文章编号:1000-5862(2019)04-0402-07

2 类组合数学问题的算法形式化推导

熊小超,杨庆红*
(江西师范大学计算机信息工程学院,江西 南昌 330022)

摘要:组合数学问题算法的研究是计算机科学的重要研究内容,但在许多相关文献中,许多组合数学问题的算法只是经过简单分析得到,并未给出算法程序的详细设计过程,导致读者无法理解算法本质,更无法保证算法程序的正确性. 该文在以组合数学中连续子序列最大乘积和第 2 类斯特林数变形问题为例的基础上,通过形式化描述问题的程序规约,使用规约变换规则,对程序规约进行一系列等价变换,获得问题求解序列的递推式,并以此为基础得到问题求解的算法程序,清晰地展示了从问题的需求到算法程序的详细推导过程. 通过对相关组合数学问题的进一步深入研究,提炼了 2 类组合数学问题的求解策略,为提高组合数学问题算法程序的正确性提供了有效途径.

关键词:形式化方法;程序规约;组合数学;递推技术
中图分类号:TP 311 文献标志码:A DOI:10.16357/j.cnki.issn1000-5862.2019.04.12

0 引言

组合数学是一门研究离散对象的科学,重点研究满足一定条件组合模型的存在、计数、构造和优化等方面的问题[1-2];而计算机科学是研究算法的科学,且计算机所处理的对象大部分是离散的数据[3]. 因此,随着计算机科学的日益发展,组合数学的重要性也日渐凸显[4],关于组合数学问题算法的研究也成为了计算机科学的重要研究内容. 然而,在许多相关文献和资料中,组合数学问题的相关算法只是经过简单分析得到,并未给出算法程序的详细设计过程,导致读者无法深入理解算法的本质[3];同时,算法程序的正确性也无法保证.

随着"软件危机"的出现,软件的正确性和可靠性成了现今重要的研究内容之一,而算法程序的正确性是软件开发中的核心问题,形式化方法是提高算法正确性的有效途径之一. 目前,形式化方法已经成为计算机科学的一个重要分支,从最简单的 1 阶谓词演算发展到基于逻辑、状态机、进程代数等众多的形式化方法[5],经过几十年的深入研究和应用,形式化方法在软件开发中取得了显著的成果[6-8].

Z 方法[9]以谓词逻辑和集合论为语义基础,提供了一种模式结构. 而 VDM 方法[10]通过谓词逻辑和已定义的抽象数据类型来对运算或函数功能进行结构化规格描述. 这 2 种方法使规格说明拥有了简短精炼、无歧义的优点,但它们不支持软件系统开发的所有阶段. B 方法[11-12]使用 1 阶谓词逻辑和集合论描述软件规范,以抽象机作为软件部件的描述单元,通过 B 语言来描述、设计和实现软件系统. B 方法支持软件系统开发的所有阶段,但使用抽象机描述软件开发过程较复杂,导致系统开发成本较高.

基于递推技术的形式化方法[13-14]根据问题需要实现的功能,形式化描述问题的程序规约[15],再通过程序规约变换技术进行一系列数学变换获取问题求解的递推关系,以此为基础获得算法程序. 该方法支持从程序规约到算法程序的整个开发阶段,可以详细展示算法的推导过程. 该过程是以 1 阶谓词逻辑为基础的,过程和递推式都是令人信服的. 同时,由于推导过程中采用数学等价变换,因而该方法通过保证推导过程的正确性来确保最终算法程序的正确性. 因此,本文采用基于递推技术的形式化推导方法展示 2 类组合数学问题的推导过程.

1 预备知识

程序规约采用量词的形式表示为 $(Qi:r(i):f(i))$,其含义是"在范围 $r(i)$ 上,对函数 $f(i)$ 应用 q

收稿日期:2018-12-17
基金项目:国家自然科学基金(61662035)资助项目.
通信作者:杨庆红(1968-),女,江西南昌人,教授,主要从事软件形式化和智能教育软件的研究. E-mail:yangqh120@163.com

运算所得到的量",其中 q 是量词 Q 对应的 2 元运算[4],Q 可以是 $\forall$(全称量词)、$\exists$(存在量词)、$\sum$(求和量词)、$\prod$(乘积量词)、MIN(求最小值量词)、MAX(求最大值量词)[5]. 本文使用到的变换规则有[6]:

(i) 范围分裂为 $(Qi:r(i):f(i)) = (Qi:r(i) \wedge b(i):f(i))(Qi:r(i) \wedge \neg b(i):f(i))$;

(ii) 单一范围为 $(Qi:i = k:f(i)) = f(k)$;

(iii) 交叉积为 $(Qi,j:r(i) \wedge s(i,j):f(i,j)) = (Qi:r(i):(Qj:s(i,j):f(i,j)))$;

(iv) 一般分配律为 $(Qi:r(i):g \odot f(i)) = g \odot (Qi:r(i):f(i))$,其中 i 不是 g 中的自由变量.

