软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(4):715−729 [doi: 10.3724/SP.J.1001.2013.04260]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

Radl 形式规格说明相对正确性研究*

王昌晶[1,2,3,4], 薛锦云[1,2]

[1](计算机科学国家重点实验室(中国科学院 软件研究所),北京 100190)

[2](江西师范大学 省高性能计算技术重点实验室,江西 南昌 330022)

[3](中国科学院 研究生院,北京 100190)

[4](江西师范大学 计算机信息工程学院,江西 南昌 330022)

通讯作者: 王昌晶, E-mail: wcj771006@163.com

摘 要: 在形式规格说明的获取任务中,一个重要问题是验证获取得到的形式规格说明的正确性.即给定一个问题需求 P,往往可以获取多种不同形式的规格说明,如何验证这些不同形式的规格说明均正确?问题需求的非(半)形式化与形式规格说明的形式化两者之间差异的本性,使得该问题成为软件需求工程中一个具有挑战性的问题.提出一种基于形式化推导的方法来验证同一问题不同形式规格说明的相对正确性,通过证明不同形式规格说明与问题需求某个最为直截明了的形式规格说明 S_i 等价来实现,而 S_i 使用 PAR 方法和 PAR 平台转换为可执行程序,通过测试已经得到确认.为了支持该方法,进一步提出了扩展的逻辑系统和辅助证明算法.使用 Radl 语言作为形式规格说明语言,通过排序搜索、组合优化领域的两个典型实例对该方法进行了详细的阐述.实际使用效果表明,该方法不仅能够有效地验证 Radl 形式规格说明的正确性,还具备良好的可扩充性.该方法在规格说明的正确性验证、算法优化、程序等价性证明等研究领域具有潜在的理论意义与应用价值.

关键词: 形式规格说明;相对正确性;确认;扩展的逻辑系统;辅助证明算法

中图法分类号: TP311 **文献标识码**: A

中文引用格式: 王昌晶,薛锦云.Radl 形式规格说明相对正确性研究.软件学报,2013,24(4):715−729. http://www.jos.org.cn/1000-9825/4260.htm

英文引用格式: Wang CJ, Xue JY. Research on relative correctness of Radl formal specification. Ruanjian Xuebao/Journal of Software, 2013,24(4):715−729 (in Chinese). http://www.jos.org.cn/1000-9825/4260.htm

Research on Relative Correctness of Radl Formal Specification

WANG Chang-Jing[1,2,3,4], XUE Jin-Yun[1,2]

[1](National Key Laboratory for Computer Science (Institute of Software, The Chinese Academy of Sciences), Beijing 100190, China)

[2](Key Laboratory for High-Performance Computing Technology, Jiangxi Normal University, Nanchang 330022, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100190, China)

[4](College of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China)

Corresponding author: WANG Chang-Jing, E-mail: wcj771006@163.com

Abstract: During the task of acquiring formal specification, an important problem is verifing the correctness of acquired formal specification. In other words, given a problem requirement P, a variety of formal specifications will be acquired, but how to verify the correctness of them all? The different nature of the non- (semi-) formal of problem requirement and formal of specification makes it a challenging software problem and requirement for engineering. This paper proposes a formal derivation method to verify the relative correctness of different forms of Radl specifications corresponding same problem. It achieves this through a proof of the equivalency

* 基金项目: 国家自然科学基金重大国际(地区)合作与交流项目(61020106009); 国家自然科学基金(61272075); 江西省自然科学青年科学基金(201222BAB211030)

收稿时间: 2011-06-17; 修改时间: 2011-11-02; 定稿时间: 2012-02-15

among different forms of Radl specifications and a certain formal specification S_i, which is straightforward to the problem requirement. S_i is converted into an execute program using PAR method and PAR platform, and is validated by test. In order to support the method, the study further put forth an extended logic system and aided certified algorithm. This paper uses Radl as formal specification language and elaborates the method using two typical examples in the domains of sort and search, combinational optimization. Practical effects manifest not only can effectively verify relatively correctness of Radl specification, but also has well extendibility. The method has potential theory significance and application value in research areas of formal specifications correctness verification, algorithms optimization and programs equivalency proof.

Key words: formal specification; relative correctness; validation; extended logic system; aided certified algorithm

近年来,形式规格说明方法如,Z,B,Raise 等方法已经应用到工业软件开发项目中,形式化方法的实用性与潜在的效益日益明显[1,2].但是,由于要求使用者具备良好的数学基础和抽象思维能力,形式规格说明的获取仍然是一项相当困难的任务,研究进展缓慢[3].在形式规格说明的获取任务中,一个重要问题是验证(verify)获取得到的形式规格说明的正确性.即给定一个问题需求 P,往往可以获取多种不同形式的规格说明 S_i,$1\leqslant i\leqslant n$,如何验证这些不同形式的规格说明均正确?

由于问题需求 P 一般是非形式化(如自然语言)或半形式化的(如图、表),难以严格形式化,而形式规格说明 S_i 是形式化的.因此,欲证明 $S_i|=P$,即验证 S_i 的正确性(其中,$1\leqslant i\leqslant n$),则通过完全形式化的方式来验证几乎是不可能的.P 的非(半)形式化与 S_i 的形式化两者之间差异的本性,使得验证规格说明的正确性成为软件需求工程中一个具有挑战性的问题,目前仍然没有根本的解决方法[4,5].

在深入研究的过程中可以发现,没有绝对的正确性,只存在满足一定前提条件下的相对正确性.本文提出形式规格说明相对正确性的概念,规格说明相对正确性是指相对一定前提条件下的正确性,即假定问题某个最为直截明了的形式规格说明 S_i 已得到确认(validation).这样,若要验证同一问题不同形式规格说明的相对正确性,可以通过证明不同形式规格说明与问题需求某个最为直截明了的形式规格说明 S_i 等价来达到,而后者使用 PAR 方法和 PAR 平台转换为可执行程序,通过测试已经得到确认.

本文提出一种基于形式化推导的方法来验证同一问题不同形式规格说明的相对正确性.为了支持该方法,进一步提出了扩展的逻辑系统和辅助证明算法.

本文通过排序搜索、组合优化领域的两个典型实例对该方法进行了详细的阐述.实际使用效果表明,该方法不仅能够有效地验证 Radl 形式规格说明的正确性,还具备良好的可扩充性(well extensibility).该方法在规格说明的正确性验证、算法优化、程序等价性证明等研究领域具有潜在的理论意义与应用价值.

本文第 1 节给出需要的预备知识,包含形式规格说明语言 Radl 简介及扩展的逻辑系统.第 2 节提出基于形式化推导的同一问题不同形式 Radl 规格说明相对正确性验证方法,包括形式化推导方法、测试确认与辅助证明算法.第 3 节通过排序搜索、组合优化领域的两个典型实例对这一方法进行详细的阐述.第 4 节对比国内外研究现状.最后是总结与展望.

1 预备知识

1.1 形式规格说明语言Radl

PAR 方法和 PAR 平台是本实验室软件形式化和自动化学术团队提出并研制成功的一个程序设计环境[6−10],它目前已能支持从软件形式规格说明 Radl 到可执行语言程序的软件(半)自动开发.为了使本文研究工作与 PAR 方法和 PAR 平台无缝衔接,本文采用 Radl 语言作为形式规格说明语言.Radl 语言不仅适应传统的数学习惯,且具有引用透明性,这十分便于形式规格说明的推导;经转换为可执行语言程序后,还可以支持直接执行,这又十分便于形式规格说明的测试确认.

Radl 语言的主要功能是描述问题的规格说明、规格说明变换规则和描述算法,由算法规格说明语言和算法描述语言两部分组成.Radl 提供了足够抽象的机制,可集中刻画问题的功能,而不为设计和实现所涉及的问题(如效率)所干扰.

我们采用如下 Radl 算法规格说明的刻画形式:

|[标识符说明]|

AQ:谓词表达式;

AR:谓词表达式.

其中,标识符说明部分主要用于说明前、后置断言中出现的变量和函数的属性及类型.属性有 3 种:一是输入变量,用关键字 IN 标识;二是输出变量,用关键字 OUT 标识;三是辅助变量,用关键字 AUX 标识.类型可以是 Radl 语言中的标准数据类型(integer,real,boolean,char,string)、自定义简单类型(记录类型、数组类型、枚举类型和子界类型)、预定义 ADT 类型(集合类型、序列类型、树类型、图类型)和自定义 ADT 类型.

