运算构造和检验系统FC的设计和实现*

陈海明

(中国科学院软件研究所计算机科学开放研究实验室 北京 100080)

摘要 运算构造和检验系统FC(function constructor)是形式规约获取系统SAQ(specification acquisition)的一个子系统.在SAQ系统中,运算用于表示规约的语义.FC提供了对运算的交互式归纳定义方式和运算的施用,支持运算的联立递归定义.详细介绍FC的功能、结构和实现技术,并讨论了下一步的改进方向.

关键词 上下文无关语言,递归函数,结构归纳,函数计算,形式规约.

中图法分类号 TP311

形式规约(Formal Specification)的获取是软件工程和软件自动化的重要课题.它对于加速形式化方法的推广和应用、提高软件生产率有着重要的作用,近年来在国际上已受到越来越多的关注.形式规约获取系统SAQ(specification acquisition)可以辅助用户在规约库的支持下获取并检验形式规约.[1]该系统面向通用领域,强调获得与真实需要相符合的形式规约,并支持规约的复用,因此在同类研究中颇具新意.SAQ系统基于MLIRF方法(该方法的特征是用到机器学习(Machine Learning)、递归函数(Recursive Function),中间要进行人机交互(Interaction),故名MLIRF)[2],用上下文无关文法CFG(context-free grammar)及相应语言上的递归函数CFRF(recursive function defined on context-free languages)[3]来分别表示规约的语法方面和语义方面.也即用CFG表达问题类,用CFRF描述算法,两者结合形成规约.SAQ系统的一项主要工作是规约语义的定义和检验,完成该任务的是SAQ的子系统之一的运算构造和检验系统FC(function constructor),其中CFRF称为运算.

同数或字上递归函数一样,运算的描述能力已达理论极限,完全胜任作为规约语义的表达工具.由于自变量和函数值具有短语结构,使得运算显示了特有的优越性——许多算法用CFRF描述相当简洁自然,尤其是在算法加工对象具有短语结构时.同时这也使得运算的定义和检验都具有特点.

FC的设计着重于对CFRF的理论、方法和应用的检验,提供对运算的辅助构造定义和施用.运算的交互式归纳构造方式保证了用户定义运算的完备性,也降低了用户定义运算的难度.定义运算的基本方式是联立递归定义,即同时定义多个相关运算.运算的施用使得用户可以检验运算定义的正确性.运算的复用体现在,构造新定义时可引用库中的已有运算.本文介绍FC的设计和实现.第1节叙述FC在设计上的考虑;第2节介绍FC实现的主要方面,即运算定义、运算施用和运算定义语言;第3节给出FC的使用情况和改进方向.

1 系统设计

1.1 系统功能

FC可以辅助用户归纳式地定义运算,也允许用户直接写运算定义.定义运算时,可以同时定义一组相关运算,即联立递归函数.运算的检验是通过施用该运算,供用户观察施用结果是否正确.为了使定义过程更加友好和有效,FC还提供对所定义运算的浏览、修改和合法性检查.FC与规约库管理系统协同工作.定义和施用运算时从库中提取概念(即上下文无关语言,见2.1.1节)和运算的定义,定义好的运算可以存入规约库中.用户定义的运算不一定都存入库中.FC提供文件操作,使用户能够由文件读写运算,并能保存交互定义的对话过程到文件.为了表达运算的定义,我们设计和实现了一个运算定义语言FDL(function definition language).

FC的主要功能如下:(1) 辅助用户构造运算定义,并允许用户直接写运算定义;(2) 通过施用运算来检验运算;

* 本文研究得到国家自然科学基金、国家863高科技项目基金和国家"九五"科技攻关计划基金资助.作者陈海明,1966年生,助理研究员,主要研究领域为计算机软件.

本文通讯联系人,陈海明,北京 100080,中国科学院软件研究所计算机科学开放研究实验室

本文1997-04-24收到原稿,1997-09-19收到修改稿

(3) 浏览、修改运算定义;(4) 对运算定义进行分析和合法性检查;(5) 存储运算定义到规约库中;(6) 文件操作.