利用基于递推技术的形式化方法进行推导的算法的基本步骤为:(i) 根据问题需要实现的功能,形式化描述问题的程序规约;(ii) 利用程序规约变换技术对程序规约进行数学等价变换,获得问题求解的递推关系;(iii) 确定问题求解的初始条件和终止条件,并结合(ii) 中的递推关系得到问题求解算法;(iv) 根据问题求解算法得到相应的算法程序.

2　连续子序列问题的形式化推导

组合数学中有许多经典及难点问题,连续子序列问题是其中的一类子问题. 本文以连续子序列最大乘积问题为例,使用基于递推技术的形式化方法,展示算法程序的形式化推导过程.

2.1　连续子序列最大乘积

问题描述: 对于一个有 n 个实数的序列 $a[1:n]$,找出序列 $a[1:n]$ 中连续子序列的最大乘积 ($n \geqslant 1$). 例如对于序列 [2,3,-1,-2,3,-4] 中的乘积最大的连续子序列为 [2,3,-1,-2,3],其乘积是 36.

(i) 形式化描述问题的程序规约.

用 $\mathrm{Max}C_J(n)$ 来表示序列 $a[1,\cdots,n]$ 中连续子序列的最大乘积.

A_Q: 已知一个序列 $a[1,\cdots,n]$,$n > 0$;

A_R: $\mathrm{Max}C_J(n) = (\mathrm{MAX}\ i,j:1 \leqslant i \leqslant j \leqslant n:(\prod k:i \leqslant k \leqslant j:a[k]))$.

(ii) 分划原问题并求解递推关系.

将问题进行一般化,使用 $\mathrm{Max}C_J(m)$ 表示序列 $a[1:m]$ 中连续子序列的最大乘积,则 $\mathrm{Max}C_J(m) = (\mathrm{MAX}\ i,j:1 \leqslant i \leqslant j \leqslant m:(\prod k:i \leqslant k \leqslant j:a[k]))$,令 $M_{ul}(i,j) = (\prod k:i \leqslant k \leqslant j:a[k])$,则 $\mathrm{Max}C_J(m) = (\mathrm{MAX}\ i,j:1 \leqslant i \leqslant j \leqslant m:M_{ul}(i,j))$ {交叉积} $= (\mathrm{MAX}\ j:1 \leqslant j \leqslant m:(\mathrm{MAX}\ i:1 \leqslant i \leqslant j:M_{ul}(i,j)))$. 令 $H(j) = (\mathrm{MAX}\ i:1 \leqslant i \leqslant j:M_{ul}(i,j))$,则 $\mathrm{Max}C_J(m) = (\mathrm{MAX}\ j:1 \leqslant j \leqslant m:H(j))$.

(a) 对 $M_{ul}(i,j)$ 进行变换.

$M_{ul}(i,j) = (\prod k:i \leqslant k \leqslant j:a[k])$ {范围分裂} $= (\prod k:i \leqslant k \leqslant j \wedge k \neq j:a[k])(\prod k:i \leqslant k \leqslant j \wedge k = j:a[k]) = (\prod k:i \leqslant k \leqslant j-1:a[k])(\prod k:k = j:a[k])$ {单一范围} $= (\prod k:i \leqslant k \leqslant j-1:a[k])a[j] = M_{ul}(i,j-1)a[j]$.

(b) 对 $H(j)$ 进行变换.

$H(j) = (\mathrm{MAX}\ i:1 \leqslant i \leqslant j:M_{ul}(i,j))$ {范围分裂} $= \max((\mathrm{MAX}\ i:1 \leqslant i \leqslant j \wedge i \neq j:M_{ul}(i,j)),(\mathrm{MAX}\ i:1 \leqslant i \leqslant j \wedge i = j:M_{ul}(i,j))) = \max((\mathrm{MAX}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j)),(\mathrm{MAX}\ i:i = j:M_{ul}(i,j)))$ {单一范围} $= \max((\mathrm{MAX}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j)),a[j]) = \max((\mathrm{MAX}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j])$.