以 AQ 和 AR 开头的谓词表达式分别称为算法的前置断言和后置断言,用于表示算法的输入参数必须满足的条件和算法的输出参数必须满足的条件,均为一阶谓词逻辑公式.使用统一的格式$(Qi{:}r(i){:}f(i))$表示"在范围$r(i)$上,对函数$f(i)$施行 q 运算所得到的量",其中,Q 是 q 运算的一般化,可以是∀(全称量词),∃(存在量词),MIN(求最小值量词),MAX(求最大值量词),∑(求和量词),∏(求积量词)等,所分别对应的 q 运算是∧,∨,min,max,+,*等[11].

下例是一个计算图单源最短路径问题的形式化 Radl 规格说明:

Algorithm: single-source min$dis(s,t)$

IN: s,t: vertex; V: set(vertex,n); C: array([1:n,1:n],sometype);

OUT: mindis: array[1:n−1] of sometype.

AQ: $s\in V$

AR: $\forall(t{:}t\in V{:}\min dis(s,t)=(\mathrm{MIN}p{:}p\in PATH(s,t){:}length(p))$

$length(p)=(\sum\langle i,j\rangle{:}\langle i,j\rangle\in p{:}C\langle i,j\rangle)$

1.2 扩展的逻辑系统

可以对一阶谓词逻辑进行扩展,扩展的逻辑系统包括与 Radl 这一小型语言相关的前提条件、定理和推导规则(逻辑定律与论域知识).为了方便后续的证明,我们可以将前提条件、定理和推导规则都放到一个统一的定律库中.具体说明如下:

前提条件:即问题需求 P 最为直截明了的 Radl 规格说明 S_i,它已经通过测试得到确认;

定理:证明过程中由 S_i 出发通过形式化推导得到的其他规格说明 S_j 均作为定理;

前提条件与定理在定律库中的编号从 $T0$ 开始.

推导规则:包括逻辑定律与论域知识.逻辑定律主要是指一阶谓词逻辑定律与Radl规格说明变换规则,扩展的逻辑系统是通过在一阶谓词逻辑中加入 Radl 规格说明变换规则来进行扩展的,其主要特点是扩充的量词,其形式为$(Qi{:}r(i){:}f(i))$,参考第 1.1 节.从理论上来讲,所有的 Radl 规格说明变换规则都可由一阶谓词逻辑定律得到证明,但在实践中,若对每一条变换规则每次都要从一阶谓词逻辑出发来加以证明,则过于繁琐.因此,需要一个含有Radl规格说明变换规则的库,库中的变换规则已经得到了证明,可以放心使用.论域知识一般包括特定领域的性质、引入新的定义、定义展开与折叠等.

为了区分逻辑定律与论域知识,逻辑定律在定律库中的编号从 $L0$~$L100$,论域知识编号从 $L101$ 开始.

下面列出本文要使用到的逻辑定律 $L1$~$L7$,其中,$L1$~$L5$ 是 Radl 规格说明变换规则,$L6$,$L7$ 是一阶谓词逻辑定律:

$L1$. 交叉积性质

$$(Qi,j{:}r(i)\wedge s(i,j){:}f(i,j))\equiv(Qi{:}r(i){:}(Qj{:}s(i,j){:}f(i,j))).$$

$L2$. 单点范围

$$(Qi{:}i=k{:}f(i))\equiv f(k).$$

$L3$. 范围分裂

$$(Qi{:}r(i){:}f(i))\equiv(Qi{:}r(i)\wedge b(i){:}f(i))q(Qi{:}r(i)\wedge\neg b(i){:}f(i)).$$

$L4$. 一般分配律

若二元运算符⊙满足交换律,且对 q 满足分配律,i 不是 g 的自由变量,则有

$$(Qi{:}r(i){:}g\wedge\odot f(i))\equiv g\odot(Qi{:}r(i){:}f(i)).$$

*L*5. ∀规则

$$(\forall i{:}r(i)\wedge s(i){:}p(i))\equiv(\forall i{:}r(i){:}s(i)\Rightarrow p(i)).$$

*L*6. $r(i)$=F⇒$(\forall i{:}r(i){:}f(i))$≡T.

*L*7. $P\vee$F≡P,$P\wedge$T≡P.

2 Radl 规格说明相对正确性验证方法

2.1 形式化推导方法

给定一个问题需求 P,欲验证多种不同形式 Radl 规格说明 $S_1,S_2,\ldots,S_n$ 的相对正确性,基于形式化推导的验证方法步骤如下:

步骤 1. 选择问题需求 P 最为直截明了的 Radl 规格说明 S_i,使用 PAR 方法与 PAR 平台,将 S_i 转换为可执行程序,通过测试来确认 S_i 满足问题 P 的需求.

步骤 2. 通过对规格说明进行形式化推导来证明其余 Radl 规格说明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 与 S_i 均等价.于是,$S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 相对正确.

步骤 2 需要分两种情形进行讨论:

情形 1. 证明过程中使用到的推导规则都是逻辑重言式,那么需要证明存在一条证明链:$S_i\equiv S_{i+1},S_{i+1}\equiv S_{i+2},\ldots,S_{n-1}\equiv S_n,S_n\equiv S_1,\ldots,S_{i-2}\equiv S_{i-1}$.这可以通过将 S_i 看作前提条件,并利用逻辑定律与论域知识作为推导规则构成一个逻辑系统,证明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 均为该逻辑系统中的定理来进行.

情形 2. 证明过程中使用到的推导规则包含逻辑蕴含式,那么需要证明存在一条证明回路:$S_i\Rightarrow S_{i+1},S_{i+1}\Rightarrow S_{i+2},\ldots,S_{n-1}\Rightarrow S_n,S_n\Rightarrow S_1,\ldots,S_{i-2}\Rightarrow S_{i-1},S_{i-1}\Rightarrow S_i$.这可以分两步来实现:首先证明 $S_i\Rightarrow S_{i+1},S_{i+1}\Rightarrow S_{i+2},\ldots,S_{n-1}\Rightarrow S_n,S_n\Rightarrow S_1,\ldots,S_{i-2}\Rightarrow S_{i-1}$,这与情形 1 类似;再证明 $S_{i-1}\Rightarrow S_i$,这也可以看成情形 1 的特例.

综上所述,无论何种情形,都需要构建一个逻辑系统,包含前提条件、定理和推导规则(逻辑定律与论域知识),第 1.2 节已对该逻辑系统进行了详细阐述.

2.2 测试确认

步骤 1 要选择问题需求 P 最为直截明了的 Radl 规格说明 S_i 对其进行确认,这是通过使用 PAR 方法与 PAR 平台将 S_i 转换为可执行程序,通过测试来确认 S_i 满足问题 P 的需求.具体步骤如下:

步骤 1.1. 选定问题需求 P 最为直截明了的 Radl 规格说明 S_i.

步骤 1.2. 使用 PAR 方法与 PAR 平台,将 S_i 经 Apla 抽象程序转换生成为可执行语言程序,如 C#,具体参考第 1.1 节.

步骤 1.3. 通过测试来确认 Radl 形式规格说明 S_i 满足问题需求 P.

以上最重要的是步骤 1.2,其中使用到了 PAR 方法与 PAR 平台的变换规则与自动转换工具,可以通过下面两种方式保证它们的高可信性:一是使用定理证明器 Isabelle 进行严格的证明[12],二是使用等价类划分与路径分析相结合的方法进行严格的测试[13].这样,步骤 1.3 通过测试可以确认 S_i 满足问题 P 的需求.

2.3 辅助证明算法

欲证不同形式 Radl 规格说明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 与问题需求 P 某个最为直截明了的形式规格说明 S_i 等价,既可以使用手工推导,也可以使用一种辅助证明算法来自动推导.手工推导可能导致推导路径变短,但是需要大量创造性工作;自动推导可以机械自动执行并可出示证据,但是推导路径可能变长.

为设计辅助证明算法,需要一个定律库 DB 支持,包含前提条件、定理和推导规则(逻辑定律与论域知识).该定律库中主要包含两种形式的定律:逻辑蕴含式“⇒”与逻辑重言式“≡”,前者可进行推导,后者可进行重写.证明过程的核心是选择可用的定律“替换”或“重写”Radl 形式规格说明.为了提高证明效率,算法采用基于目标的

推理方式.因此,逻辑定律的表示方法是:如果定律 L 有前提部分,则前提部分放在“⇐”的右边,表示为 $L.right$.下面给出该辅助证明算法:

int Function *Certify*(S_i,*spec_list*-[S_i])

/*证明 *spec_list* 中所有子目标均成立

AQ:给定一个定律库 DB,包含前提条件、定理和推导规则.输入前提条件 S_i,欲证的其他不同形式 Radl 规格说明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$,使用序列 *spec_list* 来存储.

