软件学报 ISSN 1000-9825, CODEN RUXUEW
[doi: 10.13328/j.cnki.jos.007201] [CSTR: 32375.14.jos.007201]

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

良构图类上的模型检测问题*

刘国航, 陈翌佳

(上海交通大学 电子信息与电气工程学院, 上海 200240)

通信作者: 刘国航, E-mail: ahoper@sjtu.edu.cn

摘　要: 图上的诸多计算问题都是 NP 难问题, 因此经常会将问题限定在一些特定的图类上. 这类方法在过去的几十年间收获了大量特定图类 (如度有界图类、树宽有界图类、平面图类等) 上的高效算法, 其中很大一部分都能统一到算法元定理的框架下. 算法元定理是一类通用的结论, 主要描述模型检测问题 (即判定结构的逻辑性质) 的高效算法. 现有的算法元定理主要基于现代结构图论, 并且大多研究固定参数易解算法,即参数复杂性意义下的高效算法. 在许多良构的图类上, 一些常见逻辑 (如一阶逻辑和一元二阶逻辑) 的模型检测问题是固定参数易解的. 由于不同逻辑的表达能力不同, 不同图类上的模型检测问题的易解性也有显著的区别, 因此探索易解的最大范围也是算法元定理研究的重要课题. 研究表明, 一阶逻辑模型检测问题的易解性与图的稀疏性密切关联. 经过数十年的努力, 目前学界对于稀疏图类的认识已经较为成熟, 近年的研究重心逐渐转向一些良构的稠密图类, 研究也面临着更多的挑战. 目前在稠密图类上已经得到了若干深刻的算法元定理, 相关的探索仍在继续. 将全局性地介绍算法元定理领域的发展, 旨在为国内的相关研究提供一些线索和助力.

关键词: 模型检测; 算法元定理; 稀疏性; 结构图论; 参数复杂性

中图法分类号: TP311

中文引用格式: 刘国航, 陈翌佳. 良构图类上的模型检测问题. 软件学报. http://www.jos.org.cn/1000-9825/7201.htm

英文引用格式: Liu GH, Chen YJ. Model-checking Problem on Well-structured Graph Classes. Ruan Jian Xue Bao/Journal of Software (in Chinese). http://www.jos.org.cn/1000-9825/7201.htm

Model-checking Problem on Well-structured Graph Classes

LIU Guo-Hang, CHEN Yi-Jia

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

Abstract: Many computational problems on graphs are NP hard, so a natural strategy is to restrict them to some special graphs. This approach has seen many successes in the last few decades, and many efficient algorithms have been designed for problems on graph classes including graphs of bounded degree, bounded tree-width, and planar graphs, to name a few. As a matter of fact, many such algorithmic results can be understood in the framework of the so-called algorithmic meta-theorems. They are general results that provide efficient algorithms for decision problems of logic properties on structural graphs, which are also known as model-checking problems. Most existing algorithmic meta-theorems rely on modern structural graph theory, and they are often concerned with fixed-parameter tractable algorithms, i.e., efficient algorithms in the sense of parameterized complexity. On many well-structured graphs, the model-checking problems for some natural logics, e.g., first-order logic and monadic second-order logic, turn out to be fixed-parameter tractable. Due to varying expressive power, the tractability of the model-checking problems of those logics have huge differences as far as the underlying graph classes are concerned. Therefore, understanding the maximum graph classes that admit efficient model-checking algorithms is a central question for algorithmic meta-theorems. For example, it has been long known that efficient model-checking of first-order logic is closely related to the sparsity of input graphs. After decades of efforts, our understanding of sparse graphs are fairly complete now. So much of the current research has been focused on well-structured dense graphs, where challenging open problems are abundant. Already there are a few deep algorithmic meta-theorems proved for dense graph classes, while the research frontier is still

* 基金项目: 国家自然科学基金面上项目 (62372291)
收稿时间: 2023-07-31; 修改时间: 2023-12-14, 2024-04-10; 采用时间: 2024-04-11; jos 在线出版时间: 2024-06-20

expanding. This survey aims to give an overview of the whole area in order to provide impetus of the research of algorithmic meta-theorem in China.

Key words: model-checking; algorithmic meta-theorem; sparsity; structural graph theory; parameterized complexity

理论计算机科学中的诸多经典问题都可由一些常见的逻辑描述, 例如支配集 (dominating set) 问题可由一阶逻辑 (first-order logic, FO) 描述, 3-着色 (3-coloring) 问题可由一元二阶逻辑 (monadic second-order logic, MSO) 描述等. 近几十年来, 随着逻辑方法的引入, 算法领域涌现了大量的算法元定理 (algorithmic meta-theorem). 算法元定理是一种通用结论, 可以作为模板生成大量的高效算法, 其研究的问题主要是模型检测 (model-checking) 问题. 由于模型检测问题的泛用性, 通常很难得到高效算法, 因此实际的研究中需要限定问题的范围. 现有的算法元定理所讨论的都是一些简单的对象, 其中图 (graph) 的相关研究最多. 但即使对于图这类相对简单的结构, 诸多常见逻辑的模型检测问题仍是困难的, 甚至大量的具体问题都是困难的. 于是研究者们尝试进一步限定图的具体结构, 这类研究表明模型检测问题在许多特定图类上变得相对易解, 丰富的算法元定理应运而生.

给定简单无向图 (simple undirected graph) $G=(V,E)$, V, E 分别为顶点 (vertex) 和边 (edge) 的集合, G 的平均度 (average degree) 定义为 $2|E|/|V|$, 即 2 倍边的数量与顶点数量的比值. 稀疏性 (sparsity) 可以经验性地定义如下: 对于一个图类 (graph class) C, 若任意图 $G\in C$ 的平均度均不超过某一固定的常数, 则称 C 为稀疏的 (sparse), 否则为稠密的 (dense). 此定义较容易理解, 但并不本质: 某些满足该定义的图类并不一定具有良好的稀疏性质, 例如一些局部稠密的图类, 同时也有一些性质良好但相对更加稠密的图类, 例如一些局部稀疏的图类. 因此结构图论 (structural graph theory) 中并未普遍采用这一定义, 而是对于稀疏性有着更深刻、更本质的刻画, 后文将会详述. 算法元定理的研究肇始于 20 世纪 90 年代初对某些特殊的稀疏图类的讨论, Courcelle 证明了第 1 个真正意义上的算法元定理: 树宽有界 (bounded tree-width) 图类上的 MSO 模型检测问题可以在固定参数线性时间 (fixed-parameter linear time, FPL) 内解决[1]. 历经 20 余年的探索, 稀疏图类上的相关工作已取得可观的成就, 尤其是对于 FO 模型检测问题的研究相当深入, 一系列的算法元定理说明其在下列图类上是固定参数易解 (fixed-parameter tractable, FPT) 的: 度有界 (bounded degree) 图类[2]、真子式理想 (proper minor ideal)[3]、平面图类[4]、局部树宽有界 (bounded local tree-width) 图类[4]、有界扩张 (bounded expansion) 图类[5]、无处稠密 (nowhere dense) 图类[6]等. 其中, 无处稠密图类的结果几乎是紧的: 对于某处稠密 (somewhere dense) 的子图理想 (subgraph ideal), 在一些经典的复杂性假设下 FO 模型检测问题不是 FPT 的[6]. 因此, 进一步的研究应考虑非子图理想的图类 (例如诸多稠密图类), 或考虑图以外的对象. 后者的研究面临更多困难, 相关结论尚少, 已知结论如宽度有界的偏序集 (poset) 类上的 FO 模型检测问题是 FPT 的[7]等. 故近 10 年来, 算法元定理研究的重心逐渐转移到了稠密图类上.

相比于稀疏图类, 稠密图类通常具有更复杂的结构, 我们对其理解尚不充分, 现有的工作大多是对一些“良构” (well-structured) 稠密图类的研究, 它们具有相对清晰的结构性质. 例如一些研究表明对于区间图 (interval graph) 类[8]、地图图 (map graph) 类[9]、一些几何图类 (geometric graphs)[10]等, FO 模型检测问题是 FPT 的. 一般来说, 在稠密图类上相对简单的 FO 模型检测问题都已较为困难, 但在一些性质良好的稠密图类上, 甚至 MSO 模型检测问题仍是 FPT 的, 例如团宽有界 (bounded clique-width) 图类[11]等. 各种逻辑都有其对应的易解范围, 即使得相应的模型检测问题易解的类的总体, 我们总是关心其中最复杂的类复杂到何种程度, 希望对其进行一些结构上的刻画, 探索各种逻辑的易解范围是算法元定理研究的主要目标之一. 总体来说, 逻辑的描述能力越强, 则易解范围越小, 例如树宽有界和团宽有界已经较好地刻画了 MSO (及其拓展) 的易解范围[1,11], 在更大的图类上 MSO 模型检测问题不太可能是 FPT 的, 如平面图类和度有界图类等[12], 著名的 Seese 猜想[13]则断言 MSO 的易解范围大致由团宽有界图类界定. MSO 比 FO 的表达能力更强, FO 模型检测问题的易解性应该在更多的类上得到保持, 对于 FO 易解范围的探索是当前研究的一个重要的课题.

由稀疏图类通过转导 (transduction) 得到的结构性 (structurally) 图类是一类重要的良构稠密图类, 目前有关稠密图类的研究大多集中在对结构性图类的讨论. 若某图类具有性质 P, 则称该图类转导所得的图类具有结构性性质 (structural property) P, 例如树宽有界图类通过转导即得到结构性树宽有界 (structurally bounded tree-width) 图

类. 直观来说, 转导得到的结构性图类中保留了原稀疏图类的信息, 如果能从结构性图类中恢复出原稀疏图类的信息, 就能将其上的模型检测问题归约到原稀疏图类上, 从而应用稀疏图类上的结论——我们对稀疏图类的理解已经较为全面. 但恢复原稀疏图类的信息通常并不容易, 需要对结构性图类的具体结构进行分析. 关于结构性图类的一些主要结论如下: Oum[14]依照归约的思路提出了团宽有界图类上 MSO 模型检测问题的另一种高效算法, 尽管并没有直接引入转导的概念; Gajarský 等人[15]证明了结构性度有界 (structurally bounded degree) 图类上的 FO 模型检测问题是 FPT 的, 后续研究对该结论进行了简单推广[16]; Bonnet 等人[17]证明了结构性局部团宽有界 (structurally bounded clique-width) 图类上的 FO 模型检测问题是 FPT 的; 最终, Dreier 等人[18]证明了结构性无处稠密 (structurally nowhere dense) 图类上的 FO 模型检测问题也是 FPT 的, 这标志着我们对结构性图类的认识已经较为深刻.

稠密图类的相关研究十分依赖于一些关键的结构性质. 例如有研究引进了 3 种不同的分解 (decomposition) 用以刻画结构性有界扩张图类[19], 后续研究则在此基础上给出了一些分解定理来刻画结构性无处稠密图类[20]. 还有研究讨论了图类的另外一些结构性质[21], 包括弱稀疏性 (weakly sparse)、稳定性 (stability)、一元依赖性 (monadic dependence) 等, 它们将为算法元定理的研究提供新的工具和线索.

本文将详细介绍算法元定理的研究成果, 对一些重要的方法进行统合与抽象, 旨在清晰地展现算法元定理领域的概况, 为国内的相关研究提供一些信息与线索. 本文第 1 节介绍相关的背景知识. 第 2 节将回顾算法元定理的早期研究历程, 主要包括一些基于自动机方法的研究. 第 3 节将沿着 FO 模型检测的研究路径, 介绍算法元定理研究的进一步发展, 重点介绍基于 FO 局部性的方法. 第 4 节将介绍有关 FO 模型检测问题易解范围的研究, 这些研究完善了我们对稀疏图类的认识, 并将研究拓展到了稠密图类, 这也是近年来的主要研究方向. 这一阶段也产生了一些新的技术, 本文主要介绍基于量词消去的方法和基于转导等概念的归约技术. 第 5 节为全文总结.

图 1 展示了算法元定理领域中的一些重要结果与核心技术, 有助于读者了解该领域研究的大致图景: 图中的节点内是一些结构图论中常见的图类，其中扁圆形节点为局部化图类, 细边框节点为稀疏图类, 粗边框节点为稠密图类. 箭头连线表示图类的包含关系 (例如局部有界扩张图类一定是无处稠密图类), 其中橙色连线表示终点是起点的局部化图类, 红色连线表示终点是起点的结构性图类. 节点的颜色表示该图类上的研究使用的技术, 其中蓝色表示基于自动机的方法, 橙色表示基于 FO 局部性的方法, 绿色表示基于量词消去的方法, 红色表示基于转导的归约技术等稠密图相关技术. 绿色和紫色虚线框分别表示一元二阶逻辑和一阶逻辑模型检测问题的易解范围. 特别地, 目前对于一元依赖图类 (灰色节点) 的了解较少, 但普遍猜想其在一阶逻辑的易解范围内, 确切结论尚待未来的研究.

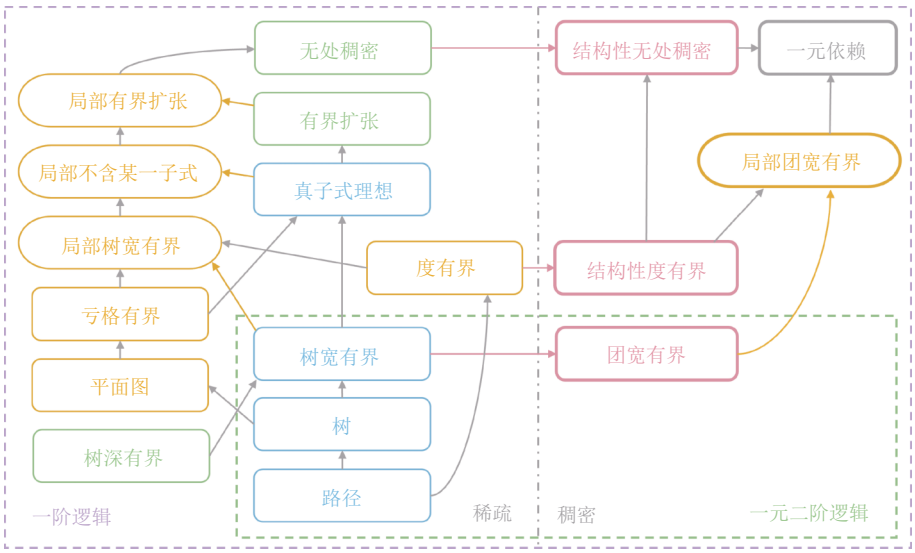

图 1　算法元定理领域的一些重要结果与核心技术

1 基础知识

在本文中, $\mathbb{N}$ 表示非负整数 (即自然数) 集, $\mathbb{N}^+$ 表示正整数集, $\mathbb{R}$ 表示实数集, $[n]$ 表示集合 $\{1,2,\ldots,n\}$.

1.1 逻辑与有限模型论

本节简要介绍数理逻辑[22]、模型论[23−25]和有限模型论 (finite model theory)[26,27]的一些基础概念. 逻辑 (logic) 可以理解为由一些特定规则构造出的所有公式 (formula) 的集合. 算法元定理主要研究一阶逻辑 FO 和一元二阶逻辑 MSO. 公式中的变元 (variable) 包括一阶变元和一元二阶变元 (即集合变元), 分别用小写字母 x,y,z 和大写字母 X,Y,Z 表示. 符号集 (symbol set) τ 是有限多个关系 (relation) 符号和常量 (constant) 符号 (简称"关系""常量") 的集合, 其中每个关系 R 都有其相应的元数 (arity) $ar(R)$. 全体一阶逻辑 τ 公式的类 $\mathrm{FO}[\tau]$ 按如下方式归纳地定义: 一阶变元或常量称为项 (term), 原子 (atomic) 公式包括 $t_1=t_2$ 和 $Rt_1t_2\ldots t_{ar(R)}$, 其中 R 是 τ 中任意关系, t_i 是任意项, 所有的原子公式属于 $\mathrm{FO}[\tau]$; 逻辑运算符 (logical connective) 包括 $\wedge$ (与)、$\vee$ (或)、$\neg$ (非), 若 $\varphi,\psi\in\mathrm{FO}[\tau]$, 则 $(\varphi\wedge\psi),(\varphi\vee\psi),\neg\varphi\in\mathrm{FO}[\tau]$; 量词 (quantifier) 包括 $\exists$ (存在)、$\forall$ (任意), 只作用于一阶变元, 若 $\varphi\in\mathrm{FO}[\tau]$, 则 $\exists x\varphi,\forall x\varphi\in\mathrm{FO}[\tau]$. 一元二阶逻辑 τ 公式的类 $\mathrm{MSO}[\tau]$ 的定义基本相同, 但允许使用一元二阶变元 X, 即在 $\mathrm{FO}[\tau]$ 定义的基础上增加原子公式 $Xt\in\mathrm{MSO}[\tau]$, 并且允许量词作用于 X, 即若 $\varphi\in\mathrm{MSO}[\tau]$, 则 $\exists X\varphi,\forall X\varphi\in\mathrm{MSO}[\tau]$. 显然, $\mathrm{FO}[\tau]\subseteq\mathrm{MSO}[\tau]$. 公式中不受量词约束的变元称为自由变元 (free variable), 否则称为约束变元 (bounded variable), 不含自由变元的公式称为语句 (sentence).

定义 1. MSO 公式的量词阶数 (quantifier rank) $qr:\mathrm{MSO}\to\mathbb{N}$ 归纳定义如下:

(1) 原子公式量词阶数为 0;

(2) $qr(\neg\varphi):=qr(\varphi)$;

(3) $qr(\varphi\wedge\psi)=qr(\varphi\vee\psi):=\max\{qr(\varphi),qr(\psi)\}$;

(4) $qr(\exists x\varphi)=qr(\forall x\varphi)=qr(\exists X\varphi)=qr(\forall X\varphi):=qr(\varphi)+1$.

类似地可以定义 FO 公式的量词阶数. 直观上, 量词阶数即公式中量词嵌套的层数, 而不是量词的数量. τ 结构 (structure) 是一个元组 $\mathfrak{A}=\left(V(\mathfrak{A}),(R^{\mathfrak{A}})_{R\in\tau},(c^{\mathfrak{A}})_{c\in\tau}\right)$, 其中论域 (universe) $V(\mathfrak{A})$ 为非空有限集, $R^{\mathfrak{A}}\subseteq V(\mathfrak{A})^{ar(R)}$ 是对关系 R 的解释 (interpretation), $c^{\mathfrak{A}}\in V(\mathfrak{A})$ 是对常量 c 的解释. $\mathfrak{A}$ 的编码长度 $\|\mathfrak{A}\|$ 称为规模 (size). 由结构构成的类称为结构类. 给定 τ 结构 $\mathfrak{A},\mathfrak{B}$, 若从 $V(\mathfrak{A})$ 的子集 S 到 $V(\mathfrak{B})$ 的单射 ϕ 满足: (1) 对所有关系 $R\in\tau$ 和任意 $\bar{a}\in S^{ar(R)},\bar{a}\in R^{\mathfrak{A}}$ 当且仅当 $\phi(\bar{a})\in R^{\mathfrak{B}}$; (2) 对所有常量 $c\in\tau,c^{\mathfrak{A}}\in S$ 且 $\phi(c^{\mathfrak{A}})=c^{\mathfrak{B}}$, 则称 ϕ 为 $\mathfrak{A}$ 到 $\mathfrak{B}$ 的一个部分同构 (partial isomorphism). 进一步, 若 ϕ 为 $V(\mathfrak{A})$ 到 $V(\mathfrak{B})$ 的双射, 则称为 $\mathfrak{A}$ 到 $\mathfrak{B}$ 的一个同构 (isomorphism). 若存在 $\mathfrak{A}$ 到 $\mathfrak{B}$ 的同构, 则称 $\mathfrak{A}$ 与 $\mathfrak{B}$ 同构 (isomorphic), 记为 $\mathfrak{A}\cong\mathfrak{B}$. 我们用 $\varphi(\bar{x})$ 表示公式 φ 的自由变元包含于 $\bar{x}$. 记号 $\mathfrak{A}\vDash\varphi(\bar{a})$ 表示当 $\bar{x}$ 被赋值为 $\bar{a}\in V(\mathfrak{A})^{|\bar{x}|}$ 时, 结构 $\mathfrak{A}$ 满足 φ. 若 φ 为语句则直接记为 $\mathfrak{A}\vDash\varphi$. 公式的语义遵循常规定义[22]. 给定不含常量的 τ 结构 $\mathfrak{A},\mathfrak{B}$, 定义二者的并 $\mathfrak{A}\cup\mathfrak{B}$ 为一个新的 τ 结构, 其论域为 $V(\mathfrak{A})\cup V(\mathfrak{B})$, 且对任意关系 $R\in\tau$, $R^{\mathfrak{A}\cup\mathfrak{B}}:=R^{\mathfrak{A}}\cup R^{\mathfrak{B}}$.

