计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210100089

基于结构和行为语义的子流程发现方法

孙善武　王　楠
吉林财经大学管理科学与信息工程学院　长春 130117
(ctusunshanwu@126.com)

摘　要　越来越多的企业以流程模型的形式进行业务管理,并且要求从不同的细节层描述相同的流程。业务流程模型抽象技术对于给定的细节流程模型,生成该模型的抽象表示,从而达到特定的抽象目标。构造流程的"概要视图"以加速对复杂流程的理解是流程抽象的一个需求最显著的用例,其中一个关键问题是将初始模型中的细节行为集合转换成抽象模型中的粗粒度的行为(子流程)。很多学者对流程抽象方法进行了研究,其中以基于结构的抽象方法居多。结构抽象方法中,初始模型中待抽象的行为集合仅仅根据控制流关系得到,没有考虑到行为的域语义,生成了大量的业务逻辑性不完整的待抽象的流程片段作为候选子流程。在流程的控制流结构基础上,引入行为语义信息,自底向上对流程结构树中每层的标准部件进行语义扩展,将标准部件与其相邻的兄弟节点之间的语义进行相似性评估,发现与标准部件语义描述最相似的行为集合,并有选择地聚合得到最接近包含完整业务含义的流程片段作为候选子流程。文章利用真实的流程案例对基于结构的抽象方法(将流程结构树中每个标准部件作为候选子流程)和提出的子流程发现方法做实验对比分析,提出的方法大大减少了不相关候选子流程的数量,生成的待抽象流程片段更加接近人工设计的子流程。

关键词:子流程发现;模型抽象;行为相似性;流程结构树;流程行为语义
中图法分类号　TP391

Subprocesses Discovery Based on Structure and Activity Semantics

SUN Shan-wu and WANG Nan
College of Management Science and Information Engineering,Jilin University of Finance and Economics,Changchun 130117,China

Abstract　More companies document their business operations in the form of process models,and require descriptions of one process on various levels of detail. Given a detailed process model,business process model abstraction (BPMA) delivers abstract representations for the same process to achieve particular abstraction goals. A prominent BPMA use case is a construction of a process "quick view" for rapidly comprehending a complex process. A key problem in this abstraction scenario is the transition from detailed activities in the initial model to coarse-grained activities. Many researchers have studied the methods of process abstraction most of which are based on structure. The structure-based abstraction derives the set of activities to be abstracted from the original model according only to control flow relations,but not considering the domain semantics,resulting in many process fragments (candidate subprocess) with incomplete business logicality. This paper bases on the structure of process and activity semantics to extend every canonical component of the process structure tree from the bottom up. This paper assesses the similarity between the canonical component and all its adjacent nodes to find out the group of activities which is the most similar to the canonical component. And then it aggregates the generated group of activities to derive the process segments as candidate subprocesses. This paper uses real-running process cases to compare structure-based abstraction method which takes every canonical component as a candidate subprocess with the proposed method. The results show that the number of the subprocesses with uncompleted business meaning is greatly reduced and the generated groups of activities are more similar to manually designed subprocesses.

Keywords　Subprocess discovery,Model abstraction,Activities similarity,Process structure tree,Process activity semantics

基金项目:吉林省教育厅"十三五"社会科学研究项目(JJKH20200156SK);吉林省教育科学"十三五"规划基金项目(ZD20024);吉林省教育厅"十三五"科学技术研究项目(JJKH20210131KJ);国家自然科学基金项目(61702213)
This work was supported by the 13th Five Year Plan Social Science Research Project of Jilin Provincial Department of Education (JJKH20200156SK),13th Five Year Plan of Educational Science in Jilin Province(ZD20024),13th Five Year Plan Science Research Project of Jilin Provincial Department of Education(JJKH20210131KJ) and National Natural Science Foundation of China(61702213).
通信作者:王楠(ctuwangnan@126.com)

1 引言

业务流程模型抽象(Business Process Model Abstraction, BPMA)是保留流程的某些重要属性,忽略那些无关的流程细节,从而达到获得与特定抽象目标相关的信息的目的[1],在BPMA的应用中,最显著的用例是构造流程的"概要视图"以加速对复杂流程的理解[1-2]。BPMA用删除(elimination)和聚合(aggregation)两个基本操作[3-5]实现从初始模型到抽象模型的转换,删除操作简单地从模型中去掉一些元素,而聚合操作则将一组低层模型元素合并为结果抽象模型中的一个粗粒度的抽象元素[1]。聚合操作是本文主要考虑的抽象运算。在"概要视图"的应用场景下,很多学者对抽象方法进行了大量研究,其中以基于结构的抽象方法[3-9]居多。结构抽象方法中,初始模型中待抽象的行为集合是根据控制流关系得到,并未考虑到行为的域语义[10],很难满足子流程或子任务的业务逻辑性及相关标准。

为了克服基于结构的业务流程模型抽象方法的局限,很多学者提出基于行为语义的流程抽象,如Smirnov等[1]研究描述行为及其之间关系的域本体对行为聚合的支持,这个方法是半自动的,而且要求事先确定域本体的部分-整体关系。文献[11]在包含丰富子流程的模型集合中挖掘人工设计子模型的行为聚集标准,模拟人的行为对模型进行聚合抽象。但由于标准不通用,对一个域的模型集合获得的聚合标准只能应用在该域的流程模型上。文献[12]给出了确定与某一个行为功能相似的一组行为的方法,但是这个方法是在不同的模型之间进行匹配,无法直接应用于单个模型内部的行为抽象。根据文献[13],一旦行为集合聚合成一个子流程,则需要考虑多方面的因素。其给出了子流程发现的3个标准:块结构化标准将RPST(Refined Process Structure Tree)树[9]中分解得到的标准部件作为候选子流程;连接化标准利用图聚类分析[19]建立流程中相互强连接的节点集合;标签相似性将具有相似标签的行为集合作为候选子流程。Sun等将半监督聚类技术和模糊聚类技术引入到BPMA中[15-17],将行为语义和控制流一致性需求结合,设计相应的约束函数指导行为聚类对流程进行抽象,但是缺少对流程结构逻辑一致性的考量,从而无法由聚类结果自动生成满足结构约束的流程块。

