形式化开发非递归 Koch 曲线算法

刘润杰 申金媛 穆维新
(郑州大学信息工程学院 郑州 450001)

摘 要 形式化方法是构建可信软件的重要途径。Koch 曲线是典型的分形图形。基于形式化方法 PAR 及循环不变式开发策略,开发了 Koch 曲线非递归算法,并对其进行了形式化的正确性证明。在得到求解 Koch 曲线算法的循环不变式的同时,直接得到易读、高效且可靠的非递归算法。对使用形式化方法及循环不变式开发策略开发分形程序非递归算法作了较深入的实践和探讨。

关键词 Koch 曲线,形式化方法,非递归,PAR 方法,循环不变式
中图法分类号 TP311.1 **文献标识码** A

Formal Development of Non-recursive Algorithm for Koch Curve

LIU Run-jie SHEN Jin-yuan MU Wei-xin
(Schools of Information Engineering, Zhengzhou University, Zhengzhou 450001, China)

Abstract Formal method is an important approach for construction of the trustworthy software. Koch curve is one of the typical fractals. A non-recursive algorithmic program of Koch curve was dveloped, employing PAR method and the strategy of developing loop invariant and the algorithm was verified formally. This paper achieved loop invariant of Koch curve with readable, efficient and reliable non-recursive algorithm finally. The paper contributed to developing non-recursive algorithm using formal method and new strategy of developing loop invariant.

Keywords Koch curve, Formal method, Non-recursive, PAR method, Loop invariant

分形是描述自然界和非线性系统中不光滑和不规则几何形体的有力工具[1,2]。分形算法在保密通信[3,4]、图像压缩[5,6]、航天器控制[7]等领域有着较为广泛并深入的应用。在这些应用中对算法的正确性和可靠性要求十分严格,开发正确可靠的分形算法程序是非常重要的。

Koch 曲线是经典的分形图形,它的算法常常使用递归的方法来实现。虽然递归算法程序具有结构清晰、易用数学归纳法证明其正确性等许多优点,但递归程序的执行过程中存在大量参数的传递和额外空间的分配,具有程序执行效率低、空间的耗费大的缺点。而对于某些特殊要求的程序,如系统核心程序,它们的效率问题至关重要,人们更愿意使用非递归算法来解决递归问题。

PAR 方法是一种建立在程序规约和归纳断言方法基础上的程序设计方法,具有严格的数学基础,是将程序开发与程序正确证明结合的一种形式化的开发方法[8]。本文基于形式化方法 PAR[9],使用循环不变式开发策略中的递归定义技术[10],在得到求解 Koch 曲线的循环不变式的同时,可直接得到清晰简短、可读性好的非递归算法程序。同时基于产生的循环不变式可对算法程序进行形式化验证,从而保证算法程序的正确性。文中所使用的技术具有通用性,可望发展成开发一系列分形递归问题非递归算法的方法。

本文详细阐述了 Koch 曲线非递归算法程序的开发过程,形式化证明了该算法程序的正确性,并对其效率进行了分析,最后作了总结。

1 开发 Koch 曲线问题非递归算法

1.1 Koch 曲线问题描述

从一条直线段开始,将线段中间的三分之一部分用一个等边三角形的两边代替,形成山丘形图形,如图 1 所示。

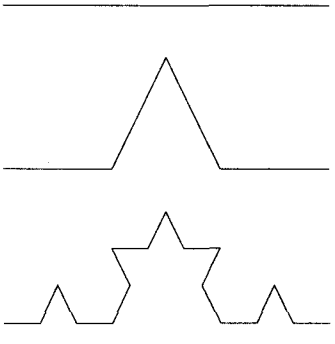

图 1 Koch 曲线

在新的图形中,又将图中每一直线段中间的三分之一部分都用一个等边三角形的两条边代替,再次形成新的图形如

到稿日期:2010-12-02 返修日期:2011-03-22 本文受河南省教育厅自然科学研究计划项目(2010A510015,2008B120010),江西省高性能计算技术重点实验室开放课题资助。

刘润杰(1969-),男,博士,讲师,主要研究领域为通信网络特性、混沌分形方法,E-mail:ierjliu@zzu.edu.cn;**申金媛**(1966-),女,博士,教授,主要研究领域为神经网络、信号处理、非线性方法;**穆维新**(1958-),男,硕士,副教授,主要研究领域为通信网络、交换技术。