当 $a[j] \geqslant 0$ 时,$H(j) = \max((\mathrm{MAX}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j]) = \max(H(j-1) \cdot a[j],a[j])$;

当 $a[j] < 0$ 时,$H(j) = \max((\mathrm{MIN}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j])$.

令 $h(j) = (\mathrm{MIN}\ i:1 \leqslant i \leqslant j:M_{ul}(i,j))$,则 $H(j) = \max(h(j-1)a[j],a[j])$.

(c) 对 $h(j)$ 进行变换.

$h(j) = (\mathrm{MIN}\ i:1 \leqslant i \leqslant j:M_{ul}(i,j))$ {范围分裂} $= \min((\mathrm{MIN}\ i:1 \leqslant i \leqslant j \wedge i \neq j:M_{ul}(i,j)),(\mathrm{MIN}\ i:1 \leqslant i \leqslant j \wedge i = j:M_{ul}(i,j))) = \min((\mathrm{MIN}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j)),(\mathrm{MIN}\ i:i = j:M_{ul}(i,j)))$ {单一范围} $= \min((\mathrm{MIN}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j)),a[j]) = \min((\mathrm{MIN}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j])$.

当 $a[j] \geqslant 0$ 时,$h(j) = \min((\mathrm{MIN}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j]) = \min(h(j-1) \cdot a[j],a[j])$;

当 $a[j] < 0$ 时,$h(j) = \min((\mathrm{MAX}\ i:1 \leqslant i \leqslant j-1:M_{ul}(i,j-1))a[j],a[j])$,即 $h(j) = \min(H(j-1)a[j],a[j])$.

由以上(b) 和(c) 的推导可知递推关系式为

$$H(j) = \begin{cases} \max(H(j-1)a[j],a[j]), & a[j] \geqslant 0, \\ \max(h(j-1)a[j],a[j]), & a[j] < 0, \end{cases} \quad (1)$$

$$h(j) = \begin{cases} \min(h(j-1)a[j], a[j]), a[j] \geqslant 0, \\ \min(H(j-1)a[j], a[j]), a[j] < 0. \end{cases} \quad (2)$$

(d) 对 $\mathrm{Max}C_J(m)$ 进行变换.

$$\mathrm{Max}C_J(m) = (\mathrm{MAX}\ j: 1 \leqslant j \leqslant m: H(m)) = \max((\mathrm{MAX}\ j: 1 \leqslant j \leqslant m \wedge j \neq m: H(m)), (\mathrm{MAX}\ j: 1 \leqslant j \leqslant m \wedge j = m: H(m))) = \max((\mathrm{MAX}\ j: 1 \leqslant j \leqslant m-1: H(m)), (\mathrm{MAX}\ j: j = m: H(m))) = \max((\mathrm{MAX}\ j: 1 \leqslant j \leqslant m-1: H(m)), H(m)) = \max(\mathrm{Max}C_J(m-1), H(m)). \quad (3)$$

(iii) 问题的求解算法.

在(1) ~ (3) 式这 3 个递推关系的基础上确定递推的初始条件和终止条件,获得问题的求解算法.

BEGIN: $m = 2$; $\mathrm{Max}C_J(1) = H(1) = h(1) = a[1]$;

TERMINATION: $m = n + 1$;

RECUR:

$$H(m) = \begin{cases} \mathrm{Max}(H(m-1)a[m], a[m]), a[m] \geqslant 0, \\ \mathrm{Max}(h(m-1)a[m], a[m]), a[m] < 0; \end{cases}$$

$$h(m) = \begin{cases} \mathrm{Min}(h(m-1)a[m], a[m]), a[m] \geqslant 0, \\ \mathrm{Min}(H(m-1)a[m], a[m]), a[m] < 0; \end{cases}$$

$\mathrm{Max}C_J(m) = \mathrm{Max}(\mathrm{Max}C_J(m-1), H(m))$;

END.

(iv) 获取问题的算法程序.

通过以上推导的递推关系可知,$H(m)$ 和 $h(m)$ 的值分别利用 $H(m-1)$ 和 $h(m-1)$ 的值进行求解,其中 m 的变化范围为 $1 \leqslant m \leqslant n$,因而可以从最小的子问题 $m = 2$ 出发,反复利用该递推式,逐步扩大 m 的值,求出规模较大子问题的解,直到最终求出原问题的解.

