文章编号:1001 - 9081(2011)08 - 02253 - 05
doi:10.3724/SP.J.1087.2011.02253

工作流集成中基于代数表达式的语义匹配机制

齐　军,张月菊,王　涛
(华南师范大学 计算机学院,广州 510631)
(qi_jun2005@126.com)

摘　要:针对现阶段工作流集成研究中功能匹配查准率和查全率低的问题,给出了基于软件功能形式化语义的匹配机制的实现。在前、后条件 pre/post 的完全匹配模式下,以高级程序设计语言中的代数表达式为基础,提出了匹配原则,并给出了具体的算法,并且用实例进行分析说明。该算法适用于工作流集成中的功能匹配,同时基于严格的形式化方法,便于分析和验证。该算法局限于初等代数性的前提。

关键词:工作流集成;形式语义;语义匹配;代数表达式;功能匹配
中图分类号: TP311.521　　**文献标志码**:A

Semantic matching mechanism based on algebraic expression in workflow integration

QI Jun, ZHANG Yue-ju, WANG Tao
(*School of Computer Science, South China Normal University, Guangzhou Guangdong* 510631, *China*)

Abstract: Concerning the problems of low precision ratio and low recall ratio of function match in the research of workflow integration, the authors implemented the matching mechanism based on formal semantic of extract pre/post match pattern, and proposed matching principles on the basis of algebraic expressions in high level programming languages. The specific algorithm was raised up and also an example was given to analyze and illustrate the algorithm. The proposed algorithm is suitable for function matching in workflow integration and it is founded on strict formal method, so that it can be analyzed and verified conveniently with mathematical methods. The limitation is that it is based on elementary algebraic expression.

Key words: workflow integration; formal semantic; semantic match; algebraic expression; function match

0　引言

跨组织工作流之间的集成技术是工作流技术的一个重要研究领域,近十年来 Web 服务发现技术应用于工作流集成的研究,极大地促进了工作流集成技术的发展。但仍存在着以下问题。

1)功能查找不全、匹配率较低。对于工作流集成中的功能集成,计算机如何自动发现功能是否匹配,是实现功能自动集成的一个关键技术。目前,Web 服务发现技术中的语法级服务发现技术的查准率和查全率不高。

2)可行性不高。服务发现技术中的语义级服务描述基于本体,使用逻辑演绎和推理进行匹配,虽然提高了查准率和查全率,但仍然存在一些不足[1]:缺乏灵活有效的服务匹配算法;服务发现效率低下等。

因此,本文将研究的重点放在功能匹配上,对以初等代数表达式为基础的功能以集合论和一阶谓词逻辑为基础进行形式化描述,关注其前条件和后条件,并对其进行数学推理,以判定功能是否匹配。

1　国内外研究现状

关于工作流集成,众多学者从 5 个方向分别进行了研究,分别是:工作流模型方向[2]、资源方向[3-4]、数据/信息方向[5]、任务/功能方向以及作业/应用方向[1]。本文研究重点在任务/功能方向,探讨工作流功能集成中的功能匹配机制。

关于语义匹配,学术界从 Web 服务发现领域[6]、信息检索和知识发现领域[7-8]、工作流模型互操作[9]等多个领域展开了研究。

目前,学术界已经存在基于各种技术的语义匹配研究,主要包括以下两种:基于本体的语义匹配和基于 VDM、B、Z、Object-Z 等形式化方法的语义匹配。其中基于本体的语义匹配方法又可分为以下四类:基于模式的匹配方法、基于实例的匹配方法、基于语言学的匹配方法和基于约束的匹配方法[10-11]。形式化方法能使软件开发人员创建比使用传统开发方法产生的需求规格说明更完整无二义性的规格说明,能帮助软件开发人员更快速准确地发现系统描述中存在的不一致性、不明确性或不完整性。在一些软构件匹配的研究中,给出了形式化语义匹配的雏形,基于形式化规格说明的前、后条件匹配,并给出了多种匹配概念:完全匹配(Extract Plug-In Match)、可替代匹配(Plug-In Match)、弱匹配(Weak Postmatch)等[11],并未见具体实现。

