点集拓扑学之杨忠道定理的一个机械化证明

献给张景中、杨路教授 85 华诞

曾振柄¹, 王建林^{2*}, 杨争峰³, 小林英恒⁴

1. 上海大学数学系, 上海 200444;

2. 河南大学计算机与信息工程学院, 开封 475004;

3. 华东师范大学上海市高可信计算重点实验室, 上海 200062;

4. Department of Mathematics, College of Science and Technology, Nihon University, Tokyo 101-8308, Japan

E-mail: zbzeng@shu.edu.cn, jlwang@henu.edu.cn, zfyang@sei.ecnu.edu.cn, hdt.coba@gmail.com

收稿日期: 2019-12-09; 接受日期: 2020-06-08; 网络出版日期: 2020-12-31; * 通信作者

国家自然科学基金 (批准号: 11471209 和 61772203) 资助项目

摘要 本文给出一种用高阶逻辑自动证明语言 Isabelle 在计算机中表示拓扑空间中开集、闭集、邻域和导集等基本概念的方法, 在此基础上证明点集拓扑学中著名的杨忠道定理, 即一拓扑空间的任意单点集的导集为闭集, 则其任意子集的导集亦为闭集.

关键词 拓扑空间 开集 闭集 导集 杨忠道定理 机器证明

MSC (2010) 主题分类 54B05, 68T15

1 引言

2000 年前后在北京召开的一次数学机械化会议上, 吴文俊指出, 数学机械化有待深入研究的两个领域, 一个是不等式定理的机器证明, 另一个是拓扑学的数学机械化. 之后二十年来, 前一个方面的研究, 经过杨路和张景中等学者的努力探索, 已经在基于柱形代数分解的证明 (参见文献 [1–3])、奇点分隔与区域分解结合的证明 (参见文献 [4]) 以及逐次差分代换的启发式证明 (参见文献 [5]) 等方向取得一定成功. 后一个问题, 过去二十年的进展则相对较少.

本文的目的是给出点集拓扑学的一个为人熟知的基本结果——杨忠道定理——的机器证明. 杨忠道定理的表述是, 拓扑空间 (X, T) 中, 每个子集的导集为闭集的充要条件是每一个单点集的导集为闭集. 这个定理是杨忠道 1950 年证明的. 最早出现在 Kelley^[6] 的书 *General Topology*. 杨忠道曾经证明过好几个重要的定理, 如 Dyson 猜测和 Blaschke 猜测等. 其中在国内知道的人最多的, 要数拓扑空间导集的这个结论. 20 世纪 80 年代出版的熊金城《点集拓扑讲义》^[7] 和蒲保明《拓扑学》^[8] 等著作也包含这个定理. 杨忠道定理把拓扑空间的单点集的一个性质, 推广到任意子集, 是一个令人惊讶的发现, 迄今

英文引用格式: Zeng Z B, Wang J L, Yang Z F, et al. A mechanical proof of the C. T. Yang's Theorem related to a property of derived sets in general topology (in Chinese). *Sci Sin Math*, 2021, 51: 257–288, doi: 10.1360/SSM-2019-0301

为止, 对于这个结论的各种证明, 也都包含一定技巧. 由此, 杨忠道定理在国内点集拓扑学的学习者中有广泛影响 (参见文献 [9]).

本文将给出用计算机证明杨忠道定理的一个方法, 即在计算机上完成杨忠道定理的表达和证明. 为了实现这个定理在计算机上的机械化证明, 需要在机器内部建立拓扑空间、导集和闭集等拓扑学中最基本概念的最适当的符号表示, 从而利用规则匹配和重写等机器证明手段完成定理的自动化证明. 在这个意义上, 杨忠道定理的机械化证明的完成, 标志着点集拓扑学机械化的一个可行开端.

下面先简要介绍数学机械化发展的历程, 以帮助不了解用计算机证明数学定理的读者更好地阅读本文. 由于本文的拓扑学机械化思路, 相比已经发展的较为成熟的初等代数、初等几何以及实代数几何、常微分方程、偏微分方程、组合数学等数学学科的机械化方法, 在数学概念的表述、推理证明方面有较大差别, 因此, 下面的简单回顾, 也有助于熟悉数学机械化但还不了解拓扑学如何机械化的读者, 对拓扑学定理的机器证明思想和实现方法, 获得一个概要的认识.

20 世纪 40 年代计算机建成伊始, 人们除了利用计算机的强大处理能力进行爆炸模拟和密码破译等这些与军事用途有关的研究, 也马上尝试用计算机模拟人类的智力活动, 如下棋、语言翻译和数学推理等. 设想在计算机上证明数学中某一类或者某些定理, 其发端可追溯到 1956 年在美国达特茅斯举办的著名活动 “Summer Research Project on Artificial Intelligence” 期间 Marvin Minsky 草创的一个几何定理机器证明方案. 这一设想在会后不久就由 Herb Gelernter 和 Nathan Rochester 给出实现, 使之能在当时的计算机上证明美国中学平面几何课本上的一些定理, 该方法用 “语义表” 可将机器进行 3 步推理所产生的上亿个搜索分支约简到只有二十几种 (参见文献 [10, 11]). 由此, 几何定理的机器证明成为人工智能研究的一部分. 同一时期机械化数学的先驱工作, 还有王浩为 IBM704 计算机编写的三个程序, 其中第一个程序, 可以在 37 分钟 (其中的 12/13 时间用于读入和输出) 证明罗素和怀特海《数学原理》前五章的全部命题; 第二个程序指导计算机构造命题, 并从中选择出 “非平凡” 的命题; 第三个程序处理关于谓词逻辑等式命题, 可在一小时以内证明《数学原理》第五章到第十章包含的命题的约 85% (参见文献 [12]). Slagle^[13]、Moses^[14] 和 Patterson^[15] 20 世纪 60 年代在自动积分的工作, 以及 Berlekamp、Yun、Wang、Zassenhaus、Lenstra 和 Kaltofen 等在多项式因式分解算法作出的突破 (参见文献 [16]), 实现了微积分和多项式在计算机上的快速计算, 也使得借助代数演算进行数学定理机器证明的设想成为可能. 吴文俊^[17-19] 在 20 世纪 70 年代开展的初等几何定理机器证明的工作, 体现了中国传统数学在求解问题时, 兼顾算法的实用性与问题的一般性的思想, 成为将一门数学学科系统化地进行机械化的范例. 与此相比, 虽然早在 1951 年 Tarski^[20] 就证明了初等几何和初等代数的命题的正确性, 可以用机械化的方法判定, 但其算法即使在现今的计算机上, 仍然难以完全实现. 随着 20 世纪 80 年代以来计算机硬件系统的快速发展和个人计算机的普及, 吴文俊在几何定理机器证明研究中建立的多项式组三角化和多项式组零点分解的 “特征列” 方法, 与 20 世纪 60 年代 Buchberger 等建立的 Gröbner 基方法 (参见文献 [21]) 和 Sylvester 早在 19 世纪提出的结式消元方法 (参见文献 [22]), 有效地应用于解多项式方程组的问题. 在吴文俊 “证定理” 和 “解方程” 的思想倡导下, 数学机械化在多个领域的探索相继获得成功, 例如, 单点例证方法^[23]、多点例证数值并行算法^[24]、基于几何不变量的 Euclid 几何和非欧几何定理可读证明方法^[25, 26]、正规特征列计算和多项式方程组的消元方法^[27]、多元多项式近似因式分解和最大公因式计算的符号 - 数值混合计算方法^[28, 29]、微分方程孤立子解计算、微分方程 Dixon 结式和微分代数消元 (参见文献 [30, 31]) 等工作. 此外, 数学机械化较为成功的还有组合定理的机器证明, 包括组合恒等式证明、超几何级数的求和等以及曲面参数化表示和曲面拼接研究等.

以上工作, 其中大部分所处理的问题, 经过代数化以后, 都归结到多元多项式的计算. 而多项式在

计算机中, 可以方便地表示为列表 (list). 例如, 经过两点 (x_1, y_1) 和 (x_2, y_2) 的直线方程可以表示为

$$(\text{= } (+ (* x (- y_1 y_2)) (* y -1 (- x_1 x_2)) (- (* x_1 y_2) (* x_2 y_1))) 0),$$

直线 $ax + by + c = 0$ 与圆 $(x - x_0)^2 + (y - y_0)^2 = R^2$ 相切于点 (x_1, y_1) 这个几何性质, 在计算机中可以表示成下面的数据:

```
( and (= (+ (* a x_1) (* b y_1) c) 0)
      (= (+ (^ (- x_1 x_0) 2) (^ (- y_1 y_0) 2)) (^ R 2))
      (= (* a (- y_1 y_0)) (* b (- x_1 x_0)))).
```

多项式的运算, 也容易通过符号计算实现. 例如, 下面的这个简短的 Lisp 程序使用 lambda 语句来递归定义多项式的乘法 mul:

```
(defun mul (p q)
  (if (not (listp p))
      (if (or (not (listp q)) (equal (car q) '^)) (list '* p q)
          (if (equal (car q) '+)
              (cons '+ (mapcar (lambda (r) (mul p r)) (cdr q)))
              (list '* p q)))
      (if (not (listp q)) (mul q p)
          (if (not (equal (car p) '+))
              (if (not (equal (car q) '+))
                  (if (and (equal (car p) '^) (equal (car q) '^))
                      (equal (cadr p) (cadr q)))
                      (list '^ (cadr p) (add (caddr p) (caddr q)))
                      ('list * p q))
                  (cons '+ (mapcar (lambda (q1) (mul p q1)) (cdr q))))
              (cons '+ (mapcar (lambda (p1) (mul p1 q)) (cdr p)))))).
```

注意, 这里为了简单, 假设多项式已经规范化表示为三种规范形式:

- (1) a (其中 a 是一个数或者原子);
- (2) $(+ p_1 p_2 \dots)$ (其中 $p_1, p_2, \dots$ 是多项式);
- (3) $(^ p_1 k)$ (其中 p_1 是多项式, k 是数或者原子).

还有, 上面的乘法程序中没有执行合并同类项运算, 该操作可通过多项式加法 add 定义. 有了多项式的符号计算程序, 基于解析几何表示的机械化证明算法就能在计算机中实现.

几何定理机器证明基于规则匹配和重写的实现算法, 也使用列表的格式在机器中表示几何性质, 例如, 下面这两个性质和一个定义:

- (1) 三角形中位线性质: 如果 D 和 E 分别是 AB 和 AC 的中点, 则 DE 平行于 BC ;
- (2) 平行关系传递性质: 如果 AB 平行于 CD 和 CD 平行于 EF , 则 AB 平行于 EF ;
- (3) 平行四边形的定义: 如果 AB 平行于 DC 和 BC 平行于 AD , 则 $ABCD$ 是平行四边形.

