个变量值范围确定。由于流程中活动类型不相同,其在控制流图中对应的控制流结构会有所不同,因而使得计算后继节点/边上的上下文方法不相同。下面将首先给出各个活动对应控制流图上变量的取值范围分析方法,然后在此基础上给出变量值范围分析算法。

(1)〈assign〉活动

〈assign〉活动结构中包括〈copy〉赋值成分和数据更新操作。下面根据这两种情况给出〈assign〉活动中变量值范围分析方法。

①〈assign〉活动结构〈copy〉赋值成分的变量值范围分析。首先计算 from 分支表达式的值 $intervalc$,然后得到 to 分支变量 x 的值 $x=intervalc$,并在该活动节点及后继边上下文中增加 to 分支中变量及其变量范围,即增加 $\{x,intervalc\}$。其它变量在该上下文中保持不变。

②〈assign〉活动结构〈copy〉数据更新操作的变量值范围分析。首先计算 from 分支表达式的区间值 $intervalu$,然后得到 to 分支变量 x 的值 $x=intervalu$,并在该活动节点及后继边上下文中修改 to 分支中变量及变量范围,即把变量的值范围 $\{x,[a,b]\}$ 修改为 $\{x,intervalu\}$,其中,$[a,b]$ 为 x 原来的值范围。其它变量在该上下文中保持不变。

(2)〈if〉活动

〈if〉活动节点的上下文为节点所有入边上下文中的所有变量值区间的结合(join),具体结合方法则根据变量的类型采用相应的结合计算方法。〈if〉活动后继边上下文的计算则需

要根据表达式的结果，采用不同的方法。①如果该条件表达式的值仅为真，则〈if〉活动后继真值边上的变量数值范围为〈if〉活动节点的上下文，且用条件表达式为真的各变量值(区间)代替该变量的原来值(区间)。②如果该条件表达式的值仅为假，则〈if〉活动后继假值边上的变量数值范围为〈if〉活动节点的上下文，且用条件表达式为假的各变量值(区间)代替该变量的原来值(区间)。

〈if〉活动节点的变量值范围分析方法是求节点的所有前驱边的上下文并集，即求各个变量值范围的结合值。

(3)〈while〉活动、〈repeatUntil〉活动、〈forEach〉活动

〈while〉活动、〈repeatUntil〉活动、〈forEach〉活动的变量取值范围计算方法类似于〈if〉活动。不同的是，在〈while〉活动、〈repeatUntil〉活动、〈forEach〉活动的汇合节点处的抽象值是每次迭代过程中为本次迭代的抽象值和上次迭代的抽象值的结合或 Widening；当汇合节点稳定后，变量值范围分析才可以继续向前传播。设前一次迭代分析时汇合节点处的抽象值为 Q_1，当前迭代分析时的抽象值为 Q_2，即当且仅当 $Q_2 \sqsubseteq Q_1$ 时，对循环活动的迭代才终止，否则在汇合节点处继续进行 Widening 或结合操作。

(4)〈receive〉活动

〈receive〉活动结构中给出了多个服务间消息变量名 $Variable_1, Variable_2, \cdots, Variable_n$，它的值范围为〈receive〉活动的伙伴服务的 WSDL 文档中相应操作的输出消息变量的值范围 $Interval_1, Interval_2, \cdots, Interval_n$。因此〈receive〉活动节点及后继边的上下文添加上述服务间消息变量名及值范围，即添加 $\{(Variable_1, Interval_1), (Variable_2, Interval_2), \cdots, (Variable_n, Interval_n)\}$。

(5)〈invoke〉活动

如果采用请求-响应方式调用伙伴服务，则当收到响应时〈invoke〉活动结束。如果采用单向方式调用伙伴服务，则当请求消息发出后〈invoke〉活动结束。故当采用单向方式调用伙伴服务时，该节点及后继边的上下文没有变化；当采用请求-响应方式调用伙伴服务时，该节点及后继边的上下文分析方法与〈receive〉相同。