2　语义匹配机制

研究基于软件功能的形式化语义,即形式化规格描述 V,其前条件和后条件分别为 V_{pre} 和 V_{post},操作为 P。当操作 P 执行之前 V_{pre} 成立,执行之后 V_{post} 成立。基于软件功能的形式化语义匹配,即形式化规格描述 V 和 M 之间的前、后条件满足一定匹配算法的匹配,V 和 M 的匹配关系可以描述为如下形式(其中关系 R_1、R_2 可以是等价关系($\Leftrightarrow$)或者蕴涵关系($\Rightarrow$),R_3 是合取词($\wedge$)或者蕴涵词($\Rightarrow$))[11-12]:

$$match(V,M) = (V_{pre} R_1 M_{pre}) R_3 (V_{post} R_2 M_{post})$$

收稿日期:2011 - 02 - 14;**修回日期**:2011 - 03 - 16。

作者简介:齐军(1987 -),男,河南息县人,硕士研究生,主要研究方向:软件工程、形式化方法;　张月菊(1980 -),女,河北晋州人,硕士,主要研究方向:软件工程、工作流;　王涛(1975 -),男,广西合浦人,副教授,博士,主要研究方向:软件工程、形式化方法、信息系统安全。

上面的描述中,已经给出了形式化语义匹配的基本思路,由于 R_1 和 R_2 可以是等价关系也可以是蕴涵关系。由前、后条件组成的匹配便有多种匹配关系。下面给出几种匹配模式[12-13]:

1)前、后条件 pre/post 的完全匹配(Extract pre/post Match)。当 R_1 和 R_2 均为等价关系(⇔)且 R_3 为合取词(∧)时,即形式化规格说明 V 和 M 的前条件和后条件严格匹配,此时 V 和 M 完全一致。形如:

$$match(V,M) = (V_{pre} \Leftrightarrow M_{pre}) \wedge (V_{post} \Leftrightarrow M_{post})$$

2)前、后条件 pre/post 的可替代匹配(Plug-In Match)。形式化规格 V 同形式化规格 M 相比,其前条件较弱,后条件较强,即形式化规格 M 可替代 V 的功能要求,匹配公式如下:

$$match(V,M) = (M_{pre} \Rightarrow V_{pre}) \wedge (V_{post} \Rightarrow M_{post})$$

3)前、后条件 pre/post 的后条件可替代匹配(Plug-In Postmatch)。不考虑前条件 Pre,只考虑 M 的后条件是否与 V 的后条件相符合。弱匹配公式如下:

$$match(V,M) = (V_{post} \Rightarrow M_{post})$$

4)前、后条件 pre/post 的弱匹配(Guarded Post match/Weak Post Match)。形如 $match(V,M) = V_{pre} \Rightarrow (V_{post} \Rightarrow M_{post})$ 或者 $match(V,M) = (V_{pre} \wedge V_{post}) \Rightarrow M_{post}$。

3 基于代数表达式的功能匹配

前面讨论的4种匹配模式中,本文只讨论前、后条件 pre/post 的完全匹配。高级程序设计语言所描述的程序中,功能从狭义上讲就是函数,而一个实现了某个功能的函数往往包含了大量的代数表达式。对于此类功能,首先研究基于代数表达式的功能匹配,提出相应的匹配原则及其算法。

3.1 设计原则

图1展示了在前、后条件 pre/post 完全匹配模式下,验证功能间的前、后条件匹配的流程。

1)首先检查输入功能的参数类型及参数个数是否完全一致(即功能的类型检查,这样做的目的是为了提高匹配的效率)。

2)然后提取功能的后条件表达式。在一个功能中,会包含很多代码或者伪代码,其中,有中间变量,也有与后条件无关的表达式等,本文只提取与后条件相关的项并进行化简等规范化处理。

3)采用预先定义的规则集对表达式进行规范化处理(表达式的形式多种多样,计算机无法像人脑那样,对不同形式的表达式进行灵活处理),以降低后面的匹配过程中的复杂度。

4)将表达式转化为二叉树。因为转换为二叉树之后,可以采用自顶向下的匹配方法,也可以采用自底向上的匹配方式。

5)最后对处理过的后条件表达式进行匹配,以便得到最终的匹配结果。

3.2 匹配规则及表达式规范化

描述一个功能,重点关注的就是前、后条件的变化,与前、后条件无关的状态、变量可以忽略。

图2描述的是对一段C语言描述的功能(函数代码)进行后条件提取的简单例子。对于功能中的表达式,可能会存在不规范的描述,为了降低功能匹配的难度,在进行功能匹配前,首先对表达式按照如下规则进行规范化处理[14-16]。

