计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2022,31(1):327−331 [doi: 10.15888/j.cnki.csa.008275]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

终止证明方法在形式化建模中的应用①

任　凭[1], 张　杰[1], 关　永[2]

[1](北京化工大学 信息科学与技术学院, 北京 100029)
[2](首都师范大学 信息工程学院, 北京 100048)
通信作者: 张　杰, E-mail: jzhang@mail.buct.edu.cn

摘　要: 随着形式化方法的普及和应用, 定理证明器 HOL4 在形式化建模过程中无法自动完成终止证明的情况越来越多, 而手动终止证明又缺少通用的证明思路. 针对这种情况, 提出规范化的手动终止证明方法. 该方法从问题产生的本质入手, 首先保证目标具备解决终止问题的必要条件, 然后通过等效替换简化证明目标, 最后以原有定理库为基础, 寻找证明过程中缺失的引理, 推进证明. 实例表明, 该方法逻辑清晰, 能够有效地解决 HOL4 中大部分情况下的手动终止证明问题.

关键词: 形式化方法; HOL4; 终止证明

引用格式: 任凭,张杰,关永.终止证明方法在形式化建模中的应用.计算机系统应用,2022,31(1):327–331. http://www.c-s-a.org.cn/1003-3254/8275.html

Application of Termination Proof Method in Formal Modeling

REN Ping[1], ZHANG Jie[1], GUAN Yong[2]

[1](College of Information Science and Technology, Beijing University of Chemical Technology, Beijing 100029, China)
[2](Information Engineering College, Capital Normal University, Beijing 100048, China)

Abstract: With the popularization and application of formal methods, there are increasingly more cases in which the theorem prover HOL4 cannot automatically complete the termination proof in the process of formal modeling. Manual termination proof still lacks a general idea. In response, a standardized manual termination proof method is proposed. Starting from the nature of the problem, the method guarantees that the target has the necessary conditions for solving the termination problem. Then, the proof target is simplified by equivalent substitution. Finally, on the basis of the original theorem library, the lacking lemma in the proof process is found to advance the proof. The example shows that this method has a clear logic and can solve the manual termination proof problem of the HOL4 in most cases.

Key words: formal method; HOL4; termination proof

形式化方法是通过数学建模和数学归纳等方法去验证目标的性质规范以及确保目标的正确性和可靠性的一条重要途径, 其广泛应用于计算机软硬件的规范, 开发和验证.

形式化验证一般分为等价性检验, 模型检验和定理证明 3 种, 并且验证方式各不相同. 等价性检验的方法主要是通过检验目标修改前后的一致性, 来确保修改后的目标至少包含修改前的性质, 但是不能进行特定性质或规范的验证. 模型检验的特点是完全自动化, 但是存在状态空间爆炸的问题, 不能用于验证无限状态空间的目标. 定理证明的形式化方法是将待验证的目标和规范通过数学建模, 描述为一系列的逻辑语言, 并通过逻辑演算的方式, 证明目标是否具有规范所描述的性质. 定理证明的验证方法可以对目标的所有状态空间进行建模和验证, 与传统的仿真测试相比, 测试空间更加完备, 其验证结果具有相当高的可靠性.

① 基金项目: 国家自然科学基金 (61876111)
收稿时间: 2021-04-02; 修改时间: 2021-04-29; 采用时间: 2021-05-07; csa 在线出版时间: 2021-12-17

当前用于定理证明方法的主流工具有半自动的一阶定理证明器 Proverif, 高阶定理证明器 HOL4, Isabelle 等; 也有全自动定理证明器 SmartVerif. 它们各有优劣, 半自动工具功能强大, 拓展性高, 但是学习门槛高; 全自动工具入门容易但功能略逊一筹, 实用性不如半自动定理证明器. 本文用的是基于 Standard ML 函数式语言的半自动高阶定理证明器 HOL4.

