数组算法 演绎综合 算法程序 开发

计算机科学2000Vol. 27№. 8

55-58

一类数组算法的演绎综合

Deduction Synthesis of a Class Array Algorithms

TP311.52

黄明和 刘艳霞 杨柳青

(江西师范大学计算机科学技术学院 南昌330027)

Abstract Deductive synthesis is one of the important means used to formally develop algorithms and programs. This paper begins with each clear and explicit problem specifications. By using program transformation techniques and deduction means, we formally derive a class of array algorithms. By appropriate changing deductive decisions, we can get different algorithms solving the same problem. This effectively realize the reuse of equations and deductive steps, and reveal the inherent characteristics and relations between the algorithms

Keywords Array, Algorithm, Synthesis, Program transformation

1. 引言

演绎综合是一种形式化开发算法程序的方法，它从给定问题的一个易于理解、简洁明确的规范说明出发(规范说明表达了所要设计程序的目的)，通过定理证明，程序变换，演绎推理等手段，形式化地推导出解决问题的正确算法或程序。算法程序演绎综合的基础是一个小的程序变换规则集和若干基于问题的演绎法则。在算法程序的演绎过程中适当改变推导决策，可以得到解决同一问题的不同算法，这有助于实现算法推导步的重用，有助于揭示各算法间的内在特系和相互关系。这种算法程序的开发方法不仅使人信服所得算法程序确实能够完成给定的任务，而且通过揭示导致最后算法程序的设计决策来清楚表达算法程序是怎样完成了这个任务。本文将这种方法应用于常见的一类数组算法，以展示这种方法的功能。

2. 基本法则

为了描述和推导算法，我们需要如下基本程序变换和演绎法则。

2.1 基本变换法则

折迭(Folding) 设 $E \Leftarrow E_1, F \Leftarrow F_1$ 是方程，如果在 F_1 中出现 E_1 的实例，则可用对应的 E 的实例替换它。

2.2 基本演绎法则

RS_1 find all f(i)where i from A to B such that G(n)
⇐φ if A>B
⇐{f(B)}∪find all f(i) where i from A to B-1 such that G(n) if (i=B)and (G(n)=True)
⇐find all f(i) where i from A to B-1 such that G(n) otherwise

RS_2 find some f(i) where i from A to B such that G(n)
⇐φ if A>B
⇐{f(B)} if (i=B)and (G(n)=True)
⇐find some f(i) where i from A to B-1 such that G(n) otherwise

RS_3 test all f(i) where i from A to B such that G(n)
⇐True if A>B
⇐test all f(i) where i from A to B−1 such that G(n) if (i=B)and (G(n)=True)
⇐False otherwise

RS_4 test some f(i)= where i from A to B such that G(n)
⇐False if A>B
⇐True if (i=B)and (G(n)=True)
⇐test some f(i) where i from A to B−1 such that G(n) otherwise

RS_5 Count f(i) where i from A to B such that G(n)
⇐0 if A>B
⇐Count f(i) where i from A to B−1 such that G(n)+1 if(i=B)and (G(n)=True)
⇐Count f(i) where i from A to B−1 such that G(n) otherwise

其中G(n)是满足n个条件表达式的布尔函数。

3. 算法的演绎综合

问题1 在数组中找出所有元素值等于常数 x 的数组元。

问题的规范描述是：

1)EA(D,A,B)⇐find all D[i] where i from A to B such that D[i]=x

根据 RS_1，方程1)可推导为：

⇐φ if A>B

⇐{D[B]}∪find all D[i] where i from A to B-1 such that D[i]=x if D[B]=x

⇐find all D[i] where i from A to B-1 such that D[i]=x otherwise

上式中两处划线部分分别是方程1)的一个实例，用方程1)folding，上式可化为

⇐φ if A>B

⇐{D[B]}∪EA(D,A,B-1) if D[B]=x

⇐EA(D,A,B-1) otherwise

由于 B 不断减少，算法的终止是明显的。最后的算法为：

A_1:EA(D,A,B)