在计算机中可分别表达为下面的数据:

```
(implies (and (mid-point D A B) (mid-point E A C)) (parallel D E B C)),
(implies (and (parallel A B C D) (parallel C D E F)) (parallel A B E F)),
(def (and (parallel A B D C) (parallel B C A D)) (parallelogram A B C D)).
```

搜索算法分前推搜索和后推搜索. 前推搜索中, 根据已知事实的集合和规则的集合通过匹配、重

写发现新的事实, 即对每一个规则

$$\mathfrak{R} = (\text{implies } (\text{and } H_1 H_2 \cdots) C),$$

这里, $H_1, H_2, \dots$ 是规则 $\mathfrak{R}$ 的条件部分, C 是其结论语句, 在当前已知事实 $C_1, C_2, \dots, C_n$ 的各种组合中找到能与 $\mathfrak{R}$ 的条件部分相匹配的“有效组合”, 得到一个可能的新结论 C' . 若确认 C' 与 $C_1, C_2, \dots, C_n$ 均不同, 就将这个新事实, 记为 C_{n+1} , 加入已知事实的集合. 这里, 一些事实的一个组合 $C_{i_1}, \dots, C_{i_k}$ 与规则 $\mathfrak{R}$ 的条件语句 $(\text{and } H_1 H_2 \cdots H_j)$ 可以匹配, 指的是

$$k = j, \quad (\text{car } C_{i_1}) = (\text{car } H_1), \quad \dots, \quad (\text{car } C_{i_j}) = (\text{car } H_j),$$

以及集合 $\bigcup_l (\text{cdr } C_{i_l})$ 和集合 $\bigcup_l (\text{cdr } H_l)$ 之间的一一对应 ϕ , 使得

$$(\phi (\text{cdr } C_{i_l})) = (\text{cdr } H_l)$$

对于 $l = 1, 2, \dots, j$ 同时成立. 重写的结果就是 $C' = (\phi C)$. 这里记号 (ϕX) 表示将 ϕ 作用于语句 X .

在后推搜索中, 先从要证明的命题 R 的结论出发, 根据规则和已知事实的匹配, 将 R 的正确性归结为一个或一些新命题 $R_1, R_2, \dots, R_k$, 假设已从当中排除跟已知事实相同的命题. 再对这些命题递归调用这一程序, 直到待证明的命题集合为空集.

由于搜索过程中, 已知事实的组合可以组织成树的形式, 所以有“深度优先”和“广度优先”两种搜索策略. 深度优先的搜索中, 把每一次得到的“新事实”立即添加到事实集合, 下次推理时就参加组合、匹配; 广度优先时, 得到的新事实放到“缓存区”, 在下一轮推理时生效. 一般, 一轮推理指的是遍历所有的规则所进行的搜索.

显然, 不论前推或后推, 搜索推理与人类证明定理的过程, 逻辑上是一致的. 从上面的介绍可以看出来, 如果一个定理的条件和结论, 以及人类证明该定理所依赖的定义、引理和定理 (统一视为推理规则), 能够用适当的符号列表在计算机里表示成列表, 只要这个定理 (目标) 已经有人工证明, 则这个证明过程就可以在机器上再现. 这样, 也就可以设计适当的程序, 自动地应用每个推理规则, 在有限轮推理之后, 推理目标就可以达到, 除非机器的存储空间在推理的中间某一步耗尽. 相应地, 如果数学的某一门学科的所有概念、定义和基本定理, 都可以用列表这种符号形式来表达, 则这门学科就可能实现机械化.

因此, 点集拓扑学的机械化, 首先要解决拓扑空间、导集和闭集等基本概念在计算机内部的符号表示. 这一步可称为点集拓扑学知识的形式化. 如同初等几何定理机器证明中的第一步代数化, 先约定采用解析几何的坐标 (或长度、面积等几何不变量)、方程表示几何对象和几何关系, 或者采用列表表示事实和规则, 然后针对代数化方式通过符号计算或者搜索推理实现证明的自动化. 第一步之后推理或计算, 可以是人机交互的, 也可以是全自动化的. 自动化的效率, 取决于符号计算或搜索推理的算法和机器的硬件性能.

要在机器中表示集合论、拓扑学和抽象代数等学科中的基本概念, 需要用高阶逻辑语言. 这些语言, 现在发展较为完善的有 Isabelle、Coq 和 Mizar 等, 其中 Isabelle 使用的是高阶逻辑语言, 而 Coq 使用的是依赖类型论, Mizar 使用的是集合论. 本文将使用高阶逻辑辅助证明语言 Isabelle 实现拓扑空间基本概念的形式化, 以给出杨忠道定理的一个机器证明. Isabelle 可以用形式化语言表达数学概念和公式, 是一个通用的、用匹配搜索等逻辑演算方法、在计算上验证或证明某些数学定理的通用工具

(参见文献 [32–34]). 最早由剑桥大学和慕尼黑工业大学的学者开发, 后为世界各地机构和学者采用, 并因此得到丰富和发展. 现在传播最广的版本是 Isabelle/HOL, 它包含的高阶逻辑证明环境支持复杂和大规模问题应用. 最近 30 年以来, 已经有越来越多的数学分支的概念用 Isabelle 语言形式化, 很多重要的定理得到形式化语言的机械化验证. 根据 the archive of formal proofs^[35] 上发表的资料, 已经被 Isabelle 语言形式化表达、验证的数学理论或定理涵盖了逻辑学、集合论、微积分、基础代数和抽象代数学、数论、组合数学等分支.

Coq 也是近年在数学定理机器证明和计算机硬件和软件设计的正确性验证中得到广泛应用的一个形式化工具 (参见文献 [36]). 用 Coq 交互式证明器完成了严格的形式化表达和证明的数学定理包括四色定理和 Feit-Thompson 奇阶定理等这些有名的结果. 由法国 Inria 维护的网站 <https://coq.inria.fr/>, 发布用 Coq 语言完成形式化验证的数学定理和计算机软硬件系统^[37]. 近年, 北京邮电大学郁文生等^[38] 用 Coq 实现了 Morse-Kelley 公理化集合论的形式化.

网站 [39] 记录了 Wiedjik 列出的 100 个证明目标迄今在 Isabelle、Coq 和 Mizar 等形式化系统中的完成情况, 其中有 83 个在 Isabelle 中完成, 有 70 个在 Coq 完成, 有 69 个在 Mizar 中完成, 大部分定理在两个或更多的形式化系统中得到证明.

表 1 列出过去 30 年用形式化方法验证的一些定理. 表格中的 Flyspeck I 指 Kepler 猜想的形式化证明 (formal proof of the Kepler conjecture). 从这些结果看出, 基于形式化方法的数学机械化研究已经广泛而且深入, 其发展也越来越快.

拓扑学的形式化方面, Hölzl 等^[46] 给出了点集拓扑学和实分析在 Isabelle 的实现, 其中包括了点集拓扑学的开集、闭集、邻域和闭包等基本概念的 Isabelle 形式化, 由于这一工作中拓扑概念的建立主要是为实分析理论的形式化服务的, 所以, 拓扑学的展开跟一般拓扑学教科书的次序略有不同, 例如, 其中的拓扑空间作为全集来构造, 而不是作为给定集合的某些子集合的一个族来处理. 这一工作

表 1 1986–2019 年间经过形式化验证的一些重要定理和算法 (根据文献 [40, 表 1] 和网站 [35, 37] 的资料制作)

年份	定理	证明系统	形式化证明者	传统证明完成者
1986	第一不完备性定理	Boyer-Moore	Shankar	Gödel
1990	二次互反律	Boyer-Moore	Russinoff	Eisenstein
1996	微积分基本定理	HOL Light	Harrison	Henstock
2000	代数基本定理	Mizar	Milewski	Brynski
2000	代数基本定理	Coq	Geuvers et al.	Kneser
2004	四色定理 ^[41]	Coq	Werner and Gonthier	Robertson et al.
2004	素数定理	Isabelle	Avigad et al.	Selberg-Erdős
2005	约当曲线定理	HOL Light	Hales	Thomassen
2005	Brouwer 不动点定理	HOL Light	Harrison	Kuhn
2006	Flyspeck I	Isabelle	Bauer-Nipkow	Hales
2007	Cauchy 留数定理	HOL Light	Harrison	Cauchy
2008	素数定理	HOL Light	Harrison	Hadamard
2010	周高张面积法 ^[42]	Coq	Narboux	周咸青、高小山和张景中
2011	吴方法 ^[43]	Coq	Génevaux et al.	吴文俊
2012	Dijkstra 最短路径算法	Isabelle	Nordhoff	Dijkstra
2012	Feit-Thompson 奇阶定理	Coq	Gonthier	Feit and Thompson
2015	Euler 分拆定理	Isabelle	Bulwahn	Euler
2016	x86 指令集 ^[44]	Coq	Kennedy et al.	Intel
2017	Lindemann-Weierstrass 定理 ^[45]	Coq	Bernard	Lindemann-Weierstrass
2018	LLL 算法	Isabelle	Bottesch et al.	Lenstra et al.
2019	Farkas 引理	Isabelle	Bottesch et al.	Farkas
2019	Nullstellensatz	Isabelle	Alexander Maletzky	Hilbert

中没有杨忠道定理的形式化证明. Gammie 和 Gioiosa^[47] 在 2020 年用 Isabelle 语言验证了如下有趣的定理: 对拓扑空间 X 的任意子集, 使用闭包和补集运算及其任意复合, 最多可产生 14 个集合, 当 X 是实数集时, 可构造 A 使得由这两个运算恰好产生 14 个集合. 这个定理是 Kuratowski^[48] 在 1922 年发现的. 与杨忠道定理一样, 这个定理是拓扑学中一个非常有名的结果 (参见文献 [9]). 另一个与拓扑学机械化有关的工作是 Friedrich^[49] 关于 lazy list 拓扑学的形式化工作, 这一工作建立了点集拓扑学的基本概念, 证明了拓扑空间与各种分离性有关的一些性质. Gammie 和 Gioiosa 及 Friedrich 的工作, 也都没有包含杨忠道定理的证明.

Coq 形式化系统发布网站^[37] 包含一个实数集合拓扑学的 Coq 形式化证明工作^[50], 包括了点集拓扑学基本概念 (开集、邻域、连续性和紧致性等) 的 Coq 形式化表达, 以及 Borel-Lebesgue 引理, Bolzano-Weierstrass 定理等结果的形式化验证. 本文作者还没有在文献中找到杨忠道定理的 Coq 形式化表达和证明. 在使用 Mizar 形式化语言的数学定理机器验证方面, Grabowski^[51] 给出了 Kuratowski 闭包定理的形式化证明. 在形式化语言 Boyer-Moore、Mizar 和 HOL Light 等系统的文献中, 目前也没见到关于杨忠道定理的验证.

本文第 2 节给出点集拓扑学中拓扑空间、邻域、开集、闭集、内部和闭包等基本概念和基本定理的形式化表达和证明. 第 3 节先给出聚点和导集的形式化, 然后给出杨忠道定理以及相关引理的形式化证明. 第 4 节简略讨论用形式化系统对定理进行机械化证明时的全自动化问题.

2 拓扑空间的形式化与相关基本定理的机器证明

本节将用 Isabelle 语言给出拓扑空间的基本概念, 包括开集、邻域、闭集、内部和闭包的形式化定义, 并验证它们的一些基本性质. 这些形式对于本文主要目标—杨忠道定理的机器证明, 都是必需的. 为了较完整呈现拓扑学数学机械化的方法, 也将给出一些特殊的拓扑空间的例子, 给出有关结果的形式化和证明. 对于了解这些拓扑学的基本知识并且了解 Isabelle 语言初步语法的读者, 本节展现的 Isabelle 证明脚本也是可读的.

为了阅读方便, 我们将把用自然语言表达的拓扑学定理, 统一使用命题 2.1, 2.2, ... 这样的格式来标记, 而把用 Isabelle 语言表示的结论, 用 lemma 来表示. 在 Isabelle 语句中, lemma 语句的第一行格式如下例:

```
lemma (in topological_space) union_in_T: ,
```

其中 (in topological_space) 指明定理成立的范围, union_in_T 标记推理中应用该定理时的名称. 我们将在自然语言文本中统一使用“引理”表示用 Isabelle 语言表达的定理, 使用 lemma union_in_T 指称具体的引理. 自然语言和 Isabelle 语言表达的定义, 都称为定义.