此迭代，从而形成 Koch 分形曲线。解决 Koch 分形曲线问题通常使用递归算法。

1.2 形式化方法 PAR 开发其非递归算法程序

步骤 1　用 $Koch(p_{0,1},p_{0,2},n)$ 表示求解原问题得到的解，即在 $p_{0,1}$ 和 $p_{0,2}$ 间画出递归 n 次的 Koch 曲线，由此可得此问题的 Radl 规约：

$|[X:list(\ list(list(\ real,2),2))\ ;\ ]|$

$AQ:n>0;\ AR:X=Koch(p_{0,1},p_{0,2},n)$

步骤 2　分划原问题。根据 Koch 曲线问题的性质，可将原问题分划为

$$Koch(p_{0,1},p_{0,2},n)=F(Koch(p_{1,1},p_{1,2},n-1),Koch(p_{1,2},p_{1,3},n-1),Koch(p_{1,3},p_{1,4},n-1),Koch(p_{1,4},p_{1,5},n-1))$$

其中

$p_{1,1}=p_{0,1}$

$p_{1,2}=p_{0,1}+(p_{0,2}-p_{0,1})/3$

$p_{1,3}=p_{0,1}+(p_{0,2}-p_{0,1})/3+((p_{0,2}-p_{0,1})/3)'$

$p_{1,4}=p_{0,1}+2(p_{0,2}-p_{0,1})/3$

$p_{1,5}=p_{0,2}$

$(p_{0,2}-p_{0,1})/3$ 表示 $p_{0,1}$，$p_{0,2}$ 间线段的三分之一，$2(p_{0,2}-p_{0,1})/3$ 表示 $p_{0,1}$，$p_{0,2}$ 间线段的三分之二，$((p_{0,2}-p_{0,1})/3)'$ 表示 $p_{0,1}$，$p_{0,2}$ 间线段的三分之一左旋 60 度。

步骤 3　根据分划寻找递推关系 F，同时求得循环不变式。基于上述分划，可得

$$Koch(p_{0,1},p_{0,2},n)=Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\uparrow Koch(p_{1,3},p_{1,4},n-1)\uparrow Koch(p_{1,4},p_{1,5},n-1)$$

根据此递推关系，容易得到解 *Koch* 曲线问题的递归算法。为了得到一个非递归的算法程序，进行下列推导：

$$\begin{aligned}
&Koch(p_{0,1},p_{0,2},n)\\
&=Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\\
&\quad\uparrow Koch(p_{1,3},p_{1,4},n-1)\uparrow Koch(p_{1,4},p_{1,5},n-1)\\
&=Koch(p_{2,1},p_{2,2},n-1)\uparrow Koch(p_{2,2},p_{2,3},n-1)\\
&\quad\uparrow Koch(p_{2,3},p_{2,4},n-1)\uparrow Koch(p_{2,4},p_{2,5},n-1)\\
&\quad\uparrow Koch(p_{2,5},p_{2,6},n-1)\uparrow Koch(p_{2,6},p_{2,7},n-1)\\
&\quad\uparrow Koch(p_{2,7},p_{2,8},n-1)\uparrow Koch(p_{2,8},p_{2,9},n-1)\\
&\quad\uparrow Koch(p_{2,9},p_{2,10},n-1)\uparrow Koch(p_{2,10},p_{2,11},n-1)\\
&\quad\uparrow Koch(p_{2,11},p_{2,12},n-1)\uparrow Koch(p_{2,12},p_{2,13},n-1)\\
&\quad\uparrow Koch(p_{2,13},p_{2,14},n-1)\uparrow Koch(p_{2,14},p_{2,15},n-1)\\
&\quad\uparrow Koch(p_{2,15},p_{2,16},n-1)\uparrow Koch(p_{2,16},p_{2,17},n-1)\\
&=\cdots\cdots
\end{aligned}$$

可以看出为求解原问题，产生出很多和原问题结构相同的子问题，其中递归深度为 0 的子问题可以通过画两点间的直线直接求解，如 $Koch(p_{0,1},p_{0,2},0)$ 的解为 $line(p_{0,1},p_{0,2})$，即在点 $p_{0,1}$ 和 $p_{0,2}$ 间画直线。$X=(\langle xp_{0,1},yp_{0,1}\rangle,\langle xp_{0,2},yp_{0,2}\rangle)=line(p_{0,1},p_{0,2})=Koch(p_{0,1},p_{0,2},0)$