⇐φ if A>B

⇐D[B]∪EA(D,A,B−1) if D[B]=x

⇐EA(D,A,B−1) otherwise

问题2 求数组中两相邻元素之和为最大值的问题的规范描述是：

2)Max(D,A,B)⇐find some D[i]+D[i+1] where i from A to B-1 such that D[i]+D[i+1]≥all D[j]+D[j+1] where j from A to B−1

根据 RS_2，方程2)可推导为：

⇐φ if A>B−1

⇐D[B−1]+D[B] if D[B−1]+D[B]≥all D[j]+D[j+1]

where j from A to B−2 (*)

⇐find some D[i]+D[i+1]where i from A to B−2 such taht D[i]+D[i+1]≥all D[j]+D[j+1] where j from A to B−2 otherwise

上式的划线部分恰是方程2)的实例，由此有

⇐φ if A>B−1

⇐D[B−1]+D[B] if D[B-1]+D[B]≥all (D[j]+D[j+1])

where j from A to B−2

⇐Max(D,A,B−1) otherwise

其中，D[B−1]+D[B]≥all (D[j]+D[j+1]) where j from A to B−2不是原始逻辑语句，我们需对其进行化简推导，在此引入更一般的方程 Fmax(D,B,h,k)来实现，其程序规范为：

3)FMax(D,B,h,k)⇐test all D[i]+D[i+1] where i from h to k−1 such that D[B−1]+D[B]≥(D[i]+D[i+1])

容易看出，现在上面的非原始逻辑语句恰好是方程3)右边的一个实例，用 folding 法则，它可转换为 Fmax(D,B,A,B−1)。现在我们来推导方程3。

根据 RS_3，方程3)可推导为：

⇐True if h>k−1

⇐test all(D[i]+D[i+1])where i from h to k−2 suth that D[B−1]+D[B]≥D[i]+D[i+1] if D[B−1]+D[B]≥D[k−1]+D[k]

⇐False otherwise

上式的划线部分恰好是方程3)的一个实例，用方程3)folding，上式可进一步推导为：

⇐True if h>k−1

⇐FMax(D,B,h,k−1) if D[B−1]+D[B]≥D[k−1]+D[k]

⇐False otherwise

由于 k 值不断减少，方程3)的终止亦是明显的，这样我们就完成了方程2)的推导并得到算法 A_2：

Max(D,A,B)

⇐φ if A>B−1

⇐D[B−1]+D[B] if FMax(D,B,A,B−1)

⇐Max(D,A,B−1) otherwise

FMax(D,B,h,k)

⇐True if h>k−1

⇐FMax(D,B,h,k−1) if D[B−1]+D[B]≥D[k−1]+D[k]

⇐False otherwise

应该指出，这个算法的效率是很低的，当数组元素个数≥2时，它总是用最后一个相邻元与前面相邻元逐个比较，如果它比前面每个相邻元之和都大，则它就是和最大的相邻元，否则取次最后相邻元和再自后往前逐个比较，按照这种做法，对于 n 个元的数组，最坏情况下将要比较：

$$\sum_{i=1}^{n-1} n-i-1=\sum_{i=1}^{n-2} i=\frac{1}{2}(n-2)(n-1)$$

这个复杂度是相当高的，原因在于算法做了许多无益的比较。试想，数组中的某对相邻元和若在比较的过程中失败则它就不可能成为相邻元和最大者，由此再用它去与前面一切相邻元和作比较是毫无意义的，前面算法正是没有考虑这一点由此形成了一个效率低下的算法。对(*)式稍加分析，易见 if D(B−1)+D[B]≥all (D[j]+D(j+1)) where j from A to B−2和 D[B−1]+D[B]大于等于数组 D 在 A 到 B-1之间相邻元和最大者是等价的，由此上式就可写成：if D(B−1)+D[B]≥max (D,A,B−1)。这样最后算法可写成：

Max(D,A,B)

⇐∅ if A>B-1

⇐D[A]+D[B] if A=B-1

⇐D[B-1]+D[B] if D(B-1)+D[B]≥Max(D,A,B-1)