AR:输出每一个 Radl 规格说明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 与已被确认的 Radl 规格说明 S_i 等价的证明证据,使用栈 *proof_stk* 来存储.返回值 *flag* 为 1 代表证明过程中使用到的推导规则包含逻辑蕴含式,为 0 代表证明过程中使用到的推导规则都是逻辑等价式.*/

```
1.  {
2.  spec_list:=[];
3.  proof_stk:=[];
4.  k:=1;
5.  WHILE k≤n
6.    {IF k≠i
7.       THEN AddQueue(spec_list,spec);}   //将欲证的各个子目标依次置入 spec_list 头部
8.  flag:=0;
9.  WHILE not QueueEmpty(spec_list) DO
10.   {spec←FirstQueue(spec_list);   //从 sg_list 的头部读取一个子目标
11.   SWITCH(spec)
12.     {CASE spec 是定律库中的前提条件或定理 Tj
13.       Push(proof_stk,(spec,'T',j));   //T 是定律标志,表示 spec 是前提条件或定理,j 是该定律在定律库
                                          中的序号
14.       RemoveQueue(spec_list);   //当前子目标得证,出队
15.     CASE spec 子项匹配定律库中逻辑蕴含定律 Lj
16.       IF spec 子项是 Lj 的实例//即 Lj 没有前件,即 Lj.right 为空
17.       THEN {Push(proof_stk,(spec,'L',j));   //压入证明证据
18.             RemoveQueue(spec_list);}   //当前子目标得证,出队
19.       ELSE {Push(proof_stk,(spec,'L',j));   //压入证明证据
20.             RemoveQueue(spec_list);}   //当前子目标出队
21.             spec:=Substitution(Lj.right);   //使用 Lj.right 替换 specs 中的子项,并将 spec 加入定律库
22.             Check(spec,spec_list);   //对 spec 中所有新子目标 t,检查 t 是否属于 spec_list
23.             AddQueue(spec_list,spec);}   //若 t 不属于 spec_list,置入 spec_list 头部
24.       flag:=1;
25.     CASE spec 子项匹配定律库中逻辑重言定律 Lj
26.       Push(proof_stk,(spec,'L',j));   //压入证明证据
27.       RemoveQueue(spec_list);}   //当前子目标出队
28.       Rewrite(spec,Lj,);   //使用 Lj 重写 spec 中的子项
29.       Check(spec,spec_list);   //对 spec 中所有新子目标 t,检查 t 是否属于 spec_list
30.       AddQueue(spec_list,spec);   //若 t 不属于 spec_list,置入 spec_list 头部
31.     DEFAULT:
32.       IF not StackEmpty(proof_stk) or Length(proof_stk)>Bound
33.       THEN {Pop(proof_stk);
```

34. GOTO 11;} //回溯
35. ELSE Exception;
36. }
37. }
38. *Print*(*proof_stk*); //以逆序方式打印输出 *proof_stk* 中的证据
39. }

根据第 2.1 节形式化推导方法,主函数按如下方式调用子函数 *Certify*,以证明 Radl 规格说明 $S_1,S_2,\ldots,S_{i-1},S_{i+1},\ldots,S_n$ 与 S_i 均等价:

1. *Certify*(S_i,*spec_list*−[S_i]);
2. IF (*flag*==1)
3. THEN *Certify*(S_{i-1},S_i);

这表明:若证明过程中用到的推导规则都是逻辑重言式,则主函数直接调用子函数 *Certify*(S_i,*spec_list*−[S_i])即可;如果使用到的推导规则包括逻辑蕴含式(*flag*==1),那么主函数还需要再调用子函数 *Certify*(S_{i-1},S_i).

算法 *Certify* 需要使用到两个抽象数据结构,其中,*spec_list* 为一个队列,用来存储证明过程中出现的各个子目标.包括队初始化:*spec_list*:=[],读对首操作:*spec*←*FirstQueue*(*spec_list*),进队操作:*AddQueue*(*spec_list*,*t*),出队操作:*RemoveQueue*(*spec_list*),读队首:*FirstQueue*(*spec_list*),判队空否:*QueueEmpty*(*spec_list*).*proof_stk* 为一堆栈,用来出示证据,即记录欲证当前子目标要使用的逻辑定律及其序号,其形式为三元组(*sub-goal*,*tag*,*index*).包括栈初始化:*proof_stk*:=[],进栈操作:*Push*(*proof_stk*,(*spec*,'*T*',*j*))或 *Push*(*proof_stk*,(*spec*,'*L*',*j*)),读栈顶:*Top*(*proof_stk*),出栈操作为 *Pop*(*proof_stk*),判栈空否:*StackEmpty*(*proof_stk*).

算法 *Certify* 证明过程可分下面几种情况进行讨论:

1) 如果当前子目标是定律库中的前提条件或定理,直接得证;
2) 如果当前子目标子项匹配定律库中某条逻辑蕴含定律 *Lj*,若该子项是 *Lj* 的实例,直接得证;否则,执行过程"替换(substitution)";
3) 如果当前子目标子项匹配定律库中逻辑重言定律 *Lj*,执行过程"重写(rewrite)";
4) 如果没有一条可应用的定律,若 *proof_stk* 不为空或当前搜索深度 *Length*(*proof_stk*)超过深度范围限制 Bound,则进行回溯;否则,执行过程 Exception.过程 Exception 需要人机交互,它向用户提示是否需要增加一条新的定律到定律库中.如果回答是肯定的,则系统添加一条新定律到定律库,继续 SWITCH 判断;否则,输出失败信息,而且证明过程终止.

假定欲证明等价的形式规格说明 Radl 个数为 *n*,*Certify* 回溯算法解空间树中从根节点到叶节点的最长路径的长度为 *d*,定律库 DB 中推导规则条数为 *b*,则 *Certify* 算法所需的计算时间至多为 nb^d.为了存储一个子目标的解向量,算法需要的计算空间为 *d*;为了存储 *n* 个子目标的解空间,算法需要的计算空间至多为 *nd*.通过对 *Certify* 回溯算法的时空复杂性分析可知,形式规格说明等价性判定问题实际上是一个 NP 完全问题.为了避免盲目搜索,提高推理效率,可以使用启发性策略来指导如何选择定律库 DB 中的推导规则.好的启发性策略可以减少盲目性,降低回溯次数,甚至不回溯就能得到解,总之一般来说有利于提高效率.

3 典型实例

本节将通过排序搜索、组合优化领域的两个典型实例对本文方法进行详细的阐述.

3.1 排序搜索:升序排序问题

问题 Sort:给定整型数组 *a*[0:*n*−1],将 *a* 中元素按升序排列.该问题需求后置断言 AR 可以使用各种形式 Radl 规格说明 $S_1,S_2,\ldots,S_5$ 表达如下(由于与证明过程关系不大,各个 Radl 规格说明中省略 *a*=*perm*(*b*),表示 *b* 是输入 *a* 的拷贝):

S_1. ($\forall i$:0≤*i*<*n*−1:*a*[*i*]≤*a*[*i*+1])

S_2. $(\forall i{:}0\leqslant i<n{:}(\forall j{:}0\leqslant j<n{:}i<j\Rightarrow a[i]\leqslant a[j]))$

S_3. $(\forall i,j{:}0\leqslant i,j<n{:}i<j\Rightarrow a[i]\leqslant a[j])$

S_4. $(\forall i{:}0\leqslant i<n{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))$

S_5. $(\forall i{:}0\leqslant i<n\wedge i<j<n{:}a[i]\leqslant a[j])$

下面根据第 2 节中介绍的 Radl 规格说明相对正确性验证方法,来验证上述不同形式规格说明 $S_1,S_2,\ldots,S_5$ 均正确.