近年来, 随着形式化方法和定理证明器系统被广泛应用于数学定理[1], 计算机软件[2], 硬件[3], 方法[4,5] 以及协议[6] 的验证和证明, 系统自带的数据结构和定理库已不能满足所有形式化工作的需要; 因此, 用户在进行形式化建模时, 常常需要自定义数据结构, 特别是在 HOL4 中定义递归函数时用户必须考虑函数的终止问题[7], 这就给递归函数的形式化建模带来困难.

1 问题的提出

介绍终止证明问题之前首先需要了解什么是递归函数. 对于任意函数, 定义函数的语句描述了输入与输出之间的逻辑关系, 当输入任意符合要求的 x, 都可以得到其对应的唯一输出. 一般非递归函数 $f(x)$ 在定义时可以调用外部函数, 但不会调用函数自身, 如图 1 所示.

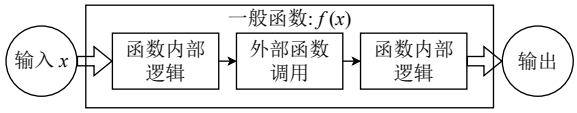

图 1 一般非递归函数 $f(x)$

而与之不同的是, 递归是通过调用自身的代码来解决问题, 递归方法是计算机科学的核心思想之一[8]. 在使用递归函数进行建模的过程中, 如果考虑不周, 使得在某些输入的情况下, 出现无限递归的情况, 如图 2 所示. 在形式化建模中不允许递归函数无限调用自身无法得到输出的情况存在. 因此, 建立满足终止证明规范的递归函数模型是完成递归函数形式化证明的必要条件.

为了使递归函数的形式化模型能够满足规范, 需要在递归调用之前添加一个判断语句, 如图 3 所示, 使得输入参数在满足某些特定条件下不再进行递归, 而是得到相应的输出; 而结束递归的判断条件被称为递归出口或终止条件.

并且, 在形式化建模过程中定义递归函数, 与使用非形式化语言定义函数的过程有很大的区别: 在非形式化的定义当中, 一个没有终止条件或终止条件不完备的递归函数也能被成功定义, 直到它在实际调用中发生错误时才会抛出错误; 而由于形式化方法的严谨性, 在其定义递归函数时, 首先必须证明它满足“输入集中的任意元素在输出集中都存在唯一对应”的规范, 该证明过程被称为终止性证明, 若无法完成终止证明, 就无法成功定义且使用该函数. 所以, 在 HOL4 中定义递归函数的关键是要确定其终止条件. 在终止条件不够全面的情况下, 终止证明必然失败, 同时证明过程将暴露终止条件中的问题, 帮助用户发现和补全终止条件的漏洞. 然而, 即使保证了终止条件的完备性, 为什么定理证明器仍然无法自动完成所有递归函数的终止证明?