1.2 系统结构

FC 主要有界面模块、运算定义模块、运算施用模块、FDL 语言分析模块和底层函数模块几部分. 模块关系如图 1 所示. 图中虚线箭头说明 FC 对规约库的操作是通过与规约库管理系统的协同进行的. 各模块的主要功能如下:

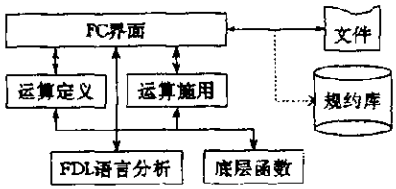

图1

(1) 界面模块:实现 FC 界面操作的全部功能,控制 FC 流程.

(2) 运算定义模块:实现运算交互定义过程,包括提问的形成、回答的记录、形式定义的生成.

(3) 运算施用模块:实现运算施用.

(4) FDL 语言分析模块:实现 FDL 语言的语法分析、运算定义的合法性分析.

(5) 底层函数模块:实现对一些基本数据结构的操作及一些公共函数.

2 系统实现

运算不同于一般的递归函数,其定义和施用都应循着上下文无关语言结构的方向进行. 定义采用结构归纳法,施用采用结构分析方法. 当前,我们只考虑一个归纳元的递归函数. 多重递归可以通过辅助函数实现.

2.1 运算定义

2.1.1 运算定义方法

在 SAQ 系统中,上下文无关语言称为概念,它对应于相关文法中的一个非终极符.

我们沿用文献[3]中项的术语,把以非终极符 X 为左部的产生式右部称为 X 的项,记为 Term(X). 即

$$\mathrm{Term}(X)=\{P\mid X\to P\in \mathbf{P}\},$$

其中 P 为文法的产生式集. X 的项可分为两类:一类只含终极符,称为 X 的基项;另一类含非终极符,称为 X 的复合项. X 的复合项中的非终极符又称为 X 的成分概念.

设 n 元函数

$$f:S_1*\ldots*S_n\to S,$$

对 S_i 进行结构归纳,定义 f 的步骤为:

(1) 基本步. 对 S_1 的基项进行定义,即

对 $f(P_i,y_2,\ldots,y_n)$定义,

其中 $P_i,i=1,\ldots,k$ 是 S_1 的基项,$y_2,\ldots,y_n$ 分别是 $S_2,\ldots,S_n$ 的变元.

(2) 归纳步. 对 S_1 的复合项进行定义,即

对 $f(P_i',y_2,\ldots,y_n)$定义,

其中 $P_i',i=k+1,\ldots,m$ 是 S_1 的复合项,其中的成分概念已替换为相应的变元,$y_2,\ldots,y_n$ 同前.

结构归纳定义函数时,进行归纳的概念 S_1 称作归纳概念.

2.1.2 运算定义过程

FC 支持运算的联立递归定义过程,即同时定义多个相关运算. 已定义的运算(在库中)以及基本运算(见 2.3 节)可以被新定义的运算引用. 联立递归定义的基础是一个运算的定义. 以下先介绍定义一个运算的实现,再介绍多个运算的联立定义过程.

归纳定义一个运算时,对基项之间以及复合项之间的顺序没有要求. 为了使构造定义的过程显得合理,并提高定义的可读性,我们对基项和复合项进行排序. 基项之间以字典序升序排列. 复合项之间按所含成分概念数的升序排列. 用一个栈来记录生成的项. 构造定义一个运算的过程描述如下.

设 $f:S_1*\ldots*S_n\to S$,

(1) 确定归纳概念,设为 S_1.

(2) 求出 Term(S_1).

(3) 将 Term(S_1)记入栈,从栈顶到栈底,项的顺序为,基项在上,复合项在下,基项、复合项顺序如前.

(4) 取出栈顶项 t,与 $S_2,\ldots,S_n$ 一起,将非终极符替换为相应的变元,请用户给出定义.

(5) 栈空时,定义完成.

在实现上,还应允许用户在定义过程中放弃当前定义,重新开始.

定义多个运算的大致过程如下.