1)表达式化简。在数学运算中,主要是针对可通过基本数学运算的化简。如:$6-4\times8=-26$; $x+0=0+x=x$; $x/x=1$; $2*x=x+x$; $x^2=x*x$。

2)表达式符号规范化。为了降低分析表达式结构时的难度,将复合赋值号转变为二元赋值号。

$x+=y$ $\quad x=x+y$

$x-=y$ $\quad x=x-y$

$x*=y$ $\quad x=x*y$

$x/=y$ $\quad x=x/y$

自增、自减也转变为相应的二元表达式。

$y=x++$ $\quad y=x;x=x+1$

$y=++x$ $\quad x=x+1;y=x$

$y=x--$ $\quad y=x;x=x-1$

$y=--x$ $\quad x=x-1;y=x$

表达式中存在括号,会增加表达式比较的难度。因此,需要消除表达式中存在的括号,可以通过使用交换律、结合律、分配率来消除表达式中的括号。

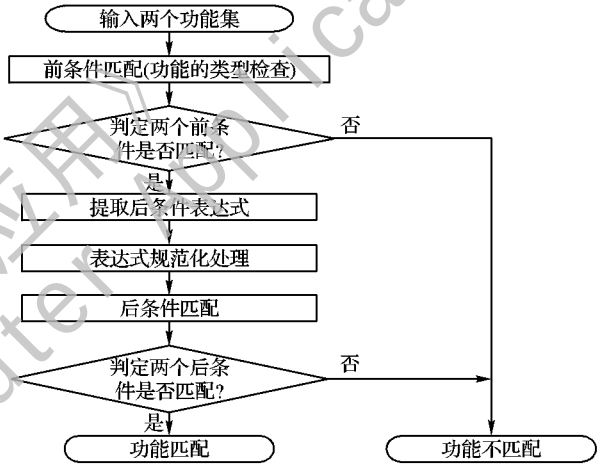

图1 代数表达式匹配流程

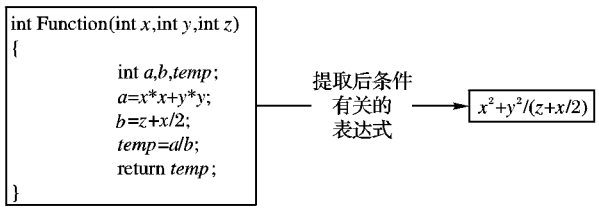

图2 后条件的提取

3)消除中间变量。中间变量的用途有多种,可以用于值交换、值传递等。例如:$a=x+2*y+3$; $b=y/7-x$; $z=a^2+b^2$。其中:变量 a 和 b 只是起到值传递的作用,可以将这两个变量消除,不会影响到 z 的取值,消除两个中间变量后 z 的赋值表达式如下:

$$z=(x+2*y+3)^2+[(y/7)-x]^2$$

4)表达式元素排序。表达式由常量、变量、运算符、括号等组成。在同级运算符下,变量间的顺序是不固定的。为了降低表达式匹配的难度,本文规定了同级运算符下常量、变量出现的次序:先常量,后变量,常量按值大小顺序,变量按字母表顺序。如:$x_1=y+z+3\times c$,经过排序后为 $x_1=3\times c+y+z$。同级运算符在表达式出现的次序:先加后减、先乘后除。如:$s=a-c+d$,经过排序后为 $s=a+c-d$。

5)表达式树形化。经过规范化处理之后的表达式是一个中缀表达式,虽然中缀表达式便于理解,但是对计算机而言中缀表达式是十分复杂的结构,而逆波兰式(后缀表达式)对计

算机来说是比较简单易懂的结构[16],因此采用算法,将中缀表达式转换为逆波兰表达式。

3.3　功能匹配算法的形式化描述

定义 1　形如:$F = \langle FN, PL, T\rangle$ 称为功能,其中 FN(Function Name)表示功能名(狭义上可指函数名);PL(Parameter List)表示功能参数列表(狭义上可指代前条件),包含参数的类型和参数的个数;T 是功能中有关后条件的表达式的集合。

定义 2　一个功能的参数列表 PL 经过计算后,其计算结果包含了该参数列表类型及个数,称作参数列表计算集,表示为:$PLC = \langle FN, PT, PTC\rangle$,其中 FN 表示功能名;PT(Parameter Type)表示参数列表中的类型的集合,PTC(Parameter Type Count)表示某个参数类型的个数。

