一种从 Z 规约到并行程序的精化方法*

万剑怡[1,2], 孙永强[1], 薛锦云[2,3]

[1](上海交通大学 计算机科学与工程系,上海 200030);
[2](江西师范大学 计算机科学系,江西 南昌 330027);
[3](中国科学院 软件研究所 计算机科学重点实验室,北京 100080)
E-mail: jywan820@sina.com
http://cs.sjtu.edu.cn

摘要: 提出了一种通过对设计模式进行精化,从 Z 规约开发并行程序的方法.该方法对 Z 语言进行了并行扩充,从 Z 功能规约出发,通过使用扩展的设计模式逐步精化得到并行的设计规约,再通过保持语义的转换,得到可最后转换为并行代码的抽象并行程序.通过实例对这一方法进行了详细的描述.

关 键 词: 设计模式;精化;并行程序开发
中图法分类号: TP311 文献标识码: A

并行程序设计是并行计算的两大难题之一,也是并行计算领域的一个研究热点.形式化并行程序开发方法是保证并行程序正确性的有效途径.但其中存在一个难点,就是从功能规约到设计规约之间需要创造性的思维,难以通过形式化的推导来实现.

设计模式是对不断重复发生的问题及其解决方案的核心的描述[1],它在面向对象软件设计中有着广泛的应用.但这一概念并不局限于该领域,我们在文献[2]中对其进行了扩展,定义了可用于问题求解的设计模式.

将设计模式引入并行程序的形式化开发,建立一个以 Z 语言为基础的并行程序精化框架.在这一框架下,使用 Z 语言描述问题的功能规约,采用设计模式和精化演算相结合的方法进行程序精化,可以弥补现有形式化方法对设计阶段支持的不足,并通过验证保证设计的正确性.

1 基本概念

1.1 PaZ

为了在 Z 语言框架下支持并行程序的精化开发,我们对 Z 语言进行了并行扩充,形成一个 Z 扩充版本 PaZ.PaZ 中定义了运算、操作模式的可并行组合关系、并行组合算子、并行函数作用等.例如,并行函数作用定义如下:

定义 **1**. 若 $f:X\rightarrow Y, A:FX$,并行函数作用 f_A 定义为

$$f_A:FY \quad f_A \Leftrightarrow \forall x \in A \bullet ||fx,$$

其中||为并行组合算子.

---

* 收稿日期: 2001-03-16; 修改日期: 2001-06-25
基金项目: 国家自然科学基金资助项目(69983003)

作者简介: 万剑怡(1974－),女,江西进贤人,博士生,讲师,主要研究领域为并行计算,软件工程环境;孙永强(1931－),男,浙江嘉善人,教授,博士生导师,主要研究领域为并行计算理论;薛锦云(1947－),男,江苏海门人,教授,博士生导师,主要研究领域为软件形式化和自动化,并行分布式计算.

1.2 扩展的设计模式及其表示

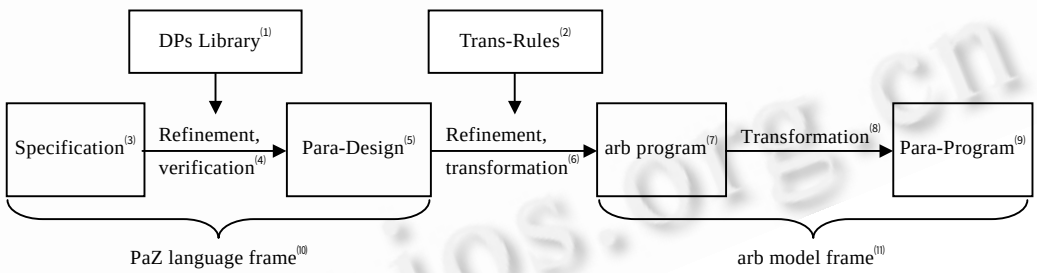
从功能规约到设计规约的过程实际上就是问题求解的过程.人们在长期的设计实践中总结出了一些问题求解的策略,并得到了广泛的应用,如分而治之法(divide and conquer)、贪心法(greedy)、分枝限界法(branch and bound)等.然而这些策略的使用往往是非形式的,取决于设计人员的知识与经验.我们认为这一类设计经验也可定义为设计模式[2].传统的设计模式表示采用自然语言叙述、程序设计语言实现的 DPs 具体样本、OMT 图相结合的方法.这些方法抽象级别低,缺乏严格性和完整性,不能支持其在形式化开发中的应用.我们采用 PaZ 表示设计模式,使其可以自然地融入 Z 语言的精化演算体系,从而支持设计阶段的形式化开发.