(6)〈pick〉活动

当某个触发条件发生时，则执行相应的子活动；当子活动结束时，则〈pick〉活动完成。故该节点及后继边的上下文分析方法与〈receive〉相同。

(7)〈throw〉活动

〈throw〉活动在一个流程内生成一个异常信息，异常信息中包括异常变量 *exceptionVariable* 及异常名 *exceptionName*。故该节点及后继边的上下文增加异常变量及异常名，即增加 $\{(exceptionVariable, exceptionName)\}$。

(8)〈rethrow〉活动

〈rethrow〉活动节点及后继边的上下文处理方法是首先确定该节点所在的异常处理程序捕获的异常，然后采用与〈throw〉活动相同的处理方法。

上述内容给出了各种影响变量值范围的有关具体活动的变量值范围分析方法，下面将在此基础上给出 BPEL 流程的变量值范围分析算法(见算法 1)。

设 BPEL 流程 P 对应的异常控制流图为 ECFG，这里的 ECFG 指考虑了异常处理程序控制依赖关系的控制流图，其伙伴服务服务间消息变量集为 SV，则 P 中各变量的取值范围分析过程如算法 1 所示。

算法 1 基于抽象解释的 BPEL 流程变量值范围分析算法

输入：流程 P 的异常控制流图 ECFG 及伙伴服务消息变量集 SV 及其取值范围

输出：带变量取值范围信息的异常控制流图

```
1. for each e∈E
2.    localContext(e)=∅
3. junctions=∅ //汇合节点集
4. edges=sucEdge(entry) //entry 表示消息输入点，edges 表示输入点的待计算的边集
5. for each p∈SV
6.    localContext(edges)=localContext(edges)∪{(p,X)} //把每个消息变量及取值加入输入点的相邻边，即根据 calls 节点对应的伙伴服务输入消息 sa∈SA 的信息更新当前上下文环境
7. while (edges!=∅){ //针对每条边的终止节点类型，采用不用的方法求该节点的后继边上下文
8.       Edge inEdge=edges.delete( )
9.       Context C=C′=localContext(inEdges)
10.      Node n=endNode(inEdge)
11.      if (n∈Junctions) junctions=junctions∪{n} //收集汇合节点
12.      case n in
13.            Assignment:
14.               Variable v=id(n) //id(n)是左端被赋值变量，expre(n)是右端赋值表达式
15.                  updateRange(sucEdge(n),(C-{(v,C(v))})∪(v,interval(expr(n)))
16.            Judgment:
17.               C=C′=localContext(inEdge)
18.               for each var in n
19.                        C=(C-{var,C(var)})∪(v,C(var)∩interval(Bexpr(n),var)) //Bexp(n)是节点n包含的条件表达式
20.                        C′=(C′-{var,C′(var)})∪(v,C(var)∩interval(!Bexp(n),var))
21.               updateValueRange(sucEdge_T(n),C) //更新真值分支的变量值范围
22.               updateValueRange(sucEdge_F(n),C′) //更新假值分支的变量值范围
23. for each n in junctions
24.    for each var in preEdge(n)
25.       outContext(var)=∪ e∈preEdge(n) localContext(e,v)//汇合节点处各个变量的值范围
26.    if (outContext!=localContext(sucEdge(n)))
27.       case n in
28.          SimpleJunctions:
29.             updateValueRange(sucEdge(n),outContext)
30.          LoopJunctions:
31.             for each var in n
32.                updateValueRange(sucEdge(n),(localContext(sucEdge(n)) ▽outContext(var))
33.    junctions=∅
```

算法 1 从异常控制流图中 entry 节点的出边集开始，首先针对服务调用节点，根据伙伴服务输入消息 *sa* 的信息为各

条边添加调用服务接收消息参数的取值信息；接着循环处理待测边集合 *edges* 中的边，根据当前边的终止节点 *n* 所描述的不同活动类型进行相应的计算：如果是汇合节点，将当前节点加入汇合节点集合 *junctions* 中；如果是赋值活动节点，将左端被赋值变量的值更新为右端表达式的值；如果是条件活动节点，分别计算条件表达式为“真”值和“假”值时各变量的取值情况，然后更新对应的“真”值分支和“假”分支上的变量值。如此不断循环重复，直到待分析节点集合为空，这些计算对应算法 1 的 7－29 行。对于汇合节点集合中的元素：如果是条件分支汇合节点，则将其各入边上的上下文进行合并；如果是循环活动的汇合节点，通过 Widening 算子更新出边上的上下文（见算法 2）。在处理汇合节点的过程中，有些边的上下文会发生变化，将这些边重新加入队列 *edges* 中进行处理。如此循环重复，直到异常控制流图中所有边上的上下文不再发生变化，算法终止。