在上述逻辑中, 我们可以定义范式 (normal form). 范式是一类具有特殊格式的公式, 任意公式都等价于某个范式, 其定义并不唯一, 本文中采用一种满足下述性质的范式: (1) 给定任意一个公式 φ, 可以高效地计算其对应的范式 φ', 并且该范式满足 $qr(\varphi')=qr(\varphi)$; (2) 对于固定的符号集 τ、量词阶数 q 和自由变元的最大下标 m, 范式的数量是有限的, 而且范式的数量是有限的, 而且存在一个算法, 输入 τ,q,m, 枚举所有符号集为 τ、量词阶数不超过 q 且自由变元的下标不超过 m 的范式. 我们总可以假设对公式的分析都是将其转化成范式后再进行的, 这样做有很多好处. 例如, 我们并不要求公式 φ 的自由变元按某种顺序显式地列出, 即不一定会写成 $\varphi(\bar{x})$, 在这种情况下记号 $\mathfrak{A}\vDash\varphi(\bar{a})$ 中对 φ 中自由变元的赋值没有精确定义, 但若 φ 是范式, 就可以默认将 φ 中的自由变元 x_i 赋值为 $\bar{a}$ 中第 i 个元素, φ 中某自由变元下标大于 $|\bar{a}|$ 时定义 $\mathfrak{A}\nvDash\varphi(\bar{a})$. 由此可以定义 q 型 (q-type) 的概念.

定义 2. 给定逻辑 L, τ 结构 $\mathfrak{A}$, 自然数 m,q 以及 $\bar{a}\in V(\mathfrak{A})^m$, $\mathfrak{A}$ 中关于 $\bar{a}$ 的 q 型 $tp_q^L(\mathfrak{A},\bar{a})$ 定义为集合 $tp_q^L(\mathfrak{A},\bar{a}):=\left\{\varphi\in L\mid\varphi\text{是范式},\mathfrak{A}\vDash\varphi(\bar{a}),qr(\varphi)\leqslant q\right\}$. 当 $m=0$ 时, 简记为 $tp_q^L(\mathfrak{A})$. 另记全体 q 型 $Type_{q,m}^L[\tau]$ 为集合 $\{tp_q^L(\mathfrak{A},\bar{a})\mid\mathfrak{A}\text{为}\tau\text{结构},\bar{a}\in V(\mathfrak{A})^m\}$.

在有限结构上 q 型具有许多良好的性质. 首先, 由于对于固定的符号集 τ、量词阶数 q 和自由变元的最大下标 m, 范式的数量是有限的, 故 $tp_q^L(\mathfrak{A})$ 和 $Type_{q,m}^L[\tau]$ 为有限集. 其次, 对于任意公式 φ, 可以通过穷举判定是否有 $\mathfrak{A}\vDash\varphi(\bar{a})$, 故可以通过检查所有符号集为 τ、量词阶数至多为 q 且自由变元下标最大为 $|\bar{a}|$ 的范式计算出 $tp_q^L(\mathfrak{A},\bar{a})$. 但需要注意的是, 通常来说我们并不能计算出 $Type_{q,m}^L[\tau]$, 因为 τ 结构的数量是无穷的, 不能通过穷举得出所有的 q 型, 事实上, FO 公式的可满足性在一般意义下是不可判定的, 因此判定某个范式是否可满足也是困难的. 关于 q 型还有一个简单的观察: 若 $I\subseteq\mathbb{N}^+$ 是一个索引集, $\bar{a}_I$ 是 $\bar{a}$ 中索引属于 I 的元素构成的序列, 则根据 $tp_q^L(\mathfrak{A},\bar{a})$ 可以很容易地计算出 $tp_q^L(\mathfrak{A},\bar{a}_I)$. 为了形式化地描述这一计算过程, 我们引入定义 3.

定义 3. 给定索引集 $I\subseteq\mathbb{N}^+$ 和 q 型 t, 定义 $[t]_I$ 为在 t 的基础上经过如下操作得到的公式集: (1) 对于任意公式 $\varphi\in t$, 若 φ 中存在下标不属于 I 的自由变元, 则删除 φ; (2) 将第 (1) 步剩余的所有公式中的自由变元 x_i 替换为 $x_{i-|S|}$, 其中 $S:=\{j\in\mathbb{N}^+\mid j<i, j\notin I\}$; (3) 第 (2) 步得到的所有公式转化为等价的范式.

若 $I=\{i\}$, 将 $[t]_I$ 简记为 $[t]_i$. 显然, 根据 t 计算 $[t]_I$ 的过程是高效的. 另外, $[t]_I$ 一定也是 q 型, 因为有 $\left[tp_q^L(\mathfrak{A},\bar{a})\right]_I=tp_q^L(\mathfrak{A},\bar{a}_I)$, 这也解释了如何根据 $tp_q^L(\mathfrak{A},\bar{a})$ 计算 $tp_q^L(\mathfrak{A},\bar{a}_I)$. 若两个 τ 结构 $\mathfrak{A},\mathfrak{B}$ 满足相同的量词阶数至多为 q 的 L 语句, 即 $tp_q^L(\mathfrak{A})=tp_q^L(\mathfrak{B})$, 则记 $\mathfrak{A}\equiv_q^L\mathfrak{B}$. 容易看出, 若 $\mathfrak{A}\cong\mathfrak{B}$, 则对任意 q 都有 $\mathfrak{A}\equiv_q^L\mathfrak{B}$. 显然 $\equiv_q^L$ 是一个等价关系, 但通常来说证明两个结构满足这一等价关系并不容易, 典型的证明中需要使用进退构造法 (back-and-forth method) 或者 Ehrenfeucht-Fraïssé博弈 (game) (简称 EF 博弈)[26].

1.2 图与有根树

在图论中, 通常使用记号 $G=(V,E)$ 表示一个图, 其中 V 为顶点集, E 为边集, uv 指代顶点 u,v 之间的边. 若无特殊说明, 本文中的图默认为有限简单无向图, 即顶点数量有限、没有重边或自环而且边没有方向的图. 为了应用逻辑方法, 我们定义符号集 $\tau_E=\{E\}$, 其中 E 为二元关系, 并将图 G 视为一个 τ_E 结构 $\mathfrak{G}=(V(\mathfrak{G}),E^{\mathfrak{G}})$, 论域 $V(\mathfrak{G})$ 即顶点集 V, $E^{\mathfrak{G}}$ 即边集 E. 若顶点 u,v 之间有边, 则称 u 与 v 相邻 (adjacent), 与 u 相邻的顶点称为 u 的邻居 (neighbor), u 的邻居的数量称为 u 的度 (degree), 记为 $deg(u)$, 图 $\mathfrak{G}$ 的度 $deg(\mathfrak{G})$ 定义为 $\mathfrak{G}$ 中顶点的最大度. 我们定义图上的一些操作如下.

(1) 顶点删除: 对于顶点 u, 从 V 中移除 u, 并从 E 中移除所有包含 u 的边;

(2) 边删除: 对于边 uv, 从 E 中移除 $(u,v),(v,u)$;

(3) 边收缩 (edge contraction): 对于边 uv, 从 V 中移除 v, 从 E 中移除 $(u,v),(v,u)$, 并将 E 中所有 v 替换为 u.

在 $\mathfrak{G}$ 的基础上进行一系列顶点删除、边删除和边收缩操作后得到的图称为 $\mathfrak{G}$ 的子式 (minor); 若仅允许顶点删除和边删除, 则称为子图 (subgraph); 若仅允许顶点删除, 则称为导出子图 (induced subgraph). 导出子图还能通过剩余的顶点来定义: 给定集合 $S\subseteq V$, 以 S 为顶点集、$S^2\cap E$ 为边集的图 $G[S]$ 即为 S 对应的导出子图. 对任意不含常量的 τ 结构 $\mathfrak{A}$, 我们也可以类似地定义 $S\subseteq V(\mathfrak{A})$ 对应的导出子结构 (induced substructure) $\mathfrak{A}[S]$, 即论域为 S、任意关系 $R\in\tau$ 的解释为 $R^{\mathfrak{A}}\cap S^{ar(R)}$ 的 τ 结构. 由图构成的类称为图类 (graph class), 所有的图构成的类记为 $Graph$. 若图类 C 中任意图 $\mathfrak{G}$ 的所有子式仍属于 C, 则称 C 对子式封闭, 并称 C 为一个子式理想 (minor ideal). 若 $C\subsetneq Graph$, 则称 C 为真子式理想. 图类 D 的子式闭包定义为 $\{\mathfrak{H}\mid\mathfrak{H}\text{为}\mathfrak{G}\text{的子式},\mathfrak{G}\in D\}$, 显然子式闭包一定是子式理想. 对子图、导出子图等也有类似的定义.

顶点 u,v 之间的一条长度为 ℓ 的路径 (path) 是一个顶点的序列 $u,w_1,w_2,\ldots,w_{\ell-1},v$, 其中顶点不重复且任意两个相继的顶点之间有边. u,v 之间最短路径的长度称为 u,v 的距离, 记作 $dist^{\mathfrak{G}}(u,v)$, 定义 $dist^{\mathfrak{G}}(u,u)=0$. 定义顶点集合 U,V 之间的距离 $dist^{\mathfrak{G}}(U,V)=\min_{u\in U,v\in V}dist^{\mathfrak{G}}(u,v)$. 若 u,v 之间存在路径或者 $u=v$, 则称 u,v 之间是可达 (reachable) 的, 若 $\mathfrak{G}$ 中任意两个顶点之间都可达, 则称 $\mathfrak{G}$ 是连通 (connected) 的. 连通图中最长路径的长度称为直径 (diameter). $\mathfrak{G}$ 的极大连通子图称为分量 (component), 一个图可能包含若干分量, 分量之间互不相交. 图的连通性不能用 FO 语句定义[28], 但很容易用 MSO 公式定义, 可见 MSO 的表达能力 (expressive power) 要强于 FO.

"路径"也指代一类图: 长度为 ℓ 的路径 $\mathfrak{P}_\ell$ 是顶点集为 $[\ell]$、边集为 $\{(i,i+1),(i+1,i)\mid i\in[\ell-1]\}$ 的图, 所有路径构成的类记为 $Path$. 长度为 ℓ 的环 (cycle) $\mathfrak{C}_\ell$ 是顶点集为 $[\ell]$、边集为 $\{(i,i+1),(i+1,i)\mid i\in[\ell-1]\}\cup\{(1,\ell),(\ell,1)\}$ 的

图, 所有环构成的类记为 $Cycle$. 若一个图不含长度超过 2 的环作为子图, 则称其为无环图. 大小为 ℓ 的团 (clique) (即完全图) $\mathfrak{K}_\ell$ 是指顶点集为 $[\ell]$、边集为 $\{(i,j)\in[\ell]^2 \mid i\neq j\}$ 的图, 所有团构成的类记为 $Clique$. 大小为 $\ell_1\times\ell_2$ 的二分团 $\mathfrak{K}_{\ell_1,\ell_2}$ 是指顶点集为 $[\ell_1+\ell_2]$、边集为 $\{(i,j+\ell_1),(j+\ell_1,i) \mid i\in[\ell_1], j\in[\ell_2]\}$ 的图, 所有二分团构成的类记为 $Biclique$.

连通的无环图称为树 (tree), 树中的顶点又称为节点 (node), 任意两个节点间都存在唯一的路径, 记所有树构成的类为 $Tree$. 任取树中的一个节点 r 作为根 (root), 就得到有根树 (rooted tree). 节点 u 到 r 的路径中的每个节点都称为 u 的祖先 (ancestor), u 则称为这些节点的后代 (descendant), 与 u 相邻的祖先 v 称为 u 的父节点 (parent), 反过来 u 称为 v 的子节点 (child). 除了 r, 所有的节点都有唯一的父节点; 没有子节点的节点称为叶节点 (leaf). u 的所有后代构成的集合对应的导出子图也是一棵树, 若其以 u 为根节点, 则称为 u 的导出子树. u 的深度 (depth) 定义为根节点到 u 的距离, 有根树的高度 (height) 定义为节点的最大深度. 定义符号集 $\tau_P=\{P\}$, 其中 P 为二元关系. 我们将有根树表示为 τ_P 结构 $\mathfrak{T}$, 其论域 $V(\mathfrak{T})$ 为节点的集合, $(u,v)\in P^{\mathfrak{T}}$ 表示 u 是 v 的父节点. 记所有有根树构成的类为 $RTree$. 若有根树中所有节点的子节点均不超过两个, 则之称为二叉树 (binary tree), 记所有二叉树构成的类为 $BTree$.

图的许多性质都可以用逻辑公式来描述, 我们用一些简单的例子来说明. 给定图 $\mathfrak{G}$, 若顶点集合 S 满足对于任意顶点 $u\in V(\mathfrak{G})$ 均有 $u\in S$ 或 u 的某个邻居 $v\in S$, 则称 S 为 $\mathfrak{G}$ 的一个支配集. 图 $\mathfrak{G}$ 有大小不超过 k 的支配集可由下述 FO 语句描述:

$$\psi_{ds}^{k}:=\exists x_1\exists x_2\ldots\exists x_k\forall y\bigvee_{i\in[k]}(y=x_i\vee Ex_iy),$$

给定图 $\mathfrak{G}$ 和 $k\in\mathbb{N}$, 判定 $\mathfrak{G}$ 是否有大小不超过 k 的支配集, 是一个经典的 NP 完全 (NP-complete) 问题.

若函数 $col: V(\mathfrak{G})\to[3]$ 满足对任意 $(u,v)\in E^{\mathfrak{G}}$ 均有 $col(u)\neq col(v)$, 则称为图 $\mathfrak{G}$ 的一个 3-着色. 图 $\mathfrak{G}$ 存在 3-着色可由下述 MSO 语句描述:

$$\psi_{3col}:=\exists X_1\exists X_2\exists X_3\forall x\bigvee_{i\in[3]}X_ix\wedge\forall x\forall y\left(Exy\to\bigwedge_{i\in[3]}\neg(X_ix\wedge X_iy)\right),$$

给定图 $\mathfrak{G}$, 判定 $\mathfrak{G}$ 是否存在 3-着色, 也是一个经典的 NP 完全问题.

一般来说, 着色 (coloring) 是指为每个顶点分配一种或多种颜色 (color), 颜色有时也称为标号 (label), 本质上每种颜色就是一个一元关系. 我们将 $\tau_E\cup\tau$ 结构 $\mathfrak{A}$ 称为标号图 (labeled graph), 其中 τ 为任意仅含一元关系的符号集, 每个一元关系称为一个标号. 特别地, τ_E 结构也是标号图. 定义 $\mathfrak{A}$ 的基础图 (underlying graph) 为去除 $\mathfrak{A}$ 中对所有一元关系的解释后剩下的 τ_E 结构. 给定一个不在 $\tau_E\cup\tau$ 中的一元关系 U 和集合 $S\subseteq V(\mathfrak{A})$, 记 $\mathfrak{A}\langle U\mapsto S\rangle$ 为在 $\mathfrak{A}$ 的基础上将 U 解释为 S 所得的标号图.

最后我们定义任意结构的 Gaifman 图.

定义 4. τ 结构 $\mathfrak{A}$ 的 Gaifman 图 $G(\mathfrak{A})$ 是以 $V(\mathfrak{A})$ 为顶点集、以 $\{(b,c) \mid R\in\tau,\bar{a}\in R^{\mathfrak{A}},b,c\in\bar{a},b\neq c\}$ 为边集的图. 对任意 $a,b\in V(\mathfrak{A})$, a 与 b 的距离 $dist^{\mathfrak{A}}(a,b):=dist^{G(\mathfrak{A})}(a,b)$.

1.3 模型检测与算法元定理

我们简要介绍计算复杂性 (computational complexity)[29,30]和参数复杂性 (parameterized complexity)[31,32]的相关理论. 给定 (非空有限) 字母表 (alphabet) Σ, 记 Σ^+ 为由 Σ 中字母构成的所有有限长度的非空字符串 (string) 的集合. 集合 $Q\subseteq\Sigma^+$ 称为 Σ 上的一个语言 (language), 由 Q 可以定义一个判定问题 (decision problem): 输入字符串 $x\in\Sigma^+$, 判定是否有 $x\in Q$. 给定多项式时间可计算 (polynomial time computable) 函数 $\kappa:\Sigma^+\to\mathbb{N}$, 由 (Q,κ) 可以定义一个参数化问题 (parameterized problem): 输入字符串 $x\in\Sigma^+$ 并计算相应的参数 (parameter) $\kappa(x)$, 判定是否有 $x\in Q$. 参数化问题和经典的判定问题描述上并无本质区别, 参数的作用体现在对计算复杂性的刻画中. 对应于经典复杂性中的多项式时间可计算, 在参数复杂性中有固定参数易解 (即固定参数多项式时间可计算) 的概念.

定义 5. 若存在可计算函数 $f:\mathbb{N}\to\mathbb{N}$ 和常数 $c\in\mathbb{N}$, 使得对于任意输入 $x\in\Sigma^+$, 某算法的运行时间至多是 $f(\kappa(x))\cdot|x|^c$, 则称该算法为 (c 次方时间的) 固定参数多项式时间算法 (简称 FPT 算法). 若存在 FPT 算法判定任意输入 $x\in\Sigma^+$ 是否属于 Q, 则称参数化问题 (Q,κ) 为固定参数易解 (简称 FPT) 的. 若 $c=1$, 相应地得到固定参数线性

时间算法 (简称 FPL 算法) 和固定参数线性时间可计算 (简称 FPL) 的概念. FPT 和 FPL 分别表示所有 FPT 和 FPL 的参数化问题构成的类.

直觉上, FPT 算法可以先对参数进行足够多的预计算, 然后再对问题本身进行求解. 通常 FPT 算法的预计算需要消耗大量的时间, 即 f 的增长速度会远超多项式函数. 但预计算的策略在参数远小于输入长度时十分有效: 当 $\kappa(x)$ 远小于 $|x|$ 时, $f(\kappa(x))\cdot|x|^c$ 的增长速度要明显慢于形如 $|x|^{\kappa(x)}$ 之类的函数, 后者是简单的穷举算法常见的运行时间. 因此, 实际应用中一般选择远小于输入长度的量作为参数, 从而发挥 FPT 算法的优势. 另外, 本文也使用渐进符号来刻画复杂性[31].

模型检测问题是算法元定理研究的主要问题. 固定逻辑 L 和结构类 C, C 上的 L 模型检测问题 $MC(L,C)$ 是指如下判定问题: 输入结构 $\mathfrak{A}\in C$, 语句 $\varphi\in L$, 判定是否有 $\mathfrak{A}\vDash\varphi$. 模型检测问题可以视为一类问题的通用模板, 许多具体的问题都可以描述为模型检测问题. 算法元定理的典型形式如下: 对于逻辑 L 和结构类 C, 存在一个“高效”的算法解决模型检测问题 $MC(L,C)$. 另外也有一些算法元定理给出否定的结论, 即不存在“高效”算法, 这通常需要基于一些复杂性假设. 解决模型检测问题 $MC(L,C)$, 实际上就是解决了所有关于 C 且能用 L 中语句描述的问题, 因此算法元定理被称为“元定理”.

“高效”在不同语境下有不同含义. 在经典复杂性中, 高效算法一般是指多项式时间可计算的算法. 但 Vardi 证明了即使对于一个固定的结构 $\mathfrak{A}$, 只要其论域中至少包含 2 个元素, FO 模型检测问题 $MC(\mathrm{FO},\{\mathfrak{A}\})$ 就已经是多项式空间完备 (PSPACE-complete) 的[33]. 另一方面, 对于任意图类 C 和 FO 语句 φ, 模型检测问题 $MC(\{\varphi\},C)$ 平凡地是多项式时间可计算的, 因为可以在多项式时间内通过穷举来判定 FO 语句的满足性. 由于 3-着色问题可以由一个 MSO 语句描述, 因此对于固定的 MSO 语句 φ, $MC(\{\varphi\},C)$ 有可能是 NP 完备的. 由此可见, 在经典复杂性的框架下很难找到模型检测问题的高效算法. 在数据库 (database)、描述复杂性 (descriptive complexity) 等理论中还有更多相关研究: 有些工作讨论一些更简单的逻辑, 例如只包含存在量词和合取的一阶逻辑的模型检测问题是 NP 完备的[34], 变元数量有界的一阶逻辑的模型检测问题是多项式时间可解的[35]; 另有一些结论建立了逻辑的表达能力和复杂性类之间的联系, 例如著名的 Fagin 定理揭示了由不含二阶全称量词的二阶逻辑定义的问题类和 NP 是相同的[36], 详见描述复杂性的相关文献[37].