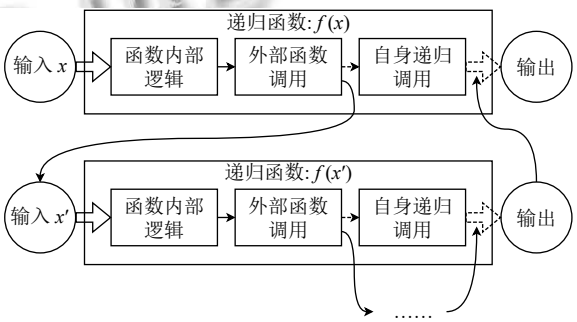

图 2 缺少终止条件的递归函数 $f(x)$

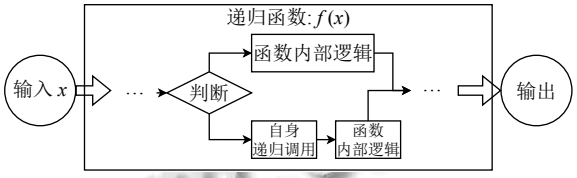

图 3 有终止条件的递归函数

在一些定理证明器的定理库中, 存在列表等基础的数据结构和相关的性质定理, 当使用这些结构进行递归定义时, 定理证明器基本能够自动的完成终止证明. 而在 HOL4 定理证明器中, 存在集合结构和相关的定理库, 但仍然无法完成一部分集合递归函数的终止证明, 这是因为 HOL4 的集合定理库中缺少一些集合递归函数终止证明所需的定理, 这些所需定理可以用已存在的定理推导得到, 但是半自动定理证明器在自动进行终止证明的过程中, 不能将定理库中定理进行组合, 推导得到新的定理后用于终止证明. 所以当系统无法自动完成终止证明时, 意味着系统的定理库中缺少关键的定理或引理, 需要用户自行推导, 并将其用于证明终止目标.

当定理证明器无法自动完成终止证明时, 如何手动辅助定理证明器完成终止证明是建模过程中的一个

难题. 本文主要介绍一种解决方案, 能够用于形式化建模过程中定义递归函数所产生的终止证明问题.

2 解决方案

终止证明问题的本质是通过定理证明的形式化方法证明新定义的递归函数满足“输入集中的任意元素在输出集中都存在唯一对应”的规范. 换而言之, 递归函数在任意输入参数时, 经过有限次递归后, 终将满足终止条件, 得到相应的输出, 所以该问题着重关注于输入参数在递归过程是否具有不断接近终止条件的趋势.

在基于 HOL4 的形式化建模工作的实践过程中, 本文总结出定义两大类定理证明器经常无法自动完成终止证明的递归函数:

1) 使用自定义数据结构作为参数时. 在工作过程中验证新目标时, 往往需要用到新的数据结构, 如链结构, 图结构等等. 但是在系统的定理库中, 只有少量自定义数据结构的衍生谓词, 性质定理和函数, 故使用自定义的数据结构定义递归函数时, 针对不同的递归函数, 需要构建和证明不同的功能函数和性质定理, 辅助系统完成终止证明.

2) 使用自定义的函数作为参数时.

$$f(x) = g(x) * f(h(x)) \tag{1}$$

当进行形如式 (1) 的递归函数定义时 (其中 $h(x)$ 也是自定义的函数). 无论参数 x 的数据结构是否是自定义结构, 都需要手动辅助其完成终止证明.

如上所述, 具有明确的终止条件是解决终止证明问题的前提, 确定待定义的递归函数的终止条件是解决问题的第一步. 充分考虑输入参数的可能性后, 一个完整的终止条件通常由复数通过析取连接的子条件组成. 这些子条件在输入集合的基础上分割出若干个非空真子集, 若输入参数属于这些子条件描述的真子集中, 则将得到确定的输出, 否则继续递归. 将终止条件描述的集合称为输入集合的终止子集, 将终止子集的补集称为输入集合的递归子集.

假设目标递归函数设置了合适的终止条件, 即可通过系统中存在的函数或定义方法, 编译定义语句. 在不同的定理证明器中所使用的函数和方法存在区别, 但是目标和思路大体相同, 以下将以 HOL4 定理证明器中的函数和方法为例进行详细分解.

在 HOL4 定理证明器中, 使用 Hol_defn 方法编译递归函数的定义语句, 系统将剥离出终止条件和递归参数, 再使用 Defn.tgoal 方法即可得到终止证明目标. 终止证明目标将剔除与递归无关的定义语句, 关注于输入参数在递归过程中的变化趋势.

系统通过证明输入参数在递归过程中存在某种指标一步步的由递归子集趋近于终止子集来说明函数必定终止, 用户需要辅助系统的工作首先是明确该指标. 例如在递归函数实现的列表遍历方法中, 若终止条件为输入参数等于空列表, 输入列表的长度在每次递归时递减. 因此列表的长度属性是指标, 通过函数量化列表的长度, 并且将终止条件描述的空列表的长度设定为固定值 x, 在递归的过程中列表长度量化值不断的向 x 趋近, 即可证明在有限次递归后, 输入参数的列表长度必将等于 x, 从而触发终止条件, 得到确定的输出.