定义 3　一个规则 p 具有如下形式:$p = \langle p_l, p_r\rangle$,其中 p_l 是规则的左部,p_r 是规则的右部,若 p_l 与表达式 T 或 T 的一部分的结构相匹配,那么就可以对表达式 T 应用规则 p,将 T 变换为 p_r 的结构形式,得到表达式 T'。

定义 4　若干规则 p 一起构成了规则集 P(或称规则库),P 中的规则须满足以下的两种性质。

1)不可逆性。若有规则 R 使得表达式由结构形式 α 变换到 β,则不存在规则 S 使得该表达式由结构形式 β 到 α。例如,若规则集中包含了 $a*(b+c) = a*b + a*c$,则不再包含规则 $a*b + a*c = a*(b+c)$。这样做可以保证在应用规则集处理之后同样功能的表达式的结构形式一致,便于功能比较,同时也可避免产生死循环。

2)非传递性。若规则集中有规则 R 使得表达式由结构形式 α 变换到 β,并且有规则 S 使得该表达式由结构形式 β 到 γ,那么规则集中不存在规则 T,使得表达式由结构形式 α 到 γ。这样可以减少规则匹配的次数。

定义 5　一个功能的后条件表达式 s 具有形式:$s = \langle C, E\rangle$,其中 E 表示与后条件相关的表达式,C 是执行该表达式需要满足的条件。s 可以描述为 $s = (C_0 \wedge E_0) \vee \cdots \vee (C_m \wedge E_m)$,$m \in \mathbf{N}$。

定义 6　一个功能与后条件相关的表达式 s 往往不止一个,那么功能的所有后条件表达式 s 放在一起构成了该功能的后条件表达式集 S,表示为:$S = (s_1, s_2, \cdots, s_n)$。

算法 1　功能的类型检查算法,如图 3 所示。

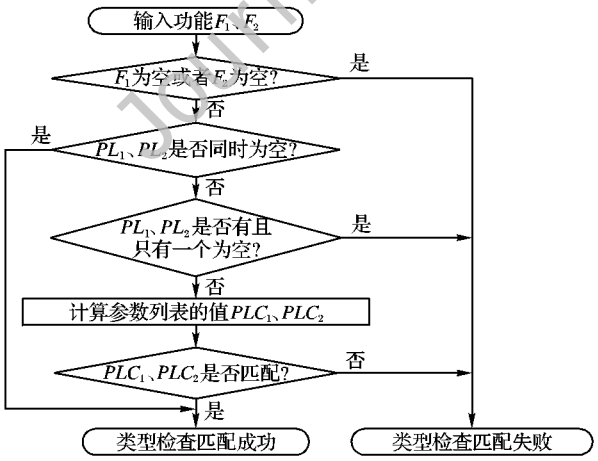

图 3　功能类型检查流程

输入　要匹配的两个功能 F_1 和 F_2,$F_1 = \langle FN_1, PL_1, T_1\rangle$,$F_2 = \langle FN_2, PL_2, T_2\rangle$。其中,表达式集 T_1 和 T_2 暂不输入。

输出　若类型检查匹配成功,返回真;否则,返回假。

1)初始如果输入的功能 F_1 或 F_2 为空,或者二者同时为空,类型检查匹配失败,算法结束,返回假。

2)接着如果功能的参数列表 PL_1、PL_2 都为空,则类型检查匹配成功,算法结束,返回真。

3)否则继续判断它们是否有且只有一个为空,是则类型检查匹配失败,算法结束,返回假。

4)否则计算它们的功能参数列表计算集:$PLC_1 = calcn(F_1)$,计算后得到结果为 $PLC_1 = \langle FN_1, (pt_1, \cdots, pt_n), (ptc_1, \cdots, ptc_n)\rangle (n \in \mathbf{N}, n \neq 0)$;$PLC_2 = calcn(F_2)$,计算后得到结果为 $PLC_2 = \langle FN_2, (pt_1, \cdots, pt_m), (ptc_1, \cdots, ptc_m)\rangle (m \in \mathbf{N}, m \neq 0)$。