算法元定理的相关研究主要基于参数复杂性框架. 模型检测问题的参数可以选为结构 $\mathfrak{A}$ 的规模或语句 φ 的长度, Vardi 的结论[33]说明以 $\mathfrak{A}$ 为参数意义不大, 故选择以 φ 为参数, 即考虑如下参数化模型检测问题 $p\text{-}MC(L,C)$: 输入结构 $\mathfrak{A}\in C$, 语句 $\varphi\in L$, 以 $|\varphi|$ 为参数, 判定是否有 $\mathfrak{A}\vDash\varphi$. 在参数复杂性的语境下, 高效算法一般是指 FPT 算法. 具体到上述参数化模型检测问题, 即运行时间至多为 $f(|\varphi|)\cdot\|\mathfrak{A}\|^c$ 的算法. 通常来说, 算法元定理给出的 FPT 算法的预计算时间函数 f 甚至不是初等 (elementary) 函数[38], 直觉上是因为算法的通用性过强. 但只要参数远小于输入长度, 总运行时间增长就不会过快. 尤其是当参数固定时, FPT 算法就变成了多项式时间算法. 许多常见问题确实可以用一个简短的逻辑语句来描述, 例如 3-着色问题, 有关 MSO 模型检测的算法元定理即给出了 3-着色问题在特定图类上的多项式时间算法, 而简单的穷举算法只能得到指数时间算法. 然而, 即使是讨论参数复杂性, 也很难在全局意义下得到模型检测问题的高效算法: 全体结构类 Str 上的参数化 FO 模型检测问题 $p\text{-}MC(\mathrm{FO},Str)$ 是参数复杂性类 AW[$*$] 中的完备问题[39], 而 AW[$*$] 严格大于 FPT 是一个被广泛接受的复杂性假设. 即使是限制在 $Graph$ 上, $p\text{-}MC(\mathrm{FO},Graph)$ 仍是 AW[$*$] 完备的[39]. 由于 3-着色问题可由 MSO 语句描述, $p\text{-}MC(\mathrm{MSO},Graph)$ 甚至不在参数复杂性类 XP 中, XP 严格大于 AW[$*$] 是另一个被广泛接受的复杂性假设. 当我们进一步限定图类的结构时, 参数化模型检测问题就有可能存在 FPT 算法, 这是算法元定理研究的主要方向. 需要指出的是, 此时在经典复杂性理论框架中讨论固定语句的模型检测问题也能得到高效算法, 但参数复杂性框架下的研究更加细致、统一且易于推广. 另外简洁起见, 本文其他章节中“模型检测问题 $MC(L,C)$”默认指“参数化模型检测问题 $p\text{-}MC(L,C)$”. 需要注意的是本文中可能会讨论一些带参数的问题, 但参数并不是关于输入多项式可计算的, 本文仍称其为参数化问题, 并不影响相关的讨论, 进一步说明参见相关文献[31].

2 基于自动机方法的研究

早期的算法元定理源于自动机 (automaton) 理论, 一些原始的算法元定理结论早在 20 世纪 60 年代就已经问世, 尽管并未以典型形式呈现. 限于篇幅, 我们略去一些基础概念的介绍, 如正则语言 (regular language)、有限自动机 (finite automaton) 等[26]. 语言是字符串的集合, 字符串可以表示为如定义 6 所示的结构.

定义 6. 给定字母表 Σ, 令符号集 $\tau_\Sigma=\{<,(C_a)_{a\in\Sigma}\}$, 其中 $<$ 为二元关系, C_a 均为一元关系. 字符串 $s\in\Sigma^+$ 对应的结构 $\mathfrak{B}_s$ 为一个 τ 结构, 论域为 $[|s|]$, $<$ 为 $[|s|]$ 上的自然顺序, $C_a^{\mathfrak{B}_s}:=\{i\mid s\text{的第}i\text{个字符为}a\}$.

最早的算法元定理相关的结论是 Büchi-Elgot-Trakhtenbrot 定理 (简称 BET 定理)[40–42], 它揭示了由 MSO 语句定义的语言与正则语言的等价性, 其严格表述如定理 1.

定理 1[40–42]. $L\subseteq\Sigma^+$ 是正则语言当且仅当存在语句 $\varphi_L\in\mathrm{MSO}[\tau_\Sigma]$ 使得满足 φ_L 的结构恰为 $\{\mathfrak{B}_s\mid s\in L\}$.

著名的 Kleene 定理[43]告诉我们, 正则语言与有限自动机接受的语言是等价的, 由相应的正则表达式 (regular expression) 可以计算出一个接受该语言的确定性有限自动机 (deterministic finite automaton, DFA). 给定字符串 s, 要判断 s 是否属于某个正则语言 (即是否符合对应的正则表达式), 只需构造并运行对应的 DFA, 观察其是否接受 s 即可. 根据 Kleene 定理和 BET 定理, MSO 语句与 DFA 也存在对应关系. 因此, 给定字符串 s 和语句 $\varphi\in\mathrm{MSO}[\tau_\Sigma]$, 要判断 $\mathfrak{B}_s$ 是否满足 φ, 只需构造并运行 φ 对应的 DFA, 观察其是否接受 s 即可. DFA 的运行显然是高效的, 关键在于如何高效地构造对应的 DFA, 这需要考察 BET 定理的证明过程.

简洁起见, 我们通过一个特例来阐述此过程: 设字母表 Σ 中仅包含一个字符 a, 于是字符串即为 a 的有限序列, 用 a^m 表示 m 个 a 构成的字符串. 此时字符串结构 $\mathfrak{B}_s$ 中的一元关系 C_a 为冗余信息, 因此可以直接去掉并得到一个 $\tau_<$ 结构, 其中 $\tau_<:=\{<\}$, 简记 a^m 对应的 $\tau_<$ 结构为 $\mathfrak{O}_m$. 实际上 $\mathfrak{O}_m$ 描述了 $[m]$ 上的自然顺序, 一般称这类结构为序 (ordering), 所有序构成的类 $Ord:=\{\mathfrak{O}_m\mid m\in\mathbb{N}^+\}$, 我们的目标即设计模型检测问题 $MC(\mathrm{MSO},Ord)$ 的高效算法. 给定语句 $\varphi\in\mathrm{MSO}[\tau_<]$, 令 $q:=qr(\varphi)$, 我们构造 φ 对应的 DFA: 状态机的字母表为 $\Sigma=\{a\}$, 以 $Type_{q,0}^{\mathrm{MSO}}[\tau_<]$ 中的若干 q 型为状态, 接受状态为其中包含 φ 的 q 型, 额外引入一个起始状态 Q_0, 状态集和状态转移函数具体按如下过程确定: (1) Q_0 转移到 $tp_q^{\mathrm{MSO}}(\mathfrak{O}_1)$; (2) 对任意 $m\geqslant 1$, $tp_q^{\mathrm{MSO}}(\mathfrak{O}_m)$ 转移到 $tp_q^{\mathrm{MSO}}(\mathfrak{O}_{m+1})$, 直至状态转移图不再发生变化. 由于 $Type_{q,0}^{\mathrm{MSO}}[\tau_<]$ 中 q 型的数量是有限的, 故状态数是有限的. 我们还需要证明状态转移函数是良定义的, 因为不能确保每个状态只有一条出边.

引理 1. 对任意 $i,j\in\mathbb{N}^+$, 若 $\mathfrak{O}_i\equiv_q^{\mathrm{MSO}}\mathfrak{O}_j$, 则 $\mathfrak{O}_{i+1}\equiv_q^{\mathrm{MSO}}\mathfrak{O}_{j+1}$.

引理 1 可以通过简单的 EF 博弈证明. 最终我们设计算法 1.

算法 1. $MC(\mathrm{MSO},Ord)$ 的高效算法.

输入: 序 $\mathfrak{O}\in Ord$, 语句 $\varphi\in\mathrm{MSO}[\tau_<]$;

(1) 构造 φ 对应的 DFA;

(2) 以 $\mathfrak{O}$ 对应的字符串 a^m 为输入运行 DFA.

输出: DFA 是否接受 a^m.

算法 1 的正确性是显然的: 根据状态转移函数的构造过程, DFA 运行结束时会停在 $tp_q^{\mathrm{MSO}}(\mathfrak{O}_m)$ 状态. 而 $\varphi\in tp_q^{\mathrm{MSO}}(\mathfrak{O}_m)$ 当且仅当 $tp_q^{\mathrm{MSO}}(\mathfrak{O}_m)$ 是接受状态, 故 $\mathfrak{O}_m\vDash\varphi$ 当且仅当 DFA 接受 a^m. 下面说明算法 1 的高效性.

(1) DFA 的构造: 确定状态集和状态转移函数, 需要通过穷举计算每个 $tp_q^{\mathrm{MSO}}(\mathfrak{O}_m)$, 由于状态转移图中一旦成环便构造完毕, 只需穷举 m 不超过状态总数量的情况, 时间仅与 q 有关; 确定接受状态, 需要判断 φ 是否包含在每个状态的 q 型中, 将 φ 转为范式后一一比对即可, 时间仅与 q 有关, 显然 $q\leqslant|\varphi|$;

(2) 根据 $\mathfrak{O}$ 获取 a^m 的过程是 $\mathrm{O}(\|\mathfrak{O}\|)$ 时间的, 运行 DFA 的时间也是 $\mathrm{O}(\|\mathfrak{O}\|)$ 时间的.

故算法 1 是一个 FPL 算法, 于是我们得到了第 1 个算法元定理.

定理 2. $MC(\mathrm{MSO},Ord)\in\mathrm{FPL}$. 具体来说, 存在一个 $\mathrm{O}(f(k)+n)$ 时间的 FPL 算法.

简洁起见, 本文默认使用 n 表示输入长度, k 表示参数, f 表示预计算时间函数, 为可计算函数.

在一般情形下 (即 Σ 为任意字母表), 同样的方法可以得到类似的算法元定理, 读者可尝试推广. 值得注意的是, 该方法构造的 DFA 除接受状态的选择外与输入语句 φ 的具体内容无关, 仅与其量词阶数有关. 换句话说, 根据量词阶数相同的不同语句构造出的 DFA 除接受状态外完全相同. 也有另一种依赖于 φ 的具体内容的构造[31], 两种构造都需要 $|\varphi|$ 的塔函数 (tower function) 时间, 而在一些复杂性假设下塔函数时间大致是最优的[38].

基于自动机的方法可以推广到树上, 相应地我们有树自动机 (tree automaton)、树语言 (tree language) 等概念[31], 下面简要介绍.

定义 7. 确定性树自动机 (deterministic tree automaton, DTA) 为一个四元组 (S,Σ,F,δ), 其中 S 为状态集, Σ 为字母表, $F\subseteq S$ 为接受状态集, $\delta: S_\perp\times S_\perp\times\Sigma\to S$ 为状态转移函数, $S_\perp := S\cup\{\perp\}$, $\perp$ 为一个特殊状态. 若一棵二叉树的左右子节点是有序的, 并且每个节点绑定唯一一个 Σ 中的字符, 则称其为 Σ 树 (Σ-tree), Σ 树的集合称为树语言. DTA 的输入为任意一棵 Σ 树, 运行过程为自底而上地根据 δ 计算每个节点的状态: 对于每个节点 u, 若左子节点的状态是 Q_1, 右子节点的状态是 Q_2, u 绑定的字符为 a, 则 u 的状态是 $\delta(Q_1,Q_2,a)$; 若 u 没有左或右子节点, 则默认子节点状态为 $\perp$. DTA 接受一棵 Σ 树当且仅当 DTA 运行结束后根节点的状态是接受状态.

令 $\tau'_\Sigma=\{E_1,E_2,(C_a)_{a\in\Sigma}\}$, 我们可以将 Σ 树表示为一个 τ'_Σ 结构, 论域为所有的节点, E_1,E_2 分别为表示左、右子节点关系的二元关系, C_a 为表示每个节点绑定字符的一元关系. 关于树语言有如定理 3 所示, 类似 BET 定理的结论.

定理 3[44,45]. 树语言 L 被某个 DTA 接受当且仅当存在语句 $\varphi_L\in\mathrm{MSO}[\tau'_\Sigma]$ 定义 L.

定理 3 的证明过程表明与字符串的情况类似, DTA 和对应的 MSO 语句之间也可以相互计算, 因此可以通过构造 DTA 的思路来解决所有 Σ 树构成的类上的 MSO 模型检测问题, 下面仍通过一个稍简单的特例来说明. 假设字母表 Σ 中仅包含一个字符 a, 则可以省略 Σ 树对应的结构中所有的一元关系 C_a, 进一步忽略左右子节点的顺序, 即令 E_1 和 E_2 合为一个二元关系 P, 最终 Σ 树即退化为二叉树, 我们的目标即设计 $MC(\mathrm{MSO},BTree)$ 的算法. 基本思路如下: 给定 $\mathfrak{T}\in BTree$ 和语句 $\varphi\in\mathrm{MSO}[\tau_P]$, 构造 φ 对应的 DTA, 以 $\mathfrak{T}$ 为输入运行 DTA, $\mathfrak{T}\models\varphi$ 当且仅当 DTA 接受 $\mathfrak{T}$. DTA 的构造需要使用著名的 Feferman-Vaught 定理 (简称 FV 定理, 即定理 4).

定理 4[46]. 固定符号集 τ 和自然数 ℓ,i,j. 根据任意 q 型 $t_1\in Type^{\mathrm{MSO}}_{q,\ell+i}[\tau]$ 和 $t_2\in Type^{\mathrm{MSO}}_{q,\ell+j}[\tau]$, 可以计算出 q 型 $t_1\otimes_\ell t_2\in Type^{\mathrm{MSO}}_{q,\ell+i+j}[\tau]$ 满足如下条件: 对于任意 τ 结构 $\mathfrak{A},\mathfrak{B}$ 和任意 $\bar{u}\in(V(\mathfrak{A})\cap V(\mathfrak{B}))^\ell,\bar{v}\in V(\mathfrak{A})^i,\bar{w}\in V(\mathfrak{B})^j$, 若 $V(\mathfrak{A})\cap V(\mathfrak{B})\subseteq\bar{u}, t_1=tp^{\mathrm{MSO}}_q(\mathfrak{A},\bar{u}\bar{v}), t_2=tp^{\mathrm{MSO}}_q(\mathfrak{B},\bar{u}\bar{w})$, 则有 $t_1\otimes_\ell t_2=tp^{\mathrm{MSO}}_q(\mathfrak{A}\cup\mathfrak{B},\bar{u}\bar{v}\bar{w})$.

FV 定理的证明一般使用归纳法[47,48], 将其中的 MSO 替换为 FO 仍然成立. FV 定理是模型论中的经典结论, 有着广泛的应用[48]. 需要注意的是, 为了将其应用于 DTA 的构造, 本文中对 FV 定理的叙述强调了 $t_1\otimes_\ell t_2$ 的计算过程并不依赖于具体的结构 $\mathfrak{A}$. 令 $q:=qr(\varphi)$, φ 对应的 DTA 构造如下: 字母表为 $\Sigma=\{a\}$, 以 $Type^{\mathrm{MSO}}_{q,1}[\tau_P]$ 中的若干 q 型为状态, 接受状态为其中所有包含 φ 的 q 型. 回顾定义 3, 按如下过程确定状态集 S 和转移函数 δ.

(1) $\delta(\perp,\perp):=tp^{\mathrm{MSO}}_q(\mathfrak{T}_1,u)\in S$, 其中 $\mathfrak{T}_1$ 为仅含节点 u 的二叉树;

(2) 若 $t\in S$, 则 $\delta(\perp,t)=\delta(t,\perp):=[t\otimes_1 t^*]_2\in S$, 其中 $t^*:=tp^{\mathrm{MSO}}_q(\mathfrak{T}_2,uv)$, $\mathfrak{T}_2$ 为仅含节点 u,v 且以 v 为根的二叉树;

(3) 若 $t_1,t_2\in S$, 则 $\delta(t_1,t_2)=\delta(t_2,t_1):=[t_1\otimes_1 t^*]_2\otimes_1[t_2\otimes_1 t^*]_2\in S$;

(4) 重复上述操作, 直至 S 不再发生变化.

由于 FV 定理中 $\otimes_\ell$ 操作的结果是确定的, 故上述过程是良定义的. 最终我们设计算法 2.

算法 2. $MC(\mathrm{MSO},BTree)$ 的高效算法.

输入: 二叉树 $\mathfrak{T}\in BTree$, 语句 $\varphi\in\mathrm{MSO}[\tau_P]$;

(1) 构造 φ 对应的 DTA.

(2) 以 $\mathfrak{T}$ 为输入运行 DTA.

输出: DTA 是否接受 $\mathfrak{T}$.

由于状态转移函数是对称的, 因此我们在输入 $\mathfrak{T}$ 时可以任意地固定其左右子树的顺序. 下面说明算法 2 的正

确性. 我们使用归纳法证明对于 $\mathfrak{T}$ 中任意节点 u, 运行 DTA 时 u 对应的状态恰为 $t_u := tp_q^{\mathrm{MSO}}(\mathfrak{T}_u, u)$, 其中 $\mathfrak{T}_u$ 为以 u 为根的导出子树.

(1) 若 u 为叶节点, 其对应的状态为 $\delta(\bot,\bot) = tp_q^{\mathrm{MSO}}(\mathfrak{T}_1, u) = t_u$;

(2) 若 u 仅有左子节点 v (右子节点类似), 令 $\mathfrak{T}_{uv}$ 为仅包含 u,v 且以 u 为根的二叉树, 则 $tp_q^{\mathrm{MSO}}(\mathfrak{T}_{uv}, vu) = t^*$ 且 $\mathfrak{T}_u = \mathfrak{T}_v \cup \mathfrak{T}_{uv}$, 故由 FV 定理可得 $t_u = \left[tp_q^{\mathrm{MSO}}(\mathfrak{T}_u, vu)\right]_2 = \left[t_v \otimes_1 tp_q^{\mathrm{MSO}}(\mathfrak{T}_{uv}, vu)\right]_2 = [t_v \otimes_1 t^*]_2 = \delta(t_v, \bot)$;

(3) 若 u 有左右子节点 v,w, 则 $\mathfrak{T}_u = (\mathfrak{T}_v \cup \mathfrak{T}_{uv}) \cup (\mathfrak{T}_w \cup \mathfrak{T}_{uw})$, 故根据 FV 定理可得:

$$t_u = \left[t_v \otimes_1 tp_q^{\mathrm{MSO}}(\mathfrak{T}_{uv}, vu)\right]_2 \otimes_1 \left[t_w \otimes_1 tp_q^{\mathrm{MSO}}(\mathfrak{T}_{uw}, wu)\right]_2 = [t_v \otimes_1 t^*]_2 \otimes_1 [t_w \otimes_1 t^*]_2 = \delta(t_v, t_w).$$

由此可得, 根节点 r 的状态为 $t_r := tp_q^{\mathrm{MSO}}(\mathfrak{T}_r, r) = tp_q^{\mathrm{MSO}}(\mathfrak{T}, r)$, 故 $\mathfrak{T} \vDash \varphi$ 当且仅当 $\varphi \in tp_q^{\mathrm{MSO}}(\mathfrak{T}, r)$, 当且仅当 DTA 接受 $\mathfrak{T}$, 因此算法 2 是正确的. 算法 2 的高效性分析如下.

(1) DTA 的构造: 使用 FV 定理确定状态集和状态转移函数是一个归纳的过程, 状态数量是有限的, 故时间只与 q 有关; 确定接受状态, 将 φ 转为范式后与 q 型中的公式一一比对即可, 时间只与 q 有关;

(2) 以 $\mathfrak{T}$ 为输入运行 DTA 的时间是 $\mathrm{O}(\|\mathfrak{T}\|)$ 时间的.

可见算法 2 是一个 FPL 算法, 故得如定理 5 所示算法元定理.

定理 5. $MC(\mathrm{MSO}, BTree) \in \mathrm{FPL}$. 具体来说, 存在一个 $\mathrm{O}(f(k)+n)$ 时间的 FPL 算法.

上述结论容易推广到所有的 Σ 树构成的类上. 算法 2 中构造的 DTA 除接受状态的选择外与语句 φ 的具体内容无关, 仅与其量词阶数有关, 另外也有依赖于语句 φ 具体内容的构造[31].

自动机的方法还能进一步地推广到更复杂的图类上, 接下来我们将介绍著名的 Courcelle 定理[1], 一般认为它是历史上第一个真正意义上的算法元定理. 该定理的原始证明即基于树自动机, 但本文中采用更现代的叙述. 树宽 (tree-width) 是一个重要的图参数, 它描述了一个图在多大程度上接近于一棵树.