在 HOL4 定理证明器的终止证明中, 需要将以上的趋近关系改写成值域为自然数的单调递减函数, 终止证明的目标变化为证明该函数的单调递减性. 在一些复杂的递归函数中, 如图的深度优先和广度优先遍历算法中, 同时对图的深度和广度进行递归, 需要选取复数个指标进行量化, 并且构建合适的单调递减函数.

将终止证明目标转化为单调递减性证明后, 根据具体目标不同, 证明方法也不同. 对于定理证明器无法自动证明的子目标, 有可能是缺少一些引理, 或者是缺少相关自定义结构的性质定理. 应推理出详细的证明过程, 证明辅助定理并推进证明目标.

最后将终止证明过程中使用的方法和对策整理为策略, 使用定理证明器中的相关方法将定义语句与策略进行整合, 规范建模格式.

通过详细分析, 无论是使用自定义数据结构或定义嵌套递归函数的情况, 都可以按照以下方法来进行函数定义和终止证明.

算法 1. 终止证明方法

(1) 分析待定义函数的输入集, 明确终止条件.
(2) 分析递归过程, 确定输入参数中在递归过程中变化最明显的可量化属性, 定义量化函数, 并根据每个终止子条件设置固定的量化值.
(3) 使用定理证明器中手动进行终止证明的定义方法, 得到终止证明目标.
(4) 根据终止子条件的固定值和输入参数在递归过程中的变化趋势, 构建单调递减函数, 并将终止证明目标简化为所构建函数的单调递减性证明.
(5) 在证明过程中补充系统定理库缺少的定理和引理, 辅助系统推进证明. 如果在证明过程中遇到无法被证明的子目标, 回到第 (1) 步检查是否缺失终止子条件.
(6) 完成终止证明, 整理定义和证明过程, 规范建模格式.

3 应用实例

在实际应用中, 经常会需要用到图结构来表示节点与节点间的连接关系. 而 HOL4 定理库中暂时没有图的数据结构和相关性质定理, 所以需要用户自定义图结构. 因此, 下面将以自定义图结构作为递归参数, 在 HOL4 中使用该终止证明方法完成一个递归函数定义和终止证明. 在该应用中, 定义 graph 结构包含了存储图中节点信息的集合 nodes, 以及节点之间连接关系的集合 edges, 其中点集 nodes 与边集 edges 的论域都为全集, graph 的论域为点集 nodes 与边集 edges 的笛卡尔乘积.

有向图的定义约束了点集与边集之间的关系, 故有向图集是 graph 的真子集. 因此将有向图定义进行形式化, 得到谓词 Digraph 将输入集切分为有向图集与非有向图, 并以此作为以 graph 作为输入参数的有向图的递归函数的第一个终止子条件: 当输入参数不属于有向图集时将终止递归, 如图 4 所示.

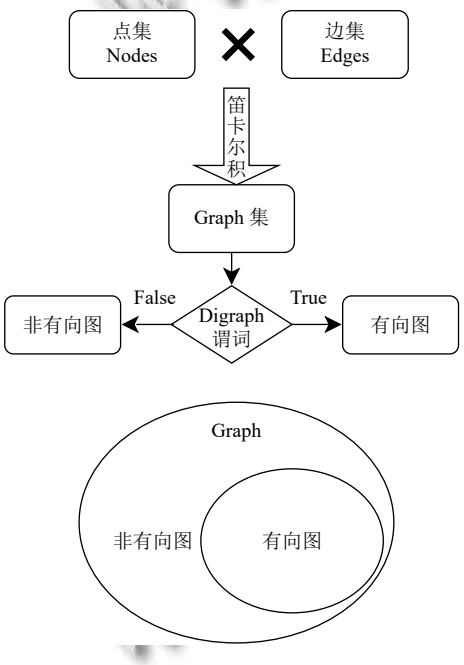

图 4 graph 类型与有向图关系示意图

此外, 在实现节点的信息统计等功能函数时, 需要遍历一个有向图中的所有节点. 在 HOL4 定理证明器使用的 Standard ML 函数式语言中, 没有循环语句, 取而代之的是使用递归的方法来实现循环和遍历的效果. 由于 HOL4 定理库中缺少自定义的 graph 结构的相关性质定理, 所以使用者必须手动辅助定理证明器完成终止证明. 下面将详细介绍如何使用以上给出的方法, 完成图节点遍历递归函数的终止证明.

