软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2014,25(6):1212−1224 [doi: 10.13328/j.cnki.jos.004515]

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

Tabular 表达式的指称语义研究*

张 鹏, 刘 磊, 刘华虓, 金 英

(吉林大学 计算机科学与技术学院,吉林 长春 130012)

通讯作者: 刘华虓, E-mail: jlulhx@gmail.com

摘 要: Tabular 表达式是一种采用表格化结构组织函数或关系的形式化描述工具,在需求工程领域中具有广泛的应用,为 Tabular 表达式建立形式的语义模型是非常必要的.针对 Tabular 表达式通用模型,给出了 Tabular 表达式的形式文法及指称语义.通过定义形式文法中各语法单元的语义指派方程,描述了 Tabular 表达式的指称语义,分别对传统类型 Tabular 表达式和新类型 Tabular 表达式中一些典型表类型的指称语义进行了描述,并与其他几种 Tabular 表达式的语义描述方法进行了比较.分析结果表明:该语义描述方法不仅准确描述了 Tabular 表达式的语义,而且不再受 Tabular 表达式模型和 Tabular 表达式类型的限制,打破了现有方法的局限性,是一种非常有效的方法.

关键词: Tabular 表达式;指称语义;软件说明文档

中图法分类号: TP311

中文引用格式: 张鹏,刘磊,刘华虓,金英.Tabular 表达式的指称语义研究.软件学报,2014,25(6):1212−1224. http://www.jos.org.cn/1000-9825/4515.htm

英文引用格式: Zhang P, Liu L, Liu HX, Jin Y. Denotational semantics of Tabular expressions. Ruan Jian Xue Bao/Journal of Software, 2014,25(6):1212−1224 (in Chinese). http://www.jos.org.cn/1000-9825/4515.htm

Denotational Semantics of Tabular Expressions

ZHANG Peng, LIU Lei, LIU Hua-Xiao, JIN Ying

(College of Computer Science and Technology, Jilin University, Changchun 130012, China)

Corresponding author: LIU Hua-Xiao, E-mail: jlulhx@gmail.com

Abstract: Tabular expressions are a formal notation using Tabular form to organize functions or relationships, and they have been widely used in documenting and analyzing software systems. To avoid misunderstanding and to support user with trustworthy tools, the meaning of these expressions must be fully defined. This paper presents the formal grammar and denotational semantics of the Tabular expressions in general model. By defining semantics assigned equation of each syntax unit of the Tabular expressions' formal grammar, denotational semantics of the Tabular expressions is specified. In addition, the meaning of some classical Tabular expression types and new table types are described based on this semantics. Comparisons made with other semantics description methods show that the denotational semantics method defines Tabular expressions' meaning precisely without subjecting to the restrictions of models and types of Tabular expressions. The proposed method breaks the existing methods' limitations and is an effective method.

Key words: Tabular expression; denotational semantic; software documentation

Tabular 表达式是一种采用表格化结构来组织关系或函数的形式化描述工具[1],它遵循分而治之的原则,对复杂的数学表达式进行分解和简化,在保持定义严谨性的同时,提高了定义的可读性.由于 Tabular 表达式的准

* 基金项目: 国家自然科学基金(61300049); 教育部新世纪优秀人才支持计划(NCET-10-0436); 教育部高等学校博士学科点专项科研基金(20120061120059); 中国博士后科学基金(2011M500612); 吉林省重点科技攻关项目(20130206052GX); 吉林省自然科学基金(20101509); 吉林省青年科研基金(20140520069JH)

收稿时间: 2013-02-26; 修改时间: 2013-08-21; 定稿时间: 2013-11-11; jos 在线出版时间: 2014-02-07

CNKI 网络优先出版: 2014-02-07 13:33, http://www.cnki.net/kcms/doi/10.13328/j.cnki.jos.004515.html

确性和严谨性,它可以用来定义软件开发中的需求文档和技术文档,使得设计者和维护人员可以直观地阅读开发文档中的各项内容.

20 世纪 70 年代末,Parnas 等人在为 A-7E 战斗机设计机载系统时,提出了采用表格的形式描述需求的方法,这就是 Tabular 表达式的雏形[2−4].此后,Parnas 教授对 Tabular 表达式进行了深入的研究,发表了一系列的论文,分别在软件安全开发[5,6]、软件文档的合理性描述[7−9]等方面阐述了 Tabular 表达式对软件开发所产生的积极影响.基于 Parnas 教授的研究成果,先后有很多学者利用 Tabular 表达式对软件开发过程中存在的问题进行了研究.Degiovanni 等人提出了一种自动化的软件测试用例生成方法,基于 Tabular 表达式需求描述规范生成复杂软件系统的测试用例[10];Feng 等人提出了一种软件故障定位方法,利用 Tabular 表达式追踪故障的产生过程[11];Herrmannsdörfer 等人提出了 5 种利用 Tabular 表达式描述有限自动机的方法,并对这些描述方法在软件开发中的价值进行了评估[12];Peters 等人开发了一个软件集成开发环境工具,利用 Tabular 表达式描述软件的需求,并对软件的设计进行测试与验证[13].这些研究表明:Tabular 表达式可以有效地解决软件开发过程中软件文档描述不准确和难理解两大难题,保障软件的准确开发.

Tabular 表达式最初只包含 10 种表,各个表的语义解释需要基于函数或者关系的联合单独定义,Janicki 对 Parnas 定义的一部分表给出了统一的形式模型,并采用关系语义对表类型进行了解释[14,15],但是该模型不能定义向量表和函数表.Abraham 指出了 Janicki 模型的局限性,对该模型进行了扩展,并开发了 Tabular 表达式求值代码的自动生成工具[16].Kahi 为 Tabular 表达式建立了基于复合形式的语法和语义,其通过增加标题操作和连接操作从一个网格逐步构造 Tabular 表达式,利用表折叠函数从如何求值的角度定义 Tabular 表达式的语义[17].以上这些模型形式语义的定义都依赖于一种特定的物理表示结构,要求将 Tabular 表达式表示成一个由主网格和若干标题网格组成的多维矩形结构,并要求主网格的索引集是其他标题网格索引集合的笛卡尔乘积.为了打破这一局限,增强 Tabular 表达式的适用性,parnas 教授和金英教授提出了一种 Tabular 表达式的通用模型[18],并从如何求值的角度给出了该通用模型的语义.在该模型中,不需要再对网格进行区分,索引集合也不再仅局限于整数集合.

通用模型增强了 Tabular 表达式的描述能力,同时也增加了 Tabular 表达式语义描述的难度.Tabular 通用模型用类型概念,从表达式组成和求值方案两个角度去刻画一类 Tabular 表达式所共有的特征,如网格个数、索引集合、网格包含的表达式类型、表达式的计算过程以及在求值过程中使用的一些辅助函数等.但是由于通用模型可以定义无穷种类型,表达式类型无法一一穷举,因而定义 Tabular 表达式如何求值无法完整地刻画 Tabular 表达式通用模型的语义,存在着一定的局限性.为克服这一不足,本文以定义 Tabular 表达式形式文法的形式给出了 Tabular 表达式通用模型的相关论域,并给出了各语法成分到论域上的语义映射,首次从指称语义的角度完善了 Tabular 表达式的形式语义,为 Tabular 表达式应用的进一步推广和相关模型验证工具的开发奠定了良好的基础.