定义 8. 图 $\mathfrak{G}$ 的一个树分解 (tree decomposition) 是指满足如下条件的元组 $\left(\mathfrak{T}, \left(\bar{b}_w\right)_{w \in V(\mathfrak{T})}\right)$:

(1) $\mathfrak{T}$ 为一棵二叉树, $\bar{b}_w \in V(\mathfrak{G})^m$ 称为节点 w 对应的包 (bag), $m \in \mathbb{N}^+$ 为一个固定的常数, 定义 $m-1$ 为树分解的宽度 (width). 方便起见, 我们要求 $|V(\mathfrak{T})| \leqslant |V(\mathfrak{G})|$ 且 $\bar{b}_w$ 中的元素各不相同;

(2) 对任意顶点 $v \in V(\mathfrak{G})$, 集合 $B_v := \{w \in V(\mathfrak{T}) | v \in \bar{b}_w\}$ 非空且导出子图 $G(\mathfrak{T})[B_v]$ 是连通的, 其中 $G(\mathfrak{T})$ 为 $\mathfrak{T}$ 的 Gaifman 图;

(3) 对任意边 $uv \in E^{\mathfrak{G}}$, 存在节点 $w \in V(\mathfrak{T})$ 使得 $u,v \in \bar{b}_w$.

$\mathfrak{G}$ 的宽度最小的树分解称为 $\mathfrak{G}$ 的最优树分解 (不唯一), $\mathfrak{G}$ 的树宽 $tw(\mathfrak{G})$ 定义为最优树分解的宽度. 对于图类 C, 若存在 $m \in \mathbb{N}^+$ 使得任意 $\mathfrak{G} \in C$ 均满足 $tw(\mathfrak{G}) \leqslant m$, 则称 C 为树宽有界图类.

显然, 任何一个图都存在平凡的树分解. 直观来说, 一个图的树宽越小, 则结构上越接近一棵树. 例如无边图的树宽为 0, 节点数量大于 1 的树的树宽为 1, 长度大于 2 的环的树宽为 2, 团 $\mathfrak{K}_\ell$ 的树宽为 $\ell-1$.

例 1. 对任意自然数 $\ell > 1$, $\ell \times \ell$ 格图 (grid) 是顶点集为 $[\ell] \times [\ell]$, 边集为 $\{((i,j),(i_0,j_0)) | |i-i_0| + |j-j_0| = 1\}$ 的图, 其树宽为 ℓ. 记所有格图构成的图类为 $Grid$, $Grid$ 不是树宽有界图类.

树分解包含了图的一些结构信息, 利用树分解可以设计图上的一些算法, 典型的思路是在树分解中自底而上地进行动态规划. 事实上, 树自动机的运行也是一种动态规划, 我们可以类似前文中树自动机的思路设计树宽有界图类上的 MSO 模型检测问题的算法, 甚至可以将树宽也作为参数, 即考虑如下问题 $MC(\mathrm{MSO}, tw)$: 输入图 $\mathfrak{G} \in Graph$, 语句 $\varphi \in \mathrm{MSO}[\tau_E]$, 参数为 $|\varphi| + tw(\mathfrak{G})$, 判定是否有 $\mathfrak{G} \vDash \varphi$. 显然, 若 $MC(\mathrm{MSO}, tw)$ 存在 FPT 算法, 则对任意树宽有界图类 C, 可以得到 $MC(\mathrm{MSO}, C)$ 的 FPT 算法, 即 $MC(\mathrm{MSO}, C)$ 可以归约到 $MC(\mathrm{MSO}, tw)$.

我们利用树分解来设计 $MC(\mathrm{MSO}, tw)$ 的高效算法. 首先面临一个自然的问题: 给定图 $\mathfrak{G}$, 如何得到 $\mathfrak{G}$ 的一个非平凡的树分解? 非平凡是指其宽度应接近于 $tw(\mathfrak{G})$, 最好能直接得到一个最优树分解, 因为 $tw(\mathfrak{G})$ 是参数的一部分, 我们要充分利用这一点. 然而, 输入图 $\mathfrak{G}$ 和自然数 m, 判定是否有 $tw(\mathfrak{G}) = m$ 是一个 NP 完备问题[49], 因此仍只能考虑参数复杂性. Bollaender[50]提出了一个经典的参数算法, 其高效性如定理 6 所示, 算法的具体细节略.

定理 6. 存在一个算法, 输入任意图 $\mathfrak{G}$, 在 $2^{\mathrm{O}(tw(\mathfrak{G})^3)} \cdot \|\mathfrak{G}\|$ 时间内计算 $\mathfrak{G}$ 的一个最优树分解.

Bollaender 算法的运行时间关于 $\|\mathfrak{G}\|$ 是线性的, 在参数复杂性的意义下可以高效地计算图的最优树分解. 给定自然数 m、从 $[m]$ 到 $[m]$ 的部分单射 p 以及顶点集为 $[m]$ 的图 $\mathfrak{H}$, 定义:

(1) $I_m := \{i \in \mathbb{N} \mid m < i \leqslant 2m\}$, $\mathfrak{H}$ 的顶点为 $\bar{u} := (1,2,\ldots,m)$;

(2) $\bar{v}[p] := (v_1, v_2, \ldots, v_m)$, 其中若 i 在 p 的值域中, 则 $v_i = p^{-1}(i)$, 否则 $v_i = m + i$;

(3) $\mathfrak{H}[p] := \mathfrak{H} \cup \mathfrak{I}[\bar{v}[p]]$, $tp(\mathfrak{H}, p) := tp_q^{\mathrm{MSO}}(\mathfrak{H}[p], \bar{v}[p]\bar{u})$, 其中 $\mathfrak{I}[X]$ 为以 X 为顶点集的无边图.

给定图 $\mathfrak{G}$ 和语句 $\varphi \in \mathrm{MSO}[\tau_E]$, $\mathfrak{G}$ 的树宽为 m, φ 的量词阶数为 q, 我们构造下述 DTA: 状态集 S 为 $Type_{q,m}^{\mathrm{MSO}}[\tau_E]$ 中的若干 q 型; 字母表 Σ 为全体三元组 $(\mathfrak{H}, p_1, p_2)$, 其中 $\mathfrak{H}$ 是顶点集为 $[m]$ 的图, p_1, p_2 为从 $[m]$ 到 $[m]$ 的部分单射; 接受状态集 F 为 S 中所有包含 φ 的 q 型; 对任意 $a := (\mathfrak{H}, p_1, p_2) \in \Sigma$, 状态集 S 和状态转移函数 δ 具体如下.

(1) $\delta(\bot, \bot, a) := tp_q^{\mathrm{MSO}}(\mathfrak{H}, \bar{u}) \in S$;

(2) 若 $t \in S$, 则 $\delta(t, \bot, a) := [t \otimes_m tp(\mathfrak{H}, p_1)]_{I_m} \in S$, $\delta(\bot, t, a) := [t \otimes_m tp(\mathfrak{H}, p_2)]_{I_m} \in S$;

(3) 若 $t_1, t_2 \in S$, 则 $\delta(t_1, t_2, a) := [t_1 \otimes_m tp(\mathfrak{H}, p_1)]_{I_m} \otimes_m [t_2 \otimes_m tp(\mathfrak{H}, p_2)]_{I_m} \in S$;

(4) 重复上述操作, 直至 S 不再发生变化.

FV 定理确保了上述过程是良定义的. 使用 Bollaender 算法得到 $\mathfrak{G}$ 的一个最优树分解 $\left(\mathfrak{T}, \left(\bar{b}_w\right)_{w \in V(\mathfrak{T})}\right)$ 之后, 我们将 $\left(\mathfrak{T}, \left(\bar{b}_w\right)_{w \in V(\mathfrak{T})}\right)$ 转化为一棵 Σ 树 $\mathfrak{T}'$, 其边满足 $E_1^{\mathfrak{T}'} \cup E_2^{\mathfrak{T}'} = P^{\mathfrak{T}}$, 左右子节点顺序任意; 对于每个节点 $w \in V(\mathfrak{T}')$, 其左右子节点分别为 v_1, v_2, w 对应的字符为 $a_w := (\mathfrak{H}_w, p_{w,1}, p_{w,2}) \in \Sigma$, 其中 $\mathfrak{H}_w$ 为与 $\mathfrak{G}\left[\bar{b}_w\right]$ 同构的顶点集为 $[m]$ 的图, 同构映射为将 $\bar{b}_w$ 中的第 i 个顶点映射到 i, $p_{w,i}$ 为从 $[m]$ 到 $[m]$ 的部分单射, $(u, j) \in p_{w,i}$ 当且仅当 $\bar{b}_w$ 中的第 u 个顶点 (对应 $\mathfrak{H}$ 中的顶点 u) 出现在 $\bar{b}_{v_i}$ 的第 j 位, 若 v_i 不存在, 则 $p_{w,i} = \varnothing$. 最后我们以 $\mathfrak{T}'$ 为输入运行 DTA, 观察 $\mathfrak{T}'$ 是否被接受. 完整的算法如算法 3.

算法 3. $MC(\mathrm{MSO}, tw)$ 的高效算法.

输入: 图 $\mathfrak{G} \in Graph$, 语句 $\varphi \in \mathrm{MSO}[\tau_E]$;

(1) 使用 Bollaender 算法计算 $\mathfrak{G}$ 的一个最优树分解 $\left(\mathfrak{T}, \left(\bar{b}_w\right)_{w \in V(\mathfrak{T})}\right)$, 并获取 $\mathfrak{G}$ 的树宽 m;

(2) 构造 φ, m 对应的 DTA;

(3) 将 $\left(\mathfrak{T}, \left(\bar{b}_w\right)_{w \in V(\mathfrak{T})}\right)$ 转换为 Σ 树 $\mathfrak{T}'$;

(4) 以 $\mathfrak{T}'$ 为输入运行 DTA.

输出: DTA 是否接受 $\mathfrak{T}'$.

首先说明算法 3 的正确性. 记 $V_w := \cup_{v \text{是} w \text{的后代}} \bar{b}_v$, $\mathfrak{G}_w := \mathfrak{G}[V_w]$, $\mathfrak{G}_{vw} := \mathfrak{G}\left[\bar{b}_w\right] \cup \mathfrak{I}\left[\bar{b}_v\right]$. 根据定义 8 可得引理 2.

引理 2. 对任意 $w \in V(\mathfrak{T})$, 若 w 有子节点 v, 则 $V(\mathfrak{G}_{vw}) \cap V(\mathfrak{G}_v) = \bar{b}_v$, $\mathfrak{G}_w = \mathfrak{G}_v \cup \mathfrak{G}_{vw}$.

我们使用归纳法证明对于 $\mathfrak{T}$ 中任意节点 w, 运行 DTA 后 w 对应的状态恰为 $t_w := tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_w, \bar{b}_w\right)$:

(1) 若 w 为叶节点, 则其对应的字符 $a_w = (\mathfrak{H}_w, \varnothing, \varnothing)$, 故状态为 $\delta(\bot, \bot, a_w) = tp_{q,m}^{\mathrm{MSO}}(\mathfrak{H}_w, \bar{u}) = t_w$;

(2) 若 w 仅有左子节点 v (右子节点类似), w 对应字符 $a_w = (\mathfrak{H}_w, p_{w,1}, \varnothing)$, 则 $tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_{vw}, \bar{b}_v \bar{b}_w\right) = tp(\mathfrak{H}_w, p_{w,1})$, 故根据引理 2 和 FV 定理可得:

$$t_w = \left[tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_w, \bar{b}_v \bar{b}_w\right)\right]_{I_m} = \left[t_v \otimes_m tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_{vw}, \bar{b}_v \bar{b}_w\right)\right]_{I_m} = [t_v \otimes_m tp(\mathfrak{H}_w, p_{w,1})]_{I_m} = \delta(t_v, \bot, a_w);$$

(3) 若 w 有左右子节点 v_1, v_2, w 对应字符 $a_w = (\mathfrak{H}_w, p_{w,1}, p_{w,2})$, 则 $tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_{v_i w}, \bar{b}_{v_i} \bar{b}_w\right) = tp(\mathfrak{H}_w, p_w, i)$, 由引理 2 和 FV 定理可得:

$$\begin{aligned} t_w &= \left[t_{v_1} \otimes_m tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_{v_1 w}, \bar{b}_{v_1} \bar{b}_w\right)\right]_{I_m} \otimes_m \left[t_{v_2} \otimes_m tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_{v_2 w}, \bar{b}_{v_2} \bar{b}_w\right)\right]_{I_m} \\ &= [t_{v_1} \otimes_m tp(\mathfrak{H}_w, p_{w,1})]_{I_m} \otimes_m [t_{v_2} \otimes_m tp(\mathfrak{H}_w, p_{w,2})]_{I_m} = \delta(t_{v_1}, t_{v_2}, a_w). \end{aligned}$$

由此可得, 根节点 r 的状态为 $t_r := tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}_r, \bar{b}_r\right) = tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}, \bar{b}_r\right)$, 故 $\mathfrak{G} \vDash \varphi$ 当且仅当 $\varphi \in tp_{q,m}^{\mathrm{MSO}}\left(\mathfrak{G}, \bar{b}_r\right)$, 当且仅当

DTA 接受 $\mathfrak{T}'$, 故算法 3 是正确的. 算法 3 的高效性分析如下.

(1) 根据定理 6 计算最优树分解 $\left(\mathfrak{T},\left(\bar{b}_w\right)_{w\in V(\mathfrak{T})}\right)$ 所需的时间为 $2^{O(m^3)}\cdot\|\mathfrak{G}\|$;

(2) 根据 FV 定理构造 DTA 所需的时间为仅与 q 和 m 相关的函数;

(3) 根据定义 8, 最优树分解的规模为 $O(m\cdot\|\mathfrak{G}\|)$, 因此将其转化为 Σ 树 $\mathfrak{T}'$ 的时间为 $O(m\cdot\|\mathfrak{G}\|)$;

(4) 以 $\mathfrak{T}'$ 为输入运行 DTA 的时间为 $O(\|\mathfrak{G}\|)$.

可见算法 3 是一个 FPL 算法, 故得 Courcelle 定理 (定理 7).

定理 7[1]. $MC(\mathrm{MSO},tw)\in\mathrm{FPL}$. 具体来说, 存在一个 $f(k)+2^{O(tw(\mathfrak{G})^3)}\cdot n$ 时间的 FPL 算法.

推论 1. 对任意树宽有界图类 C, $MC(\mathrm{MSO},C)\in\mathrm{FPL}$. 具体来说, 存在一个 $O(f(k)+n)$ 时间的 FPL 算法.

同样地, 算法 3 构造的 DTA 除接受状态的选择外与语句 φ 的具体内容无关, 仅与其量词阶数有关. 我们可以对上述结论稍做推广: 给定 τ 结构类 D, 若由 D 中所有结构的 Gaifman 图构成的类 $\{G(\mathfrak{A})\mid\mathfrak{A}\in D\}$ 是树宽有界图类, 则有 $MC(\mathrm{MSO},D)\in\mathrm{FPL}$, 只需在 Coucelle 定理的证明中考虑 Gaifman 图即得证. 需要说明的是, 类似的推广并不对所有的算法元定理都适用, 需要分析具体的证明过程. 在后续的工作中, Bodlaender 算法以及 Courcelle 定理都被优化到了对数空间复杂度[51].

类似的基于自动机和动态规划的策略还能够推广到更复杂的图类上.

定义 9. 给定图类 C, 若存在图 $\mathfrak{H}$ 满足对任意 $\mathfrak{G}\in C$ 均有 $\mathfrak{H}$ 不为 $\mathfrak{G}$ 的子式, 则称 C 为不含某一子式 (excluding a minor) 的图类, 或更具体地称为不含 $\mathfrak{H}$ 子式 ($\mathfrak{H}$-minor free) 的图类.

对于真子式理想 C, 至少存在一个图 $\mathfrak{H}\notin C$, 于是所有以 $\mathfrak{H}$ 为子式的图也不属于 C, 故真子式理想是不含某一子式的图类. 易见, 一个图类 C 是不含某一子式的图类当且仅当 C 的子式闭包为真子式理想. 通过简单的分析可得, 对任意图 $\mathfrak{G},\mathfrak{H}$, 若 $\mathfrak{H}$ 为 $\mathfrak{G}$ 的子式, 则 $tw(\mathfrak{H})\leqslant tw(\mathfrak{G})$. 因此, 树宽有界图类的子式闭包仍为树宽有界图类, 故为真子式理想 (显然 *Graph* 不为树宽有界图类), 于是树宽有界图类为不含某一子式的图类. 后文中我们将看到反方向不成立, 即不含某一子式的图类是树宽有界图类的严格推广. 关于不含某一子式的图类有定理 8.

定理 8[3]. 对于任意不含某一子式的图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$.

定理 8 的证明基于真子式理想的一种具有特殊性质的树分解, 通过类似前面的动态规划过程自底而上地计算每个节点对应的 q 型. 由于这种树分解并不要求宽度有界, 而是要求相邻两个包的公共元素的数量有界, 动态规划的时间只依赖于参数. 该算法中 q 型的计算使用了 FO 的局部性 (locality), 故只对 FO 成立, FO 的局部性将在第 3 节中具体介绍. 需要注意的是, 证明中用到了有关 C 的一些存在性的结论, 因此可能存在一些不可计算的常数. 这一问题在后续的工作中得到了解决, Dawar 等人[52]证明了下述问题 $MC(\mathrm{FO},minor)$ 是 FPT 的: 输入图 $\mathfrak{G},\mathfrak{H}\in Graph$, $\mathfrak{H}$ 不为 $\mathfrak{G}$ 的子式, 语句 $\varphi\in\mathrm{FO}[\tau_E]$, 参数为 $|\varphi|+|\mathfrak{H}|$, 判定是否有 $\mathfrak{G}\vDash\varphi$.

定理 9[52]. $MC(\mathrm{FO},minor)\in\mathrm{FPT}$.

定理 9 的证明也是类似的动态规划算法, 但需要基于一些可以根据 $\mathfrak{H}$ 计算出的更深刻的结构性质[52].

3 基于 FO 局部性的研究

第 2 节中我们已经看到 MSO 模型检测问题在一些常见图类上是 FPT 的, 其中较典型的是树宽有界图类. 事实上, 树宽有界图类在稀疏图类上已经较好地刻画了 MSO 的易解范围, 稍微复杂一些的图类上的 MSO 模型检测问题就不太可能是易解的. 例如我们考虑如定义 10 所述图类.

定义 10. 给定图类 C: (1) 若存在 $m\in\mathbb{N}^+$ 使得任意 $\mathfrak{G}\in C$ 的度不超过 m, 则称 C 为度有界图类. (2) 若任意 $\mathfrak{G}\in C$ 均为平面图 (planar graph)[53], 则称 C 为平面图类. 记所有平面图构成的类为 *Planar*.

关于平面图类, 有著名的 Wagner 定理 (定理 10).

定理 10[54]. 任意图 $\mathfrak{G}$ 是平面图当且仅当团 $\mathfrak{K}_5$ 和二分团 $\mathfrak{K}_{3,3}$ 不为 $\mathfrak{G}$ 的子式.

我们用一些例子熟悉一下这些新的图类. 回顾例 1, *Grid* 既为度有界图类, 又为平面图类, 但不为树宽有界图类, 故 *Planar* 不为树宽有界图类; *Tree* 既为树宽有界图类, 又为平面图类, 但不为度有界图类; 根据定理 10, $\{\mathfrak{K}_5\}$ 既

为度有界图类, 又为树宽有界图类, 但不为平面图类; *Planar* 为真子式理想, 故平面图类为不含某一子式的图类; 度有界图类的子式闭包仍为度有界图类, 故为真子式理想, 因此度有界图类也为不含某一子式的图类. 定理 11 说明在一般情况下平面图类或者度有界图类上的 MSO 模型检测问题不太可能是易解的.

定理 11[12]. 所有度不超过 4 的平面图构成的图类 C 上的 3-着色问题是 NP 完备的.

由于一个图存在 3-着色可以描述为 MSO 语句 ψ_{3col}, 故上述定理即模型检测问题 $MC(\{\psi_{3col}\},C)$ 是 NP 完备的, 因此只要 NP ≠ P, 则有 $MC(\mathrm{MSO},C)\notin \mathrm{FPT}$.