算法 2 更新值范围算法 updateValueRange

输入：待更新的边 e 及该边上的上下文 c

输出：更新上下文后的边 e

```
1. if (c!=localContext(e)
2.     localContext(e)=c∪localContext(e)
3.     edges=edges∪e
```

通过上述算法 1、算法 2 可得到异常控制流图中各边的上下文环境。显然算法是可终止的，理由是：由抽象解释理论中 Widening 算子的定义可知，异常控制流图中各循环汇合节点的出边上的上下文序列构成一个严格上升链，该链一定收敛，即算法 1、算法 2 一定是可终止的。

在上述基于异常控制流分析的前向算法（即算法 1、算法 2）中，整个 BPEL 流程可以看成一个有限长度的活动序列，因而根据活动序列[a1;a2]的迁移计算可表示从流程的初始接收活动节点开始，按照流程的执行顺序依次对流程活动进行迁移计算，直到到达其不动点。

4.3 服务间消息变量的数据约减分析

为了有效地配置 BPEL 形式语义模型，以避免模型检测时可能出现的状态空间爆炸问题，需要对各个服务间消息变量的值进行约减分析。以下将首先给出服务间消息变量的数据约减分析的基本思路，然后在此基础上给出具体的数据约减分析算法。

服务间消息变量的数据约减分析是在获得 BPEL 流程的变量值范围后，采用先前向、后逆向的遍历方式分析控制流图，以获得服务间消息变量的区间划分，然后给出服务间消息变量的约减值。其基本思路是：

(1)从带有变量值上下文的异常控制流图的 entry 节点开始，沿着控制流图前向遍历到异常控制流图的 exit 节点，找出并标注控制流图路径上的所有最后条件判断节点，即〈if〉活动节点、〈while〉活动节点、〈repeatUntil〉活动节点、〈forEach〉活动节点。这里最后条件判断节点指左右分支最多只能被一个分支覆盖的节点。但对于只有一个分支可能被满足的分支节点，只要该分支被满足，即被认为不是最后分支节点。

(2)从这些最后条件判断节点开始，针对每个最后条件判断节点的每条出边上的上下文，沿着控制流图逆向分析，并传播该上下文。在传播上下文的过程中：

①如遇到赋值节点，即〈assign〉活动节点、〈throw〉活动节点、〈rethrow〉活动节点，则对赋值节点出边上的上下文逆向计算以获得赋值前的相应区间。

②如遇到条件判定节点，则对条件判定节点出边上的上下文中同一变量的不同区间不予合并，原样向上传播；对条件判定节点出边上的上下文中同一变量的两个相同区间，保留一个并原样向上传播。

③如遇到汇合节点，则根据该汇合节点类型不同，采用不同的逆向计算方法：如是简单汇合节点，即〈if〉活动汇合节点，采用汇合前汇合点汇合方法逆向拆分汇合区间。对于循环汇合节点，即〈while〉活动节点、〈repeatUntil〉活动节点、〈forEach〉活动节点，需要采用 Widening 算子的逆运算 Narrowing 算子逆向计算第一次汇合前汇合点的值范围。

④如遇到消息输入节点，即〈receive〉活动、〈invoke〉活动、〈pick〉活动，则该节点的服务间消息变量的范围区间传播分析结束。

(3)依此类推，直到分析到达 *entry* 节点，这样就获得了所有服务间消息变量值区间范围集。

(4)针对每一个服务间消息变量值区间范围集进行分析，如果变量区间范围均没有重叠，则可终止性验证初始配置时可在每个区间取一个典型代表值。例如区间是数值型区间[3,6]，则取其内一个整数值 5。