由此可以得到用非递归算法解 Koch 曲线问题的总策略：将每个子问题表示成三元实数序列的形式[起始点坐标，终止点坐标，递归深度]；引进 3 个序列变量 X、q、S，其中序列变量 X 用于存放部分子问题的解，也就是已得到的 Koch 曲线坐标序列，每个坐标表示成二元序列$[p_a,p_b]$的形式，对应着的动作是在点 p_a 和 p_b 间画直线，循环终止时 $X=Koch(p_{0,1},p_{0,2},n)$。$q$ 为三元序列，用于存放正准备解决的子问题，$q[1]$为该子问题中起始点坐标，$q[2]$为终止点坐标，$q[3]$为递归深度，如 $q=Koch(p_a,p_b,n)$表示在 p_a 和 p_b 间画出递归 n 次的 Koch 曲线；S 是一起堆栈作用的序列变量，用于存放尚待解决的子问题，如 $Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\uparrow Koch(p_{1,3},p_{1,4},n-1)\uparrow Koch(p_{1,4},p_{1,5},n-1)$；$S$ 的内容由函数 F 给出，F 的定义为

(1) $F([])=[]$；

(2) $F(q\uparrow S)=Koch(q)\uparrow F(S)$。

根据总策略，得到 X、q、S 满足如下等式，构成所需的循环不变式。

$\rho:X\uparrow Koch(q)\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

其中的 $Koch(q)$ 表示子问题 $Koch(p_a,p_b,n)$。

步骤 4　基于递推关系和循环不变式，可以导出下列非递归 Apla 抽象算法程序。

```
program Koch
  {PQ:n>0; PR:X= Koch(p0,1,p0,2,n)}
  var X:list( list(list(real,2),2))
  q:list(list(list(real,2),2); interger)
  S:list(list(list(real,2),2); interger )
begin
  X,S,q:=[],[],[p0,1,p0,2,n]
  do q≠[]∧q[3]≠0
       →q,S:= Koch(p1,1,p1,2,n-1),Koch(p1,2,p1,3,n-1)
        ↑Koch(p1,3,p1,4,n-1) ↑Koch(p1,4,p1,5,n-1)↑S
    [] q≠[]∧q[3]=0
      →X,q:=X↑[pa,pb],[]
    [] q=[]∧S≠[]
      →q,S:=S[h],S[h+1..t]
    od
end
```

该程序首先将 q 初始化为原问题，而 X、S 初始化为空。*do* 语句的第一个分支表示子问题 q 要画的 Koch 曲线递归深度大于 0，不能直接求解，将子问题根据分划原则继续分划成 4 个子问题，并将分划出来的另 3 个子问题存入序列 S；第二个分支表示子问题 q 要画的 Koch 曲线递归深度等于 0，可直接画出，在点 p_a 和 p_b 间画直线，该结果表示成二元坐标序列$[p_a,p_b]$存入结果序列 X 中，并置 q 为空；第三个分支表示 q 为空而 S 非空，即 q 中的子问题已经求解，需取序列 S 的头元素(即一个未解决的子问题)赋给 q，以便下次求解，同时将该子问题(S 头元素)从尚未解决的序列中删除。

另外，若只需画出 Koch 曲线而不需保存起来，则此程序中的序列变量 X 可去掉，同时将第二个 *do* 分支改成“[] $q\neq[]\wedge q[3]=0\rightarrow line(p_a,p_b)$; $q:=[]$;”即可。

2　形式化证明

基于所得到的循环不变式，可用 Dijkstra 的最弱前置谓词法[11]形式化证明上述程序正确。

2.1 证明 ρ 在循环执行前为真

$$\begin{aligned}
&wp(\text{“}X,S,q:=[],[],[Koch(p_{0,1},p_{0,2},n)];\text{”},\rho)\\
&\equiv[]\uparrow Koch(p_{0,1},p_{0,2},n)\uparrow F(\ [])=Koch(p_{0,1},p_{0,2},n)\\
&\equiv\text{true}
\end{aligned}$$