最初已知概念的集合 S 和欲定义的运算的集合 F. 对 F 中的运算一个个地进行定义. 定义一个运算时,对于复杂的运算,用户在 FC 的帮助下交互得到运算的定义. 对于用户认为可以直接写出的运算,在 FC 编辑窗直接写出运算的定义. 用户直接定义运算时,需要根据系统提供的运算定义语言 FDL 给出合法的定义. 交互定义时,FC 的提问是非形式的,用户的回答是采用 FDL 语言写的表达式. FC 对用户直接定义的运算及交互定义时每一步的表达式进行合法性检查,对于不合法的定义和表达式,反馈错误给用户,并拒绝接受. 对于在上述两种定义方式中出现的新运算(即运算名不是已定义的,且不在 F 中),FC 要求用户给出其定义域和值域,并将运算名加入 F,如果其定义域和值域中出现了 S 中没有的概念名,则将概念名加入 S. 最后,当 F 中的运算均已定义时,定义过程完成.

2.2　运算施用

运算施用是根据运算的 FDL 定义,计算出运算应用于概念实例的结果. 在 SAQ 系统中,运算施用是用于对运算定义的检验. 我们实现了一个 FDL 解释器,它采取 input-eval-print 的工作方式,使用户能够方便地随时进行运算的检验. 其工作过程如图 2 所示.

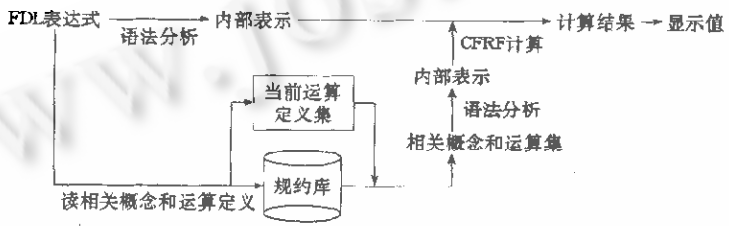

图2

一个 FDL 表达式的相关概念和运算是指该表达式所依赖的概念和运算的闭包. 对于不在当前运算定义集中的运算,需从库中取出定义,所有概念的定义则都从库中取出. 它们经过语法分析后,形成抽象语法树形式的内部表示. 经过对表达式的计算,得到内部表示的计算结果. 然后显示其值. 以下主要介绍 CFRF 的计算方法. 如前所述,CFRF 的计算与结构分析紧密相关. 我们根据所选用的结构分析方法,实现了一个 CFRF 计算算法.

2.2.1　上下文无关语言分析方法

CFRF 要求通用的上下文无关语言分析方法能够处理空串,而不是某个上下文无关语言子类的分析方法,因此,一些有效的分析方法都不合适. 能够满足要求的有 Earley 算法.[4]Earley 算法以构造分析表列的方式找出句子的一个右分析序列,并以产生式的编号表示. 对输入长度为 n 的句子,分析时间为 $O(n^3)$.

2.2.2　CFRF 的计算

设有 n 元函数

$$g: L_1 * \ldots * L_n \to L,$$

归纳概念为首一概念,L_1 由文法 $G_1=(V_N, V_T, X_1, P_1)$ 产生.

对于 $L_1, \ldots, L_n$ 的一组句子 $u_1, \ldots, u_n$,计算 $g(u_1, \ldots, u_n)$ 时,首先对 u_1 按 L_1 的文法进行分析,得到以 P_1 中产生式编号表示的右分析序列 $i_1 i_2 \ldots i_n$,即线性表示的分析树. u_1 的右分析序列有两种情况:

① 只有一条产生式 i_1. 此时产生式 i_1 必定是:$X_1 \to u_1$,从 g 的定义式中找到结构模式为 u_1 的等式(由 g 的结构归纳定义方法,这个等式必然存在),将右端表达式 e 中的变元替换为相应的值. 若表达式 e 中没有除了连接算子以外的函数,则得到计算结果. 若有这样的函数,找到定义,继续进行计算.

② 有多个产生式. 此时产生式 i_1 是:$X_1 \to P$,P 是 X_1 的复合项. 设 P 中含有成分概念 $x_1, \ldots, x_r$,由 $i_2 \ldots i_m$ 容易求出 $x_1, \ldots, x_r$ 各自对应的子树的右分析序列和值 $R_1, \ldots, R_r$. 从 g 的定义式中找到结构模式与 P 对应的等式,将右端表达式 e 中的变元替换为相应的值,并且成分概念的子树右分析序列分别记入表达式 e 中相应变元. 若表达式 e 中没有除了连接算子外的函数,可得计算结果. 若有这样的函数,则找到定义,继续进行计算.