5)最后 $match(PLC_1, PLC_2)$,匹配失败,返回假,匹配成功,返回真。

算法 2　功能的后条件提取算法,如图 4 所示。

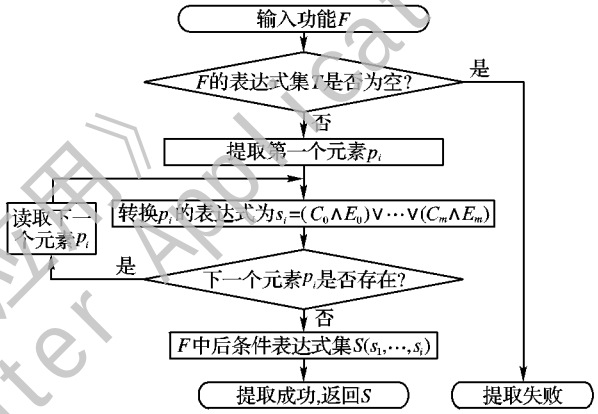

图 4　提取后条件表达式流程

输入　经过类型检查匹配成功之后的功能 F,包括功能名称 FN、参数列表 PL 以及功能 F 的表达式集 T(后条件表达式的介绍参见 3.2 节)。

输出　提取成功返回功能 F 的后条件表达式集 S;提取失败,则返回空。

1)初始若 $T = \varnothing$,提取失败,算法结束,返回空。

2)$s_1 = search(p_1) = (C_0 \wedge E_0) \vee \cdots \vee (C_m \wedge E_m)$,$search(p_1)$ 查找出某个后条件的表达式,并且转换为符合上式右部的形式。对于后条件表达式中的逻辑表达式采取拆分的处理方式,比如 $(a > 0)\&\&(b < 0)$,匹配时看做 $a > 0$ 和 $b < 0$ 分开处理,这样做既方便后面进行匹配且仍然符合语义匹配机制中的前、后条件 pre/post 的完全匹配机制。

3)若 T 中存在下一个元素,取 T 的下一个元素 p_i,重复执行步骤 2)。

4)否则,功能 F 的后条件表达式集 $S = (s_1, s_2, \cdots, s_{i-1})$,提取成功,返回 S。

算法 3　表达式规范化处理算法,如图 5 所示。

输入　执行算法 2 之后得到的后条件表达式集 S;规则集 P。

输出　规范化处理之后的后条件表达式集 S'。

1)初始 $S' = \varnothing$;若 $S = \varnothing$,算法结束,返回 S'。

2)对于 S 的第一项 s_1,处理得到 $s'_1 = change(s_1, P)$,即依据规则集 P 对 s_1 进行规范化处理,将规范化后的表达式保存到 s'_1 中。

3)$sort(s'_1)$,依据 3.2 节提到的表达式元素排序规则对

s'_1 中的项排序。读取表达式集 S 中的下一项 s_i，执行以上两个步骤。

4）规范化处理完 S 中的所有项后，返回新的后条件表达式集 $S'=(s'_1,s'_2,\cdots,s'_{i-1})$。

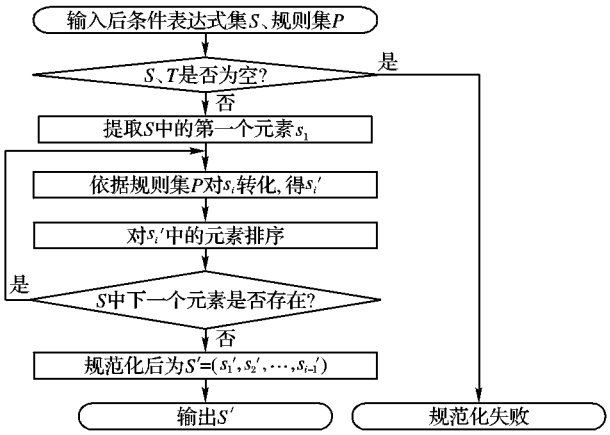

图 5 表达式集的规范化处理流程

算法 4 两个功能的后条件匹配算法，如图 6 所示。

输入 经过类型检查和规范化处理之后的功能 F_1 和 F_2，包括功能名 FN_1 和 FN_2，参数列表 PL_1 和 PL_2；规范化处理之后的后条件表达式集 S_1 和 S_2。

输出 匹配成功返回真，否则返回假。

1）初始 $ismatch$ = False。

2）$replace(PL_2,PL_1)$，$s_{21}=replace(S_2,PL_1)$，将 F_2 中的参数列表 PL_2 的值和 S_2 中每一项 s_i 的参数值用 PL_1 替换得到 s_{2i}，对于 S_1 来说，实际上 $s_{1i}=s_i$。

