文章编号:1674-2869(2010)11-0091-03

软件进程分解方法模式

庄 芸[1],李 晖[2]

(1.武汉工程大学环境与城市建设学院,湖北 武汉 430074;
2.武汉大学计算机学院,湖北 武汉 430072)

摘 要:建立了对软件进程模块 M 的一种规范化的分解方法,这种方法视为分解 M 的一种方法模式,称为矩阵模式,表示为 $A=[w,t]$. 这个模式把 M 的递归过程的理论分析和实际展开结合于一体,为它的递归展开过程形成有效的条件,生成 M 的优化的计算原型 Pri-cTr.

关键词:分解模式;计算原型;复杂性;计算原型集合

中图分类号:TU375.1 文献标识码:A doi:10.3969/j.issn.1674-2869.2010.11.024

0 引 言

对软件体系结构形式化一直都是软件工程的一个重要的研究和应用领域[1-5]. 本文的目的在于:依据分解运算集合 ρ_θ 等效划分原理,综合分析原型树 Pri-cTr 五个基本结构参量特征和它们之间的关系,最终综合它们于一个框架;这个框架形成软件进程模块[6] M 一般的分解模式.

文献[6]的推论6说明:任何一个软件进程模块 $M(\theta)(\theta\in ASet_{eq}(M))$,关于算子 θ 的分解运算集合 ρ_θ 可由一个值区间 $VL(O(f))$ 等效划分.

$$\left.\begin{aligned}&\frac{\rho_\theta}{VL(O(f))}=\{[\rho_j]\rho_j(\theta)\in\rho_\theta\}\\&\forall\rho_i\in\lceil\rho_j\rceil\Rightarrow O(f_i)\equiv O(f_j)\end{aligned}\right\}\qquad(1)$$

组合函数 f_i,f_j 分别对应两个分解运算 $\rho_i,\rho_j\in\rho_\theta$. $VL(O(f))$ 是集合 ρ_θ 上所有分解运算 ρ_i 对应组合函数 f_i 复杂性值集合. 这是一个实数区间. 如果在区间有些值 z 不存在有 ρ_θ 的组合函数的复杂性值与之相等则记为 $[\Phi]=z$. 此外,虽然在该区间有些值很大甚至无穷,但这些值只要有对应的组合函数复杂性值,则对应的等效子集 $[\rho_j]$ 就一定存在,$\lceil\rho_j\rceil\neq\Phi$. 这个事实,概括为如下的推论.

推论1 任何一个软件进程模块 $M(\theta)(\theta\in ASet_{eq}(M))$,

$$\exists\lceil\rho_j\rceil\in\frac{\rho_\theta}{VL(O(f))}\Rightarrow|\lceil\rho_j\rceil|\geqslant 1\qquad(2)$$

这个推论的正确性可由软件工程提供的一组曲线图说明. 这组曲线图如下:

如图1所示的组合函数 $f_j(\theta_1,\theta_2,\cdots,\theta_k)$ 对应分解运算 $\rho_j(\theta)=(\theta_1,\theta_2,\cdots,\theta_k)$,由组合-分解一致性原则,它的的复杂性 $O(f_j)$ 就是算子 θ 的复杂性,有下面复杂性关系式.

$$\begin{aligned}&O(f_j(\theta_1,\theta_2,\cdots,\theta_k))=O(\theta)\\&O(f_j)=O(\theta_i)_M\times O(f_jl)\\&O(\theta_i)_M=\mathrm{Max}\,O(\theta_i)\{1\leqslant i\leqslant k\}\end{aligned}\qquad(3)$$

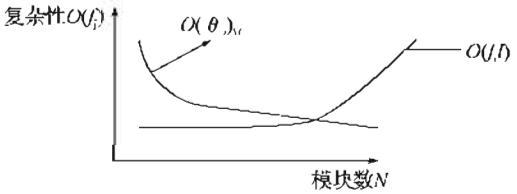

图1 软件进程分解模块数与复杂性

Fig.1 The number of software process modules and complexity

由图1可以看出:

1. 随着模块数 N 的增加,分解运算 $\rho_j(\theta)$ 的最大子算子的复杂性 $O(\theta_i)_M$ 逐步下降,而它的组合函数 f_j 的连接复杂性 $O(f_jl)$ 却逐步上升.