2.2 根据循环体证明 ρ 确实是循环不变式

①$\rho\wedge q\neq[]\wedge q[3]\neq0==>wp(\text{“}q,S:=(p_{1,1},p_{1,2},n-1),$

$(p_{1,2},p_{1,3},n-1)\uparrow(p_{1,3},p_{1,4},n-1)\uparrow(p_{1,4},p_{1,5},n-1)\uparrow S'',\rho)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(p_{1,1},p_{1,2},n-1)\uparrow F((p_{1,2},p_{1,3},n-1)\uparrow(p_{1,3},p_{1,4},n-1)\uparrow(p_{1,4},p_{1,5},n-1)\uparrow S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\uparrow F((p_{1,3},p_{1,4},n-1)\uparrow(p_{1,4},p_{1,5},n-1)\uparrow S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\uparrow Koch(p_{1,3},p_{1,4},n-1)\uparrow F((p_{1,4},p_{1,5},n-1)\uparrow S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(p_{1,1},p_{1,2},n-1)\uparrow Koch(p_{1,2},p_{1,3},n-1)\uparrow Koch(p_{1,3},p_{1,4},n-1)\uparrow Koch(p_{1,4},p_{1,5},n-1)\uparrow F(S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(q[1],q[2],q[3])\uparrow F(S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]\neq 0==>X\uparrow Koch(q[1],q[2],q[3])\uparrow F(S)$

$=Koch(p_{0,1},p_{0,2},n)$

$\equiv X\uparrow mov(q)\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)\wedge q\neq[]\wedge q[3]\neq 0$

$==>X\uparrow Koch(q)\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv$true

② $\rho\wedge q\neq[]\wedge q[3]=0==>wp("X,q:=X\uparrow[p_{1,1},p_{1,2}],[]",\rho)$

$\equiv\rho\wedge q\neq[]\wedge q[3]=0$

$==>X\uparrow[p_{1,1},p_{1,2}]\uparrow Koch([])\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q\neq[]\wedge q[3]=0$

$==>X\uparrow[p_{1,1},p_{1,2}]\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv X\uparrow Koch(q)\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)\wedge q\neq[]\wedge q[3]=0$

$==>X\uparrow[p_{1,1},p_{1,2}]\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv X\uparrow[q[1],q[2]]\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$==>X\uparrow[p_{1,1},p_{1,2}]\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv$true

③ $\rho\wedge q=[]\wedge S\neq[]==>wp("q,S:=S[h],S[h+1..t];",\rho)$

$\equiv\rho\wedge q=[]\wedge S\neq[]$

$==>X\uparrow Koch(S[h])\uparrow F(S[h+1..t])=Koch(p_{0,1},p_{0,2},n)$

$\equiv\rho\wedge q=[]\wedge S\neq[]==>X\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv X\uparrow Koch([])\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$==>X\uparrow F(S)=Koch(p_{0,1},p_{0,2},n)$

$\equiv$true

2.3 证明后置断言 PR 在循环终止时为真

$\rho\wedge\neg B==>PR$

$\equiv\rho\wedge\neg((q\neq[]\wedge q[3]\neq 0)\vee(q\neq[]\wedge q[3]=0)\vee(q=[]\wedge S\neq[]))==>PR$

$\equiv\rho\wedge\neg(q\neq[]\vee(q=[]\wedge S\neq[]))==>PR$

$\equiv\rho\wedge\neg(q\neq[]\vee S\neq[])==>PR$

$\equiv\rho\wedge q=[]\wedge S=[]==>X=Koch(p_{0,1},p_{0,2},n)$

$\equiv X\uparrow Koch([])\uparrow F([])=Koch(p_{0,1},p_{0,2},n)$

$==>X=Koch(p_{0,1}p_{0,2},n)$

$\equiv X=Koch(p_{0,1},p_{0,2},n)==>X=Koch(p_{0,1},p_{0,2},n)$

$\equiv$true

2.4 循环的终止性显然成立

至此,完成了该程序的正确性证明。

3 Koch 曲线问题非递归算法实现

给 Apla 抽象程序加上输入、输出语句后,基于支持 Apla 抽象数据类型的可重用部件库及自动程序转换系统,可将其转换成某一可执行语言程序。递归 5 次的 Koch 曲线如图 2 所示。