⇐Max(D,A,B-1) otherwise

实现时将Max(D,A,B-1)上提，使第三分支和第四分支中的Max(D,A,B-1)只需算一次。这样求Max(D,A,B)问题就转化为求Max(D,A,B-1)的问题，而求Max(D,A,B-1)问题又转化为求Max(D,A,B-2)的问题，一直下去，容易分析出，此时的相邻元和的比较次数将是n-2(它比前面算法复杂度好得多)。事实上对于n个元的数组，共有n-1个相邻元，而在n-1个相邻元中求最大相邻元和至少需要n-2次比较，由此上述算法在比较类中已达到最佳。

问题3 在一个数组中找第k小的元素

问题的规范描述是：

4)Fke(D,A,B)⇐find some D[i] where i from A to B such that(Count D[j] where j from A to B such that D[j]<D[i])=k-1

此问题的讨论与问题2类似，为了求解方程4)，我们需要先将其扩广为一个更一般的方程Gfke(D,A,B,M,N)，其程序规范为：

5)Gfke(D,A,B,M,N)⇐find some D[i] where i from A to B such that(Count D[j]where j from M to N such that D[j]<D[i])=k-1

其语义是，在数组D中找一个元素D[i]，该元素在数组D的M到N范围内是第k小的元素。容易看出，方程4)恰好是方程5)在M=A,N=B时的特例，也即

Fke(D,A,B)⇐Gfke(D,A,B,A,B)

根据RS_2，方程5)可推导为：

⇐φ if A>B

⇐D[B] if (Count D[j] where j from M to N such that D[j]<D[B])=k-1

⇐Gfke(D,A,B-1,M,N) otherwise

其中划线部分Count D[j] where j from M to N such that D[j]<D[B]不是原始逻辑语句，必须对其进行化简。为此引入方程Sum(D,F,G,B)来实现，其程序规范为：

6)Sum(D,F,G,B)⇐Count D[j] where j from F to G such that D[j]<D[B]

易见，现在上面的非原始逻辑语句正好是方程6)的实例，并可以写成Sum(D,M,N,B)

根据RS_5，方程6)可推导为：

⇐0 if F>G

⇐Sum(D,F,G-1,B)+1 if D[G]<D[B]

⇐Sum(D,F,G-1,B) otherwise

根据方程4)，5)，6)我们综合出最后的算法AS_3：

Fke(D,A,B)⇐Gfke(D,A,B,A,B)

Gfke(D,A,B,M,N)

⇐φ if A>B

⇐D[B] if Sum(D,M,N,B)=k-1

⇐Gfke(D,A,B-1,M,N) otherwise

Sum(D,F,G,B)

⇐0 if F>G

⇐Sum(D,F,G-1,B)+1 if D[G]<D[B]

⇐Sum(D,F,G-1,B) otherwise

问题4 平台问题

在给出此问题的规范描述之前，我们首先引进两个方程：

第一个方程Equ(D,k,j)，判断数组D中k+1到j之间的数是否全相等。

7)Equ(D,k,j)⇐test all D[i] where i from k+1 to j-1 such that D[i]=D[j]

根据RS_3并稍作变形，方程7)可推导为：

⇐True if k+1≥j

⇐test all D[i] where i from k+1 to j-2 such that D[i]=D[j-1] if D[j]=D[j-1]

⇐False otherwise

上式划线部分是方程7)的一个实例，用方程7)folding，上式可推导为：

⇐True if k+1>j

⇐Equ(D,k,j-1) if D[j]=D[j-1]

⇐False otherwise

第二个方程DC(D,A,B,M)，判断数组D中A到B之间是否有长度为M的平台。

8)DC(D,A,B,M)⇐test some j where j from A+M-1 to B such that Equ(D,j-M,j)

根据RS_4，方程8)可推导为：

⇐False if A+M-1>B

⇐True if Equ(D,B-M,B)

⇐Test some j where j from A+M-1 to B-1 such that Equ(D,j-M,j) otherwise

上式划线部分是方程8)的一个实例，用方程8)folding可得：

⇐False if A+M-1>B

⇐True if Equ(D,B-M,B)

⇐DC(D,A,B-1,M) otherwise

所谓平台问题即找出数组D中A到B之间长度小于或等于B-A+1的最大平台，于是我们可以得到

问题4的规范描述如下.

