形式语言的代数模型研究

陈　星[1],汪海涛[1],胡景荣[2]
(1.昆明理工大学 计算机技术应用重点实验室,云南 昆明 650051; 2.昆明市科技情报研究所,云南 昆明 650021)

摘要: 用统一的基础理论定义形式语言的基本概念是实现软件自动化的重要基石,试图用递归论的术语定义串、序列函词、串等价等程序语言中的基本概念,证明了任何上下文无关语言的串实例都与一个半格系统相对应,对自动化建立语言的代数语义模型做出了有益的尝试.

关键词: 形式语言;代数模型;软件工程
中图分类号: TP309　　**文献标识码**: A　　**文章编号**: 1007 - 855X(2007)06 - 0017 - 04

Research on Algebraic Model of Formal Languages

CHEN Xing[1], WANG Hai-tao[1], HU Jing-rong[2]
(1. Yunnan Provincial Key Laboraty of Computer Application, Kunming University of Science and Technology, Kunming 650051, China; 2. Institute of Science and Technology Information of Kunming, Kunming 650021, China)

Abstract: A unified theory of formal languages is the basis for automatic software developing. Some concepts of programming languages, such as string, sequential functions and string equivalence, are defined in this paper by Recursion Theory. It is then proved that strings of any context-free language can be mapped to a semi-lattice structure, which hopefully will be a beneficial attempt.

Key words: formal language; algebraic model; software engineering

0 引 言

提高软件质量,降低开发成本,缩短开发时间一直都是软件工程研究领域的一项重要目标[2].而实现这一目标的最佳途径是形式化软件开发技术,即用形式化规格说明书定义软件的特性和行为,把规格说明书翻译为程序代码,再用模型验证或定理证明的方法最大程度的保证程序代码与规格说明书相一致[3].

对软件开发的问题实现形式化的基础是用形式语言给出问题的描述,再建立这种语言的形式语义.对于代数语义而言,就是给出与这种语言对应的代数系统[4],例如:一般讨论的 Σ基调代数、λ演算系统的 *Scott*域等都是反映形式语言的代数系统[5].自动实现形式语言的代数建模是基于代数语义的软件自动化技术之基石[6],目的是对任何一种上下文无关语言实现自动代数建模的基础理论进行讨论.为了计算机能够自动化的处理这类问题,统一的数学基础是必要的,只有当计算机能够理解这样的数学基础时,这类问题的软件自动化才能从根本上被实现[7].论文使用了递归论的成果对诸多基本概念进行了规范的定义,专门讨论这个问题的文献迄今尚未见到.

1 基础概念

集合的定义方法是必须要澄清的问题,文中的定义集合只允许使用枚举法和构造法.

枚举法:

〈枚举定义〉::=〈集合名〉={〈枚举〉}

〈枚举〉::=〈元素〉|〈元素序〉

收稿日期: 2007 - 05 - 22.**基金项目**:昆明理工大学科学研究基金的资助 (项目编号: 2006 - 57)
第一作者简介:陈星 (1976 -),男,硕士,助理研究员.主要研究方向:软件自动化、形式化技术.
E - mail: cx_kmust@126.com

〈元素序〉::=〈元素〉,〈枚举〉

构造法:

〈构造定义〉::=〈集合名〉={〈元素变量〉|〈命题〉}

其中命题中允许使用可计算谓词,而且必须满足集合论的基本公理[8]:集合可以由比它更早的集合构造,形成一个关系链,不能有循环,也就是自己不能参与到自己的定义中. 例如: $G=F\cup\{x\cdot y\mid x\in F\ \&\ y\in F\}\cup\{x\cdot y\mid x\in F\ \&\ y\in G\}$, G出现在等式的右侧,这是一个循环定义,并不合法.

有一些文献使用如下的文法定义集合: <集合名>⊆<集合名> AND <命题>,试图把要定义的集合描述为另一个集合满足某些条件的子集,这是本文不允许的. 例如对 S的加标集合 F给出描述: $F\subseteq S\times N$ AND $(\forall t_1, t_2)((t_1=(s_1, i)$ AND $t_1\in F\ t_2=(s_2, j)$ AND $t_2\in F$ AND $i=j)\rightarrow s_1=s_2)$,我们认为这是加标集合的定理,但不是定义形式. 进一步说,这类描述是无法直接计算的,因为 $S\times N$ 不是有限集合,无法计算 $S\times N$. 因此,实际计算时,仍然要把它翻译成构造性定义才能进行计算.

关于特殊集合的符号,本文用 N代表自然数集,用 Φ代表空集.