本文第 1 节介绍 Tabular 表达式通用模型的形式文法和一些基本定义.第 2 节阐述 Tabular 表达式通用模型指称语义的描述方法.第 3 节和第 4 节分别就传统类型和新定义类型中一些 Tabular 表达式的指称语义进行描述,说明本文方法的有效性.第 5 节对本文的方法和一些相关进行了比较.最后是本文工作的总结.

1 基本定义

Tabular 表达式通用模型是目前为止扩展性最好的模型.本文针对该模型从 Tabular 表达式语法单元的功能的角度给出 Tabular 表达式的形式语义,该模型的形式化文法定义如下:

- *Tabular*::=*Indexes*→*Grids*
- *Indexes*::={*Index*}
- *Index*::=*integer*|*identifier*|*Index*×*Index*
- *Grids*::={*Grid*}
- *Grid*::=*Indexes*→*Expresses*

- *Expresses*::={*Express*}
- *Express*::=*Tabular*|*Scalar_express*
- *Scalar_express*::=*variable*|*constant*|*R*(*Scalar_express*,*Scalar_express*)

其中,*integer* 表示整数,*identifier* 表示表示符,*variable* 表示变量,*constant* 表示常量,*Scalar_express* 表示标量表达式,*R* 表示标量表达式间的二元关系.Tabular 表达式的本质是一个索引化的网格集合,而网格的本质是一个索引化的表达式集合.

定义 1. 设存在 n 个逻辑公式 $F_1,F_2,\ldots,F_n$ 和指令集 *Instruct_Set*,给定指派 *assign*,若 *assign* 满足所有 $F_i(1\leqslant i\leqslant n)$,则执行指令集 *Instruct_Set*,称 $F_1\rightarrow F_2\rightarrow\ldots\rightarrow F_n\rightarrow Instruct\_Set$ 为一个指引,n 为指引的维度,将 n 维指引记做 $FI(n)$.

指引的物理意义是指令集被执行的前提,是满足 n 个条件,只有 n 个条件全被满足时,该指令集才能被执行,当 n=0 时,指令集可以永远被执行.这里的指令集与程序语言中指令集有所区别,指令可以是一种操作,也可以是一种判断结果,在不同情况下可以被赋予不同的意义,后文将在 Tabular 语义情景环境下对指令的功能指代物进行讨论.

定义 2. 已知 S 为逻辑公式的集合,$\forall F_1,F_2\in S$,若有 $F_1\wedge F_2=\varnothing$,则称 S 是互斥的.

定义 3. 已知 S 为逻辑公式的集合,设 $S=\{F_1,F_2,\ldots,F_n\}$,若 $F_1\vee F_2\vee\ldots\vee F_n=1$,则称 S 是完备的.

互斥公式集合 S 可以用来形容函数的定义域,函数的参数作为公式中的变量,一组参数最多只能满足 S 中一个逻辑公式.由于函数可能是部分函数,因而存在一组参数不满足 S 中任何公式的情况.若用一个互斥且完备的公式集合 S 来形容函数的定义域,则该函数为完全函数,一组参数能且只能满足 S 中的一个公式.

定义 4. 已知 S 为包含 n 个逻辑公式的集合,设 $S=\{F_1,F_2,\ldots,F_n\}$,给定一个公式的解释 I,P 为 S 中在解释 I 下为真的逻辑公式的集合,称 $S\rightarrow P$ 为集合 S 的一个指派,记做 *assign*(S).

当 S 为互斥集合时,P 中最多只有一个逻辑公式;当 S 为互斥且完备集合时,P 中有且只有一个逻辑公式.

定义 5. 已知 E 为标量表达式,称 $E\rightarrow Value$ 为 E 的一个指派,记做 *assign*(E).

互斥集合指派的物理意义是:给定参数的赋值,使得互斥集合中的一个公式为真,其他的公式为假,进而执行相应的指令集;表达式指派的意义是:给定参数的赋值,求取标量表达式的计算结果.

定义 6. 已知 G_1 和 G_2 是 Tabular 表达式中的两个 Grid,若 G_1 中的任何一个表达式都可以在 G_2 中找到,则称 G_1 是 G_2 的子网格.

网格 G 实际上相当于一个表达式的集合,因而任何该集合子集构成的网格都是 G 的子网格.

2 Tabular 表达式的语义

本节将对 Tabular 表达式中各语法单位的功能进行分析,进而给出各语法单位的语义指代物和相应的语义指派方程.

Tabular 表达式中各语法单位的功能如下:

- *Tabular*:一组说明规范,明确不同条件下应对哪些事物进行处理;
- *Index*:对规范和事物进行分类;
- *Grid*:一项说明规范,明确不同事物的处理条件;
- *Scalar_express*:一项事物的处理过程;
- *Variable*:一个变量;
- *Constant*:一个常量.

通过上面的说明,可以给出 Tabular 表达式中各语法单位所对应的语义域,见表 1.

指引可以表示一件事物的说明规范,因而可以用 n 维指引集$\{FI(n)\}$表示 Tabular 表达式的指称语义,Tabular 表达式的语义本质就是根据 n 个辅助主题来决定怎样处理一组事物;Tabular 表达式中的每一个 *Grid* 都对事物的最终处理有影响,但给定一个 *Grid* 的指派并不能完全决定事物如何被处理,只能降低影响事物因素的数目,

因而一个 *Grid g* 的指称语义为通过给定一个指派 *assign*(*g*)将 *n* 维指引集{*FI*(*n*)}映射为一个维度较小的 *m* 维指引集{*FI*(*m*),大多数情况下,*m*=*n*−1,当 *n*=0 时,*assign*(*g*)将一组指令集映射为操作结果;*Scalar_express* 的指称语义为根据当前环境的指派情况给出一件事物的处理结果.*Variable* 和 *Constant* 都是特殊的 *Scalar_ express*,其语义本质与 *Scalar_express* 相同;*Index* 在 Tabular 表达式中起到索引的作用,其指称语义就是网格或者表达式的索引标签.

Table 1 Semantic domain of tabular expressions

表 1 Tabular 表达式的语义域

语法域	语义域
Tabular	*Account*(*n*)={*FI*(*n*)}
Index	*Tabular*→*Grid*|*Grid*→*Express*
Grid	*assign*(*g*)×*Account*(*n*)→*Account*(*m*)(*n*>*m*)|*assign*(*g*)×*Account*(0)→{*Value*}
Scalar_express	{*assign*(*E*)}→*Value*
Variable	变量在当前环境下的取值
Constant	常量的值
Express	{*assign*(*E*)}×*Account*(*n*)→*Account*(*m*)(*n*>*m*)|{*assign*(*E*)}→*Value*

Grid 是索引化的表达式集合,由于表达式包含标量表达式和 Tabular 表达式两种,因而表达式的计算结果可以是一个值,也可以是指引集,只是指引集的维度要小于当前指引集的维度.值得注意的是,这里,*Value* 的意义并不仅仅局限于数学上的数值,而是一件事物的处理结果,可以是数值或状态,也可以是一个决定,具体取值取决于事物本身的属性.