关于 Isabelle 的语法, 详细可参见 Wenzel^[52, 53] 的著作.

2.1 拓扑空间与开集

拓扑空间是拓扑学的基本研究对象. 在拓扑学发展过程中, 拓扑空间曾经有许多不同的定义方式. 现在一般使用开集公理的方式来定义拓扑空间. 本文也采用这个方式来定义抽象的拓扑空间. 下面是用自然语言表达的拓扑空间的一个定义:

定义 2.1^[54] 集合 X 上的一个拓扑 (topology) 是 X 的子集的一个族 T , 满足以下条件:

- (1) $\emptyset$ 和 X 在 T 中;

(2) T 的任意子族的元素的并在 T 中;

(3) T 的任意两个元素的交在 T 中.

集合 X 连同 X 上的拓扑 T 构成的二元组 (X, T) 称为一个拓扑空间 (topological space).

注释: 为了与 Isabelle 语言里表达子集族的字母的字体相一致, 这里使用 T 来表示拓扑, 而不是像一般的教科书里使用字体 $\mathcal{T}$.

为了在 Isabelle 语言中表达拓扑空间的概念, 首先引入了一个新的数据类型 `top`, 用于表示空间中的拓扑类型, 即集合的集合. 这个数据类型具体表示如下:

```
type_synonym
  'a top = "'a set set".
```

这里的 Isabelle 语言使用 `locale` 表示拓扑空间. 其中, `locale` 是 Isabelle 语言中的多参数结构体 (参见文献 [55, 56]), 一般用来表示较为复杂的概念, 以简化 Isabelle 中的形式表达. `locale` 也可以用于定义模块, 对模块中的元素进行封装, 使 Isabelle 中的数学对象便于管理.

拓扑空间的定义 2.1 在 Isabelle 语言中表示为下面的形式:

```
locale topological_space =
  fixes T :: "'a top" (structure)
  and X :: "'a set" (structure)
  assumes O1 : "T <= Pow X"
  and O2 : "{} : T"
  and O3 : "X : T"
  and O4 : "[| x : T ; y : T |] ==> x Int y : T"
  and O5 : "[| T' <= T |] ==> UN T' : T".
```

上面关于拓扑空间的 Isabelle 形式化表示中, `fixes` 用于定义 `locale` 中的对象, 指明对象的名称和类型, 包括集合 X 和它的拓扑 T . 这里 `structure` 指明该对象是实体, 含有结构, 而非函数. 接着的 `assumes` 部分, `O1` 表示 T 是 X 的一个子集族, `O2` 表示 T 中包含空集, `O3` 表示 X 是 T 的元素, `O4` 表示 T 中任意两个元素的交仍在 T 中, `O5` 表示 T 中任意集合 (有限或无限) 的并仍在 T 中.

从 `O5` 中容易看出, T 中任意两个元素的并仍然属于 T . 但是在 Isabelle 形式系统中这个结论并不是显然的, 因此, 我们需要证明如下命题:

命题 2.1 设 (X, T) 为拓扑空间, 对于任意的 $x, y \in T$, 有 $x \cup y \in T$.

这个命题在 Isabelle 语言中可表示为下面的形式:

```
lemma (in topological_space) union_in_T:
  assumes a1 : "x : T"
  and a2 : "y : T"
  shows "x Un y : T".
```

下面讨论如何在 Isabelle 形式系统验证 `lemma union_in_T`. 证明的路线是, 先从命题的两个假设 $x \in T$ 和 $y \in T$, 得到

$$\{x, y\} \subseteq T.$$

这一点是先由事实 `O5`:

$$S \subseteq T \Rightarrow \cup S \in T$$

得到结论

$$\cup\{x, y\} \in T,$$

然后得到 $x \cup y \in T$. 上述证明过程, 用 Isabelle 形式语言表示如下:

```
proof-
  from a1 a2 have b: "{x,y} <= T"
    apply simp done
  from b have c: "UN {x,y} : T"
    apply (simp add: O5) done
  have d: "UN {x,y} = x Un y"
    apply blast done
  from c d have e: "x Un y : T"
    apply simp done
  shows ?thesis apply (blast intro : e) done
qed.
```

上述 Isabelle 证明脚本虽然证明了 lemma union_in_T, 但是自动化程度较低. 下面利用证明策略序列的方式, 给出一个较为机械化的证明. 其过程可分解为如下步骤:

- 首先抽象出待证命题的结论 $x \cup y \in T$ 和拓扑空间定义中 O5 的 $\cup S \in T$ 的形式结构;
- 然后在 Isabelle 的定理库中寻找与两者匹配的定理, 可以找到一个已知的定理: Un_eq_Union: $A \cup B = \cup\{A, B\}$, 这是 Isabelle 形式系统中已经内建的定理 (Un_eq_Union 是定理的名称). 使用替换策略把这个已知定理与 O5 联系在一起. 这里替换策略的具体形式是 (subst Un_eq_Union), 其中 subst 表示构建一个替换策略, 它把已知等式性事实的左边替换成右边. 产生的子目标如下:

> 1. $[| x : T; y : T |] ==> UN \{x, y\} : T;$

- 进而注意到这个子目标和拓扑空间定义中的 O5 有相似的结论, 使用 (rule O5) 策略, 得到下面新的子目标:

> 1. $[| x : T; y : T |] ==> \{x, y\} <= T;$

- 最后使用 simp 自动证明策略, 完成引理 union_in_T 的证明.

以上的机械化证明过程, 用 Isabelle 语言写成如下完整证明脚本:

```
lemma (in topological_space) union_in_T:
  "[| x : T ; y : T |] ==> x Un y : T"
  apply (subst Un_eq_Union)
  apply (rule O5)
  apply simp
done.
```

在拓扑空间 (X, T) 中, 显然 T 是 T 的子集, 那么到 O5, 有如下结论:

命题 2.2 设 (X, T) 是拓扑空间, 则有 $\cup T \in T$.

命题 2.2 可在 Isabelle 语言中表示为

```
lemma (in topological_space) Un_T_in_T:
  "Union T : T".
```

要证明 $\cup T \in T$, 只需要运用拓扑空间的定义 O5. 在 Isabelle 证明器中, 将下面的两行接着写到上面的两行之后:

```
  apply (simp add: O5)
done.
```

进一步, 有如下结论:

命题 2.3 设 (X, T) 是拓扑空间, 则有 $\cup T = X$.

这个命题在 Isabelle 中的表示如下:

```
lemma (in topological_space) Un_T_eq_X:
  "Union T = X".
```

要证明 lemma `Un_T_eq_X`, 首先使用子目标策略引入一个子目标

$$T \subseteq \text{Pow } X,$$

使得待证命题的目标变成:

```
> 1. T <= Pow X ==> Union T = X
   2. T <= Pow X;
```

然后使用子目标策略引入另一个子目标 $X \in T$, 使得命题的目标变成:

```
> 1. [| T <= Pow X; X : T |] ==> Union T = X
   2. T <= Pow X ==> X : T
   3. T <= Pow X;
```

再运用 `blast` 证明策略, 证明第一子目标, 这时命题的状态变成:

```
> 1. T <= Pow X ==> X : T
   2. T <= Pow X;
```

最后使用两次 `simp` 策略, 命题即得证. 这里为了简明, 不列出这个命题的证明的完整脚本.

对于任意集合 X , 可在 X 上构造两个特殊的拓扑空间, 一个是最粗 (小) 的拓扑, 其拓扑 T 包含 $\emptyset$ 和 X , 另一个是最细 (大) 拓扑, 其拓扑 T 包含 X 的所有子集. 二者分别称为 X 上的平凡拓扑和离散拓扑.

定义 2.2 设集合 X 上的拓扑 T 仅由空集 $\emptyset$ 和 X 组成, 称这样的拓扑 T 为平凡拓扑 (trivial topology).

在 Isabelle 中, 平凡拓扑的定义可形式化表示如下:

```
definition trivial_topology ::
  "'a set => 'a top"
  where
    "trivial_topology X = {{}, X} ".
```

在 Isabelle 证明系统中, 不能直接得出集合 X 和 X 的平凡拓扑构成一个拓扑空间, 需要证明以下命题.

命题 2.4 设 T 是 X 上的平凡拓扑, 则 (X, T) 是一个拓扑空间.

在 Isabelle 语言中命题 2.4 可表示如下:

```
lemma trivial_is_top:
  "topological_space (trivial_topology X) X".
```

下面描述上述命题在 Isabelle 中的证明过程. 首先引入拓扑空间的定义, 将命题分解为与拓扑空间定义中的 O1 到 O5 相关的子命题:

```
apply (rule topological_space.intro),
```

得到如下 5 个子目标:

```

> 1. trivial_topology X <= Pow X
2. {} : trivial_topology X
3. X : trivial_topology X
4. !!x y.
   [| x : trivial_topology X; y : trivial_topology X |]
   ==> x Int y : trivial_topology X
5. !!T1. T1 <= trivial_topology X
   ==> Union T1 : trivial_topology X.

```

通过使用 `simp` 策略, 并引入平凡拓扑的定义, 可以证明前 3 个子目标, 并把子目标 4 和 5 按照平凡拓扑的定义展开如下:

```

> 1. !!x y.
   [| x = {} | x = X; y = {} | y = X |]
   ==> x Int y = {} | x Int y = X
2. !!T1. T1 <= trivial_topology X
   ==> Union T1 : trivial_topology X.

```

然后使用 `blast` 策略, 证明子目标 1, 于是待证的命题转化为如下结论:

```

> 1. !!T1. T1 <= trivial_topology X
   ==> Union T1 : trivial_topology X.

```

再次使用 `simp` 策略, 引入平凡拓扑的定义, 使得前面这个证明目标转变成:

```

> 1. !!T1. T1 <= {{}, X}
   ==> (ALL x : T1. x = {}) | Union T1 = X.

```

这时, 要证明的命题转换为

```

> 1.  $\forall T. T1 \subseteq \emptyset, X \Rightarrow (\forall x \in T. x = \emptyset) \vee \cup T1 = X,$ 

```

最后再次使用 `blast` 策略, 即完成证明. 以下是完整的证明脚本:

```

apply (rule topological_space.intro)
apply (simp add: trivial_topology_def)+
apply blast
apply (simp add: trivial_topology_def)
apply blast
done.

```

定义 2.3 设集合 X 上的拓扑 T 由 X 的一切子集组成, 则称 T 为离散拓扑 (discrete topology).

离散拓扑在 Isabelle 中的表示如下:

```

definition discrete_topology ::
  "'a set => 'a top"
  where
    "discrete_topology X = Pow X".

```

类似于平凡拓扑, 可证明 X 和其上的离散拓扑构成一个拓扑空间, 即如下命题.

命题 2.5 设 T 是 X 上的离散拓扑, 则 (X, T) 是一个拓扑空间.

下面是命题 2.5 的 Isabelle 语言形式化和完整的证明脚本.

```

lemma discrete_is_top :
  "topological_space (discrete_topology X) X"
  apply (rule topological_space.intro)
  apply (simp add: discrete_topology_def)+
  apply blast
  apply (simp add: discrete_topology_def)
  apply blast
done.

```

注 2.1 离散拓扑有一个有趣的性质^[57]: 设 T 是 X 上的拓扑, 则 T 是离散拓扑, 当且仅当对于所有的 $x \in X$, 有 $\{x\} \in T$. 读者可试着给出其在 Isabelle 语言中的形式化表示.