2. $O(\theta_i)_M$ 与 $O(f_jl)$ 两者的反向运动,在某些时候它们变化的绝对值可能是相等的,从而保持它们的积 $O(f_j)$ 不变或近似.

上面证明了式(1)的在正确性. 对于一个软件进程模块 $M(\theta)(\theta\in ASet_{eq}(M))$,它的算子 θ 的分解运算集合[7] ρ_θ 中,不同的分解运算 ρ_i,ρ_j 它们对应的组合函数的结构虽然不同,但它们的复杂性可能相等或近似. 值得指出的是:(3)式中的 $O(\theta)$ 是算子 θ 经过 $f_j-\rho_j(\theta)$ 处理后形成的复杂性.

1 原型树的复杂性

组合函数 f_j 的复杂性公式(3)的因子 $O(\theta_i)_M$

收稿日期:2010-07-09

作者简介:庄 芸(1969-),女,湖北武汉人,讲师,硕士. 研究方向:结构工程与计算机应用.

是子算子 θ_i 的复杂度，它在 $\rho_j(\theta)$ 形成的所有子算子中具有最大值，$\rho_j(\theta)$ 是 f_j 对应的分解运算. $O(\theta_i)_M$ 引导模块 $M(\theta)$ 的垂直分解倾向. 事实上，子算子 θ_i 的复杂性同样可以表达成式(3)；随着算子 θ 的递归展开，各个子算子也将逐层同时递归展开，但算子 θ 的最终递归深度将取决于子算子 θ_i 的抽象程度，也就是该子算子的递归深度. 抽象化方法是化解软件模块复杂性的最基本的方法，算子的递归深度描述了该算子的抽象度. 事实上，算子 θ_i 的抽象度越高，它要求更多次的递归实现抽象分解，则意味着复杂性越大. 这同样意味着，一个软件进程模块 $M(\theta)(\theta \in ASet_{eq}(M))$，对于算子 θ 的任何一个实际的计算原型 Pri-cTr，其递归展开表达式为

$$\rho^{t}(\theta) = \text{Pri-cTr}(\theta, t) \tag{4}$$

$$\forall \rho_j(\theta) \in \rho_\theta \exists \rho_j(\theta) = (\theta_1, \theta_2, \cdots, \theta_k) \Rightarrow \exists O(\theta_i)_M \lceil \theta_i \in \rho_j(\theta) \rceil = \text{Max}\, O(\theta_i) \{1 \leqslant i \leqslant k\} \tag{5}$$

类似地，算子 θ_i 的递归展开表达式为

$$\rho^{\alpha}(\theta_i) = \text{Pri-cTr}(\theta_i, t_i) \tag{6}$$

在这里

$$t = t_1 + 1 \tag{7}$$

递归深度 $t \in C_{cons} = \{mn, w, t, C_{in}, C_{out}\}$（$C_{cons}$ 称为 Pri-cTr 的结构参量），又称为计算原型 Pri-cTr(θ, t) 的抽象度，正是按垂直方向逐层化解该模块的复杂性的层次数. 在此时，子算子 θ_i 因为具有最大复杂度 $O(\theta_i)_M$，所以它的递归深度值

$$t_i = t - 1 \tag{8}$$

$\rho_j(\theta)$ 形成的其它子算子的递归深度值，一般有

$$\forall \theta_r \in \rho_\theta (r \neq i) \Rightarrow t_r \leqslant t - 1 \tag{9}$$

这说明：其它子算子 $\theta_r \in \rho_\theta$ 到达 t 层之前或至多到达 t 层时递归过程结束.

下面分析原型树 Pri-cTr(θ, t) 各个层次[8]的复杂性.