在算法具体实现时,使用数组 $H[n]$ 的每个元素依次记录子问题 $H(m)$ 的值,用数组 $h[n]$ 的每个元素依次记录子问题 $h(m)$ 的值,用变量 $\mathrm{Max}C_J$ 来记录子问题 $\mathrm{Max}C_J(m)$ 的值,即 $\mathrm{Max}C_J = \mathrm{Max}(\mathrm{Max}C_J, H[n])$ 表示连续子序列的最大乘积;然后根据上述递推关系,借助循环结构,不断由规模较小子问题的解求出规模较大子问题的解,最终求出原问题的解. 具体算法程序如下:

```
H[1] = h[1] = a[1];
Double MaxC_J;
for(int m = 2; m <= n; m++){
    if(a[m] >= 0){
    H[m] = Max(H[m-1] * a[m], a[m]);
    h[m] = Min(h[m-1] * a[m], a[m]);
    }
    else{
    H[m] = Max(h[m-1] * a[m], a[m]);
    h[m] = Min(H[m-1] * a[m], a[m]);
    }
MaxC_J = Max(MaxC_J, H[n]);
}.
```

由于以上算法程序完全是建立在递推关系的基础上实现的,而递推关系是之前经过严格数学推导得到的,因而有效地保证了以上算法程序的正确性.

2.2 统一求解连续子序列问题的策略

根据上述的推导,可以得到以下求解递推子序列问题的统一求解策略:

(i) 利用程序规约描述语言形式化地描述相关问题 P 的程序规约 $\langle A_Q, A_R \rangle$;

(ii) 分划原问题并利用范围分裂、单一范围、交叉积等量词性质对问题的程序规约进行数学等价变换获取递推关系 f: $P(i+1) = f(P(i), \Delta P_i)$,其中 $P(i)$ 是 $P(i+1)$ 的子问题,由 $P(i)$ 和 ΔP_i 通过递推关系 f 可以获得 $P(i+1)$ 的解;

(iii) 对于 ΔP_i 部分,需分不同情况使用规则进行等价变换:

若约束条件为 C_{C_1},则 ΔP_i 变换成 ΔP_{i1};

若约束条件为 C_{C_2},则 ΔP_i 变换成 ΔP_{i2};

…

若约束条件为 C_{C_n},则 ΔP_i 变换成 ΔP_{in};

其中 $C_{C_1} \vee C_{C_2} \vee \cdots \vee C_{C_n} = \mathrm{true}$;

(iv) 由以上(ii) 和(iii) 可获得递推关系式为

$$P(i+1) = \begin{cases} f(P(i), \Delta P_{i1}), \text{约束条件为 } C_{C_1}, \\ f(P(i), \Delta P_{i2}), \text{约束条件为 } C_{C_2}, \\ \cdots \\ f(P(i), \Delta P_{in}), \text{约束条件为 } C_{C_n}. \end{cases}$$

(v) 通过以上递推关系式,确定递推关系的起止条件,获取问题的求解算法;

(vi) 用变量来记录递推关系式中子问题的解,根据(v) 中的算法,借助循环结构,编写问题求解的算法程序.

实践证明,该统一求解策略还可用于最大连续子段和、最长公共子序列、最长连续上升子序列以及最大连续回文子串等连续子序列问题的算法形式化推导.

3　斯特林数变形问题的算法形式化推导

斯特林数[2](Stirling 数)是组合数学中的一个重要内容,具有重要的实际应用意义. 它由 18 世纪的苏格兰数学家 James Stirling 首先发现并说明了它们的重要性. 斯特林数主要处理的是把 N 个不同的元素分成 K 个集合或环的个数问题. 现今有第 1 类斯特林数和第 2 类斯特林数,有关斯特林数存在许多变形问题,本小节主要研究第 2 类斯特林数变形问题的算法形式化推导.

3.1　关于第 2 类斯特林数变形问题

问题描述: 给定 N 本不同的书,M 个不同的抽屉(抽屉足够大),要求将这 N 本书放入到 M 个抽屉中,并且要求每个抽屉不能为空,求共有多少种方案?

(i) 形式化描述问题的程序规约.

用 $a_{ns}(N,M)$ 来表示 N 本不同的书放入到 M 个不同的抽屉中的总方案数.

A_Q: 已知 N 本不同的书和 M 个不同的抽屉.