一般来说, MSO 的表达能力强于 FO, 因此 FO 模型检测问题的易解范围应当比 MSO 更广, 即 FO 模型检测问题可能在更多的图类上是易解的, 更有希望产生新的算法元定理. 因此下一阶段的研究重心转向 FO 模型检测的易解性, 而这是一个更加复杂的课题, 时至今日相关理论仍在继续发展. FO 模型检测的早期研究主要使用一些基于 FO 局部性的方法, 下面将详细介绍这类技术的框架与相关的结论.

从某种意义上说, FO 所能描述的性质都是"局部的", 这一特征被称为 FO 的局部性, 局部性是 FO 与 MSO (乃至更一般的二阶逻辑) 最本质的区别之一. 为了形式化地描述 FO 的局部性, 需要引入一些概念: 给定自然数 r、图 $\mathfrak{G}$ 和顶点 $v\in V(\mathfrak{G})$, 定义 v 的 r 邻域 (r-neighborhood) $N_r^{\mathfrak{G}}(v)$ 为 $\mathfrak{G}$ 中所有到 v 的距离不超过 r 的顶点的集合, 即 $N_r^{\mathfrak{G}}(v):=\left\{u\in V(\mathfrak{G})\mid dist^{\mathfrak{G}}(u,v)\leqslant r\right\}$, r 称为该邻域的半径 (radius). 进一步, 顶点集合 U 的 r 邻域 $N_r^{\mathfrak{G}}(U)$ 定义为 U 中所有顶点的 r 邻域的并集. 对任意结构 $\mathfrak{A}$ 和 $a\in V(\mathfrak{A})$, a 的 r 邻域 $N_r^{\mathfrak{A}}(a)$ 定义为 Gaifman 图 $G(\mathfrak{A})$ 中 a 的 r 邻域 $N_r^{G(\mathfrak{A})}(a)$. 由于 $\mathfrak{A}$ 的符号集 τ 是有限集, 很容易构造一个 FO 公式 $dist^{\leqslant r}(x,y)$ 满足 $\mathfrak{A}\vDash dist^{\leqslant r}(a,b)$ 当且仅当 $dist^{\mathfrak{A}}(a,b)\leqslant r$. 类似地有 $dist^{<r}(x,y)$, $dist^{>r}(x,y)$, $dist^{\geqslant r}(x,y)$ 等. FO 公式的相对化 (relativization) 定义如定义 11.

定义 11. 给定任意 FO 公式 $\varphi(x)$, 其对应的 r 相对化公式 $\varphi^r(x)$ 归纳构造如下:

(1) 若 φ 为原子公式, 则 $\varphi^r:=\varphi$;

(2) $(\neg\varphi)^r:=\neg\varphi^r$, $(\varphi*\psi)^r:=\varphi^r*\psi^r$, 其中 $*\in\{\wedge,\vee\}$;

(3) $(\exists y\varphi)^r:=\exists y\left(dist^{\leqslant r}(x,y)\wedge\varphi^r\right),(\forall y\varphi)^r:=\forall y\left(dist^{\leqslant r}(x,y)\to\varphi^r\right)$.

本文只讨论仅含一个自由变元 x 的 FO 公式的相对化. 直观来说, 相对化即将所有量词都限制到自由变元 x 的 r 邻域中, 因此公式的满足性也仅与结构的某个 r 邻域有关. 形式化地, 对于任意结构 $\mathfrak{A}$ 和 $a\in V(\mathfrak{A})$, $\mathfrak{A}\vDash\varphi^r(a)$ 当且仅当 $\mathfrak{N}_r^{\mathfrak{A}}(a)\vDash\varphi^r(a)$, 其中 $\mathfrak{N}_r^{\mathfrak{A}}(a):=\mathfrak{A}[N_r^{\mathfrak{A}}(a)]$ 为 a 的 r 邻域对应的导出子结构, 类似地对任意 $U\subseteq V(\mathfrak{A})$ 定义 $\mathfrak{N}_r^{\mathfrak{A}}(U)=\mathfrak{A}[N_r^{\mathfrak{A}}(U)]$. 基本局部语句 (basic local sentence) 是指具有如下形式的语句 (对某 $\ell,r\in\mathbb{N}$):

$$\xi:=\exists x_1\ldots\exists x_\ell\left(\bigwedge_{1\leqslant i<j\leqslant\ell}dist^{>2r}(x_i,x_j)\wedge\bigwedge_{i\in[\ell]}\psi^r(x_i)\right).$$

若干基本局部语句的布尔组合 (即由逻辑运算符 $\neg,\vee,\wedge$ 连接而成的语句) 称为局部语句 (local sentence). Gaifman 定理 (定理 12) 形式化地描述了 FO 的局部性.

定理 12[55]. 任意 FO 语句 φ 等价于一个局部语句 φ', 而且根据 φ 可以计算出 φ'.

Gaifman 原始的证明[47,55]基于归纳法和 FV 定理, 其证明过程是构造性的, 即给出了根据 φ 计算 φ' 的算法. 另有一些基于 EF 博弈的现代证明, 更富有限模型论特色[26]. 需要指出的是, 有些 FO 语句 φ 对应的 φ' 的长度至少是 $|\varphi|$ 的塔函数[56]. 算法元定理领域中最典型的基于 FO 局部性的方法可以抽象为下述定理, 该定理最早用于证明局部树宽有界图类上模型检测问题的参数易解性[4], 后续工作对此方法进行了系统化的描述[47,57,58], 但均与本文中的叙述存在细节上的差异, 有些文献甚至存在一些疏漏, 此处一并修正.

定理 13. 令 U 为一元关系. 给定图类 C, 定义问题 $MC^{\mathrm{loc}}(\mathrm{FO},C)$ 如下: 输入图 $\mathfrak{G}\in C,W\subseteq V(\mathfrak{G}),v\in V(\mathfrak{G})$, 语句 $\varphi\in\mathrm{FO}[\tau_E\cup\{U\}]$, $r\in\mathbb{N}$, 参数为 $|\varphi|+r$, 判定是否 $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v)\vDash\varphi$. 若 $MC^{\mathrm{loc}}(\mathrm{FO},C)\in\mathrm{FPT}$, 则 $MC(\mathrm{FO},C)\in\mathrm{FPT}$. 具体地, 若前者存在一个 c 次方时间的 FPT 算法 ($c\in\mathbb{R}$), 则后者存在一个 $c+1$ 次方时间的 FPT 算法.

下面详细证明定理 13. 依题设 $MC^{\mathrm{loc}}(\mathrm{FO},C)$ 存在算法 A. 考虑模型检测问题 $MC(\mathrm{FO},C)$: 给定图 $\mathfrak{G}\in C$ 和公式 $\varphi\in\mathrm{FO}[\tau_E]$, 判定是否有 $\mathfrak{G}\vDash\varphi$. 首先, 使用 Gaifman 定理计算 φ 对应的局部公式 φ'. 由于 φ' 是一些基本局部语句的布尔组合, 故原问题可以归约为判定基本局部语句在 $\mathfrak{G}$ 中的满足性. 考虑基本局部语句 ξ, 我们设计算法 4 判定

$\mathfrak{G}\models\xi$ 是否成立.

算法 4. 基本局部语句的模型检测算法.

输入: 图 $\mathfrak{G}\in C$, 基本局部语句 ξ;

输出: $\mathfrak{G}\models\xi$ 是否成立.

(1) 调用算法 A, 计算顶点集合 $W:=\{v\in V(\mathfrak{G})\mid \mathfrak{R}_r^{\mathfrak{G}}(v)\models\psi^r(v)\}$.

(2) 计算 W 的子集 $V\leftarrow GetVoters(W,2r)$. 若 $|V|=\ell$, 输出"成立".

函数 $GetVoters(X,m)$ 的输入为 $X\subseteq V(\mathfrak{G})$, $m\in\mathbb{N}$, 具体实现如下:

$X'\leftarrow X, Y\leftarrow\varnothing$;

While $X'\neq\varnothing$ 且 $|Y|\neq\ell$:

任取 $x\in X'$, $Y\leftarrow Y\cup\{x\}$, $X'\leftarrow X'-N_m^{\mathfrak{G}}(x)$;

返回 Y;

(3) 否则, 计算 V 的子集 $S\leftarrow GetBalls(V,3r)$. 记 S 中元素为 $(v_1,r_1),\ldots,(v_{|S|},r_{|S|})$.

函数 $GetBalls(X,m)$ 的输入为 $X\subseteq V(\mathfrak{G})$, $m\in\mathbb{N}$, 具体实现如下:

$Y\leftarrow X$

For $x\in X$

$m_x\leftarrow m$;

While 存在 $y\neq z\in Y$ 使得 $N_{m_y}^{\mathfrak{G}}(y)\cap N_{m_z}^{\mathfrak{G}}(z)\neq\varnothing$:

$m_y\leftarrow m_y+2m_z, Y\leftarrow Y-\{z\}$;

返回 $\{(x,m_x)\mid x\in Y\}$;

(4) 穷举将 ℓ 分解为 $|S|$ 个正整数之和的所有方式, 对每一分解 $\ell=\ell_1+\ldots+\ell_{|S|}$ 和所有 $i\in[|S|]$, 调用算法 A 判定是否有 $\mathfrak{R}_{r_i}^{\mathfrak{G}\langle U\mapsto W\rangle}(v_i)\models\vartheta_{\ell_i}$, 其中,

$$\vartheta_m:=\exists x_1\ldots\exists x_m\left(\bigwedge_{1\leqslant i<j\leqslant m}dist^{>2r}(x_i,x_j)\wedge\bigwedge_{i\in[m]}Ux_i\right).$$

若存在某一分解 $\ell=\ell_1+\ldots+\ell_{|S|}$ 满足 $\mathfrak{R}_{r_i}^{\mathfrak{G}\langle U\mapsto W\rangle}(v_i)\models\vartheta_{\ell_i}$ 对所有 $i\in[|S|]$ 均成立, 输出"成立"; 否则输出"不成立".

首先说明算法 4 的正确性. W 即 $\mathfrak{G}$ 中所有满足 $\psi^r(x)$ 的顶点集合, 故 $\mathfrak{G}\models\xi$ 当且仅当 W 中存在两两在 $\mathfrak{G}$ 中距离大于 $2r$ 的 ℓ 个顶点. 根据函数 $GetVoters$ 的定义可得 $V\subseteq W$, 并且 V 中的任意两个顶点在 $\mathfrak{G}$ 中的距离大于 $2r$. 显然, 若 $|V|=\ell$, 则 $\mathfrak{G}\models\xi$. 若 $|V|<\ell$, 函数 $GetBalls$ 从 V 中选取了 $|S|$ 个顶点 $v_1,\ldots,v_S$, 并为每个 v_i 分配了一个自然数 r_i, 其含义是选取了 v_i 的 r_i 邻域 (邻域亦称球 (ball)). 由 $GetBalls$ 的定义可得, 对任意 $i\in[|S|]$, 均有 $r_i\geqslant 3r$, 并且这些邻域是两两不相交的. 下述断言说明这些邻域覆盖了 $N_{3r}^{\mathfrak{G}}(V)$, 而且这种覆盖具有很好的性质.

断言 1. 对任意 $u\in V$, 存在 $i\in[|S|]$ 满足 $N_{3r}^{\mathfrak{G}}(u)\subseteq N_{r_i}^{\mathfrak{G}}(v_i)$.

证明: 对 $GetBalls$ 中 While 循环的迭代次数 j 使用归纳法, 证明对任意 $j\geqslant 0$ 均有: 对任意 $x\in X$, 存在 $a\in Y_j$ 满足 $N_m^{\mathfrak{G}}(x)\subseteq N_{m_{a,j}}^{\mathfrak{G}}(a)$, 其中 $Y_j, m_{a,j}$ 分别是变量 Y, m_a 在第 j 轮迭代结束时的值. $j=0$ 时, 平凡成立. 假设对第 $j-1$ 轮迭代也成立, 现考虑第 j 轮迭代. 迭代开始时, 选取 $y\neq z\in Y_{j-1}$ 使得 $N_{m_{y,j-1}}^{\mathfrak{G}}(y)\cap N_{m_{z,j-1}}^{\mathfrak{G}}(z)\neq\varnothing$. 然后将 m_y 增大, 即 $m_{y,j}:=m_{y,j-1}+2m_{z,j-1}$, 并从 Y 中删除 z, 即 $Y_j:=Y_{j-1}-\{z\}$. 由于 y,z 的距离不超过 $m_{y,j-1}+m_{z,j-1}$, 恰好使得 $N_{m_{z,j-1}}^{\mathfrak{G}}(z)\subseteq N_{m_{y,j}}^{\mathfrak{G}}(y)$. 根据归纳假设, 对任意 $x\in X$ 均存在 $a\in Y_{j-1}$ 使得 $N_m^{\mathfrak{G}}(x)\subseteq N_{m_{a,j-1}}^{\mathfrak{G}}(a)$, 于是有: 若 $a\in\{y,z\}$, 则 $N_m^{\mathfrak{G}}(x)\subseteq N_{m_{a,j-1}}^{\mathfrak{G}}(a)\subseteq N_{m_{a,j}}^{\mathfrak{G}}(y)$; 否则 $m_{a,j-1}=m_{a,j}$, 故 $N_m^{\mathfrak{G}}(x)\subseteq N_{m_{a,j-1}}^{\mathfrak{G}}(a)\subseteq N_{m_{a,j}}^{\mathfrak{G}}(a)$. 证毕.

由于邻域 $N_{r_i}^{\mathfrak{G}}(v_i)$ 互不相交, 可得 W 的一个划分 $\left\{W_i\mid W_i=W\cap N_{r_i}^{\mathfrak{G}}(v_i), i\in[|S|]\right\}$. 由于 $v_i\in V\subseteq W$, 故 W_i 均非空. 于是有断言 2.

断言 2. 对任意 $i\neq j\in[|S|]$, $dist^{\mathfrak{G}}(W_i,W_j)>2r$.

证明: 此时 $|V|<\ell$. 根据 *GetVoters* 的定义可得 $W\subseteq N^{\mathfrak{G}}_{2r}(V)$, 即对任意 $w\in W$ 均存在 $u\in V$ 使得 $w\in N^{\mathfrak{G}}_{2r}(u)$, 故 $N^{\mathfrak{G}}_{r}(w)\subseteq N^{\mathfrak{G}}_{3r}(u)$. 进一步由断言 1 可得, 存在 $i'\in[|S|]$ 满足 $N^{\mathfrak{G}}_{r}(w)\subseteq N^{\mathfrak{G}}_{3r}(u)\subseteq N^{\mathfrak{G}}_{r_{i'}}(v_{i'})$. 现考虑任意 $w_1\in W_i$ 与 $w_2\in W_j$, 必有 $N^{\mathfrak{G}}_{r}(w_1)\subseteq N^{\mathfrak{G}}_{r_i}(v_i)$, $N^{\mathfrak{G}}_{r}(w_2)\subseteq N^{\mathfrak{G}}_{r_j}(v_j)$. 由于 $N^{\mathfrak{G}}_{r_i}(v_i)\cap N^{\mathfrak{G}}_{r_j}(v_j)=\varnothing$, 故 $dist^{\mathfrak{G}}(w_1,w_2)>2r$. 证毕.

我们的目标是判定 W 中是否存在两两在 $\mathfrak{G}$ 中距离大于 $2r$ 的 ℓ 个顶点, 根据断言 2, 我们可以将该问题转化为求每个 W_i 中两两在 $\mathfrak{G}$ 中距离大于 $2r$ 的顶点数量, 然后全部相加. 形式化描述如下: 存在两两在 $\mathfrak{G}$ 中距离大于 $2r$ 的 ℓ 个 W 中的顶点, 当且仅当存在将 ℓ 分解成 $|S|$ 个正整数之和的一种分解 $\ell=\ell_1+\ldots+\ell_{|S|}$ 使得对于所有 $i\in[|S|]$, W_i 中存在两两在 $\mathfrak{G}$ 中距离大于 $2r$ 的 ℓ_i 个顶点. 断言 3 将后者归约为问题 $MC^{\mathrm{loc}}(\mathrm{FO},C)$ 的实例.

断言 3. 对任意 $i\in[|S|]$ 和 $w_1,w_2\in W_i$, $dist^{N^{\mathfrak{G}}_{r_i}(v_i)}(w_1,w_2)>2r$ 当且仅当 $dist^{\mathfrak{G}}(w_1,w_2)>2r$.

证明: 充分性是显然的, 下证必要性. 若 $dist^{\mathfrak{G}}(w_1,w_2)\leqslant 2r$, 则在 $\mathfrak{G}$ 中 w_1,w_2 之间的任意一条最短路径包含于 $N^{\mathfrak{G}}_{r}(w_1)\cup N^{\mathfrak{G}}_{r}(w_2)$. 根据断言 2 的证明过程, $N^{\mathfrak{G}}_{r}(w_1)\cup N^{\mathfrak{G}}_{r}(w_2)\subseteq N^{\mathfrak{G}}_{r_i}(v_i)$, 故 $dist^{N^{\mathfrak{G}}_{r_i}(v_i)}(w_1,w_2)\leqslant 2r$. 证毕.

根据断言 3, 只需判定是否存在将 ℓ 分解成 $|S|$ 个正整数之和的一种分解 $\ell=\ell_1+\ldots+\ell_{|S|}$ 使得对于所有 $i\in[|S|]$, W_i 中存在两两在 $\mathfrak{N}^{\mathfrak{G}}_{r_i}(v_i)$ 中距离大于 $2r$ 的 ℓ_i 个顶点. 注意到 $\mathfrak{N}^{\mathfrak{G}\langle U\mapsto W\rangle}_{r_i}(v_i)=\mathfrak{N}^{\mathfrak{G}\langle U\mapsto W_i\rangle}_{r_i}(v_i)$, 故 W_i 中存在两两在 $\mathfrak{N}^{\mathfrak{G}}_{r_i}(v_i)$ 中距离大于 $2r$ 的 ℓ_i 个顶点当且仅当 $\mathfrak{N}^{\mathfrak{G}\langle U\mapsto W\rangle}_{r_i}(v_i)\vDash\vartheta_{\ell_i}$. 综上, 算法 4 的正确性得证. 设算法 A 的运行时间为 $f(k)\cdot n^c$, 算法 4 的时间复杂性分析如下.

(1) 计算 W: 由于 $\psi'\in\mathrm{FO}[\tau_E\cup\{U\}]$, 故对任意 $v\in V(\mathfrak{G})$, $\mathfrak{N}^{\mathfrak{G}}_{r}(v)\vDash\psi'(v)$ 当且仅当 $\mathfrak{N}^{\mathfrak{G}\langle U\mapsto\varnothing\rangle}_{r}(v)\vDash\psi'(v)$, 因此可以调用算法 A 计算集合 W, 需要遍历所有顶点, 时间为 $f(|\psi'|+r)\cdot\mathrm{O}\left(\|\mathfrak{G}\|^{c+1}\right)$;

(2) 计算 V: *GetVoters* 函数中, While 循环至多迭代 ℓ 轮, 每轮需要的时间为 $f'(r)\cdot\mathrm{O}(\|\mathfrak{G}\|)$, 其中 f' 为可计算函数, 总时间为 $\ell\cdot f'(r)\cdot\mathrm{O}(\|\mathfrak{G}\|)$;

(3) 计算 S: *GetBalls* 函数中, For 循环为初始化过程, 需要 $\mathrm{O}(\|\mathfrak{G}\|)$ 时间; 由于 $|V|<\ell$ 且 Y 是严格变小的, 故 While 循环至多迭代 $\ell-1$ 轮, 每轮中主要的时间开销在于判断循环条件, 经过简单的分析可得 m_y,m_z 不超过 $3^{\ell}\cdot r$, 因此 While 循环需要的时间为 $f''(\ell+r)\cdot\mathrm{O}(\|\mathfrak{G}\|)$, 其中 f'' 为可计算函数. 故总时间为 $f''(\ell+r)\cdot\mathrm{O}(\|\mathfrak{G}\|)$;

(4) 最后一步: ℓ 的分解方式的数量仅与 ℓ 有关. 调用算法 A 的次数为 $|S|^2<\ell^2$, 每次调用所需的时间为 $f\left(|\vartheta_\ell|+3^{\ell}\cdot r\right)\cdot\mathrm{O}(\|\mathfrak{G}\|^c)$. 故总时间为 $f'''(\ell+r)\cdot\mathrm{O}(\|\mathfrak{G}\|^c)$, 其中 f''' 为可计算函数.

注意到 $|\psi'|,r,\ell$ 均可以根据 ξ 计算出, 而 φ' 可以根据 φ 计算出, 即 φ' 中基本局部语句的长度和数量均为 $|\varphi|$ 的可计算函数, 故算法 4 为模型检测问题 $MC(\mathrm{FO},C)$ 的一个 $c+1$ 次方时间的 FPT 算法. 定理 13 证毕.