根据上文的分析可知,Tabular 的指称语义是一个指引集,在这个情景环境下,指引中的指令相当于 Tabular 表达式中的 *express*,因而指引中的指令可以是标量表达式,也可以是 Tabular 表达式.标量表达式由变量、常量和变量表达式间的二元关系构成,由于任何一种运算的本质都是将算子作用在变量或常量上,算子就是量纲之间的二元关系.因而,本文用逻辑公式来描述标量表达式,当两个量纲 *x* 和 *y* 具有关系 *R* 时,逻辑公式 *R*(*x*,*y*)返回为真值,否则为假值.

若设指引集 *S* 包含 *n* 个指引,第 *i*(1≤*i*≤*n*)个指引 $FI_i=F_{i1}\rightarrow F_{i2}\rightarrow\ldots\rightarrow F_{im}\rightarrow Instruct\_Set_i$,则经过如下变换可以将 *S* 转换为等价的逻辑公式 *F*:

1) 将第 *i*(1≤*i*≤*n*)个指引 FI_i 转换 $F_i=\neg F_{i1}\vee\neg F_{i2}\vee\ldots\vee\neg F_{im}\vee Instruct\_Set_i$;

2) $F=F_1\vee F_2\vee\ldots\vee F_n$.

由于每个指引 FI_i 的指令集是标量表达式或者 Tabular 表达式,并且可以变换为相应的逻辑公式,因而指引集可以转换为等价的逻辑公式.也就是说, *express* 可以转换为等价的逻辑公式. *Grid* 是索引化的表达式集合,因而 *Grid* 的语义本质是逻辑公式的集合.

语义指派函数将语法域中的每个语法单元映射为相应的语义指代物,Tabular 表达式各语法单元语义指派函数的函数空间如下:

- K_T:*Tabular*→*Account*(*n*)
- K_G:*Grid*→[*assign*(*g*)×*Account*(*n*)→*Account*(*m*)]+[*assign*(*g*)×*Account*(0)→{*Value*}](*n*>*m*)
- K_E:*Express*→[{*assign*(*g*)+*assign*(*E*)}×*Account*(*n*)→*Account*(*m*)+*Value*](*n*>*m*)
- K_{SE}:*Scalar_express*→[{*assign*(*E*)}→*Value*]
- K_I:*Index*→[*Tabular*→*Grid*]+[*Grid*→*Express*]
- K_V:*Variable*→[{*assign*(*s*)}→*Value*]
- K_C:*Constant*→*Constant*

由于变量本身也属于标量表达式,因而它们的语义指派函数具有相同的函数空间;常量虽然也属于标量表达式,但由于常量的值和指派环境无关,只与自身有关,因而常量的语义指代物是自身的值.在对维度为 0 的指引集做指派时,相当于对 *Grid* 中的标量表达式做指派,此时,*assign*(*g*)的作用相当于 *assign*(*E*)的集合,如果 *Grid* 中只有一个标量表达式,则 *assign*(*g*)与 *assign*(*E*)的作用相同.

我们曾在文献[18]中从如何求值的角度研究了 Tabular 表达式的语义,用通用函数 *eval* 解释 Tabular 表达式的语义,*eval* 对传统的表达式计算进行了扩充,其定义域是传统表达式和所有 Tabular 表达式.在这里,本文从指称语义的角度给出 *eval* 的语义:

$$eval{:}K_T[Tabular]\times\{assign(s)\}\rightarrow Value.$$

文献[18]将类型的概念引入到了 Tabular 中,从约束谓词、求值方案和辅助函数这 3 个方面定义 Tabular 表达式的类型,将具有相同组织形式的不同 Tabular 表达式划分为同一类型.本文从以下 4 个方面解读 Tabular 表达式中类型的语义:

1) 若设 Tabular 表达式 *T* 的指称语义为指引集 *S*,则 *T* 的类型需要给出 *S* 的维度 *n*;
2) 若 *AS* 为 Tabular 表达式 *T* 对应的 *eval* 函数的指派集,则 *T* 的类型需要给出 *AS* 中指派的指派顺序;
3) 若 Tabular 表达式 *T* 索引的 *Gird* 集合为$\{g_1,g_2,\ldots,g_n\}$,则 *T* 的类型需要给出 $g_i(1\leqslant i\leqslant n)$所对应的公式集合的互斥性与完备性如何;
4) 若 Tabular 表达式 *T* 索引的 *Gird* 集合为$\{g_1,g_2,\ldots,g_n\}$,则 *T* 的类型需要给出 $g_i(1\leqslant i\leqslant n)$中表达式的具体类型.

Tabular 表达式的类型体现了表达式的组织形式和求值策略,从组成和求值两个角度完成对 Tabular 表达式类别的划分.值得注意的是,表达式的具体类型不仅仅包括标量表达式和 Tabular 表达式的划分,还包括表达式的具体组织形式.

由于根据 Tabular 表达式通用模型的定义可以定义无穷种 Tabular 表达式的类型,因而无法从类型的角度对 Tabular 表达式的语义进行全面地分析.根据 Tabular 表达式通用模型的形式文法,可以定义任意类型的 Tabular 表达式,从 Tabular 表达式的形式文法分析入手,针对各语法单元给出相应的语义指派方程,可以使 Tabular 表达式的语义分析不再受类型的限制.任何一种类型的 Tabular 表达式,只要满足 Tabular 表达式通用模型的形式文法,就适用于本文的方法.后文将就 Tabular 表达式通用模型中具有代表性的表对本文方法的有效性进行分析.

3 传统类型 Tabular 表达式的语义

Tabular 表达式最初只包含正规函数表、反转函数表、向量函数表等 10 种表的形式化定义.本节将对传统类型 Tabular 表达式中一些具有代表性的表类型进行语义描述.

3.1 正规函数表

一个正规函数表中包含 1 个主 *Grid* 和 *n* 个辅助 *Grid*:主 *Grid* 中包含的表达式都是由常量、变量或函数的施用组成,本文将这种表达式统称为算术表达式;辅助 *Grid* 中的表达式都是谓词表达式.通常用一个正规函数表来描述一个函数,该函数的定义域是一组赋值,*n* 个辅助 *Grid* 描述主 *Grid* 各项取值需满足的各种条件,主 *Grid* 描述了函数的具体取值结果.图 1 是一个正规函数表的例子.

T[2] \ *T*[1]	*A*>0	*A*=0	*A*<0
B>3	*A*+*B*	*A*–*B*	*A***B*
B=3	*A*–*B*	*A*+*B*	*A*+*B*
B<3	*A*+*B*	*A*–*B*	*A***B*

T[0]

Fig.1 Example of a normal function table

图 1 正规函数表的例子

图 1 中的正规函数表包含 3 个 *Grid*,分别是 *T*[0],*T*[1]和 *T*[2],*T*[1]和 *T*[2]描述参数可能出现的各种情况,*T*[0]描述了各种情况下函数 *f*(*A*,*B*)的取值.该正规函数表描述的函数为

$$f(A,B)=\begin{cases}A+B, & (A>0\wedge B>3)\vee(A\leqslant 0\wedge B=3)\vee(A>0\wedge B<3)\\ A-B, & (A>0\wedge B=3)\vee(A=0\wedge B>3)\vee(A=0\wedge B<3)\\ A\times B, & (A<0\wedge B>3)\vee(A<0\wedge B<3)\end{cases}.$$

若设一个正规函数表 T 由 1 个主 *Grid* $T[0]$和 n 个辅助 *Grid* $T[1],T[2],\ldots,T[n]$组成,第 $i(1\leqslant i\leqslant n)$个辅助 *Grid* $T[i]$中包含 m_i 个谓词表达式,S 为 T 对应指称语义的指引集,则正规函数表类型具有如下语义:

1) S 中指引的维度为 $n(n\geqslant 1)$;
2) T 的指派顺序为先辅助 *Grid*、后主 *Grid*,$T[i](1\leqslant i\leqslant n)$的指派间无先后关系;
3) $T[i](1\leqslant i\leqslant n)$是互斥且完备的;
4) $T[i](1\leqslant i\leqslant n)$中的表达式为谓词表达式,$T[0]$中的表达式为算术表达式.

