文章编号:1000-5862(2020)04-0378-07

模型驱动的 Dafny 程序形式化生成与自动验证

王昌晶[1],贺江飞[1],罗海梅[2,3],左正康[1*],许 帆[1]
(1. 江西师范大学计算机信息工程学院,江西 南昌 330022;2. 江西师范大学物理与通信电子学院,江西 南昌 330022;
3. 江西师范大学江西省光电子与通信重点实验室,江西 南昌 330022)

摘要: Dafny 是一种内置规范结构的编程语言和静态程序证明器,它能验证程序的功能正确性以及将证明过程自动化,这既提高了软件开发的效率,又极大增强了软件开发的可靠性. 该文探索了一种模型驱动的 Dafny 程序形式化生成的方法. 首先,从问题的 Radl 规约出发,根据规约变换技术得到其 Radl 算法;然后,根据 PAR 方法中循环不变式开发新策略得到问题的循环不变式;最后,在 Radl 算法和循环不变式基础上利用模型等价转换规则生成 Dafny 程序,并由 Dafny 证明器自动验证其功能正确性. 用该方法解决了2 个典型问题的算法程序开发与验证,证实了该方法能够有效地提高 Dafny 程序的生成效率和可靠性.

关键词: 模型驱动;Dafny 程序;循环不变式;形式化生成;自动验证
中图分类号: TP 311 **文献标志码:** A **DOI:** 10.16357/j.cnki.issn1000-5862.2020.04.09

0 引言

近年来,随着形式验证技术的发展,形式验证工具在程序验证中的成功应用不断增多. 由于形式化方法在开发复杂和可靠软件中的优势,形式验证工具在未来一定会大大提高验证可靠算法软件的效率,如微软、华为招聘专家和优秀毕业生专门从事形式化方法及验证工具的研究与开发工作,希望提高日益庞大的商业软件的开发效率和可靠性[1]. 与此同时,由于自动证明理论和 CPU 计算能力的高速发展,定理证明器的能力不断增强,促进了基于定理证明器的程序证明器的快速发展,出现了许多设计优良的程序证明器. 程序证明器的优点在于验证的效率较高,不需要人们手工写证明,Dafny 就是其中最具代表性的一个程序证明器[2]. 它可以验证程序的功能正确性,并提供完全的自动化验证,这极大减少了对程序进行全面验证的工作量.

Xue Jinyun 等[3]给出了一种形式化方法的新定义:形式化方法是一种严格的基于数学和工具支持的软硬件系统技术,包括高级抽象规范、建模语言和不同层次的模型转换工具. 规范和建模语言应尽可能抽象,并根据软件和硬件的特点反映不同的抽象级别. 所有的模型和模型转换工具都应该进行形式化地或自动化地验证.

模型驱动的软件开发是软件开发的一种新范型,它包括建模语言和模型转换方法及其支持工具. 在该方法中,模型(而不是程序)成为开发过程中的主要输出,在硬件/软件平台上执行的程序会由模型自动生成. 它提高了软件工程的抽象水平,使得软件工程师不必再去关心编程语言的细节或执行平台的特性[4]等.

本文将形式化方法与模型驱动方法相结合,采用模型驱动的方法来形式化生成 Dafny 程序并实现自动验证,充分利用了这 2 种方法的优点. 本文的模型主要有形式规约语言及算法建模语言 Radl[5],以及用于验证的 Dafny 语言. 模型驱动的 Dafny 程序形式化生成与自动验证方法步骤如下:首先,在程序开发的前期采用规约变换技术从问题 Radl 规约出发得到问题的 Radl 算法;然后,利用在 PAR 方法[5]中循环不变式开发的新策略得到循环不变式[6-7];最后,以 Radl 算法和循环不变式为依据,根据模型

收稿日期:2019-10-17
基金项目:国家自然科学基金(61762049,11804133,61862033)和江西省科技厅课题(2020BABL202025,2020BABL202026,2018BAB206034)资助项目.
作者简介:王昌晶(1977-),男,江西南昌人,教授,博士,博士生导师,主要从事泛型程序设计和可信软件的研究. E-mail:wcj771006@163.com
通信作者:左正康(1980-),男,江西抚州人,教授,博士,主要从事泛型程序设计和可信软件的研究. E-mail:kerrykaren@126.com

等价转换规则得出问题的 Dafny 程序,并由 Dafny 证明器自动验证该程序功能的正确性.