3.1.1 步骤 1

选择问题需求 Sort 最为直截明了的 Radl 规格说明 S_1,使用 PAR 方法与 PAR 平台将 S_1 转换为可执行程序,通过测试来确认 S_1 满足问题 Sort 的需求.具体过程如下:

步骤 1.1. 选定问题需求 Sort 最为直截明了的 Radl 规格说明 S_1;

步骤 1.2. 使用 PAR 方法与 PAR 平台,将 S_1 经 Apla 抽象程序转换生成为 C#可执行语言程序;

其中,

- S_1 转换为 Apla 抽象程序的具体过程见文献[14],限于篇幅,这里从略;
- 而 Apla 抽象程序转换为 C#可执行语言程序是通过 PAR 平台自动转换工具 Apla→C#来进行的,如图 1 所示,其中,左部表示由 S_1 生成的 Apla 程序,右部表示由 Apla 生成的 C#程序.

步骤 1.3. 经测试得到的可执行程序正确,可以确认 Radl 形式规格说明 S_1 满足问题 Sort 的需求.

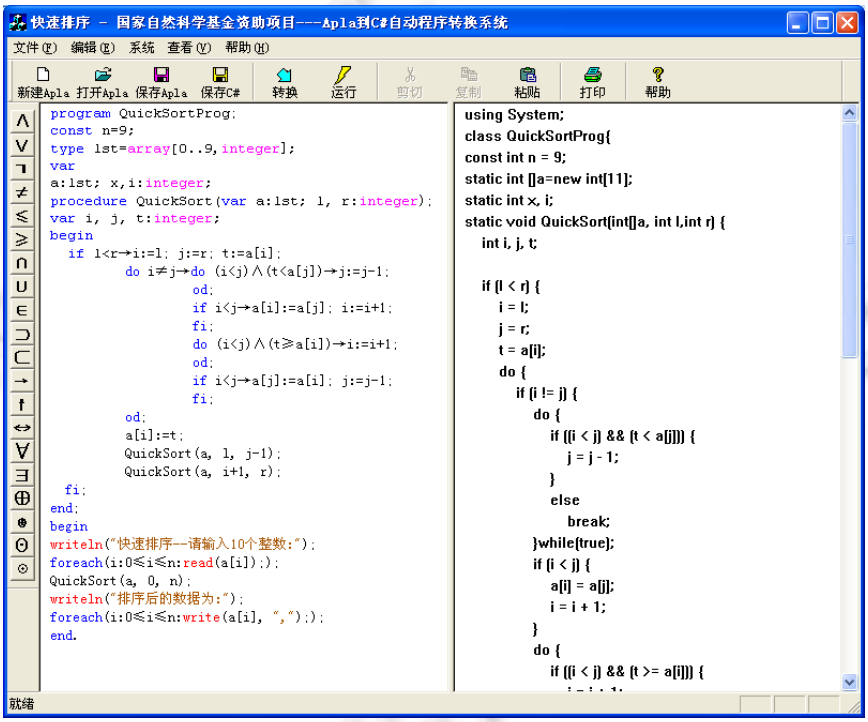

Fig.1 S_1 transform into C# via Apla

图 1 S_1 经 Apla 抽象程序转换为 C#程序

3.1.2 步骤 2

将经过确认的 Radl 形式规格说明 S_1 作为前提条件,利用第 1.2 节提出的扩展的逻辑系统来证明其他各种形式 Radl 形式规格说明 S_2,S_3,S_4,S_5 和已被确认的 Radl 形式规格说明 S_1 等价.

下面分别使用手工推导与自动推导来进行上述证明,具体推导过程分别如下:

1. 手工推导

由 $L1$.交叉积性质,易知 $S_2\equiv S_3, S_4\equiv S_5$;又由 $L5$.∀规则,可知 $S_3\equiv S_5$;

$S_4\equiv(\forall i{:}0\leqslant i<n{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))$

$\equiv${$L3$.范围分裂}

$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge(\forall j{:}n-1<j<n{:}a[n-1]\leqslant a[j])$

$\equiv${$L6.r(i)=F\Rightarrow(\forall i{:}r(i){:}f(i))\equiv T$}

$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge T$

$\equiv${$L7.P\wedge T\equiv P$}

$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))$

$\equiv${论域知识∵对所有的 $i,j,0\leqslant i<n-1, i<j<n, a[i]\leqslant a[j]$,于是 $a[i]\leqslant a[i+1]\leqslant\ldots\leqslant a[n-2]\leqslant a[n-1]$,∴对所有的 $i,0\leqslant i<n-1$,有 $a[i]\leqslant a[i+1]$}

$(\forall i{:}0\leqslant i<n-1{:}a[i]\leqslant a[i+1]))\equiv S_1$

于是,$S_1\equiv S_2\equiv S_3\equiv S_4\equiv S_5$,得证.

2. 自动推导

以上是通过手工推导所作的证明,推导路径较短,但是需要人的创造性工作.下面使用第 2.3 节提出的辅助证明算法来自动证明并出示证据,仍以上例进行说明.

首先将经过确认的 S_1 作为前提条件 $T1$ 加入定律库中,并将论域知识中特定领域的性质 $L101$ 也加入定律库:

- $T1$. $(\forall i{:}0\leqslant i<n-1{:}a[i]\leqslant a[i+1]))$
- $L101$. $(\forall j{:}i<j<n{:}a[i]\leqslant a[j])\equiv(\forall j{:}i<j<n{:}a[i]\leqslant a[i+1])$

欲证各种形式的 Radl 形式规格说明 S_2,S_3,S_4,S_5 和已被确认的 Radl 形式规格说明 S_1 等价,主函数需要调用子函数 $Certify(S_1,spec\_list-[S_1])$,即证明子目标 S_2,S_3,S_4 与 S_5 均成立.

首先执行子函数 $Certify(S_1,spec_list-[S_1])$中的步骤 1~步骤 3,设 $proof\_stk$ 为空,并将各个子目标 S_2,S_3,S_4 与 S_5 依次移入 $spec\_list$;

先取出子目标 S_2,形如:S_2. $(\forall i{:}0\leqslant i<n{:}(\forall j{:}0\leqslant j<n{:}i<j\Rightarrow a[i]\leqslant a[j]))$,对其进行证明.

在 DB 中查找到定律 $L1$.交叉积性质能与之匹配,进行合一运算:

$$u=\{0\leqslant i<n/r(i),0\leqslant j<n/s(i,j),i<j\Rightarrow a[i]\leqslant a[j]/f(i,j)\}.$$

定律 $L1$ 的左边重写为$(\forall i,j{:}0\leqslant i,j<n{:}i<j\Rightarrow a[i]\leqslant a[j])$,并作为定理加入定律库,标号为 $T2$;

继续在 DB 中查找到定律 $L5$.∀规则能与之匹配,进行合一运算:

$$u=\{i,j/i,0\leqslant i,j<n/r(i),i<j/s(i),a[i]\leqslant a[j]/p(i)\}.$$

定律 $L5$ 的左边重写为$(\forall i,j{:}0\leqslant i<n\wedge i<j<n{:}a[i]\leqslant a[j])$,并作为定理加入定律库,标号为 $T3$;

继续在 DB 中查找到定律 $L1$.交叉积性质能与之匹配,进行合一运算:

$$u=\{i<j<n/s(i,j),a[i]\leqslant a[j]/f(i,j)\}.$$

定律 $L1$ 的右边重写为$(\forall i{:}0\leqslant i<n{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))$,并作为定理加入定律库,标号为 $T4$;

继续在 DB 中查找到定律 $L3$.范围分裂能与之匹配,进行合一运算,定律 $L3$ 的右边重写为

$$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge(\forall j{:}n-1<j<n{:}a[n-1]\leqslant a[j]),$$

并作为定理加入定律库,标号为 $T5$;

继续在 DB 中查找到定律 $L6$.恒等元性质与之匹配,进行合一运算,定律 $L6$ 的右边重写为

$$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge T,$$

并作为定理加入定律库,标号为 $T6$;

继续在 DB 中查找到同一律 $L7$ 能与之匹配,进行合一运算,定律 $L7$ 的右边重写为

$$(\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j])),$$

并作为定理加入定律库,标号为 $T7$;

继续在 DB 中查找到特定领域的性质 $L101$ 能与之匹配,进行合一运算,定律 $L101$ 的右边重写为

$$(\forall i{:}0\leqslant i<n-1{:}a[i]\leqslant a[i+1]).$$

这时,在 DB 中查找到该式已经是定律库中的前提 $T1$,故子目标 S_2 得证.

然后,从 *spec_list* 中依次取出子目标 S_3,S_4 与 S_5 分别进行证明,我们发现,它们均已是定律库中的定理 $T2$, $T4$ 与 $T3$ 了,故子目标 S_3,S_4 与 S_5 亦得证.