A_R: $a_{ns}(N,M) = (\sum h:(k:1\leqslant k\leqslant N:h[k])\in\{1,\cdots,M\})\wedge(\bigcup k:1\leqslant k\leqslant N:h[k]) = \{1,\cdots,M\}:1)$,其中 $1\sim M$ 为编号不同的抽屉,在 A_R 中出现的$(\bigcup k:1\leqslant k\leqslant N:h[k])$ 表示求 $h[k]$ 的并集,而 $h[k]$ 表示编号为 k 的书所放入的抽屉编号.

(ii) 分划原问题并求解递推关系.

将问题进行一般化表示,$a_{ns}(u,v)$ 表示将 u 本不同的书放入 v 个不同的抽屉中的方案数,则

$$a_{ns}(u,v) = (\sum h:(k:1\leqslant k\leqslant u:h[k])\in\{1,\cdots,v\})\wedge(\bigcup k:1\leqslant k\leqslant u:h[k]) = \{1,\cdots,v\}:1).$$

由于需要将 N 本不同书放入 M 个不同抽屉中,需要把问题分成 3 步来求解:

Step 1　把 u 本不同的书分成 v 堆,方案数记为 $a'_{ns}(u,v) = (\sum h:(k:1\leqslant k\leqslant u:h[k])\in\{1,\cdots,v\})\wedge(\bigcup k:1\leqslant k\leqslant u:h[k]) = \{1,\cdots,v\}:1)$;

Step 2　把 v 个不同的抽屉进行排列组合,与第 1 步中得到的堆进行一一配对,排列数记为 $f_{act}(v)$;

Step 3　根据组合数学乘法原理可知,将 u 本不同的书放入到 v 个不同抽屉的总方案数为 $a_{ns}(u,v) = a'_{ns}(u,v)f_{act}(v)$.

首先求解第 1 步的 $a'_{ns}(u,v)$.

(a) 假设将第 u 本书单独分成一个堆,由于这 v 个堆可视为等同的,从而可假设 $h[u] = v$,则

$$a'_{ns}(u,v) = (\sum h:\{\forall k:1\leqslant k\leqslant u:h[k]\in\{1,\cdots,v\})\wedge(\bigcup k:1\leqslant k\leqslant u:h[k]) = \{1,\cdots,v\}:1) = (\sum h:((\forall k:1\leqslant k\leqslant u\wedge k\neq u:h[k]\in\{1,\cdots,v\})\wedge(\forall k:1\leqslant k\leqslant u\wedge k = u:h[k]\in\{1,\cdots,v\}))\wedge((\bigcup k:1\leqslant k\leqslant u-1:h[k])\cup\{h[u]\}) = (\{1,\cdots,v-1\}\cup\{v\}):1) = (\sum h:(\forall k:1\leqslant k\leqslant u-1:h[k]\in\{1,\cdots,v\})\wedge h[u]\in\{1,\cdots,v\}\wedge((\bigcup k:1\leqslant k\leqslant u-1:h[k])\cup\{h[u]\}) = (\{1,\cdots,v-1\}\cup\{v\}):1).$$

由于 $h[u] = v$,因此 $h[u]\in\{1,\cdots,v\}$ 为 true;又由于第 u 本书单独分成编号为 v 的堆,其他 $u-1$ 本书分成 $v-1$ 堆,则

$$a'_{ns}(u,v) = (\sum h:(\forall k:1\leqslant k\leqslant u-1:h[k]\in\{1,\cdots,v-1\})\wedge(\bigcup k:1\leqslant k\leqslant u-1:h[k]) = \{1,\cdots,v-1\}:1) = a'_{ns}(u-1,v-1).$$

(b) 设第 u 本书不单独分成一个堆,可以和其他书共同分到某一堆中,即 $h[u] = 1\vee h[u] = 2\vee\cdots\vee h[u] = v$,则

$$a'_{ns}(u,v) = (\sum h:(\forall k:1\leqslant k\leqslant u:h[k]\in\{1,\cdots,v\})\wedge(\bigcup k:1\leqslant k\leqslant u:h[k]) = \{1,\cdots,v\}:1) = (\sum h:((\forall k:1\leqslant k\leqslant u\wedge k\neq u:h[k]\in\{1,\cdots,v\})\wedge(k:1\leqslant k\leqslant u\wedge k = u:h[k]\in\{1,\cdots,v\}))\wedge((\bigcup k:1\leqslant k\leqslant u-1:h[k])\cup\{h[u]\}) = (\{1,\cdots,v-1\}\cup\{v\})\wedge(h[u] = 1\vee h[u] = 2\vee\cdots\vee h[u] = v):1).$$

