软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(6):2127−2149 [doi: 10.13328/j.cnki.jos.006574]

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

步进索引模型下的语义及其形式化*

郭 昊, 曹钦翔

(上海交通大学 电子信息与电气工程学院, 上海 200240)

通信作者: 曹钦翔, E-mail: caoqinxiang@gmail.com

摘 要: 霍尔逻辑作为计算机程序的逻辑基础, 可以用于描述一般程序的验证. 分离逻辑作为霍尔逻辑的扩展, 可以支持很多现代程序语言中的高阶特性. 步进索引模型被用于定义自递归谓词. 步进索引逻辑被广泛应用于各种基于交互式定理证明器的程序验证工具中, 然而, 基于步进索引逻辑的推理却比经典逻辑复杂、繁琐. 事实上, 也可以在步进索引模型上定义更加简洁清晰的、与"步数"无关的经典逻辑体系下的非步进索引程序语义. 人们希望找到步进索引逻辑和非步进索引逻辑之间的关系, 但发现两种逻辑并不等价. 对实际的程序验证工作中涉及的命题进行归纳总结, 找出它们共同的特征, 给出关于程序状态的断言的约束条件; 分别定义步进索引逻辑和非步进索引逻辑体系中断言的语义, 并证明在该约束条件下两种语义的等价性; 在 Coq 中, 形式化以上所有定义和证明; 最后, 对未来值得关注的研究方向进行初步探讨.

关键词: 程序状态模型; 程序语言的语义; 形式化验证
中图法分类号: TP311

中文引用格式: 郭昊, 曹钦翔. 步进索引模型下的语义及其形式化. 软件学报, 2022, 33(6): 2127–2149. http://www.jos.org.cn/1000-9825/6574.htm

英文引用格式: Guo H, Cao QX. Semantics under Step-indexed Model and Formalization. Ruan Jian Xue Bao/Journal of Software, 2022, 33(6): 2127−2149 (in Chinese). http://www.jos.org.cn/1000-9825/6574.htm

Semantics under Step-indexed Model and Formalization

GUO Hao, CAO Qin-Xiang

(School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai 200240, China)

Abstract: Hoare logic is the logic base of computer programming. It is used to describe verification of general programs. Separation logic as an extension of Hoare logic, provides supports for high order features used in modern programming languages. Step-indexed model is used to define self-referential predicates. Step-indexed logic is widely used in various program verification tools based on interactive theorem prover, but the reasoning based on step index logic is more complex and complicated than that based on classical logic. On step-indexed model, it is also able to define the non-step-indexed semantics under classical logic system which is more concise and clearer, and independent of the number of steps. Aiming at studying the relationship between stepping index logic and non-stepping index logic, it is found that the two logics are not equivalent. This study summarizes the propositions involved in practical program verification, finds out their common characteristics, and gives the constraint conditions of assertions about program states. The semantics of assertions in step-indexed logic and non-step-indexed logic are defined respectively, and the equivalence of the two semantics is proved under the constraint conditions. All the above definitions and proofs are formalized in Coq. Finally, the future research directions are discussed preliminarily.

Key words: model of program states; semantics of programming language; formal verification

程序开发中, 通常使用测试的方法找出和修复错误. 然而这种测试只能帮助修复错误, 却无法保证完成

* 基金项目: 国家自然科学基金青年科学基金(61902240)
本文由"定理证明理论与应用"专题特约编辑曹钦翔副教授、詹博华副研究员、赵永望教授推荐.
收稿时间: 2021-09-02; 修改时间: 2021-10-14; 采用时间: 2022-01-04; jos 在线出版时间: 2022-01-28

的程序没有错误. 在航天器材或医疗设备等应用场景中, 程序中潜在的错误可能会造成严重的后果, 因此我们希望使用形式化验证方法严格地证明程序的正确性和可靠性. 形式化验证方法将程序代码转换为逻辑公式, 通过逻辑推理证明程序的性质.

Floyd 在“Assigning Meanings to Programs”一文中提供了形式化定义程序含义的基本方法[1], Hoare 在这一工作的基础上, 提出了计算机程序的逻辑基础——霍尔逻辑(Hoare logic)[2]. Hoare 认为, 计算机编程是一门精确的科学, 因为在任何给定的环境中, 程序的所有属性和执行程序的所有结果原则上都可以通过纯粹的演绎推理(deductive reasoning)从程序本身的文本推演得到[2]. Hoare 使用霍尔三元组 $P\{c\}Q$ 表示命题: “若命题 P 在执行程序 c 之前成立, 则命题 Q 在程序 c 执行结束时成立”[2]. 在霍尔逻辑的基础上, 我们可以在程序代码和谓词逻辑公式之间建立起等语义关系的转化, 从而确保我们的验证结果是有效的. 霍尔逻辑被广泛用于描述一般程序的验证, 但是对于和内存状态相关的程序, 并不能很好地进行处理. 现代程序语言中有很多高阶特性, 如复杂数据结构、动态指针、函数指针等. 关于这些高阶特性的验证, 需要分离逻辑(separation logic)的支持. 分离逻辑是对一般霍尔逻辑的一种扩展, 其核心内容是引入了分离合取(separating conjunction)运算,记为 $A*B$[3,4]. 命题“内存 m 满足 $A*B$”的语义为: “m 可以被划分为不相交的两部分 m_1 和 m_2, 它们分别满足命题 A 和命题 B”. 分离逻辑在对并发程序和资源管理的验证方面具有更强的验证能力[5]. 分离逻辑和递归数据结构以及递归函数都非常契合, 对于具有复杂数据结构的动态内存程序的验证非常实用, 在定理证明领域应用十分广泛. HOCAP[6]、iCAP[7]等工具都基于分离逻辑开发.

有了分离逻辑的支持, 这些程序语言的高阶特性还需要复杂的程序状态模型的支持. 例如, 为了支持函数指针, 程序状态的定义由“从地址到值的映射”拓展为“从地址到值或函数的映射”:

$$Mem=Loc\rightharpoonup(Value+Func).$$

在本文使用的模型中, Loc 和 $Value$ 均被简化为自然数集$\mathbb{N}$. 由于函数与调用该函数前后的程序状态有关, 函数类型 $Func$ 是一个与$(Mem\times Mem)$有关的类型. 显然, 该模型的定义涉及了一个自反域方程(reflexive domain equation), 而这样的自反域方程一般来说是不可解的. 为了解决这类问题, America 和 Rutten 提出了这样一种技术, 可以在完备度量空间的范畴(category of complete metric spaces)中找到在一定限制下的自反域方程的解[8,9]. 这一研究结果是很多程序状态模型研究的基础.

步进索引模型(step-indexed model)被用于定义自递归(self-referential)的谓词[10]. 链表、树是最简单的自递归谓词. 以链表为例, 用“$list(p)$”表示“p 指向一个链表”. 不严格地, $list(p)$当且仅当:

$$(^*p=NULL\wedge empty)\vee(^*p\neq NULL\wedge\exists r\in\mathbb{N},(^*p=r\wedge list(r))).$$

然而, 我们不能直接将这样自递归的描述作为链表结构的定义. 注意到: 对于指针这一数据类型, 我们至少要执行一步读取(load)命令才能对其指向的内容做进一步操作, “later”运算符$\triangleright$被引入, 以表示“在执行至少一步命令后命题成立”, 从而避免了循环定义的问题(注: $list$ 这个自递归谓词的定义并非只有引入 later 运算符这一种方法, 但另一个关于协变递归谓词(covariant recursive predicates)的方法[10]与本文的主要内容无关, 故不加赘述). 同样以链表的情况为例, $list$ 可以被定义为函数 F_{list} 的不动点, 其中, F_{list} 的定义为

$$F_{list}(Q,p)\triangleq(^*p=NULL\wedge empty)\vee(^*p\neq NULL\wedge\exists r\in\mathbb{N},(^*p=r\wedge\triangleright Q(r))).$$

收缩的(contracting)运算符$\triangleright$保证了上述函数 F_{list} 有不动点, 而该不动点就是我们希望定义的链表结构. 我们通过引入$\triangleright$, 将命题 Q 转化为了一个稍弱的“Q 的近似”, $\triangleright Q$. 这种稍弱的近似与“步数”有如下关联: 若我们只关心程序执行 k+1 步内的状态, 则命题 Q 表示我们需要 Q 在 k+1 步内成立, 而$\triangleright Q$ 表示我们只需要 Q 在 k 步内成立. 只要我们知道在检验 Q 是否成立之前至少要执行一步命令, $\triangleright Q$ 这种稍弱的近似就足够强. 仍然以链表为例, 因为对 r 进行操作前至少需要执行一步对指针 p 的读取命令, $list(p)$在 k+1 步内成立只需要 $list(r)$在 k 步内成立. 函数指针的类型也可以用类似的方法定义. 类比于指针的读取, 我们至少要执行一步函数调用(function call)命令才能对函数指针指向的内容做进一步操作(注: 本文讨论的主要内容并不涉及自递归谓词).

步进索引方法通过引入“步数”这一概念, 将难以求解的原始方程转化为更容易求解的近似方程, 然后构

造出满足近似方程的结构, 更简单地解决了构造递归结构的问题[11]. Iris[12]、VST[13]等基于交互式定理证明器的程序验证工具都广泛采用步进索引语义. 步进索引方法的核心思想是, 引入步数 n 来表示相应的命题"在 n 步内无法证伪". 步进索引逻辑根据这一思想, 对经典逻辑下的各个逻辑连接词的语义做出了相应的修改. 步进索引模型提供了简洁的定义递归结构的方法, 但基于步进索引逻辑的推理却比经典逻辑复杂、繁琐. 事实上, 我们也可以在步进索引模型上定义更加简洁清晰的、与"步数"无关的经典逻辑体系下的非步进索引程序语义. 经典逻辑体系中的推理更为简单、简洁, 被广泛使用的可满足性模理论求解器(satisfiability modulo theories solver) Z3[14]和合作有效性检查器(cooperating validity checker) CVC4[15]也都是基于经典逻辑开发的. 本文通过讨论命题 P 与命题"P 在任意步数内均无法证伪"的等价性, 探究两种逻辑之间的关系, 举出了在两个逻辑体系中语义不同的简单例子, 找到了保证两个逻辑体系等价的命题约束, 并说明了该约束的合理性. 通过对一些实际的程序验证工作的总结, 我们发现这些程序验证工作涉及到的命题总是符合一些形式, 有一些特殊的性质, 而这些性质又恰好与上述保证两个逻辑体系等价的命题约束有关.

交互式定理证明方法采用严格的数学理论, 对目标对象进行准确且无二异性的严格数学建模、描述、推导与证明. 将以上的所有内容组合到一起是一项非常复杂的工作, 因此我们需要使用交互式定理证明器来确保这些内容的正确性和完整性. 对程序理论琐碎的细节进行形式化, 一方面可以检验理论本身, 另一方面也便于以后将这些理论应用于形式化证明程序的正确性. 因此, 本文使用 Coq[16]对以上工作进行了形式化.

本文第 1 节展示反例并说明两种语义如何不同. 第 2 节提出使两种语义等价的约束. 第 3 节给出该约束下两种语义等价性的证明. 第 4 节说明如何形式化以上内容. 最后总结全文, 并对未来值得关注的研究方向进行初步探讨.

1 两种逻辑体系的不同点

为简明表述, 本文使用 $m\vDash P$ 表示非步进索引逻辑意义下的"程序状态 m 满足断言 P", $m, n\vDash P$ 表示步进索引逻辑意义下的"程序状态 m 在 n 步内满足断言 P".

本文使用 $l\mapsto\{P\}\{Q\}$表示断言"地址 l 上存储着语义为$\{P\}\{Q\}$的函数". 这样的断言是步进索引语义和非步进索引语义主要的分歧点, 其他断言的语义将在第 3.1 节与第 3.2 节中定义. $l\mapsto\{P\}\{Q\}$的非步进索引语义为: "对任意程序状态 m_1, m_2, 若 m_1 满足 P 且在程序状态 m_1 上调用地址 l 上存储的函数, 执行结束时的程序状态是 m_2, 则 m_2 满足 Q". 其(不严格的)步进索引语义为

对任意程序状态 m_1, m_2 和小于 n 的自然数 k, 若 m_1 满足 P,
且在 m_1 上调用地址 l 上存储的函数会在 k 步内终止并到达 m_2,
则 m_2 满足 Q.