1　相关技术概述

1.1　Dijkstra 最弱前置谓词方法

Dijkstra 最弱前置谓词方法定义了一个非确定性的小型语言,包括 5 种基本语句以及过程调用语句. 使用 S 表示语句,R 表示谓词公式或 S 的后置断言,由 S 和 R 定义另一个谓词 $WP(S,R)$,这是一个状态集合. 若 S 从集合中任意状态开始执行并在有限时间内终止,且终止状态满足后置断言 R,则称 $WP(S,R)$ 为最弱前置谓词[8],其循环语句的形式为

do ▯ $C_1 \rightarrow S_1$;
　　▯ $C_2 \rightarrow S_2$;
　　…
　　▯ $C_n \rightarrow S_n$;
od

其中 $C_1, C_2, \cdots, C_n$ 和 $S_1, S_2, \cdots, S_n$ 分别为布尔表达式和任意语句,而 $C_i \rightarrow S_i (i = 1,2,\cdots,n)$ 是条件子句.

最弱前置谓词方法要证明形如$\{Q\}$ do$\{R\}$ 的循环语句的正确性,必须找到其循环不变式 ρ 和界函数 τ,并使它们满足条件:

(i) $Q \Rightarrow \rho$;
(ii) $\rho \wedge C_i \Rightarrow WP("S_i", \rho)$　$1 \leqslant i \leqslant n$;
(iii) $\rho \wedge \neg\ Guard \Rightarrow R$;

以证明循环语句的正确性. 若要想证明完全正确,则要添加 2 个条件:

(i) $\rho \wedge Guard \Rightarrow \tau > 0$;
(ii) $\rho \wedge C_i \Rightarrow WP("\tau_1 := \tau; S_i"\ \tau < \tau_1)$.

本文所用 Dafny 程序证明器证明程序正确性的内在机理是最弱前置谓词.

1.2　Dafny 简介及其验证机理

Dafny 是一种编程语言和静态程序证明器. Dafny 语言结合了函数式编程和基于对象编程的编程范式,它是命令式的、顺序的,支持泛型类和动态分配,并且内置规范结构(如 Eiffel、JML 和 Spec#),Dafny 编程语言旨在支持程序的静态验证. 规范(specifications)包括标准的前后置条件、框架结构和终止度量. 为了在规范中提供验证帮助,该语言包括用户定义的数学函数和影子(ghost)变量. 这些特性允许对指定程序进行模块化验证(modular verification),因此程序的每个部分的单独验证意味着整个程序的正确性. 最后,除了 class 类型之外,该语言还支持集合、序列和代数数据类型等.

Dafny 使用的语法类似于其他编程语言,同时它可以生成".net"可执行文件,但它需要一些特殊的工具来帮助进行有效地验证. 标注(annotations)是 Dafny 为用户提供的帮助验证程序的一个工具. 标注不是实际程序的一部分,而是提供有关程序中方法的规范和信息的语句; 通过使用标注,Dafny 能够更直接地通过考虑函数中的标注来验证程序的正确性. Dafny 将编写无 bug 代码的负担提升为编写无 bug 标注的负担,这比编写代码更容易,因为标注更短、更直接. 此外,编写标注的行为可以帮助人们理解代码在更深层次上所做的事情. Dafny 还可以证明在没有运行时的错误,例如索引超出界限、空取消、除以 0 等.