现在 *spec_list* 为空,跳出 WHILE 循环,执行 *print*(*proof_stk*),以逆序打印 *proof_stk* 中的证据如下:

$((\forall i{:}0\leqslant i<n{:}(\forall j{:}0\leqslant j<n{:}i<j\Rightarrow a[i]\leqslant a[j])),\text{‘}L\text{’},1)$ //证明 S_2

$((\forall i,j{:}0\leqslant i,j<n{:}i<j\Rightarrow a[i]\leqslant a[j]),\text{‘}L\text{’},5)$ //加入定律库,标号为 $T2$

$((\forall i,j{:}0\leqslant i<n\wedge i<j<n{:}a[i]\leqslant a[j]),\text{‘}L\text{’},1)$ //加入定律库,标号为 $T3$

$((\forall i{:}0\leqslant i<n{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j])),\text{‘}L\text{’},3)$ //加入定律库,标号为 $T4$

$((\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge(\forall j{:}n-1<j<n{:}a[n-1]\leqslant a[j]),\text{‘}L\text{’},6)$ //加入定律库,标号为 $T5$

$((\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j]))\wedge T,\text{‘}L\text{’},7)$ //加入定律库,标号为 $T6$

$((\forall i{:}0\leqslant i<n-1{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j])),\text{‘}L\text{’},101)$ //加入定律库,标号为 $T7$

$((\forall i{:}0\leqslant i<n-1{:}a[i]\leqslant a[i+1]),\text{‘}T\text{’},1)$

$((\forall i,j{:}0\leqslant i,j<n{:}i<j\Rightarrow a[i]\leqslant a[j]),\text{‘}T\text{’},2)$ //证明 S_3

$((\forall i{:}0\leqslant i<n{:}(\forall j{:}i<j<n{:}a[i]\leqslant a[j])),\text{‘}T\text{’},4)$ //证明 S_4

$((\forall i{:}0\leqslant i<n\wedge i<j<n{:}a[i]\leqslant a[j]),\text{‘}T\text{’},3)$ //证明 S_5

主函数调用子函数 *Certify*(S_1,*spec_list*−[S_1])完成后返回,这时候,由于返回值 *flag* 不为 1,算法结束.

3.2 组合优化:最小和问题

问题 Minsum:给定整型数组 $a[0{:}n-1]$,求 a 中最小相邻元素之和.该问题需求后置断言 AR 可以使用各种形式的 Radl 规格说明 $S_1,S_2,\ldots,S_8$ 表达如下:

S_1. min*sum*(n)=(MIN $i{:}0\leqslant i\leqslant j<n{:}\sum(k{:}i\leqslant k<j{:}a[k])$)

S_2. min*sum*(n)=(MIN $i{:}0\leqslant i<n\wedge i\leqslant j<n{:}\sum(k{:}i\leqslant k<j{:}a[k])$)

S_3. min*sum*(n)=(MIN $i{:}0\leqslant i<n{:}$(MIN $j{:}i\leqslant j<n{:}\sum(k{:}i\leqslant k<j{:}a[k])$))

S_4. min*sum*(n)=(MIN $i,j{:}0\leqslant j<n\wedge 0\leqslant i\leqslant j{:}\sum(k{:}i\leqslant k<j{:}a[k])$)

S_5. min*sum*(n)=(MIN $j{:}0\leqslant j<n{:}$(MIN $i{:}0\leqslant i\leqslant j{:}\sum(k{:}i\leqslant k<j{:}a[k])$))

S_6. min*sum*(n)=(MIN $i{:}0\leqslant i\leqslant j<n{:}$*sum*($i$,$j$))

其中,*sum*(i,j)=($\sum k{:}i\leqslant k<j{:}a[k]$)表示数组段 $a[i,...,j]$元素之和.

S_7. min*sum*(n)=min(min*sum*($n-1$),*ms*(n))

其中,*ms*(n)=(MIN $i{:}0\leqslant i\leqslant n{:}$*sum*($i$,$n$))表示以 n 为最右下标的数组段 $a[0,...,n]$元素之和.

S_8. min*sum*(n)=min(min*sum*($n-1$),*ms*(n))

ms(n)=(*ms*($n-1$)+$a[n]$,$a[n]$).

下面要根据第 2 节中介绍的 Radl 规格说明相对正确性验证方法,来验证上述不同形式规格说明 $S_1,S_2,\ldots,S_8$ 均正确.

3.2.1 步骤 1

选择问题需求 Min*sum* 最为直截明了的 Radl 规格说明 S_1,使用 PAR 方法与 PAR 平台将 S_1 转换为可执行程序,通过测试来确认 S_1 满足问题 Min*sum* 的需求.具体过程如下:

步骤 1.1. 选定问题需求 Min*sum* 最为直截明了的 Radl 规格说明 S_1;

步骤 1.2. 使用 PAR 方法与 PAR 平台,将 S_1 经 Apla 抽象程序转换生成为 C#可执行语言程序;

其中,

- S_1 转换为 Apla 抽象程序具体过程参考文献[14],限于篇幅,这里从略;

- 而 Apla 抽象程序转换为 C#可执行语言程序是通过 PAR 平台自动转换工具 Apla→C#来进行的,如图 2 所示,其中,左部表示由 S_1 生成的 Apla 程序,右部表示由 Apla 生成的 C#程序.

步骤 1.3. 经测试得到的可执行程序正确,可以确认 Radl 形式规格说明 S_1 满足问题 Min*sum* 的需求.

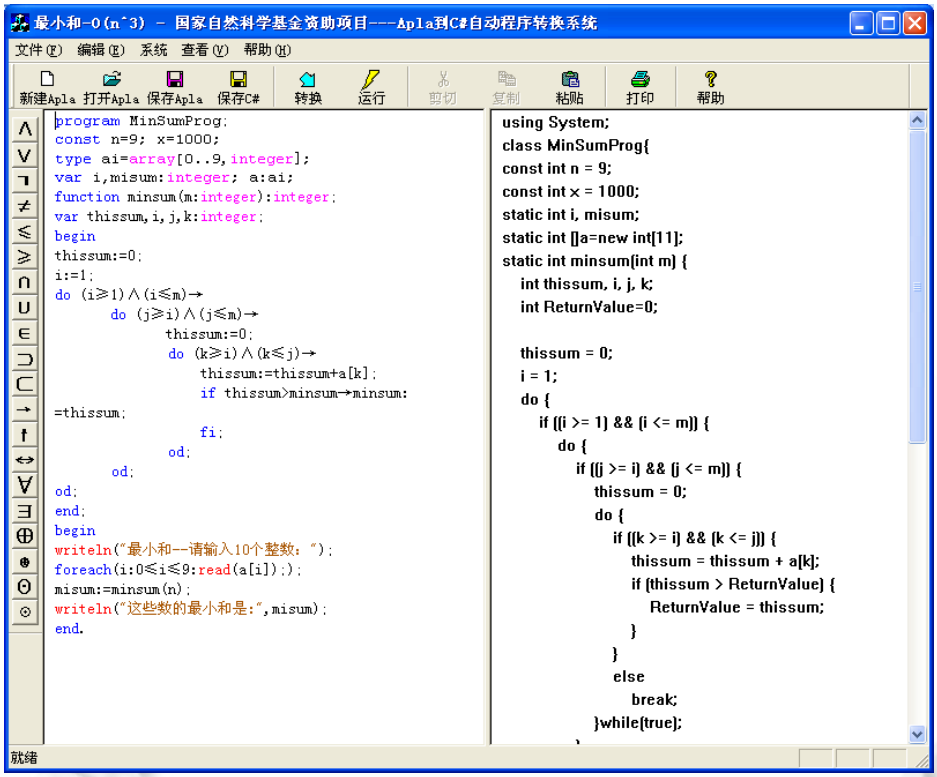

Fig.2 S_1 transform into C# via Apla

图 2 S_1 经 Apla 抽象程序转换为 C#程序

3.2.2 步骤 2

将经过确认的 Radl 形式规格说明 S_1 作为前提条件,利用第 1.2 节中提出的扩展的逻辑系统来证明其他各种形式的 Radl 形式规格说明 $S_2,S_3,\ldots,S_8$ 和已被确认的 Radl 形式规格说明 S_1 等价.