此处的"满足"应为步进索引逻辑意义下的, 即:

$m, n\vDash l\mapsto\{P\}\{Q\}$当且仅当对任意 m_1, m_2 和 $k<n$:

若 $m_1, n\vDash P$ 且 $m_1 \xrightarrow{m(l)}_{k,n} m_2$, 则 $m_2, n-k\vDash Q$.

我们希望上述断言在两种逻辑体系中的语义是等价的, 即:

$$\forall l\ m\ P\ Q, (\forall n, (m, n\vDash l\mapsto\{P\}\{Q\}))\Leftrightarrow m\vDash l\mapsto\{P\}\{Q\}.$$

并将上述与函数相关的断言作为可组合的断言的一部分, 使得对任意断言 P 都有:

$$\forall m, (\forall n, (m, n\vDash P))\Leftrightarrow m\vDash P.$$

然而该命题并不成立, 以下展示一个简单的反例.

考虑程序 c: **while** (*0x10)>0 **do** (*0x10):=(*0x10)−1 **endwhile**, 其中, *0x10 表示从程序状态的地址 0x10 处取值, 赋值语句*0x10:=e 表示将表达式 e 的值存入程序状态的地址 0x10 处. 我们可以用程序 c 的语义定义一个函数 f: 在程序状态 m_1 上调用函数 f 执行结束的程序状态是 m_2 当且仅当在程序状态 m_1 上执行程序 c 执行结束的程序状态是 m_2, 调用函数 f 和执行程序 c 所用的步数也相同. 假设程序状态 m 的地址 l 上存储着这个

函数 f. 考虑这样一个断言: $\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\}$, 其中, $0x10\mapsto x$ 意为“程序状态的地址 0x10 处存储着值 x”, False 表示“不存在满足该断言的程序状态”. 符合直觉地, $m\vDash\exists x\in\mathbb{N}, P(x)$意为“存在自然数 x, 使得 m 满足断言 $P(x)$”; $m, n\vDash\exists x\in\mathbb{N}, P(x)$意为“存在自然数 x, 使得 m 在 n 步内满足断言 $P(x)$”. 以下说明该断言的两种语义如何不等价.

在非步进索引逻辑意义下, $m\vDash\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\}$容易证伪, 因为显然不存在自然数 x 可以使程序 c 无法终止, 因此无论 x 取何值, 调用函数 f 也总会终止, 而只要执行终止, 其终止时的程序状态总不满足断言 False, 因此不存在 x 使得 m 满足 $l\mapsto\{0x10\mapsto x\}\{\text{False}\}$, 即 $m\vDash\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\}$不成立.

在步进索引逻辑意义下, 我们可以证明$\forall n, (m, n\vDash\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\})$. 不难发现: 对于任意步数 n, 我们总能找到自然数 $x=n+1$, 使得程序 c 的执行无法在 n 步内终止(故函数 f 的调用无法在 n 步内终止), 所以对任意程序状态 m_1、m_2 和小于 n 的自然数 k, “在 m_1 上调用函数 f 会在 k 步内终止并到达 m_2”都因为不可能在 k 步内终止而不成立, 所以“m 在 n 步内满足 $l\mapsto\{0x10\mapsto x\}\{\text{False}\}$”成立. 因此, 对任意 n, 都存在自然数 $x=n+1$, 使得 m 在 n 步内满足 $l\mapsto\{0x10\mapsto x\}\{\text{False}\}$, 即$\forall n, (m, n\vDash\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\})$成立.

综上所述, $\exists x\in\mathbb{N}, l\mapsto\{0x10\mapsto x\}\{\text{False}\}$的非步进索引语义与步进索引语义不等价.

2 两种语义等价的约束条件

尽管两种逻辑体系并不等价, 我们注意到, 上述反例中的情况并不会出现在真实的程序验证场景中. 因此, 我们希望找到这样一种约束, 它既可以保证满足该约束的命题在两个逻辑体系中完全等价, 又不会影响实际程序验证.

2.1 程序功能正确性证明中使用的断言

本节通过 4 个实际的程序验证工作的例子, 发掘并提出一些可行的约束条件.

2.1.1 归并排序

归并排序是一种经典的排序算法, 它将数组等分为两部分递归地分别排序, 再将两个排序后的子数组合并为一个有序数组. 本文中, 我们用 $ArrayRep(l,_,n)$表示“程序状态上以地址 l 开始存储着一个长度为 n 的数组”, $ArrayRep(l,A)$表示“程序状态上以地址 l 开始存储着数组 A”, 其定义为:

- 对任意地址 l 和正整数 n, $ArrayRep(l,_,n)$当且仅当
$$(l\mapsto\_)*(l+1\mapsto\_)*\dots*(l+n-1\mapsto\_).$$
其中, $(l\mapsto_)$表示“程序状态有地址 l 的权限”, 不关心地址 l 处存储着什么;
- 对任意地址 l 和数组 A, $ArrayRep(l,A)$当且仅当
$$(l\mapsto A[0])*(l+1\mapsto A[1])*\dots*(l+len(A)-1\mapsto A[len(A)-1]).$$
其中, $len(A)$表示数组 A 的长度, $A[i]$表示数组 A 的第 i 位$(0\leqslant i<len(A))$.

归并排序函数可以被描述为
$$\forall A,\{ArrayRep(x,A)\}$$
$$MergeSort(x)$$
$$\{\exists A',(A'\text{是}A\text{排序后的结果})\wedge ArrayRep(x,A')\}.$$

2.1.2 Tarjan 算法

Tarjan 算法是求解有向图强连通分量的经典算法. Tarjan 算法通过深度优先搜索对图进行遍历, 在遍历过程中维护包含 DFN 和 LOW 的状态. $DFN[i]$表示节点 i 在深度优先搜索中被搜索的次序(时间戳), $LOW[i]$表示节点 i 或 i 的子树能追溯到的最早的栈中的节点的次序号. 当 $DFN[i]=LOW[i]$时, 节点 i 及其子树就构成一个强连通分量. 本文中, 我们用 $GraphRep(l,G)$表示“程序状态上地址 l 处存储着有向图 G”. 假设我们将算法的结果, 即图中的所有强连通分量, 存储在 sccList 中, 则该算法可被描述为

$$\forall G,\left\{\begin{array}{l}GraphRep(l,G)*ArrayRep(DFN,\_,|G|)*ArrayRep(LOW,\_,|G|)*\\ ArrayRep(sccList,\_,|G|)*idx\mapsto 0\end{array}\right\}$$
$$Tarjan(l)$$
$$\left\{\begin{array}{l}\exists S,(S\text{中存储着图}G\text{的所有强连通分量})\wedge\\ \left(\begin{array}{l}GraphRep(l,G)*ArrayRep(DFN,\_,|G|)*ArrayRep(LOW,\_,|G|)*\\ ArrayRep(sccList,S)*idx\mapsto\_\end{array}\right)\end{array}\right\},$$

其中, *idx* 表示当前节点的编号; |*G*|表示图 *G* 中点集的大小, 即图中有多少个顶点.

2.1.3 红黑树

红黑树是一种自平衡二叉查找树, 是计算机科学中一种常用的数据结构. 红黑树的基本操作包括插入节点、删除节点和根据键值查找节点. 本文中, 我们用 *TreeRep*(*l*,*T*)表示"程序状态上地址 *l* 处存储着红黑树 *T*".

插入节点的函数可以被描述为

$$\forall T\ k\ v,\{TreeRep(l,T)*\ key\mapsto k*value\mapsto v\}$$
$$Insert(l,key,value)$$
$$\{\exists T',(T'\text{是将键值对}(k,v)\text{插入树}T\text{得到的红黑树})\wedge TreeRep(l,T')\}.$$

删除节点的函数可以被描述为

$$\forall T\ k,\{TreeRep(l,T)*key\mapsto k\}$$
$$Delete(l,key)$$
$$\{\exists T',(T'\text{是从树}T\text{中删除键为}k\text{的节点后得到的红黑树})\wedge TreeRep(l,T')\}.$$

我们将根据键值查找节点的返回值存储在地址 *ret* 中, 则根据键值查找节点函数可以被描述为

$$\forall T\ k,\{TreeRep(l,T)*key\mapsto k\}$$
$$Search(l,key)$$
$$\left\{\begin{array}{l}\exists v,(v\text{是树}T\text{中键为}k\text{的节点的值})\wedge\\ (TreeRep(l,T)*ret\mapsto v)\end{array}\right\}.$$

可以对上述红黑树的描述做出带有函数指针的拓展, 假设用于比较键值的函数存储在地址 *comp* 上, 令:

$$P(comp)\triangleq comp\mapsto\forall k_1\ k_2,\ \{l_1\mapsto k_1*l_2\mapsto k_2\}\{l_1\mapsto k_1*l_2\mapsto k_2*ret'\mapsto(k_1\text{是否小于}\ k_2)\},$$

则插入节点的函数可以被描述为

$$\forall T\ k\ v,\{TreeRep(l,T)*key\mapsto k*value\mapsto v*P(comp)\}$$
$$Insert(l,key,value,comp)$$
$$\left\{\begin{array}{l}\exists T',(T'\text{是将键值对}(k,v)\text{插入树}T\text{得到的红黑树})\wedge\\ (TreeRep(l,T')*P(comp))\end{array}\right\}.$$

删除节点的函数可以被描述为

$$\forall T\ k,\{TreeRep(l,T)*key\mapsto k*P(comp)\}$$
$$Delete(l,key,comp)$$
$$\left\{\begin{array}{l}\exists T',(T'\text{是从树}T\text{中删除键为}k\text{的节点后得到的红黑树})\wedge\\ (TreeRep(l,T')*P(comp))\end{array}\right\}.$$

根据键值查找节点函数可以被描述为

$$\forall T\ k,\{TreeRep(l,T)*key\mapsto k*P(comp)\}$$
$$Search(l,key,comp)$$
$$\left\{\begin{array}{l}\exists v,(v\text{是树}T\text{中键为}k\text{的节点的值})\wedge\\ (TreeRep(l,T)*ret\mapsto v*P(comp))\end{array}\right\}.$$

2.1.4 斐波那契数列

斐波那契数列指的是这样一个数列: 1, 1, 2, 3, 5, 8, 13, 21, 34, ... 这个数列从第 3 项开始, 每一项都等于

前两项之和. 为方便表述, 我们不妨设该数列的第 0 项为 0. 我们可以通过一个循环简单地求出斐波那契数列的第 N 项(执行该循环前$(*n)=N$).

```
1  while (*n)>0 do
2     (*n):=(*n)–1;
3     *c:=(*b)+(*a);
4     *a:=*b;
5     *b:=*c
6  endwhile
```

用上述程序的语义定义求解斐波那契数列的第 N 项的函数 f, 则 f 可以被描述为

$$\forall N,\{n\mapsto N*a\mapsto 0*b\mapsto 1*c\mapsto \_\}$$

$$f(\cdot)$$

$$\{n\mapsto 0*a\mapsto fib(N)*b\mapsto fib(N+1)*c\mapsto \_\},$$

其中, $fib(i)$表示斐波那契数列的第 i 项. 注意到, $fib(i)+fib(i+1)=fib(i+2)$对全体自然数 i 都成立.

为了验证这一描述的正确性, 我们找到循环不变量:

$$\{\exists k,\ (0\leqslant k\leqslant N)\wedge(n\mapsto(N-k)*a\mapsto fib(k)*b\mapsto fib(k+1)*c\mapsto \_)\}.$$

借助这一循环不变量, 函数 f 描述的正确性不难证明.

2.1.5 总结断言的特征

通过以上 4 个例子, 我们可以总结出实际使用的断言的普遍特征.

(1) 描述一段程序(或函数)的性质时, 使用任意量词而不用存在量词.

在实际的程序验证中, 我们只可能用任意量词修饰描述函数的谓词(即 $l\mapsto\{P\}\{Q\}$), 而不会使用存在量词. 以常见的阶乘函数举例, 我们希望一个阶乘函数可以正确计算并返回任意自然数的阶乘(不严格地, $\forall x$, $(l\mapsto\{0x10\mapsto x\}\{0x10\mapsto x!\})$是一个判断地址 l 上是否存储着一个阶乘函数的断言), 而不是可以对某个自然数进行计算(与前面的例子相对应, $\exists x$, $(l\mapsto\{0x10\mapsto x\}\{0x10\mapsto x!\})$不是一个有实际意义的断言, 它也不会出现在实际的程序验证工作中). 从上文的例子中也能够发现: 归并排序可以对任意数组进行排序, Tarjan 算法可以对任意有向图求解, 红黑树相关的函数可以对任意红黑树进行操作, 斐波那契数列也可以对任意项求解. 将这些例子的函数描述中的“任意”换成“存在”, 显然会将这些函数的描述变成没有实际意义的命题.