业务流程通常要求具有单入口、单出口[18],比如块结构语言BPEL[18]的结构本身就强制了这个要求,其他一些语言[19-20]也将这个要求作为附加需求。因此,块结构化标准可以作为发现子流程的一个很好的基础[13]。子流程具备流程的结构性质,本文基于RPST树,提取标准部件对应的流程片段的候选语义描述,结合子流程的标签相似性标准,发现与标准部件语义描述最相似的行为集合作为待聚合行为集合,并根据已经存在的不同子流程之间的语义相似性确定相似性阈值对该行为集合进行限制,最终确定与标准部件聚合的行为并与之聚合生成新的流程片段作为候选子流程。文中通过真实的流程案例对基于结构的抽象方法(将RPST树中每个标准部件作为候选子流程)和提出的结合语义的抽象方法做实验对比分析,从中得到一些结论和未来工作的方向。

2 相关概念

业务流程模型抽象往往指行为抽象,即低层步骤转换成高层任务[7]。因此,我们引用文献[7]和文献[10]中对流程模型相关概念的定义。

定义1 一个业务流程模型定义为一个元组 $PM=(A,G,F,t,s,e)$,其中:

(1)A 是有限非空行为(activities)集合。

(2)G 是有限的网关(gateways)集合。

(3)$N=A\cup G$,是有限节点集合,且 $A\cap G=\emptyset$。

(4)$F\subseteq N\times N$ 表示流关系,使得(N,F)是一个连接图。

(5)每个行为至多有一条输入边和一条输出边。

(6)s 是唯一没有输入边的行为,称为起始行为。e 是唯一没有输出边的行为,称为结束行为。

(7)$t:N_G\rightarrow\{and,xor,or\}$是一个函数,为每个网关分配一个控制流类型。

(8)每个网关可以是一个分解或一个合并:分解表示只有一条输入边,至少有两条输出边;合并表示至少有两条输入边,只有一条输出边。

在这个定义中,一个行为有不超过一条输入边和一条输出边,文献[7]指出有些建模语言,如EPC,支持这种限制,但是对于其他一些语言,如BPMN,则允许行为具有多条输入输出边,但是在引入了gateways后,总可以将所有模型转化为每个行为恰好有一条输入边和一条输出边。

定义2 令 $PM=(A,G,F,t,s,e)$是一个流程模型,则定义流程模型 PM 的流程模型片段为元组 $f=(A_f,G_f,F_f,t_f)$,其中$(A_f\cup G_f,F_f)$是图$(A\cup G,F)$的连接子图,函数 t_f 是 PM 的函数 t 到集合G_f的受限约束。

一个流程模型可以用不同的方法分解成多个流程片段,只有特殊类型的流程片段分解才有意义[21]。文献[10]介绍了两种类似的流程模型分解方法:一种是单入边和单出边的片段分解(single entry/exit edge),构造流程结构树(Process Structure Tree,PST)[8];另一种是单入节点和单出节点的片段分解(single entry/exit node),构造细化的流程结构树(Refined Process Structure Tree,RPST)[9]。其给出了这两种分解方法具备的几个属性,表明分解得到的流程片段适合作为流程模型抽象的基础。我们以RPST树作为本文研究的基础,首先引入文献[10]中的几个相关概念。

定义3 令 $PM=(A,G,F,t,s,e)$是一个流程模型,PM 包含流程片段 $PMF=(A_{PMF},G_{PMF},F_{PMF},t_{PMF})$。如果 $\exists e\in in(n)\cup out(n)$,则节点 $n\in N_{PMF}$ 是PMF的边界节点。如果 n 是一个边界节点,并且 $in(n)\cap F_{PMF}=\emptyset$,则 n 为PMF的入口。如果 n 是一个边界节点,并且 $out(n)\cap F_{PMF}=\emptyset$,则 n 为PMF的出口。

定义4 令 $PM=(A,G,F,t,s,e)$是一个流程模型,PM 包含流程片段 $PMF=(A_{PMF},G_{PMF},F_{PMF},t_{PMF})$。如果 PMF 恰好有两个边界节点:一个入口节点和一个出口节点,则称 PMF 为一个部件。

令 F 是流程模型 PM 中所有部件的集合。

定义5 一个部件 $PMF=(A_{PMF},G_{PMF},F_{PMF},t_{PMF})$称为标准部件,如果对于 $\forall PMF'\in F$:

$$PMF\neq PMF'\Rightarrow(F_{PMF}\cap F_{PMF'}=\emptyset\vee(F_{PMF}\subset F_{PMF'})\vee(F_{PMF'}\subset F_{PMF}))$$

定义6 令 $PM=(A,G,F,t,s,e)$是一个流程模型,流程

模型 PM 的细化流程结构树(Refined Process Structure Tree，RPST)定义为一个树形图$RPST_{PM}=(\Omega,r,\chi)$，其中：

(1)Ω 是 PM 所有标准部件构成的集合。

(2)r 表示树的根节点对应的部件。

(3)$\chi\subseteq\Omega\times\Omega$ 是部件和其孩子部件之间的关系。

可以在流程模型节点数的线性时间内构造流程模型对应的 RPST 树[22-24]。

3 子流程发现过程

文献[13]将 RPST 树分解中的每个标准部件作为候选子流程，其中远离根节点的部件对应的流程片段与人工构造的子流程相比，往往业务含义不完整，丢失了很多应该属于本子流程中的行为。相反，处于树中较高层次的节点对应的流程片段则往往包含了某些不属于人工定义的子流程的行为。在子流程的块结构化标准基础上，我们可以通过自底向上地对每层的标准部件进行语义扩展，将标准部件与其相邻的兄弟节点之间的语义进行相似性评估并扩展，得到最接近包含完整业务含义的流程片段。

在介绍具体方法之前，我们首先引入本文算法中需要用到的几个相关技术。

3.1 提取主控元素与从属元素

文献[25]指出大多数行为标签具有规则的结构，并定义了 3 种不同的行为标签类型，本文利用文献[26]提出的标签解析技术，从不同类别的行为标签中提取出行为的动作部分(action)和行为的业务对象部分(business object)，将这两部分称为该行为的构成元素。