1.3 在SRVRT模型下进行傅里叶变换的并行程序开发

arb 模型是一个描述并行程序的操作语义模型[3],它将程序描述成一个状态转换系统,定义了一个并行程序的基本框架,并提供了一系列转换规则,将 arb 并行程序转换为共享内存模型或消息传递模型下的具体并行程序.因此,只要能从 Z 规约出发,逐步精化得到 arb 并行程序,再对其实施一定的转换就可得到一个具体的并行程序实现.我们开发了从 PaZ 规约到 arb 程序的一组精化、转换规则,通过使用这些规则对 PaZ 设计规约实施保持语义的转换,可以得到等价的 arb 程序表示.这阶段的精化是程序级的,因此可以形式化地实现,从而保证所得的 arb 程序的正确性.

2 并行程序开发模型

我们的并行程序开发过程定义为从一个 Z 语言描述的问题规约开始,使用模型中预先定义的设计模式进行逐步精化,得到并行设计,再通过保持语义的转换得到 arb 并行程序的过程.精化包括数据精化、运算精化和并行性开发.数据精化是指为一个抽象的状态空间寻找具体的数据结构表示;运算精化是指抽象运算的具体化;并行性开发包括数据的并行分解和运算的并行分解以及并行分解所带来的数据复制、通信操作等,通过使用设计模式来实现.我们将支持这一并行程序开发方法的模型称为 SRVRT(由每个开发步骤的首字母组成)模型,如图 1 所示.



(1)设计模式库,(2)转换规则库,(3)功能规范,(4)精化、验证,(5)并行设计,(6)精化、转换,(7)arb 程序,(8)转换,(9)并行程序,(10)PaZ 语言框架,(11)arb 模型框架.

Fig.1 SRVRT parallel program develop model

图 1 SRVRT 并行程序开发模型

具体而言,在该模型下开发并行程序分成以下步骤进行:

第 1 步. Specification.使用 Z 语言对应用问题进行形式化的描述,得到初始的功能规约.

第 2 步. Refinement using DPs.使用 PaZ 的精化规则,从问题规约出发,将一定的设计模式进行逐步精化,得到对问题进行并行求解的并行设计.

第 3 步. Verification.证明所得到的设计满足问题的规约.

第 4 步. Refinement using laws.使用 PaZ 到 arb 程序的转换规则将并行设计精化、转换为 arb 程序.

第 5 步. Transformation.使用文献[3]中的规则对 arb 程序实施转换,得到具体的并行程序.

3 并行程序开发实例

我们以基数为 2 的傅里叶变换问题的并行程序开发为例,说明 SRVRT 模型下进行并行程序开发的过程.

3.1 并行分而治之设计模式(Parallel Divide & Conquer)

分而治之是一种常用的问题求解策略.它将原始问题分解成性质相同、规模更小的子问题进行独立求解,再将子问题的解合并成原问题的解.并行分而治之就是将各子问题的求解并行进行.将这一问题求解策略构造成设计模式 Parallel Divide & Conquer,可用于一类问题的并行求解.在 SRVRT 模型下使用 Parallel Divide & Conquer,我们开发了矩阵相乘、傅里叶变换、排序等问题的并行程序.

以 PaZ 语言描述的分而治之设计模式 Parallel Divide & Conquer 如下:

[DATA,RESULT]
datas: DATA
result: RESULT
divide: DATA→*F* DATA
merge: *F* REAULT→RESULT
baseoperate: DATA→RESULT
basedatas: DATA→BOOL
D&C: DATA→RESULT

D&C datas = if basedatas datas then baseoperate datas
Else merge $D\&C_{\text{divide datas}}$
Result=D&C data

3.2 在SRVRT模型下进行傅里叶变换的并行程序开发

下面我们以基数为 2 的傅里叶变换问题的并行程序开发为例,详细描述 SRVRT 模型下开发并行程序的各个步骤.