Dafny 程序的基本单元是方法(method),方法是一段具有方法头的可执行代码,其中声明了多个命名参数和多个命名结果. Dafny 还提供了用户定义的函数(function). 一般情况下,在 Dafny 中,函数只能在规范中使用以帮助验证,因此它们不生成代码. 若要覆盖此一般情况,以便编译器生成函数的代码,则须使用关键字函数方法(function method)来声明该函数. 同理,谓词(predicate)是一种特殊的返回布尔值的函数,而谓词方法可以生成代码. Dafny 为 assertion 内置了规范结构,如 requires 表示前置条件、ensures 表示后置条件等. 关键字的用法与含义如表 1 所示[9].

表 1　关键字含义

requires	boolean expression(preconditions)
ensures	boolean expression(postconditions)
assert	boolean expression
invariant	boolean expression (that holds before, during, and at the conclusion of a while loop)
decreases	ranking function
forall	counter:: range⇒boolean expression
reads	immutable object
modifies	mutable object

Dafny 代码在通过 Dafny 编译器之后,有 2 条路径: 一条是直接生成 C#代码,然后经过".net"编译器生成可执行文件; 另一条是给 Dafny 验证器,它的验证方式是将通过编译器的 Dafny 程序转换为中间验证语言 Boogie[10],Dafny 程序的正确性转化为 Boogie 程序的正确性. 然后 Boogie 工具生成传递到定理证明器的 1 阶谓词验证条件,特别是到 SMT 求解器 Z3[11]. Dafny 验证器作为编译器的一部分运行. 因此,程序员与它的交互方式与静态类型检查器的方式一样,当工具报告错误时,程序员通过更改程

序的类型声明、规范和语句来响应. 具体验证机理如图 1 所示.

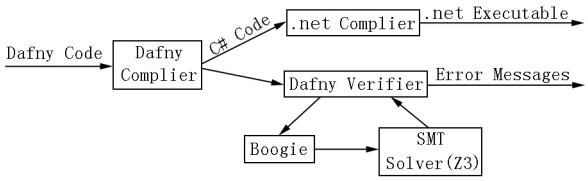

图 1 Dafny 验证机理图

1.3 PAR 方法概述

PAR(Partition-and-Recur)方法是通过寻找问题求解序列的递推关系来开发算法程序,它由薛锦云教授长期对算法设计方法的研究而提出的一种实用的程序算法设计和证明的方法. 程序正确性证明理论如 Hoare 逻辑和 Dijkstra 都是 PAR 方法的理论基础[12],同时 PAR 方法也包含基于分划、递推、扩充的量词变换规则和循环不变式的新技术[13]等.

关于循环不变式的开发,本文所用的是 PAR 方法中关于未知算法所提出的循环不变式开发策略. 主要内容有先分析问题背景和相关数学性质,描述后置断言并利用推导规则推导出递推关系,然后在递推关系的基础上得到循环语句,最后分析循环语句的特征得到循环不变式.

Radl 语言为实现算法程序形式化和半自动开发而定义的算法设计语言,它是 PAR 方法的重要组成部分. 由于它面向用户的特点,所以比用自然语言、程序框图和程序更简单、精确和易于理解地描述算法. 同时它也保留了传统的数学习惯以及具有引用透明性,对算法的形式推导非常方便. 描述问题的规约、规约变换规则和算法是其主要功能[14].

Radl 语言的其他优点包括丰富的数据类型,主要有[14]:标准数据类型、自定义简单类型、预定义 ADT、自定义 ADT 和泛型 ADT 机制. Radl 算法的结构为

ALGORITHM:〈algorithm name〉

SPECIFICATION:〈algorithm specification〉

BEGIN:〈initialization of variables and function in the recurrence〉

TERMINATION:〈termination condition of the recurrence〉

A_I:〈algorithm invariant〉

RECUR:〈set of recurrences〉END,

其中 SPECIFICATION:〈algorithm specification〉具体定义为

〈|[identifier description]||〉

AQ:〈predict expression〉

AR:〈predict expression〉.

Radl 表达式具有引用透明性,使得算法的形式化推导成为可能. 该算法的独特优点是易于理解和证明算法的巧妙性和正确性.