原型树 Pri-cTr(θ, t) 的宽度 $w \in C_{cons}$ 说明：在 Pri-cTr(θ, t) 中各层能够达到的最大节点数. 由于原型树 Pri-cTr(θ, t) 的特殊结构，每层的节点 θ_j，或为原子节点，或为也引导一棵子原型树 Pri-cTrj(θ_j, t_j) 的复合节点，如式(6)所示. 该层的所有子节点有它们自身的复杂性，它们在该层的连接复杂性；还有在它们自身复杂性中存在有最大复杂性. 一般来说，各层节点通过上辈的多个子计算原型树分成若干子集横向分布于这一层. 因此，此层的连接复杂性将会大大降低. 如在 Pri-cTr(θ, t) 的 i 层，它的宽度记为 $w_i \leqslant w$，则上辈的各个子 Pri-cTrj(θ_j, t_j) 通过 i 层时的宽度记为 w_{ij}，于是 w_i 被分解成若干(如 k)个子集，并且有，

$$w_i = \sum_j = 1^k w_{ij} \tag{10}$$

此时，Pri-cTr(θ, t) 在 i 层的连接复杂性有下面的关系表达式，

$$O(f_i l) = \sum_{j-1}^{k} O(f_{ij} l) \tag{11}$$

其中 $O(f_i l)$ 是对应 w_{ij} 的连接复杂性. 显然，下面的表达式是正确的，

$$O(f_i l) = O(f_{ij} l)_M \tag{12}$$

$$O(f_{ij} l)_M = \text{Max}\, O(f_{ij} l) \{1 \leqslant j \geqslant k\} \tag{13}$$

关于原型树 Pri-cTr，它的任意层的复杂性公式以一个推论表述如下.

推论 2　一个软件进程模块 $M(\theta)(\theta \in ASet_{eq}(M))$，对于算子 θ 的任何一个实际计算原型 Pri-cTr(θ, t)，它的任意层的复杂性公式为

$$\left.\begin{aligned} &\forall i \in [1, t] \Rightarrow O(f_i) = O(\theta_{ir})_M \times O(f_{ij} l)_M \\ &O(\theta_{ir})_M = \text{Max}\, O(\theta_i) \{1 \leqslant r \leqslant w_i, w_i \leqslant w\} \\ &O(f_{ij} l)_M = \text{Max}\, O(f_{ij} l) \{1 \leqslant j \leqslant k, r = / \neq j\} \end{aligned}\right\} \tag{14}$$

式(14)中，w_i 是第 i 层的宽度，k 表示通过该层的子原型树数，$t \in C_{cons}$ 为 Pri-cTr(θ, t) 的深度，$w \in C_{cons}$ 为它的宽度.

首先，推论 2 说明：Pri-cTr(θ, t) 在任意层的连接复杂性为 $O(f_{ij} l)M$ 只取决于该层某个最大的子集，也就是该子集上辈节点的扇出值. 其次，Pri-cTr(θ, t) 的参量 $w \in C_{cons}$ 及其相关推论 2 表达了模块 $M(\theta)(\theta \in ASet_{eq}(M))$ 横向(水平)抽象分解，参量 $t \in C_{cons}$ 则表达了它的纵向(垂直)抽象分解. 该模块最终分解的模块数(Pri-cTr(θ, t) 的节点数) $mn \in C_{cons}$ 则取决于 t 和 w 两个参量综合作用的结果.

2　软件进程的分解模式

对于软件进程模块 $M(\theta)(\theta \in ASet_{eq}(M))$，它的任意一个计算原型树 Pri-cTr$(\theta, t)$ 的生成过程说明：在分解运算的连续递归作用下，这个计算原型按水平和垂直两个方向逐层展开. 同时根据上面对 Pri-cTr(θ, t) 的结构参量 C_{cons} 的基本分析，它的两个因子 $w, t \in C_{cons}$ 在这个递归展开过程中有着重要的作用和意义. 这两个因子参量正好刻画了计算原型按水平和垂直两个方向的深度和宽度. 它们的适当地选取有可能构成一个等效的面积，指导着该过程有效地展开. 这个面积适当的变化(长与宽相对变化，保持面积大小不变，)，尽管该面积里包含的模块数 mn 可能有变化，但保持它们对应的计算原型树 Pri-cTr(θ, t) 的复杂性不变.