步骤 1. Specification.以 Z 语言描述离散傅里叶变换问题的功能规约如下:

DFT
a: seq complex
b: seq complex
FFT: seq complex→seq complex
n,*k*: N
w: complex

$n=\#a=\#b\wedge$
$n=2^k \wedge w^n=1\wedge\forall 0<i<n\bullet w^i\neq 1$
$\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n}a(i)w^{(i-1)(j-1)}$
b=FFT *a*

步骤 2. Refinement using DPs.使用设计模式 Parallel Divide & Conquer 对 DFT 进行并行设计,精化过程包括数据精化和运算精化.即要使用 DFT 中的数据和类型表示 Parallel Divide & Conquer 中的抽象变量和类型,并使用这些数据将 Parallel Divide & Conquer 中的运算进行具体化.

显然,DFT 规约中的数据 *a* 和 *b* 及其类型应作为 Parallel Divide & Comquer 中基本数据 *datas* 和 *result* 及其类型的精化,即数据精化包括:

$$DATA\subseteq seq\ complex,\quad RESULT\subseteq seq\ complex,\quad Datas\subseteq a,\quad Result\subseteq b.$$

下面需要对 Parallel Divide & Conquer 中的各个函数进行具体化,假设 *basedatas*,*baseoperate*,*merge*,*divide*,*D&C* 分别精化为 *FFTbasedatas*,*FFTbaseoperate*,*FFTmerge*,*FFTdivide*,*FFTD&C*,则精化所得的结果需要满足 FDT 问题的规约,即

$$n=\#a=\#b\wedge n=2^k\wedge w^n=1\wedge\forall 0<i<n\bullet w^i\neq 1\wedge b=FFTD\&C\ a\Rightarrow\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n}a(i)w^{(i-1)(j-1)}.$$

在 DFT 中,当问题的规模等于 1 时,该问题可直接求解,因此 *FFTbasedatas*, *FFTbaseoperate* 定义为

FFTbasedatas a = (*n*=1), *FFTbaseoperate a* = *a*.

当 *n*>1 时,按照 Parallel Divide & Conquer 的思想,需将 *a* 进行分解.若将 *a* 按下标的奇偶分解成两个部分,则 $\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n}a(i)w^{(i-1)(j-1)}$ 可变换为 $\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n/2}a\ (2i-1)w^{2(i-1)(j-1)}+w^{j-1}\sum_{i=1}^{n/2}a\ (2i)w^{2(i-1)(j-1)}$,且根据 *w* 的性质,若将 *b* 分成前后两个长度相等的序列,则有

$$\forall 1\le j\le n/2\bullet b(j)=\sum_{i=1}^{n/2}a(2i-1)w^{2(i-1)(j-1)}+w^{j-1}\sum_{i=1}^{n/2}a(2i)w^{2(i-1)(j-1)},$$

$$\forall 1\le j\le n/2\bullet b(j+n/2)=\sum_{i=1}^{n/2}a(2i-1)w^{2(i-1)(j-1)}-w^{j-1}\sum_{i=1}^{n/2}a(2i)w^{2(i-1)(j-1)}.$$

因此,可以分别定义 FFTdivide 和 FFTmerge 如下:

a: seq complex subdatas: *F* seq complex a_{even},a_{odd}: seq complex FFTdivide: seq complex→ *F* seq complex	Subresults: *F* seq complex b_{even},b_{odd}: seq complex *b*: seq complex FFTmerge: *F* seq complex→seq complex
$n/2=\#a_{odd}=\#a_{even}$ Subdatas=FFTdivide *a* Subdatas={a_{even},a_{odd}} $\forall 1\le i\le n/2\bullet a_{even}(i)=a(2i)\wedge a_{odd}(i)=a(2i-1)$	$n/2=\#b_{odd}=\#b_{even}$ *b*=FFTmerge subresults Subresults={b_{even},b_{odd}} $\forall 1\le i\le n/2\bullet b(i)=b_{odd}(i)+w^{i-1}b_{even}(i)\wedge b(i+n/2)=b_{odd}(i)-w^{i-1}b_{even}(i)$

FFTD&C 的定义可以从 Parallel Divide & Conquer 的性质直接得到:

FFTD&C a = if *FFTbasedatas a then FFTbaseoperate a*

Else *FFTmerge FFTD & C*$_{FFTdivide\ a}$

步骤 3. Verification.为了保证上述设计的正确性,需要证明设计规约中的性质满足问题 DFT 的后置条件,也即

$$n=\#a=\#b\wedge n=2^k\wedge w^n=1\wedge\forall 0<i<n\bullet w^i\ne 1\wedge b=FFTD\&C\ a\Rightarrow\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n}a(i)w^{(i-1)(j-1)}. \quad (1)$$

证明分成两个部分:

(1) 当 if 条件成立时,即 *n*=1 时,由于 *n*=1,所以 *w*=1,则 *FFTbaseoperate a*=*a*(1)满足 *b*(1)=*a*(1),则式(1)成立;

(2) If 条件不成立时,假设当 $n=2^k$ 时式(1)成立,为了有所区别,此时的 *w* 使用 w_k 表示,即

$$\forall 1\le j\le n\bullet b(j)=\sum_{i=1}^{n}a(i)w_k^{(i-1)(j-1)}. \quad (2)$$

证明 $n=2^{k+1}$ 时,式(1)成立(具体证明过程略). □

步骤 4. Refinement using laws.使用 PaZ 到 arb 程序的精化规则将上述设计进一步精化为 arb 程序.精化规则包括对 PaZ 中的各种数据结构、模式、运算、表达式、程序结构等向 arb 程序相应结构的转换.例如含有并行组合运算的全称量词表达式的精化规则为

$\forall 1\le i\le n\bullet \parallel op_i\subseteq arball(i=1:N)$

call op_i

end *arb*

将步骤 2 得到的并行设计进行转换后所得到的快速傅里叶变换的 arb 程序如下:

```
integer::N
complex::a(N),b(N),aeven(N/2),aodd(N/2),beven(N/2),bodd(N/2)
complex::w
subroutine fftbaseoperate(a)
  b(1)=a(1)
```

```
end subroutine fftbaseoperate
subroutine fftdivide(a)
   arball (i=1:N/2)
      aeven(i)=a(2i)
      aodd(i)=a(2i−1)
   end arb
end subroutine fftdivide
subroutine fftmerge
   arball (i=1:N/2)
      b(i)=bodd(i)+w^(i−1)beven(i)
      b(i+N/2)=bodd(i) −w^(i−1)beven(i)
   end arb
end subroutine fftmerge
recursive subroutine fft(a)
   if fftbasedatas(a) then    call fftbaseoperate(a)
   else    call fftdivide(a)
           arb
              call fft(aeven)
              call fft(aodd)
           end arb
           call fftmerge(beven,bodd)
   end if
end subroutine fft
```

容易看出,该程序与 PRAM 下的并行快速傅里叶变换算法[4]是等价的.

步骤 5. Transform.使用 arb 模型下的转换规则,将上述程序转换为具体并行程序(具体步骤略,参见文献[3]).

4 相关工作比较与总结

本文提出了一种通过对设计模式进行精化从 Z 规约开发并行程序的方法.该方法的主要贡献在于它首次将设计模式的思想引入并行程序的形式化开发,并将其统一在一个完整的框架之下,既弥补了形式化方法在设计能力上的不足,又保证了并行程序设计的正确性.

并行程序的形式化开发领域集中了大量的研究工作.近年来有基于不同理论的各种形式化框架和模型,如文献[5~7]等提出来.这些形式化开发方法从问题的规约出发,对初始规约使用模型中的精化规则进行精化演算,使其逐步转换成便于实现的形式,直至代码.在 SRVRT 模型下开发并行程序,并不直接对问题规约进行精化演算,而是以问题规约的满足为设计的目标,对求解该问题可用的设计模式进行精化,使所得的设计满足问题的规约.通过引入设计模式,在功能规约和设计规约之间架起了一道桥梁,因此,SRVRT 模型比其他并行程序形式化开发模型具有更强的实用性.