同一集合 X 上的不同拓扑, 可以定义如下偏序关系.

定义 2.4^[57] 设 X 是一个集合, T_1 和 T_2 是 X 上的两个拓扑. 若

$$T_1 \subseteq T_2,$$

则称 T_2 细于 T_1 , 而称 T_1 粗于 T_2 . 此外, 若 T_2 细于 T_1 , 但不等于 T_1 , 则称 T_2 严格细于 T_1 . “严格粗于”可类似定义.

这里只给出“细于”在 Isabelle 中的表示, “粗于”以此类推.

```

definition
  finer :: "'a set => 'a top => 'a top => bool"
  where
    "finer T2 T1 X == topological_space T1 X &
                      topological_space T2 X &
                      T1 <= T2".

```

显然有这样的事实: 离散拓扑细于平凡拓扑. 这一事实在 Isabelle 中表示如下:

```

lemma "finer (discrete_topology X) (trivial_topology X) X".

```

这个引理的证明中, 先引入 `finer` 的定义, 对结论进行化简. 考虑到子目标是合取式, 使用 (`rule conjI`) 策略将目标分解为两个子目标, 再引入 `trivial_is_top` 简化每个子目标, 以此类推, 直到引理得证. 以下是 Isabelle 证明脚本.

```

  apply (simp add :finer_def)
  apply (rule conjI)
  apply (rule trivial_is_top)
  apply (rule conjI)
  apply (rule discrete_is_top)
  apply (simp add: trivial_topology_def discrete_topology_def)
done.

```

下面给出一种既非平凡拓扑又非离散拓扑但结构也很简单的拓扑空间. 这个结论虽然对于本文要证明的杨忠道定理的证明是不必要的, 但我们仍然给出它和它的形式化证明, 以便读者熟悉 Isabelle 的证明方法.

命题 2.6^[58] 设

$$X = \{a, b\}, \quad T = \{\emptyset, \{a\}, X\},$$

则 (X, T) 是一个拓扑空间.

命题 2.6 中确定的拓扑空间也称为 Sierpinski 空间. 这个性质在 Isabelle 语言中的证明, 可表示为下面形式:

```
lemma "[| X = {a, b} ;
          T = {{}, {a}, X}
        |] ==> topological_space T X"
  apply (simp add: topological_space_def)
  apply blast
done.
```

开集是拓扑空间中的最重要的概念之一. 相关开集的定理虽然简单, 但是在以后的证明中能起到非常重要的作用. 设 (X, T) 是一个拓扑空间, T 的每一个元素都叫作拓扑空间的一个开集. 这一概念在 Isabelle 中的形式化表示如下:

```
definition (in topological_space)
  open_set :: "'a set => bool"
  where
    "open_set s == s : T".
```

这里把 `open_set` 定义成 “ $a \text{ set} \Rightarrow \text{bool}$ ” 的一个函数, 即集合到布尔值的一个函数, 符合检验集合是否为开集的习惯. 从开集的定义可知, 拓扑空间 (X, T) 中, T 中的每个元素是开集, 因此, 空集和 X 本身也是开集. 开集的这个性质在 Isabelle 中表示为下面两个引理:

```
lemma (in topological_space) empty_is_open:
  "open_set {}",

lemma (in topological_space) X_is_open:
  "open_set X".
```

上述两个引理的证明中, 只需要在证明脚本中引入开集的定义, Isabelle 即可自动完成证明.

从拓扑空间定义中的 O4 可知, 两个开集的交仍旧是开集, 两个开集的并仍是开集. 这两个事实在 Isabelle 中可以表示成下面的两个引理:

```
lemma (in topological_space) inter_is_open:
  "[| open_set A ; open_set B |]
   ==> open_set (A Int B)",

lemma (in topological_space) union_is_open:
  "[| open_set A ; open_set B |]
   ==> open_set (A Un B)".
```

Isabelle 系统可以自动证明上述两个引理. 具体方法是在引理的三行描述之后, 执行下面两行程序:

```
apply auto
done.
```

根据开集的定义易知, 任意 (包括有限和无限) 开集的并仍是开集. 用 Isabelle 语言, 这一命题可表示如下:

```
lemma (in topological_space) UN_is_open:
  "[| All x : T' ; open_set x |]
   ==> open_set (Union T')".
```

在这个结论的证明过程中, 除了引入开集的定义之外, 还需要引入一个子目标 $T' \subseteq T$ 和事实

subsetI: " $(\bigwedge x. x \in A \Rightarrow x \in B) \Rightarrow A \subseteq B$ ",

具体的证明脚本如下:

```

    apply (simp add: open_set_def)
    apply (subgoal_tac "T' <= T")
    apply simp
    apply (simp add: subsetI)
done.

```

下面引理说的是, 拓扑空间 (X, T) 中开集的元素是 X 的元素, 这个命题在 Isabelle 形式系统中表示如下:

```

lemma (in topological_space) openE_in_X:
  "[| open_set t; x : t |] ==> x : X".

```

这个引理的证明中, 先引入子目标 $T \subseteq \text{Pow } X$ 和 $t \in T$, 该引理变成下面的 3 个:

```

> 1. [| open_set t; x : t; T <= Pow X; t : T |] ==> x : X
  2. [| open_set t; x : t; T <= Pow X |] ==> t : T
  3. [| open_set t; x : t |] ==> T <= Pow X,

```

然后使用 blast 和 simp 策略, 即完成如下证明:

```

    apply (subgoal_tac "T <= Pow X" "t : T")
    apply blast
    apply (simp add: open_set_def)+
done.

```

开集中还有一些简单的性质, 也可用 Isabelle 直接证明, 例如:

```

lemma (in topological_space) subE_is_open:
  "[| t : M; M <= T |] ==> open_set t",

lemma (in topological_space) open_set_E:
  "[| x : X;
    !!t. [| open_set t; x : t |] ==> R
  |] ==> R".

```

2.2 邻域的概念与性质

邻域是拓扑空间一个十分重要的概念, 是讨论连续性的必要工具.

定义 2.5^[7] 设 (X, T) 是一个拓扑空间, 且 $x \in X$, 如果 U 是 X 的一个子集, 满足条件: 存在一个开集 $V \in T$, 使得 $x \in V \subseteq U$, 则称 U 是点 x 的一个邻域. 点 x 的所有邻域构成的 X 的子集族称为点 x 的邻域系. 如果 U 是包含着点 x 的一个开集, 则它一定是 x 的一个邻域, 于是称 U 是点 x 的一个开邻域.

在 Isabelle 中可将定义 2.5 形式化表示如下:

```

definition (in topological_space) neighborhood ::
  "'a set => 'a => bool"
  where
    "neighborhood U x == U <= X &
      (EX V : T. x : V & V <= U)".

```

这里, 邻域定义成 $(\text{'a set}, \text{'a}) \Rightarrow \text{bool}$ 的一个函数. 下面给出开邻域在 Isabelle 中的定义形式:

```

definition (in topological_space) open_neighborhood::
  "'a set => 'a => bool"
  where
    "open_neighborhood U x == x : U & open_set U".

```

注意, 定义 2.5 中的语句“如果 U 是包含着点 x 的一个开集, 则它一定是 x 的一个邻域”虽然是显然的, 但在 Isabelle 形式系统中, 这一点还需要加以严格证明.

下面是这个语句在 Isabelle 中的表达形式:

```

lemma (in topological_space) open_nb_is_nb ::
  "open_neighborhood U x ==> neighborhood U x".

```

上面这个引理的证明思路是, 先引入 `open_neighborhood` 和 `neighborhood` 的定义, 结合 `rule conjI` 策略, 将问题转化成若干个相对简单的问题, 再引入两个子目标, $T \subseteq \text{Pow } X$ 和 $U \subseteq \text{Pow } X$, 使用策略 `rule exI` 去掉存在量词, 最后运用 `blast` 和 `simp` 等策略对所有的简单问题逐一证明. 这里省略具体的证明过程.

定义 2.5 中还定义了邻域系, 这也是点集拓扑学中的重要概念. Isabelle 中邻域系可如下表示:

```

definition (in topological_space) neighborhood_system ::
  "'a => 'a set set"
  where
    "neighborhood_system x = { U. (neighborhood U x) }".

```

从这个定义看出, 邻域系的数据类型是集合的集合. 可以证明下面的结果.

命题 2.7 设 (X, T) 是拓扑空间, 如果 $x \in X$, 则 X 也属于 x 的邻域系.

可将命题 2.7 在 Isabelle 语言中表示如下:

```

lemma (in topological_space) X_in_nbs:
  "x : X ==> X : neighborhood_system x".

```

这个命题的证明思路是, 先引入邻域系和邻域的定义, 把问题展开, 再引入 `O3:  $X \in T$` 这个事实, 最后使用 `rule bexI` 策略去掉存在量词. 完整的 Isabelle 证明脚本如下:

```

apply (simp add:neighborhood_system_def)
apply (simp add:neighborhood_def)
apply (insert O3)
apply (rule_tac x = "X" in bexI)
apply (rule conjI)
apply assumption
apply (rule subset_refl)
apply assumption
done.

```

以下的几个命题给出邻域的更多性质. 我们给出每个性质的 Isabelle 形式化及其证明.

命题 2.8 设 (X, T) 是拓扑空间, NBS 为点 $x \in X$ 的邻域系 (neighbourhood system), 则

- (1) 对于任何 $x \in X$, NBS 不为空, 并且如果 $U \in \text{NBS}$, 则 $x \in U$;
- (2) 如果 $U, V \in \text{NBS}$, 则 $U \cap V \in \text{NBS}$;
- (3) 如果 $U \in \text{NBS}$, 并且 $U \subseteq V$, 则 $V \in \text{NBS}$;
- (4) 如果 $A \subseteq X$, A 是开集, $x \in A$, 则存在 x 的邻域 U 使得 $U \subseteq A$;
- (5) 如果 $U \in \text{NBS}$, 则存在开集 V , 使得 $x \in V$ 和 $V \in \text{NBS}$.

这个命题在 Isabelle 中表示成 5 个 lemma 的形式, 具体表达和证明脚本如下:

```

lemma (in topological_space) nbs_in:
  "U : neighborhood_system x ==> x : U"
  apply (simp add: neighborhood_system_def)
  apply (simp add: neighborhood_def)
  apply blast
done,

lemma (in topological_space) nbs_inter:
  "[| U : neighborhood_system x;
    V : neighborhood_system x
  |] ==> U Int V : neighborhood_system x"
  apply (simp add: neighborhood_system_def)
  apply (simp add: neighborhood_def)
  apply blast
done,

lemma (in topological_space) nbs_sub_in:
  "[| A : neighborhood_system x;
    A <= B ;
    B <= X
  |] ==> B : neighborhood_system x"
  apply (simp add: neighborhood_system_def)
  apply (simp add: neighborhood_def)
  apply blast
done,

lemma (in topological_space) nb_subset_open:
  "[| A <= X ;
    open_set A ;
    x : A
  |] ==> EX U. neighborhood U x & U <= A"
  apply (unfold neighborhood_def)
  apply (unfold open_set_def)
  apply auto
done,

lemma (in topological_space) nb_ex_onb:
  "[| U : neighborhood_system x |] ==>
    EX U1. U1 : open_neighborhood_system x"
  apply (simp add: neighborhood_system_def
    open_neighborhood_system_def
    neighborhood_def
    open_neighborhood_def)
  apply auto
done.