设表达式 e 中有函数 g',如果 g' 的归纳概念对应于 X_1 的成分概念 x_i,即 g' 归纳概念值为 R_i,则可用 x_i 对应的右分析序列继续进行上述计算;如果 g' 的归纳概念不对应于 X_1 的成分概念,则对 g' 的计算需要重新进行结构分析. 对于嵌套的函数(即函数的参数是函数),外层函数也无法利用传下来的分析序列. 只有内层函数才有可能用得着传下来

的分析序列.可见,在计算过程中,可能经常地调用语法分析程序对句子进行分析.另外,基本运算的计算不需对句子进行语法分析.

2.3 运算定义语言

本节简介运算定义语言FDL,有关FDL的详细介绍参见文献[5,6].

2.3.1 FDL语言设计

FDL设计特点有:① 支持CFRF的定义.即支持项、常量、变元、函数名、函数复合、连接算子等的表达.② 表达式最基本的形式是只含常量、变元和函数(包括算子).为了定义的方便性与直观性,增加中缀表示形式,如用"[]"表示连接算子.③ 常量是上下文无关语言的句子,为了使SAQ能表达尽可能广泛的上下文无关语言,常量的字符集应尽量大.④ 为了提高运算定义和施用的效率、方便用户的使用,预定义一些基本概念和基本运算.

一个运算定义由声明和定义两部分组成.声明包括运算声明和变量声明,定义包括一组等式.

一个运算定义的结构为

运算声明
变量声明
定义

等式形式为

函数名(结构模式表)=表达式.

FDL预定义的基本概念有Char(Char的元素是SAQ可用字符,即任何ASCII字符)、String(String的元素是SAQ可用字符的字符串,即任何ASCII字符串,包括空串.SAQ的任何上下文无关语言都是String的子集.因此,可以在String上定义一些通用函数)和Bool(Bool有两个元素:"True"和"False").

FDL还预定义了16个基本运算.

下面是用FDL定义的运算的例子.

例1:概念letterString的文法为

⟨letterString⟩→⟨Letter⟩
⟨letterString⟩→⟨Letter⟩⟨letterString⟩
⟨Letter⟩→$a|\ldots|z$

(1) 取字符串的第1个字符

```
dec   first:letterString->Letter;
var   x0:Letter;
      x1:letterString;
def   first(x0)=x0;
      first(x0[]x1)=x0;
```

(2) 取字符串的第2个字符

```
dec   next:letterString->Letter;
var   x0:Letter;
      x1:letterString;
def   next(x0)=undefined;
      next(x0[]x1)=first(x1);
```

(3) 取字符串的最后一个字符

```
dec   last:letterString->Letter;
var   x0:Letter;
      x1:letterString;
def   last(x0)=x0;
      last(x0[]x1)=last(x1);
```

2.3.2 FDL语言的分析

对FDL语言的分析在FC中多处用到,如构造定义时表达式的分析、施用运算时对一个定义的分析、运算定义合法性检查时对多个定义的分析等.为了以尽量小的代价提高分析程序的灵活性,我们采用了自己编程与Yacc相结合的方法.自行实现了一个符号分析程序,其中表达式部分用Yacc生成.

2.3.3 FDL定义的运算的合法性分析

定义的运算除了进行语法分析之外,还要进行一些上下文关系的分析.主要有:(1) 定义中出现的变元名都应在变元声明中声明;(2) 等式右端引用的变元都应在等式左端出现;(3) 定义中是否出现未知运算名,即未在定义运算集中,也不在规约库中,又不是基本运算的运算;(4) 等式左端的运算名应与运算声明名相同;(5) 定义完整性,即定义是否枚举了归纳概念的项;(6) 归纳一致性,即各个等式的归纳概念应一致.