2 模型驱动的 Dafny 程序生成

模型驱动的 Dafny 程序形式化生成主要包括以下 3 个步骤:

(i) 从问题的 Radl 规约出发,根据规约变换技术进行推导,得到其递推关系 $S_i = F(S_j)$ 以及 Radl 算法;

(ii) 根据在 PAR 方法中关于未知算法的循环不变式的开发策略,定义变量保存递推关系中循环函数的值并写出循环控制变量的范围,构成循环不变式;

(iii) 将循环不变式加入 Radl 算法中,然后根据模型等价转换规则将完整 Radl 算法转化为 Dafny 程序,并用 Dafny 证明器自动验证该程序. 具体的模型等价转换规则为

(a) ALGORITHM 部分: Radl 算法中的〈algorithm name〉转换成 Dafny 中方法名;若有返回值,则需要在 Dafny 方法名中给出返回值.

(b) SPECIFICATION 部分:〈algorithm specification〉中〈identifier description〉部分转换为 Dafny 程序中方法 method 的参数. 若标识符代表的对象被改变了,则在方法中添加标注 modifies,否则不加. 这是因为方法 method 可以读取任何对象不需加标注 reads,而辅助验证函数 function 或谓词 predicate 则需加标注 reads;

Radl 算法〈algorithm specification〉中的 AQ 和 AR 分别转换成 Dafny 的前置条件 requires 和后置条件 ensures,其中后置条件中的 ∀、∃ 需要分别转换成 Dafny 中量词 forall、exist,并加上变量范围与函数部分.

AQ 部分若为 true,则转换成 Dafny 的 requires 可以省略. AR 部分定义的辅助函数需要转换为 Dafny 的函数方法 function method 或谓词方法 predicate method.

(c) BEGIN 部分: Radl 算法中的〈initialization of variables and function in the recurrence〉转换为 Dafny 程序中循环的初始条件.

(d) TERMINATION 部分:〈termination condition of the recurrence〉主要有 2 个作用:一方面是转换为 Dafny 程序中的循环终止条件,在转换成程序时应

将其取非; 另一方面转换成 Dafny 中 decreases 语句,用来证明程序终止(完全正确性). 一般 Dafny 能自动给出循环或递归的终止条件,但有时仍需要显示指出.

(e) RECUR 部分: 〈set of recurrences〉转换为 Dafny 程序中的循环体. 具体转换方式如下: 若递推函数是 1 元函数,则真正起数组变量作用的变量使用一个普通变量来换名,而仅起递推作用的变量却直接退化为普通变量; 若递推函数是 2 元以上函数,则每个递推函数使用一个普通变量来换名.

(f) A_I 部分: Radl 算法中的〈algorithm invariant〉转换为 Dafny 程序中的 invariant. 通过分析〈algorithm invariant〉,一般是多个合取项,主要包括变量范围和递推函数 2 个部分. 变量范围部分直接将其转换为 Dafny 循环中的 invariant 语句. 递推函数部分按 2 种情况讨论: 若递推函数是 1 元函数,则通过给出各个递推函数的定义,直接转换为 Dafny 循环中 invariant 语句,真正起数组变量作用的递推函数,还需弱化递推函数的定义,并使用一个普通变量来换名; 若递推函数是 2 元以上函数,则通过弱化各个递推函数的定义,直接转换为 Dafny 循环中 invariant 语句,每个递推函数使用一个普通变量来换名.

3　案例

应用上述方法对 2 个典型问题进行形式化生成与自动验证: 一个是计算数组元素的立方和; 另一个是计算数组相邻元素的最小和.

3.1　立方和问题

问题描述: 给定空数组,只用加法使其满足 $(\forall i: 0 \leqslant i < n: a(i) = i^3)$,$n$ 为一个给定的整数.