下面分别使用手工推导与自动推导来进行上述证明,具体推导过程分别如下:

1. 手工推导

由论域知识 $0\leqslant i\leqslant j<n\equiv 0\leqslant i<n\wedge i\leqslant j<n\equiv 0\leqslant j<n\wedge 0\leqslant i\leqslant j, S_1=S_2=S_4$

$S_1\equiv minsum(n)=(\text{MIN } i{:}0\leqslant i\leqslant j<n{:}\sum(k{:}i\leqslant k<j{:}a[k])$

$\equiv${引入新的定义 $sum(i,j)=(\sum k{:}i\leqslant k<j{:}a[k])$}

$minsum(n)=(\text{MIN } i{:}0\leqslant i\leqslant j<n{:}sum(i,j))=S_6$

$S_2\equiv minsum(n)=(\text{MIN } i{:}0\leqslant i<n\wedge i\leqslant j<n{:}\sum(k{:}i\leqslant k<j{:}a[k]))$

$\equiv${$L1$.交叉积性质}

$minsum(n)=(\text{MIN } i{:}0\leqslant i<n{:}(\text{MIN } j{:}i\leqslant j<n{:}(\sum k{:}i\leqslant k<j{:}a[k])))\equiv S_3$

$S_4\equiv minsum(n)=(\text{MIN } i,j{:}0\leqslant j<n\wedge 0\leqslant i\leqslant j{:}\sum(k{:}i\leqslant k<j{:}a[k]))$

$\equiv${$L1$.交叉积性质}

$minsum(n)=(\text{MIN } j{:}0\leqslant j<n{:}(\text{MIN } i{:}0\leqslant i\leqslant j{:}\sum(k{:}i\leqslant k<j{:}a[k]))\equiv S_5$

S_6≡min*sum*(*n*)=(MIN *i*:0≤*i*≤*j*<*n*:*sum*(*i*,*j*))

{*L*1.交叉积性质}

≡min*sum*(*n*)=(MIN *j*:0≤*j*<*n*:(MIN *i*:0≤*i*≤*j*:*sum*(*i*,*j*))

{引入新的定义 *ms*(*j*)=(MIN *i*:0≤*i*≤*j*:*sum*(*i*,*j*))}

≡min*sum*(*n*)=(MIN *j*:0≤*j*<*n*:*ms*(*j*))

≡{*L*3.范围分裂与 *L*2.单点范围}

min*sum*(*n*)=min(MIN *j*:0≤*j*<*n*−1:*ms*(*j*)),*ms*(*n*))

≡{min*sum*(*n*)=(MIN *j*:0≤*j*<*n*:*ms*(*j*))

min*sum*(*n*)=min(min*sum*(*n*−1),*ms*(*n*))≡S_7

由于:

ms(*n*)≡(MIN *i*:0≤*i*≤*n*:*sum*(*i*,*n*))

≡{*sum*(*i*,*j*)定义展开}

(MIN *i*:0≤*i*≤*n*:(∑*j*:*i*≤*j*<*n*:*a*[*j*]))

≡{*L*3.范围分裂与 *L*2.单点范围}

ms(*n*)≡(MIN *i*:0≤*j*≤*n*:(∑*j*:*i*≤*j*<*n*−1:*a*[*j*])+*a*[*n*])

≡{*L*4.一般分配律与 *sum*(*i*,*j*)定义折叠}

ms(*n*)≡(MIN *i*:0≤*j*≤*n*:*sum*(*i*,*n*−1))+*a*[*n*]

≡{*L*3.范围分裂与 *L*2.单点范围}

ms(*n*)≡min((MIN *i*:0≤*j*≤*n*−1:*sum*(*i*,*n*−1)),*sum*(*n*,*n*−1))+*a*[*n*]

≡{*ms*(*j*)定义折叠}

ms(*n*)≡min(*ms*(*n*−1),0)+*a*[*n*]

≡{+对 min 的分配律}

ms(*n*)≡min(*ms*(*n*−1)+*a*[*n*],*a*[*n*])

因此,S_7≡S_8.

综上所述,相继证明得到 S_1=S_2=S_4=S_6=S_3=S_5=S_7=S_8.

2. 自动推导

以上是通过手工推导进行证明,推导路径较短,但是需要人的创造性工作.下面使用第 2.3 节中提出的辅助证明算法来自动证明并出示证据,仍以上例加以说明.

首先将经过确认的 S_1 作为前提条件 *T*10 加入定律库中,并将论域知识中的特定领域的性质 *L*110,*L*114、定义 *L*111~定义 *L*113 也加入定律库:

*T*10. min*sum*(*n*)=(MIN *i*:0≤*i*≤*j*<*n*:∑(*k*:*i*≤*k*<*j*:*a*[*k*]))

*L*110. 0≤*i*≤*j*<*n*≡0≤*i*<*n*∧*i*≤*j*<*n*≡0≤*j*<*n*∧0≤*i*≤*j*

*L*111. *sum*(*i*,*j*)=∑(*k*:*i*≤*k*<*j*:*a*[*k*])

*L*112. *ms*(*n*)=(MIN *i*:0≤*i*≤*n*:*sum*(*i*,*n*))

*L*113. min*sum*(*n*)=(MIN *j*:0≤*j*<*n*:*ms*(*j*))

*L*114. *a*+min(*b*,*c*)=min(*a*+*b*,*a*+*c*)

欲证各种形式的 Radl 形式规格说明 S_2,S_3,...,S_8 和已被确认的 Radl 形式规格说明 S_1 等价,主函数需要调用子函数 *Certify*(S_1,*spec_list*−[S_1]),即证明子目标 S_2,S_3,…,S_8 均成立.

首先执行子函数 *Certify*(S_1,*spec_list*−[S_1])中的步骤 1~步骤 3,设 *proof_stk* 为空,并将各个子目标 S_2,S_3,…,S_4 与 S_5 依次移入 *spec_list*;

先取出子目标 S_2,形如:S_2. min*sum*(*n*)=(MIN *i*:0≤*i*<*n*∧*i*≤*j*<*n*:∑(*k*:*i*≤*k*<*j*:*a*[*k*])),对其进行证明.

在 DB 中查找到定律特定领域的性质 *L*110 能与之匹配,进行合一运算:

$$u=\{0\leqslant i\leqslant j<n\equiv 0\leqslant i<n/0\leqslant i<n\wedge i\leqslant j<n\equiv 0\}.$$

S_2 重写为 min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:$\sum(k$:$i\leqslant k<j$:$a[k]$)),这时,在 DB 中查找到该式已经是定律库中的前提 $T10$,故子目标 S_2 得证.将 S_2 作为定理加入定律库,标号为 $T11$;

然后取出子目标 S_3,形如:S_3. min$sum(n)$=(MIN i:$0\leqslant i<n$:(MIN j:$i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$))),对其进行证明.

在 DB 中查找到定律 $L1$.交叉积性质能与之匹配,进行合一运算:

$$u=\{0\leqslant i<n\wedge i\leqslant j<n/s(i,j),(\sum k:i\leqslant k<j:a[k])/f(i,j)\}.$$

S_3 重写为 min$sum(n)$=(MIN i:$0\leqslant i<n\wedge i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$)),这时,在 DB 中查找到该式已经是定律库中的定理 $T11$,故子目标 S_3 得证.将 S_3 作为定理加入定律库,标号为 $T12$.

同理,可证明子目标 S_4,S_5,S_6,S_7 与 S_8.

现在 *spec_list* 为空,跳出 WHILE 循环,执行 *print*(*proof_stk*),以逆序打印 *proof_stk* 中的证据如下:

(min$sum(n)$=(MIN i:$0\leqslant i<n\wedge i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$)),‘$L$’,110) //证明 S_2

(min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$),‘$T$’,10) //加入定律库,标号为 $T11$

(min$sum(n)$=(MIN i:$0\leqslant i<n$:(MIN j:$i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$))),‘L’,1) //证明 S_3

(min$sum(n)$=(MIN i:$0\leqslant i<n\wedge i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$)),‘$T$’,11) //加入定律库,标号为 $T12$

(min$sum(n)$=(MIN i,j:$0\leqslant j<n\wedge 0\leqslant i\leqslant j$:($\sum k$:$i\leqslant k<j$:$a[k]$)),‘$L$’,110) //证明 S_4

(min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$)),‘$T$’,10) //加入定律库,标号为 $T13$

(min$sum(n)$=(MIN j:$0\leqslant j<n$:(MIN i:$0\leqslant i\leqslant j$:$\sum(k$:$i\leqslant k<j$:$a[k]$))),‘L’,1) //证明 S_5