进一步地，我们定义流程片段的主控元素表示在给定业务流程片段中出现频率最高的主控动作或主控业务对象。文献[27]利用提取流程中的主控元素表示流程的语义，生成流程名字描述(action+business object 的形式)的候选。而根据语义的特征理论[28-30]，在业务流程模型上下文中，整个流程的语义与出现频率高的元素相关性最大[27]。因此，主控元素不仅可以用来作为整个业务流程名字描述的候选，也可以作为一个业务流程片段的语义候选。

当每个行为标签中的动作部分和业务对象部分确定后，求得主控元素的过程比较简单，文献[27]给出了求解流程模型中的主控元素的算法，本文引用并修改该算法参数为：extractDominatingElements (ProcessFragment fragment, String type)，该算法在业务流程片段 fragment 中扫描每个行为，统计构成行为的动作元素和业务对象元素出现的次数，并返回出现次数最多的元素作为主控元素，同时返回主控元素的类型 type(action 或者 business object)。

当业务流程片段中包含主控元素时，算法 extractDominatingElements 输出业务流程片段中的主控元素及类型，否则，输出空字符串。详细的算法描述见文献[27]。

将从属元素定义为业务流程片段中与主控元素共同发生的行为元素。如果主控元素是动作类型，则从属元素为业务对象类型，反之亦然。文献[27]中给出的算法 extractSubordinateElements (List activityList, String dominatingElem, String type)用于生成业务流程片段中的从属元素，该算法在业务流程片段包含的行为列表 activityList 中，找出与主控元素 dominatingElem(类型为 type)共同发生的所有从属元素。

从属元素发生次数越多，说明它与主控元素的组合越贴近对流程片段语义的描述，因此，我们进一步修改算法 extractSubordinateElements，除了输出所有从属元素外，同时输出每个从属元素出现的次数，如算法 1 所示。

算法 1 extractSubordinateElements

输入：行为列表 activityList 和主控元素 dominatingElem 及类型 type

输出：从属元素列表

```
1. List subordinateElements = new List ( String element, Integer
   count);
2. FOR all activities a in activitiyList DO
3.   IF a contains dominatingElem then
4.     IF type="Action" then
5.       element= a. getBusinessObject;
6.     ELSE
7.       element= a. getAction;
8.     k=subordinateElements. Find(element);
9.     IF k>0
10.       subordinateElements(k). count++;
11.    ELSE subordinateElements. add(element,1).
```

根据算法 1，检查给定流程片段中的行为列表中的每个行为是否包含求得的主控元素(第 3 行)，如果包含，则根据主控元素类型提取该行为中包含的从属元素 element(第 4 行到第 7 行)。在已经建立的从属元素列表 subordinateElements 中查找是否包含提取的从属元素 element(第 8 行)，若包含，则将对应元素的计数值 count 累加(第 9－10 行)，若不包含，则将 element 加入到从属元素列表中(第 11 行)。

对于属元素列表中的各个元素，利用其出现次数进行加权，即：

$$weight(subordinateElements(i))=\frac{subordinatElement(i).count}{\sum_{j=1}^{k}subordinateElement(j).count}$$

其中，k 表示获取的不同的从属元素个数，主控元素的权值为 1。最终获得两个列表：动作元素列表 actionList 和业务对象元素列表 objectList，分别存储由主控元素提取过程与从属元素提取过程求得的带权动作元素和业务对象元素。对属元素列表中的各个元素的加权过程如算法 2 所示。

算法 2 Weight

输入：主控元素 dominatingElem 及类型 type 和从属元素列表 subordinateElements

输出：带权动作元素列表和业务对象元素列表

```
1. List actionList new List(String element,Integer weight);
2. List objectList new List(String element,Integer weight);
3. countSum= Σ(j=1..k) subordinateElement(j). count;
   // k 表示获取的不同的从属元素个数
4. IF type="Action" THEN
5.   actionList. add(dominatingElem,1);
6.   FOR all elements s in subordinateElements DO
7.     objectList. add(s. element,s. count/countSum);
8. ELSE
9.   objectList. add(dominatingElem,1);
```

```
10.    FOR all elements s in subordinateElements DO
11.    actionList.add(s.element, s.count/countSum).
```

3.2 生成候选子流程

在对业务流程模型进行结构分解后，我们得到了由若干个标准 SESE 部件构成的分层 RPST 树。根据子流程的连接性标准和标签相似性标准[13]，我们通过将 RPST 树中获得的标准部件与其相邻的兄弟节点或父亲节点进行适当地合并，得到业务意义完整的流程片段作为候选子流程。其中有两个问题需要考虑：一是如何判断一个业务流程片段是否已经具备了比较完整的业务含义；二是如何选择可能的行为集合，与标准部件进行合并生成业务含义完整的流程片段。

RPST 树中每一个标准部件都是子流程的候选，如果某个标准部件满足业务含义的完整性，则表示可以直接构成子流程，这个过程可以通过人工判断实现。如果自动实现流程片段语义完整性的判别工作，则需要引入更加详细的流程上下文，比如事件、操作的数据、角色等，除了利用前面提到的语义的特征理论生成流程片段的语义候选外，还可以引入语义原型理论[31]、语义的实用主义理论[32]等，在业务流程其它上下文中提取多角度语义描述，并进行相似性比较。如何在 RPST 树中引入其他的流程上下文，并根据比较结果判别业务流程片段是否可以作为候选子流程超出了本文研究的范围，将作为我们下一步工作的内容。

本文重点探讨第二个问题，通过利用主控元素和从属元素提取，从与标准部件相邻的兄弟节点中，选择最可能与其构成子流程的行为集合进行合并。这是一种贪心选择，在与待扩展标准部件相邻的兄弟节点对应的所有可能行为子集中分别求得主控元素与从属元素，生成其对应的动作元素列表和业务对象元素列表，然后与该标准部件生成的动作元素列表和业务对象元素列表进行相似性比较，得到相似性值最大的行为集合。该行为集合最可能与待扩展的标准部件语义相关。再利用最初根据经验设定的相似性阈值判断是否需要将标准部件与求得的行为集合进行聚合，以得到新的流程片段，这种方法同时考虑了子流程的连接性标准和标签相似性标准。

