论文题目

XCFG Based Data Flow Analysis of Business Processes

论文作者

S. Ji, B. Li and P. Zhang

发表期刊信息

2019 5th International Conference on Information Management (ICIM), Cambridge, United Kingdom, 2019, pp. 71-76.

技术问题

本文提出了一种基于XCFG（扩展控制流图）的方法来分析BPEL规范中定义的业务流程的数据流。通过对BPEL过程的XCFG建模，定义了改进的定义方程，并考虑了XCFG的新特性。然后采用迭代算法求解方程，计算出可能到达每个XCFG节点的定义。

现实背景

随着云计算的快速发展，大量基于服务的业务流程被开发出来。BPEL通过以某种逻辑顺序描述流程和合作伙伴服务之间的交互来定义组合服务来描述基于服务的流程业务的规范。由于BPEL规范的复杂特性，BPEL业务流程很容易出错。数据流的设计和构造尤其容易出错，因为它隐含在业务流程中。因此，数据流测试对于确保BPEL流程的正确性起着重要的作用。而数据流分析是数据流测试的前提任务。此外，它还支持数据流异常检测、验证和优化。

作者思路

本文提出了一种基于XCFG（扩展控制流图）的方法来分析BPEL规范中定义的业务流程的数据流。首先，通过构建XCFG对BPEL流程进行精确建模，实现数据流分析的自动化。XCFG模型显式地描述了BPEL流程的控制流，并对XCFG元素进行了注释，以便进行数据流分析。然后定义了用于BPEL流程到达定义分析的改进方程。采用迭代算法得到可能到达每个XCFG节点的定义。

解决方案

XCFG MODEL

XCFG是一种控制流图，可以直观地描述BPEL流程的工作流。

数据流分析

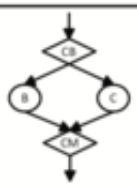
Case	Traditional Approach	Practical Analysis	Our Approach
Out(CB)={(x,A)} Gen(B)={(x,B)} Kill(B)={(x,A)} Gen(C)={(y,C)} Kill(C)= ∅ 	In(B)={(x,A)} Out(B)={(x,B)} In(C)={(x,A)} Out(C)={(x,A),(y,C)} In(CM)={(x,A),(x,B),(y,C)} Out(CM)={(x,A),(x,B),(y,C)}	In(B)={(x,A),(y,C)} Out(B)={(x,B),(y,C)} In(C)={(x,A),(x,B)} Out(C)={(x,A),(x,B),(y,C)} In(CM)={(x,B),(y,C)} Out(CM)={(x,B),(y,C)}	In(B)={(x,A)} Out(B)={(x,B)} In(C)={(x,A)} Out(C)={(y,C)} In(CM)={(x,B),(y,C)} Out(CM)={(x,B),(y,C)}

Figure 1. Reaching definitions of concurrency activity.

Algorithm 1 calcReachingDefinitions()

Input:
$xcfg$: the XCFG model of BPEL process;
Output:
$In()$: the definitions reaching before each XCFG node;

```
1: get the set of nodes in xcfg: nodeSet;
2: for node ∈ nodeSet do
3:     identify the definitions generated by node: Gen(node);
4:     identify the definitions killed by node: Kill(node);
5:     mustKill(node) = ∅;
6: end for
7: identify the end node of concurrency activity and the target node of link:
   reqAnlNodes;
8: for node ∈ reqAnlNodes do
9:     compute the nodes that are enclosed in the concurrency structure and execute
       before n : mustNodes;
10:    for node' ∈ nodeSet do
11:        if node' ∈ N_ED then
12:            computes the child activities enclosed in the choice activity:
               ifNodes;
13:            mustNodes = mustNode − ifNodes;
14:        end if
15:    end for
16:    for node' ∈ mustNodes do
17:        mustKill(node) = mustKill(node) ∪ Kill(node');
18:    end for
19: end for
20: initiate In() and Out() of each node as ∅;
21: while Out() of a node changes do
22:    for node ∈ nodeSet do
23:        Out(node) = Gen(node) ∪ (In(node) − Kill(node));
24:        for node' ∈ pred(node) do
25:            In(node) = ∪Out(node');
26:        end for
27:        In(node) = In(node) − mustKill(node);
28:    end for
29: end while
```

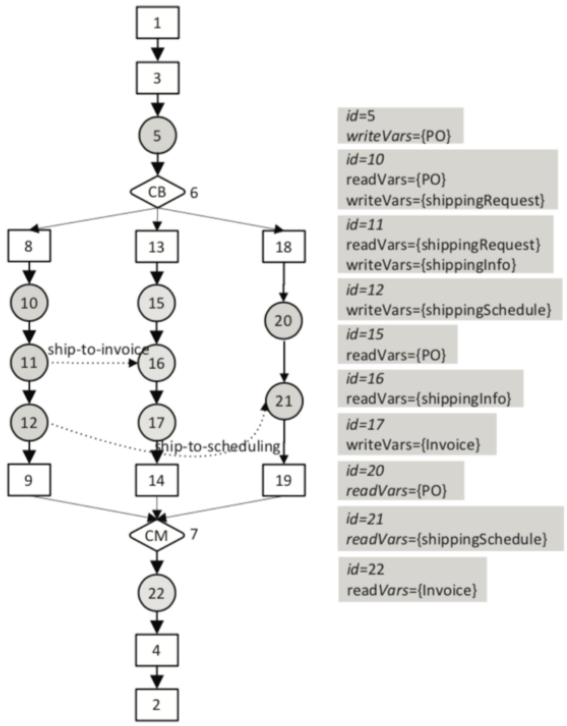
创新贡献

- 提出了一种基于XCFG的BPEL流程数据流分析方法，并给出了具体算法。
- 给出了该方法的正确性分析。
- 案例研究表明了该方法的有效性和实用性关于异常检测和数据流测试。

效果评价

POP的XCFG模型如图所示。附加到每个XCFG节点的编号是它的id文件。访问变量的灰色框中列出了一些XCFG节点的详细字段。将算法 `calcReachingDef()` 应用于POP，其中节点用id字段表示。数据流分析结果与人工分析结果基本一致，说明了该方法的有效性。随着每个XCFG节点定义的深入，可以检测到数据流异常，并为数据流测试准备定义使用对。

node	In(node)
$6, 8, 10, 13, 15, 18, 20$	$(PO, 5)$
11	$(PO, 5), (shippingRequest, 10)$
$12, 16, 17$	$(PO, 5), (shippingRequest, 10), (shippingInfo, 11)$
$9, 21, 19$	$(PO, 5), (shippingRequest, 10), (shippingInfo, 11), (shippingSchedule, 12)$
14	$(PO, 5), (shippingRequest, 10), (shippingInfo, 11), (Invoice, 17)$
$7, 22, 4, 2$	$(PO, 5), (shippingRequest, 10), (shippingInfo, 11), (shippingSchedule, 12), (Invoice, 17)$



个人感想

本文的优势

本文定义了基于XCFG模型的BPEL流程数据流分析的定义达成的改进方程，并采用迭代算法求解。该方法虽然不完善，但对检测数据流异常和构造定义使用依赖关系没有负面影响。实例分析表明了该方法的有效性。

本文的劣势

- 应用本文方法可能会导致并发结构中未关闭的子活动具有不准确的到达定义。
- 在BPEL中尚未定义的DPE，区分复杂类型变量的部分和整体的定义/使用。