(2) 描述一个程序状态时, 尤其是在循环不变量中, 使用存在量词而不用任意量词.

与函数的描述相反, 我们只会使用存在量词来修饰一个关于程序状态的断言, 而不会使用任意量词. 例如, 断言$\forall l$, $l\mapsto_$不是一个有意义的断言, 因为任何一段程序都不可能需要内存上每个地址的权限(注意: 在我们的模型中, 内存上有无穷多个地址).

(3) 可以分为“内存如何存储某些数学对象”和“这些数学对象满足什么条件”这两部分.

“内存如何存储某些数学对象”这一部分内部用分离合取*而不用合取∧连接, “这些数学对象满足什么条件”这一部分可以是任意的数学命题, 两部分之间用合取∧连接. 一个断言可能有多种逻辑上等价的形式, 如:

$$(P(x_1)\text{且 }Q(x_2))\wedge(l_1\mapsto x_1*l_2\mapsto x_2)\Leftrightarrow(P(x_1)\wedge l_1\mapsto x_1)*(Q(x_2)\wedge l_2\mapsto x_2).$$

但出于习惯, 我们总会使用符合该约束的形式.

此外, 我们还对断言额外提出两个要求: 带有参数的断言参数取值的唯一性和带有分离合取的断言内存划分方式的唯一性. 这两个唯一性要求在后续的等价性证明中起到了关键的作用, 在实际的程序验证工作中也有合理性. 以下简单说明这两个要求.

(1) 要求带有参数的断言有唯一性.

在实际的程序验证过程中, 对于类似上述反例中的存在性断言$\exists x\in\mathbb{N}$, $P(x)$, 其中, $P(x)$是一个带有参数的断言. 如果某个程序状态在一定步数以上满足$\exists x\in\mathbb{N}$, $P(x)$, 我们总能根据这个程序状态找到唯一确定的 x, 即

“存在正整数 N 使得对任意 $m, n\geqslant N, x, y$, 若 $m, n\vDash P(x)$且 $m, n\vDash P(y)$, 则 $x=y$”. 以断言$\exists x\in\mathbb{N}, 0x10\mapsto x$ 为例, 显然对于任何一个程序状态 m, 其地址 0x10 上存储的值都是唯一确定的, 故而对任意 x 和 y 都有: “若 $m, 1\vDash 0x10\mapsto x$ 且 $m, 1\vDash 0x10\mapsto y$, 则 $x=y$”. 不难发现: 上文例子中的表示内存中某地址上存在一张图、一棵树或一个数组的断言, 以及描述循环不变量的断言也都符合这一约束. 事实上, 实际的程序验证中涉及的大多数带有参数的断言都像这些例子一样, 其参数的取值可以在一步以内就被唯一确定(即“对任意 m, x, y, 若 $m, 1\vDash P(x)$且 $m, 1\vDash P(y)$, 则 $x=y$”). “存在正整数 N 使得对任意 $m, n\geqslant N, x, y$, 若 $m, n\vDash P(x)$且 $m, n\vDash P(y)$, 则 $x=y$”的约束有效地防止了第 1 节举出的反例中, 根据不同剩余步数可以找到不同的值, 使得步进索引命题成立而非步进索引命题不成立的情况. 实际的程序验证中涉及的断言也都满足这一约束.

上述对带有参数的断言唯一性的约束是语义上的约束.

实际使用的断言大多满足这一语法约束: 存在性断言通常形如$\exists x, P(x)\wedge l\mapsto x$ 而不会是$\exists x, P(y)\wedge x\mapsto y$ 的形式. 不难发现, 上文 4 个实际的程序验证工作的例子都满足这一语法上的约束. 显然, 在给定程序状态的情况下, 该语法约束事实上可以推导出上述语义约束. 因此, 本文不再对语法约束作进一步讨论.

注意: 这里要求的参数的唯一性是指给定程序状态的情况下唯一, 并非对程序的所有可达状态(即未知程序状态的情况下)唯一.

(2) 要求对于任意与分离合取相关的断言都存在一个步数 N, 使得在 N 步以上内存的划分方式是唯一的.

不满足这个约束的断言通常没有实际意义. 以断言$(l_0\mapsto x_0\vee l_1\mapsto x_1)*l_2\mapsto x_2$ 为例, 我们无法判断满足该断言的程序状态究竟满足 $l_0\mapsto x_0$ 或 $l_1\mapsto x_1$ 中的哪一条, 而这导致关于 l_0、x_0、l_1 或 x_1 的推理均无法进行下去. 这一约束防止了含有分离合取的断言根据不同剩余步数有不同的内存分配方式导致步进索引语义和非步进索引语义出现分歧的问题, 同时, 实际的程序验证中涉及的断言也都满足这一约束.

2.2 断言及其约束条件的定义

程序状态的断言 P 的语法可以被归纳地定义为

$$P\triangleq l\mapsto v|l\mapsto\{P\}\{P\}|P*P|P\vee P|A\wedge P|\exists v,P(v)|l\mapsto\forall v,\{P(v)\}\{P(v)\},$$

其中, l 代表程序状态的一个地址, v 代表一个值, A 代表一个与程序状态无关的命题(本文的模型中多为数学命题, 如“$x<y$”, “$x+y=z$”等). 注意到, 断言只在描述函数的性质时使用任意量词. 为了强调这一性质并避免混淆, 我们将$\forall v, l\mapsto\{P(v)\}\{P(v)\}$写成 $l\mapsto\forall v, \{P(v)\}\{P(v)\}$.

接下来我们给出第 2.1.5 节中对断言的两个额外要求的形式化定义.

定义 1. 断言 P 和 Q 可以“合法进行分离合取”当且仅当:

存在正整数 N, 使得对任意 m_1, m_2, m_1', m_2' 和 $n\geqslant N$ 都有:

若(m_1,m_2)和 (m_1',m_2') 都是 m 的划分,

且 $m_1, n\vDash P$, $m_1', n\vDash P$, $m_2, n\vDash Q$, $m_2', n\vDash Q$, ,

则 $m_1=m_1'$ 且 $m_2=m_2'$.

这一性质与分离逻辑中断言的准确性有关. 一个断言 p 是准确的(precise)当且仅当对任意 s 和 h 都最多只有一个 $h'\subseteq h$, 使得 $s, h'\vDash p$[17]. 一般来说, 分离逻辑中的断言准确性不在步进索引模型下进行讨论, 这里相当于添加了该性质和步数的关系.

定义 2. 带有参数的断言 P(即 $P(v)$是一个断言)具有唯一性当且仅当:

存在正整数 N, 使得对任意 m, x, y 和 $n\geqslant N$ 都有:

若 $m, n\vDash P(x)$且 $m, n\vDash P(y)$, 则 $x=y$.

综合上述约束条件, 断言的合法性可以被严格定义.

定义 3. 断言 P'是合法的, 当且仅当:

1. $P'=l\mapsto v$ 或
2. $P'=l\mapsto\{P\}\{Q\}$且 P 和 Q 都是合法的或

3. $P'=P*Q$ 且 P 和 Q 都是合法的且 P 和 Q 可以合法进行分离合取(定义 1)或
4. $P'=P\vee Q$ 且 P 和 Q 都是合法的或
5. $P'=A\wedge P$ 且 P 是合法的或
6. $P'=\exists v, P(v)$且 $P(v)$是合法的且 P 具有唯一性(定义 2)或
7. $P'=l\mapsto\forall v, \{P(v)\}\{Q(v)\}$且 $P(v)$和 $Q(v)$都是合法的且 P 具有唯一性(定义 2).

为了之后证明的简洁, 我们进一步将对函数前条件的唯一性约束加强为

对任意 m, n_1, n_2, v_1 和 v_2, 若 $m, n_1\vDash P(v_1)$且 $m, n_2\vDash P(v_2)$, 则 $v_1=v_2$.

以下我们证明在这些约束条件下两种语义的等价性.

3 约束条件下两种语义等价性的证明

断言的步进索引语义和非步进索引语义分别可以有如下定义.

3.1 步进索引语义的定义

1. $m, n\vDash l\mapsto v$ 当且仅当程序状态 m 的地址 l 上存储着值 v;
2. $m, n\vDash l\mapsto\{P\}\{Q\}$当且仅当对任意 $n'\leqslant n, m_1, m_2$ 和 $k<n'$:

   若 $m_1, n'\vDash P$ 且 $m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则 $m_2, n'-k\vDash Q$;
3. $m, n\vDash P*Q$ 当且仅当存在 m 的一种划分(m_1,m_2), 使得 $m_1, n\vDash P$ 且 $m_2, n\vDash Q$;
4. $m, n\vDash P\vee Q$ 当且仅当 $m, n\vDash P$ 或 $m, n\vDash Q$;
5. $m, n\vDash A\wedge P$ 当且仅当 A 成立且 $m, n\vDash P$;
6. $m, n\vDash\exists v, P(v)$当且仅当存在 v, 使得 $m, n\vDash P(v)$;
7. $m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}$当且仅当对任意 $v, n'\leqslant n, m_1, m_2$ 和 $k<n'$:

   若 $m_1, n'\vDash P(v)$且 $m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则 $m_2, n'-k\vDash Q(v)$.

本文暂不考虑“m 在 0 步内满足断言 P”的语义, 步进索引语义中涉及的步数均为正整数.

3.2 非步进索引语义的定义

1. $m\vDash l\mapsto v$ 当且仅当程序状态 m 的地址 l 上存储着值 v;
2. $m\vDash l\mapsto\{P\}\{Q\}$当且仅当对任意 m_1, m_2 和 k:

   若 $m_1\vDash P$ 且对任意 n'都有 $m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则 $m_2\vDash Q$;
3. $m\vDash P*Q$ 当且仅当存在 m 的一种划分(m_1,m_2), 使得 $m_1\vDash P$ 且 $m_2\vDash Q$;
4. $m\vDash P\vee Q$ 当且仅当 $m\vDash P$ 或 $m\vDash Q$;
5. $m\vDash A\wedge P$ 当且仅当 A 成立且 $m\vDash P$;
6. $m\vDash\exists v, P(v)$当且仅当存在 v, 使得 $m\vDash P(v)$;
7. $m\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}$当且仅当对任意 v, m_1, m_2 和 k:

   若 $m_1\vDash P(v)$且对任意 n'都有 $m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则 $m_2\vDash Q(v)$.

本文中, 用“$m_1\xrightarrow{f}_{k,n} m_2$ 对任意 n 都成立”表示非步进索引意义下的“在 m_1 上调用函数 f, 执行结束时的程序状态是 m_2”.

3.3 其他重要的定义

为了证明上述约束条件下两种逻辑体系的等价性, 我们引入$\Diamond^n$符号, 用来修饰断言 P. $m\vDash\Diamond^n P$ 表示“存在 n 步内无法与 m 区分的 m', 使得 $m'\vDash P$”. 其中, “无法区分”定义为定义 4.

定义 4. 程序状态 m_1 与 m_2 在 $n+1$ 步内无法区分, 当且仅当对任意地址 l 都有以下 3 项中某一项成立.

1. m_1 和 m_2 均没有地址 l 的权限;

2. m_1 和 m_2 在地址 l 上储存的值相等;

3. m_1 和 m_2 在地址 l 上储存的函数在 n 步内无法区分.

记为“$m_1 \equiv_n m_2$”. 其中, “函数 f_1 和 f_2 在 n 步内无法区分”定义为定义 5.

定义 5. 函数 f_1 和 f_2 在 n 步内无法区分当且仅当:

对任意程序状态 m_1, m_2 和步数 $k<n$, $m_1 \xrightarrow{f_1}_{k,n} m_2$ 当且仅当 $m_1 \xrightarrow{f_2}_{k,n} m_2$.

显然, 若两个函数在较大的步数内无法区分, 则它们在较小的步数内一定也无法区分; 若两个程序状态在较大的步数内无法区分, 则它们在较小的步数内也无法区分.

步进索引语义应当有以下两条性质.