由于第 u 本书不单独分成一个堆,则它和其他的书共同分成一堆,所以 $1\sim(u-1)$ 本书必须要分到 v 堆中,则

$$a'_{ns}(u,v) = (\sum h:(\forall k:1\leqslant k\leqslant u:h[k]\in\{1,\cdots,v\})\wedge((\bigcup k:1\leqslant k\leqslant u-1:h[k]) = \{1,\cdots,v\})\wedge(h[u] = 1\vee h[u] = 2\vee\cdots\vee h[u] = v):1) = (\sum h:((\forall k:1\leqslant k\leqslant u\wedge k\neq u:h[k]\in\{1,\cdots,v\})\wedge(k:1\leqslant k\leqslant u\wedge k = u:h[k]\in\{1,\cdots,v\}))\wedge((\bigcup k:1\leqslant k\leqslant u-1:h[k]) = \{1,\cdots,v\})\wedge(h[u] =$$

$1 \vee h[u] = 2 \vee \cdots \vee h[u] = v)):1) = (\sum h:(\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge h[u] \in \{1,\cdots,v\} \wedge ((\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\}) \wedge (h[u] = 1 \vee h[u] = 2 \vee \cdots \vee h[u] = v)):1) = (\sum h:((\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge h[u] \in \{1,\cdots,v\} \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = 1)) \vee ((\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge h[u] \in \{1,\cdots,v\} \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = 2)) \vee \cdots \vee ((\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge h[u] \in \{1,\cdots,v\} \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = v)):1)$.

由于 $h[u] = 1 \vee h[u] = 2 \vee \cdots \vee h[u] = v$, $h[u] \in \{1,\cdots,v\}$ **为** true,**则**

$$a'_{ns}(u,v) = (\sum h:(\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = 1):1) + (\sum h:(\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = 2):1) + \cdots + (\sum h:(\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\} \wedge (h[u] = v):1).$$

由于 $h[u] = 1, h[u] = 2, \cdots, h[u] = v$ **的情况可以视为等同,则**

$$a'_{ns}(u,v) = v(\sum h:(\forall k:1 \leqslant k \leqslant u-1:h[k] \in \{1,\cdots,v\}) \wedge (\bigcup k:1 \leqslant k \leqslant u-1:h[k]) = \{1,\cdots,v\}:1) = v\,a'_{ns}(u-1,v).$$

因为以上的 2 **种假设情况是相互独立的,根据组合数学中的加法原理可以得到递推关系式**

$$a'_{ns}(u,v) = a'_{ns}(u-1,v-1) + va'_{ns}(u-1,v). \quad (4)$$

接下来根据第 2 **步,要求对** v **个抽屉进行排列,即求** v **的阶乘. 由于求解阶乘的递推关系比较简单,因此推导过程省略,直接给出其递推关系为**

$$f_{act}(v) = (\prod i:1 \leqslant i \leqslant v:i) = vf_{act}(v-1). \quad (5)$$

最后根据第 3 **步,将** $a_{ns}(u,v) = a'_{ns}(u,v) \cdot f_{act}(v)$ **的解求出,即为将** u **本不同的书放入** v **个不同的抽屉的方案数.**

(iii) **求解算法.**

在得到以上递推关系的基础上确定递推的初始条件和终止条件,得到问题的求解算法.

```
BEGIN:
    u = 1; v = 1;
    f_act[0] = 1; a'_ns(1,1) = 1;
    a'_ns(u,u) = 1, a'_ns(u,0) = 0 (1 ≤ u ≤ N);
TERMINATION:
    (u = N + 1) ∧ (v = M + 1 ∨ v = u + 1);
RECUR:
    a'_ns(u,v) = a'_ns(u - 1,v - 1) + v* a'_ns(u -
     1,v);
    f_act(v) = f_act(v - 1) * v;
    a_ns(u,v) = a'_ns(u,v) * f_act(v);
END.
```

(iv) **获取问题的算法程序.**

通过以上推导的递推关系可知, $a'_{ns}(u,v)$ **的值可以利用** $a'_{ns}(u-1,v-1)$ **和** $a'_{ns}(u-1,v)$ **的值进行求解,其中** u **和** v **的变化范围分别为** $1 \leqslant u \leqslant N, 1 \leqslant v \leqslant M$. **因而可以从最小子问题** $u = 2$ **和** $v = 1$ **出发,反复利用递推式,逐步扩大** u **和** v **的值,不断求出规模较大子问题的解,直到最终求出原问题的解.**