利用本文方法对该问题进行循环不变式的推导和 Dafny 程序的生成,步骤如下:

(i) 从问题的 Radl 规约出发根据规约变换技术推导得到其 Radl 算法,其归纳为

{AQ: true},

$\{AR: (\forall i: 0 \leqslant i < n: a(i) = i^3)\}$,

再分化原问题和构造问题的递推关系及 Radl 算法.

由后置断言 $R: (\forall i: 0 \leqslant i < n: a(i) = i^3)$ 可以得出 $a(i+1) = (i+1)^3 = i^3 + 3i^2 + i + 1$. 设 $a(i) = i^3$,$a(i+1) = a(i) + 3i^2 + i + 1$,为了计算 $3i^2$,设 $b(i) = 3i^2$,则 $b(i+1) = 3(i+1)^2 = b(i) + 6i + 3$; 为了计算 $6i$,设 $c(i) = 6i$,则 $c(i+1) = 6i + 6 = c(i) + 6$; 为计算 $3i$,设 $d(i) = 3i$,$d(i+1) = d(i) + 3$,分别计算得出 $a(i+1) = a(i) + 3i^2 + 3i + 1$,$b(i+1) = b(i) + c(i) + 3$,$c(i+1) = c(i) + 6$,$d(i+1) = d(i) + 3$. 最后得 $a(i+1) = a(i) + b(i) + d(i) + 1$.

结合上面的式子,并给其中递推函数赋初值,得到解该问题的 Radl 算法为

```
ALGORITHML: Cubes( )
SPECIFICATION:
    | [i: integer; a,b,c,d: array(0: n,integer)] |
    {AQ ∧ AR}
BEGIN: i: =0 + +1; a(i): =0; b(i): =0; c(i): =0; d(i): =0;
TERMINATION: i = n
A _I: ρ
RRCUR:
    //引用透明性
    a(i+1) = a(i) + b(i) + d(i) + 1;
    b(i+1) = b(i) + c(i) + 3;
    c(i+1) = c(i) + 6;
    d(i+1) = d(i) + 3;
END.
```

(ii) 基于循环不变式开发策略,定义变量存放递推函数的值. 事实上,算法中数组 $a(0: n-1)$ 必须用来存放 i^3 的值,数组 b、c、d 可以用简单变量 b、c、d 来替代,于是得到循环不变式为

$\rho: 0 \leqslant i \leqslant n \wedge a(i) = a(i-1) + b(i-1) + d(i-1) + 1 \wedge b = b(i-1) + c(i-1) + 3 \wedge c = c(i-1) + 6 \wedge d = d(i-1) + 3$.

根据 A_I 转换规则,可以定义变量 s 表示数组 a,并将所有递推函数表示成循环变量 i 的形式得到的循环不变式为

$\rho_i: 0 \leqslant i \leqslant n \wedge s = i^* \ i^* \ i \wedge b = 3^* \ i^* \ i \wedge c = 6^* \ i \wedge d = 3^* \ i$,

并将其加入 Radl 算法中.

(iii) 结合上面抽象 Radl 算法与循环不变式,利用模型等价转换规则,很容易将其转换成如下 Dafny 程序:

```
method Cubes(a: array〈int〉)
modifies a //数组 a 是变化的
ensures forall i: : 0 ≤ i < a. Length ⇒
                    a[i] = = i* i* i //后置断言 AR
{ //给函数赋初值
    var i: =0;
    var s: =0; //a[i]
    var b,c,d: =0,0,0;
    while i < a. Length
        invariant 0 ≤ i ≤ a. Length
```

```
        invariant forall k ::
            0≤k<i⇒a[k] ==k* k* k
        invariant s ==i* i* i
        invariant b ==3* i* i
        invariant c ==6* i
        invariant d ==3* i
        decreases a.Length - i
    {
        a[i]:=s;
        s:=s+b+d+1;
        b:=b+c+3;
        c:=c+6;
        d:=d+3;
        i:=i+1;
    }
}.
```