1. 对任意 m, n_1, n_2 和 P, 若 $m, n_1 \vDash P$ 且 $n_1 \geqslant n_2$, 则 $m, n_2 \vDash P$;

2. 对任意 m_1, m_2, n 和 P, 若 $m_1 \equiv_n m_2$ 且 $m_1, n \vDash P$, 则 $m_2, n \vDash P$.

这两条性质根据上述定义均能简单加以证明.

我们假设函数有以下性质.

1. 保距性: 对任意 $f, m_1, m_2, m_1', m_2', k$ 和 n:

若 $m_1 \equiv_n m_1'$ 且 $m_2 \equiv_{n-k} m_2'$, 则 $m_1 \xrightarrow{f}_{k,n} m_2$ 当且仅当 $m_1' \xrightarrow{f}_{k,n} m_2'$.

2. 向下闭合: 对任意 f, m_1, m_2, k, n 和 n':

若 $m_1 \xrightarrow{f}_{k,n} m_2$ 且 $n \geqslant n'$, 则 $m_1 \xrightarrow{f}_{k,n'} m_2$.

3. 对任意 f, m_1, m_2, n 和 $k<n$:

若 $m_1 \xrightarrow{f}_{k,n} m_2$, 则存在 $m_1' \equiv_n m_1$ 和 $m_2' \equiv_{n-k} m_2$, 使得 $m_1' \xrightarrow{f}_{k,n'} m_2'$ 对任意 n'都成立.

4. 对任意 f, m_1, m_2, n 和 $k<n$:

若 $m_1 \xrightarrow{f}_{k,n} m_2$, 则对任意 $m_1' \equiv_n m_1$, 都存在 $m_2' \equiv_{n-k} m_2$, 使得 $m_1' \xrightarrow{f}_{k,n'} m_2'$ 对任意 n'都成立.

注意: 这里的函数 f 不是由语法树定义的类型, 而是对函数语义的描述(即函数的 denotation). 本文中, 我们将函数类型 $Func$ 定义为满足以上 4 条性质的四元关系. 不妨假设: 对任意 f, m_1, m_2, k 和 $n \leqslant k$, 都有 $m_1 \xrightarrow{f}_{k,n} m_2$. 这一假设不会违反以上任意一条性质, 并且可以避免对 k 和 n 大小情况进行分类讨论.

3.4 等价性的证明

以下证明两种语义的等价性, 即:

对任意合法的断言 P 和程序状态 m, ($m, n \vDash P$ 对任意正整数 n 都成立)当且仅当 $m \vDash P$.

我们先将这一结论扩展为

对任意合法的断言 P 和程序状态 m,

$(\forall n, (m, n \vDash P \Leftrightarrow m \vDash \Diamond^n P))$且

$((\forall n, (m, n \vDash P)) \Leftrightarrow m \vDash P)$且

$((\forall n, (m \vDash \Diamond^n P)) \Leftrightarrow m \vDash P)$.

注意到, 对任意 m 和 n 都有 $m \equiv_n m$, 故对任意 m 和 P, $m \vDash P \Rightarrow \forall n, (m \vDash \Diamond^n P)$显然成立.

对任意断言 P, “对于任意 m 都有$(\forall n, m \vDash \Diamond^n P \Rightarrow m, n \vDash P)$”需依赖“对于任意 m 都有 $m \vDash P \Rightarrow \forall n, (m, n \vDash P)$”进行证明: 对任意 m, 由 $\Diamond^n$ 符号的定义, $m \vDash \Diamond^n P$ 即为$\exists m' \equiv_n m, m' \vDash P$; 由“对于任意 m 都有 $m \vDash P \Rightarrow \forall n, (m, n \vDash P)$”可推得 $m', n \vDash P$; 由步进索引语义的性质 2, $m, n \vDash P$.

对任意断言 P, “对于任意 m 都有$(\forall n, (m \vDash \Diamond^n P)) \Leftrightarrow m \vDash P$”需依赖“对于任意 m 都有$\forall n, (m, n \vDash P \Leftrightarrow m \vDash \Diamond^n P)$”和“对于任意 m 都有$(\forall n, (m, n \vDash P)) \Leftrightarrow m \vDash P$”进行证明. 在这两个前提条件下, 证明十分显然.

以下使用结构归纳法证明另外 3 个分支, 即对任意合法的断言 P 和程序状态 m:

$(\forall n, (m, n \vDash P \Leftrightarrow m \vDash \Diamond^n P))$且$((\forall n, (m, n \vDash P)) \Leftrightarrow m \vDash P)$.

作为归纳假设, 以下涉及的断言 $P, Q, P(v)$和 $Q(v)$均满足:

对任意程序状态 m,

$$(\forall n, (m, n\vDash X\Leftrightarrow m\vDash\Diamond^{n}X))\text{且}$$
$$((\forall n, (m, n\vDash X))\Leftrightarrow m\vDash X)\text{且}$$
$$((\forall n, (m\vDash\Diamond^{n}X))\Leftrightarrow m\vDash X),$$

其中, $X=P$ 或 Q 或 $P(v)$或 $Q(v)$.

与函数指针有关的断言是上述命题证明中的关键部分, 我们以引理的方式给出这些关键证明的详细过程. 首先, 我们对断言 $l\mapsto\{P\}\{Q\}$证明该命题, 即证:

对任意程序状态 m:

$$(\forall n, (m, n\vDash X\Leftrightarrow m\vDash\Diamond^{n}X))\text{且}((\forall n, (m, n\vDash X))\Leftrightarrow m\vDash X),$$

其中, $X=l\mapsto\{P\}\{Q\}$.

引理 1. 对任意 m, $(\forall n, (m, n\vDash l\mapsto\{P\}\{Q\}))\Leftrightarrow m\vDash l\mapsto\{P\}\{Q\}$,即:

对任意 m,

若 (对任意n, $n'\leqslant n$, m_1, m_2和$k<n'$, 若m_1, $n'\vDash P$且$m_1\xrightarrow{m(l)}_{k,n'}m_2$, 则$m_2$, $n'-k\vDash Q$),

则 (对任意m_1, m_2, k, 若$m_1\vDash P$且($m_1\xrightarrow{m(l)}_{k,n}m_2$对任意$n$都成立), 则$m_2\vDash Q$).

证明:

对任意 m_1, m_2 和 k, 由归纳假设可知:

$$m_1\vDash P\Rightarrow(\forall n, (m_1, n\vDash P)), (\forall n, (m_2, n\vDash Q))\Rightarrow m_2\vDash Q.$$

由$(\forall n, (m, n\vDash l\mapsto\{P\}\{Q\}))$, 对于任意 n, 要证明 $m_2, n\vDash Q$, 只需证明:

(1) $m_1, n+k\vDash P$; 和

(2) $m_1\xrightarrow{m(l)}_{k,n+k}m_2$.

由“$m_1\vDash P\Rightarrow(\forall n, (m_1, n\vDash P))$”和“$m_1\xrightarrow{m(l)}_{k,n}m_2$ 对任意 n 都成立”, 上述两个条件显然对任意 n 都成立, 故有 $(\forall n, (m_2, n\vDash Q))$. 因此该引理成立. □

引理 2. 对任意 m, $m\vDash l\mapsto\{P\}\{Q\}\Rightarrow(\forall n, (m, n\vDash l\mapsto\{P\}\{Q\}))$, 即:

对任意 m,

若 (对任意m_1, m_2, k, 若$m_1\vDash P$且($m_1\xrightarrow{m(l)}_{k,n}m_2$对任意$n$都成立), 则$m_2\vDash Q$),

则 (对任意n, $n'\leqslant n$, m_1, m_2和$k<n'$, 若m_1, $n'\vDash P$且$m_1\xrightarrow{m(l)}_{k,n'}m_2$, 则$m_2$, $n'-k\vDash Q$).

证明:

对任意 m_1, m_2, k, n 和 $n'\leqslant n$, 由归纳假设可知:

$$m_1, n'\vDash P\Rightarrow m_1\vDash\Diamond^{n'}P, \text{ 即 } \exists m_1'\equiv_{n'}m_1, m_1'\vDash P.$$

由函数性质 4, $m_1\xrightarrow{m(l)}_{k,n'}m_2$ 可以推出:

对于 $m_1'\equiv_{n'}m_1$, 存在 $m_2'\equiv_{n'-k}m_2$, 使得 $m_1'\xrightarrow{m(l)}_{k,n_0}m_2'$ 对任意正整数 n_0 都成立.

由 $m\vDash l\mapsto\{P\}\{Q\}$, $m_1'\vDash P$ 和 $\forall n_0, m_1'\xrightarrow{m(l)}_{k,n_0}m_2'$, 可以推得 $m_2'\vDash Q$.

再由归纳假设可知, $m_2'\vDash Q\Rightarrow\forall n, (m_2, n\vDash Q)$.

故有 $m_2', n'-k\vDash Q$. 由步进索引语义的性质 2 可以推出 $m_2, n'-k\vDash Q$.

因此, 该引理成立. □

引理 3. 对任意 m, n, m, $n\vDash l\mapsto\{P\}\{Q\}\Rightarrow m\vDash\Diamond^{n}l\mapsto\{P\}\{Q\}$, 即:

对任意 m, n,

若 (对任意$n'\leqslant n$, m_1, m_2和$k<n'$, 若m_1, $n'\vDash P$且$m_1\xrightarrow{m(l)}_{k,n'}m_2$, 则$m_2$, $n'-k\vDash Q$),

则存在 $m'\equiv_n m$, 使得 (对任意m_1, m_2, k, 若$m_1\vDash P$且($m_1\xrightarrow{m'(l)}_{k,n}m_2$对任意$n$都成立), 则$m_2\vDash Q$).

证明:

对任意 m, n, 该引理的证明本质上是要找到与 $f=m(l)$在 n 步内无法区分的函数 f'. 取 f'为

$$m_1 \xrightarrow{f'}_{k,n'} m_2 \text{ 当且仅当: } n' \leqslant n \Rightarrow m_1 \xrightarrow{f}_{k,n'} m_2 \text{ 且}$$
$$n' > n \Rightarrow (m_1 \xrightarrow{f}_{k,n} m_2 \text{且} \forall n'' \leqslant n',\ k < n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n'' - k \vDash Q).$$

显然有 f'和 f 在 n 步内无法区分.

令 $m'(l)=f'$, 且 m'在其他地址上与 m 保持一致, 则有 $m' \equiv_n m$.

以下证明:

$$(\text{对任意} m_1,\ m_2,\ k,\ \text{若} m_1 \vDash P \text{且} (m_1 \xrightarrow{m'(l)}_{k,n} m_2 \text{对任意} n \text{都成立}),\ \text{则} m_2 \vDash Q).$$

对任意 m_1, m_2 和 k, 由归纳假设可知:

$$m_1 \vDash P \Rightarrow (\forall n,\ (m_1,\ n \vDash P)),\ (\forall n,\ (m_2,\ n \vDash Q)) \Rightarrow m_2 \vDash Q.$$

要证明 $m_2 \vDash Q$, 只需证明对任意 n_0, 都有 $m_2,\ n_0 \vDash Q$.

由 $m_1 \xrightarrow{m'(l)}_{k,n} m_2$ 对任意 n 都成立, 可知 $m_1 \xrightarrow{m'(l)}_{k,n_0+k} m_2$.

以下按 n_0+k 和 n 的大小关系分情况证明.

1) 当 $n_0+k \leqslant n$ 时, 由 f'的定义可知 $m_1 \xrightarrow{m(l)}_{k,n_0+k} m_2$.

因此, 对于 $n_0+k \leqslant n$, m_1, m_2 和 $k<n_0+k$,

由 $m, n \vDash l \mapsto \{P\}\{Q\}$, $m_1,\ n_0+k \vDash P$ 和 $m_1 \xrightarrow{m(l)}_{k,n_0+k} m_2$,

可以推出 $m_2,\ n_0+k-k \vDash Q$, 即 $m_2,\ n_0 \vDash Q$.

2) 当 $n_0+k>n$ 时, 由 f'的定义可知:

$$\forall n'' \leqslant n_0+k,\ k<n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n''-k \vDash Q.$$

取 $n''=n_0+k$, 则显然有 $k<n_0+k$ 和 $m_1,\ n_0+k \vDash P$,

故 $m_2,\ n_0+k-k \vDash Q$, 即 $m_2,\ n_0 \vDash Q$.