正规函数表 T 的具体指称语义如下:

- $K_T[T]=\{T[1][i_1]\rightarrow T[2][i_2]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_1,i_2,\ldots,i_n]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[k]](1\leqslant k\leqslant n)=K_T[T]\times assign(T[k])=\{T[1][i_1]\rightarrow\ldots\rightarrow T[k-1][i_{k-1}]\rightarrow T[k+1][i_{k+1}]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_1,\ldots,i_{k-1},a_k,i_{k+1},\ldots,i_n]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[0]]=[K_T[T]\times assign(T[1])\times assign(T[2])\times\ldots\times assign(T[n])]\times assign(T[0])=\{Value\}$;
- $K_{SE}[T[i][j]](1\leqslant i\leqslant n,0\leqslant j\leqslant m_i)=assign(T[i])\rightarrow Bool$;
- $K_{SE}[T[0][i_1,i_2,\ldots,i_n]](1\leqslant i_j\leqslant m_j,1\leqslant j\leqslant n)=assign(T[0])\times assign(T[1])\times\ldots\times assign(T[n])\rightarrow Value$;
- $eval(T)=K_T[T]\times assign(T[1])\times assign(T[2])\times\ldots\times assign(T[n])\times assign(T[0])\rightarrow Value$.

$T[i](1\leqslant i\leqslant n)$的指派情况将参数映射到具体求值表达式,$T[0]$的指派情况给出了参数在各种条件下的计算结果.正规函数表的取值结果就是函数针对一组参数赋值的计算结果.对 $T[0]$中表达式的结果进行计算不仅仅需要 *Grid* $T[0]$的指派,还需要辅助 *Grid* $T[i](1\leqslant i\leqslant n)$的指派,因为辅助 *Grid* 的指派情况决定了表达式的计算结果为空还是具体的值.

上文的 *Bool* 代表波尔类型的变量,只能取真值或假值;*Value* 代表一个数值,是算术表达式的计算结果;a_k 代表 $assign(T[k])$的指派结果.下文在没有特殊说明的情况下,上述符号的意义与此处相同.索引只是为 Tabular 表达式和 *Grid*,或者为 *Grid* 和表达式建立关联,语义并不会随着 Tabular 表达式类型的变化而发生变化;常量和变量都是构成标量表达式的基本元素,语义也不会随着 Tabular 表达式类型的变化而变化,因而关于索引、变量和常量的指称语义本文不再重复赘述.

3.2 反转函数表

反转函数表同样由 1 个主 *Grid* 和 n 个辅助 *Grid* 组成,与正规函数表不同的是:主 *Grid* 中的表达式是谓词表达式,而 1 个辅助 *Grid* 中的表达式是算术表达式,其余 $n-1$ 个辅助 *Grid* 中的表达式是谓词表达式.反转函数表描述的仍然是函数,任何被反转函数表描述的函数都可以用正规函数表来描述.图 2 是反转函数表的例子.

$T[1]$

$x+y$	$x-y$	$x*y$

$T[2]$

$y<0$
$y=0$
$y>0$

$x<0$	$x=0$	$x>0$
$x>0$	$x<0$	$x=0$
$x=0$	$x<0$	$x>0$

$T[0]$

Fig.2 Example of an inverted function table

图 2 反转函数表的例子

图 2 中描述了 3 个 *Grid*,分别是 $T[0],T[1]$和 $T[2],T[0]$和 $T[2]$描述了参数的各种取值情况,$T[1]$描述了函数 $f(x,y)$的在各种参数下的具体取值.该反转函数表描述的函数为

$$f(x,y)=\begin{cases}x+y, & (x<0\wedge y<0)\vee(x>0\wedge y=0)\vee(x=0\wedge y>0)\\ x-y, & (x=0\wedge y<0)\vee(x<0\wedge y\geqslant 0)\\ x\times y, & (x>0\wedge y<0)\vee(x>0\wedge y>0)\vee(x=0\wedge y=0)\end{cases}.$$

若设反转函数表 T 由 1 个主 *Grid* $T[0]$和 n 个辅助 *Grid* $T[1],T[2],\ldots,T[n]$组成,$T[1]$中包含 m_1 个算术表达式,$T[i](1\leqslant i\leqslant n)$中包含 m_i 个谓词表达式,$T[0]$中包含 $m_1\times m_2\times\ldots\times m_n$ 个谓词表达式,S 为 T 对应指称语义的指引集,则反转函数表类型具有如下语义:

1) S 中指引的维度为 $n(n\geqslant 1)$;
2) T 的指派顺序是先 $T[i](2\leqslant i\leqslant n)$,然后 $T[0]$,最后 $T[1],T[i](2\leqslant i\leqslant n)$的指派间无先后关系;
3) $T[i](2\leqslant i\leqslant n)$是互斥且完备的,$T[0]$是完备的;
4) $T[i](2\leqslant i\leqslant n),T[0]$中的表达式是谓词表达式,$T[1]$中的表达式是算术表达式.

反转函数表 T 的具体指称语义如下:

- $K_T[T]=\{T[2][i_2]\rightarrow T[3][i_3]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_1,i_2,i_3,\ldots,i_n]\rightarrow T[1][i_1]\}|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[k]](2\leqslant k\leqslant n)=K_T[T]\times assign(T[k])=\{T[2][i_2]\rightarrow\ldots\rightarrow T[k-1][i_{k-1}]\rightarrow T[k+1][i_{k+1}]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_1,i_2,\ldots,i_{k-1},a_k,i_{k+1},\ldots,i_n]\rightarrow T[1][i_1]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[0]]=[K_T[T]\times assign(T[2])\times assign(T[3])\times\ldots\times assign(T[n])]\times assign(T[0][a_2,a_3,\ldots,a_n])=\{T[1][a_1]\}$;
- $K_G[T[1]]=[K_T[T]\times assign(T[2])\times assign(T[3])\times\ldots\times assign(T[n])\times assign(T[0][a_2,a_3,\ldots,a_n])]\times assign(T[1]=\{Value\}$;
- $K_{SE}[T[i][j]](2\leqslant i\leqslant n,1\leqslant j\leqslant m_i)=assign(T[i])\rightarrow Bool$;
- $K_{SE}[T[0][i_1,i_2,i_3,\ldots,i_n]](1\leqslant i_j\leqslant m_j,1\leqslant j\leqslant n)=assign(T[2])\times assign(T[3])\times\ldots\times assign(T[n])\times assign(T[0][a_2,a_3,\ldots,a_n])\rightarrow Bool$;
- $K_{SE}[T[1][j]](1\leqslant j\leqslant m_i)=assign(T[2])\times assign(T[3])\times\ldots\times assign(T[n])\times assign(T[0][a_2,a_3,\ldots,a_n])\times assign(T[1])\rightarrow Value$;
- $eval(T)=K_T[T]\times assign(T[2])\times assign(T[3])\times\ldots\times assign(T[n])\times assign(T[0][a_2,a_3,\ldots,a_n])\times assign(T[1])\rightarrow Value$.