函数的定义是一个值得关注的问题. 文中把函数看作一种特殊的多元关系,因此使用关系的集合表示函数,例如:函数 $F(x)=2*x-1$,在可以用表达式 $F=\{(x, 2*x-1)\}$代替. 作为递归函数,将出现表达式左侧和右侧都出现函数名的情况,例如: $F=\{(0, 0)\}\cup\{(x, y)\mid x\in N\ \&\ x>0\ \&\ (x-1, t)\in F\ \&\ y=2*t\}$. 从定义集合的观点看,这样的表达式从形式上违背了集合论基本公理,但通常情况下,可以用塔斯基定理或其他的归纳技术计算出此表达式的不动点. 因此,在文中允许使用上述递归表达式. 一个多元关系 $M(X_1, X_2, \cdots, X_n, Y)$是函数关系,必须满足一个条件:

$$\forall(x_1, x_2, \cdots, x_n, y_1, y_2)((x_1, x_2, \cdots, x_n, y_1)\in M\ \&\ (x_1, x_2, \cdots, x_n, y_2)\in M\rightarrow y_1=y_2)$$

语言是串所构成的集合,串是字符的序列. 序列在计算机程序语言中是一个自然的概念,但作为形式语义的基础,需要用定义序列函词的语义.

论文用所谓加标集合定义序列,集合 S的加标集合,用集合算子 $S_i(S)$表示, $S_i(S)\subseteq S\times N$ AND $(\forall t_1, t_2)((t_1=(s_1, i)$ AND $t_1\in S_i(S)\ t_2=(s_2, j)$ AND $t_2\in S_i(S)$ AND $i=j)\rightarrow s_1=s_2)$. 用构造法给出加标集合的定义,需要使用如下递归论的概念.

定义 2.1:若 Φ 是一个函数集,迭置子 $\Delta(\Phi)$的结果是 Φ 中函数相互迭置所得到的全体函数集合. 若 $\sum$为值集,则 $\Delta(\Phi, \sum)=\{t\mid t=\psi(\theta_1, \theta_2, \cdots, \theta_n)\ \&\psi\in\Delta(\Phi)\ \&\theta_1, \theta_2, \cdots, \theta_n\in\sum\}$[9].

定义 2.2:设集合 $S=\{(s, i)\mid i\in N\}$,集合 $J(S)=\{j\mid j\in N\ \&\ \forall t(t=(s, i)\ \&\ t\in S\rightarrow j>i)\}$,函数 $idx=\{(\Phi, 0)\}\cup\{(s, k)\mid s\subset S\ \&\ k\in J(S)\ \&\ \forall t(t\in J(S)\rightarrow k\leqq t)\}$. 增元函数 $\psi=\{(s, t, r)\mid s\subset S\ \&\ r=s\cup(t, idx(s))\}$. 设 A为一个集合,单序增元函数 $\Omega=\{((s, A), a, r)\mid s\subset S\ \&\ a\in A\ \&\ r=(\psi(s, a), A-\{a\})\}$.

定义 2.3:设 A为一个集合,加标迭置集 $SItr(A)=\Delta(\{\Omega\}, (\Phi, A)\cup A)$.

定义 2.4:加标总集 $SI(A)=\{a\mid(a, \Phi)\in SItr(A)\}$.

由于集合没有序的概念, $SI(A)$中每一元素都是合法的 $S_i(A)$.

定义 2.5:序列是加标集合与序列函词 <序列名>[<序号>]组成的系统.

序列函词的语法是 <序列名>[<整数>],如果序列名是"Seq",则语义为 $\{(i, \mathrm{Seq}[i])\mid i\in N\ \&\ (i, \mathrm{Seq}[i])\in \mathrm{Seq}\}$,其中 Seq是加标集合.

序列的长度就是其加标集合的元素数目,本文依照惯例用 |Seq|来表示.

定义 2.6:串是字符的序列.

即串是字符集上的加标集合,全体串的集合用 String表示,在本文中用双引号引起的字符代表串,例如: "*abc*"显然任何一个语言 $L\subseteq$ String

一般来说,串的相等要求彼此的加标集合相等. 如果认为某些字符是词法分隔符,则串相等的概念就被扩大了,例如用 * 作为分隔符,则: "*abc* * *cba*"与 "*abc* * * *cba*"也是等价的,尽管它们所对应的加标集合并不相等.