在算法具体实现时,使用 2 **维数组** $a'_{ns}[u][v]$ **的每个元素依次记录子问题** $a'_{ns}(u,v)$ **的值,用变量** a_{ns} **来记录子问题** $a_{ns}(u,v)$ **的值,即** $a_{ns} = a'_{ns}(u,v) f_{act}(v)$ **表示斯特林数变形问题;然后根据上述递推关系,借助循环结构,不断由规模较小子问题的解求出规模较大子问题的解,最终求出原问题的解. 具体算法程序如下:**

```
…
long a_ns;
f_act[0] = 1; a'_ns(1,1) = 1;
for(int u = 1; u ≤ N; u ++){
    a'_ns[u][u] = 1;
    a'_ns[u][0] = 0;
}
for(int u = 2; u ≤ N; u ++){
    for(int v = 1; v ≤ u && v ≤ M; v ++){
        a'_ns[u][v] = a'_ns[u - 1][v - 1] +
            v* a'_ns[u - 1][v];
    }
}
for(int v = 1; v ≤ M; v ++)
    f_act[v] = f_act[v - 1] * v;
a_ns = a'_ns[N][M] * f_act[M];
…
```

由于以上算法程序完全是建立在递推关系的基础上实现的，而递推关系是之前经过严格数学推导得到的，因而有效地保证了以上算法程序的正确性.

3.2　统一求解斯特林数变形问题的策略

(i) 利用程序规约描述语言来形式化地描述相关问题 P 的程序规约 $\langle A_Q, A_R \rangle$；

(ii) 划分原问题并寻找问题求解递推关系：

(a) 在原问题中某个元素固定的条件下，对 P 进行数学等价变换，根据组合数学的乘法原理可获得递推关系为 $P_1(i+1) = xf(P_1(i))$，其中 $P_1(i)$ 是对原问题进行划分后获得的子问题，$x > 0$ 是子问题的相关系数；

(b) 在原问题中某个元素不固定(即与其它元素混为一体) 的条件下，对 P 进行数学等价变换，根据组合数学的乘法原理可获得递推关系为 $P_2(i+1) = yf(P_2(i))$，其中 $P_2(i)$ 是对原问题进行划分后获得的子问题，$y > 0$ 是子问题的相关系数；

(c) 根据组合数学加法原理，获得最终的递推关系：$P(i+1) = P_1(i+1) + P_2(i+1)$.

(iii) 根据递推关系式，确定递推的起始条件和终止条件，获得问题的求解算法.

(iv) 用变量来记录递推关系式中子问题的解，根据(iii) 中的算法，借助循环结构，编写问题求解的算法程序.

研究表明，该统一求解策略可推广至卡特兰(Catalan) 数、阶梯问题和 Bell 数的求解以及更多斯特林数变形问题的求解.

4　总结和展望

本文采用基于递推技术的形式化推导方法求解了 2 个典型的组合数学问题实例，详细展示了从求解问题的程序规约到算法程序的推导过程. 基于递推技术的形式化推导方法的意义在于把算法的设计工作从单纯依靠程序员技能与灵感的思维活动转化为规范化生成活动，不仅清晰地展示了算法设计过程，而且确保最终算法程序的正确性. 该方法已经应用到组合数学的许多经典问题中，典型的有汉诺塔问题、0-1 背包及背包的变形问题、大数和问题、不定积分等各种问题，因此其在现今科学界的应用已经相当地广泛.

组合数学问题是算法程序开发中的难点问题，而在组合数学问题的算法推导过程中，问题规约的形式化描述以及递推关系的求解是关键，同时也存在诸多难点，这将是今后进一步探索和努力的方向.

5　参考文献

[1] 高逸人. 组合数学在软件工程领域中的应用研究 [J]. 科技与创新，2017(23)：143-144.

[2] 布鲁迪. 组合数学 [M]. 冯速，译. 5 版. 北京：机械工业出版社，2012.

[3] 孙凌宇. PAR 方法在组合数学问题中的应用研究 [D]. 南昌：江西师范大学，2005.

[4] 石海鹤，石海鹏，薛锦云. 形式化开发若干组合数学问题的算法 [J]. 江西师范大学学报：自然科学版，2006，30(5)：423-427.

[5] 游颖. 算法形式化方法在三类组合数学问题求解中的应用研究 [D]. 南昌：江西师范大学，2017.