相似性阈值可以利用经验值获得，通过人工设计的包含子流程的业务流程模型中，两个不同子流程之间的主控元素和从属元素的相似性值来确定。为了提高精确性，可以取多个不同的子流程对，将求得的相似性值平均值作为相似性阈值。由于我们只是利用不同子流程之间的标签相似性值，而不涉及到具体的域语义相关性，因此在一个语义域上求得的相似性阈值对另一个语义域的流程模型仍然适用。

当我们求得与标准部件待聚合的行为集合后，若两者的相似性值小于或等于相似性阈值时，说明标准部件与待聚合行为集合之间距离较远，属于同一子流程的可能性较小，因此不予聚合。

我们根据语义的特征理论，通过提取流程片段中的主控元素和从属元素来生成流程片段的语义描述候选，因此两个流程片段的相似性可以通过计算得到的各自语义描述候选之间的相似性得到。

将基于字符串编辑距离的语法相似性方法[33]作为测量两个标签相似性的基本方法，根据文献[33]，字符串 s 和 t 的编辑距离表示为 $ed(s,t)$，是将 s 转换为 t（或者将 t 转换成 s）的原子字符串操作的最小步骤数。原子字符串操作是：插入一个字符、删除一个字符或者将一个字符替换为另一个字符。

虽然这是一种相对粗糙的测量方法，没有考虑词汇之间的复杂关系，如同义词，但是正如文献[18]所指出的，工作于同一个业务流程模型的人员通常使用一致的术语而并不使用同义词。另外，子流程更可能针对同一个业务对象进行操作，因此在行为标签中更可能包含该业务对象。

通过行为集合与标准部件各自的加权动作元素列表和业务对象元素列表求得二者的相似性。设 $sese$ 为 RPST 树中的一个标准部件，$actionList_{sese}$ 和 $objectList_{sese}$ 分别表示主控元素和从属元素提取过程及加权过程作用于 $sese$ 后获得的动作和业务对象的加权列表，其中，$actionList_{sese}=\{(action_{sese_i},w_{sese_i})\mid action_{sese_i}$ 表示第 i 个动作，w_{sese_i} 表示第 i 个动作的权值，$1\leqslant i\leqslant k\}$，$objectList_{sese}=\{(object_{sese_i},w'_{sese_i})\mid object_{sese_i}$ 表示第 i 个对象，w'_{sese_i} 表示第 i 个对象的权值，$1\leqslant i\leqslant l\}$。

设 F 是与 $sese$ 合并的行为的候选集合，$actionList_F$ 和 $objectList_F$ 分别表示主控元素和从属元素提取过程及加权过程作用于 F 后获得的动作和业务对象的加权列表，其中，$actionList_F=\{(action_{F_i},w_{F_i})\mid action_{F_i}$ 表示第 i 个动作，w_{Fi} 表示第 i 个动作的权值，$1\leqslant i\leqslant q\}$，$objectList_F=\{(object_{F_i},w'_{F_i})\mid object_{F_i}$ 表示第 i 个对象，w'_{F_i} 表示第 i 个对象的权值，$1\leqslant i\leqslant r\}$。

$sese$ 与 F 的相似性如式(1)所示：

$$SIM_{sese\text{-}F}=\max\{sim(action_{sese_i},action_{F_j})\mid 1\leqslant i\leqslant k,1\leqslant j\leqslant q\}+\max\{sim(object_{sese_i},object_{F_j})\mid 1\leqslant i\leqslant l,1\leqslant j\leqslant r\} \tag{1}$$

其中，

$$sim(action_{sese_i},action_{F_j})=ed(action_{sese_i},action_{F_j})\times\frac{w_{sese_i}+w_{F_j}}{\sum_{t=1}^{k}w_{sese_t}+\sum_{t=1}^{q}w_{F_t}}$$

$$sim(object_{sese_i},object_{F_j})=ed(object_{sese_i},object_{F_j})\times\frac{w'_{sese_i}+w'_{F_j}}{\sum_{t=1}^{l}w'_{sese_t}+\sum_{t=1}^{r}w'_{F_t}}$$

在 RPST 树中，父亲节点与儿子节点之间是真包含关系，兄弟节点之间前驱与后继的关系，假设 RPST 树中兄弟节点按照业务流程模型中的执行顺序从左至右依次分布，即假设节点 $sese_1,\cdots,sese_k$ 是节点 $sese$ 的 k 个儿子，且在业务流程模型中的执行顺序为 $sese_i$ 先于 $sese_j$（$1\leqslant i<j\leqslant k$），则 $sese_1,\cdots,sese_k$ 在 RPST 树中的分布如图 1 所示。

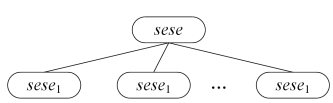

图 1 RPST 中节点分布

Fig. 1 Nodes distribution in RPST

由标准部件的定义可知，流程模型中的每个单个行为是一个最小的标准部件，因此，在 RPST 树中，叶子节点为单个行为，设事先获得的相似性阈值为 ω，RPST 树的层数为 h（根节点为第 0 层）。

自底向上对 RPST 树中每一层的标准部件进行操作，生成一棵新的候选子流程树，过程如算法 3 所示，其输入为业务流程模型对应的 RPST 树，业务流程模型中包含的行为经过

了标签解析技术预处理。

算法 3 reconfigureRPST(Tree rPST)