3）对 $s_{11}=(C_0\wedge E_0)\vee\cdots\vee(C_m\wedge E_m)$（这里 s_{11} 表示 S_1 中第一项 s_1 的参数用 PL_1 替换，实际上还是 s_1 自身）和 $s_{21}=(C'_0\wedge E'_0)\vee\cdots\vee(C'_m\wedge E'_m)$ 中的各项，进行逆波兰式排序 $revesort(C_i)(i\in\mathbf{N},i\neq 0)$ 和 $revesort(E_i)(i\in\mathbf{N},i\neq 0)$。

4）对表达式集的各项做匹配[17]，$match'(C_i,C'_j)(i\in\mathbf{N},i\neq 0;j\in\mathbf{N},j\neq 0)$ 和 $match'(E_i,E'_j)(i\in\mathbf{N},i\neq 0;j\in\mathbf{N},j\neq 0)$。

5）若此处匹配失败，则后条件匹配失败，$ismatch$ = False；否则，则取 S_1 和 S_2 的下一个元素，继续进行第 2）、3）、4）步的变换与匹配。

6）若各项均匹配成功，$ismatch$ = True。

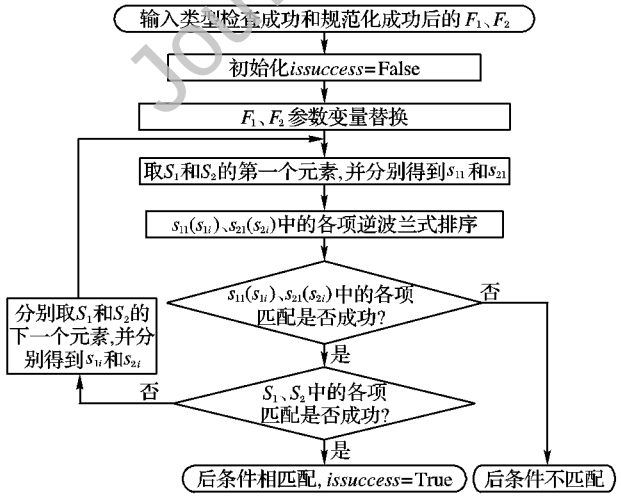

图 6 后条件匹配流程

4 算法验证

本文将通过一个实例，验证以上算法的有效性，即验证该算法能够从数学体系上检验两个功能是否一致，并且也不存在词义理解偏差问题。下面有 4 个不同的功能，描述如下。

1）求自然数 N 的阶乘。

方法 1 用递归方法求 N 的阶乘，代码如下：

```
int fact1(int n)
{
  int sum=0;
  if(n==0) {sum=1;}
  else{sum=n*fact1(n-1);}
  return sum;
}
```

方法 2 用 while 循环求 N 的阶乘，代码如下：

```
int fact2(int m)
{
  int temp=1;
  while(m>0)
  {temp*=m; m-=1;}
  return temp;
}
```

方法 3 用 for 循环求 N 的阶乘，代码如下：

```
int fact3(int k)
{
  int r=1;
  for(int i=k; i>0; i--)
  {r*=i;}
  return r;
}
```

2）求自然数 1,2,…,N 的和。

```
int sum(int k)
{
  int sum=0;
  for(int i=k; i>0; i--)
  {sum+=i;}
  return sum;
}
```

匹配过程如下。

首先运用算法 1，方法 1 中功能 $F_1=\langle fact_1,\text{int } n,T_1\rangle$，对其参数列表进行计算 $PLC_1=\langle fact_1,(\text{int},1)\rangle$；方法 2 中功能 $F_2=\langle fact_2,\text{int } m,T_2\rangle$，计算 $PLC_2=\langle fact_2,(\text{int},1)\rangle$；方法 3 中功能 $F_3=\langle fact_3,\text{int } k,T_3\rangle$，计算 $PLC_3=\langle fact_3,(\text{int},1)\rangle$；对自然数求和方法中功能 $F=\langle sum,\text{int } k,T\rangle$，计算 $PLC=\langle sum,(\text{int},1)\rangle$。$PLC_1$、$PLC_2$、$PLC_3$ 和 PLC 进行两两匹配，匹配成功，返回匹配结果：类型检查全部匹配。