(min$sum(n)$=(MIN i,j:$0\leqslant j<n\wedge 0\leqslant i\leqslant j$:($\sum k$:$i\leqslant k<j$:$a[k]$)),‘$T$’,13) //加入定律库,标号为 $T14$

(min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:$\sum(k$:$i\leqslant k<j$:$a[k]$)),‘$L$’,111) //证明 S_6

(min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:($\sum k$:$i\leqslant k<j$:$a[k]$),‘$T$’,10) //加入定律库,标号为 $T15$

(min$sum(n)\equiv$min(min$sum(n-1)$,$ms(n)$)),‘L’,113) //证明 S_7

(min$sum(n)$=min(MIN j:$0\leqslant j<n-1$:$ms(j)$),$ms(n)$)),‘L’,3) //加入定律库,标号为 $T16$

(min$sum(n)$=min(MIN j:$0\leqslant j<n-1$:$ms(j)$),$ms(n)$)),‘L’,2)

(min$sum(n)\equiv$(MIN j:$0\leqslant j<n$:$ms(j)$),‘L’,112) //加入定律库,标号为 $T17$

(min$sum(n)\equiv$(MIN j:$0\leqslant j<n$:(MIN i:$0\leqslant i\leqslant j$:$sum(i,j)$))),‘L’,1) //加入定律库,标号为 $T18$

(min$sum(n)$=(MIN i:$0\leqslant i\leqslant j<n$:$sum(i,j)$)),‘$T$’,15) //加入定律库,标号为 $T19$

(min$sum(n)$=min(min$sum(n-1)$,$ms(n)$)),‘T’,16) //证明 $S_8(a)$

($ms(n)$=($ms(n-1)$+$a[n]$,$a[n]$),‘L’,114) //证明 $S_8(b)$

($ms(n)\equiv$min($ms(n-1)$,0)+$a[n]$,‘L’,112) //加入定律库,标号为 $T20$

($ms(n)\equiv$min((MIN i:$0\leqslant j\leqslant n-1$:$sum(i,n-1)$),0)+$a[n]$,‘$L$’,3)

($ms(n)\equiv$(MIN i:$0\leqslant j\leqslant n$:$sum(i,n-1)$)+$a[n]$,‘L’,2) //加入定律库,标号为 $T21$

($ms(n)\equiv$(MIN i:$0\leqslant j\leqslant n$:$sum(i,n-1)$)+$a[n]$,‘L’,4) //加入定律库,标号为 $T22$

($ms(n)\equiv$(MIN i:$0\leqslant j\leqslant n$:$sum(i,n-1)$)+$a[n]$),‘L’,111) //加入定律库,标号为 $T23$

($ms(n)\equiv$(MIN i:$0\leqslant j\leqslant n$:($\sum j$:$i\leqslant j<n-1$:$a[j]$))+$a[n]$),‘L’,3)

((MIN i:$0\leqslant i\leqslant n$:($\sum j$:$i\leqslant j<n$:$a[j]$))‘L’,2) //加入定律库,标号为 $T24$

((MIN i:$0\leqslant i\leqslant n$:$\sum(j$:$i\leqslant j<n$:$a[j]$)),‘L’,111) //加入定律库,标号为 $T25$

($ms(n)\equiv$(MIN i:$0\leqslant i\leqslant n$:$sum(i,n)$)),‘L’,112) //加入定律库,标号为 $T26$

主函数调用子函数 *Certify*(S_1,*spec_list*−[S_1])完成后返回,这时,由于返回值 *flag* 不为 1,算法结束.

4 相关工作比较

国内外学者在形式规格说明的正确性验证方面开展了许多研究工作,归纳起来,目前使用的技术主要包括测试、人工评审、原型法、动画与模型检查等.

测试[15−18]包括使用测试数据来运行软件的实现,检验软件的输出及其操作行为,以检查它是否如所要求的那样执行.测试是一种动态确认技术;人工评审[19]通过分析和检查系统表示来发现错误、遗漏和异常,例如需求文档、设计图和程序源代码,可以辅之以对某些系统源文本或者是相关文档的自动分析.评审及其自动化分析是静态确认技术,因为无需在计算机上运行系统;原型法[20]通过呈现各种图形用户界面,产生对各种用户事件的模拟的系统行为,以直观的方式与需求提供者沟通,以便需求提供者确认系统行为,或者提出对系统行为的修改意见;动画[21−24]通过模拟可执行的操作模型并将模拟结果以可视化形式反馈给用户;模型检查[25−27]用于检查行为模型是否违背时序逻辑属性,如果违背,还将给出反例的执行轨迹.

上述方法中,测试、人工评审、原型法、动画均属于非形式化方法,模型检查属于形式化方法.非形式化方法与形式化方法相比,严格性不够,但是强调证明过程中用户的重要作用,证明花费的代价较少;而形式化方法要求具有专业化的符号系统,用户难以理解和掌握,确认花费的代价也较高.

注意到问题需求的非(半)形式化与形式规格说明的形式化两者之间差异的本性,要完全形式化地证明同一问题不同形式 Radl 规格说明的正确性几乎是不可能的.本文提出一种基于形式化推导的方法来验证同一问题不同形式规格说明的相对正确性,规格说明相对正确性是指相对一定前提条件下的正确性,即假定问题 P 某个最为直截明了的形式规格说明 S_i 已经通过测试得到确认.该方法将形式化方法与非形式化方法相结合,既利用了形式化方法(等价性证明)的严格性,又利用了非形式化方法(测试确认)的用户参与性;该方法还有效地区分了证明过程中的形式化部分与非形式化部分,追求整个证明过程尽可能多的形式化(理性因素),减少非形式化的部分(主观因素),使得整个证明过程高可信.具体来说,基于形式化推导的相对正确性验证方法执行了两个步骤:步骤 1 包含形式化部分(由 Radl 规格说明经 Apla 转换为可执行程序)与非形式化部分(测试确认);步骤 2 仅包含形式化部分(证明等价性),整个验证过程非形式化部分极少.该方法主要的创造性在于有助于增强获取得到的形式规格说明的可信性.

5 总结与展望

在形式规格说明的获取任务中,一个重要问题是获取得到的形式规格说明的正确性.本文提出一种基于形式化推导的方法来验证同一问题不同形式规格说明的相对正确性,为了支持该方法,进一步提出了一种扩展的逻辑系统与辅助证明算法,最后,通过排序搜索、组合优化领域的两个典型实例对该方法进行了详细的阐述.实际使用效果表明,该方法能够有效地验证 Radl 形式规格说明的正确性.使用本文提出的 Radl 形式规格说明相对正确性证明方法,已经成功地证明了排序查找、组合优化(包括树、图等复杂数据结构)领域许多算法程序形式规格说明的相对正确性,如排序问题、搜索问题、最小和问题、最大平台问题、最小生成树问题、单源最短路径问题等,结果令人鼓舞.该方法还具备良好的可扩充性,主要体现在如下两点:一是只要在扩充的逻辑系统的定律库中增加关系代数等价变换规则,就能有效地验证数据库应用程序形式规格说明的相对正确性(注:Radl 语言目前已经能够支持算法程序、数据库应用程序与软件构件形式化规格说明的描述及后继的开发);二是本文虽然使用 Radl 语言作为形式规格说明语言,但是该方法可以扩展到一般的形式规格说明语言,如 Z 语言(Z 模式类似于 Radl 规格说明)、B 语言(抽象机类似于 Radl 规格说明)、VDM 及其后续的 Raise 语言(前后置断言类似于 Radl 规格说明)等.

该方法不仅在规格说明正确性验证领域,还在算法优化、程序等价性证明等研究领域具有潜在的理论意义与应用价值,说明如下:

(1) 使用扩展的逻辑系统定律库中的推导规则,通过形式化推导可以验证不同形式 Radl 规格说明的等价性,这将在 Radl 规格说明正确性验证中得到应用,如本文所述;

(2) 使用扩展的逻辑系统定律库中的推导规则,从 Radl 规格说明出发(一般是一种低效算法)通过形式化推导可以得到高效的 Radl 算法,这将在算法优化中得到应用;

(3) 通过 PAR 方法与 PAR 平台,每一个 Radl 规格说明经过转换对应到一个可执行语言程序(如 C#).反过来,通过谓词抽象或者逆向工程,同一问题不同程序的正确性问题可以逆向转换为同一问题不同

Radl 规格说明的正确性问题,这将在程序自动评判中得到应用.众所周知,程序自动评判迄今为止仍然是一个公开的难题,这为该问题的解决提供了一条新的思路.