另一方面,国际上也有一些基于设计模式思想的并行程序开发方法的研究,如基于算法骨架的方法和目前进行的 PQE2000 Project[8]、基于并行程序设计典型的方法和系统[9]、基于并行结构骨架的方法及其面向对象的实现[10]等.这些方法所采用的手段是通过将并行计算模式扩充到顺序语言环境中,以此来设计实现并行程序的开发环境(包括程序设计模型、语言、工具及集成环境).它们对设计模式思想的使用主要在程序设计(programming)的层面.而我们对设计模式的使用主要在设计(design)层面,采用形式化手段对设计模式进行精化而得到并行设计,并通过严格的验证保证设计的正确性,进而保证并行程序的正确性.

目前,从现有实例的开发来看,使用 SRVRT 模型开发并行程序是一种实用、可行的方法.在 SRVRT 模型中我们定义了一些常用的设计模式,开发并验证了一些经典问题的并行程序.我们进一步的工作是对设计模式库的开发完善以及从 PaZ 设计规约到 arb 程序精化转换的自动实现.

References:

[1] Gamma, E., Helm, R., Johnson, R., *et al*. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, 1995.

[2] Wan, Jian-yi, Sun, Yong-qiang, Xue, Jin-yun. Expanding design patterns to parallel programming. In: Chen, Jian, Chen, Ping, Meyor, B., eds. Proceedings of the 38th International Conference on Technology of Object Oriented Languages and Systems, TOOLS 36. Los Alamitos, CA: IEEE Computer Society, 2000.

[3] Massingill, Berna. A Structured Approach to Parallel Programming [Ph.D. Thesis]. California Institute of Technology, 1998.

[4] Berman, K.A., Berman, K., Paul, J. Fundamentals of Sequential and Parallel Algorithms. Boston, MA: PWS Publishing Company, 1997.

[5] Dingel, J. A trace-based refinement calculus for shared-variable parallel programs. In: Haeberer, A.M., ed. Algobraic Methodology and Software Technology, Proceedings of the 7th International Conference, AMAST'98. LNCS 1548, Berlin: Spring-Verlag, 1999.

[6] Bodeveix, J.P., Filali, M. A framework for parallel program refinement. In: Brinksma, E., Cleaveland, W.R., Larsen, K.G., *et al*., eds. Proceedings of the Workshop on Tools and Algorithms for the Construction and Analysis of Systems. Berlin: Spring-Verlag, 1995.

[7] Chin, Wei-Ngan, Khoo, Siau-Cheng, Hu, Zhen-jiang, *et al*. Deriving parallel codes via invariants. In: Static Analysis, Proceedings of the 7th International Symposium, SAS 2000. LNCS 1824, Spring-Verlag, 2000.

[8] Bacci, B., Danelutto, M., *et al*. SkIE: a heterogeneous environment for HPC applications. Parallel Computing, 1999,25:1827~1852.

[9] Massingill, B.L., Chandy, K.M. Parallel program archetypes. In: IPPS/SPDP, Proceedings of the 13th International Parallel Processing Symposium, 10th Symposium on Parallel and Distributed Processing. Los Alamitos, CA: IEEE Computer Society, 1999. http://resolver.library.caltech.edu/caltechCSTR:1997.cs-tr-96-28.

[10] Goswami, D., Singh, A., Preiss, B.R. Building parallel applications using design patterns. In: Advances in Software Engineering: Topics in Comprehension, Evolution and Evaluation. New York: Springer-Verlag, 2000.

Refinement from Z Specification to Parallel Program*

WAN Jian-yi[1,2], SUN Yong-qiang[1], XUE Jin-yun[2,3]

[1](*Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai* 200030, *China*);

[2](*Department of Computer Science, Jiangxi Normal University, Nanchang* 330027, *China*);

[3](*Key Laboratory for Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing* 100080, *China*)

E-mail: jywan820@sina.com

http://cs.sjtu.edu.cn

Abstract: In this paper, a method of getting parallel program from Z specification through refinement of design patterns is proposed. It expands parallel concepts into Z notation. Beginning with Z functional specification, it refines expanded design patterns step by step to get parallel design, and succeed in obtaining abstract parallel program through semantic-preserving transformations, which can be transformed to parallel code finally. This method is described in detail through an example.

Key words: design patterns; refinement; parallel program development

* Received March 16, 2001; accepted June 25, 2001
Supported by the National Natural Science Foundation of China under Grant No.69983003