输入：业务流程模型对应的 RPST 树 rPST

输出：重构并标识了子流程的结构树

1. i=h−1；
2. 当 i≥1 时，执行
3. 从左至右扫描 RPST 树中第 i 层的节点 $sese_k$($1\leqslant k\leqslant m_i$，$m_i$ 表示第 i 层节点数)
4. 如果 $|sese_k|>1$，则
5. 调用算法 extractDominatingElements 提取 $sese_k$ 的主控元素 $dominatingElem_{sese_k}$ 及类型 type；
6. 调用算法 extractSubordinateElements 生成从属元素列表 $subordinateElements_{sese_k}$；
7. 调用算法 Weight 生成动作元素列表 $actionList_{sese_k}$ 和业务对象元素列表 $objectList_{sese_k}$；
8. lmaxsim=0；rmaxsim=0；
9. FOR j=k−1 TO 1 //$sese_l$ 是 $sese_k$ 左侧最后一个未标识为 newAggregation 的兄弟节点；
10. 调用算法 extractDominatingElements 提取流程片段 $F_j=\{sese_j,\cdots,sese_{k-1}\}$ 的主控元素 $dominatingElem_j$ 及类型 $type_j$；
11. 调用算法 extractSubordinateElements 生成从属元素列表 $subordinateElements_j$；
12. 调用算法 Weight 生成动作元素列表 $actionList_j$ 和业务对象元素列表 $objectList_j$；
13. 利用式(1)计算 $SIM_{sese_k\text{-}F_j}$；
14. 如果 $lmaxsim<SIM_{sese_k\text{-}F_j}$ 并且 $SIM_{sese_k\text{-}F_j}>\omega$
15. $lmaxSIM=SIM_{sese_k\text{-}F_j}$；
16. laggLOC=j；
17. FOR j=k+1 TO r //$sese_r$ 是 $sese_k$ 右侧最后一个未标识为 newAggregation 的兄弟节点；
18. 调用算法 extractDominatingElements 提取流程片段 $F_j=\{sese_{k+1},\cdots,sese_j\}$ 的主控元素 $dominatingElem_j$ 及类型 $type_j$；
19. 调用算法 extractSubordinateElements 生成从属元素列表 $subordinateElements_j$；
20. 调用算法 Weight 生成动作元素列表 $actionList_j$ 和业务对象元素列表 $objectList_j$；
21. 利用式(1)计算 $SIM_{sese_k\text{-}F_j}$；
22. 如果 $rmaxsim<SIM_{sese_k\text{-}F_j}$ 并且 $SIM_{sese_k\text{-}F_j}>\omega$
23. $rmaxSIM=SIM_{sese_k\text{-}F_j}$；
24. raggLOC=j；
25. 如果 k−laggLoc>0 或者 raggLOC−k>0，则
26. 如果 $sese_{laggLOC}$ 和 $sese_{raggLOC}$ 分别是 $sese_k$ 的最左边和最右边的兄弟节点，则标识 $sese_k$ 的父亲节点为 candidate-subprocess；
27. 否则，为 $sese_k$ 的父亲节点添加新的儿子节点 n，并标识为 candidate-subprocess 和 newAggregation，同时将 $sese_{laggLOC}$ 到 $sese_{raggLOC}$ 之间的所有节点作为 n 的儿子节点，并删除 $sese_{laggLOC}$ 到 $sese_{raggLOC}$ 之间的所有节点与其父亲节点之间的关系；
28. 否则，标识 $sese_k$ 为 candidate-subprocess.

算法 3 从 RPST 树的第 $h-1$ 层开始，自底向上、从左至右对每层中包含行为数大于 1 的 SESE 部件 $sese_k$ 进行处理(第 1−4 行)。首先提取标准部件 $sese_k$ 对应的流程片段的候选业务含义描述，利用前面给出的算法 extractDominatingElements，extractSubordinateElements，Weight 求出由主控元素和从属元素构成的动作元素列表 $actionList_{sese_k}$ 和业务对象元素列表 $objectList_{sese_k}$(第 5−7 行)。然后以 $sese_k$ 为中心，依次向左和向右找出最有可能与 $sese_k$ 聚合生成候选子流程的标准部件集合(第 8−24 行)，具体过程可以用图 2 直观表示。

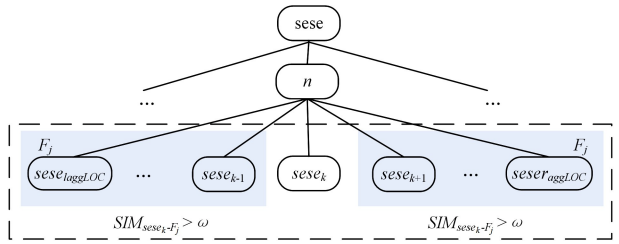

图 2 与 $sese_k$ 聚合生成候选子流程过程

Fig. 2 Generating candidate subprocesses integrating with $sese_k$

如果存在与 $sese_k$ 聚合的节点集合并且该集合与 $sese_k$ 共同构成了其父亲节点 *sese* 的所有儿子，则说明 *sese* 可以作为一个候选子流程，此时将 *sese* 标识为 candidate-subprocess(第 25−26 行)。如果求得的与 $sese_k$ 聚合的节点集合与 $sese_k$ 是其父节点 *sese* 的部分儿子节点，则将 $sese_k$ 与该集合聚合为新的节点 n，并同时将 n 标识为 candidate-subprocess 和 newAggregation(第 27 行)。如果不存在与 $sese_k$ 聚合的节点集合，则 $sese_k$ 本身标识为 candidate-subprocess。

4 实验与分析

本文的研究以作者所在的物流产业经济与智能物流重点实验室为平台，研究内容是与某大型汽车企业及某国际物流公司合作的实验室开放项目的一部分，利用真实的流程模型进行方法评估。我们从该汽车企业的多条生产线和物流公司物流管理线上选取 6 个流程模型(文献[1]、文献[11]和文献[13]中仅选取 1 个复杂流程模型)，其中 3 个模型由人工构建的子流程构成(部分子流程重用)，子流程包含生产子流程和物流子流程，其对应的细节展开模型中包含的节点数均在 100 以上，对于重用的子流程重复展开。另外 3 个流程选取生产管理流程和物流管理流程，不包含人工设计的子流程，节点数均在 80 到 95 之间。

实验分为两部分，第一部分将 3 个包含人工构建的子流程的流程模型(M_1，M_2，M_3)转换为对应的细节展开模型，分别用块结构化标准和本文提出的语义与结构结合的方法自动生成候选子流程，然后对比生成的流程片段与人工构建的子流程之间的接近度；第二部分对剩余的 3 个不包含子流程的细节模型(M_4，M_5，M_6)通过语义与结构结合的方法自动生成候选子流程，并将生成结果分发给相关部门的工作人员，对各个候选子流程的合理性进行人工评估。