```

命题 2.9 设 (X, T) 是拓扑空间, 对任意的 $x \in X$, 有 x 的邻域系不为空.

使用 Isabelle 语言, 命题 2.9 的表示形式如下:

```

lemma (in topological_space) nbs_not_empty:
  "[| x : X |] ==> neighborhood_system x ~= {}".

```

其证明过程如下:

```

    apply (simp add: neighborhood_system_def neighborhood_def)
    apply (rule_tac x = "X" in exI)
    apply (rule conjI)
    apply (simp)
    apply (rule_tac x = "X" in bexI)
    apply (rule conjI)
    apply assumption
    apply simp
    apply simp
done.

```

命题 2.10 设 (X, T) 是拓扑空间, A 是开集, x 是 X 的元素, 并且 A 属于 x 的邻域系, 则有 A 属于 x 的开邻域系.

上述命题在 Isabelle 中的证明过程如下:

```

lemma (in topological_space) open_nbs:
  "[| open_set A ; A : neighborhood_system x |] ==>
    A : open_neighborhood_system x"
  apply (simp add: open_neighborhood_system_def
    neighborhood_system_def
    open_neighborhood_def
    neighborhood_def
    closed_set_def
    open_set_def)
  apply blast
done.

```

命题 2.11 设 (X, T) 是拓扑空间, x 是 X 的元素, 并且 U 属于 x 的开邻域系, 则 U 是开集.

这个命题在 Isabelle 中的表示和证明过程如下:

```

lemma (in topological_space) open_nbs_o:
  "[| U : open_neighborhood_system x |]
    ==> open_set U"
  apply (simp add: open_neighborhood_system_def
    neighborhood_def
    open_neighborhood_def
    open_set_def)
done.

```

命题 2.12 设 (X, T) 为拓扑空间, 对于任意 $x \in X$, x 的开邻域不为空.

下面是命题 2.12 在 Isabelle 中的具体证明过程.

```

lemma (in topological_space) open_nbs_not_empty:
  "[| x : X |] ==> open_neighborhood_system x ~= {}"
  apply (simp add: open_neighborhood_system_def
    open_neighborhood_def
    neighborhood_def)
  apply (rule_tac x = "X" in exI)

```



```

    apply (rule conjI)
    apply assumption
    apply simp
done.

```

2.3 闭集

定义 2.6 拓扑空间 (X, T) 的子集 C 叫作闭集, 当且仅当它的余集 $X - C$ 为开集.

下面是这个定义在 Isabelle 中的形式化.

```

definition (in topological_space) closed_set ::
  "'a set => bool"
  where
    "closed_set s == open_set (X - s) &
      s <= X".

```

类似于开集, 拓扑空间 (X, T) 中的闭集有如下性质:

命题 2.13 设 (X, T) 是一个拓扑空间, 则

- (1) X 和 $\emptyset$ 是闭集;
- (2) 设 A 和 B 是闭集, 则 $A \cup B$ 是闭集;
- (3) 若 C' 是 (X, T) 中的闭集族, 则 $\cap C'$ 是闭集.

在 Isabelle 中可将命题 2.13 表示成如下 4 个 lemma 的形式加以证明.

```

lemma (in topological_space) X_is_closed:
  "closed_set X"
  apply (simp add: closed_set_def)
  done,

lemma (in topological_space) empty_is_closed:
  "closed_set {}"
  apply (simp add: closed_set_def)
  done,

lemma (in topological_space) union_is_closed:
  "[| closed_set A ;
    closed_set B ;
  |] ==> closed_set (A Un B) "
  apply (simp add: closed_set_def)
  apply (simp add: open_set_def)
  apply (subst Diff_Un)
  apply (erule_tac P="X - A : T" and Q =" A <= X" in conjE)
  apply (erule_tac P="X - B : T" and Q =" B <= X" in conjE)
  apply simp
  done,

lemma (in topological_space) INT_is_closed:
  "[| ALL A : C' . closed_set A
  |] ==> closed_set (INT C') "
  apply (auto simp : closed_set_def open_set_def 05)
  done.

```

下面是离散拓扑空间的一个性质.

命题 2.14 设拓扑空间 (X, T) 为离散拓扑空间, A 为 X 的任意一个子集, 则 A 既是 (X, T) 的开集, 也是 (X, T) 的闭集.

这个命题在 Isabelle 中可表示为两个引理的形式, 前一个证明任意子集合是开集, 后一个证明它是闭集.

```
lemma dis_all_is_open:
  "!!A. A <= X ==>
    topological_space.open_set (discrete_topology X) A",

lemma dis_all_is_closed :
  "!!A. A <= X ==>
    topological_space.closed_set
      (discrete_topology X) X A".
```

下面介绍这两个引理的形式化证明. 前一个引理 (lemma dis_all_is_open) 的证明, 先引用 insert 策略, 引入事实 discrete_is_top, 证明离散拓扑是一个拓扑, 通过 unfold 把离散拓扑和开集按定义展开, 运用

$$\text{PowI} : A \subseteq B \Rightarrow A \in \text{Pow } B$$

重写子目标. 完整的 Isabelle 证明代码如下:

```
apply (insert discrete_is_top[of "X" ])
apply (unfold discrete_topology_def
  topological_space.open_set_def)
apply (rule PowI)
apply assumption
done.
```

后一个引理 (lemma dis_all_is_closed), 即离散拓扑空间的每个子集都是闭集, 证明方法类似. 其证明过程如下:

```
lemma dis_all_is_closed :
  "!!A. A <= X ==>
    topological_space.closed_set
      (discrete_topology X) X A"
apply (insert discrete_is_top[of "X" ])
apply (unfold discrete_topology_def
  topological_space.closed_set_def
  topological_space.open_set_def)
apply (rule conjI)
apply (rule_tac A = "X - x" in PowI)
apply (rule Diff_subset)
apply assumption
done.
```

针对平凡拓扑空间, 有类似的性质和形式化证明. 这里直接写出两个 Isabelle 语句表示的引理. 为了节省篇幅, 这里略去它们的形式化证明. 这不会影响本文主要定理的证明.

```
lemma tri_all_is_open:
  "!!A. A <= X ==>
    topological_space.open_set
      (trivial_topology X) A",
```

```
lemma tri_all_is_closed :
  "!!A. A <= X ==>
    topological_space.closed_set
      (trivial_topology X) X A".
```

显然拓扑空间 (X, T) 的闭集属于 X , 这个性质在 Isabelle 中的表示如下:

```
lemma (in topological_space) close_sub_X:
  "[| closed_set s |] ==> s <= X".
```

上面这个引理的证明, 先引入开集和闭集的定义, 然后使用 `simp` 方法, 即可由 Isabelle 系统自动证明其结论. 从而, 把下面的两行代码加到上面引理的叙述之后就得到完整证明过程.

```
apply (simp add: closed_set_def open_set_def)
done.
```

命题 2.15^[57] 设 (X, T) 是一个拓扑空间, 若 A 是开集, B 是闭集, 则 $A - B$ 是开集, 而 $B - A$ 是闭集.

这个命题在 Isabelle 中也拆分成两个引理的形式来证明.

```
lemma (in topological_space)
  "[| open_set A ;
    closed_set B
  |] ==> open_set (A - B)",
```

```
lemma (in topological_space)
  "[| open_set A ;
    closed_set B
  |] ==> closed_set (B - A)".
```

它们的证明过程类似于命题 2.14, 这里列出前一个引理的证明.

```
lemma (in topological_space)
  "[| open_set A;
    closed_set B
  |] ==> open_set (A - B)"
  apply (frule_tac A = "A" in open_subset)
  apply (simp add: open_set_def closed_set_def 04)
  apply (subgoal_tac "A-B = A Int (X - B)")
  apply (simp)
  apply blast
done.
```

2.4 闭包与内部

一个拓扑空间 (X, T) 的一个子集, 可能既不是开集也不是闭集. 它包含的最大开集和包含这个子集的最小闭集, 分别称为它的内部和闭包. 下面的定义是很多拓扑学教科书中定义内部和闭包的常见形式.

定义 2.7^[57] 设 A 是拓扑空间 (X, T) 的一个子集, 包含于 A 的所有开集的并集称为 A 的内部, 记为 $\text{interior}(A)$; 包含 A 的所有闭集的交集称为 A 的闭包, 记为 $\text{closure}(A)$.

上述定义在 Isabelle 中表示成如下的形式.

```
definition (in topological_space)
  interior :: "'a set => 'a set"
  where
    "interior A = Union {a. (A <= X &
                           open_set a & A <= a)}",

definition (in topological_space)
  closure :: "'a set => 'a set"
  where
    "closure A = Inter {c. (A <= X &
                           closed_set a & A <= a)}".
```

由内部的定义可以看出, 任意集合 A 的内部是开集. 这一性质在 Isabelle 中表示和证明如下:

```
lemma (in topological_space) int_is_open:
  "A <= X ==> open_set (interior A)"
  apply (simp add: open_set_def)
  apply (simp add: interior_def)
  apply (simp add: open_set_def)
  apply (rule O5)
  apply blast
done.
```

同样地, 任意集合 A 的闭包是闭集. 这一性质在 Isabelle 中的表示形式如下:

```
lemma (in topological_space) closure_is_closed:
  "A <= X ==> closed_set (closure A)".
```

这个引理的证明与 lemma int_is_open 的证明类似, 将引入内部定义的语句换成引入闭包定义的语句即可. 内部和闭包有很多性质, 例如:

```
lemma (in topological_space) closure_is_closed:
  "[| open_set U ; U <= A |]
   ==> U <= interior A".
```

上面这个引理是说, 如果 U 是一个开集, 而且 $U \subseteq A$, 则由于 A 的内部是包含 A 的所有开集的并, 于是 U 是 $\text{interior } A$ 的子集. 这里略去证明. 闭集也有类似的性质, 在 Isabelle 中将其表示如下:

```
lemma (in topological_space) closure_is_closed:
  "[| closed_set C ; A <= C |]
   ==> closure A <= C".
```

下面的几个性质也只列出其 Isabelle 形式化表示. 它们各自的含义都可以从这些性质的表达式看得较为清楚, 这里不一一解释.

```
lemma (in topological_space) interior_trans:
  "A <= B ==> interior A <= interior B",

lemma (in topological_space) closure_trans:
  "A <= B ==> closure A <= closure B",
```

```

lemma (in topological_space) interior_is_open:
  "A = (interior A) ==> open_set A",

lemma (in topological_space) closure_is_closed:
  "A = (closure A) ==> closed_set A".

```

3 杨忠道定理的机械化证明

本节使用 Isabelle 形式语言描述一般拓扑学中的两个重要概念: 聚点和导集, 然后在计算机中验证著名的杨忠道定理.

3.1 导集的形式化

先看聚点和导集的自然语言表达的定义.

定义 3.1^[54] 设 (X, T) 是拓扑空间, A 是 X 的一个子集, $x \in X$. 如果对任意的 $U \in \mathcal{U}(x)$, 有

$$U \cap (A - \{x\}) \neq \emptyset,$$

则称 x 是 A 的聚点. A 的所有聚点的集合称为 A 的导集, 用 $d(A)$ 表示.

下面是聚点在 Isabelle 系统的形式化定义:

```

definition (in topological_space)
  cluster_point :: "'a => 'a set => bool"
  where
    "cluster_point x A == A <= X &
      x : X &
      ( ALL U : neighborhood_system x.
        U Int (A - {x}) ~= {} )".

```

导集在 Isabelle 中的定义形式如下:

```

definition (in topological_space)
  derived_set :: "'a set => 'a set"
  where
    "derived_set A == {x. (cluster_point x A)}".