20 世纪 90 年代中期, Seese 证明了对于任意度有界图类 C 和 FO 语句 φ, 模型检测问题 $MC(\{\varphi\},C)$ 是线性时间可解的[2], 这一经典结论是最早的基于 FO 局部性的算法元定理, 其原始证明基于 Hanf 定理. Hanf 定理从另一个角度描述了 FO 的局部性, 其最初版本讨论的是无穷结构[59], Fagin 等人将结论迁移到了有限结构上[60], 该定理在有限模型论中的叙述与证明又有所不同[26]. 需要注意的是, Seese 的工作[2]并不能说明 $MC(\mathrm{FO},C)\in\mathrm{FPL}$, 因为证明中给出的算法使用了一些与 φ 有关但不可根据 φ 计算的信息. 事实上, 使用定理 13 就可以得到.

定理 14. 对任意度有界图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPL}$.

证明: 考虑问题 $MC^{\mathrm{loc}}(\mathrm{FO},C)$, 输入为图 $\mathfrak{G}\in C$, 语句 $\varphi\in\mathrm{FO}[\tau_E\cup\{U\}]$, $v\in V(\mathfrak{G}),W\subseteq V(\mathfrak{G}),r\in\mathbb{N}$. 由于 C 为度有界图类, 设 C 中图的度有上界 m, 则有 $deg(\mathfrak{G})\leqslant m$. 容易得出, $\mathfrak{N}^{\mathfrak{G}}_{r}(v)$ 的中顶点数量不超过 $(m+1)^r$, 因此可以在关于 $\|\mathfrak{G}\|$ 的常数时间内计算 $\mathfrak{N}^{\mathfrak{G}\langle U\mapsto W\rangle}_{r}(v)$ 并通过穷举判定其是否满足 φ. 因此 $MC^{\mathrm{loc}}(\mathrm{FO},C)$ 存在常数时间的 FPT 算法, 根据定理 13 可得 $MC(\mathrm{FO},C)\in\mathrm{FPL}$. 证毕.

前文提到, 定理 13 最初用于证明局部树宽有界图类上模型检测问题的易解性, 局部树宽有界图类是一类更复杂的稀疏图类, 其定义如定义 12.

定义 12[61]. 图 $\mathfrak{G}$ 的局部树宽 (local tree-width) 函数 $ltw^{\mathfrak{G}}:\mathbb{N}\to\mathbb{N}$ 定义为 $ltw^{\mathfrak{G}}(r)=\max_{v\in V(\mathfrak{G})}\{tw(\mathfrak{N}^{\mathfrak{G}}_{r}(v))\}$. 给定图类 C, 若存在可计算函数 $g:\mathbb{N}\to\mathbb{N}$ 使得对任意 $\mathfrak{G}\in C,r\in\mathbb{N}$ 均有 $ltw^{\mathfrak{G}}(r)\leqslant g(r)$, 则称 C 为局部树宽有界图类.

直观上, 局部树宽有界图类的稀疏性体现在 $\mathfrak{G}$ 的每个邻域的树宽都是有界的, 这说明 $\mathfrak{G}$ 在每个局部中都是相对稀疏的, 但 $\mathfrak{G}$ 的平均度不一定有界[4]. 事实上, 后面我们将会看到局部稀疏性比平均度定义的稀疏性更加本质.

应用定理 13 可得定理 15.

定理 15[4]. 对任意局部树宽有界图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$. 具体地, 存在一个平方时间的 FPT 算法.

证明: 考虑问题 $MC^{\mathrm{loc}}(\mathrm{FO},C)$, 输入为图 $\mathfrak{G}\in C$, 语句 $\varphi\in\mathrm{FO}[\tau_E\cup\{U\}]$, $v\in V(\mathfrak{G}),W\subseteq V(\mathfrak{G}),r\in\mathbb{N}$. 显然, $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v)$ 的计算至多需要 $\mathrm{O}(\|\mathfrak{G}\|)$ 时间. 根据定义 12, 存在与 $\mathfrak{G}$ 无关的可计算函数 $g:\mathbb{N}\to\mathbb{N}$ 使得 $tw(\mathfrak{N}_r^{\mathfrak{G}}(v))\leqslant g(r)$. 如前文所述, Courcelle 定理可以通过考虑相应 Gaifman 图的树宽从而推广到一般的结构上, 因此存在一个算法在 $f(|\varphi|)+2^{\mathrm{O}(g(r)^3)}\cdot\|\mathfrak{G}\|$ 时间内判定是否有 $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v)\vDash\varphi$. 因此 $MC^{\mathrm{loc}}(\mathrm{FO},C)\in\mathrm{FPL}$, 根据定理 13 即得结论. 证毕.

许多常见图类是局部树宽有界图类, 平凡的例子如度有界图类、树宽有界图类等, 非平凡的例子如平面图类[62]、亏格有界 (bounded genus) 图类[63]等. 亏格 (genus) 是拓扑学中的概念, 直观上是指一个曲面沿闭合曲线剪开而不断成多个部分的最多可剪次数, 例如平面与球面的亏格为 0, 甜甜圈形曲面的亏格为 1 等. 若图 $\mathfrak{G}$ 能在曲面 S 上边互不相交地画出来, 则称 $\mathfrak{G}$ 能够嵌入 S. $\mathfrak{G}$ 的亏格 $gen(\mathfrak{G})$ 定义为 $\mathfrak{G}$ 能嵌入的曲面的最小亏格. 若图类 C 中所有图的亏格均不超过某个固定的常数, 则称 C 为亏格有界图类. 根据定义, 平面图类显然为亏格有界图类. 研究表明, 平面图类[62]和亏格有界图类[63]均为局部树宽有界图类, 于是有推论 2 和推论 3.

推论 2. 对任意平面图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$. 具体地, 存在一个平方时间的 FPT 算法.

推论 3. 对任意亏格有界图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$. 具体地, 存在一个平方时间的 FPT 算法.

根据定义易得, 平面图类 (或亏格有界图类) 的子式闭包仍为平面图类 (或亏格有界图类). 但下例说明, 局部树宽有界图类的子式闭包不一定为局部树宽有界图类.

例 2[57]: 对任意 $\ell\in\mathbb{N}^+$, 记 $\mathfrak{S}_\ell$ 为将团 $\mathfrak{K}_\ell$ 中的每条边都替换成一条长度为 ℓ 的路径所得到的图, 并且满足这些路径是互不相交的, 即相当于在每条边上插入 $\ell-1$ 个顶点. 任意给定 $r\in\mathbb{N}$, 若 $\ell\geqslant r$, 则 $\mathfrak{S}_\ell$ 的任意 r 邻域均为一棵树, 故 $ltw^{\mathfrak{S}_\ell}(r)\leqslant 1$; 若 $\ell<r$, 则 $\mathfrak{S}_\ell$ 的任意 r 邻域至多为整个 $\mathfrak{S}_\ell$, 故 $ltw^{\mathfrak{S}_\ell}(r)\leqslant tw(\mathfrak{S}_\ell)=\ell-1<r-1$. 故对任意 $r\in\mathbb{N}$, $ltw^{\mathfrak{S}_\ell}(r)\leqslant r+1$, 即由所有的 $\mathfrak{S}_\ell$ 构成的图类 S 是局部树宽有界图类. 但显然 S 的子式闭包为 $Graph$, 因为它包含所有的团.

由例 2 还可看出, 局部树宽有界图类有可能不为不含某一子式的图类. 反过来, 不含某一子式的图类也有可能不为局部树宽有界图类, 一个典型的例子如例 3.

例 3: 对任意图 $\mathfrak{G}$, 定义 $\mathfrak{G}_{\mathrm{apex}}$ 为在 $\mathfrak{G}$ 中增加一个顶点并使之与 $\mathfrak{G}$ 中所有顶点之间均有边所得到的图. 对任意图类 C, 定义 $C_{\mathrm{apex}}=\{\mathfrak{G}_{\mathrm{apex}}\mid\mathfrak{G}\in C\}$. 直观上, 在 $\mathfrak{G}_{\mathrm{apex}}$ 的一个半径很小的邻域中就包含整个 $\mathfrak{G}$. 由于 $Planar$ 不为树宽有界图类, 故 $Planar_{\mathrm{apex}}$ 不为局部树宽有界图类. 但由定理 10 易得, $Planar_{\mathrm{apex}}$ 中的图均不以 $\mathfrak{K}_6$ 为子式, 故 $Planar_{\mathrm{apex}}$ 为不含某一子式的图类.

类似地可以定义局部不含某一子式 (locally excluding a minor) 的图类, 即不含某一子式的图类 (见定义 9) 的局部化版本[52].

定义 13. 给定图类 C, 若存在可计算函数 $g:\mathbb{N}\to\mathbb{N}$ 使得对任意 $r\in\mathbb{N}$ 均存在图 $\mathfrak{H}$ 满足 $\|\mathfrak{H}\|\leqslant g(r)$ 且对任意 $\mathfrak{G}\in C,v\in V(\mathfrak{G})$ 均有 $\mathfrak{H}$ 不为 $\mathfrak{N}_r^{\mathfrak{G}}(v)$ 的子式, 则称 C 为局部不含某一子式的图类.

平凡地, 不含某一子式的图类均为局部不含某一子式的图类; 根据定义 12, 对任意 $r\in\mathbb{N}$, 团 $\mathfrak{K}_{g(r)}$ 均不可能是某一 r 邻域的子式, 故局部树宽有界图类均为局部不含某一子式的图类. 但例 2 和例 3 表明, 反方向均不成立, 即局部不含某一子式的图类是局部树宽有界图类和不含某一子式的图类的严格推广. 回顾定理 9, 给定图 $\mathfrak{G},\mathfrak{H}$ 满足 $\mathfrak{H}$ 不为 $\mathfrak{G}$ 的子式, 则可以高效地判定 $\mathfrak{G}$ 中 FO 语句的满足性. 引理 3 是图子式理论中的著名结论, 它表明可以在多项式时间内判定两个图之间是否存在子式关系.

引理 3[64]. 对任意固定的图 $\mathfrak{H}$, 存在一个算法, 输入任意图 $\mathfrak{G}$, 在 $\mathrm{O}\left(\|\mathfrak{G}\|^3\right)$ 时间内判定 $\mathfrak{H}$ 是否为 $\mathfrak{G}$ 的子式.

结合定理 13 可得定理 16.

定理 16[52]. 对任意局部不含某一子式的图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$.

证明: 考虑问题 $MC^{\mathrm{loc}}(\mathrm{FO},C)$, 输入为图 $\mathfrak{G}\in C$, 语句 $\varphi\in\mathrm{FO}[\tau_E\cup\{U\}]$, $v\in V(\mathfrak{G}),W\subseteq V(\mathfrak{G}),r\in\mathbb{N}$. 显然, $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v)$ 的计算至多需要 $O(\|\mathfrak{G}\|)$ 时间. 根据定义 13, 存在与 $\mathfrak{G}$ 无关的可计算函数 $g:\mathbb{N}\to\mathbb{N}$ 和某个不为 $\mathfrak{N}_r^{\mathfrak{G}}(v)$

子式的图 $\mathfrak{H}$ 满足 $\|\mathfrak{H}\| \leqslant g(r)$. 由于 g 为可计算函数, 可以先计算出 $g(r)$, 然后穷举所有规模不超过 $g(r)$ 的图, 并通过引理 3 找到一个符合条件的 $\mathfrak{H}$, 整个搜索 $\mathfrak{H}$ 的过程是固定参数多项式时间的. 考察定理 9 证明的细节[52], 可以通过考虑相应的 Gaifman 图从而将该定理推广到一般的结构上, 故存在一个 FPT 算法判定是否有 $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v) \vDash \varphi$. 因此 $MC^{\mathrm{loc}}(\mathrm{FO}, C) \in \mathrm{FPL}$, 根据定理 13 即得结论. 证毕.

事实上, 在上述证明中并不需要搜索一个 $\mathfrak{H}$, 可以直接使用团 $\mathfrak{K}_{g(r)}$, 因为 $\mathfrak{K}_{g(r)}$ 显然不为 $\mathfrak{N}_r^{\mathfrak{G}}(v)$ 的子式, 证明仍然成立. 这也说明定义 13 和下述定义 14 是等价的.

定义 14. 给定图类 C, 若存在可计算函数 $g: \mathbb{N} \to \mathbb{N}$ 使得对任意 $r \in \mathbb{N}, \mathfrak{G} \in C, v \in V(\mathfrak{G})$ 均存在图 $\mathfrak{H}$ 满足 $\|\mathfrak{H}\| \leqslant g(r)$ 且不为 $\mathfrak{N}_r^{\mathfrak{G}}(v)$ 的子式, 则称 C 为局部不含某一子式的图类.

最后我们对定理 13 中方法作进一步的抽象.

定义 15. 给定图类 C 和函数 $h: Graph \to \mathbb{N}$: (1) 若存在 $m \in \mathbb{N}$, 使得对任意 $\mathfrak{G} \in C$ 均有 $h(\mathfrak{G}) \leqslant m$, 则称 C 为 h 有界图类; (2) 若存在可计算函数 $g: \mathbb{N} \to \mathbb{N}$, 使得对任意 $r \in \mathbb{N}, \mathfrak{G} \in C, v \in V(\mathfrak{G})$ 均有 $h(\mathfrak{N}_r^{\mathfrak{G}}(v)) \leqslant g(r)$, 则称 C 为局部 h 有界图类.

h 有界和局部 h 有界图类通常没有包含关系, 但若 h 为导出子图单调的 (induced-subgraph-monotone), 即对任意 $\mathfrak{G}, \mathfrak{H} \in Graph$ 满足 $\mathfrak{H}$ 是 $\mathfrak{G}$ 的子式, 均有 $h(\mathfrak{H}) \leqslant h(\mathfrak{G})$, 则 h 有界图类也是局部 h 有界图类, 反方向不一定成立. 前文中提到的许多图类都满足定义 15, 例如 (局部) 树宽有界图类对应函数 tw, 度有界图类对应函数 deg, 亏格有界图类对应函数 gen, 显然 tw, deg, gen 均为导出子图单调的. (局部) 不含某一子式的图类也可由定义 15 刻画, 定义函数 $mec: Graph \to \mathbb{N}$ 为 $mec(\mathfrak{G}) = \min\{\ell | \text{团} \mathfrak{K}_\ell \text{不是} \mathfrak{G} \text{的子式}\}$. 显然, 若图 $\mathfrak{H}$ 不为 $\mathfrak{G}$ 的子式, 则团 $\mathfrak{K}_{|V(\mathfrak{H})|}$ 不为 $\mathfrak{G}$ 的子式, 故可等价定义 $mec(\mathfrak{G}) = \min\{|V(\mathfrak{H})| | \text{图} \mathfrak{H} \text{不是} \mathfrak{G} \text{的子式}\}$. 于是由定义 15 可得 (局部) 不含某一子式的图类即 (局部) mec 有界图类, 显然 mec 是导出子图单调的.

一般地, 我们有定理 17.

定理 17[47,57]. 令 U 为一元关系, 给定函数 $h: Graph \to \mathbb{N}$. 定义问题 $MC^{\mathrm{loc}}(\mathrm{FO}, h)$: 输入图 $\mathfrak{G}$, $W \subseteq V(\mathfrak{G})$, 语句 $\varphi \in \mathrm{FO}[\tau_E \cup \{U\}]$, 参数为 $|\varphi| + h(\mathfrak{G})$, 判定是否有 $\mathfrak{G}\langle U \mapsto W\rangle \vDash \varphi$. 若 $MC^{\mathrm{loc}}(\mathrm{FO}, h) \in \mathrm{FPT}$, 则对任意局部 h 有界图类 C 均有 $MC(\mathrm{FO}, C) \in \mathrm{FPT}$. 具体地, 若对于某实数 $c \geqslant 1$, 前者存在一个 c 次方时间的 FPT 算法, 则后者存在一个 $c+1$ 次方时间的 FPT 算法.

证明: 首先需要说明的是, 若 $MC^{\mathrm{loc}}(\mathrm{FO}, h) \in \mathrm{FPT}$, 平凡地就有对任意 h 有界图类 D, $MC(\mathrm{FO}, D) \in \mathrm{FPT}$. 回到图类 C, 考虑问题 $MC^{\mathrm{loc}}(\mathrm{FO}, C)$, 输入为图 $\mathfrak{G} \in C$, 语句 $\varphi \in \mathrm{FO}[\tau_E \cup \{U\}], v \in V(\mathfrak{G}), W \subseteq V(\mathfrak{G}), r \in \mathbb{N}$. 显然, $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v)$ 的计算至多需要 $O(\|\mathfrak{G}\|)$ 时间, 顺便可以计算出 $W' = W \cap N_r^{\mathfrak{G}}(v)$, 易见 $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v) = \mathfrak{N}_r^{\mathfrak{G}}(v)\langle U \mapsto W'\rangle$. 根据定义 15, 存在与 $\mathfrak{G}$ 无关的可计算函数 $g: \mathbb{N} \to \mathbb{N}$ 使得 $h(\mathfrak{N}_r^{\mathfrak{G}}(v)) \leqslant g(r)$. 根据题设, 存在一个算法在 $f(|\varphi| + g(r)) \cdot \|\mathfrak{G}\|^c$ 时间内判定是否有 $\mathfrak{N}_r^{\mathfrak{G}}(v)\langle U \mapsto W'\rangle \vDash \varphi$, 故存在算法在 $f(|\varphi| + g(r)) \cdot \|\mathfrak{G}\|^c + O(\|\mathfrak{G}\|)$ 时间内判定是否有 $\mathfrak{N}_r^{\mathfrak{G}\langle U\mapsto W\rangle}(v) \vDash \varphi$. 当 $c \geqslant 1$ 时, 上述算法是一个 c 次方的 FPT 算法, 由定理 13 即得结论. 证毕.

定理 17 是对上述基于 FO 局部性的方法的进一步抽象: 定理 15 实际上就是将上述定理中的 h 取为 tw, 定理 16 也可以通过 h 取为 mec 来证明. 需要注意的是, 将 h 取为 deg 无法得到定理 14, 只能得到一个平方的 FPT 算法.

4 FO 易解范围边界的研究

随着算法元定理研究的不断发展, 我们对于稀疏性的认识逐渐深刻. 最初, 我们经验性地将稀疏性定义为图的平均度, 将稀疏图类定义为平均度不超过某个常数的图类. 许多简单图类满足这一定义, 如树宽有界图类、度有界图类、平面图类等, 而稍微复杂些的图类则往往并不能满足该定义, 如局部树宽有界图类、不含某一子式的图类等, 但若考察它们的各种性质, 将其也归为稀疏图类更为合理. 另一方面, 这一定义并不与图的结构产生直接关联, 难以得到良好的性质和结论, 后续的发展也是极有限的. 一个尤其明显的问题是, 在稠密图中增加大量顶点可以降低平均度, 但显然所得的图在局部上仍是稠密的, 通常来说它与原图并没有本质的区别.

进一步, 我们开始寻求与图的结构性质关联的稀疏性刻画. 在这一阶段中, 相关研究引入了逻辑和结构图论的

工具, 例如一些基于 FO 局部性的方法, 一系列的算法元定理应运而生. 学界对于稀疏性的认识也进一步深化, 我们逐渐意识到相较于平均度, 局部稀疏性是对稀疏性更本质的刻画. 局部稀疏性是一个相对化的概念: 若一个图类为稀疏图类, 则其对应的局部化图类也为稀疏图类. 直觉上, 若一个图是稀疏的, 它的任意局部也应当是稀疏的, 这样也避免了通过增加顶点能降低平均度的问题. 基于此, 我们可以研究已知图类对应的局部化图类, 将已有的结论迁移到相应的局部化图类上. 通常来说, 考虑局部化图类确实能够实质性地扩大稀疏图类的范围, 因此稀疏图类的边界将会随着新的非局部稀疏图类的出现而拓展. 然而, 这样一种相对化的概念仍未揭示稀疏性的本质, 我们仍不清楚稀疏和稠密图类的分界线应当如何划定, 这也是后续研究的主要课题.

随着大量 FO 模型检测相关的算法元定理的出现, 研究者们意识到 FO 模型检测问题的易解性与图的稀疏性有着深刻的联系. 尽管尚未明确稀疏性的概念, 但前文提到的所有图类都具有某种稀疏的特征, 而这些图类上的 FO 模, 型检测问题都是易解的, 而一些有关复杂性下界的研究表明在一些相对稠密的图类上 FO 模型检测问题不太可能是易解的[57,58]. 由此产生的新“哲学”是“在一定条件下以 FO 模型检测问题的易解性作为稀疏性的判据”, 这引发了对 FO 易解范围边界的深入研究. 在新哲学的指导下, 研究者们开始关注 FO 所具有的能力, 并有针对性地构造出了一些重要的图类, 例如有界扩张图类、无处稠密图类等, 结合结构图论中的另一些证据和线索, 最终划定了稀疏和稠密性的界线. 本节将循着这条研究路径深入介绍稀疏图类上 FO 易解范围边界的探索历程, 这些新的研究使用了一些全新的技术, 其中最典型的是基于量词消去 (quantifier elimination) 的方法, 下面将对其进行详细介绍.