$T[i](2\leqslant i\leqslant n)$与 $T[0]$一起给出了函数参数的各种取值情况,$T[i]$的指派情况将 $T[0]$中的参数取值情况缩至 $T[0]$的一个子网格 $T[0][a_2,a_3,\ldots,a_n]$中,$T[0][a_2,a_3,\ldots,a_n]$包含 m_1 个谓词表达式,而 $T[0][a_2,a_3,\ldots,a_n]$的指派情况决定了函数最终的计算结果 $T[1][a_1]$.反转函数表的取值结果 $eval(T)$同样为函数的计算结果.

3.3 向量函数表

向量函数表由 1 个主 *Grid*、1 个变量 *Grid* 和 n–1 个辅助 *Grid* 组成.主 *Grid* 中的表达式为谓词表达式,变量 *Grid* 中的表达式为变量,辅助 *Grid* 中的表达式为谓词表达式.向量函数表描述的是一个函数,其定义域是一组赋值,值域是 n 元组的集合,变量 *Grid* 用于给出一组赋值,辅助 *Grid* 为这组赋值索引赋值被接受应满足的条件,在主 *Grid* 中找对应的列,主 *Grid* 判断这组赋值是否被该函数所接受.

图 3 是向量函数表的例子.

$T[2]$	$T[1]$	$w<0$	$w=0$	$w>0$
x		$x=w$	$x^2=4$	$x^2=w$
v		$v^2=x+2$	$v=x+2$	$v=\lvert x\rvert+2$
z		$z^2=x^2+v^2+w$	$z^2=x^2+y$	$z=5$
			$T[0]$	

Fig.3 Example of a tuple table

图 3 向量函数表的例子

图 3 中描述了 3 个 *Grid*,分别是主 *Grid* $T[0]$、辅助 *Grid* $T[1]$和变量 *Grid* $T[2]$.$T[2]$给出了函数的定义域,$T[1]$

和 $T[0]$给出了赋值被接受需满足的条件.

若设向量函数表 T 由主 *Grid* $T[0]$、辅助 *Grid* $T[i](1\leqslant i\leqslant n-1)$和变量 *Grid* $T[n]$组成,$T[i](1\leqslant i\leqslant n-1)$中包含 m_i 个谓词表达式,$T[i]$中包含 m_n 个变量,$T[0]$中包含 $m_1\times m_2\times...\times m_n$ 个谓词表达式,S 为 T 对应指称语义的指引集,则向量函数表类型具有如下语义:

1) S 中指引的维度为 $n(n\geqslant 1)$;
2) T 的指派顺序是先 $T[n]$,之后 $T[i](1\leqslant i\leqslant n-1)$,最后 $T[0]$,$T[i](1\leqslant i\leqslant n-1)$的指派间无先后关系;
3) $T[i](1\leqslant i\leqslant n-1)$是互斥且完备的;
4) $T[i](0\leqslant i\leqslant n-1)$中的表达式是谓词表达式,$T[n]$中的表达式是变量.

向量函数表 T 的具体指称语义如下:

- $K_T[T]=\{T[1][i_1]\to T[2][i_2]\to\ldots\to T[n-1][i_{n-1}]\to T[0][i_1,i_2,\ldots,i_{n-1},1]\wedge T[0][i_1,i_2,\ldots,i_{n-1},2]\wedge...\wedge T[0][i_1,i_2,\ldots,i_{n-1},m_n]\to Bool|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[k]](1\leqslant k\leqslant n-1)=K_T[T]\times assign(T[k])=\{T[1][i_1]\to\ldots\to T[k-1][i_{k-1}]\to T[k+1][i_{k+1}]\to\ldots\to T[n-1][i_{n-1}]\to T[0][i_1,\ldots,i_{k-1},a_k,i_{k+1},\ldots,i_{n-1},1]\wedge T[0][i_1,\ldots,i_{k-1},a_k,i_{k+1},\ldots,i_{n-1},2]\wedge...\wedge T[0][i_1,\ldots,i_{k-1},a_k,i_{k+1},\ldots,i_{n-1},m_n]\to Bool|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[n]]=[K_T[T]\times assign(T[1])\times assign(T[2])\times...\times assign(T[n-1])]\times assign(T[n])=\{T[0][a_1,a_2,\ldots,a_{n-1},1]\wedge T[0][a_1,a_2,\ldots,a_{n-1},2]\wedge...\wedge T[0][a_1,a_2,\ldots,a_{n-1},m_n]\}$;
- $K_G[T[0]]=[K_T[T]\times assign(T[1])\times assign(T[2])\times...\times assign(T[n])]\times assign(T[0][a_1,a_2,\ldots,a_{n-1}])=\{Bool\}$;
- $K_{SE}[T[i][j]](1\leqslant i\leqslant n-1,1\leqslant j\leqslant m_i)=assign(T[i])\to Bool$;
- $K_{SE}[T[n][j]](1\leqslant j\leqslant m_n)=assign(T[i])\to Value$;
- $K_{SE}[T[0][i_1,i_2,\ldots,i_n]](1\leqslant i_j\leqslant m_i,1\leqslant j\leqslant n)=assign(T[1])\times assign(T[2])\times...\times assign(T[n])\times assign(T[0][a_1,a_2,\ldots,a_{n-1}])\to Bool$;
- $eval(T)=\{assign(T[n])|K_T[T]\times assign(T[1])\times assign(T[2])\times...\times assign(T[n])\}\times assign(T[0][a_1,a_2,\ldots,a_{n-1}])\to\text{true}\}$.

$T[i](1\leqslant i\leqslant n-1)$的指派情况决定了 $T[n]$中的赋值满足哪些 $T[0]$中的哪些条件才能被接受,$T[0]$的指派决定了各种条件下赋值变量被接受的具体条件.由于 $T[i](1\leqslant i\leqslant n-1)$的指派将 $T[n]$中赋值被接受的条件缩减至 $T[0]$的一个字网格 $T[0][a_1,a_2,\ldots,a_{n-1}]$中,因而判断 $T[n]$的赋值能否被接受,只需给出 $T[0][a_1,a_2,\ldots,a_{n-1}]$的指派情况,而不需对整个 $T[0]$进行指派.向量函数表的取值结果 $eval(T)$不再是一个具体的值,而是一个能被 $T[0]$所接受的赋值集合.

3.4 决策表

决策表包含经典决策表和通用决策表两种类型,任何经典决策表都可以经过等价变换转换为通用决策表.经典决策表由 1 个主 *Grid* 和 2 个辅助 *Grid* 组成,主 *Grid* 中的表达式是波尔类型的常量,辅助 *Grid* 中的表达式,一个为变量表达式,一个为谓词表达式;通用决策表由 1 个主 *Grid* 和 1 个辅助 *Grid* 组成,主 *Grid* 中为谓词表达式,辅助 *Grid* 中为算术表达式.图 4 和图 5 分别是同一个决策表的经典决策表和通用决策表.本文将就通用决策表分析决策表的指称语义.