综上所述, 该引理成立. □

注意到, 我们找到的函数 f'不能违反前面假设的函数性质. 以下给出 f'符合这些性质的证明.

利用函数 f 的性质 1 和步进索引语义的性质, 证明函数 f'具有性质 1.

性质 1. 对任意 $m_1,\ m_2,\ m_1',\ m_2'$, k 和 n':

$$\text{若 } m_1 \equiv_{n'} m_1' \text{ 且 } m_2 \equiv_{n'-k} m_2',\ \text{则 } m_1 \xrightarrow{f'}_{k,n'} m_2 \Leftrightarrow m_1' \xrightarrow{f'}_{k,n'} m_2'.$$

证明:

对于 $n' \leqslant n$, 由函数 f 的性质 1 可以简单推出该命题成立;

对于 $n'>n$, 即证:

$$(m_1 \xrightarrow{f}_{k,n} m_2 \text{且} \forall n'' \leqslant n',\ k < n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n'' - k \vDash Q) \Leftrightarrow$$
$$(m_1' \xrightarrow{f}_{k,n} m_2' \text{且} \forall n'' \leqslant n',\ k < n'' \Rightarrow m_1',\ n'' \vDash P \Rightarrow m_2',\ n'' - k \vDash Q),$$

其中, $m_1 \xrightarrow{f}_{k,n} m_2 \Leftrightarrow m_1' \xrightarrow{f}_{k,n} m_2'$ 也可以由函数 f 的性质 1 简单推出.

以下给出关于$\forall n'' \leqslant n',\ k \leqslant n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n''-k \vDash Q$ 这一部分的证明, 以一个方向为例:

对任意 $n'' \leqslant n'$, 若 $k \leqslant n''$且 $m_1',\ n'' \vDash P$,

由步进索引语义的两条性质可推得 $m_1,\ n'' \vDash P$,

故有 $m_2,\ n''-k \vDash Q$. 再由步进索引语义的两条性质可推得 $m_2',\ n'' - k \vDash Q$, 故有:

$$(\forall n'' \leqslant n',\ k < n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n'' - k \vDash Q) \Rightarrow$$
$$(\forall n'' \leqslant n',\ k < n'' \Rightarrow m_1',\ n'' \vDash P \Rightarrow m_2',\ n'' - k \vDash Q).$$

反方向同理.

因此, 函数 f'具有函数的性质 1. □

利用函数 f 的性质 2, 证明函数 f'具有性质 2.

性质 2. 对任意 $m_1,\ m_2,\ k,\ n_1$ 和 n_2:

$$\text{若 } m_1 \xrightarrow{f'}_{k,n_1} m_2 \text{ 且 } n_1 \geqslant n_2,\ \text{则 } m_1 \xrightarrow{f'}_{k,n_2} m_2.$$

证明:

对于 $n_2 \leqslant n$, 由函数 f 的性质 2 可以简单推出该命题成立;

对于 $n_2 > n$, 显然有 $n_1 \geqslant n_2 > n$. 原命题转化为证明:

$$(m_1 \xrightarrow{f}_{k,n} m_2 且 \forall n'' \leqslant n_1,\ k < n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n'' - k \vDash Q) \Rightarrow$$
$$(m_1 \xrightarrow{f}_{k,n} m_2 且 \forall n'' \leqslant n_2,\ k < n'' \Rightarrow m_1,\ n'' \vDash P \Rightarrow m_2,\ n'' - k \vDash Q).$$

由于 $n_1 \geqslant n_2$, 该命题显然成立.

因此, 函数 f' 具有函数的性质 2. □

利用函数 f 的性质和步进索引语义的性质, 按照 n' 和 n 的大小关系分类讨论, 证明函数 f' 具有性质 3.

性质 3. 对任意 m_1, m_2, n' 和 $k<n'$:

若 $m_1 \xrightarrow{f'}_{k,n'} m_2$, 则存在 $m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立.

证明:

对于 $n' \leqslant n$, 由函数 f'的定义可知 $m_1 \xrightarrow{f}_{k,n'} m_2$.

由函数 f 的性质 3 可以推出:

存在 $m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f}_{k,n''} m_2'$ 对任意 n''都成立.

以下按照 $m_1', n \vDash P$ 是否成立, 分情况证明:

存在 $m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立.

1) 若 $m_1', n \vDash P$ 成立, 由 $m, n \vDash l \mapsto \{P\}\{Q\}$, $m_1', n \vDash P$ 和 $m_1' \xrightarrow{f=m(l)}_{k,n} m_2'$ 可以推出 $m_2', n-k \vDash Q$.

由归纳假设, 有 $m_2', n-k \vDash Q \Rightarrow m_2' \vDash \diamondsuit^{n-k} Q$,

即存在 $m_2'' \equiv_{n-k} m_2'$, 使得 $m_2'' \vDash Q$.

存在 $m_1' \equiv_{n'} m_1$ 和 $m_2'' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2''$ 对任意 n''都成立!

$m_1' \equiv_{n'} m_1$ 显然成立.

由 $n' \leqslant n$ 可以推出 $m_2'' \equiv_{n'-k} m_2'$, 又因为 $m_2' \equiv_{n'-k} m_2$, 故 $m_2'' \equiv_{n'-k} m_2$.

由函数 f 的性质 1, $m_2'' \equiv_{n-k} m_2'$ 和 $m_1' \xrightarrow{f}_{k,n} m_2'$ 可以推得 $m_1' \xrightarrow{f}_{k,n} m_2''$.

再由函数 f 的性质 2, $m_1' \xrightarrow{f}_{k,n''} m_2''$ 对任意 $n'' \leqslant n$ 都成立.

因此, 要证 $m_1' \xrightarrow{f'}_{k,n''} m_2''$ 对任意 n''都成立, 我们只需证明:

对任意 $n'' > n$ 和 $n_0 \leqslant n''$, $k < n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2'', n_0 - k \vDash Q$.

而由于 $m_2'' \vDash Q$, 由归纳假设可知, $\forall n_0, (m_2'', n_0 \vDash Q)$.

故该命题成立.

2) 若 $m_1', n \vDash P$ 不成立, 则:

存在 $m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立!

$m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$ 显然成立.

由于 $m_1' \xrightarrow{f}_{k,n''} m_2'$ 对任意 n''都成立,

要证 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立, 只需证明:

对任意 $n'' > n$ 和 $n_0 \leqslant n''$, $k < n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2', n_0 - k \vDash Q$.

对于任意 $n_0 \geqslant n$, 显然有 $m_1', n_0 \nvDash P$, 故该命题成立;

对于任意 $n_0 < n$, 由 $m, n \vDash l \mapsto \{P\}\{Q\}$和步进索引语义的性质 1 可以推得:

$$m, n_0 \vDash l \mapsto \{P\}\{Q\}.$$

由 $m, n_0 \vDash l \mapsto \{P\}\{Q\}$, $m_1', n_0 \vDash P$ 和 $m_1' \xrightarrow{m(l)=f}_{k,n_0} m_2'$ 可以推出:

$$m_2', n_0 - k \vDash Q.$$

故该命题成立.

对于 $n'>n$, 按照 $m_1, n' \vDash P$ 是否成立, 分情况证明:

存在 $m_1' \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立.

1) 若 $m_1, n' \vDash P$ 成立, 由 f'的定义可推得 $m_1 \xrightarrow{f}_{k,n} m_2$ 和 $m_2, n'-k \vDash Q$.

由归纳假设, $m_2, n'-k \vDash Q \Rightarrow m_2, \Diamond^{n'-k} \vDash Q$, 即存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_2' \vDash Q$.

存在 $m_1 \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1 \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立!

$m_1 \equiv_{n'} m_1$ 和 $m_2' \equiv_{n'-k} m_2$ 显然成立.

因为 $n'>n$, 由 $m_2' \equiv_{n'-k} m_2$ 可以推出 $m_2' \equiv_{n-k} m_2$.

由函数 f 的性质 1 可以推出 $m_1 \xrightarrow{f}_{k,n} m_2'$.

再由函数 f 的性质 2, 可以推出 $m_1 \xrightarrow{f}_{k,n''} m_2'$ 对任意 $n'' \leqslant n$ 都成立.

因此, 要证 $m_1 \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立, 只需证明:

对任意 $n''>n$ 和 $n_0 \leqslant n''$, $k < n_0 \Rightarrow m_1, n_0 \vDash P \Rightarrow m_2', n_0 - k \vDash Q$.

而由于 $m_2' \vDash Q$, 由归纳假设可知, $\forall n_0, (m_2', n_0 \vDash Q)$.

故该命题成立.

2) 若 $m_1, n' \vDash P$ 不成立, 则

存在 $m_1 \equiv_{n'} m_1$ 和 $m_2 \equiv_{n'-k} m_2$, 使得 $m_1 \xrightarrow{f'}_{k,n''} m_2$ 对任意 n''都成立!

$m_1 \equiv_{n'} m_1$ 和 $m_2 \equiv_{n'-k} m_2$ 显然成立.

由 f'的定义可知 $m_1 \xrightarrow{f}_{k,n} m_2$,

由函数 f 的性质 2 可以推出 $m_1 \xrightarrow{f}_{k,n''} m_2$ 对任意 $n'' \leqslant n$ 都成立.

因此, 要证 $m_1 \xrightarrow{f'}_{k,n''} m_2$ 对任意 n''都成立, 只需证明:

对任意 $n''>n$ 和 $n_0 \leqslant n''$, $k<n_0 \Rightarrow m_1, n_0 \vDash P \Rightarrow m_2, n_0 - k \vDash Q$.

对于任意 $n_0 \leqslant n'$, 由 f'的定义可知该命题成立;

对于任意 $n_0>n'$, 显然有 $m_1, n_0 \nvDash P$.

故该命题成立.

综上所述, 函数 f'具有函数的性质 3. □

利用函数 f 的性质和步进索引语义的性质, 按照 n'和 n 的大小关系分类讨论, 证明函数 f'具有性质 4.

性质 4. 对任意 m_1, m_2, n'和 $k<n'$, 若 $m_1 \xrightarrow{f'}_{k,n'} m_2$, 则:

对任意 $m_1' \equiv_{n'} m_1$, 都存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立.

证明:

对于 $n' \leqslant n$, 由函数 f'的定义可知, $m_1 \xrightarrow{f}_{k,n'} m_2$.

由函数 f 的性质 4 可以推出:

对于 $m_1' \equiv_{n'} m_1$, 存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f}_{k,n''} m_2'$ 对任意 n''都成立.

以下按照 $m_1', n \vDash P$ 是否成立, 分情况证明:

存在 $m_2'' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2''$ 对任意 n''都成立.

1) 若 $m_1', n \vDash P$ 成立, 由 $m, n \vDash l \mapsto \{P\}\{Q\}$, $m_1', n \vDash P$ 和 $m_1' \xrightarrow{f}_{k,n} m_2'$ 可以推出 $m_2', n-k \vDash Q$.

由归纳假设有 $m_2', n-k \vDash Q \Rightarrow m_2' \vDash \Diamond^{n-k} Q$,

即: 存在 $m_2'' \equiv_{n-k} m_2'$, 使得 $m_2'' \vDash Q$.

存在 $m_2'' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2''$ 对任意 n''都成立!

由 $n' \leqslant n$ 可以推出 $m_2'' \equiv_{n'-k} m_2'$, 又因为 $m_2' \equiv_{n'-k} m_2$, 故 $m_2'' \equiv_{n'-k} m_2$.

由函数 f 的性质 1、 $m_2'' \equiv_{n-k} m_2'$ 和 $m_1' \xrightarrow{f}_{k,n} m_2'$ 可以推得 $m_1' \xrightarrow{f}_{k,n} m_2''$.

再由函数 f 的性质 2, $m_1' \xrightarrow{f}_{k,n''} m_2''$ 对任意 $n''\leqslant n$ 都成立.

因此, 要证 $m_1' \xrightarrow{f'}_{k,n''} m_2''$ 对任意 n''都成立, 我们只需证明:

对任意 $n''>n$ 和 $n_0\leqslant n''$, $k<n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2'', n_0-k \vDash Q$.

而由于 $m_2'' \vDash Q$, 由归纳假设可知, $\forall n_0, (m_2'', n_0 \vDash Q)$.

故该命题成立.

2) 若 $m_1', n \vDash P$ 不成立, 则存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立!

$m_2' \equiv_{n'-k} m_2$ 显然成立.