定义 2.7:串上的加分隔函数为 $A\mathrm{Sep}=\{(s, c, i, t)\mid s, t\in \mathrm{String}\ \&\ i\in N\ \&\ i<|s|\ \&\ t[i]=c\ \&\forall j((j\in N$

& $j \geqq i) \rightarrow t[j+1] = s[j]) \}$

加分隔函数不能保证得到的结果与原串等价,所以,我们需要定义等价加分隔函数. 设分隔符集合为 Sep.

定义 2.8:串上的等价加分隔函数 EASep = { (s, c, m, i, t) | m ∈Sep & (s, c, i, t) ∈Asep & s[i] ∈Sep}.

显然, EASep能保证求出的新串与原串等价. 设 I_t为串的恒等函数,考虑迭置子 Δ({EASep, It}, {s, Sep} ∪Sep ∪N)的结果,得到从串 s出发加上分隔符以后,所有情况构成的等价串与 s本身构成的集合.

定义 2.9:串等价由谓词 eq给出, eq = { (s, t, Sep, true) | ∃m (m ∈Δ({EASep, It}, {s, Sep} ∪Sep ∪N) & m ∈Δ({EASep, It}, {t, Sep} ∪Sep ∪N)) } ∪{ (s, t, false) | NOT∃m (m ∈Δ({EASep, It}, {s, Sep} ∪Sep ∪N) & m ∈Δ({EASep, It}, {t, Sep} ∪Sep ∪N)) }.

定义 2.10:子串函数 subChain = { (s, i, j, t) | s∈String & i∈N & i < |s| & t∈String & |t| = j - i + 1 & ∀k(k∈N & k < j - i + 1→t[k] = s[i + k]) }

定义 2.11:后子串函数 lastChain = { (s, i, t) | s∈String & i∈N & i < |s| & t∈String & ∀j(j∈N & j < |s| - i→t[j] = s[i + j]) }

定义 2.12:串的连接函数 linkChain = { (Φ, Φ, Φ) | s∈String} ∪{ (s, Φ, s) | s∈String } ∪{ (s, t, u) | s∈String & t∈String & |u| = |s| + |t| & subChain(u, 0, |s| - 1) = s & subChain(u, |s|, |s| + |t| - 1) = t}. 为了方便起见,以后用 "+" 代替 linkChain,其文法是 <串> + <串>,语义是 linkChain(<串>, <串>).

2 上下文无关语言的代数模型分析

一个上下文无关语言的文法系统,通常使用 BNF范式对其描述. 从抽象文法的角度看,语法成分 = (子成分$_1$, 子成分$_2$, …, 子成分$_n$),只关注语法成分的组成过程,从 BNF范式很容易得到这种语言的抽象文法. 如何用 BNF范式描述一个上下文无关语言的文法[10],已经有众多的文献、专著讨论它,本文不予赘述. 设描述一个语言的文法系统中,所有 BNF范式的语法名的集合是 UG

研究某种语言的代数语义的基础是这种语言的抽象文法,我们使用语法函数 G来描述语法对象与串的关系. G = { (L, g, s) },其中 L一个串, g是语法成分名, s是 L中语法成分所对应的子串, G是一个 L × g →s的二元函数.

定义 3.1:类子链 TypeChain为 UG中元素构成的序列.

定义 3.2:增强语法函数 H = { (u, Φ, u) | u∈String } ∪{ (u, tc, s) | u∈String & tc∈TypeChain & |tc| = 1 & (u, tc[0], s) ∈G} ∪{ (u, tc, s) | u∈String & tc∈TypeChain & CL(tc) > 1 & (G(u, tc[0]), lastChain(tc, 1), s) ∈H }.

H是一个递归函数的定义,根据塔斯基定理给出其最小不动点,可避免等式右侧也出现 H的形式,但由于篇幅所限,本文不再赘述.

定义 3.3:设代数系统 L(@) = { a | a = (H(u, c), c) & c∈TypeChain},

(∀a, b) (a∈L & b∈L & a@b = c) ↔

(c = ∈L & (H(u, l_c), l_c) = c & (∃l_1) ((H(u, l_c + l_1), l_c + l_1) = a) & (∃l_2) ((H(u, l_c + l_2), l_c + l_2) = b) & (∀c_1, l_3, l_4, l_b) ((H(u, l_c + l_b), l_c + l_b) = c_1 & (H(u, l_c + l_b + l_3), l_c + l_b + l_3) = a & (H(u, l_c + l_b + l_4), l_c + l_b + l_4) = b →c = c_1))).

定理 3.1:代数系统 L(@)是一个半格.