若设通用决策表 T 由主 *Grid* $T[0]$和辅助 *Grid* $T[1]$组成,$T[1]$包含 m 个算术表达式,$T[0]$中包含 $n\times m$(n 为等价经典决策表中谓词表达式的数量)个谓词表达式,S 为 T 对应指称语义的指引集,则通用决策表类型具有如下语义:

1) S 中指引的维度为 $n(n\geqslant 1)$;
2) T 的指派顺序是先 $T[0][i](1\leqslant i\leqslant n)$,再 $T[1]$,$T[0][i]$的指派间无先后关系;
3) $T[0][i]$是完备的;
4) $T[0]$中的表达式是谓词表达式,$T[1]$中的表达式为算术表达式.

通用决策表 T 的具体指称语义如下:

- $K_T[T]=\{T[0][1,i_0]\rightarrow T[0][2,i_1]\rightarrow\ldots\rightarrow T[0][n,i_n]\rightarrow T[1][k]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n),1\leqslant k\leqslant m\}$;
- $K_G[T[0]]=K_T[T]\times assign(T[0][1])\times assign(T[0][2])\times\ldots\times assign(T[0][n])=\{T[1][a_1]\}$;
- $K_G[T[1]]=[K_T[T]\times assign(T[0][1])\times assign(T[0][2])\times\ldots\times assign(T[0][n])]\times assign(T[1])=\{Value\}$;
- $K_{SE}[T[0][i,j]](1\leqslant i\leqslant n,1\leqslant j\leqslant m)=assign(T[0][i])\rightarrow Bool$;
- $K_{SE}[T[1][j]](1\leqslant j\leqslant m)=assign(T[0])\times assign(T[1])\rightarrow Value$;
- $eval(T)=K_T[T]\times assign(T[0][1])\times assign(T[0][2])\times\ldots\times assign(T[0][n])\times assign(T[1])\rightarrow Value$.

决策表描述了一件事务的决策过程,$T[0]$给出了影响该事务的各种因素,$T[1]$给出了各种条件下,该事务实施的各种操作.$T[0]$指派情况决定了 $T[1]$中操作的选择情况,$T[1]$的指派情况决定整个事务的决策结果.决策表的取值结果 $eval(T)$就是事务的决策结果.

	$T[1]$	$x+1$	$x+1$	$y+2$	$y+2$	$x+y$	$x+y$	$x-y$	$y-x$
$T[2]$									
$V_1>1$		Y	Y	Y	N	Y	N	N	N
$V_2>5$		N	N	Y	Y	Y	Y	N	N
$V_1+3>5$		Y	N	Y	Y	N	N	Y	N
	$T[0]$								

Fig.4 Example of a classic decision table

图 4 经典决策表的例子

$T[1]$	$x+1$	$y+2$	$x+y$	$x-y$	$y-x$
$T[0]$	$V_1>1$	true	true	$\neg(V_1>1)$	$\neg(V_1>1)$
	$\neg(V_2>5)$	$V_2>5$	$V_2>5$	$\neg(V_2>5$	$\neg(V_2>5$
	true	$V_1+3>5$	$\neg(V_1+3>5)$	$V_1+3>5$	$\neg(V_1+3>5)$

Fig.5 Example of a generalized decision table

图 5 通用决策表的例子

4 扩展类型 Tabular 表达式的语义

上一节对一些传统类型 Tabular 表达式的语义进行了描述,但通用模型不仅仅包含传统类型的 10 种表,还包含无穷种自定义表类型.本节将对一些根据通用模型自定义的 Tabular 表达式进行语义描述,进一步说明本文方法的适用性与一般性.

4.1 冗余信息表

冗余信息表中包含 1 个主 *Grid* 和 4 个辅助 *Grid*,主 *Grid* $T[0]$中存放的表达式为常量,辅助 *Grid* $T[i](1\leqslant i\leqslant 4)$中存放的是谓词表达式,其中,$T[1]$和 $T[3]$是对同一组谓词公式的不同描述,$T[2]$和 $T[4]$也是对同一组谓词公式的不同描述.图 6 是一个冗余信息表的例子,该表描述的是服装店对一款热销上衣的尺码推荐信息,$T[1]$和$T[3]$分别用千克和磅两种单位描述了体重信息,$T[2]$和 $T[4]$分别用英寸和厘米描述了身高信息,$T[3]$根据顾客身高和体重的信息对衣服的尺码进行推荐.

若设一个冗余信息表 T 由主 *Grid* $T[0]$和辅助 *Grid* $T[i](1\leqslant i\leqslant 4)$组成,$T[1]$和 $T[3]$描述同一组谓词表达式,$T[2]$和 $T[4]$描述同一组谓词表达式,$T[1]$和 $T[3]$中包含 m 个谓词表达式,$T[2]$和 $T[4]$中包含 n 个谓词表达式,$T[0]$中包含 $n\times m$ 个常量,S 为 T 对应指称语义的指引集,则冗余信息表类型具有如下语义:

1) S 中指引的维度为 2;
2) T 的指派是任意 2 个描述不同组谓词表达式的辅助 *Grid* 的指派,二者无先后顺序;
3) $T[i](1\leqslant i\leqslant 4)$是互斥的;
4) $T[0]$中的表达式为常量,$T[i](1\leqslant i\leqslant 4)$中的表达式为谓词表达式.

冗余信息表 T 的具体指称语义如下:

- $K_T[T]=\{T[1][i]\vee T[3][i]\rightarrow T[2][j]\vee T[4][j]\rightarrow T[0][i,j]|1\leqslant i\leqslant m,1\leqslant j\leqslant n\}$;
- $K_G[T[0]]=[K_T[T]\times assign(T[1])\times assign(T[2])]+[K_T[T]\times assign(T[1])\times assign(T[4])]+$
  $[K_T[T]\times assign(T[3])\times assign(T[2])]+[K_T[T]\times assign(T[3])\times assign(T[4])]=\{Constant\}$;
- $K_G[T[i]](i=1\vee i=3)=[K_T[T]\times assign(T[2])]+[K_T[T]\times assign(T[4])]=\{T[2][j]\vee T[4][j]\rightarrow T[0][i,j]|1\leqslant j\leqslant n\}$;
- $K_G[T[i]](i=2\vee i=4)=[K_T[T]\times assign(T[1])]+[K_T[T]\times assign(T[3])]=\{T[1][j]\vee T[3][j]\rightarrow T[0][i,j]|1\leqslant j\leqslant m\}$;
- $K_{SE}[T[i][j]](i=2\vee i=4,1\leqslant j\leqslant m)=assign(T[i])\rightarrow Bool$;
- $K_{SE}[T[i][j]](i=1\vee i=3,1\leqslant j\leqslant n)=assign(T[i])\rightarrow Bool$;
- $K_{SE}[T[0][i,j]](1\leqslant i\leqslant n,1\leqslant j\leqslant m)=Constant$;
- $eval(T)=[K_T[T]\times assign(T[1])\times assign(T[2])]+[K_T[T]\times assign(T[1])\times assign(T[4])]+$
  $[K_T[T]\times assign(T[3])\times assign(T[2])]+[K_T[T]\times assign(T[3])\times assign(T[4])]\rightarrow Constant$.