由于 $m_1' \xrightarrow{f}_{k,n''} m_2'$ 对任意 n''都成立,

要证 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立, 只需证明:

对任意 $n''>n$ 和 $n_0\leqslant n''$, $k<n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2', n_0-k \vDash Q$.

对于任意 $n_0\geqslant n$, 显然有 $m_1', n_0 \nvDash P$, 故该命题成立;

对于任意 $n_0<n$, 由 $m, n \vDash l \mapsto \{P\}\{Q\}$和步进索引语义的性质 1 可以推得:

$$m, n_0 \vDash l \mapsto \{P\}\{Q\}.$$

由 $m, n_0 \vDash l \mapsto \{P\}\{Q\}$, $m_1', n_0 \vDash P$ 和 $m_1' \xrightarrow{m(l)=f}_{k,n_0} m_2'$ 可以推出:

$$m_2', n_0-k \vDash Q.$$

故该命题成立.

对于 $n'>n$, 按照 $m_1', n' \vDash P$ 是否成立, 分情况证明:

存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立.

1) 若 $m_1', n' \vDash P$ 成立, 则由步进索引语义的性质 2 可推得 $m_1, n' \vDash P$.

由 f'的定义可推得 $m_1 \xrightarrow{f}_{k,n} m_2$ 和 $m_2, n'-k \vDash Q$.

由归纳假设, $m_2, n'-k \vDash Q \Rightarrow m_2 \vDash \Diamond^{n'-k} Q$,

即存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_2' \vDash Q$.

存在 $m_2' \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立!

由 $n'>n$, 可以推出 $m_1' \equiv_n m_1$ 和 $m_2' \equiv_{n-k} m_2$.

由函数 f 的性质 1 可以推出 $m_1' \xrightarrow{f}_{k,n} m_2'$.

再由函数 f 的性质 2, 可以推出 $m_1' \xrightarrow{f}_{k,n''} m_2'$ 对任意 $n''\leqslant n$ 都成立.

因此, 要证 $m_1' \xrightarrow{f'}_{k,n''} m_2'$ 对任意 n''都成立, 只需证明:

对任意 $n''>n$ 和 $n_0\leqslant n''$, $k<n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2', n_0-k \vDash Q$.

而由于 $m_2' \vDash Q$, 由归纳假设可知, $\forall n_0, (m_2', n_0 \vDash Q)$.

故该命题成立.

2) 若 $m_1', n' \vDash P$ 不成立, 则:

存在 $m_2 \equiv_{n'-k} m_2$, 使得 $m_1' \xrightarrow{f'}_{k,n''} m_2$ 对任意 n''都成立!

$m_2 \equiv_{n'-k} m_2$ 显然成立.

由 f'的定义可推得 $m_1 \xrightarrow{f}_{k,n} m_2$.

由 $n'>n$, 可推得 $m_1' \equiv_n m_1$.

由函数 f 的性质 1 可以推出 $m_1' \xrightarrow{f}_{k,n} m_2$.

由函数 f 的性质 2, 可以推出 $m_1' \xrightarrow{f}_{k,n''} m_2$ 对任意 $n''\leqslant n$ 都成立.

因此, 要证 $m_1' \xrightarrow{f'}_{k,n''} m_2$ 对任意 n''都成立, 只需证明:

对任意 $n''>n$ 和 $n_0\leqslant n''$, $k<n_0 \Rightarrow m_1', n_0 \vDash P \Rightarrow m_2, n_0-k \vDash Q$.

对于任意 $n_0\leqslant n'$, 由 f'的定义和步进索引语义的性质 2 可推出该命题成立;

对于任意 $n_0>n'$, 显然有 m_1', $n_0 \nvDash P$.

故该命题成立.

综上所述, 函数 f'具有函数的性质 4. □

至此, 我们证明了引理 3 的证明过程中构造的 f'具有函数的全部性质, 是一个合法的函数.

以上 3 条引理(引理 1–引理 3)证明:

对任意程序状态 m, $(\forall n, (m, n\vDash X\Rightarrow m\vDash \Diamond^n X))$且$((\forall n, (m, n\vDash X))\Leftrightarrow m\vDash X)$,

其中, $X=l\mapsto\{P\}\{Q\}$.

接下来, 我们对断言 $l\mapsto\forall v, \{P(v)\}\{Q(v)\}$进行证明, 即证:

对任意程序状态 m, $(\forall n, (m, n\vDash X\Rightarrow m\vDash \Diamond^n X))$且$((\forall n, (m, n\vDash X))\Leftrightarrow m\vDash X)$,

其中, $X=l\mapsto\forall v, \{P(v)\}\{Q(v)\}$.

引理 4. 对任意程序状态 m:

$$(\forall n, (m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}))\Rightarrow m\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}.$$

即:

对任意 m, 若 (对任意n, v, $n'\leqslant n$, m_1, m_2和$k<n'$, 若m_1, $n'\vDash P(v)$且$m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则$m_2$, $n'-k\vDash Q(v)$),

则 (对任意v, m_1, m_2, k, 若$m_1\vDash P(v)$且($m_1\xrightarrow{m(l)}_{k,n} m_2$对任意$n$都成立), 则$m_2\vDash Q(v)$).

证明:

对任意 v, m_1, m_2 和 k, 由归纳假设可知:

$$m_1\vDash P(v)\Rightarrow\forall n, (m_1, n\vDash P(v)); (\forall n, (m_2, n\vDash Q(v)))\Rightarrow m_2\vDash Q(v).$$

由$\forall n, (m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\})$,

要证明 $m_2, n\vDash Q(v)$, 只需证明 $m_1, n+k\vDash P(v)$和 $m_1\xrightarrow{m(l)}_{k,n+k} m_2$.

由 $m_1\vDash P(v)\Rightarrow\forall n, (m_1, n\vDash P(v))$和 ($m_1\xrightarrow{m(l)}_{k,n} m_2$对任意$n$都成立) 可知,

上述条件显然对任意 n 都成立.

故有$\forall n, (m_2, n\vDash Q(v))$.

因此, 该引理成立. □

引理 5. 对任意程序状态 m:

$$m\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}\Rightarrow\forall n, (m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}).$$

即:

对任意 m, 若 (对任意v, m_1, m_2, k, 若$m_1\vDash P(v)$且($m_1\xrightarrow{m(l)}_{k,n} m_2$对任意$n$都成立), 则$m_2\vDash Q(v)$),

则 (对任意n, v, $n'\leqslant n$, m_1, m_2和$k<n'$, 若m_1, $n'\vDash P(v)$且$m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则$m_2$, $n'-k\vDash Q(v)$).

证明:

对任意 v, m_1, m_2, n'和 k, 由归纳假设可知:

$$m_1, n'\vDash P(v)\Rightarrow m_1\vDash\Diamond^{n'}P(v).$$

即: 存在 $m_1'\equiv_{n'} m_1$, 使得 $m_1'\vDash P(v)$.

由函数性质 4 和 $m_1\xrightarrow{m(l)}_{k,n'} m_2$, 可以推出:

存在 $m_2'\equiv_{n'-k} m_2$, 使得 $m_1'\xrightarrow{m(l)}_{k,n_0} m_2'$ 对任意 n_0 都成立.

由 $m\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}$, $m_1'\vDash P(v)$ 和 $m_1'\xrightarrow{m(l)}_{k,n_0} m_2'$ 对任意 n_0 都成立,

可推得 $m_2'\vDash Q(v)$.

再由归纳假设可知, $m_2'\vDash Q(v)\Rightarrow\forall n_0, (m_2', n_0\vDash Q(v))$.

故有 m_2', $n'-k\vDash Q(v)$. 由步进索引语义的性质 2 可以推出, m_2, $n'-k\vDash Q(v)$.

因此, 该引理成立. □

引理 6. 对任意程序状态 m 和步数 n:

$$m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}\Rightarrow m\vDash\Diamond^n l\mapsto\forall v, \{P(v)\}\{Q(v)\}.$$

即:

对任意 m 和 n, 若 (对任意v, $n'\leqslant n$, m_1, m_2和$k<n'$, 若$m_1, n'\vDash P(v)$且$m_1\xrightarrow{m(l)}_{k,n'} m_2$, 则$m_2, n'-k\vDash Q(v)$), 则存在 $m'\equiv_n m$, 使得 (对任意v, m_1, m_2, k, 若$m_1\vDash P(v)$且($m_1\xrightarrow{m'(l)}_{k,n} m_2$对任意$n$都成立), 则$m_2\vDash Q(v)$).

证明:

对任意 m, n, 该引理的证明本质上是要找到与 $f=m(l)$在 n 步内无法区分的函数 f'. 取 f'为

$$m_1\xrightarrow{f'}_{k,n'} m_2 \text{ 当且仅当对任意 } v \text{ 都有 } n'\leqslant n\Rightarrow m_1\xrightarrow{f}_{k,n'} m_2, \text{ 且}$$

$$n'>n\Rightarrow(m_1\xrightarrow{f}_{k,n} m_2 \text{且} \forall n''\leqslant n', k<n''\Rightarrow m_1, n''\vDash P(v)\Rightarrow m_2, n''-k\vDash Q(v)).$$

显然有, f'和 f 在 n 步内无法区分.

令 $m'(l)=f'$, 且 m'在其他地址上与 m 保持一致, 则有 $m'\equiv_n m$.

以下证明:

(对任意v, m_1, m_2, k, 若$m_1\vDash P(v)$且($m_1\xrightarrow{m'(l)}_{k,n} m_2$对任意$n$都成立), 则$m_2\vDash Q(v)$).

对任意 v, m_1, m_2 和 k, 由归纳假设可知:

$$m_1\vDash P(v)\Rightarrow\forall n, (m_1, n\vDash P(v)); (\forall n, (m_2, n\vDash Q(v)))\Rightarrow m_2\vDash Q(v).$$

要证明 $m_2\vDash Q(v)$, 只需证明对任意 n_0, 都有 $m_2, n_0\vDash Q(v)$.

由 ($m_1\xrightarrow{m'(l)}_{k,n} m_2$对任意$n$都成立) 可知, $m_1\xrightarrow{m'(l)}_{k,n_0+k} m_2$.

以下按 n_0+k 和 n 的大小关系, 分情况证明.

1) 当 $n_0+k\leqslant n$ 时, 由 f'的定义可知, $m_1\xrightarrow{f=m(l)}_{k,n_0+k} m_2$.

因此, 对于 v, $n_0+k\leqslant n$, m_1, m_2 和 $k<n_0+k$:

由 $m, n\vDash l\mapsto\forall v, \{P(v)\}\{Q(v)\}$, $m_1, n_0+k\vDash P(v)$和 $m_1\xrightarrow{f=m(l)}_{k,n_0+k} m_2$ 可以推出 $m_2, n_0+k-k\vDash Q(v)$,

即 $m_2, n_0\vDash Q(v)$.

2) 当 $n_0+k>n$ 时, 由 f'的定义可知:

对任意 v 和 $n''\leqslant n_0+k$, 都有:

$$k<n''\Rightarrow m_1, n''\vDash P(v)\Rightarrow m_2, n''\vDash Q(v).$$

取 $n''=n_0+k$, 则显然有 $k<n_0+k$ 和 $m_1, n_0+k\vDash P(v)$, 故

$$m_2, n_0+k-k\vDash Q(v).$$

即 $m_2, n_0\vDash Q(v)$.

综上所述, 该引理成立. □

同样地, 我们找到的函数 f'必须符合函数的性质. f'符合这些性质的证明与引理 3 中的证明类似, 其中, 函数的性质 3 和性质 4 的证明用到了关于断言 $l\mapsto\forall v, \{P(v)\}\{Q(v)\}$的约束条件, 即:

$$\text{对任意 } m, n_1, n_2, v_1 \text{ 和 } v_2, \text{ 若 } m, n_1\vDash P(v_1) \text{且 } m, n_2\vDash P(v_2), \text{ 则 } v_1=v_2.$$

以上 3 条引理(引理 4–引理 6)证明:

$$\text{对任意程序状态 } m, (\forall n, (m, n\vDash X\Rightarrow m\vDash\Diamond^n X))\text{且}((\forall n, (m, n\vDash X))\Leftrightarrow m\vDash X).$$

其中, $X=l\mapsto\forall v, \{P(v)\}\{Q(v)\}$.