证明:代数系统 L(@)需要满足 3个条件: a@a = a, a@b = b@a, a@b@c = a@(b@c)

①证明 a@a = a

(1) c = a@a　　(附加前提)

(2) c = ∈L & (H(u, l_c), l_c) = c & (∃l_1) ((H(u, l_c + l_1), l_c + l_1) = a) & (∀c_1, l_3, l_b) ((H(u, l_c + l_b), l_c + l_b) = c_1 & (H(u, l_c + l_b + l_3), l_c + l_b + l_3) = a→c = c_1))　　(定义 3.2)

(3) (∀c_1, l_3, l_b) ((H(u, l_c + l_b), l_c + l_b) = c_1 & (H(u, l_c + l_b + l_3), l_c + l_b + l_3) = a→c = c_1)), (2)

(4) $\mathbf{H}(u, l_a), l_a) = a$ (附加前提)

(5) $(\mathbf{H}(u, l_a + \phi), l_a + \phi) = a \ \&\ (\mathbf{H}(u, l_a + \phi + \phi), l_a + \phi + \phi) = a$, (4)

(6) $c_1 = a \ \&\ l_3 = \phi \ \&\ l_0 = \phi \ \&\ (\mathbf{H}(u, l_c + l_0), l_c + l_0) = c_1 \ \&\ (\mathbf{H}(u, l_c + l_0 + l_3), l_c + l_0 + l_3) = a$, (5)

(7) $c = c_1 = a$, (3)

(8) $a = a@a$, (7) (1)

②证明 $a@b = b@a$

(1) $c = a@b$ (附加前提)

(2) $(c = \in L \ \&\ (\mathbf{H}(u, l_c), l_c) = c \ \&\ (\exists l_1)((\mathbf{H}(u, l_c + l_1), l_c + l_1) = a) \ \&\ (\exists l_2)((\mathbf{H}(u, l_c + l_2), l_c + l_2) = b) \ \&\ (\forall c_1, l_3, l_4, l_0)((\mathbf{H}(u, l_c + l_0), l_c + l_0) = c_1 \ \&\ (\mathbf{H}(u, l_c + l_0 + l_3), l_c + l_0 + l_3) = a \ \&\ (\mathbf{H}(u, l_c + l_0 + l_4), l_c + l_0 + l_4) = b \rightarrow c = c_1)))$ (定义 3.2)

(3) $d = b@a$ (附加前提)

(4) $(d = \in L \ \&\ (\mathbf{H}(u, l_d), l_d) = d \ \&\ (\exists l_1)((\mathbf{H}(u, l_d + l_1), l_d + l_1) = a) \ \&\ (\exists l_2)((\mathbf{H}(u, l_d + l_2), l_d + l_2) = b) \ \&\ (\forall d_1, l_3, l_4, l_0)((\mathbf{H}(u, l_d + l_0), l_d + l_0) = d_1 \ \&\ (\mathbf{H}(u, l_d + l_0 + l_3), l_d + l_0 + l_3) = a \ \&\ (\mathbf{H}(u, l_d + l_0 + l_4), l_d + l_0 + l_4) = b \rightarrow d = d_1)))$ (定义 3.2)

(5) $(\mathbf{H}(u, l_c + x), l_c + x) = a$ (2)

(6) $(\mathbf{H}(u, l_d + y), l_d + y) = a$ (4)

(7) $l_c + x = l_d + y$ (5) (6)

(8) $(\mathbf{H}(u, l_c + s), l_c + s) = b$ (2)

(9) $(\mathbf{H}(u, l_d + t), l_d + t) = b$ (4)

(10) $l_c + s = l_d + t$ (8) (9)

(11) $l_c + i = l_d$ (附加前提)

(12) $(\forall c_1, l_3, l_4, l_0)((\mathbf{H}(u, l_c + l_0), l_c + l_0) = c_1 \ \&\ (\mathbf{H}(u, l_c + l_0 + l_3), l_c + l_0 + l_3) = a \ \&\ (\mathbf{H}(u, l_c + l_0 + l_4), l_c + l_0 + l_4) = b \rightarrow c = c_1))$ (2)

(13) $(\mathbf{H}(u, l_c + i), l_c + i) = d \ \&\ (\mathbf{H}(u, l_c + i + y), l_c + i + y) = a \ \&\ (\mathbf{H}(u, l_c + i + t), l_c + i + t) = b$ (4) (11) (5) (9)

(14) $c_1 = d \ \&\ l_0 = i \ \&\ l_3 = y \ \&\ l_4 = t \rightarrow (\mathbf{H}(u, l_c + l_0), l_c + l_0) = c_1 \ \&\ (\mathbf{H}(u, l_c + l_0 + l_3), l_c + l_0 + l_3) = a \ \&\ (\mathbf{H}(u, l_c + l_0 + l_4), l_c + l_0 + l_4) = b \rightarrow c = c_1)$ (13)

(15) $c = d$ (14) (12)

同理,若 $l_d + i = l_c$ 也能得到 $c = d$,因此 $a@b = b@a$

由于③$a@b@c = a@(b@c)$的证明需要更多篇幅,证明技术与上文相同,只好将其略去.