我们引用文献[13]中定义的部分相关指标来比较人工定义的子流程分解与自动生成的子流程分解之间的各种特征，具体说明如下。

Precision：自动生成的并且也是人工定义的子流程数除以自动生成的子流程数。

Recall：自动生成的并且也是人工定义的子流程数除以人工定义的子流程数。

Overshoot：自动生成的候选子流程的节点中，不属于与

该子流程匹配的人工定义的子流程的节点比例。

Undershoot：应该属于某个人工定义的子流程，但是在自动生成的候选子流程中没有生成的节点比例。

在比较自动生成的子流程与人工定义的子流程是否匹配时，采用节点匹配法，而不是流程的整体匹配法。各个测量指标的计算定义如下，详见文献[13]。

令 N 是一个流程中的所有节点集合(包括其子流程)，$P_M \subseteq$ PN 是人工确定的子流程集合，$P_A \subseteq$ PN 是自动确定的候选子流程集合。$P_M \in P_M$ 是一个人工确定的子流程，$P_A \in P_A$ 是一个自动确定的候选子流程。P_A 与 P_M 之间的覆盖 overlap 定义为[13]：

$$Overlap=\frac{|P_A \bigcap P_M|}{\max(|P_A|,|P_M|)}$$

如果 P_A 与 P_M 之间的覆盖 $overlap>0$，并且不存在其他自动生成的子流程 $P_A' \in P_A$ 与 P_M 之间的覆盖 $overlap'>overlap$，则称 P_A 是 P_M 的一个最相关匹配。令函数 $match: P_M \rightarrow$ PN 返回每个人工定义的子流程的最相关匹配，如果不存在，则返回空集。

Precision 和 Recall 定义如下：

$$Precision=\frac{\sum_{P_M \in P_M} |P_M \bigcap match(P_M)|}{\sum_{P_A \in P_A} |P_A|}$$

$$Recall=\frac{\sum_{P_M \in P_M} |P_M \bigcap match(P_M)|}{\sum_{P_M \in P_M} |P_M|}$$

F 值定义为 $Precision$ 和 $Rrecall$ 两个值的调和平均数：

$$F=\frac{2 \cdot Precision \cdot Recall}{Precision+Recall}$$

Overshoot 和 Undershoot 分别定义如下：

$$Overshoot=\frac{\sum_{P_M \in P_M} |match(P_M)-P_M|}{\sum_{P_A \in P_A} |P_A|}$$

$$Undershoot=\frac{\sum_{P_M \in P_M} |P_M-match(P_M)|}{\sum_{P_M \in P_M} |P_M|}$$

表 1 第一部分实验评估结果

Table 1 Evaluation results of the first part of the experiment

指标	块结构化			语义与结构结合			原模型		
	M_1	M_2	M_3	M_1	M_2	M_3	M_1	M_2	M_3
子流程总数	71	77	68	32	31	35	18	20	16
子流程的节点数平均值	5.31	6.11	5.72	5.2	4.89	7.1	9.23	8.4	9.11
子流程节点数最大值	113	103	109	55	71	57	34	27	30
子流程节点数最小值	1	1	1	2	2	3	3	2	3
Precision	0.15	0.12	0.16	0.45	0.59	0.31	—	—	—
Recall	0.33	0.34	0.44	0.45	0.53	0.53	—	—	—
F 值	0.2	0.18	0.24	0.45	0.56	0.39	—	—	—
Overshoot	0.23	0.19	0.21	0.25	0.34	0.3	—	—	—
Undershoot	0.44	0.45	0.4	0.19	0.2	0.16	—	—	—

根据文献[13]，F 值是一个很重要的指标，它给出了自动子流程分解与人工设计的子流程分解之间的接近程度。根据表 1 的结果可以看出，结合了语义的抽象过程比直接基于结构的划分方法要更加接近于人工划分结果，而文献[13]中仅针对一个流程的实验结果中，连接性标准、标签相似性标准以及两者结合所获得的 F 值最大为 0.35。

基于结构的划分方法与结合语义的方法都有较高的 *Overshoot* 值和 *Undershoot* 值，但是结合语义的方法得到的 *Undershoot* 值(3 个模型平均为 0.18)明显低于基于结构的方法得到的 *Undershoot* 值(3 个模型平均值为 0.43)，说明基于结构的方法划分的结果子流程中丢失了 43% 本应属于某个人工定义的子流程的节点。而通过结合语义，发现了一部分本应属于某个子流程中的节点，因此将丢失节点数降低为 18%。但是结合语义的方法得到的 *Overshoot* 值甚至高于基于结构的方法，说明其过量地发现了某些不应该属于某个子流程的节点。

基于结构的划分方法与结合语义的方法生成的子流程包含的平均节点数都高于人工构建的子流程平均节点数，由于基于结构的方法将 RPST 树中所有的标准部件都作为候选子流程，因此节点数最少的子流程就是包含一个行为的流程片段，节点数为 1，而节点数最大的子流程是根节点对应的子流程，也就是原模型，节点数为原模型的总节点数，这两种情况都没有任何实际意义。而结合语义的方法虽然生成子流程的平均节点数高于人工构建的子流程平均节点数，但是实际在 3 个模型使用结合语义方法自动生成的子流程中，都有 50% 以上的子流程，其丢失节点(或冗余节点)个数低于与其匹配的人工构建的子流程节点个数(或该子流程的节点数)的 10%，说明通过对 RPST 树的自底向上的聚合和标识，排除了树中很多没有意义的候选子流程。

第二部分实验我们将原始细节模型和由自动生成候选子流程构成的结果模型分发给 20 个参与项目合作的工作人员，其中 2 个物流流程模型分发给相同的 10 个相关工作人员，1 个生产流程分给另外不同的 10 个相关工作人员。对每个生成的候选子流程做一些人工的分析和评估，即根据经验，每个工作人员独立在自动生成的候选子流程基础上进行节点扩充或删除，将候选子流程完善成为子流程。完善过程只针对每个生成的候选子流程，而没有生成最终的流程模型。

按照第一部分实验方法，对 3 个模型可以得到各个指标的 10 组不同的值，将 10 组值取平均后得到各个指标的实验评估结果。

为了简化，对于每个候选子流程 P_A，我们只统计工作人员人工添加节点的个数 n_{add}(表示本应属于子流程但是未被自动生成的节点)和删除节点的个数 n_{del}(表示自动生成但不属于该子流程的节点)，则 $|P_A|+n_{add}-n_{del}$ 表示人工修正后得到的子流程节点数，当 $|P_A|+n_{add}-n_{del}=0$ 时表示该流程片段 P_A 不构成有意义的子流程，即不匹配任何人工设计的子流程。