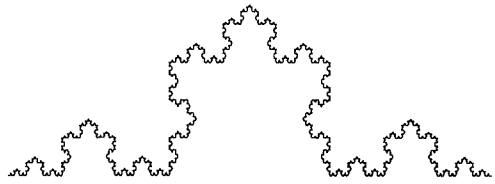

图 2 非递归程序产生的 Koch 曲线

算法程序中空间开销较大的是序列 S,因为 S 仅存放尚待求解的子问题,整个算法的空间复杂性远远小于定义栈或数组来存放结果的非递归算法。该算法不需要使用堆栈。而递归算法堆栈需要较大的内存空间。由此可看出该非递归算法在空间上占有绝对优势。针对此算法,利用类型转换函数,可将 S 定义成字符型序列来进一步优化其空间效率。

结束语 本文形式化开发了 Koch 曲线的非递归算法程序,并对其正确性进行了证明,所得算法程序可在相应平台及部件库的支撑下进一步得到可执行程序。

Koch 曲线的非递归算法在其它文献中也有叙述,如高军等[12]使用 DELPH 语言以递归方式实现了 Koch 曲线,孙继军等[13]使用 C++语言以递归方式实现了 Koch 曲线。总体来说,前述工作编程思路较复杂,且所使用的技术难以掌握,不具有通用性。本文产生的算法抽象、简洁、易读,可靠性高;所使用的技术简单易用,具有通用性,可望发展成求解一系列类似递归问题的方法。另外,现有的 Koch 曲线算法程序的研究没有涉及到其循环不变式的开发,无法开展对算法程序的形式化证明,难以保证算法程序的正确性。本文直接面向非递归算法程序的开发,使用循环不变式开发新策略,简捷地得到了 Koch 曲线算法程序的循环不变式的简单表达形式,从而可对产生的算法程序开展形式化证明,保证其正确性。

参 考 文 献

[1] Peitgen H-O, Jurgens H, Saupe D. Chaos and Fractals: New Frontiers of Science[M]. New York: Springer-Verlag, 1992

[2] 张济忠. 分形[M]. 北京:清华大学出版社,1995

[3] Kocarev L. Chaos-based cryptography: a brief overview [J]. IEEE Circuits and Systems Maganize, 2001, 1(3): 6-21

[4] Yang T. A survey of chaotic secure communication systems[J].

Int J CompCognit,2004,2(2):81 - 130.
[5] 赵耀,王红星,袁保宗. 分形图像编码研究的进展[J]. 电子学报,2000,28(4):95-101,106
[6] Liu M Q,Zhao Y,et al. A fast fractal image coding algorithm based on FGSE[A] // Signal Processing[C]. Beijing: IEEE Press,2006:1153-1156
[7] Bobtsov A,Nikolaev N,Slita O. Control of Chaotic Oscillations of a Satellite[J]. Applied Mathematics and Mechanics(English Edition),2007,28(7):893-900
[8] 屈文建,薛锦云. PAR 方法和循环不变式的范畴语义[J]. 计算机工程与应用,2009,45(8):50-54
[9] Xue J Y. A unified approach for developing efficient algorithmic programs[J]. Journal of Computer Science and Technology,1997,12(4):314-329
[10] Xue JY. Two new strategies for developing loop invariants and their applications[J]. Journal of Computer Science and Technology,1993,8(2):147-154
[11] Dijkstra E W. A discipline of programming [M]. Englewood Cliffs:Prentice Hall,1976
[12] 高军,孙博玲. KOCH 曲线的 DELPHI 程序设计[J]. 电脑学习,2000(2):35-36
[13] 孙继军,卢玉蓉. Von Koch 曲线的 Visual C++程序实现[J]. 攀枝花学院学报,2004,21(1):90-92

(上接第 118 页)

4.2 基于置信度数据库的反馈机制性能分析

本节针对基于置信度数据库的反馈机制对上下文不一致性检测及消除的时间消耗的影响进行实验研究。实验使用的上下文不一致性消除算法为基于上下文不一致性次数的循环消除算法;并且为了便于进行置信度调整,将数据库内存储的关于每个上下文模式相匹配的上下文实例带入进行检测消除的总实例数设为 100;调整其中成功和失败的实例数,即改变了该上下文模式的置信度,从而可以实现对检测器和消除器的不同的反馈机制,使其能够对二者分别产生作用。实验结果如图 4 所示。