为了证明“对任意 m, $(\forall n, (m, n\vDash P\vee Q))\Rightarrow m\vDash P\vee Q$”, 我们需要证明如下引理.

引理 7. $(\forall n, (m, n\vDash P\vee Q))\Rightarrow(\forall n, (m, n\vDash P))$或$(\forall n, (m, n\vDash Q))$.

证明:

使用反证法证明.

假设“$(\forall n, (m, n\vDash P))$或$(\forall n, (m, n\vDash Q))$”不成立,

即$(\forall n, (m, n\vDash P))$和$(\forall n, (m, n\vDash Q))$都不成立.

由"$(\forall n, (m, n\vDash P))$不成立"可以推出: 存在 n_1 使得 $m, n_1\nvDash P$.

由步进索引语义的性质 1 可推得: 对所有 $n_1'\geqslant n_1$, 都有 $m, n_1'\nvDash P$.

同理, 存在 n_2, 使得对所有 $n_2'\geqslant n_2$, 都有 $m, n_2'\nvDash Q$.

显然, 对于 n_1 和 n_2 中较大的一个, 不妨设为 n_1, 有 $m, n_1\nvDash P$ 且 $m, n_1\nvDash Q$.

根据定义, 这与$(\forall n, (m, n\vDash P\vee Q))$即$(\forall n, (m, n\vDash P$ 或 $m, n\vDash Q))$矛盾!

因此, 该引理成立. □

接下来利用上述引理, 使用结构归纳法证明语义等价性尚未证明的 3 个分支, 即, 对任意合法的断言 P 和程序状态 m:

$$(\forall n, (m, n\vDash P\Rightarrow m\vDash \diamond^n P))且((\forall n, (m, n\vDash P))\Leftrightarrow m\vDash P).$$

1. 对于断言 $l\mapsto v$, 证明十分显然;
2. 对于断言 $l\mapsto\{P\}\{Q\}$, 证明见引理 1–引理 3;
3. 对于断言 $P*Q$, 注意, 根据前文的约束条件(定义 3), m 的划分方式是唯一的, 即:
   存在正整数 N, 使得对任意 m_1, m_2, m_1', m_2' 和 $n\geqslant N$, 都有:
   $$若(m_1,m_2)和\ (m_1',m_2')\ 都是\ m\ 的划分,$$
   $$且\ m_1, n\vDash P,\ \ m_1', n\vDash P,\ m_2, n\vDash Q,\ m_2', n\vDash Q,\ \ 则\ m_1=m_1'\ 且\ m_2=m_2'.$$
   在这一约束条件下, 3 个分支均容易由归纳假设证明;
4. 对于断言 $P\vee Q$, $(\forall n, (m, n\vDash P\vee Q))\Rightarrow m\vDash P\vee Q$ 的证明需要借助引理 7:
   $$(\forall n, (m, n\vDash P\vee Q))\Rightarrow(\forall n, (m, n\vDash P))\Rightarrow或(\forall n, (m, n\vDash Q)).$$
   根据归纳假设, $(\forall n, (m, n\vDash P))\Rightarrow m\vDash P$, $(\forall n, (m, n\vDash Q))\Rightarrow m\vDash Q$,
   故有 $m\vDash P$ 或 $m\vDash Q$, 即 $m\vDash P\vee Q$.
   另外两个分支容易由归纳假设证明;
5. 对于断言 $A\wedge P$, 注意到命题 A 成立与否与取何种语义无关, 3 个分支均容易由归纳假设证明;
6. 对于断言$\exists v, P(v)$, 注意, 根据前文的约束条件(定义 3), 存在的 v 是唯一的, 即:
   存在正整数 N, 使得对任意 m, x, y 和 $n\geqslant N$, 都有:
   $$若\ m, n\vDash P(x)且\ m, n\vDash P(y),\ 则\ x=y.$$
   在这一约束条件下, 3 个分支均容易由归纳假设证明;
7. 对于断言 $l\mapsto\forall v, \{P(v)\}\{Q(v)\}$, 证明见引理 4–引理 6.

4 证明的形式化

为了确保上述理论的正确性和便于以后将这些理论应用于形式化证明程序的正确性, 本文使用交互式定理证明器 Coq[16]将上述工作进行了形式化. 以下列举其中重要的定义和定理是如何形式化的.

4.1 程序状态模型的形式化

```
1  Parameter M: Type.
2  Parameter feM: nat→M→M→Prop.
3  Definition preFunc: Type:=(M*M)→nat→nat→Prop.
4  Definition is_func(f: preFunc): Prop:=(forall m1 m2  m1' m2'  k n',
5    (feM n m1  m1'→feM(n−k) m2  m2'→f(m1,m2) k n↔ f (m1',m2')  k n) (*Func_NDI*)∧
6    (f(m1,m2) k n→n≥n'→f(m1,m2) k n') (*Func_Downwards_closed*)∧
7    (k<n→f(m1,m2) k n→exists m11 m22,
8      (feM n m1 m11∧feM(n−k) m2 m22∧(forall nn, f(m11,m22) k nn))) (*Func_Prop*)∧
```

```
9     (k<n→f(m1,m2) k n→(forall m11, feM n m1 m11→(exists m22,
10        feM(n−k) m2 m22∧forall nn, f(m11,m22) k nn))) (*Func_Property*)
11 ).
12 Definition Func: Type:=sig is_func.
13 Definition FM: Type:=nat→option(nat+Func).
14 Parameter i1: M→FM.
15 Parameter i2: FM→M.
16 Axiom i1_i2: forall m, i1(i2 m)=m.
17 Axiom i2_i1: forall m, i2(i1 m)=m.
```

其中, M 和 FM 是程序状态的类型, $feM\ n\ m_1\ m_2$ 表示两个程序状态 m_1 和 m_2 在 n 步内无法区分(符号表示即 $m_1\equiv_n m_2$). $Func$ 是函数的类型, $f\ (m_1,m_2)\ k\ n$ 表示 $m_1 \xrightarrow{f}_{k,n} m_2$. $Func$ 类型中的函数 f 都满足 is_func 中定义的性质, 即第 3.3 节中提及的函数应该满足的 4 条性质. 如引言中所述, $FM=Loc\rightharpoonup(Value+Func)$而 $Func=(M\times M)\to\mathbb{N}\to\mathbb{N}\to Prop$, 方程 $M=FM$(即 $M=\mathbb{N}\rightharpoonup(\mathbb{N}+((M\times M)\to\mathbb{N}\to\mathbb{N}\to Prop))$)一般不可解. 然而, 函数的 4 条性质保证了方程中几个映射的特殊性质, 将上述方程转化为

$$M=\mathbb{N}\rightharpoonup(\mathbb{N}+((M\times M)\to_{NDI}\mathbb{N}\to\mathbb{N}_{\downarrow}\to Prop)),$$

其中, "$\mathbb{N}_{\downarrow}$"表示在自然数集上向下闭合). 根据 America 和 Rutten 的研究[8], 该方程在完备度量空间的范畴中存在唯一解, 即 M 和 FM 之间存在一一映射关系. 以下简单说明这一问题.

归纳定义完备度量空间的范畴上的函子(functor)集合 $F\in Func(F:\mathcal{C}\to\mathcal{C})$:

$$F\triangleq F_M\mid id^{\varepsilon}\mid F_1\to F_2\mid F_1\to^1 F_2\mid F_1\cup F_2\mid F_1\times F_2\mid F_1\circ F_2\mid \mathcal{P}_{cl}(F),$$

其中, M 是任意的完备度量空间, $\varepsilon>0$. 可以证明, 集合 $Func$ 中的元素都是完备度量空间的范畴上定义完善的函子. 归纳定义 F 的收缩系数(contraction coefficient) $c(F)$.

1. 若 $F=F_M$, 则 $c(F)=0$;
2. 若 $F=id^{\varepsilon}$, 则 $c(F)=\varepsilon$;
3. 若 $F=F_1\to F_2$, 则 $c(F)=\max\{\infty\cdot c(F_1),c(F_2)\}$;
4. 若 $F=F_1\to^1 F_2$, 则 $c(F)=c(F_1)+c(F_2)$;
5. 若 $F=F_1\cup F_2$, 则 $c(F)=\max\{c(F_1),c(F_2)\}$;
6. 若 $F=F_1\times F_2$, 则 $c(F)=\max\{c(F_1),c(F_2)\}$;
7. 若 $F=F_1\circ F_2$, 则 $c(F)=c(F_1)\cdot c(F_2)$;
8. 若 $F=\mathcal{P}_{cl}(F)$, 则 $c(F)=c(F_1)$.

其中, 关于∞的乘法定义为$\infty\cdot 0=0\cdot\infty=0$, $\forall c>0$, $\infty\cdot c=c\cdot\infty=\infty$. America 和 Rutten 证明了, 集合 $Func$ 中的每个元素 F 都满足如下条件: 当 $c(F)<1$ 时, 方程 $\mathcal{C}=F\mathcal{C}$ 有(同构意义下的)唯一解[1]. 注意到, 本文中的程序状态模型 M 满足方程:

$$M=\mathbb{N}\rightharpoonup(\mathbb{N}+((M\times M)\to_{NDI}\mathbb{N}\to\mathbb{N}_{\downarrow}\to Prop)).$$

其中的函子 $F=F_{\mathbb{N}}\to\frac{1}{2}(F_{\mathbb{N}}\cup((id^1\times id^1)\to^1 F_{\mathbb{N}}\to\mathbb{N}_{\downarrow}\to Prop))$, 类似于:

$$F'=F_{\mathbb{N}}\to\frac{1}{2}(F_{\mathbb{N}}\cup\mathcal{P}_{cl}((id^1\times id^1)\to^1 F_{\mathbb{N}})),\quad c(F)<1.$$

故该方程有唯一解.

Iris 项目[12]已对该模型进行了形式化, 因此我们直接以此为基础给出进一步证明.

定义 $Func\_EQ\ n\ f_1\ f_2$ 表示两个函数 f_1 和 f_2 在 n 步内无法区分.

```
1  Definition Func_EQ(n: nat) (f1 f2: Func): Prop:=
2     forall m1 m2 k, k<n→(f1(m1,m2) k n↔f2(m1,m2) k n).
```

定义程序状态无法区分的关系 *feM*(即第 3.3 节中的定义 4).

```
1  Axiom feM_imply_EQ: forall m1 m2 n,
2    feM (S n) m1 m2↔
3      forall x,
4        (i1 m1 x=None∧i1 m2 x=None)∨
5        (exists l, i1 m1 x=Some(inl l)∧i1 m2 x=Some(inl l))∨
6        (exists f1 f2,
7          i1 m1 x=Some(inr f1)∧
8          i1 m2 x=Some(inr f2)∧
9          Func_EQ n f1 f2).
```

4.2 断言及其语义的定义

定义 *join_fm* m_1 m_2 m_3 表示(m_1,m_2)是 m_3 的一个划分.

```
1  Definition join_fm(m1 m2 m3: FM): Prop:=forall l,
2    (m1 l=None∧m2 l=None∧m3 l=None)∨
3    (exists v, m1 l=None∧m2 l=Some v∧m3 l=Some v)∨
4    (exists v, m1 l=Some v∧m2 l=None∧m3 l=Some v).
```

定义 *diam n P m* 表示 $m\vDash\Diamond^n P$.

```
1  Definition diam(n: nat) (P: M→Prop) (m: M): Prop:=exists m', feM n m m'∧P m'.
```

断言被归纳定义为(对应第 2.2 节中的定义).

```
1  Definition var_lang: Type:=nat.
2  Inductive term: Type:=
3    |Var(v: var_lang)
4    |Const(l: nat).
5  Inductive lang: Type:=
6    |MapstoV(l v: term)
7    |MapstoF(l: term) (P Q: lang)
8    |MapstoF_forall(l: term) (v: var_lang) (P Q: lang)
9    |Or (P Q: lang)
10   |And (P: Prop) (Q: lang)
11   |Sepcon (P Q: lang)
12   |Exists(v: var_lang) (P: lang).
```

使用 *interp* 存储变量的值, 使用 *interp_update* 实现上文中提及的带有参数的断言的形式化表示.

```
1  Definition interp: Type:=var_lang→nat.
2  Definition interp_update(i: interp) (v: var_lang) (t: term): interp:=
3    fun x: var_lang⇒if beq_nat x v then (denote_term i t) else i x.
4  Definition denote_term(i: interp) (t: term): nat:=
5    match t with
6    |Var v⇒i v
7    |Const l⇒l
8    end.
```