首先引入一些新的概念. 我们允许符号集 τ 中可以包含有限个函数 (function) 符号 (简称函数), 为此需要修改项的定义: 一阶变元和 τ 中的常量是项; 若 $t_1,\dots,t_{ar(f)}$ 是项, 则 $ft_1\dots t_{ar(f)}$ 也是项, 其中 f 是 τ 中的任意函数, $ar(f)$ 为 f 的元数. τ 公式的定义不变, 只是项的类型变得更加丰富, 从而公式的结构变得更加复杂. 相应地, τ 结构是一个元组 $\mathfrak{A}=\left(V(\mathfrak{A}),(R^{\mathfrak{A}})_{R\in\tau},(f^{\mathfrak{A}})_{f\in\tau},(c^{\mathfrak{A}})_{c\in\tau}\right)$, 增加了对函数 f 的解释 $f^{\mathfrak{A}}\colon V(\mathfrak{A})^{ar(f)}\to V(\mathfrak{A})$. 公式的语义依然遵循常规的定义[22]. 方便起见, 额外引入记号 $\mathbf{T}:=\forall x x=x$ 和 $\mathbf{F}:=\neg\mathbf{T}$, $\mathbf{T}$ 和 $\mathbf{F}$ 分别为恒真和恒假的语句, 均仅含有一个量词, 但本节中默认 $qr(\mathbf{T})=qr(\mathbf{F})=0$. 换言之，即允许量词阶数为 0 的公式含有子句 $\mathbf{T}$ 和 $\mathbf{F}$. Gaifman 图的边集增加 $\{(c,d)\mid f\in\tau,f^{\mathfrak{A}}(\bar{a})=b,c,d\in\bar{a}\cup\{b\},c\neq d\}$.

量词消去是 FO 模型检测的另一种思路, 其核心是将公式转化为与之等价的量词阶数不超过某个常数 (甚至无量词) 的公式. 无论是否引入函数, 只要输入公式的量词阶数不超过某个常数, 总能在多项式时间内通过穷举判定 FO 模型检测问题. 量词消去不是无条件的, 通常需要限定在特定结构上. 形式化描述如定义 16.

定义 16. 给定 τ 结构类 C, 若对于任意公式 $\varphi(\bar{x})\in\mathrm{FO}[\tau]$ 均存在量词阶数不超过某一常数 q 的公式 $\varphi^*(\bar{x})\in\mathrm{FO}[\tau]$, 满足对任意 $\mathfrak{A}\in C,\bar{a}\subseteq V(\mathfrak{A})$, $\mathfrak{A}\vDash\varphi(\bar{a})$ 当且仅当 $\mathfrak{A}\vDash\varphi^*(\bar{a})$, 则称在 C 上可以进行量词消去. 进一步, 若 $q=0$, 则称在 C 上可以进行完全量词消去.

量词消去是经典模型论的重要研究课题, 一些典型的可以进行量词消去的类如下: 无端点稠密线性序构成的类、Presburger 算术的模型构成的类、代数闭域构成的类等[24], 大多是一些无限结构的类, 具有一些良好的结构性质. 在有限结构的类上通常很难进行量词消去, 但如果允许引入一些额外的信息 (比如一些标号或者代数结构), 也能得到类似的广义量词消去 (generalized quantifier elimination) 过程.

定义 17. 给定 τ 结构类 C, 若存在一个算法, 输入任意公式 $\varphi(\bar{x})\in\mathrm{FO}[\tau]$ 和 $\mathfrak{A}\in C$, 可以在 $f(|\varphi|)\cdot\|\mathfrak{A}\|^c$ 时间内计算出量词阶数不超过某一常数 q 的公式 $\varphi^*(\bar{x})\in\mathrm{FO}[\tau']$ 和论域与 $\mathfrak{A}$ 相同的 τ' 结构 $\mathfrak{A}^*$ (τ' 可以不同于 τ), 满足对任意 $\bar{a}\in V(\mathfrak{A})$, $\mathfrak{A}\vDash\varphi(\bar{a})$ 当且仅当 $\mathfrak{A}^*\vDash\varphi^*(\bar{a})$, 则称在 C 上可以高效地进行广义量词消去. 进一步, 若 $q=0$, 则称在 C 上可以高效地进行广义完全量词消去.

定义 17 中考虑高效性是基于算法元定理研究的需求. 由定义 17 立刻可得定理 18.

定理 18. 若在结构类 C 上可以高效地进行广义量词消去, 则 $MC(\mathrm{FO},C)\in\mathrm{FPT}$.

证明: 考虑问题 $MC(\mathrm{FO},C)$, 其输入为 $\mathfrak{A}\in C$ 和语句 $\varphi\in\mathrm{FO}[\tau]$. 根据定义 17, 存在一个算法 A, 根据 $\mathfrak{A}$ 和 φ 计算 φ^* 和 $\mathfrak{A}^*$, 所需时间为 $f(|\varphi|)\cdot\|\mathfrak{A}\|^c$. 于是只需判定 $\mathfrak{A}^*\vDash\varphi^*$ 是否成立. 由于 $\mathfrak{A}^*$ 和 $\mathfrak{A}$ 论域相同, 且 $qr(\varphi^*)\leqslant q$, 故穷举

判定即可. 由于 φ^* 的长度和 $\mathfrak{A}^*$ 的规模均不超过 $f(|\varphi|)\cdot\|\mathfrak{A}\|^c$, 故所需时间为 $\mathrm{O}\left((f(|\varphi|)\cdot\|\mathfrak{A}\|^c)^{q+2}\right)$. 证毕.

通过将公式 $\forall x\varphi$ 替换为等价的 $\neg\exists x\neg\varphi$ 可以去掉公式中的全称量词. 如果对任意无量词的公式 ψ, $\exists x\psi$ 都在 C 上等价于某个无量词的 ψ^*, 则可以一层一层地消去公式中的存在量词. 于是有引理 4.

引理 4. 给定 τ 结构类 C, 若存在一个算法, 输入任意无量词公式 $\psi(x,y)\in\mathrm{FO}[\tau]$ 和 $\mathfrak{A}\in C$, 可以在 $f(|\psi|)\cdot\|\mathfrak{A}\|^c$ 时间内计算出无量词的公式 $\psi^*(x)\in\mathrm{FO}[\tau']$ 和论域与 $\mathfrak{A}$ 相同的 τ' 结构 $\mathfrak{A}^*$ (τ' 可以不同于 τ), 满足 $|\psi^*|=g(|\psi|)$, g 为可计算函数, $\|\mathfrak{A}^*\|=\mathrm{O}(\|\mathfrak{A}\|)$, 而且对任意 $\bar{a}\in V(\mathfrak{A})$, $\mathfrak{A}\vDash\exists y\psi(\bar{a},y)$ 当且仅当 $\mathfrak{A}^*\vDash\psi^*(\bar{a})$, 则在 C 上可以高效地进行广义量词消去.

证明: 对于任意公式 $\varphi\in\mathrm{FO}[\tau]$, 首先将其转化为前束范式 (prenex normal form) $\varphi_0:=Q_1x_1\ldots Q_kx_k\psi_0$, 其中 $Q_i\in\{\exists,\forall\}$. 若 Q_k 为存在量词, 则直接消去得到 $\varphi_1:=Q_1x_1\ldots Q_{k-1}x_{k-1}\psi_1$, 其中 $\psi_1:=\psi_0^*$; 若 Q_k 为全称量词, 则替换为等价公式 $Q_1x_1\ldots\neg\exists x_k\neg\psi_0$ 然后消去, 得到 $\varphi_1:=Q_1x_1\ldots Q_{k-1}x_{k-1}\neg\psi_1$, 其中 $\psi_1:=(\neg\psi_0)^*$. 同时对输入的 $\mathfrak{A}\in C$, 计算相应的 $\mathfrak{A}_1:=\mathfrak{A}^*$, 显然满足 $\mathfrak{A}\vDash\varphi$ 当且仅当 $\mathfrak{A}_1\vDash\varphi_1$. 以此类推, 逐层消去量词, 最终得到无量词的公式 φ_k 和结构 $\mathfrak{A}_k$, 满足 $\mathfrak{A}\vDash\varphi$ 当且仅当 $\mathfrak{A}_k\vDash\varphi_k$. 由于消去 Q_i 的时间为 $\mathrm{O}\left(f\left(\left|\psi_{i-1}\right|\right)\cdot\|\mathfrak{A}_i\|^c\right)$, $|\psi_i|$ 是 $\left|\psi_{i-1}\right|$ 的可计算函数, $\|\mathfrak{A}_i\|=\mathrm{O}(\|\mathfrak{A}_{i-1}\|)$, 故总时间为 $f'(|\psi|)\cdot\|\mathfrak{A}\|^c$, 其中 f' 为可计算函数. 证毕.

应用量词消去方法可以得到一些算法元定理, 最简单的例子是函数有根树上的 FO 模型检测问题. 函数有根树是指将有根树中表示父子节点的关系 P 替换为一个函数 p 并且引入若干标号后所得的结构, 函数有根树上的相关概念和有根树保持一致. 记全体高度不超过 d 的函数有根树构成的类为 FTree_d.

定理 19[5]. 在 FTree_d 上可以高效地进行广义完全量词消去, 故 $\mathit{MC}(\mathrm{FO},\mathit{FTree}_d)\in\mathrm{FPT}$ (定理 18).

证明: 限于篇幅仅给出证明思路. 根据引理 4, 仅考虑形如 $\exists y\psi(\bar{x},y)$ 的公式, 其中 ψ 为原子公式. 原子公式 $\psi(\bar{x},y)$ 能描述的内容即 $\bar{x},y$ 与其所有祖先构成的子树的结构, 子树的高度至多为 d. 构造等价公式 ψ^* 时, 用原子公式的析取穷举该子树的所有可能情况. 在每种情况中, 与 y 相关的信息均为一条路径 P, P 于 $\bar{x}$ 的某个祖先 a 处与子树的其他部分汇合. 只需要用一个额外的标号标记函数有根树中从 a 出发且结构与 P 完全相同的路径的数量, 在 ψ^* 中就可以通过计算结构与 P 完全相同的路径的条数来表示 y 的存在性. 证毕.

基于定理 19, 可以在更复杂的图类上进行量词消去. 树深 (tree-depth) 是一个重要的图参数, 它描述了一个图在多大程度上接近于一颗星 (即高度为 2 的树).

定义 18. 图 $\mathfrak{G}$ 的树深 $td(\mathfrak{G})$ 归纳定义如下: (1) 若 $|V(\mathfrak{G})|=1$, 则 $td(\mathfrak{G}):=1$; (2) 若 $\mathfrak{G}$ 连通且 $|V(\mathfrak{G})|>1$, 则 $td(\mathfrak{G}):=1+\min_{v\in V(\mathfrak{G})}td(\mathfrak{G}[V(\mathfrak{G})-\{v\}])$; (3) 其余情况下 $td(\mathfrak{G}):=\max\left\{td(\mathfrak{H})\mid\mathfrak{H}\text{是}\mathfrak{G}\text{的分量}\right\}$. 给定图类 C, 若存在 $m\in\mathbb{N}^+$ 使得任意 $\mathfrak{G}\in C$ 满足 $td(\mathfrak{G})\leqslant m$, 则称 C 为树深有界图类.

例如, 无边图的树深为 1, 星的树深为 2, 路 $\mathfrak{P}_\ell$ 的树深为 $\lceil\log(\ell+1)\rceil$, 团 $\mathfrak{K}_\ell$ 的树深为 ℓ. 直观来看, 定义 18 所描述的过程即在图中不断地选择一个顶点删去, 如果图被拆散成了多个分量, 则在每个分量中继续重复此删点过程, 直至所有顶点均被删除. 按照顶点的删除顺序可以构造一棵消除树 (elimination tree), 高度最小的消除树的高度即树深. 根据图 $\mathfrak{G}$ 的任意消除树可以构造宽度恰为消除树高度减 1 的树分解[65], 故有 $tw(\mathfrak{G})\leqslant td(\mathfrak{G})-1$, 从而任意树深有界图类 C 为树宽有界图类, 根据推论 1 立刻有 $\mathit{MC}(\mathrm{MSO},C)\in\mathrm{FPL}$. 通过引入一些代数结构和常量, 树深有界图类上可以高效地进行广义量词消去[66], 由此可得更优的结论, 即定理 20.

定理 20[66]. 对任意树深有界图类 C, $\mathit{MC}(\mathrm{MSO},C)\in\text{para-}\mathrm{AC}_0$.

para-AC_0 是电路复杂性类 AC_0 的参数化版本, para-$\mathrm{AC}_0\subseteq\mathrm{FPT}$. 直觉上, 消除树的构造是一种并行的过程, 因此树深相关的结论都隐含某种并行性, 而并行性是电路的重要特性.

函数图 (functional graph) 是指把图中的边替换成若干一元函数并且引入一些标号后得到的结构. 给定函数图类 C, 若对应的 Gaifman 图类 $D:=\{G(\mathfrak{A})\mid\mathfrak{A}\in C\}$ 为树深有界图类, 则称 C 树深有界. 由定理 19 可得定理 21.

定理 21. 在树深有界的函数图类 C 上可以高效地进行广义完全量词消去, 故 $\mathit{MC}(\mathrm{FO},C)\in\mathrm{FPT}$ (定理 18).

证明: 大致证明思路如下. 对任意 $\mathfrak{G}\in C$, 若其树深为 d, 则 $\mathfrak{G}$ 中不含有长度超过 2^d 的路径[67]. 故首先引入新的函数, 构造 $\mathfrak{G}$ 的一棵消除树, 其高度至多为 2^d. 其次, 引入若干新的标号来表示 $\mathfrak{G}$ 中的边在消除树中的位置和方

向. 于是就将 $\mathfrak{G}$ 上的模型检测问题归约到了消除树上的模型检测问题, 根据定理 19 即得证. 证毕.

量词消去技术的一个重要成果是有界扩张图类上 FO 模型检测的易解性. 有界扩张图类最早由 Nešetřil 等人[68]提出, 是结构图论的重要研究对象. 回顾子式的定义, 在图 $\mathfrak{G}$ 上进行一系列顶点删除、边删除和边收缩操作后得到的图称为 $\mathfrak{G}$ 的子式, 事实上有如下等价定义: 取 $\mathfrak{G}$ 的一个子图 $\mathfrak{H}$, 选取 $\mathfrak{H}$ 的若干互不相交的连通子图 $\mathfrak{H}_1,\ldots,\mathfrak{H}_m$, 将每个 $\mathfrak{H}_i$ 缩为一个顶点后得到的图 $\mathfrak{H}'$ 称为 $\mathfrak{G}$ 的子式. 若限制每个 $\mathfrak{H}_i$ 的直径不超过 $2r$ (即 $\mathfrak{H}_i$ 是 $\mathfrak{G}$ 的某个 r 邻域的子图), 则称 $\mathfrak{H}'$ 为 $\mathfrak{G}$ 的 r 浅子式 (r -shallow minor).

定义 19. 定义 $\mathfrak{G}$ 的 r 梯度 (grad) $\nabla_r(\mathfrak{G})$ 为 r 浅子式的最大密度, 即 $\max\left\{|V(\mathfrak{H})|/|E(\mathfrak{H})|\,\middle|\,\mathfrak{H}是\mathfrak{G}的r浅子式\right\}$. 给定图类 C, 若存在函数 $g:\mathbb{N}\to\mathbb{N}$ 使得对任意 $\mathfrak{G}\in C$ 和 $r\in\mathbb{N}$ 均有 $\nabla_r(\mathfrak{G})\leqslant g(r)$, 则称 C 为有界扩张图类.

可以证明, 真子式理想均为有界扩张图类[69]; 另有一些例子表明, 局部树深有界图类不一定为有界扩张图类[47], 有界扩张图类也不一定为局部不含某一子式的图类. 直观上, 有界扩张图类中的图在经过局部的收缩操作之后仍是稀疏的, 这是从另一个角度对稀疏性的刻画, 不同于局部稀疏性. 局部的收缩操作可以由 FO 描述, 因此直觉上有界扩张图类上的 FO 模型检测问题很有可能是易解的. 事实上, 有界扩张图类具有如引理 5 所述性质.

引理 5[68]. 给定有界扩张图类 C, 存在函数 $h:\mathbb{N}\to\mathbb{N}$ 使得对任意 $\mathfrak{G}\in C$ 和 $p\in\mathbb{N}$, 均存在 $\mathfrak{G}$ 的一个 $f(p)$ 种颜色的着色, 使得其中任意 p 种颜色的顶点构成的导出子图的树深不超过 p.

根据引理 5 和定理 21 可得定理 22.

定理 22[5]. 在有界扩张图类 C 上可以高效地进行广义完全量词消去, 故 $MC(\mathrm{FO},C)\in\mathrm{FPT}$ (定理 18).

另有工作给出了定理 22 的另一种基于型的证明[58]. 可以利用局部化方法将上述结论进一步推广到局部有界扩张 (locally bounded expansion) 图类 (即有界扩张图类对应的局部化图类) 上[5]. 平凡地, 有界扩张图类是局部有界扩张图类; 容易证明, 局部不含某一子式的图类为局部有界扩张图类.

由于 FO 能够描述局部的收缩操作, 如果浅子式上的 FO 模型检测是困难的, 则在原图上也是困难的. 研究者们通过精确地刻画 FO 的这种表达能力, 得到了一类至关重要的图类——无处稠密图类.

定义 20[70]. 对任意图 $\mathfrak{G}$, 定义 $\omega_r(\mathfrak{G})=\max\left\{\ell\,\middle|\,团\mathfrak{K}_\ell是\mathfrak{G}的r浅子式\right\}$. 给定图类 C, 若存在函数 $g:\mathbb{N}\to\mathbb{N}$ 使得对任意 $\mathfrak{G}\in C$ 和 $r\in\mathbb{N}$ 均有 $\omega_r(\mathfrak{G})\leqslant g(r)$, 则称 C 为无处稠密图类, 否则称为某处稠密图类.

根据定义, 对于无处稠密图类 C 中任意图 $\mathfrak{G}$, 团 $\mathfrak{K}_{g(r)}$ 均不为 $\mathfrak{G}$ 的 r 浅子式, 这比全局不包含某一子式条件更弱, 故真子式理想为无处稠密图类; 有界扩张图类均为无处稠密图类, 而且经验性地, 有界扩张图类的各种性质会以一种相似的但更复杂的形式在无处稠密图类中呈现[71]. 相应地可以定义局部无处稠密 (locally nowhere dense) 图类, 局部有界扩张图类均为局部无处稠密图类. 值得注意的是, 对无处稠密图类进行局部化操作并不能得到新的图类, 局部无处稠密图类和无处稠密图类是等价的[71], 这说明无处稠密图类的定义具有相当的稳健性. 借助稀疏邻域覆盖 (sparse neighborhood cover) 和一种恰好刻画了无处稠密图类的博弈, 可以证明无处稠密图类上的 FO 模型检测问题是 FPT 的[6], 这拓展了 FO 模型检测的易解性边界. 更为关键的是反方向的结论, 即定理 23.

定理 23[6]. 假设 $\mathrm{FPT}\neq\mathrm{AW}[*]$, 对任意子图理想 C, 若 $MC(\mathrm{FO},C)\in\mathrm{FPT}$, 则 C 是无处稠密图类.

根据定义易得, 无处稠密图类的子图闭包仍为无处稠密图类. 因此定理 23 表明, 在 $\mathrm{FPT}\neq\mathrm{AW}[*]$ 这一常见的复杂性假设下, 对于子图理想来说, 无处稠密图类就是 FO 模型检测的易解性边界, 而直觉上稀疏图类的子图闭包也确实应当是稀疏的, 故学界倾向于将稀疏图类直接定义为无处稠密图类, 稠密图类则定义为某处稠密图类, 这样一来就划定了稀疏与稠密图类的界线, 一举解决了图的平均度、局部稀疏性等遇到的问题. 至此, 可以说稀疏图类上的 FO 模型检测问题均为 FPT 的, 稀疏图类相关的算法元定理研究已经臻于成熟.