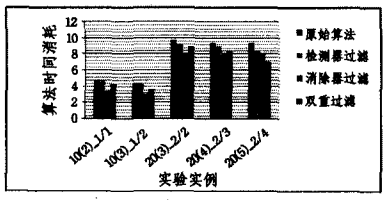

图 4 基于置信度数据库的反馈机制对上下文不一致性检测及消除算法的影响

其中,纵坐标表示时间消耗;横坐标表示带入实验的实例组,具体表示为"实例数(不一致数)_将被检测器过滤实例数/将被消除器过滤实例数"。实验所得算法时间为单次上下文不一致性检测及消除循环执行 1000 次并重复运行 5 次取其平均值所得到。本文将检测器过滤的阈值预设为 0.1,即上下文模式的不一致性置信度低于 0.1 将被认定为错误的上下文实例,不进行不一致性检测即将其直接消除。而消除器过滤的阈值预设为 0.2,同理,一旦该上下文模式被检测到和其他上下文模式存在不一致,即将其直接消除,不执行上下文不一致消除算法。

如图 4 所示,采用了基于置信度数据库的反馈机制的上下文检测及消除算法在时间耗费上明显少于原始的算法。这其中,检测器过滤的力度较小,所以降幅不明显,说明该反馈对于检测器和原始算法的影响较小。但最后一组实验实例说明,当实例设计及规模不定的情况下,检测器过滤也有一定的效果。消除器过滤,虽然只在消除器处进行了算法优化,但由于消除算法在整体算法中的主导地位以及阈值的设定可以过滤掉较多的不一致上下文实例,使得该反馈机制的时间耗费降幅较为明显。而双重过滤的方式在算法的时间消耗上较少,但是需要综合考虑数据库查询的次数,所以对其的使用需要根据不同情况进行分析。

结束语 本文以上下文不一致性的检测及消除算法的研究为核心,提出了 4 种上下文不一致性消除算法和循环排序的消除算法扩展,同时提出了针对上下文检测及消除的基于置信度数据库的反馈机制,并对其正确性和时间消耗进行了评估,丰富了上下文不一致性检测及消除的框架和方法。后续工作将从以下几方面入手:针对置信度数据库在阈值方面的设定对上下文不一致性检测及消除算法的影响进行研究;对上下文不一致性检测及消除流程图所示的前提过滤条件进行研究,希望能有效地过滤上下文实例的输入从而提高上下文不一致性的检测及消除的效率;针对不同领域的上下文信息进行评估,研究本文提出的算法在不同领域模型中的实际应用情况。

参 考 文 献

[1] 郑笛,朱珊. 普适计算环境下上下文不一致性的消除算法研究[J]. 计算机应用研究,2009,26(1):152
[2] 郑迪. 基于上下文感知服务的构件化中间件关键技术研究[D]. 长沙:国防科学技术大学,2008
[3] Xu C,Cheung S C. Inconsistency detection and resolution for context-aware middleware support[C]// the Joint 10th European Software Engineering Conference and 13th ACM SIGSOFT Symposium on the Foundations of Software Engineering(ESEC/FSE 2005). Lisbon:ISBN,2005:336-345
[4] Xu C,Cheung S C,Chan W K. Incremental consistency checking for pervasive context[C]// the 28th International Conference on Software Engineering(ICSE 2006). Shanghai:ISBN,2006:292-301
[5] Huang Yu,Yu Jian-ping,Cao Jian-nong,et al. Checking Behavioral Consistency Constraints for Pervasive Context in Asynchronous Environments[D]. Nanjing:Nanjing University,2009
[6] Huang Yu,Yu Jian-ping,Cao Jian-nong,et al. Concurrent Event Detection for Asynchronous Consistency Checking of Pervasive Context[D]. Nanjing:Nanjing University,2009
[7] Huang Yu,Yu Jian-ping,Cao Jian-nong,et al. A Probabilistic Approach to Consistency Checking for Pervasive Context[D]. Nanjing:Nanjing University,2008
[8] 叶光昶. 普适计算环境下自适应技术研究[D]. 上海:上海交通大学,2007
[9] 百度百科,置信度[OL]. http://baike.baidu.com/view/434404.htm