根据统计的数据得到各个指标值的计算公式如下：

$$Overshoot^*=\left(\sum_{person \in Group} \frac{\sum_{P_A \in P_A} n_{del}}{\sum_{P_A \in P_A} |P_A|}\right)/|Group|$$

$$Undershoot^*=\left(\sum_{person \in Group} \frac{\sum_{P_A \in P_A} n_{add}}{\sum_{P_A \in P_A} (|P_A|+n_{add}-n_{del})}\right)/|Group|$$

$$Precision^*=\left(\sum_{person \in Group} \frac{\sum_{P_A \in P_A} (|P_A|-n_{del})}{\sum_{P_A \in P_A} |P_A|}\right)/|Group|$$

$$Recall^*=\left(\sum_{person \in Group} \frac{\sum_{P_A \in P_A} (|P_A|-n_{del})}{\sum_{P_A \in P_A} (|P_A|+n_{add}-n_{del})}\right)/|Group|$$

$$F^*=\left(\sum_{person \in Group} \frac{2 \cdot precision^* \cdot recall^*}{precision^*+recall^*}\right)/|Group|$$

实验结果如表 2 所列。

表 2 第二部分实验评估结果

Table 2 Evaluation results of the second part of the experiment

指标	M_4	M_5	M_6
子流程总数	21	27	22
子流程的节点数平均值	9.7	5.33	4.92
子流程节点数最大值	47	54	39
子流程节点数最小值	2	2	2
*Precision**	0.76	0.51	0.31
*Recall**	0.76	0.53	0.44
F^* 值	0.76	0.52	0.36
*Overshoot**	0.24	0.34	0.17
*Undershoot**	0.24	0.22	0.35

对于生产流程模型 M_4，实验结果表明生成的子流程更接近于人工定义的子流程，查看最后人工扩展后的子流程，包含的大多数行为标签都涉及同一个操作对象（主控对象），因此在算法自动选择与 SESE 标准部件进行聚合的行为集合时，会发现比较全面的属于该子流程中的相似行为。

在第二部分实验结果子流程中，由于工作人员单独对每个子流程进行节点添加或节点删除，而不考虑整体流程的连接性，所以对于修正后的子流程，我们发现很多子流程在添加节点过程中，重用了相同的行为，甚至相同的其他子流程。

结束语 本文基于业务流程模型的基于结构的分解获得的 RPST 树，提取标准部件对应的流程片段的候选语义描述，结合子流程的标签相似性标准，发现与标准部件语义描述最相似的行为集合作为待聚合行为集合，并从已经存在的不同的子流程之间的语义相似性确定相似性阈值对该行为集合进行限制，最终确定与标准部件聚合的行为并与之聚合生成新的流程片段作为候选子流程。我们通过实际的流程案例对基于结构的抽象方法（将 RPST 树中每个标准部件作为候选子流程）和提出的结合语义的抽象方法做实验对比分析，从中得到一些结论和未来工作的方向。

由于提出的方法基于 RPST 树，利用语义信息对各个标准部件进行扩展和标识，使得最后得到的候选子流程数目大大缩小，并且从实验结果中可以看出，其与人工构建的子流程更加接近。但是仍然存在很多应该属于子流程而未被算法发现，或者不应该属于子流程而被算法发现的行为，前者说明仅仅依赖于行为的标签信息获得行为的语义还远远不够，如何通过引入其他上下文信息，比如操作数据、执行角色、组织机构等，得到更精确的流程片段的语义描述候选将是我们下一步工作研究的方向之一；对于后者，说明对于某个流程片段，算法生成了多余的行为与之聚合，可以从文献[30]提出的几种流程模型命名的方法中得到启发，从语义角度判断聚合后的新的流程片段和原有的流程片段哪一个更符合子流程的标准，这也是我们下一步工作研究的方向之一。另外，在第二部分实验中，人工修正后的流程片段之间出现很多重复的行为甚至其他流程片段，这也是基于结构的流程模型抽象方法存在的问题，即初始模型中的每个行为只是属于一个抽象行为，因此，在接下来的改进算法中，我们将重点研究行为或流程片段重用问题。

参 考 文 献

[1] SMIRNOV S, DIJKMAN R, MENDLING J, et al.. Meronymy-based aggregation of activities in business process models[C]// Conceptual Modeling - ER 2010, Lecture Notes in Computer Science, 2010: 1-14.

[2] SMIRNOV S, REIJERS H A, WESKE M H, et al. Business process model abstraction: a definition, catalog, and survey[J]. Distributed and Parallel Databases, 2012, 30(1): 63-99.

[3] BOBRIK R, REICHERT M, BAUER T. View-Based Process Visualization[C]// Proceedings of the 5th International Conference on Business Process Management (BPM 2007). Brisbane, Australia, 2007: 88-95.

[4] LIU D, SHEN M. Workflow Modeling for Virtual Processes: an Order-preserving Process-view Approach[J]. IS, 2003, 28(6): 505-532.

[5] POLYVYANYY A, SMIRNOV S, WESKE M. The Triconnected Abstraction of Process Models[J]. Business Process Management, Lecture Notes in Computer Science, 2009, 5701: 229-244.

[6] POLYVYANYY A, SMIRNOV S, WESKE M. Reducing Complexity of Large EPCs[C]// MobIS. 2008: 195-207.

[7] POLYVYANYY A, SMIRNOV S, WESKE M. On Application of Structural Decomposition for Process Model Abstraction [C]// Proceedings of the BPSC. 2009: 110-122.

[8] VANHATALO J, VÖLZER H, LEYMANN F. Faster and More Focused Control-Flow Analysis for Business Process Models Through SESE Decomposition[C]// Proceedings of the Fifth International Conference of Service-Oriented Computing(ICSOC 2007). Vienna, Austria, 2007: 43-55.

[9] VANHATALO J, VÖLZER H, KOEHLER J. The Refined Process Structure Tree[C]// Proceedings of the 6th International Conference on Business Process Management (BPM 2008). Milan, Italy, 2008: 100-115.