```

上面, 将 `cluster_point` (聚点) 定义成形如 “ $'a \Rightarrow 'a \text{ set} \Rightarrow \text{bool}$ ” 的函数, 而 `derived_set` (导集) 定义成形如 “ $'a \text{ set} \Rightarrow 'a \text{ set}$ ” 的函数 (即集合到集合的函数). 将聚点形式化定义成判定函数, 可以简化证明过程, 降低证明的复杂度.

下面的引理 `sp_notin_dset_pre` 证明对于任意拓扑空间 (X, T) 和点 $x \in X$, 有 $x \notin d(x)$, 即 x 不属于其单点集的导集. 为了简便, 这里不另外给出其自然语言形式的表述. 引理证明分成两步. Isabelle 中的证明脚本如下.

```

lemma (in topological_space) sp_notin_dset_pre:
  assumes a1: "x : derived_set {x}"
  shows False

```

```

proof-
  from a1 have s1: "{x} <= X"
    by (simp add: derived_set_def cluster_point_def)
  from a1 have s2: "x : X"
    by (simp add: derived_set_def cluster_point_def)
  from a1 have s3: "ALL U : neighborhood_system x.
    U Int ({x} - {x}) ~= {}"
    by (simp add: derived_set_def
      cluster_point_def)
  have s4: "U Int ({x} - {x}) = {}" by blast
  from s1 s3 s4 shows ?thesis
    by (simp add: nbs_not_empty)
qed,

lemma (in topological_space) sp_notin_dset:
  "x ~: derived_set {x}"
  apply (rule_tac Q = "False" in contrapos_np)
  apply simp+
  apply (rule_tac x = x in sp_notin_dset_pre)
  apply assumption
done.

```

一个简单的性质是任意集合 A 的导集必然是 A 的闭包的子集, 即 $d(A) \subset \text{closure}(A)$. 下面是这个性质的形式化及其证明.

```

lemma (in topological_space) derive_subset_closure:
  assumes a1 "A <= X"
  assumes a2 "x ~: closure A"
  assumes a3 "x : derive_set A"
  shows False

```

```

proof-
  from a2 to have s1:
    "EX U : neighborhood_system x.
      U Int A = {}"
    by (simp closure_def)
  from s1 to have s2:
    "U Int (A - {x}) = {}"
    by auto
  from s2 to have s3:
    "x ~: derive_set A"
    by (simp derive_set_def)
  from a3 s3 shows ?thesis by auto
qed
done.

```

由上面的引理立即看出, 对任何集合 $A \subseteq X$ 都有关系式 $A \cup d(A) \subseteq \text{closure}(A)$. 下面的命题是这一结果的逆命题. 它反映拓扑空间导集和闭包之间的一个更紧密的联系.

命题 3.1 如果 (X, T) 是拓扑空间, 则对任意 $A \subset X$, 有 $\text{closure}(A) \subseteq A \cup d(A)$.

以下是命题 3.1 的逆否命题, 即 $x \notin A \wedge x \notin dZ(A) \Rightarrow x \in \text{closure}(A)$ 的形式化表达和证明.

```

lemma (in topological_space) closure_set_derive:
  assumes a1: "A <= X"
  assumes a2: "x ~: A"
  assumes a3: "x ~: derived_set A"
  shows a4: "x ~: (closure A)"

proof-
  from a3 have s2:
    "EX U : neighborhood_system x.
     U Int (A-{x}) = {}"
    by (simp derived_set_def)
  from s2 have s3: "(U-{x}) Int (A-{x}) = {}" by auto
  from s2 have s4: "(U-{x}) Int {x} = {}" by auto
  from s3 s4 have s5: "(U-{x}) Int A = {}" by auto
  from a2 s5 have s6: "U Int A = {}" by auto
  from a1 s6 to have a4 by (simp closure_def)
qed
done.

```

下面的命题将在杨忠道定理的证明中引用. 这个命题是命题 3.1 的简单推论, 所以我们只给出它的人工证明, 省略其篇幅较长的形式化证明.

命题 3.2 设 (X, T) 是拓扑空间, 集合 $A \subset X$. 若 $d(A) \subset A$, 则 A 是闭集.

证明 由 $d(A) \subset A$ 和 $\text{closure}(A) = A \cup d(A)$ 立即得到 $\text{closure}(A) = A$. 从而 A 是闭集. $\square$

杨忠道定理的证明还需要下面的命题.

命题 3.3 设 (X, T) 是拓扑空间, X 的子集 A 是闭集, 则有 $d(A) \subseteq A$.

这个命题等价于, 若 A 是闭集, 则不存在 x 使得 $x \in d(A)$ 而 $x \notin A$. 其形式化表达如下:

```

lemma (in topological_space) closed_is_derived_1:
  assumes a1: "closed_set A"
  and a2: "x : derived_set A"
  and a3: "x ~: A"
  shows False.

```

证明过程如下:

```

proof-
  from a2 have s2: "A <= X"
    by (auto simp: derived_set_def cluster_point_def)
  from a2 have s3: "x : X"
    by (auto simp: derived_set_def cluster_point_def)
  from a2 have s4:
    "ALL U : neighborhood_system x.
     U Int (A - {x}) ~= {}"
    by (auto simp: derived_set_def cluster_point_def)
  from a3 s2 s3 have s5: "x : X - A"
    by blast
  from a1 s2 have s6: "open_set (X - A)"
    by (simp add: closed_set_def)

```

```

from s6 have s7: "ALL y : X - A.
  EX B : neighborhood_system y.
  y : B & B <= X - A "
by (simp add: open_nb_ex_a
      nbs_in
      neighborhood_system_def)
from s5 s7 have s8:
  "EX B : neighborhood_system x.
  x : B & B <= X - A" by auto
from s8 obtain B where
  s9: "B : neighborhood_system x" and
  s10: "x : B" and
  s11: "B <= X - A" by blast
from s11 have s12: "B Int A = {}" by blast
from s12 have s13: "B Int (A - {x}) = {}" by blast
from s4 s9 s13 have s14: "False" by auto
from a3 s14 shows ?thesis by auto
qed.

```

有如上引理, 即可以证明当 A 是闭集, 就有 $x \in \text{derived_set } A$ 蕴涵 $x \in A$, 即如下引理.

```

lemma (in topological_space) closed_is_derived_2:
  assumes a1: "closed_set A"
  and a2: "x : derived_set A"
  shows "x : A"
proof-
  have a0: "x ~: A ==> False"
  by (simp add: closed_is_derived_1)
  from a0 shows ?thesis by auto
qed.

```

下面的引理说明, 当 A 是闭集, $d(A)$ 包含在集合 A 中. 虽然从自然语言来理解, 这个结论可以由 lemma `closed_is_derived_2` 直接导出, 但在形式化系统 Isabelle 中, 这一步需要进行推导.

```

lemma (in topological_space) closed_is_derived:
  assumes a1: "closed_set A"
  shows "derived_set A <= A"
proof-
  from a1 have s0: "!!x. x : derived_set A ==> x : A"
  proof-
    fix x assumes a2: "x : derived_set A"
    shows s1: "x : A"
    proof-
      from a1 a2 shows ?thesis
      by (auto simp: closed_is_derived_2)
    qed
  qed
  from s0 shows ?thesis by auto
qed.

```


3.2 杨忠道定理的机器证明

杨忠道定理断言, 通过单点集的导集是否为闭集, 即可判断任意集合的导集是否为闭集. 以下是它的自然语言表达.

定理 3.1 (杨忠道定理) 拓扑空间的每一个子集的导集是闭集当且仅当每一个单点集的导集是闭集.

先给出杨忠道定理的一个手工证明, 后面将遵循它建立杨忠道定理的形式化证明. 为此, 把杨忠道定理拆分成如下两个命题.

命题 3.4 (充分性) 如果拓扑空间的每一个子集的导集是闭集, 则这个拓扑空间中的每一个单点集的导集是闭集.

命题 3.5 (必要性) 如果拓扑空间的每一个单点集的导集是闭集, 则这个拓扑空间中的每一个子集的导集是闭集.

充分性部分是显然的. 下面是必要性的一個手工证明. 后面的形式化证明将遵循这里的手工证明. 为了理解形式化证明的细致性, 下面的手工证明也尽量把每一个推理环节写出来.

命题 3.5 的证明 设拓扑空间 (X, T) 的每一个单点集的导集是闭集. 根据命题 3.2 可知, 只要证明, 对于任意 $A \subseteq X$, $d(d(A)) \subseteq d(A)$ 即可. 为与后面的形式化证明对照, 下面在每一个断言后面标示编号.

假设 $B = d(A)$, $x \in d(d(A)) = d(B)$, U 是 x 的任意一个开邻域. (S0)

根据定义 3.1 可知, 由于 $x \in d(B)$, 对 x 的这个开邻域 U , 必然有 $U \cap (B - \{x\}) \neq \emptyset$, 即

$$U \cap (d(A) - \{x\}) \neq \emptyset. \quad (S1)$$

由命题 3.5 的假设, $d(\{x\})$ 是闭集. (S2)

由 lemma `sp_notin_dset` 可知, $x \notin d(\{x\})$. (S3)

令 $V = U - d(\{x\})$, 则 V 是 x 的开邻域. 这里根据 (S3) 和命题 2.15: 开集 - 闭集 = 开集. (S4)

由于 $x \in d(B)$, V 是 x 的开邻域, 根据定义 3.1, 有 $V \cap (B - \{x\}) = V \cap (d(A) - \{x\}) \neq \emptyset$, (S5)

故存在 $y \in V \cap (d(A) - \{x\})$, 因为语句 (S5) 和选择公理: 集合不空, 必可从其中取一元素. (S6)

因为 (S6) 和 (S4), $y \in V \cap (d(A) - \{x\}) = (U - d(\{x\})) \cap (d(A) - \{x\})$ 蕴涵

$$y \in V \text{ 并且 } y \notin d(\{x\}) \text{ 及 } y \neq x. \quad (S7)$$

根据 (S7) 中的 $y \notin d(\{x\})$ 以及定义 3.1 可知, 必存在 y 的邻域 W , 使得 $W \cap (\{x\} - \{y\}) = \emptyset$, (S8)

进而由于 $y \neq x$, 有 $W \cap (\{x\}) = \emptyset$, 由此可知 $x \notin W$. (S9)

因为 $W, V \in \mathcal{U}(y)$, 令 $K = W \cap V$, 则 K 是 y 的一个开邻域, 因为开邻域的交是开邻域. (S10)

由 (S6) 和 $y \neq x$ 可知, $y \in d(A)$, 进而由导集的定义可知, 存在 $t \in K \cap (A - \{y\}) \neq \emptyset$. (S11)

由 $t \in K \subseteq W$, 可知 $t \neq x$. 注意, 这里引用断言 (S9), $x \notin W$. (S12)

因此有 $t \in U \cap (A - x)$, 这一步是因为 $t \in K \cap A = (W \cap V) \cap A \subseteq V \cap A = (U - d(\{x\})) \cap A$, 以及 $t \neq x$. (S13)

故

$$U \cap (A - \{x\}) \neq \emptyset.$$

进而, 由于 U 是 x 的任意一个邻域, 根据定义 3.1 得知, 必有 $x \in d(A)$. (S14)

所以,

$$d(d(A)) \subseteq d(A).$$

这一步是由 (S0) 和 (S14) 得到的. (S15)

根据命题 3.2, $d(A)$ 是闭集. 此即命题 3.5 的结论. (S16)