我们进一步的工作将在上述领域继续探索,深入挖掘本文所提方法潜在的理论意义与应用价值.

References:

[1] Bjorner D, Henson MC. Logics of Specification Languages. Springer-Verlag, 2009. [doi: 10.1007/978-3-540-74107-7]

[2] Boca PP, Bowen JP, Siddiqi JI. Formal Methods: State of the Art and New Directions. Springer-Verlag, 2009. [doi: 10.1007/978-1-84882-736-3]

[3] Jin Z, Liu L, Jin Y. Requirement Engineering of Software: Principle and Methods. Beijing: Science Press, 2008 (in Chinese).

[4] Cheng BHC, Atlee JM. Research directions in requirements engineering. In: Briand L, Wolf AL, eds. Proc. of the Future of Software Engineering. Washington: IEEE Computer Society, 2007. [doi: 10.1109/FOSE.2007.17]

[5] Liu SY. A rigorous approach to reviewing formal specifications. In: Proc. of the 27th Annual IEEE/NASA Int'l Software Engineering Workshop. Washington: IEEE Computer Society, 2002. 75−81. [doi: 10.1109/SEW.2002.1199452]

[6] Xue JY. Two new strategies for developing loop invariants and their applications. Journal of Computer Science and Technology, 1993,8(2):147−154. [doi: 10.1007/BF02939477]

[7] Xue JY. A unified approach for developing efficient algorithmic programs. Journal of Computer Science and Technology, 1997, 12(4):314−329. [doi: 10.1007/BF02943151]

[8] Xue JY. Formal derivation of graph algorithmic programs using partition-and-recur. Journal of Computer Science and Technology, 1998,13(6):553−561. [doi: 10.1007/BF02946498]

[9] Xue JY, Yang B, Zuo ZK. A linear in-situ algorithm for the power of cyclic permutation. In: Proc. of the 2nd Int'l Frontiers of Algorithmics Workshop. Berlin: Springer-Verlag, 2008. 113−123. [doi: 10.1007/978-3-540-69311-6_14]

[10] Xue JY. PAR method and its supporting platform. In: Proc. of the 1st Int'l Workshop on Asian Working Conf. on Verified Software. Macao: UNU-IIST, 2006. 10−20.

[11] Shi HH, Xue JY. PAR-Based formal development of algorithms. Chinese Journal of Computers, 2009,32(5):982−991 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2009.00982]

[12] You Z, Xue JY. Formal verification of algorithmic programs based on the isabelle theorem prover. Computer Engineering and Science, 2009,31(10):85−89 (in Chinese with English abstract). [doi: 10.3969/j.issn.1007-130X.2009.10.024]

[13] Yang L, Xue JY, Wan Y. The research of PAR platform-oriented test case generation. Microcomputer Information, 2009,25(11): 204−205 (in Chinese with English abstract).

[14] Wang CJ, Xue JY. Algorithmic derivation and automatic generation from specification on PAR platform. Computer Engineering and Applications, 2007,43(2):41−42 (in Chinese with English abstract).

[15] Liu SY. Verifying consistency and validity of formal specifications by testing. In: Wing JM, Woodcock J, Davies J, eds. Proc. of the World Congress on Formal Methods in the Development of Computing Systems. LNCS 1708, Berlin, Heidelberg: Springer-Verlag, 1999. 896−914. [doi: 10.1007/3-540-48119-2_49]

[16] Miao HK, Liu L. An approach to formalizing specification-based class testing. Journal of Shanghai University, 2006,10(1):25−32. [doi: 10.1007/s11741-006-0101-y]

[17] Yan J, Zhang J. Combinatorial testing: Principles and methods. Ruanjian Xuebao/Journal of Software, 2009,20(6):1393−1405 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3497.htm [doi: 10.3724/SP.J.1001.2009.03497]

[18] Dong YM, Li KD, Chen HM, Hu YQ, Zhang RL, Tang RQ, Wan ZY, Chen ZM. Design and implementation of the formal specification acquisition system SAQ. In: Feng YL, Notkin D, Gaudel MC, eds. Proc. of the Conf. on Software: Theory and Practice, IFIP 16th World Computer Congress 2000. Beijing: Publishing House of Electronics Industry, 2000. 201−211.

[19] Liu SY. An automated rigorous review method for verifying and validating formal specifications. In: Proc. of the 2nd Int'l Symp. on Automated Technology for Verification and Analysis. LNCS 3299, Berlin, Heidelberg: Springer-Verlag, 2004. 15−19. [doi: 10.1007/978-3-540-30476-0_6]

[20] Thompson JM, Heimdahl MPE, Miller SP. Specification-Based prototyping for embedded systems. In: Nierstrasz O, Lemoine M, eds. Proc. of the ESEC'99 and ESEC-FSE'99. LNCS 1687, Berlin, Heidelberg: Springer-Verlag, 1999. 163−179. [doi: 10.1007/3-540-48166-4_11]

[21] Heitmeyer CL, Kirby J, Labaw BG, Bharadwaj R. SCR*: A toolset for specifying and analyzing software requirements. In: Vardi YM, ed. Proc. of the CAV'98. LNCS 1427, Berlin, Heidelberg: Springer-Verlag, 1998. 526−531. [doi: 10.1007/BFb0028775]

[22] Magee J, Pryce N, Giannakopoulou D, Kramer J. Graphical animation of behavior models. In: Proc. of the IEEE Int'l Conf. on Soft. Eng. (ICSE). Washington: IEEE Computer Society, 2000. 499−508. [doi: 10.1109/ICSE.2000.10106]

[23] Van HT, van Lamsweerde A, Massonet P, Ponsard C. Goal-Oriented requirements animation. In: Proc. of the IEEE Int'l Req. Eng. Conf. (RE). Washington: IEEE Computer Society, 2004. 218−228. [doi: 10.1109/ICRE.2004.1335679]

[24] Zhu J, Chen YH, Miao HK. Structure animation of Object-Z specification. Journal of Shanghai University, 2005,11(6):589−595 (in Chinese with English abstract).

[25] Anderson RJ, Beame P, Burns S, Chan W, Modugno F, Notkin D, Reese JD. Model checking large software specifications. In: Garlan D, ed. Proc. of the 4th ACM SIGSOFT Symp. on the Foundations of Software Engineering. New York: ACM Press, 1996. 156−166. [doi: 10.1145/239098.239127]

[26] Easterbrook S, Chechik M. A framework for multi-valued reasoning over inconsistent viewpoints. In: Proc. of the IEEE Int'l Conf. on Soft. Eng. (ICSE). Washington: IEEE Computer Society, 2001. 411−420. [doi: 10.1109/ICSE.2001.919114]

[27] Sreemani T, Atlee JM. Feasibility of model checking software requirements. In: Proc. of the Conf. on Comp. Assur. National Institute of Standards and Technology, 1996. 77−88. [doi: 10.1109/CMPASS.1996.507877]

附中文参考文献:

[3] 金芝,刘璘,金英.软件需求工程:原理与方法.北京:科学出版社,2008.

[11] 石海鹤,薛锦云.基于 PAR 的算法形式化开发.计算机学报,2009,32(5):982−991. [doi: 10.3724/SP.J.1016.2009.00982]

[12] 游珍,薛锦云.基于 Isabelle 定理证明器算法程序的形式化验证.计算机工程与科学,2009,31(10):85−89. [doi: 10.3969/j.issn.1007-130X.2009.10.024]

[13] 杨乐,薛锦云,万韵.面向 PAR 平台的测试用例生成技术研究.微计算机信息,2009,25(11):204−205.

[14] 王昌晶,薛锦云.PAR 平台从规约出发的算法推导与自动生成.计算机工程与应用,2007,43(2):41−42.

[17] 严俊,张健.组合测试:原理与方法.软件学报,2009,20(6):1393−1405. http://www.jos.org.cn/1000-9825/3497.htm [doi: 10.3724/SP.J.1001.2009.03497]

[24] 朱江,陈怡海,缪淮扣.Object-Z 规格说明的结构模拟动画技术.上海大学学报,2005,11(6):589−595.

王昌晶(1977－),男,江西丰城人,博士,副教授,CCF 学生会员,主要研究领域为软件形式化与自动化,软件规格说明方法.
E-mail: wcj771006@163.com

薛锦云(1947－),男,教授,博士生导师,CCF 高级会员,主要研究领域为软件形式化与自动化,可信软件构造与验证.
E-mail: jinyun@vip.sina.com