冗余信息表中描述冗余信息(同一组谓词表达式的不同描述)的 *Grid* 不仅仅可以是 2 个,也可以是多个,推荐结果的前提条件也可以是多个.事务被执行的前提是满足多个条件中的一个,这种情况适合用冗余信息表来描述.

$T[4]$ \ $T[1]$	40≤kg<50	50≤kg<55	55≤kg<60	60≤kg<65	65≤kg<70	70≤kg<75	75≤kg<80	80≤kg<85	$T[2]$
155≤cm<165	S	S	L	XL	XL	XL	XL	XXL	61≤i<65
166≤cm<170	S	L	L	XL	XL	XL	XXL	XXL	65≤i<67
170≤cm<175	M	L	L	XL	XL	XXL	XXL	XXXL	67≤i<69
175≤cm<180	M	L	XL	XL	XXL	XXL	XXL	XXXL	69≤i<71
180≤cm<185	L	XL	XL	XXL	XXL	XXL	XXXL	XXXL	71≤i<73
185≤cm<190	L	XL	XXL	XXL	XXXL	XXXL	XXXL		73≤i<75
$T[3]$ \ $T[0]$	88≤p<110	110≤p<121	121≤p<132	132≤p<143	143≤p<154	154≤p<165	165≤p<176	176≤p<187	

Fig.6 Example of a tabular expression with redundant headers

图 6 冗余信息表的例子

4.2 循环表

循环表中包含 1 个主 *Grid* 和 *n* 个辅助 *Grid*,主 *Grid* 中的表达式可以是任意类型,包括 Tabular 表达式,辅助 *Grid* 中的表达式为谓词表达式.当主 *Grid* 中的表达式为算术表达式时,循环表的功能和正规函数表很相似,但正规函数表中辅助 *Grid* 的指派间是相互独立的,而循环表中 $T[i]$的指派会影响 $T[i+1]$的指派.图 7 给出了一个循环表的例子,图中的循环表包含 1 个主 *Grid* $T[0]$和 3 个辅助 *Grid* $T[i](1\leqslant i\leqslant 3)$,$T[1]$的指派影响 $T[2]$的指派,$T[2]$的指派影响 $T[3]$的指派,$T[3]$的指派影响循环表的最终取值结果.

若设一个循环表 T 由 1 个主 *Grid* $T[0]$和 n 个辅助 *Grid* $T[1],T[2],\ldots,T[n]$组成,第 $i(1\leqslant i\leqslant n)$个辅助 *Grid* $T[i]$中包含 m_i 个谓词表达式,*Grid* $T[0]$中包含 m_n 个谓词表达式,$T[i,i-1]$为 $T[i]$中受 $T[i-1,i-2]$的指派情况影响的谓词表达式的集合,S 为 T 对应指称语义的指引集,则循环表类型具有如下语义:

1) S 中指引的维度为 n;
2) T 的指派顺序为 $T[1],T[2],....,T[n]$,最后为 $T[0]$;
3) $T[i](1\leqslant i\leqslant n$ 是互斥的;
4) $T[0]$中的表达式为任意类型,$T[i](1\leqslant i\leqslant n)$中的表达式为谓词表达式.

为了方便说明,下面将就 $T[0]$中的表达式为算术表达式的情况说明循环表 T 的指称语义:

- $K_T[T]=\{T[1][i_1]\rightarrow T[2][i_2]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_n]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[k]](1\leqslant k\leqslant n)=[K_T[T]\times assign(T[1])\times assign(T[2,1])\times\ldots\times assign(T[k-2,k-1])]\times assign(T[k-1,k])$
  $=\{T[k+1][i_{k+1}]\rightarrow\ldots\rightarrow T[n][i_n]\rightarrow T[0][i_n]|1\leqslant i_j\leqslant m_j(1\leqslant j\leqslant n)\}$;
- $K_G[T[0]]=[K_T[T]\times assign(T[1])\times assign(T[2,1])\times\ldots\times assign(T[n,n-1])]\times assign(T[0])=\{Value\}$;
- $K_{SE}[T[i][j]](1\leqslant i\leqslant n,0\leqslant j\leqslant m_i)=assign(T[i])\rightarrow Bool$;
- $K_{SE}[T[0][j]](1\leqslant j\leqslant m_n)=assign(T[1])\times assign(T[2,1])\times\ldots\times assign(T[n,n-1]\times assign(T[0])\rightarrow Value$;
- $eval(T)=K_T[T]\times assign(T[1])\times assign(T[2,1])\times\ldots\times assign(T[n,n-1]\times assign(T[0])\rightarrow Value$.

循环表中,每一个辅助 *Grid* $T[i]$中表达式的取值不仅受自身指派情况的影响,还与前 $i-1$ 个辅助 *Grid* 的指派情况有关.n 个辅助 *Grid* 的指派情况和主 *Grid* 的指派情况共同影响 $T[0]$中表达式的计算结果,进而影响循环表的取值情况.

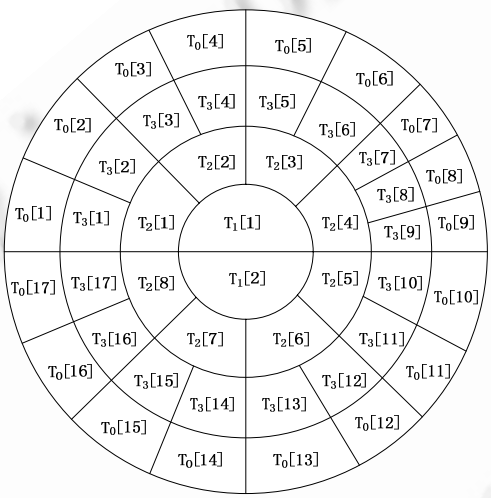

Fig.7 Example of a circular table

图 7 循环表的例子

5 Tabular 表达式语义描述方法的比较

从本质上说,当前Tabular表达式语义模型的研究可以分为以下3个角度:1) 基于函数和关系的联合单独定义每一个表的语义;2) 通过定义如何求值给出表达式的语义;3) 从功能的角度给出 Tabular 表达式中各语法单位的形式语义.只是每种语义模型针对的Tabular表达式模型有所差别,进而导致了语义模型在适用性和扩展性上有所差别.表 2 给出了当前主流 Tabular 表达式语义描述方法的比较.