证毕. □

在 Isabelle 形式系统中证明杨忠道定理, 还需要如下 4 个引理:

```
lemma subset_mui :
  "[/ A <= X ; B <= X /] ==> (A - B) = A Int (X - B)"
  apply blast
done.
```

这个引理断言 $A \setminus B = A \cap (X \setminus B)$ 对于所有子集合 $A \subset X$ 和 $B \subset X$ 成立. 这个结果在集合论中是一个显然的结论.

```
lemma (in topological_space) open_is_onb:
  "[/ open_set A; x : A /] ==>
    A : open_neighborhood_system x"
  apply (simp add: open_neighborhood_system_def
    open_neighborhood_def
    open_set_def)
done.
```

上面的引理指出, 拓扑空间 (X, T) 的任何一个包含点 x 的开集 A , 必定是 x 的邻域系的成员. 下面的性质是说, 点 x 的邻域系的任何成员 A 一定包含点 x .

```
lemma (in topological_space) onbs_in:
  "A : open_neighborhood_system x ==> x : A"
  apply (simp add: open_neighborhood_system_def
    open_neighborhood_def)
done,
```

```
lemma (in topological_space) obb_mui_closed_onb:
  "[/ U : open_neighborhood_system x ;
    closed_set A; x ~: A /] ==>
    U - A : open_neighborhood_system x"
  apply (frule open_nbs_o)
  apply (frule_tac B=A in open_mui_closed_is_open)
  apply assumption
  apply (subgoal_tac "x : (U - A)")
  apply (simp add: open_is_onb)
  apply (simp add: onbs_in)
done.
```

上面的引理指出, 若 U 是 x 的一个开邻域, 闭集合 A 不包含 x , 则集合 $U \setminus A$ 也是 x 的一个开邻域.

现在可以写出 Isabelle 中杨忠道定理的形式化表述及其证明的详细过程.

```
lemma (in topological_space) yangzhongdao_thm:
  assumes a0: "A <= X"
    and a1: "ALL x : X. closed_set (derived_set {x})"
  shows "derived_set(derived_set A) <= derived_set A"
```

```

proof-
  from a1 have s1: "!!x. x : derived_set(derived_set A)
    ==> x : derived_set A"
  proof-
  fix x
  assumes a2: "x : derived_set(derived_set A)"
  shows "x : derived_set A"
  proof-
    from a2 have s1: "derived_set A <= X"
      by (simp add: derived_set_def
        cluster_point_def)
    from a2 have s2: "x : X"
      apply (simp add: derived_set_def
        cluster_point_def) done
    from a2 have s3: "ALL U: neighborhood_system x.
      U Int ((derived_set A) - {x}) ~= {}"
      by (simp add: derived_set_def
        cluster_point_def)
    from s3 have s4: "ALL U: open_neighborhood_system x.
      U Int ((derived_set A) - {x}) ~= {}"
      apply (drule_tac x="x" and
        P="λ UU. (λ xx. (UU Int ((derived_set A) - {xx}) ~= {}))"
        in open_nbs_intro)
      apply assumption done
    have s5:
      "!!U. U : neighborhood_system x ==> U Int (A - {x}) ~= {}"
    proof-
      fix U1
      assumes a3: "U1 : neighborhood_system x"
      shows "U1 Int (A - {x}) ~= {}"
    proof-
      from a3 have s6:
        "EX U. U : open_neighborhood_system x"
        apply (auto simp: nb_ex_onb) done
      from s6 obtain U where s7:
        "U : open_neighborhood_system x"
        apply auto done
      from s7 s4 a3 have s8:
        "U Int ((derived_set A) - {x}) ~= {}"
        by auto
      let ?V = "U - (derived_set {x})"
      from s2 have s9: "?V : open_neighborhood_system x"
        apply (rule_tac A = "(derived_set {x})"
          in obb_mui_closed_onb)
        apply (simp add: s7)
        apply (simp add: a1)
        apply (simp add: sp_notin_dset) done
      from s4 s9 have s10:
        "?V Int ((derived_set A) - {x}) ~= {}"
        by blast
      from s10 have s11:
        "EX y. y : ?V Int ((derived_set A) - {x})"

```

```

      by auto
    from s11 obtain y where s12: "y : ?V"
      and s13: "y : ((derived_set A) - {x})"
      by blast
    from s12 s13 have s14: "y ~= x"
      by auto
    from s12 s13 s14 have s15:
      "y ~: (derived_set {x})" by blast
    from s2 have s16: "{x} <= X" by blast
    from s2 s3 s15 s16 have s17:
      "EX W : open_neighborhood_system x. W Int ({x} - {y}) = {}"
      by auto
    from s17 s14 obtain W where s18:
      "W : open_neighborhood_system x"
      and s19: "W Int {x} = {}" by auto
    from s19 have s20: "x ~: W" by blast
    from s18 s9 have s21:
      "W Int ?V : (open_neighborhood_system y)"
      by auto
    from s21 have s22: "W Int ?V : (neighborhood_system y)"
      by auto
    from s13 s14 have s23: "y : derived_set A" by auto
    from s23 have s24: "ALL K : neighborhood_system y.
      K Int (A - {y}) ~= {}"
      by (auto simp: derived_set_def cluster_point_def)
    from s22 s24 have s25: "(W Int ?V) Int (A - {y}) ~= {}"
      by blast
    from s25 have s26: "EX t. t : (W Int ?V) Int (A - {y})"
      by auto
    from s26 obtain t where s27: "t : (W Int ?V) Int (A - {y})"
      by auto
    from s27 have s28: "t : W" by auto
    from s27 s20 have s29: "t ~= x" by auto
    from s29 s27 s20 have s30: "t : U Int (A - {x})" by auto
    from s30 have s31: "U Int (A - {x}) ~= {}" by auto
    from s31 shows ?thesis by auto
  qed
qed
from s5 a0 s2 shows ?thesis
  by (simp add: derived_set_def cluster_point_def)
qed
shows ?thesis by auto
qed.

```

至此, 完成了杨忠道定理的形式化证明.

4 基于形式化系统的数学机械化和定理机器证明的全自动化

前面介绍了基于 Isabelle 平台的点集拓扑学机械化的一项工作, 我们采用高阶逻辑来表达一般拓

扑学中诸如拓扑空间、开集、闭集、邻域、闭包和连续函数等抽象概念, 将这些抽象概念表示为可计算函数的形式, 使其能够在 Isabelle 系统中通过 λ -演算实现演绎推理, 并通过构造证明策略序列, 以逐步求精的方式完成了杨忠道定理的机械化证明任务. 这一工作, 建立了点集拓扑学定理证明的机械化的一个基础. 这一工作, 最初是本文作者之一王建林在博士学位论文^[59]中完成的. 事实上, 王建林的博士学位论文还证明了点集拓扑学的一些其他基本定理, 包括黏结引理和 Kuratowski 定理 (设 c 为 X 上的闭包算子, T 为所有满足 $A^c = A$ 的子集合组成的族, 则 T 的所有成员的补集组成的集合族是 X 上的一个拓扑). 考虑到点集拓扑学是拓扑学的机械化的第一步, 而杨忠道定理深为国内拓扑学的广大学习者所喜欢, 因此, 本文对文献^[59]中与杨忠道定理证明相关的内容加以整理和改进, 希望发表之后吸引更多的研究者进一步研究拓扑学的机械化.

最近几年, 国内外学者用 Isabelle 证明了大量重要的数学定理, 如 2018–2019 年, 国内学者完成的有 Li^[60] 的 Budan-Fourier 定理形式化、Zhan^[61,62] 的 Auto2 prover、Xu 等^[63] 的“通用图灵机器”的形式化和 Liu 等^[64] 完成的量子 Hoare 逻辑等. 本文作者期待将来有更多的研究者应用 Isabelle 和 Coq 等形式化工具, 开展更多数学领域的机械化研究.

正如在前面两节中看到的, 使用 Isabelle 形式化语言进行数学机械化, 在证明过程中要使用大量的方法和技巧来构造证明策略, 如引入策略、重写策略和分类策略等, 逐步求精的过程也包含一定技巧. 所以, 用形式化语言表达和证明数学定理的数学机械化, 还处于人机交互的半自动化阶段. 未来随着更多研究的出现, 将证明经验总结和分类, 人们将会更大限度发挥计算机硬件性能, 在辅助证明器中更多运用大范围比对和搜索, 逐步提高定理机器证明的自动化程度.

与目前人工智能其他研究领域 (如破解密码、深度神经网络、深度学习和区块链等) 相比, 定理机器证明这些更深层的人工智能研究, 从耗费的计算资源来说, 只是很小一部分. 例如, 验证 Kepler 猜想的 Flyspeck^[65] 项目, 其中包含的 43,078 个线性规划, 在 2.4 GHz 处理器的计算机上完成验证工作, 用了大约 15 个小时, 另外, 证明脚本的形式化验证, 大约用了 5 小时. 人工智能的终极目标包括让机器理解人类的思想发明, 并且有自我意识地延伸人类知识, 把人类的知识带到人类不能到达的时空 (关于这一设想及为之进行的努力, 参见著名荷兰数学家 Freudenthal^[66] 的 Lincos). 在这个意义上, 可以预期 Isabelle 等形式化工具将在数学机械化研究中发挥重要作用.

致谢 作者感谢时任华东师范大学软件学院计算机理论研究所所长的杨路教授对于本文研究工作给予的支持, 感谢当时一起在华东师范大学软件学院工作的同事郁文生、吴敏、赵世忠和陈良育等的有启发性的讨论, 谨以此文纪念逝去的岁月. 感谢匿名审稿人对于本文初稿提出的非常有价值的修改建议.

参考文献

- 1 Yang L, Xia S H. Automated proving for a class of constructive geometric inequalities (in Chinese). *Chinese J Comput*, 2003, 26: 769–778 [杨路, 夏时洪. 一类构造性几何不等式的机器证明. *计算机学报*, 2003, 26: 769–778]
- 2 Yang L, Xia B C. *Automated Inequality Proving and Discovering*. Singapore: World Scientific, 2016 [杨路, 夏壁灿. *不等式机器证明与自动发现*. 北京: 科学出版社, 2008]
- 3 Collins G E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: *Automata Theory and Formal Languages*. Lecture Notes in Computer Science, vol. 33. Berlin: Springer-Verlag, 1975, 134–183
- 4 Zeng Z B, Zhang J Z. A mechanical proof to a geometric inequality of Zirakzadeh through rectangular partition of polyhedra (in Chinese). *J Systems Sci Math Sci*, 2010, 30: 1430–1458 [曾振柄, 张景中. 基于矩形区域剖分的不等式机器证明方法—以 Zirakzadeh 的一个几何不等式为例. *系统科学与数学*, 2010, 30: 1430–1458]
- 5 Yang L. Solving harder problems with lesser mathematics. In: *Proceedings of the 10th Asian Technology Conference in Mathematics*. Blacksburg: ATCM Inc., 2005, 37–46
- 6 Kelley J L. *General Topology*. New York: Springer-Verlag, 1955