[6] 游颖，杨庆红，齐蕾蕾. 3 个变形背包问题的形式化推导 [J]. 江西师范大学学报：自然科学版，2017，41(2)：116-121.

[7] 张园，杨庆红，胡昊. 基于递推技术的算法设计方法的应用研究 [J]. 计算机与现代化，2012(6)：37-39.

[8] 单学广. 基于递推技术的算法程序设计方法的研究与应用 [D]. 南昌：江西师范大学，2011.

[9] 刘峰涛，贺国光. 基于 L-Z 方法的宏观交通运输系统复杂性测度 [J]. 哈尔滨工业大学学报，2008，40(12)：2058-2061.

[10] Larsen P G，Fitzgerald J. The evolution of VDM tools from the 1990s to 2015 and the influence of CAMILA [J]. Journal of Logical and Algebraic Methods in Programming，2016，85(5)：985-998.

[11] 丁湘陵，王志刚. 基于 B 方法的体系结构描述语言的形式化研究 [J]. 计算机工程与科学，2013，35(1)：100-106.

[12] 黄昱，马殿富，赵永望，等. 一种基于 B 方法的形式化开发方法 [C] // 全国抗恶劣环境计算机第二十四届学术年会论文集. 北京：北京航空航天大学出版社，2014：278-287.

[13] 胡启敏，薛锦云，游珍，等. PAR 平台中若干软件构件形式化验证技术研究 [J]. 计算机工程与科学，2018，40(2)：268-274.

[14] 胡启敏，薛锦云. 若干算法程序的形式化推导与生成技术研究 [J]. 计算机研究与发展，2008，45(S1)：148-153.

[15] You Zhen，Xue Jinyun，Zuo Zhengkang. Unified formal derivation and automatic verification of three binary-tree traversal non-recursive algorithms [J]. Cluster Computing，2016，19(4)：2145-2156.

The Formal Derivation for Two Kinds of Combinatorial Mathematical Problems

XIONG Xiaochao, YANG Qinghong*
(College of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

Abstract: The research of combinatorial mathematics problem algorithm is an important research content of computer science. However, in many related literatures, most of the algorithms for combining mathematics problems are obtained through simple analysis. The detailed design process of the algorithm program is not given, which leads the reader to fail to understand the essence of the algorithm. It is impossible to guarantee the correctness of the algorithm program. The maximum product of continuous subsequences and the deformation problem of the second type of Stirling numbers in combinatorial mathematics are taken as an example. Based on this example, a series of equivalent transformations are performed on the program specification to obtain the problem solving by formalizing the program specification of the problem, using the rule transformation rules. The recursive formula of the sequence, based on which the algorithm program of the problem solving is obtained, clearly shows the requirements from the problem to the detailed derivation process of the algorithm program, and through the further in-depth study of the related combinatorial mathematics, two types of combinations are extracted. The solution strategy of mathematical problems provides an effective way to improve the correctness of the algorithm program of combinatorial mathematics.
Key words: formal method; program specification; combinatorial mathematics; recursive technique

(责任编辑: 冉小晓)

(上接第 360 页)

The Research on the Default Problem of Supply Chain Enterprises Based on Accounts Receivable Financing

ZHOU Yongsheng[1], CUI Jiali[1], LIU Xinrui[2], YANG Xiaolin[1]
(1. Business School, Beijing Technology and Business University, Beijing 100048, China;
2. China International Capital Corporation Limited, Beijing 100020, China)

Abstract: An evolutionary game model for the core enterprises and small and medium-sized enterprises default problem in the supply chain finance accounts receivable financing mode is set up, the evolutionary stable strategy of two parties involved in the subject is analyzed, aimed at improving bank, core enterprises and small and medium-sized enterprises mutual benefits. The study shows that the probability that core enterprises and small and medium-sized enterprises adopt (honesty, honesty) strategy is positively correlated with the external income brought by good business reputation, the enterprise's default penalties, additional benefits of supply chain enterprises brought by long-term cooperation in the supply chain. The probability that core enterprises and small and medium-sized enterprises adopt (honesty, honesty) strategy is negatively correlated with bank's lending rate, the total amount of receivable accounts, the receivable mortgage rate. The proposed conclusions are verified through a case study, and the relevant recommendations are provided for the banks and supply chain enterprises.
Key words: accounts receivable financing; supply chain finance; default problem; game analysis

(责任编辑: 曾剑锋)