类型检查匹配成功后，运用算法 2，提取后条件表达式。上述前 3 个方法均是求阶乘的运算，第 4 个方法是求和运算。

在方法 1 中，后条件为自然数 N 的阶乘，只有一个值，其表示形式为 $S_1=(s_{11})$，$s_{11}=(C_0\wedge E_0)\vee(C_1\wedge E_1)$，其中 s_{11} 中各项的值为：$C_0=\{n==0\}$；$E_0=\{sum=1\}$；$C_1=\{n>0\}$；$E_1=\{sum=n*(n-1)*\cdots*1\}$。

同样在方法 2 中，$S_2=(s_{21})$，$s_{21}=(C'_0\wedge E'_0)\vee(C'_1\wedge E'_1)$，其中 s_{21} 中各项值为：$C'_0=\{m==0\}$；$E'_0=\{temp=1\}$；$C'_1=\{m>0\}$；$E'_1=\{temp=1;temp*=m*(m-1)*\cdots*1\}$。

在方法 3 中,$S_3 = (s_{31})$,$s_{31} = (C''_0 \wedge E''_0) \vee (C''_1 \wedge E''_1)$,$s_{31}$ 中各项值为:$C''_0 = \{k == 0\}$; $E''_0 = \{r = 1\}$; $C''_1 = \{k > 0\}$; $E''_1 = \{r = 1; r** = k*(k-1)*\cdots*1\}$。

在方法 4 中,$S = (s_1)$,$s_1 = (C_0^3 \wedge E_0^3) \vee (C_1^3 \wedge E_1^3)$,其中 s_1 中各项值为:$C_0^3 = \{k == 0\}$; $E_0^3 = \{r = 1\}$; $C_1^3 = \{k > 0\}$; $E_1^3 = \{r = 1; r += k + (k-1) + \cdots + 1\}$。

返回表达式集 S_1、S_2、S_3 和 S。

采用算法 3 对 S_1、S_2、S_3 和 S 进行规范化处理,结果如下:

S_1 规范化处理之后的表达式为:$C_0 = \{n == 0\}$; $E_0 = \{1\}$; $C_1 = \{n > 0\}$; $E_1 = \{n*(n-1)*\cdots*1\}$。

S_2 规范化处理之后为:$C'_0 = \{m == 0\}$; $E'_0 = \{1\}$; $C'_1 = \{m > 0\}$; $E'_1 = \{m*(m-1)*\cdots*1\}$。

S_3 规范化处理之后为:$C''_0 = \{k == 0\}$; $E''_0 = \{1\}$; $C''_1 = \{k > 0\}$; $E''_1 = \{k*(k-1)*\cdots*1\}$。

S 规范化处理之后为:$C_0^3 = \{k == 0\}$; $E_0^3 = \{1\}$; $C_1^3 = \{k > 0\}$; $E_1^3 = \{k + (k-1) + \cdots + 1\}$。

S_1、S_2、S_3 和 S 采用算法 4 进行匹配,首先对 S_2、S_3 和 S 中参数列表及表达式中的参数值进行替换:

$C'_0 = \{n == 0\}$;　　$C''_0 = \{n == 0\}$;

$E'_0 = \{1\}$;　　$E''_0 = \{1\}$;

$C'_1 = \{n > 0\}$;　　$C''_1 = \{n > 0\}$;

$E'_1 = \{n*(n-1)*\cdots*1\}$;　　$E''_1 = \{n*(n-1)*\cdots*1\}$;

$C_0^3 = \{n == 0\}$;

$E_0^3 = \{1\}$;

$C_1^3 = \{n > 0\}$;

$E_1^3 = \{n + (n-1) + \cdots + 1\}$

接下来对四个表达式集的各项进行逆波兰式排序:

$C_0 = \{n0 ==\}$;　　$C'_0 = \{n0 ==\}$;

$E_0 = \{1\}$;　　$E'_0 = \{1\}$;

$C_1 = \{n0 >\}$;　　$C'_1 = \{n0 >\}$;

$E_1 = \{n(n-1)*\cdots*1*\}$;　　$E'_1 = \{n(n-1)*\cdots*1*\}$;

$C''_0 = \{n0 ==\}$;　　$C_0^3 = \{n0 ==\}$;

$E''_0 = \{1\}$;　　$E_1^3 = \{1\}$;

$C''_1 = \{n0 >\}$;　　$C_1^3 = \{n0 >\}$;