[10] SMIRNOV S. Business Process Model Abstraction[D]. Germany: University of Potsdam. 2021.

[11] SMIRNOV S, REIJERS H A, WESKE M. A Semantic Approach for Business Process Model Abstraction[C]// Proceedings o the CAiSE 2011. Springer, 2011: 497-511.

[12] WEIDLICH M, DIJKMAN R, MENDLING J. The ICoP framework-Identification of correspondences between process models, Advanced Information Systems Engineering[C]// Lecture Notes in Computer Science. 2010: 483-498.

[13] REIJERS H A, MENDLING J, DIJKMAN R M. On the Usefulness of Subprocesses in Business Process Models[C]// BPM Center Report BPM-10-03, BPMcenter. org, 2010.

[14] SCHAEFFER S. Graph clustering-survey[J]. Computer Science Review, 2007(1): 27-64.

[15] SUN S W, WANG N, OUYANG D T. Business Process Model Abstraction Based on Cluster Analysis[J]. Computer Science, 2016, 43(5): 193-197, 229.

[16] WANG N, SUN S W, OUYANG D T. Business Process Modeling Abstraction Based on Semi-Supervised Clustering Analysis [J]. Business & Information Systems Engineering, 2018, 60(6): 525-542.

[17] WANG N, SUN S, LIU Y, et al. Business Process Model Abstraction Based on Fuzzy Clustering Analysis[J]. International Journal of Cooperative Information Systems, 2019, 28(9).

（下转第 687 页）

结束语　思维图技术主要面向复杂算法研发过程，应对内部逻辑、功能和流程不能一次确定，要根据情况不断进行修改和更新的情况。应用思维图技术可以迅速定位问题，索引原有逻辑和功能，查询原有版本输入/输出格式和接口，并存档内部逻辑和数据格式的变动信息。在文本数据元语义识别算法研发过程中，采用思维图的方法编制算法研发文档、简便易行、操作实用、方便逻辑功能调试，提高了研发工作效率，对于其他复杂算法研发工作有借鉴意义。下一步，将继续探索思维图表述的形式化方法，依靠思维图的继承和多态，实现逻辑模块代码的复用；探索思维图查询、更新和备份机制及其软件实现，使思维图成为复杂算法设计和维护的实用工具。

参 考 文 献

[1] XU X Z. The research and implementation of technology in MDA[J]. Computer Science, 2006, 133(112): 278-279.

[2] CHEN Y H. The study of construction and validation of Object-oriented formal specification[D]. Shanghai: Shanghai University Doctoral Dissertation, 2007.

[3] CHEN R S. Software development of single chip computer based on flow chart programming[J]. Intelligent Computer and Applications, 2013, 3(4): 95-97.

[4] WANG L M, WANG G N, ZHOU M Y, et al. Research on and implementation of algorithm from the program flowchart to the code[J]. Journal of Xidian University(Natural Science Edition), 2012, 39(6): 70-77.

[5] ZHONG Z C, ZHANG Z S, DAI M, et al. Real-time conversion method for flowchart and similar C language[J]. Journal of Southeast University(Natural Science Edition), 2009, 39(3): 502-507.

ZHU Ping, born in 1970, Ph.D, professor. His main research interests include thinking machine design, text semantic recognition and software engineering.

(上接第 665 页)

[18] JURIC M B, MATHEW B, SARANG P. Business Process Execution Language for Web Services[OL]. http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=C3280AF5C27D29D5A33E10341A664E79?doi=10.1.1.201.4876&rep=rep1&type=pdf.

[19] VAN DER AALST W. Verification of workflow nets[C]// Application and Theory of Petri Nets. Springer, 1997: 407-426.

[20] VAN DER MP W. Formalization and verification of event-driven process chains[J]. Information & Software Technology, 1999, 41(10): 639-650.

[21] HAUSER R, FRIESS M, KUSTER J M, et al. Combining Analysis of Unstructured Workflows with Transformation to Structured Workflows[C]// Proceedings of the EDOC 2006. IEEE Computer Society, 2006: 129-140.

[22] FUSSELL D S, RAMACHANDRAN V, THURIMELLA R. Finding Triconnected Components by Local Replacement[J]. SIAM Journal on Computing, 1993, 22(3): 587-616.

[23] GUTWENGER C, MUTZEL P. A Linear Time Implementation of SPQR-Trees[C]// Proceedings of the Graph Drawing 2001. Springer, 2001: 77-90.

[24] POLYVYANYY A, VANHATALO J, VOLZER H. Simpli_ed Computation and Generalization of the Refined Process Structure Tree[C]// Proceedings of the WS-FM 2010. Springer, 2011: 25-41.

[25] MENDLING J, REIJERS H A, RECKER J. Activity Labeling in Process Modeling: Empirical Insights and Recommendations[J]. Information Systems, 2010, 35(4): 467-482.

[26] LEOPOLD H, SMIRNOV S, MENDLING J. On the refactoring of activity labels in business process models[J]. Information Systems, 2012, 37(5): 443-459.

[27] LEOPOLD H, MENDLING J, REIJERS H A, et al. Simplifying process model abstraction: techniques for generating model names[J]. Information Systems, 2013, 39: 134-151.

[28] FREGE G. On sense and reference[J]. Philosophical Review, 1948, 1892(57): 209-230.

[29] RUSSELL B. On denoting[J]. Mind, 1905(56): 479-493.

[30] DANTO A C, RUSSELL B, MARSH R C. Logic and Knowledge: Essays, 1901-1950[J]. The Journal of Philosophy, 1956, 54(11): 360.

[31] ROSCH E. Cognitive representations of semantic categories[J]. Journal of Experimental Psychology: General, 1975, 104(3): 192-233.

[32] WITTGENSTEIN L. Philosophical Investigations, The German Text with an English Translation (4th Edition)[M]. Basil Blackwell, Oxford, 2009.

[33] VAN DONGEN B, DIJKMAN R, MENDLING J. Measuring similarity between business process models[C]// Proceedings of the 20th International Conference on Advanced Information Systems Engineering (CAiSE). 2008: 450-464.

SUN Shan-wu, born in 1969, associate professor. His main research interests include machine learning business process modeling, and network security.

WANG Nan, born in 1980, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include machine learning, public opinion analysis and BPMA.