3.1 实现过程

首先, 继续设置终止条件切分输入集. 待实现的递归函数功能为遍历图中所有节点, 若点集 nodes 为无限集, 则遍历行为无意义, 故得到第 2 个终止子条件: 当输入参数的点集 nodes 不属于有限集时将终止递归; 若输入为空图, 则不需要继续遍历, 将输出固定值, 因此得到第 3 个终止子条件: 当输入参数不属于非空图时将终止递归. 通过 3 个终止子条件, 将输入集切分为如图 5 所示的真子集.

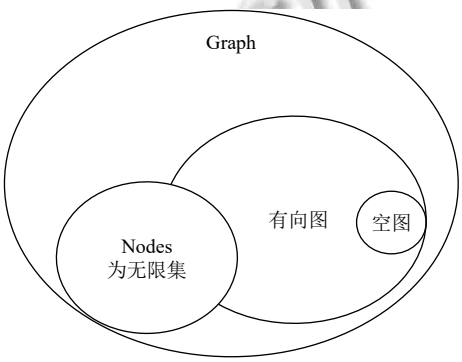

图 5 通过谓词切割有向图集

当输入参数不仅属于如图所示的有向图集合时, 将终止递归并输出.

终止条件的确定也明确了递归过程中, 输入参数在递归的过程中应不断趋向于终止条件. 在图节点遍历函数中, 可以将输入的图参数视作“已遍历”和“未遍历”两部分. 在递归过程中, “未遍历”的部分将持续减少, 在遍历完成时“未遍历”部分应当为空, 满足第 3 个终止子条件. 因此, 对于该递归函数的终止证明问题, 应当关注输入参数“未遍历”部分的节点规模, 并定义合适图规模量化函数, 设置空图的规模为 0.

在 HOL4 定理证明器中, 使用 Hol_defn 方法和 Defn.tgoal 方法可以得到终止证明目标, 为了简化目标中不必要的部分, 关注于证明递归参数量化指标的递减性, 使用 WF_REL_TAC 对策简化目标并重写展开自定义的函数后, 得到两个子目标.

子目标 1. 证明点集非空前提下目标的递减性

0. Digraph nodeG
1. FINITE nodeG.nodes
2. FINITE nodeG.edges
3. nodeG.nodes ≠ Ø

CARD {x | x ∈ nodeG.nodes ∧ x ≠ CHOICE nodeG.nodes} < CARD nodeG.nodes

查阅定理证明器中的集合定理库可知, 系统抛出该子目标无法继续的原因是库中缺少处理{x | x ∈ s ∧ x ≠ CHOICE s}的相关定理, 需要用户辅助证明. 因此, 利用定理证明器提供的集合定理库, 证明以下引理:

引理 1. 对于任意的集合 s, 由 s 中 CHOICE s 元素以外的元素组成的新集合, 等于 REST s 集合.

∀s. {x | x ∈ s ∧ x ≠ CHOICE s} = REST s

引理 2. 对于任意集合 s, 如果 s 为非空有限集, 则 CARD s 大于 0.

∀s. FINITE s ∧ s ≠ Ø ⇒ 0 < CARD s

借助引理即可完成子目标 1 的证明. 然后关注子目标 2.

子目标 2. 证明边集非空前提下目标的递减性

0. Digraph nodeG
1. FINITE nodeG.nodes
2. FINITE nodeG.edges
3. nodeG.edges ≠ Ø

CARD {x | x ∈ nodeG.nodes ∧ x ≠ CHOICE nodeG.nodes} < CARD nodeG.nodes

观察可知, 系统抛出此目标的原因不仅是缺少子目标 1 中的两个引理, 而且缺少图的相关性质定理: 当边集非空时, 点集必定非空. 该定理由图的定义与空图的定义推导而来.