然后将得到的 Dafny **程序用** Dafny **二进制代码验证或者在线验证,全部验证通过.验证通过结果如图** 2 **所示.**

```
Dafny 2.3.0.10506
Dafny program verifier finished with 3 verified,0 errors
Running…
Before:0,0,0,0,0,0,0
After:0,1,8,27,64,125,216
```

图 2 立方和 Dafny 自动验证图

3.2 最小和问题

问题描述:计算给定整型数组 $a[0:n-1]$ **中相邻元素的最小和.**

(i) **从问题的** Radl **规约出发根据规约变换技术推导得到其** Radl **算法.**

AQ:**给定非空数组** $a[0:n-1](n>0)$,

AR: $\text{minsum}(n-1)=(\text{MIN } i,j: 0\leqslant i\leqslant j<n: \text{sum}(i,j))$.

定义辅助函数 $\text{sum}(i,j)$:

$\text{sum}(i,j)=(\sum k: i\leqslant k\leqslant j: a[k])=\text{sum}(i,j-1)+a[j-1]$.

当 $i\leqslant j$ **时,定义辅助函数** min **如下:**

$\min(a,b)=\text{if } a>b \text{ then } b \text{ else } a$,

分化原问题为结构相同的子问题,构造问题求解的递推关系和初步 Radl **算法,即** $minsum(a[0:n-1])=F(minsum(a[0:m-2],a[m-1]))$,$0\leqslant m\leqslant n$. **从后置断言出发寻找递推关系** F:

$minsum(a[0:m])=(\text{MIN } i,j: 0\leqslant i\leqslant j\leqslant m: sum(i,j))=(\text{MIN } j: 0\leqslant j\leqslant m: (\text{MIN } i: 0\leqslant i\leqslant j: sum(i,j)))=(\text{MIN } j: 0\leqslant j\leqslant m: ms(j))$,

定义 $ms(j)=(\text{MIN } i: 0\leqslant i\leqslant j: sum(i,j))=\min((\text{MIN } j: 0\leqslant j\leqslant m: ms(j)),ms(m))=\min(minsum(a[0:m-1]),ms(m))$.

第 1 **个递推关系式为** $minsum(a[0:m])=\min(minsum(a[0:m-1]),ms(m))$,$0\leqslant m\leqslant n$. **因为递推式** 1 **中含有** $ms(m)$ **的函数,所以需要找到** $ms(m)$ **和** $ms(m-1)$ **的关系:**

$ms(m)=(\text{MIN } i: 0\leqslant i\leqslant m: sum(i,m))=(\text{MIN } i: 0\leqslant i\leqslant m: (\sum k: i\leqslant k\leqslant m: a[k]))=(\text{MIN } i: 0\leqslant i\leqslant m: (\sum k: i\leqslant k\leqslant m-1: a[k]))+a[m])=(\text{MIN } i: 0\leqslant i\leqslant m: sum(i,m-1))+a[m]=\min((\text{MIN } i: 0\leqslant i<m: sum(i,m-1)),sum(m,m-1))+a[m]=\min(ms(m-1),0)+a[m]=\min(ms(m-1)+a[m],a[m])$,

得到第 2 **个递推关系式为** $ms(m)=\min(ms(m-1)+a[m],a[m])$. **当** $m=0$ **时,有** $ms(m-1)=0$,$minsum(a[0:m-1])=0$,**因此有** $m=0\wedge ms(m-1)=0\wedge minsum(a[0:n-1])=0$

将初始化、递推关系 1 **和递推关系** 2 **结合在一起,得到一个时间复杂度** $O(n)$ **的算法如下:**

```
ALGORITHML: minsum(a[0:n-1])
SPECIFICATION:
    |[a: |array(0:n-1,integer); m,minsum,
    |ms: integer]|
    {AQ∧AR}
BEGIN: m=0++1,minsum(a[0:m-1])=0,
        ms(m-1)=0;
TERMINATION: m=n
RRCUR:
    ms(m):=min(ms(m-1)+a[m],
            a[m]);
    minsum(a[0:m]):=min(minsum(a[0:
            m-1]),ms(m));
END.
```