稀疏图类的任何一个子类以及子图闭包均为稀疏的, 但稠密图类并不具有这样的性质, 例如 *Clique* 显然是某处稠密图类, 但并非子图理想, 然而 *Clique* 上的 FO 模型检测问题显然是易解的, 因为完全图和无边图并无本质区别. 可见在稠密图类上 FO 模型检测问题仍有可能易解, FO 的易解边界仍可能进一步拓展, 稠密图类上的相关研究是近十年来算法元定理的主要研究方向. 在一些特殊的稠密图类上, 甚至 MSO 模型检测问题仍是易解的, 例如团宽有界图类[11]等, 团宽是树宽的概念在稠密图类中的推广, 类似还有丛深 (shrub-depth) 对应树深的概念. 著名

的 Seese 猜想[13]断言 MSO 的表达能力与团宽有界图类在某种意义上等价, 但尚未得到证明. FO 的情况则更加复杂, 研究发现 FO 模型检测问题在诸多稠密图类上仍是易解的, 这些图类大多具有相当复杂的结构.

一类重要的稠密图类是由稀疏图类通过转导得到的结构性图类. 简单来说, 转导就是将一个图类 C (通常是稀疏图类) 通过一些操作 (主要是使用逻辑公式进行翻译 (interpretation)) 转化为另一个图类 D (通常是稠密图类), 若 C 具有性质 P, 则称 D 具有结构性性质 P. 如果能够从 D 高效地恢复出 C 中的信息, 则能将 D 上的模型检测问题翻译回 C 上的模型检测问题, 从而应用 C 上的模型检测算法. 但一般来说, 恢复出 C 中的信息并不容易, 因为具体的转导操作信息并不会直接保留到 D 中. 如果只知道 D 是由 C 转导得到的, 则需要从 D 的结构反推出转导的过程, 甚至并不一定能精确地得到 C. 值得一提的是, 转导操作是自封闭的, 即对 C 做两次转导可以复合为一次转导, 因此具有结构性性质 P 的图类经过转导得到的图类仍具有结构性性质 P, 这体现了结构性性质的稳健性. 近年来, 结构性性质的相关研究进展迅速, 相关工作证明了在结构性度有界图类[15]、结构性局部团宽有界图类[17]、结构性无处稠密图类[18]上的 FO 模型检测问题均是 FPT 的. 其中, 结构性无处稠密图类上的结论具有里程碑意义, 它说明即使是最稠密的稀疏图类 (即无处稠密图类) 转导后得到的结构性图类上的 FO 模型检测问题仍是易解的, 这极大地拓宽了 FO 的易解范围. 该结论的证明方法[18]继承自无处稠密图类上对应结论的证明[6]: 借助稀疏弱邻域覆盖 (sparse weak neighborhood cover) 以及一种刻画结构性无处稠密图类的博弈, 依照博弈的过程递归地缩小待检测模型的规模, 从而有效减少模型检测的时间.

目前的研究仍在进一步探索 FO 的易解范围. 除了结构性性质之外, 图类还有一些深刻的性质, 如弱稀疏性、稳定性、一元依赖性等[21]. 如果一个图类中所有的图都不包含某个特定的二分团作为子图, 则称该图类是弱稀疏的. 弱稀疏的概念从另一个角度建立了稀疏和稠密图类上重要参数之间的联系, 例如团宽有界图类中所有弱稀疏的图类恰为所有的树宽有界图类, 丛深有界图类中所有弱稀疏的图类恰为所有的树深有界图类等. 稳定性则是源于模型论中的稳定性理论 (stability theory), 与结构性、弱稀疏性密切相关, 许多稀疏图类对应的具有稳定性质 (stable property) 的图类与具有结构性性质的图类恰好是等价的[21]. 一个图类具有一元依赖性是指它不能与 $Graph$ 互相通过转导得到. 值得一提的是, 具有一元依赖性的图类中弱稀疏的图类恰为所有的无处稠密图类[72,73]. 这些基于弱稀疏性的对应关系很自然地导出了一些新的问题, 例如哪个稠密图类中弱稀疏的图类恰为所有的有界扩张图类? 目前这仍是一个未解问题. 由于前文所述的各个 FO 模型检测易解的图类均具有一元依赖性, 故学界关于 FO 的易解范围有如下猜想, 有待进一步的研究.

猜想: 对于任意图类 C, $MC(\mathrm{FO},C)\in\mathrm{FPT}$ 当且仅当 C 具有一元依赖性.

5 总 结

本文回顾了算法元定理领域近几十年的研究, 对一些关键技术进行了系统性的介绍, 包括自动机方法、基于 FO 局部性的方法、基于量词消去的方法等, 并修正了前人的少量疏漏. FO 的易解范围与图的稀疏性有着密切的关联, 随着 FO 的易解范围不断拓展, 我们对于稀疏性的认识也臻于成熟. 稀疏图类上的算法元定理研究表明 FO 模型检测在所有的稀疏图类上都是易解的, 但 FO 易解范围的相关研究并未止步于此, 近年来的一些研究发现 FO 模型检测在诸多良构稠密图类上仍具有易解性, 由此引发了算法元定理的新一轮研究, 尤其是对 FO 易解范围的进一步探索. 一些新近的研究让我们得以窥见稠密图类宇宙的一隅, 同时也带来了更多的未解问题. 目前学界对稠密图类的认识尚少, 这些研究将极大地加深我们对稠密图类的理解, 对结构图论和图算法领域有着重要的意义.

References:

[1] Courcelle B. Graph rewriting: An algebraic and logic approach. In: van Leeuwen J, ed. Formal Models and Semantics. Amsterdam: Elsevier, 1990. 193, 195–242. [doi: 10.1016/B978-0-444-88074-1.50010-X]

[2] Seese D. Linear time computable problems and first-order descriptions. Mathematical Structures in Computer Science, 1996, 6(6): 505–526. [doi: 10.1017/S0960129500070079]

[3] Flum J, Grohe M. Fixed-parameter tractability, definability, and model-checking. SIAM Journal on Computing, 2001, 31(1): 113–145. [doi: 10.1137/S0097539799360768]

[4] Frick M, Grohe M. Deciding first-order properties of locally tree-decomposable structures. Journal of the ACM, 2001, 48(6): 1184–1206. [doi: 10.1145/504794.504798]

[5] Dvořák Z, Král D, Thomas R. Deciding first-order properties for sparse graphs. In: Proc. of the 51st IEEE Annual Symp. on Foundations of Computer Science. Las Vegas: IEEE, 2010. 133–142. [doi: 10.1109/FOCS.2010.20]

[6] Grohe M, Kreutzer S, Siebertz S. Deciding first-order properties of nowhere dense graphs. Journal of the ACM, 2017, 64(3): 17. [doi: 10.1145/3051095]

[7] Gajarský J, Hlinený P, Lokshtanov D, Obdrálek J, Ordyniak S, Ramanujan MS, Saurabh S. FO model checking on posets of bounded width. In: Proc. of the 56th IEEE Annual Symp. on Foundations of Computer Science. Berkeley: IEEE, 2015. 963–974. [doi: 10.1109/FOCS.2015.63]

[8] Ganian R, Hliněný P, Král D, Obdržálek J, Schwartz J, Teska J. FO model checking of interval graphs. In: Proc. of the 40th Int'l Colloquium on Automata, Languages, and Programming. Riga: Springer, 2013. 250–262. [doi: 10.1007/978-3-642-39212-2_24]

[9] Eickmeyer K, Kawarabayashi KI. FO model checking on map graphs. In: Proc. of the 21st Int'l Symp. on Fundamentals of Computation Theory. Bordeaux: Springer, 2017. 204–216. [doi: 10.1007/978-3-662-55751-8_17]

[10] Hliněný P, Pokrývka F, Roy B. FO model checking on geometric graphs. Computational Geometry, 2019, 78: 1–19. [doi: 10.1016/j.comgeo.2018.10.001]

[11] Courcelle B, Makowsky JA, Rotics U. Linear time solvable optimization problems on graphs of bounded clique-width. Theory of Computing Systems, 2000, 33(2): 125–150. [doi: 10.1007/s002249910009]

[12] Garey MR, Johnson DS, Stockmeyer L. Some simplified NP-complete problems. In: Proc. of the 6th Annual ACM Symp. on Theory of Computing. Seattle: ACM, 1974. 47–63. [doi: 10.1145/800119.803884]

[13] Seese D. The structure of the models of decidable monadic theories of graphs. Annals of Pure and Applied Logic, 1991, 53(2): 169–195. [doi: 10.1016/0168-0072(91)90054-P]

[14] Oum SI. Approximating rank-width and clique-width quickly. ACM Trans. on Algorithms, 2008, 5(1): 10. [doi: 10.1145/1435375.1435385]

[15] Gajarský J, Hliněný P, Obdržálek J, Lokshtanov D, Ramanujan MS. A new perspective on FO model checking of dense graph classes. In: Proc. of the 31st Annual ACM/IEEE Symp. on Logic in Computer Science (LICS). New York: IEEE, 2016. 1–9.

[16] Gajarský J, Král D. Recovering sparse graphs. In: Proc. of the 43rd Int'l Symp. on Mathematical Foundations of Computer Science. Liverpool: MFCS, 2018. 29: 1–29: 15.

[17] Bonnet É, Dreier J, Gajarský J, Kreutzer S, Mählmann N, Simon P, Toruńczyk S. Model checking on interpretations of classes of bounded local cliquewidth. In: Proc. of the 37th Annual ACM/IEEE Symp. on Logic in Computer Science. Haifa: ACM, 2022. 54. [doi: 10.1145/3531130.3533367]

[18] Dreier J, Mählmann N, Siebertz S. First-order model checking on structurally sparse graph classes. In: Proc. of the 55th Annual ACM Symp. on Theory of Computing. Orlando: ACM, 2023. 567–580. [doi: 10.1145/3564246.3585186]

[19] Dreier J. Lacon-, shrub- and parity-decompositions: Characterizing transductions of bounded expansion classes. Logical Methods in Computer Science, 2023, 19(2): 14:1–14:25. [doi: 10.46298/lmcs-19(2:14)2023]

[20] Dreier J, Gajarský J, Kiefer S, Pilipczuk M, Toruńczyk S. Treelike decompositions for transductions of sparse graphs. In: Proc. of the 37th Annual ACM/IEEE Symp. on Logic in Computer Science. Haifa: ACM, 2022. 31. [doi: 10.1145/3531130.3533349]

[21] Gajarský J, Pilipczuk M, Toruńczyk S. Stable graphs of bounded twin-width. In: Proc. of the 37th Annual ACM/IEEE Symp. on Logic in Computer Science. Haifa: ACM, 2022. 39. [doi: 10.1145/3531130.3533356]

[22] Ebbinghaus HD, Flum J, Thomas W. Mathematical Logic. 2nd ed., New York: Springer, 1994. 1910.

[23] Chang CC, Keisler HJ. Model Theory. Amsterdam: Elsevier, 1990.

[24] Marker D. Model Theory: An Introduction. Springer Science & Business Media, 2006.

[25] Hodges W. Model Theory. New York: Cambridge University Press, 1993.

[26] Ebbinghaus HD, Flum J. Finite Model Theory. 2nd ed., New York: Springer, 2005.

[27] Libkin L. Elements of Finite Model Theory. Berlin: Springer, 2004. 41. [doi: 10.1007/978-3-662-07003-1]

[28] Gaifman H, Vardi MY. A simple proof that connectivity of finite graphs is not first-order definable. Bulletin of the EATCS, 1985, 26: 43–44.

[29] Papadimitriou CH. Computational Complexity. Reading: Addison-Wesley, 1994.

[30] Goldreich O. Computational Complexity: A Conceptual Perspective. Cambridge: Cambridge University Press, 2008.

[31] Flum J, Grohe M. Parameterized Complexity Theory. Berlin: Springer, 2006. [doi: 10.1007/3-540-29953-X]

[32] Downey RG, Fellows MR. Fundamentals of Parameterized Complexity. London: Springer, 2013. 4. [doi: 10.1007/978-1-4471-5559-1]

[33] Vardi MY. The complexity of relational query languages. In: Proc. of the 14th Annual ACM Symp. on Theory of Computing. San

Francisco: ACM, 1982. 137–146. [doi: 10.1145/800070.802186]

[34] Chandra AK, Merlin PM. Optimal implementation of conjunctive queries in relational data bases. In: Proc. of the 9th Annual ACM Symp. on Theory of Computing. Boulder: ACM, 1977. 77–90. [doi: 10.1145/800105.803397]

[35] Vardi MY. On the complexity of bounded-variable queries. In: Proc. of the 14th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems. San Jose: ACM, 1995. 266–276. [doi: 10.1145/212433.212474]

[36] Fagin R. Generalized first-order spectra and polynomial-time recognizable sets. In: Society for Industrial and Applied Mathematics, ed. Complexity of Computation. Providence: American Mathematical Society, 1974. 27–41.

[37] Immerman N. Descriptive Complexity. Springer Science & Business Media, 2012.

[38] Frick M, Grohe M. The complexity of first-order and monadic second-order logic revisited. Annals of Pure and Applied Logic, 2004, 130(1–3): 3–31. [doi: 10.1016/j.apal.2004.01.007]

[39] Downey RG, Fellows MR, Taylor U. The parameterized complexity of relational database queries and an improved characterization of W[1]. In: Proc. of the 1st Conf. of the Centre for Discrete Mathematics and Theoretical Computer Science. Auckland: DMTCS, 1996. 194–213.

[40] Büchi JR. Weak second-order arithmetic and finite automata. Mathematical Logic Quarterly, 1960, 6(1–6): 66–92. [doi: 10.1002/malq.19600060105]

[41] Elgot CC. Decision problems of finite automata design and related arithmetics. Trans. of the American Mathematical Society, 1961, 98(1): 21–51. [doi: 10.2307/1993511]

[42] Trakhtenbrot BA. Finite automata and the logic of oneplace predicates. Sibirskii Matematicheskii Zhurnal, 1962, 3(1): 103–131.

[43] Kleene SC. Representation of events in nerve nets and finite automata. Automata Studies, 1956, 34: 3–42. [doi: 10.1515/9781400882618-002]

[44] Doner J. Tree acceptors and some of their applications. Journal of Computer and System Sciences, 1970, 4(5): 406–451. [doi: 10.1016/S0022-0000(70)80041-1]

[45] Thatcher JW, Wright JB. Generalized finite automata theory with an application to a decision problem of second-order logic. Mathematical Systems Theory, 1968, 2(1): 57–81. [doi: 10.1007/BF01691346]

[46] Feferman S, Vaught RL. The first order properties of products of algebraic systems. Fundamenta Mathematicae, 1959, 47(1): 57–103. [doi: 10.4064/fm-47-1-57-103]

[47] Grohe M. Logic, graphs, and algorithms. In: Flum J, Grädel E, Wilke T, eds. Logic and Automata-History and Perspectives. Amsterdam: Amsterdam University Press, 2007. 357–422.

[48] Makowsky JA. Algorithmic uses of the feferman-vaught theorem. Annals of Pure and Applied Logic, 2004, 126(1–3): 159–213. [doi: 10.1016/j.apal.2003.11.002]

[49] Arnborg S, Corneil DG, Proskurowski A. Complexity of finding embeddings in a *k*-tree. SIAM Journal on Algebraic Discrete Methods, 1987, 8(2): 277–284. [doi: 10.1137/0608024]

[50] Bodlaender HL. A linear time algorithm for finding tree-decompositions of small treewidth. In: Proc. of the 25th Annual ACM Symp. on Theory of Computing. San Diego: ACM, 1993. 226–234. [doi: 10.1145/167088.167161]

[51] Elberfeld M, Jakoby A, Tantau T. Logspace versions of the theorems of bodlaender and courcelle. In: Proc. of the 51st IEEE Annual Symp. on Foundations of Computer Science. Las Vegas: IEEE, 2010. 143–152. [doi: 10.1109/FOCS.2010.21]

[52] Dawar A, Grohe M, Kreutzer S. Locally excluding a minor. In: Proc. of the 22nd Annual IEEE Symp. on Logic in Computer Science (LICS 2007). Wroclaw: IEEE, 2007. 270–279. [doi: 10.1109/LICS.2007.31]

[53] Diestel R. Graph Theory. 5th ed., Berlin: Springer, 2017. [doi: 10.1007/978-3-662-53622-3]

[54] Wagner K. Über eine Eigenschaft der ebenen Komplexe. Mathematische Annalen, 1937, 114(1): 570–590. [doi: 10.1007/BF01594196]

[55] Gaifman H. On local and non-local properties. Studies in Logic and the Foundations of Mathematics, 1982, 107: 105–135. [doi: 10.1016/S0049-237X(08)71879-2]

[56] Dawar A, Grohe M, Kreutzer S, Schweikardt N. Model theory makes formulas large. In: Proc. of the 34th Int'l Conf. on Automata, Languages and Programming. Wrocław: Springer, 2007. 913–924.

[57] Kreutzer S. Algorithmic meta-theorems. In: Proc. of the 3rd Int'l Workshop on Parameterized and Exact Computation. Victoria: Springer, 2008. 10–12. [doi: 10.1007/978-3-540-79723-4_3]

[58] Grohe M, Kreutzer S. Methods for algorithmic meta theorems. In: Grohe M, Makowsky JA, eds. Model Theoretic Methods in Finite Combinatorics. Providence: American Mathematical Society, 2011. 181–206. [doi: 10.1090/conm/558]

[59] Hanf W. Model-theoretic methods in the study of elementary logic. In: Addison J, Henkin L, Tarski A, eds. The Theory of Models. Amsterdam: North-Holland, 1965.

[60] Fagin R, Stockmeyer LJ, Vardi MY. On monadic NP vs monadic co-NP. Information and Computation, 1995, 120(1): 78–92. [doi: 10.

1006/inco.1995.1100]

[61] Eppstein D. Subgraph isomorphism in planar graphs and related problems. In: Tamassia R, Tollis IG, eds. Graph Algorithms and Applications I. River Edge: World Scientific, 2002. 283–309. [doi: 10.1142/9789812777638_0014]

[62] Robertson N, Seymour PD. Graph minors. III. planar tree-width. Journal of Combinatorial Theory, Series B, 1984, 36(1): 49–64. [doi: 10.1016/0095-8956(84)90013-3]

[63] Eppstein D. Diameter and treewidth in minor-closed graph families. Algorithmica, 2000, 27(3): 275–291. [doi: 10.1007/s004530010020]

[64] Robertson N, Seymour PD. Graph minors. XIII. The disjoint paths problem. Journal of Combinatorial Theory, Series B, 1995, 63(1): 65–110. [doi: 10.1006/jctb.1995.1006]

[65] Bodlaender HL, Gilbert JR, Hafsteinsson H, Kloks T. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. Journal of Algorithms, 1995, 18(2): 238–255. [doi: 10.1006/jagm.1995.1009]

[66] Chen YJ, Flum J. Tree-depth, quantifier elimination, and quantifier rank. In: Proc. of the 33rd Annual ACM/IEEE Symp. on Logic in Computer Science. Oxford: ACM, 2018. 225–234. [doi: 10.1145/3209108.3209160]

[67] Nešetřil J, De Mendez PO. Sparsity: Graphs, Structures, and Algorithms. Berlin: Springer, 2012. 28. [doi: 10.1007/978-3-642-27875-4]

[68] Nešetřil J, De Mendez PO. Grad and classes with bounded expansion I. Decompositions. European Journal of Combinatorics, 2008, 29(3): 760–776. [doi: 10.1016/j.ejc.2006.07.013]

[69] Nešetřil J, De Mendez PO. Grad and classes with bounded expansion II. Algorithmic aspects. European Journal of Combinatorics, 2008, 29(3): 777–791. [doi: 10.1016/j.ejc.2006.07.014]

[70] Nešetřil J, De Mendez PO. First order properties on nowhere dense structures. The Journal of Symbolic Logic, 2010, 75(3): 868–887. [doi: 10.2178/jsl/1278682204]

[71] Nešetřil J, De Mendez PO. On nowhere dense graphs. European Journal of Combinatorics, 2011, 32(4): 600–617. [doi: 10.1016/j.ejc.2011.01.006]

[72] Dvořák Z. Induced subdivisions and bounded expansion. European Journal of Combinatorics, 2018, 69: 143–148. [doi: 10.1016/j.ejc.2017.10.004]

[73] Nešetřil J, De Mendez PO, Rabinovich R, Siebertz S. Classes of graphs with low complexity: The case of classes with bounded linear rankwidth. European Journal of Combinatorics, 2021, 91: 103223. [doi: 10.1016/j.ejc.2020.103223]

刘国航(1999－), 男, 硕士, 主要研究领域为逻辑, 算法图论.

陈翌佳(1972－), 男, 博士, 教授, CCF 专业会员, 主要研究领域为逻辑, 计算复杂性, 算法图论.