(5)如果在求得抽象区间的变量数值范围的有效划分后，对一些服务间消息变量的范围区间集只有一个为该变量初始值区间，就说明该变量对可终止性验证没有影响，验证配置时可以去掉（注意：需要在着色 Petri 网中去掉与该变量有关的设施）。

根据服务间消息变量在异常控制流图上的数据约减分析的基本思路(1)－(3)，本文将归纳得到其相应的具体约减分析算法（见算法 3）。

算法 3 服务间消息变量的数据约减分析算法

输入：带有变量值范围的异常控制流图 ECFG

输出：服务间消息变量的数据抽象区间集

```
1. judgementNodes=travelECFG(ECFG);//获得所有控制流路径中
   的最后一个条件判定节点
2. foreach n∈judgementNodes
3.   if n is AssignmentNodes
4.       foreach e∈sucEdge(n)
5.           localContext(e)→localContext(preEdge(n));//逆运
             算，求赋值节点被赋值变量赋值前的值范围
6.   if n is JugdementNodes
7.     e_t,e_f∈sucEdge(n);
8.       foreach x∈localContext(e_t)∧x∈localContext(e_f)
9.         if e_t.x.interval=e_f.x.interval
10.            localContext(preEdge(n)).x.interval=e_t.x.interval;
11.        else
12.            localContext(preEdge(n)).x.interval={e_t.x.in-
               terval,e_f.x.interval};
13.      foreach x∈localContext(e_t)∨x∈localContext(e_f)
14.            localContext(preEdge(n)).x.interval={e_t.x.in-
               terval,e_f.x.interval};
15.  if n is Junctions
16.    if n is SimpleJunctions
```

```
17.    if x∈localContext(n) ∧ x. interval= interval1 ∪interval2
       //interval1、interval2 分别汇合分支汇合前区间
18.        localContext(preEdge(n)1). x. interval= interval1, lo-
           calContext(preEdge(n)2). x. interval=interval2;
19.        else localContext(preEdge(n)). x. interval=localContext
           (n). x. interval;
20.    if n is LoopJunctions
21.        localContext(preEdge(n))=(localContext(n))△local-
           Context(preEdge(n));//进行 Narrowing 运算
22.  if n is Calls //消息输入节点
23.      localContext(preEdge(n))=localContext(n);
24.      foreach x∈localContext(n) ∧ x∈messageVariables
25.          localContext(preEdge(n))=localContext(preEdge
           (n))-localContext(n). x;
```

5 相关工作

近年来，随着 Web 服务技术的飞速发展，用 BPEL 等流程语言编写面向服务软件逐渐成为一种新的开发范式，但由于该类软件运行环境复杂多变，导致软件容易产生各种异常，需要复杂的异常处理机制。为了确保异常处理的正确性，如何验证异常处理的可终止性等问题逐渐引起了国内外研究者的关注。Weissbach 与 Zimmermann 认为面向服务软件可终止性验证与一般软件可终止性验证的不同之处是 Web 服务的实现不知道它们客户端的终止性情况[2,3]。因而，面向服务软件的可终止性验证不能直接依照以前的方法，特别是不能通过验证被调用服务的源代码而获得其可终止性信息，而只能依赖被调用服务的相应 WSDL 文档提供的可终止性信息。以上研究虽然给出了 BPEL 流程验证需要给出的假定条件，本文研究也借鉴了这一思想，但他们的可终止性验证的思想仅沿用传统的程序可终止性证明方法，而本文采用了模型转换和模型检测思想对异常处理可终止性进行了验证。