Table 2 The comparison of different semantic description methods of tabular expressions

表 2 Tabular 表达式不同语义描述方法的比较

语义描述方法	针对的 TE 模型	统一模型	语义解释角度	局限性
Parnas 早期方法[2−4]	Parnas 早期定义的 10 种表(传统 TE)	否	分别定义,函数或关系的联合	无
Jancki 方法[14,15]	传统 TE 中有限种表	是	定义在一组函数或者关系的项来求值	有
Abraham 方法[16]	传统 TE 中有限种表	是	定义在一组函数或者关系的项来求值	有
Kahl 方法[17]	传统 TE 中有限种表	是	表折叠函数定义如何求值	有
Panas,Jingying 新方法[18]	通用模型(传统 TE 和新定义的类型)	是	通用函数定义如何求值	有
本文方法	通用模型(传统 TE 和新定义的类型)	是	为每个语法单位给出语义域和相应的语义指派方程	无

最初的 Tabular 表达式只有 10 种表,没有统一的模型对其进行描述,只能通过单独定义函数或关系的联合对每个表的语义进行描述,因而该方法的形式化程度较低.Jancki,Abraham 和 Kahl 等人分别基于早期的 10 种表给出了各自统一的形式化模型,并都从如何求值的角度对 Tabular 表达式的语义进行了研究.Jancki 和 Abraham 的模型很相近,因此后来有学者将二者的模型统称为 J-A 模型.J-A 模型由原始结构、谓词规则、关系规则、单元连接图、单元到表达式的映射以及复合操作组成,该模型根据单元连接图、表谓词规则和表关系规则为表中每一个元素定义一种关系,利用所有元素对应关系的复合来反映 Tabular 表达式的语义;在 Kahl 模型中,Tabular 表达式是由表和一组语义规则构成,表可以通过增加标题和表连接的操作,由单元组成的表逐步构造而成,而语义规则体现了表达式的求值结构,该模型的语义解释是通过应用结构化归约把表格映射到值.

与本文方法相比,这 3 种方法的主要缺陷在于 Tabular 表达式形式化模型本身的缺陷.从本质上讲,这 3 种方法都将 Tabular 表达式的结构定义成一个由主网格和若干标题网格组成的多维矩形结构,并要求主网格索引集合是其他标题网格索引集合的笛卡尔乘积,索引集合都是连续整数集合,而且它们对形式语义的定义也依赖于这种物理表示结构.本文方法针对 Tabular 表达式通用模型,该模型不依赖于特定的物理表示结构,对于同一个 Tabular 表达式定义,可以由不同的物理布局形式,具有良好的扩展性,可以定义新的表类型,如上文提到的冗余信息表、循环表等,都是其他模型无法表示的.

Tabular 表达式通用模型由 Parnas 教授和金英教授共同给出,他们从如何求值的角度对 Tabular 表达式通用模型的操作语义进行了研究.本文从指称语义的角度分析了 Tabular 表达式通用模型的形式语义,从理论上讲,操作语义与指称语义的描述能力是等价的,但是 Tabular 表达式通用函数的操作语义描述需要针对特定的类型,Tabular 表达式通用模型中包含无穷种类型,无法一一穷举,这为相关应用工具的开发造成了很大困难,每当需要在工具中增加新类型时,都会伴随着大量的改动.本文的研究依赖于 Tabular 表达式通用模型的形式文法,可以随时根据应用创建满足 Tabular 表达式通用模型形式文法的任意类型,使得 Tabular 表达式相关分析验证工具的开发不再受类型的限制,为 Tabular 表达式通用模型的进一步推广奠定了良好的基础.

6 结束语

本文基于 Tabular 表达式通用模型给出了其形式文法,并首次从指称语义的角度 Tabular 表达式的语义描述方法进行了研究.分别从传统类型 Tabular 表达式和新类型 Tabular 表达式两个角度对本文的语义描述方法进行了验证,研究分析表明:本文的方法不再受 Tabular 表达式模型和类型的限制,打破了现有方法的局限性,为 Tabular 表达式通用模型的推广和相关分析验证工具的开发奠定了良好的基础.

References:

[1] Jin Z, Liu L, Jin Y. Software Requirements Engineering: Principles and Methods. Beijing: Science Press, 2008. 282−299 (in Chinese).

[2] Heninger KL, Kallander J, Parnas DL, Shore JE. Software requirements for the A-7E aircraft. NRL Report 9194, U.S. Naval Research Lab., 1978.

[3] Heninger KL. Specifying software requirements for complex systems: New techniques and their applications. IEEE Trans. on Software Engineering, 1980,SE-6(1):2−13. [doi: 10.1109/TSE.1980.230208]

[4] Parnas DL. Software aspects of strategic defense systems. Communications of the ACM, 1985,28(12):1326−1335. [doi: 10.1145/214956.214961]

[5] Parnas DL, Asmis GLK, Madey J. Assessment of safety-critical software in nuclear power plants. Nuclear Safety, 1991,32(2):189−198.

[6] Parnas DL. Inspection of safety critical software using function tables. In: Proc. of the IFIP World Congress. Vol.3. 1994. 270−277.

[7] Parnas DL, Madey J. Functional documentation for computer systems engineering. Science of Computer Programming, 1995, 25(1):41−61. [doi: 10.1016/0167-6423(95)96871-J]

[8] Parnas DL, Madey J, Iglewski M. Precise documentation of well-structured programs. IEEE Trans. on Software Engineering, 1994,20(12):948−976. [doi: 10.1109/32.368133]

[9] Parnas DL. Document based rational software development. Knowledge-Based Systems, 2009,22(3):132−141. [doi: 10.1016/j.knosys.2008.11.001]

[10] Degiovanni R, Ponzio P, Aguirre N, Frias M. Abstraction based automated test generation from formal tabular requirements specifications. LNCS 6706: Tests and Proofs, 2011. 84−101. [doi: 10.1007/978-3-642-21768-5_8]

[11] Feng X, Parnas DL, Tse TH. Fault propagation in Tabular expression-based specifications. In: Proc. of the 32nd Annual IEEE Int'l Computer Software and Applications Conf. 2008. 180−183. [doi: 10.1109/COMPSAC.2008.115]

[12] Herrmannsdörfer M, Konrad S, Berenbach B. Tabular notations for state machine-based specifications. Crosstalk, 2008,21(3): 18−23.

[13] Peters DK, Lawford M, Widemann BT. An IDE for software development using Tabular expressions. In: Proc. of the 2007 Conf. of the Center for Advanced Studies on Collaborative Research. 2007. 248−251. [doi: 10.1145/1321211.1321238]

[14] Janicki R. Towards a formal semantics of Parnas tables. In: Proc. of the 17th Int'l Conf. on Software Engineering. 1995. 231−240. [doi: 10.1145/225014.225036]

[15] Janicki R, Khedri R. On a formal semantics of Tabular expressions. Science of Computer Programming, 2001,39(2-3):189−213. [doi: 10.1016/S0167-6423(00)00004-6]

[16] Abraham RF. Evaluating Generalized Tabular Expressions. McMaster University Canada, 1997.

[17] Kahl W. Compositional syntax and semantics of tables. Technical Report, SQRL 15, Software Quality Research Laboratory, McMaster University Canada, 2003.

[18] Jin Y, Parnas DJ. Defining the meaning of Tabular mathematical expressions. Science of Computer Programming, 2010,75(11): 980−1000. [doi: 10.1016/j.scico.2009.12.009]

附中文参考文献:

[1] 金芝,刘璘,金英.软件需求工程:原理和方法.北京:科学出版社,2008.282−299.

张鹏(1986－),男,吉林辉南人,博士生,主要研究领域为软件形式化.
E-mail: zhangpeng0521@mails.jlu.edu.cn

刘华虓(1986－),男,博士,讲师,主要研究领域为软件形式化,软件需求工程.
E-mail: jlulhx@gmail.com

刘磊(1960－),男,教授,博士生导师,主要研究领域为软件形式化,语义网与本体工程.
E-mail: liulei@jlu.edu.cn

金英(1971－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为需求工程,软件形式化.
E-mail: jinying@jlu.edu.cn