(ii) **根据** PAR **方法中循环不变式开发方法,把递推函数的值保存在变量中;**$minsum(a[0:m-1])$ **和** $ms(m)$ **的值分别用** s **和** c **保存.结合约束变量范围可得不变式为** $\rho: 0\leqslant m\leqslant n\wedge s=minsum(a[0:m-1])\wedge c=ms(m)$,**即** Radl **算法中的** A_I,**并将其补全到** Radl **算法中,得到完整的** Radl **算法.**

(iii) **根据** Radl **算法和循环不变式,依照等价转换规则将它们转换成** Dafny **程序并验证,验证通过**

结果如图 3 所示,其中函数 *min* 和 *sum* 易转换成对应的 Dafny 函数:

```
function method sum( a: array? 〈int〉, i: int, j: int) : int
reads a
requires a! = null
requires 0≤i≤j≤a. Length
decreases j - i
{
if (j = = i) then 0 else sum( a,i,j - 1) + a [j - 1]
}
function method min( a: int,b: int) : int
{ if a > b then b else a}
method minsum( a: array? 〈int〉) returns( s: int)
requires a! = null
ensures forall i,j: : 0≤i <j≤a. Length - 1⇒sum( a, i,j) 〉= s
{
var m: = 0; s: = 0; var c: = 0;
//m 是循环控制变量
//s 和 c 分别代表 2 个递推函数
while( m < a. Length)
invariant 0≤m≤a. Length
invariant forall i,j: : 0≤i <j≤m⇒sum( a,i,j) ≥s
invariant forall i: : 0≤i < m⇒sum( a,i,m) ≥c
decreases a. Length - m
{
    c: = min( c + a [m] ,a [m]) ;
    s: = min( s,c) ;
    m: = m + 1;
    }
}.
```

```
Dafny 2. 3. 0. 1056
Dafny program verifier finished with 4 verified,0 errors
Running …
minsum: - 13
```

图 3　最小和 Dafny 自动验证图

4　相关工作比较

算法程序的可靠性对软件系统至关重要,高可靠性算法的开发与验证一直是人们研究的热点. 用程序证明器(如 Dafny、Why3 [14]) 对算法程序进行自动化验证比交互式定理证明器(如 Isabelle、Coq) 更加高效、快捷,并且自动化程度更高,这是将来的发展趋势. 以 Isabelle [15] 为例,目前 Isabelle 对线性算法、文件比较算法以及 Linux0. 11 内核调度模块等进行了形式化验证,它所实现的是人机交互的半自动化且需要用户的大量手工劳动来构造定理的证明,这个实现过程效率相对较低.

5　总结与展望

软件开发的形式化方法和模型驱动的方法在软件开发与验证中各有优缺点. 本文将形式化方法与模型驱动方法 [16] 相结合,采用模型驱动的方法来形式化生成 Dafny 程序并实现自动验证,充分利用了这 2 种方法的优点. 形式化方法因为严格基于数学,可以进行形式验证与推导,从而保证了软件开发的可靠性; 模型驱动的方法包含建模语言和不同层次的模型转换工具,这提高了软件的生成效率 [4] . 本文将这 2 种方法结合起来,相互补充,有效地提高了 Dafny 程序的生成效率和可靠性.

下一步的研究工作主要有: (i) 使用该方法对更多复杂算法(如排序、搜索) 进行形式化生成与自动验证,不断完善该方法; (ii) 将该方法应用在 Web 服务及其组合上 [17-18] .

6　参考文献

[1] 王戟,詹乃军,冯新宇,等. 形式化方法概貌 [J] . 软件学报,2019,30(1) : 33-61.

[2] Rustan K, Leino M. Accessible software verification with Dafny [J] . IEEE Software,2017,34(6) : 94-97.

[3] Xue Jinyun, Zheng Yujun, Hu Qimin, et al. PAR: a practicable formal method and its supporting platform [EB/OL] . [2019-06-16] . https: //link. springer. com/chapter/10. 1007/978 - 3 - 030 - 02450 - 5_5.