有向图性质 1 边集非空的有向图点集必然非空.

∀G. Digraph G ∧ G.edges ≠ Ø ⇒ G.nodes ≠ Ø

借由引理 1 和引理 2, 图的性质定理和定理证明器的相关定理库, 即可完成该遍历函数的终止证明. 最后将对策整理为策略, 并使用 Defn.tprove 方法定义该函数.

在本实例中, 可见使用本文提出的终止证明方法解决终止证明问题, 思路非常清晰, 迅速确定终止条件, 补充必要的谓词和函数, 将终止证明目标简化为图的节点数量在递归过程中的递减性证明, 准确找到终止证明过程中缺少的两个集合引理和一个图的性质定理, 顺利完成终止目标的证明, 解决终止证明问题, 从而在 HOL4 中成功定义图的节点遍历递归函数.

4 结语

形式化建模工作中对递归函数的应用十分广泛而且重要. 由于定理证明器的定理库更新上的严谨和滞后, 在涉及到递归函数的建模过程中, 特别是使用到自定义的数据结构进行递归的建模中, 经常出现定理证明器无法自动完成终止证明的问题, 导致递归函数无法被成功定义和使用, 阻碍了形式化方法的应用和发展.

本文针对上述问题, 介绍了在形式化工作中为什么会遇到终止证明问题, 以及解决终止证明问题的必要性; 分析了定理证明器无法自动完成终止证明的原因; 设计了一种在形式化建模中通用的终止证明方法, 并将该方法应用于基于 HOL4 定理证明器的一个实例中. 在不同的定理证明器中, 使用的具体对策和函数有所不同, 但是方法的步骤和思路是相同的, 以可量化的指标作为终止条件的递归函数可以使用该方法进行定义并证明. 该方法很大程度方便了形式化建模的初学者在定理证明器中定义和使用自己的数据结构和递归函数, 使用形式化方法解决更具体和实际的问题.

$$f(x) = g(x) * f(f(x')) \quad (2)$$

同时, 该方法尚存不足, 无法解决式 (2) 形式的自嵌套递归函数的终止证明问题.

参考文献

1 Shi ZP, Gu WQ, Li XJ, *et al*. The gauge integral theory in HOL4. Journal of Applied Mathematics, 2013, 2013: 160875. [doi: 10.1155/2013/160875]

2 Myreen MO, Fox ACJ, Gordon MJC. Hoare logic for ARM machine code. In: Arbab F, Sirjani M, eds. International Conference on Fundamentals of Software Engineering. Berlin, Heidelberg: Springer, 2007. 272–286.

3 Habibi A, Tahar S, Ghazel A. Formal modelling of the ADSP-2100 processor using HOL. Proceedings of Canadian Conference on Electrical and Computer Engineering. Winnipeg: IEEE, 2002. 614–619. [doi: 10.1109/CCECE.2002.1013012]

4 Elderhalli Y, Hasan O, Tahar S. A methodology for the formal verification of dynamic fault trees using HOL theorem proving. IEEE Access, 2019, 7: 136176–136192. [doi: 10.1109/ACCESS.2019.2942829]

5 Coble AR. Formalized information-theoretic proofs of privacy using the HOL4 theorem-prover. Proceedings of the 8th International Symposium on Privacy Enhancing Technologies Symposium. Leuven: Springer, 2008: 77–98. [doi: 10.1007/978-3-540-70630-4_6]

6 尚亚龙. 基于 RT-中间件数据传输协议的形式化研究 [硕士学位论文]. 北京: 北京化工大学, 2019.

7 Slind K, Norrish M. A brief overview of HOL4. Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics. Berlin: Springer, 2008. 28–32.

8 Dijkstra EW. Recursive programming. Numerische Mathematik, 1960, 2(1): 312–318. [doi: 10.1007/BF01386232]