3 总 结

FC已在Sun工作站上实现,操作系统是Solaris2.4,界面图形工具使用XView3.2.用C语言编程,程序量约15 000行.利用本系统,已经对一些数据结构上的运算、整数的算术运算、初等函数的形式微分等进行了定义和计算.

使用情况表明，CFRF 的理论正确、方法可行，用 CFRF 定义算法自然、简单，有其特有的优越性．一些用其他算法语言表达相当复杂的算法，在这里变得十分简单．

在使用中也发现系统存在一些不足之处．我们实现的运算定义方法是对一个概念进行归纳．通过由简单到复杂的过程，可以构造出复杂的运算．定义能力是够的．但在定义多元运算时，有时定义过程较为繁琐，需要引出多个辅助运算．这是因为，这时运算的定义需要对多个概念的结构进行归纳．因此，允许运算定义同时对多个概念进行归纳，会有助于提高定义的效率，增加定义手段．

函数计算的效率较低，除了多次进行语法分析，使效率降低外，主要的原因还有通用上下文无关语法分析 Earley 算法的效率较低，实现上也有待改进．更深层的原因是 CFRF 的高效计算方法有待研究．

今后的研究方向是：(1) 完善定义手段；(2) 研究计算方法，改进计算效率；(3) 改进实现．

致谢　本文的研究工作得到董韫美院士的指导，在此表示衷心的感谢．文中使用的语法分析程序由张珊岭同志实现；万战勇、陈自明同志使用了本系统，并提出了不少有益的建议，也一并表示谢意．

参考文献

1　董韫美等．形式规约获取系统 SAQ 的设计和实现．见：Collection of SAQ Reports no. 8-16，中国科学院软件研究所计算机科学开放研究实验室报告，ISCAS-LCS-96-1，1996
(Dong Yun-mei *et al*. Design and implementation of specification acquisition system SAQ. In: Collection of SAQ Reports no. 8-16. Technical Report, ISCAS-LCS-96-1. Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, 1996)

2　董韫美．规约获取与复用的 MLIRF 方法．见：中国计算机科学技术新发展(中国计算机学会第 9 次全国学术会议论文集，特邀报告)．1996
(Dong Yun-mei. MLIRF method for specification acquisition and reuse. In: New Advances in Computer Science and Technology of China. Proceedings of the 9th National Conference of China Computer Federation. 1996)

3　董韫美．上下文无关语言上的递归函数．见：Collection of SAQ Reports no. 1-7，中国科学院软件研究所计算机科学开放研究实验室报告，ISCAS-LCS-95-09，1995
(Dong Yun-mei. Recursive functions defined on context-free languages. In: Collection of SAQ Reports no. 1～7. Technical Report, ISCAS-LCS-95-09. Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, 1995)

4　Aho A V, Ullman J D. The Theory of Parsing, Translation, and Compiling. Volume 1: Parsing. Prentice-Hall, Inc., 1972

5　陈海明．运算构造和检验子系统 FC 的设计和实现．见：Collection of SAQ Reports no. 8-16，中国科学院软件研究所计算机科学开放研究实验室报告，ISCAS-LCS-96-1，1996
(Chen Hai-ming. Design and implementation of function construction and checking system FC. In: Collection of SAQ Reports no. 8-16, Technical Report, ISCAS-LCS-96-1. Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, 1996)

6　SAQ 课题组．形式规约获取系统 SAQ 用户手册．中国科学院软件研究所计算机科学开放研究实验室，1995
(SAQ Group. Specification Acquisition System SAQ: User's Manual. Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, 1995)

Design and Implementation of the Function Construction and Checking System FC

CHEN Hai-ming

(*Laboratory of Computer Science　Institute of Software　The Chinese Academy of Sciences　Beijing*　100080)

Abstract　The function construction and checking system FC (function constructor) is a component of SAQ (specification acquisition) which utilizes CFRFs (recursive functions defined on context-free languages) to represent the semantics of a specification. FC provides the interactive and inductive definition, and the evaluation of CFRF, supporting mutually recursive definition process of CFRFs. In this paper, the functions, structure and implementation details of FC are described, the possible improvement is discussed.

Key words　Context-free language, recursive function, structural induction, function evaluation, formal specification.