抽象解释是一种针对系统语义的可靠近似理论，可以有效地用于分析变量的取值范围[4]。基于抽象解释的变量值范围分析是一种把区间范围传播和区间范围分析结合起来的分析方法，因而它能较好地提高变量值范围分析的精度[5]。Cousot 等人全面阐述了抽象解释理论[6]，其中部分内容涉及到变量值范围分析；文献[7]给出了一个基于抽象解释和单调数据流框架的变量值范围分析框架。然而，已有基于抽象解释理论的变量值范围分析方法一般仅针对数值型变量，因而需要扩展抽象解释理论，使得其能够对其它类型数据变量求变量值范围，即需要在抽象解释框架下定义其它类型数据变量的区间抽象域。Clarke 等描述了一种使用抽象技术以约减时序逻辑复杂性的方法[8]，该方法中采用的技术类似抽象解释中涉及的技术，他基于该方法构建了一个程序抽象模型且实现了一个验证系统，同时通过大量的例子说明了这些方法的实用性。本文基于前期异常处理的建模和验证工作[9,10]，探讨如何约减服务间消息变量的数据值，以避免模型检测异常处理终止性时的状态空间爆炸问题。

结束语 本文针对服务间消息变量的数据约减问题，首先扩展了数据类型区间抽象域；其次在统一的数据类型区间抽象域中借助异常控制流图分析了变量值范围；在此基础上对服务间消息变量进行了数据约减分析。通过该方法，可以获得服务间消息变量的约减值，从而避免状态空间爆炸。未来的研究工作是，继续完善服务间消息的数据约减方法、异常处理可终止性验证方法，提高异常处理建模、数据约减的精确性及验证结果的准确性，并进一步提高该方法的自动化程度。

参 考 文 献

[1] Cousot P, Cousot R. A gentle introduction to formal verification of computer systems by abstract interpretation[M]// Esparza J, Grumberg O, Broy M, et al. , eds. Logics and Languages for Reliability and Security, NATO Science Series III: Computer and Systems Sciences. IOS Press, 2010: 1-29

[2] Weissbach M, Zimmermann W. A service-level agreement approach towards termination analysis of service-oriented systems [C]// International Conference on Cloud Computing, GRIDs, and Virtualization. Rome, Italy, 2011: 26-31

[3] Weissbach M, Zimmermann W. Termination analysis of business process workflows [C]// Proceedings of the 5th International Workshop on Enhanced Web Service Technologies, Ayia Napa (Cyprus, 2010). New York: ACM, 2011: 18-25

[4] 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
Li Meng-jun, Li Zhou-jun, Chen Huo-wang. Program Verification Techniques Based on the Abstract Interpretation Theory [J]. Journal of Software, 2008, 19(1): 17-26

[5] Dams D. Abstraction in software model checking: principles and practice[C]// Proceedings of the 9th International SPIN Workshop: Model Checking Software. LNCS 2318, 2002: 14-21

[6] Cousot P, Cousot R. An abstract interpretation framework for termination[C]// Conference Record of the 39th ACM SIGPLAN SIGACT Symposium on Principles of Programming Languages. ACM Press, 2012: 245-258

[7] 姬孟洛，王怀民，李梦君，等. 一种基于抽象解释和通用单调数据流框架的值范围分析方法[J]. 计算机研究与发展，2006，43(11)：2020-2026
Ji Meng-luo, Wang Huai-min, Li Meng-jun, et al. An Value Range Analysis Based on Abstract Interpretation and Generalized Monotone Data Flow Framework[J]. Journal of Computer Research and Development, 2006, 43(11): 2020-2026

[8] Clarke E M, Grumberg O, Long D E. Model checking and abstraction[J]. ACM Transactions on Programming Languages and Systems, 1994, 16(5): 1512-1542

[9] 蒋曹清，应时，文静，等. 面向服务软件中异常处理的形式化建模方法[J]. 西安交通大学学报，2013，47(4)：118-124
Jiang Cao-qing, Ying Shi, Wen Jing, et al. A Modeling Approach for Exception Handling in Service-Oriented Software[J]. Journal of Xi'an Jiaotong University, 2013, 47(4): 118-124

[10] 蒋曹清，应时，文静，等. 面向服务软件异常处理过程的可终止性验证[J]. 计算机科学与探索，2012，6(3)：208-220
Jiang Cao-qing, Ying Shi, Wen Jing, et al. Verification of Termination for Exception Handling Process in Service-Oriented Software[J]. Journal of Frontiers of Computer Science & Technology, 2012, 6(3): 208-220