定理3.1说明,按照定义3.3实现的算法,都能自动的得到一个半格系统,这是自动建立语义模型的基础.如果把语法分隔符集考虑进来,就可以把"="的含义考虑为串等价谓词 eq,这就把定理3.1的作用向实用推进了一大步.

3 结论和展望

1)给出了为一个上下文无关语言的串建立半格这种代数系统的方法,对自动化建立这类语言的代数模型进行了初步探索,自动化的建立形式语言的代数模型是基于代数语义的形式化软件开发技术的前提之一.

(下转第24页)

示, M20 ON水位下液位指示,M21 ON电机过热指示,水位上液位触点闭合 X0 ON,水位下液位触点闭合 X1 ON,过热继电器动作 X2 ON. 当 M0断开,M32闭合, X0断开, X2断开时, Y0吸合 (电机手动启动). 到达上液位或是电机发生过热故障时, Y0断开 (电机停止). 当 M0吸合时, X0断开, X2断开, X1吸合时, Y0吸合 (电机自动启动). 当水位离开下液位时, X1断开. 此时通过与 X1并联的 Y0常开触点构成通路,称之为自锁. 到达上液位或是电机发生过热故障时, Y0断开 (电机停止). 在水位又一次回到下液位时, Y0吸合 (电机启动). Y0吸合的同时,M17吸合. VB采集到 M17状态的变化,相应改变控制界面上的电机运行状态指示变化. 上液位到达时, X0吸合, X0的常开触点控制 M19吸合. VB采集到 M19状态的变化,相应改变控制界面上的液位指示变化.

3 结束语

利用 VB设计通信程序,实现了计算机对 PLC的监控. 该设计阐述的实时控制方案成功运用在水位控制系统中,可对现场电动机进行启、停控制,对液位进行测控. 人机交互性好,可操作性强.

参考文献:

[1] 三菱公司. FX通信用户手册 [Z]. 2001.
[2] 杜秋华,康慧芳. 可视化编程应用 [M]. 北京:人民邮电出版社, 2004.
[3] 周海涛. 用 VB6. 0实现三菱 PLC与微机的通信 [M]. 制造业自动化, 2002.
[4] 三菱公司. 三菱微型可编程控制器编程手册 [Z]. 1998.
[5] PETROUTSOS E. Visual Basic 6从入门到精通 [M]. 邱仲潘译. 北京:电子工业出版社, 2002.
[6] 常斗南. 可编程控制原理应用实验 [M]. 北京:机械工业出版社, 2003.

(上接第 20页)

2)用递归论的概念和术语给出了若干核心概念的精确定义,阐述了这些问题的可计算性本质,也说明了一个问题:实现一个非实验性的基于代数语义的自动化开发工具,使其接受递归论的概念和公理可能是必要的.

3)证明了任何一种上下文无关语言的串实例,都与一个半格系统相对应,这是讨论这种语言的代数语义的基础,对上下文无关语言的代数建模自动化作出了有益的探索.

参考文献:

[1] Ravi Sethi,裘宗燕. 程序设计语言概念和结构 [M]. 2版. 北京:机械工业出版社, 2002.
[2] Roges S Pressman. Software Engineering[M]. McGraw - Hill, New York, 1999.
[3] Stacy J. Prowell,贲可荣,等译. 净室软件工程:技术与过程 [M]. 北京:电子工业出版社, 2001.
[4] 陆汝钤. 计算机语言的形式语义 [M]. 北京:科学出版社, 1992.
[5] 石纯一,王家嵌. 数理逻辑与集合论 [M]. 北京:清华大学出版社, 2000.
[6] 徐家福. 软件自动化 [M]. 北京:清华大学出版社, 1994.
[7] 郑红军,张乃孝. Garment中的归约语义 [J]. 计算机研究与发展, 1998, 35(6): 486 - 490.
[8] Glynn Winskel,宋国新. 程序设计语言的形式语义 [M]. 北京:机械工业出版社, 2005.
[9] 莫绍揆. 递归论 [M]. 北京:科学出版社, 1983.
[10] 林茨. 形式语言与自动机导论 [M]. 北京:机械工业出版社, 2004.