[4] Ke Wei, Li Xiaoshan, Liu Zhiming, et al. rCOS: a formal model-driven engineering method for, component-based software [J] . The frontier of Chinese Computer Science: English Version,2012,6(1) : 17-39.

[5] 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.

[6] 杨黄磊,薛锦云. 一类单元赋值语句型循环不变式的开发方法研究 [J] . 江西师范大学学报: 自然科学版,2014,38(4) : 378-382.

[7] 万松松,薛锦云,谢武平. 循环不变式开发技术研究 [J] . 计算机工程与科学,2010,32(9) : 84-88.

[8] 游珍. Isabelle 定理证明器的剖析及其在 PAR 方法/PAR 平台中的应用 [D]. 南昌:江西师范大学,2009.
[9] Leino K R M. Dafny: an automatic program verifier for functional correctness [EB/OL]. [2019-06-16]. https://dl.acm.org/doi/10.5555/1939141.1939161.
[10] Barnett M, Chang B Y E, Deline R, et al. Boogie: a modular reusable verifier for object-oriented programs [EB/OL]. [2019-06-16]. https://link.springer.com/chapter/10.1007%2F11804192_17.
[11] Moura L D, Nikolaj Bjørner. Z3: an efficient SMT solver [J]. Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, 2008, 4963: 337-340.
[12] 冉小晓. Radl→Apla 自动程序转换系统研究与实现 [D]. 南昌:江西师范大学,2005.
[13] 杨晨. 基于 PAR 平台的最弱前置谓词生成器的设计与实现 [D]. 南昌:江西师范大学,2010.
[14] Jean-Christophe Filliâtre, Paskevich A. Why3: where programs meet provers [J]. Lecture Notes in Computer Science, 2013, 7792: 125-128.
[15] 游珍,薛锦云. 基于 Isabelle 定理证明器算法程序的形式化验证 [J]. 计算机工程与科学,2009,31(10):85-89.
[16] 李智,庞柳,刘国源,等. 一种模型驱动的软件需求分析方法及技术支持 [J]. 广西师范大学学报:自然科学版,2013(2):22-29.
[17] Saleh I, Kulczycki G, Blake M B. Formal specification and verification of data-centric service composition [C]. Miami, Florida: IEEE, 2010: 131-138.
[18] Iman Saleh. Formalizing data-gentric web services [M]. New York: Springer, 2015: 1-126.

The Model-Driven Dafny Program Generation and Automatic Verification

WANG Changjing[1], HE Jiangfei[1], LUO Haimei[2,3], ZUO Zhengkang[1*], XU Fan[1]
(1. College of Computer Information Engineering, Jiangxi Normal University, Nanchang Jiangxi 330022, China;
2. College of Physics and Communication Electronics, Jiangxi Normal University, Nanchang Jiangxi 330022, China;
3. Key Laboratory of Photoelectronics and Telecommunication of Jiangxi Province, Jiangxi Normal University, Nanchang Jiangxi 330022, China)

Abstract: Dafny is a programming language with built-in specification structure and a static program prover. It can verify the program's functional correctness and automate the certification process. This not only improves the efficiency of software development, but also greatly enhances the reliability of software development. A model-driven method for the formal generation of Dafny programs is explored in the paper. Firstly, the Radl algorithm is derived from the Radl specification of the problem according to the protocol transformation technology. Then, a new strategy is developed based on the loop invariant of the PAR method to obtain the loop invariance of the problem. Finally, based on Radl algorithm and loop invariant, Dafny program is generated using model equivalent transformation rules, and its functional correctness is automatically verified by Dafny prover. The method proposed in this paper solves two typical problems of algorithm program development and verification, and it is proved that this method can effectively improve the generation efficiency and reliability of Dafny programs.

Key words: model driven; Dafny program; loop invariant; formal generation; automatic verification

(责任编辑:冉小晓)