这就使得这两个因子形成的矩形，形同一个框架，这个框架把 C_{cons} 的四个因子 $mn, w, t,$ 和 C_{out} 有机地联系在一起. 同时，由于 Pri-cTr(θ,t) 是树结构，它的扇入值是一个常量，$C_{in}=1$. 所以这个框架实际上综合引导 Pri-cTr(θ,t) 递归展开过程：在确定复杂性要求时，在一个确定的矩形面积里，指导 $M(\theta)$ 软件实体分解递归过程[9]. 这个模式也就称为矩阵分解模式，下面给这个模式正式定义.

定义 1　对于一个可求解问题 P，它的软件进程模块 $M(\theta)(\theta\in ASet_{eq}(M))$ 的矩阵分解模式记为 A.

$$A=[w,t]w,t\in C_{cons}=\{mn,w,t,C_{in},C_{out}\} \tag{15}$$

其中：

1）参量 t 称为计算原型 Pri-cTr(θ,t) 的（递归）深度，满足条件，

$$\rho'(\theta)=\text{Pri-cTr}(\theta,t)=\text{Tree}(\theta,D,E) \tag{16}$$

θ 是树 Tree 的根节点，D 是它的节点集，E 节点之间连接边集；

2）参量 w 称为计算原型 Pri-cTr(θ,t) 的宽度，是该原型中最宽层的节点数；

3）C_{cons} 称为计算原型 Pri-cTr(θ,t) 的结构参量，mn 是节点（模块）数，即

$$mn=|D|,D\subset\text{Pri-cTr}(\theta,t) \tag{17}$$

4）C_{in},C_{out} 分别是计算原型 Pri-cTr(θ,t) 扇入和扇出值，并规定原型中任何一个节点的扇入和扇出值分别不超过 C_{in},C_{out}.

矩阵分解模式 A 的意义在于把软件进程模块 $M(\theta)(\theta\in ASet_{eq}(M))$ 的分解函数与它的计算原型 Pri-cTr(θ,t) 的结构参量融合为一体，确立了 $M(\theta)$ 进行理论分析和分解运算的制约条件；通过矩阵 A 参量的调整，引导递归分解过程优化运行并取得最佳的分解效果，即生成优化的计算原型 Pri-cTr(θ,t).

3　结　语

以上建立了对软件进程模块 M 的一种规范化的分解方法，这种方法视为分解 M 的一种方法模式——矩阵模式，表示为 $A=[w,t]$. 本文说明了 A 形成的基因及其某些特征. 这个模式把 M 的递归过程的理论分析和实际展开结合于一体，为它的递归展开过程形成有效的条件，生成 M 的优化的计算原型 Pri-cTr.

在 A 的基础上，将继续深入分析了软件进程模块 M 分解运算及其生成的计算原型集合的若干重要性质. 如：1 计算原型的复杂性；2 等面积计算原型子族；3 计算原型集合 Set-Pri-cTr(M) 生成表达式和它的等效划分理论；4 等面积计算原型子族 ArPrTset$[w,t]$.

软件分解历来是软件学界关心的问题，本文的工作主要是希望用数学语义来描述软件的开发过程，揭示其内在的规律，使软件的开发不再是一种经验和试探，而是在一种规则和模式的指导下开发出精确而有效的系统，当然这需要很大量研究和探索工作，还有不少问题等待进一步分析与研究.

参考文献：

[1] LauKung-Kiu, Wang Zheng. Software component models [J]. IEEE Transactions on Software Engineering, 2007, 33(10): 37－45.

[2] Broy M. Taward a mathematical foundation of software engineering methods [J]. IEEE Transactions On Software Engineering, 2001, 12(3): 21－57.

[3] 王忠，王春丽，刘莉. 基于 SVM 的多类分类算法改进[J]. 武汉工程大学学报，2010，32(7)：89－93.