定义断言的非步进索引语义(对应第 3.1 节中的定义).

```
1  Fixpoint nonidx_denote(i: interp) (P: lang): M→Prop:=
```

```
2    match P with
3    |MapstoV l v⇒fun m⇒
4      i1 m (denote_term i l)=Some(inl (denote_term i v))∧
5      forall l', l'<>denote_term i l→i1 m l'=None
6    |MapstoF l P Q⇒fun m⇒
7      (forall l', l'<>denote_term i l→i1 m l'=None)∧
8      exists f,
9        i1 m (denote_term i l)=Some(inr f)∧
10       (forall m1 m2 k,
11         nonidx_denote i P m1→
12         (forall n, f(m1,m2) k n)→
13         nonidx_denote i Q m2)
14   |Or P Q⇒fun m⇒nonidx_denote i P m∨nonidx_denote i Q m
15   |And P Q⇒fun m⇒P∧nonidx_denote i Q m
16   |Sepcon P Q⇒fun m⇒
17     exists m1 m2,
18       join_m m1 m2 m∧
19       nonidx_denote i P m1∧
20       nonidx_denote i Q m2
21   |Exists v P⇒fun m⇒exists t, nonidx_denote(interp_update i v t) P m
22   |MapstoF_forall l v P Q⇒fun m⇒forall t,
23     (forall l', l'<>denote_term i l→i1 m l'=None)∧
24     exists f,
25       i1 m (denote_term i l)=Some(inr f)∧
26       (forall m1 m2 k,
27         nonidx_denote(interp_update i v t) P m1→
28         (forall n, f(m1,m2) k n)→
29         nonidx_denote(interp_update i v t) Q m2)
30   end.
```

定义断言的步进索引语义(对应第 3.1 节中的定义).

```
1  Fixpoint index_denote_aux(i: interp) (P: lang): M→nat→Prop:=
2    match P with
3    |MapstoV l v⇒
4      fun m n⇒match n with|0⇒True|S_⇒
5        i1 m (denote_term i l)=Some (inl (denote_term i v))∧
6        forall l', l'<>denote_term i l→i1 m l'=None end
7    | MapstoF l P Q⇒
8      fun m n⇒match n with|0⇒True|S n'⇒
9        (forall l', l'<>denote_term i l→i1 m l'=None)∧
10       forall n0, n0≤n'→
11         (exists f,
12           i1 m(denote_term i l)=Some(inr f)∧
```

```
13              (forall m1 m2 k, k<n0→
14                 index_denote_aux i P m1 n0→
15                 f(m1,m2) k n0→
16                 index_denote_aux i Q m2(n0–k))) end
17   |Or P Q⇒fun m n⇒index_denote_aux i P m n∨index_denote_aux i Q m n
18   |And P Q⇒fun m n⇒P∧index_denote_aux i Q m n
19   |Sepcon P Q⇒fun m n⇒
20      exists m1 m2,
21        join_m m1 m2 m∧
22        index_denote_aux i P m1 n∧
23        index_denote_aux i Q m2 n
24   |Exists v P⇒fun m n⇒exists t, index_denote_aux(interp_update i v t) P m n
25   |MapstoF_forall l v P Q⇒
26      fun m n⇒match n with|0⇒True|S n′⇒forall t,
27        (forall l′, l′<>denote_term i l→i1 m l′=None)∧
28        forall n0, n0⩽n′→
29          (exists f,
30             i1 m(denote_term i l)=Some (inr f)∧
31             (forall m1 m2 k, k<n0→
32                index_denote_aux(interp_update i v t) P m1 n0→
33                f(m1,m2) k n0→
34                index_denote_aux(interp_update i v t) Q m2 (n0–k))) end
35   end.
```

定义对断言的约束条件(即第 2.2 节中的定义 3).

```
1   Fixpoint legal(P: lang): Prop:=
2      match P with
3      |MapstoV l v⇒True
4      |MapstoF l P Q⇒legal P∧legal Q
5      |Or P Q⇒legal P∧legal Q
6      |And P Q⇒legal Q
7      |Sepcon P Q⇒
8         legal P∧legal Q∧
9         exists N, forall n, n⩾N→forall m1 m2  m1′ m2′  m i,
10           join_m m1 m2 m→
11           join_m  m1′ m2′  m→
12           index_denote_aux i P m1 N→
13           index_denote_aux i P  m1′  n→
14           index_denote_aux i Q m2 N→
15           index_denote_aux i Q  m2′  n→
16            m1 = m1′ ∧ m2 = m2′
17     |Exists v P ⇒legal P∧
18        exists N, forall n, n⩾N→forall m i t1 t2,
19           index_denote_aux(interp_update i v t1) P m N→
```

```
        index_denote_aux(interp_update i v t2) P m n→
        t1=t2
  |MapstoF_forall l v P Q⇒
    legal P∧legal Q∧
    forall n1 n2, n1>0→n2>0→forall m i t1 t2,
      (index_denote_aux(interp_update i v t1) P m n1→
      index_denote_aux(interp_update i v t2) P m n2→
      t1=t2)
  end.
```

步进索引语义满足两条性质的定理(对应第 3.3 节中提及的步进索引语义应满足的两条性质).

```
Definition inner_NE(P: M→nat→Prop): Prop:=
  (forall m n1 n2, P m n1→n1≥n2→P m n2)∧
  (forall m1 m2 n, feM n m1 m2→P m1 n→P m2 n).
Theorem index_denote_inner_NE: forall i P,
  inner_NE(index_denote_aux i P).
```

其中, *P m n* 表示 *m*, *n*⊨*P*.

4.3 两种语义的等价性证明

代码中, 用字母 N 表示非步进索引语义, I 表示步进索引语义, D 表示带有◇符号的近似语义. 要证明的 6 条等价性.

```
(*N for non-indexed, I for indexed, and D for diamond indexed*)
Definition DeriveN2D(P: lang): Prop:=(*non-indexed logic imply diamond indexed logic*)
  forall m i, nonidx_denote i P m→forall n, diam n(nonidx_denote i P) m.
Definition DeriveN2I(P: lang): Prop:=(*non-indexed logic imply indexed logic*)
  forall m i, nonidx_denote i P m→forall n, n>0→index_denote_aux i P m n.
Definition DeriveI2D(P: lang): Prop:=(*indexed logic imply diamond indexed logic*)
  forall m i n, n>0→index_denote_aux i P m n→diam n(nonidx_denote i P) m.
Definition DeriveI2N(P: lang): Prop:=(*indexed logic imply non-indexed logic*)
  forall m i, (forall n, n>0→index_denote_aux i P m n)→nonidx_denote i P m.
Definition DeriveD2N(P: lang): Prop:=(*diamond indexed logic imply non-indexed logic*)
  forall m i, (forall n, n>0→diam n(nonidx_denote i P) m)→nonidx_denote i P m.
Definition DeriveD2I(P: lang): Prop:=(*diamond indexed logic imply indexed logic*)
  forall m i n, n>0→diam n(nonidx_denote i P) m→index_denote_aux i P m n.
Theorem fully_equal: forall P, legal P→
  DeriveN2D P∧DeriveN2I P∧DeriveI2D P∧DeriveI2N P∧DeriveD2N P∧DeriveD2I P.
```

5 总 结

本文基于 America-Rutten 定理, 定义了支持函数指针的程序状态模型. 以在简单的步进索引模型下使用简洁清晰的经典命题逻辑推理方法为目标, 我们研究了步进索引逻辑和非步进索引逻辑的关系, 但发现两种逻辑在一些不常见的断言上并不等价. 通过对一些实际的程序验证工作的研究, 我们发掘了实际使用的断言的共有特征. 之前的很多程序验证工作基于经验和习惯使用了拥有这些共有特征的断言, 本文则从理论上找到了这些共有特征和两种逻辑等价性之间的关联, 基于这些特征定义了断言的约束条件, 并说明了这些约束条件的合理性. 本文证明了满足这些约束条件的断言的步进索引语义和非步进索引语义的等价性. 在保证涉

及的断言都满足这些约束条件的情况下, 我们可以避免使用高深困难的基于区域理论方法的程序状态模型和模糊繁琐的步进索引逻辑推理方法. 最后, 我们给出了以上内容的形式化定义和证明, 以保证理论的正确性和便于以后将这些理论应用于实际的程序验证工具中.

在本文研究成果的基础上, 未来可能的发展方向包括: (1) 以上述不针对程序语言的理论为基础, 设计开发针对具体程序语言的验证工具; (2) 研究形如$\exists r, P(r)$或$l \mapsto \forall r, \{P(r)\}\{Q(r)\}$ (其中, r 是一个断言)的断言是否也有两种语义的等价性, 如果有, 需要什么约束条件.

References:

[1] Floyd RW. Assigning meanings to programs. Proc. of the American Mathematical Society Symposia on Applied Mathematics, 1967, 19: 19−31.

[2] Hoare CAR. An axiomatic basis for computer programming. Communications of the ACM, 1969, 12(10): 576−580.

[3] O'Hearn PW, Reynolds JC, Yang H. Local reasoning about programs that alter data structures. In: Proc. of the 15th Int'l Workshop on Computer Science Logic (CSL 2001), the 10th Annual Conf. of the EACSL. Paris: Springer-Verlag, 2001.

[4] Reynolds JC. Separation logic: A logic for shared mutable data structures. In: Proc. of the 17th Annual IEEE Symp. on Logic in Computer Science. IEEE, 2002.

[5] Huang DM, Zeng QK. Program verification techniques based on separation logic. Ruan Jian Xue Bao/Journal of Software, 2009, 20(8): 2051−2061 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3636.htm [doi: 10.3724/SP.J.1001.2009.03636]

[6] Svendsen K, Birkedal L, Parkinson AM. Modular reasoning about separation of concurrent data structures. In: Proc. of the European Conf. on Programming Language & Systems. Springer-Verlag, 2013.

[7] Svendsen K, Birkedal L. Impredicative concurrent abstract predicates. In: Proc. of the European Conf. on Programming Language & Systems. Springer-Verlag, 2014.

[8] America P, Rutten J. Solving reflexive domain equations in a category of complete metric spaces. Journal of Computer and System Sciences, 1989, 39(3): 343−375.

[9] Birkedal L, Støvring K, Thamsborg J. The category-theoretic solution of recursive metric-space equations. TCS, 2010, 411(47): 4102−4122.

[10] Appel AW, Dockins R, Hobor A, Beringer L, Dodds J, Stewart G, Blazy S, Leroy X. Program Logics for Certified Compilers. Cambridge University Press, 2014.

[11] Birkedal L, Reus B, Schwinghammer J, Støvring K, Thamsborg J, Yang HS. Step-indexed Kripke models over recursive worlds. ACM SIGPLAN Notices, 2011, 46(1): 119−132.

[12] Jung R, Krebbers R, Jourdan JH, Bizjak A, Birkedal L, Dreyer D. Iris from the ground up. In: Proc. of the Submitted to JFP. 2017.

[13] Appel AW. Verified software toolchain. In: Proc. of the European Symp. on Programming. Berlin, Heidelberg: Springer, 2011.

[14] de Moura L, Bjørner N. Z3: An efficient SMT solver. In: Proc. of the Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. 2008.

[15] Barrett C, Conway CL, Deters M, Hadarean L, Jovanovic D, King T, Reynolds A, Tinelli C. CVC4. In: Proc. of the 23rd Int'l Conf. on Computer Aided Verification (CAV 2011). Springer-Verlag, 2011. 171−177.

[16] The Coq proof assistant. Version 8.13.2. 2021. https://coq.inria.fr

[17] Reynolds JC. An introduction to separation logic. 2011. http://www.cs.cmu.edu/afs/cs.cmu.edu/project/fox-19/member/jcr/www15818As2011/notes2.pdf

附中文参考文献:

[5] 黄达明, 曾庆凯. 基于分离逻辑的程序验证技术. 软件学报, 2009, 20(8): 2051−2061. http://www.jos.org.cn/1000-9825/3636.htm [doi: 10.3724/SP.J.1001.2009.03636]

郭昊(1997－), 男, 硕士, 主要研究领域为基于定理证明的程序验证.

曹钦翔(1990－), 男, 博士, 副教授, 博士生导师, CCF 专业会员, 主要研究领域为基于定理证明的程序验证, 程序逻辑.