9)DM(D,A,B,B−A+1)⇐find some L where L from 1 to B−A+1 such that (DC(D,A,B,L) and L≥all k where k from 1 to B−A+1 such that DC(D,A,B,k))

根据 RS_2,方程9)可推导为:

⇐φ　　if A>B
⇐B−A+1　if DC(D,A,B,B−A+1)
⇐Find some L where L from 1 to B−A such that (DC(D,A,B,L)and L≥all (k) where k from 1 to B−A such that DC(D,A,B,k)) otherwise

上式的划线部分是方程9)的一个实例,用方程9) folding 可得:

⇐φ　　if A>B
⇐B−A+1　　if DC(D,A,B,B−A+1)
⇐DM(D,A,B,B−A)　otherwise

根据方程7),8),9),我们可得到最后的算法 A_4:

DM(D,A,B,B−A+1)
⇐φ　　if A>B
⇐B−A+1　　if DC(D,A,B,B−A+1)
⇐DM(D,A,B,B−A)　otherwise
DC(D,A,B,M)⇐False　if A+M−1>B
⇐True　if Equ(D,B−M,B)
⇐DC(D,A,B−1,M)　otherwise
Equ(D,k,j)⇐True　if k−1≥j
⇐Equ(D,k,j−1)　if D[j]=D[j−1]
⇐False　otherwise

结束语　形式化开发算法程序是提高软件生产率和可靠性,最终实现软件自动生成的重要途径,从上述算法程序的演绎综合实践,我们看到将算法的共性上移,个性延迟决策是实现算法程序推导步重用的重要手段,这有助于利用尽可能少的推导步形式化地开发出尽可能多的一类算法。容易看到,使用这种方法我们还能推导出类似子集和问题、背包问题等其它一类数组问题的算法程序。

参 考 文 献

1 Darlington J. A synthesis of several sorting algorithms. Acta Informatic,1978,11:1～30
2 许卓群,张乃孝,杨冬青,唐世渭. 数据结构. 高等教育出版社,1987
3 黄明和,刘润杰. 图回路算法的演绎综合. 计算机科学,1999 11

(上接第51页)

```
  }
  !data. "return the sorted data"
 }
} "end of the methods."
```

如果其它对象执行下面这一未来型信息传送式:

newobject=quick new.

于是,就生成了新的动态对象,该动态对象指针由于 class 内回答信息的返回而被代入变量 newobject 中。这样,发出生成动态对象委托的对象可用变量 newobject 来引用新生成的动态对象,亦可用 newobject 作为参数向其它对象传送信息,即向其它对象通知新的动态对象的诞生。于是,其它对象也可以引用新生成的动态对象了,在 class 内部引用临时变量 cell 来引用新生成的对象。

4.2 动态对象的消除

与对象的动态生成相对应,NETL 设置了动态消除对象的功能,这由执行下面的语句来实现:

objectName delete.

即向要消除的对象发出特殊的信息 delete 即可。被消除的对象将从系统的管理表中彻底删除掉,因此,用户适当使用生成动态对象和消除对象的功能,可使有限的硬件资源得以充分的利用。

4.3 method 执行的中止

某对象启动另一个对象,当已接收到来自被启动的对象的回答信息后,如果此时被启动的对象还在继续执行,就可以中止被启动的对象的处理。NETL 利用特殊信息"kill"来中止被委托的 method 的执行。假如被委托的对象为 anobject,其 method 的信息模式为 job 的话,中止该 method 的描述为:

anobject kill:job.

有权发出处理中止的只能是启动该 method 的对象。

参 考 文 献

1 吕英华. A-NET 模拟程序中进程间通信接口的实现. 计算机科学,1999,26(11),8～10
2 吕英华. UNIXシステムコールによるA-NETルータ间通信模拟の实现. In:研究纪要(日本数学教育学会春季年会发表论文),1998. 19～21
3 吕英华. 并列计算机シミュレータにおけるHostとRouter间のインタフエース. In:研究纪要(日本数学教育学会秋季例会发表论文),1998. 61～63
4 吕英华. UNIXのペイプによるセマフオの作成. In:日本第42回数学の文化史研究会发表论文,1998
5 吕英华. A-NETL 中对象的动态生成和在 A-NET 机中的实现. 计算机科学,1998,25(4):111～112