[4] 田裕康，胡荣强. 可变时延网络控制系统的建模和稳定性分析[J]. 武汉工程大学学报，2010，32(7)：94－98.

[5] 赵会群，王国仁，高远. 软件体系结构抽象模型[J]. 软件学报，2002，25(7)：730－736.

[6] 李晖，庄芸. 软件体系结构的数学论域[J]. 武汉大学学报：工学版，2003，36(4)：107－110.

[7] 庄芸，李晖. 软件体系结构的数学描述[J]. 计算机应用研究，2005，22(2)：492－493.

[8] 李晖，庄芸. 软件体系结构层次模型[J]. 计算机应用研究，2003，2(4)：181－182.

[9] Li hui, Chen shihong. Recursive Decomposition of Software Process [J]. Wuhan University Journal of Natural Sciences, 2009, 2(14): 143－147.

（下转第 100 页）

因此,用上述方法计算 AISI4340 钢制超高压容器最佳自增强时的弹-塑性交界面半径是合理的.

4 结 语

超高压容器用钢 AISI4340 的包辛格系数是一个常数,其大小与塑性层的位置和容器自增强压力的大小无关. AISI4340 钢包辛格系数的平均值为 0.967 1. 工程实例验证表明,用文中方法计算 AISI4340 钢制超高压容器最佳自增强时的弹-塑性交界面半径是合理的.

参考文献:

[1] 郎国华,魏龙灿. 超高压容器[M]. 北京:化学工业出版社,2002.

[2] [日]HPIS-C-103-1980. 超高压圆筒容器设计指针[S].

[3] 蔡洪涛. 压力容器法兰的参数化绘制方法[J]. 武汉工程大学学报,2010,32(9):86-89.

[4] 华南工学院高压容器研究室. 自增强容器最佳超应变的实验研究[J]. 压力容器,1984,1(2):17-24.

[5] 刘小宁. 超高压管式反应器的最佳弹性-塑性交界面半径[J]. 化工设计,2006,16(5):29-30,42.

[6] 陈进锋,陈国理. 恢复超高压聚乙烯反应管自增强残余应力[J]. 石油化工设备,1987,16(9):23-27.

Bauschinger coefficient of AISI4340 steel for super-high pressure vessel

LIU Bing[1], *YUAN Xaio-hui*[1], *LIU Cen*[2], *WU Yuan-xiang*[1], *ZHANG Hong-wei*[1], *LIU Xiao-ning*[1]

(1. Department of Mechanical Manufacture Engineering, Wuhan Polytechnic College of Software and Engineering, Wuhan 430205, China;
2. School of Mechanical & Electrical Engineering, Wuhan Institute of Technology, Wuhan 430074, China)

Abstract: The method of determining the steel bauschinger coefficient was proposed according to the plastic level pre-stressed actual value of single-layer super-high pressure autofrettage vessel. The bauschinger coefficient of AISI4340 steel for the super-high pressure vessel was analyzed based on the test data. The research results show that: (1) The AISI4340 bauschinger coefficient is a constant, and the value has nothing to do with the plastic level position and the autofrettage pressure size. (2) The bauschinger coefficient mean value of AISI4340 is equal to 0.967 1. (3) The rationality of the method is proved using two project examples.

Key words: super-high pressure vessel; AISI4340 steel; bauschinger coefficient

本文编辑:陈小平

☆

(上接第 93 页)

Model of software process decomposition method

ZHUANG Yun[1], *LI Hui*[2]

(1. School of Environmental and Civil Engineering, Wuhan Institute of Technology, Wuhan 430074, China;
2. School of Computer, Wuhan University, Wuhan 430072, China)

Abstract: In this article, we built on software process module M rule of decomposition method, this method as a decomposition method M model, called a matrix, expressed as $A = [w, t]$. This mode is a recursive procedure to M: theoretical analysis and actual binding in one, as it's recursive expansion process to form a valid conditions, generate optimized computing M prototype Pri-cTr.

Key words: decomposition model; computing prototype; complexity; set of computing prototype

本文编辑:陈小平