- 7 Xiong J C. Lectures on Point-Set Topology (in Chinese). Beijing: Higher Education Press, 1981 [熊金城. 点集拓扑讲义. 北京: 高等教育出版社, 1981]
- 8 Pu B M. Topology (in Chinese). Beijing: Higher Education Press, 1985 [蒲保明. 拓扑学. 北京: 高等教育出版社, 1985]
- 9 Han G. History studies on two important theorems in topology (in Chinese). PhD Thesis. Hohhot: Inner Mongolia Normal University, 2016 [韩刚. 拓扑学中两个重要定理的历史研究. 博士学位论文. 呼和浩特: 内蒙古师范大学, 2016]
- 10 Gelernter H. Realisation of a geometry-theorem proving machine. In: Proceedings of the 1959 International Conference on Information Processing. München: Oldenbourg Verlag, 1960, 273–282
- 11 Robinson J A. Review: H. Gelernter, theorem proving by machine; H. Gelernter, realization of a geometry theorem proving machine. J Symbolic Logic, 1967, 32: 522–523
- 12 Wang H. Toward mechanical mathematics. IBM J Res Dev, 1960, 4: 2–22
- 13 Slagle J. A heuristic program that solves symbolic integration problems in freshman calculus. PhD Thesis. Cambridge: MIT, 1961
- 14 Moses J. Symbolic integration. PhD Thesis. Cambridge: MIT, 1967
- 15 Patterson T N L. Algorithm 468: Algorithm for automatic numerical integration over a finite interval. Commun ACM, 1973, 16: 694–699
- 16 Kaltofen E. Factorization of polynomials. In: Computer Algebra: Symbolic and Algebraic Computation. Wien: Springer-Verlag, 1982, 95–113
- 17 Wu W T. On the decision problem and the mechanization of theorem-proving in elementary geometry (in Chinese). Scientia Sinica, 1977, 7: 507–516 [吴文俊. 初等几何判定问题与机械化证明. 中国科学, 1977, 7: 507–516]
- 18 Wu W T. Basic Principles of Mechanical Theorem Proving in Elementary Geometries (in Chinese). Beijing: Science Press, 1984 [吴文俊. 几何定理机器证明的基本原理: 初等几何部分. 北京: 科学出版社, 1984]
- 19 Wu W T. Mathematics Mechanization (in Chinese). Beijing: Science Press, 2003 [吴文俊. 数学机械化. 北京: 科学出版社, 2003]
- 20 Tarski A. A Decision Method for Elementary Algebra and Geometry, 2nd ed. Oakland: University of California Press, 1951
- 21 Cox D, Little J, O'shea D. Ideals, Varieties and Algorithms. An Introduction to Computational Algebraic Geometry and Commutative Algebra, 2nd ed. Undergraduate Texts in Mathematics. New York: Springer-Verlag, 1997
- 22 Gel'fand I M, Kapranov M M, Zelevinsky A V. Discriminants, Resultants, and Multidimensional Determinants. Mathematics: Theory & Applications. Boston: Birkhäuser, 1994
- 23 Hong J W. Can we prove geometry theorems by examples (in Chinese)? Sci Sinica Ser A, 1986, 16: 234–242 [洪加威. 能用例证法来证明几何定理吗? 中国科学 A 辑, 1986, 16: 234–242]
- 24 Zhang J Z, Yang L. An introduction to the parallel numerical method and the one-example illustration method of mechanical theorem proving (in Chinese). Math Practice Theory, 1989, 20: 34–43 [张景中, 杨路. 定理机械化证明的数值并行法及单点例证法原理概述. 数学的实践与认识, 1989, 20: 34–43]
- 25 Zhang J Z, Gao X S. Automated generation of human readable machine proof of geometry theorems (in Chinese). Chinese J Comput, 1995, 18: 380–393 [张景中, 高小山. 几何定理可读证明的自动生成. 计算机学报, 1995, 18: 380–393]
- 26 Yang L, Gao X S, Chou S C, et al. Automated production of readable proofs for theorems in non-Euclidean geometries. In: Automated Deduction in Geometry. Lecture Notes in Artificial Intelligence, vol. 1360. Berlin-Heidelberg: Springer, 1996, 171–188
- 27 Wang D M, Xia B C. Computer Algebra (in Chinese). Beijing: Tsinghua University Press, 2004 [王东明, 夏壁灿. 计算机代数. 北京: 清华大学出版社, 2004]
- 28 Corless R M, Watt S M, Zhi L. QR factoring to compute the GCD of univariate approximate polynomials. IEEE Trans Signal Process, 2004, 52: 3394–3402
- 29 Kaltofen E, Yang Z F, Zhi L H. Approximate greatest common divisors of several polynomials with linearly constrained coefficients and singular polynomials. In: Proceedings of the 2006 International Symposium on Symbolic and Algebraic Computation. New York: Association for Computing Machinery, 2006, 169–176
- 30 Yang L, Zeng Z B, Zhang W N. Differential elimination with Dixon resultants. Appl Math Comput, 2012, 218: 10679–10690
- 31 Zhang Z Y, Yuan C M, Gao X S. Matrix formulae of differential resultant for first order generic ordinary differential

- polynomials. In: Computer Mathematics. Berlin-Heidelberg: Springer, 2014, 479–503
- 32 Paulson L C. Isabelle: A Generic Theorem Prover. Berlin-Heidelberg: Springer-Verlag, 1994
- 33 Nipkow T, Paulson L C, Wenzel M. Isabelle/HOL: A proof Assistant for Higher-Order Logic. Heidelberg: Springer-Verlag, 2002
- 34 Nipkow T. Structured proofs in Isar/HOL. In: Types for Proofs and Programs. Lecture Notes in Computer Science, vol. 2646. Berlin-Heidelberg: Springer-Verlag, 2003, 259–278
- 35 Archive of formal proofs. <https://www.isa-afp.org>
- 36 Bertot Y, Casteran P. Interactive Theorem Proving and Program Development: Coq'Art: The Calculus of Inductive Constructions. Berlin-Heidelberg: Springer-Verlag, 2004 [Bertot Y, Casteran P. 交互式定理证明与程序开发—Coq 归纳构造演算的艺术. 顾明, 译. 北京: 清华大学出版社, 2010]
- 37 The Coq package index. <https://coq.inria.fr/opam/www>
- 38 Yu W S, Sun T Y, Fu Y S. A machine proof system for axiomatic set theory (in Chinese). Beijing: Science Press, 2020 [郁文生, 孙天宇, 付尧顺. 公理化集合论机器证明系统. 北京: 科学出版社, 2020]
- 39 Formalizing 100 theorems. <https://www.cs.ru.nl/~freek/100>
- 40 Hales T C. Formal proof. Notices Amer Math Soc, 2008, 55: 1370–1380
- 41 Gonthier G. Formal proof—the four color theorem. Notices Amer Math Soc, 2008, 55: 1382–1393
- 42 Narboux J. The Chou, Gao and Zhang area method. <https://github.com/coq-contribs/area-method>
- 43 Gènevaux J D, Narboux J, Schreck P. Formalization of Wu's simple method in Coq. In: Certified Programs and Proofs. Lecture Notes in Computer Science, vol. 7086. Berlin-Heidelberg: Springer, 2011, 71–86
- 44 Kennedy A, Benton N, Jensen J B, et al. Coq: The world's best macro assembler? In: Proceedings of the 15th Symposium on Principles and Practice of Declarative Programming. New York: Association for Computing Machinery, 2013, 13–24
- 45 Bernard S. Formalization of the Lindemann-Weierstrass theorem. In: Interactive Theorem Proving. Lecture Notes in Computer Science, vol. 10499. Cham: Springer, 2017, 65–80
- 46 Hölzl J, Immler F, Huffman B. Type classes and filters for mathematical analysis in Isabelle/HOL. In: Interactive Theorem Proving. Lecture Notes in Computer Science, vol. 7998. Berlin-Heidelberg: Springer, 2013, 279–294
- 47 Gammie P, Gioiosa G. The Kuratowski closure-complement theorem. https://www.isa-afp.org/browser.info/current/AFP/Kuratowski_Closure_Complement/outline.pdf
- 48 Kuratowski K. Sur l'opération $\bar{A}$ de l'analysis situs. Fund Math, 1922, 3: 182–199
- 49 Friedrich S. The topology of lazy lists. <https://www.isa-afp.org/entries/Topology.html>
- 50 Library Coq.Reals.Rtopology. <https://coq.inria.fr/library/Coq.Reals.Rtopology.html>
- 51 Grabowski A. Solving two problems in general topology via types. In: Types for Proofs and Programs. Lecture Notes in Computer Science, vol. 3839. Berlin-Heidelberg: Springer, 2004, 138–153
- 52 Wenzel M. Isabelle/Isar—a versatile environment for human readable formal proof documents. PhD Thesis. München: Technische Universität München, 2002
- 53 Wenzel M. The Isabelle/Isar reference manual. <https://isabelle.in.tum.de/doc/isar-ref.pdf>
- 54 Munkres J R. Topology. Upper Saddle River: Prentice Hall, 2000
- 55 Kammüller F, Wenzel M, Paulson L C. Locales: A sectioning concept for Isabelle. In: Theorem Proving in Higher Order Logics. Lecture Notes in Computer Science, vol. 1690. Berlin-Heidelberg: Springer, 2009, 149–165
- 56 Ballarin C. Locales and locale expressions in Isabelle/Isar. In: Types for Proofs and Programs. Lecture Notes in Computer Science, vol. 3085. Berlin-Heidelberg: Springer, 2004, 34–50
- 57 Adams C C, Franzosa R D. Introduction to Topology: Pure and Applied. Upper Saddle River: Pearson Prentice Hall, 2008
- 58 Zhu P Y, Lei Y B. An Introduction to Topology (in Chinese). Beijing: Science Press, 2009 [朱培勇, 雷银彬. 拓扑学导论. 北京: 科学出版社, 2009]
- 59 Wang J L. Studies on automated theorem proof and mechanization of general topology based on Isabelle (in Chinese). PhD Thesis. Shanghai: East China Normal University, 2012 [王建林. 基于 Isabelle 平台的一般拓扑学机械化及自动定理证明研究. 博士学位论文. 上海: 华东师范大学, 2012]
- 60 Li W D. The Budan-Fourier theorem and counting real roots with multiplicity. https://www.isa-afp.org/entries/Budan_Fourier.html
- 61 Zhan B H. Auto2 prover. https://www.isa-afp.org/entries/Auto2_HOL.html
- 62 Zhan B H. Verifying imperative programs using Auto2. https://www.isa-afp.org/entries/Auto2_Imperative_HOL.html

- 63 Xu J, Zhang X Y, Urban C, et al. Universal Turing machine. https://www.isa-afp.org/entries/Universal_Turing_Machine.html
- 64 Liu J Y, Zhan B H, Wang S L, et al. Quantum Hoare logic. <https://www.isa-afp.org/entries/QHLProver.html>
- 65 Hales T, Adams M, Bauer G, et al. A formal proof of the Kepler conjecture. *Forum Math Pi*, 2017, 5: e2
- 66 Freudenthal H. *Lincos: Design of a Language for Cosmic Intercourse. Part I*. Amsterdam: North-Holland, 1960

A mechanical proof of the C. T. Yang's Theorem related to a property of derived sets in general topology

Zhenbing Zeng, Jianlin Wang, Zhengfeng Yang & Hidetsune Kobayashi

Abstract In this paper, we presented a formalization of the basic concepts of topological spaces including open sets, neighborhoods, closed sets and closures using the higher order logic interactive assistant tool Isabelle, and gave a mechanical proof of the C. T. Yang's Theorem which states that in any topological space (X, T) , if the derived set of $\{x\}$ is closed for all the points $x \in X$, then the derived set of any subset $S \subset X$ is also closed.

Keywords topological space, open set, closed set, derived set, C. T. Yang's Theorem, machine proof

MSC(2010) 54B05, 68T15

doi: 10.1360/SSM-2019-0301