$E''_1 = \{n(n-1)*\cdots*1*\}$;　　$E_1^3 = \{n(n-1) + \cdots + 1 +\}$

之后两两进行匹配,S_1、S_2 和 S_3 匹配成功,说明 $fact_1$、$fact_2$ 和 $fact_3$ 这三个功能实现的功能是匹配的,S 和 S_1、S_2、S_3 三个中任一匹配都不成功,说明 *sum* 与 $fact_1$、$fact_2$、$fact_3$ 要实现的功能不同。

5　结语

本文以实现工作流系统自动集成为目标,对基于形式化语义的前、后条件 pre/post 完全匹配机制进行了实现,提出了以代数表达式为基础的前后条件 pre/post 的完全匹配的原则,并且给出了相应算法,最后用一个实例验证了该算法的有效性。该匹配机制源自于以集合论和一阶谓词逻辑为基础的形式化规格描述,避免了关键字理解意义上的偏差,一定程度上提高了功能查准率和查全率,适用于工作流集成中的功能匹配,为工作流功能的自动查找奠定了基础。该方法局限于初等代数性的前提,而实际情况中,抽象机作为一种伪程序设计语言,用它描述的程序不仅能够采用过程性程序设计语言书写,更关键的是它的每种结构都有精确的公理化定义,对于数学分析来说尤为重要。因此,基于抽象机的功能匹配是下一步的研究内容。

参考文献:

[1] PAPAZOGLOU M P, van den HEUVEL W J. Service oriented architectures: Approaches, technologies and research issues [J]. The International Journal on Very Large Data Bases, 2007, 16(3): 389 - 415.

[2] 张静,王海洋,崔立真. 基于 Pi 演算的跨组织工作流建模研究[J]. 计算机研究与发展,2007,44(7):1243 - 1251.

[3] XU LIDA, LIU HUIMIN, WANG SONG, *et al*. Modelling and analysis techniques for cross-organizational workflow systems [J]. Systems Research and Behavioral Science, 2009, 26(3): 367 - 389.

[4] LIN QIANG, GE JIDONG, HU HAO, *et al*. An approach to model cross-organizational processes using object Petri net [C]// SCW 2007: 2007 IEEE International Conference on Services Computing-Workshops. Washington, DC: IEEE Computer Society, 2007: 146 - 152.

[5] 吴长中. 工作流中的数据集成技术研究[D]. 长沙:国防科学技术大学,2002.

[6] KOURTESIS D, PARASKAKIS I. Combining SAWSDL, OWL-DL and UDDI for semantically enhanced Web service discovery [C]// ESWC 08: Proceedings of the 5th European Semantic Web Conference on the Semantic Web: Research and Applications, LNCS 5021. Berlin: Springer-Verlag, 2008: 614 - 628.

[7] 张小娟,李华. 网格环境下基于语义关联的信息检索[J]. 计算机应用,2009,29(6):1517 - 1526.

[8] 樊晓光,褚文奎,万明. 基于领域本体的软构件检索[J]. 计算机科学,2009,36(6):156 - 158.

[9] 袁世伦,李胜利,袁平鹏,等. 一种基于规则的工作流模型互操作的实现方法[J]. 计算机应用,2007,27(2):400 - 402.

[10] ZHANG JIDONG, HUO WEIPENG. Research on the composition mechanism of semantic Web services [C]// ICCSIT 2010: Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology. Washington, DC: IEEE Computer Society, 2010: 100 - 103.

[11] 王淑红,袁兆山. 基于排序形式化规格说明的软构件匹配[J]. 合肥工业大学学报,2000,23(4):477 - 481.

[12] HEMER D. Semi-automated component-based development of formally verified software [C]// REFINE 2006: Proceedings of the 11th Refinement Workshop on Electronic Notes in Theoretical Computer Science. Amsterdam, Netherlands: Elsevier, 2007: 173 - 188.

[13] 马亮,孙家. 基于规约匹配的构件检索[J]. 小型微型计算机系统,2002,23(10):1153 - 1157.

[14] 樊敏. 程序作业自动测评的研究与实现[D]. 广州:广东工业大学,2005.

[15] 刘洋,欧阳春宜,施崇明. 一类算术表达式的语法分析及规范化方法[J]. 赣南师范学院学报,2003(3):21 - 23.

[16] 刘峰,袁春风. 基于 MathML 的数学表达式等价性的研究[J]. 计算机应用研究,2004,21(11):54 - 56.

[17] 曹京,谭建龙,刘萍,等. 布尔表达式匹配问题研究[J]. 计算机应用研究,2007,24(9):70 - 72.