arXiv:2603.06345v1 [cs.LO] 6 Mar 2026

通过可满足性求解寻找连接

Clemens Eisenhofer[1](✉), Michael Rawson[2], 与 Laura Kov´acs[1]

[1] 维也纳工业大学，维也纳，奥地利{clemens.eisenhofer,laura.kovacs}@tuwien.ac.at[2] 南安普顿大学，南安普顿，英国 michael@rawsons.uk

摘要。自动推理器常用的证明策略，或通过基于排序的饱和来组织证明搜索，或通过将目标归约为子目标。本文结合这两种方法，倡导一种带对称性破缺的、面向一阶逻辑连接演算的基于 SAT 的方法，旨在进一步推进一阶经典逻辑证明的自动化。与将一阶逻辑归约为命题逻辑的经典做法不同，我们的方法对证明搜索本身的结构进行编码。我们给出三种不同的连接演算 SAT 编码，分析其理论性质，并讨论在这些编码上使用 SAT/SMT 求解器的效果。我们在新求解器 UPCoP 中实现了本工作，并展示了其实际可行性。

1 引言

一阶逻辑自动定理证明器采用的搜索策略可分为两大类 [16]：基于排序的策略和子目标归约策略。第一类包括基于饱和的定理证明器，如 Vampire [35]、E [54] 和 Spass [60]，其工作方式是不断从已有公式集合中推导新事实，并用这些新事实扩展搜索空间。第二类包括 Setheo [38] 或 leanCoP [45]，等系统，其工作方式是操作部分证明，并在必要时执行回溯。

子目标归约类的缺点是，搜索空间中冗余的“无用”公式可能会被重复探索，除非特别注意“记住”证明器此前已经探索过的路径。为了提高推理效率而避免这类冗余情况，是一个备受关注的课题。因此，人们提出并研究了定理证明中子目标归约方法内的全局细化 [11]。一般而言，这类细化可能包含复杂的命题结构；例如，信息“如果子句 C 和 D 出现在当前证明尝试中，并且当前替换将 $x \mapsto t$ 和 $y \mapsto s$ 绑定，那么我们就陷入了死胡同，必须回溯”便是一个难以推理的公式。

布尔可满足性（SAT）求解器中通常会实现回溯机制 [13], ，以完善（部分）证明。现代 SAT

求解器会在运行过程中学习相关信息，甚至允许用户在求解器搜索模型时，根据求解器当前的（部分）赋值添加约束。当 SAT 求解器无法找到满足赋值时，会给出以不可满足核形式呈现的解释。本文将探讨一阶定理证明中的这些学习和解释功能；在这一领域，SAT 求解器通常是管理全局信息及其完善过程的理想工具。

本文关注 SAT 求解与子目标归约的结合，重点是 Bibel 连接法 [10]。连接方法为表达力强的逻辑的自动化提供了强大的演算系统，例如经典一阶逻辑 [10,38], 高阶逻辑 [2], 线性逻辑 [31], 以及若干模态逻辑 [46]。我们的工作将连接证明的搜索直接编码为布尔可满足性问题，使求解器能够主导搜索决策，并通过断言约束作出响应，从而在得到满足赋值时，该赋值即表示一个完整证明。为此，我们引入三种连接编码（第 3–5 节），并展示它们的互补性与效力（第 7 节）。

首先，我们介绍该方法在连接表上的应用（第 3 节），并指出该设定的一些不利性质。因此，我们利用矩阵形式细化连接演算的编码（第 4 节）。此外，我们说明如何利用不可满足核在 SAT 求解过程中引导迭代加深（第 5 节）。所得编码允许许多在其他方法中通常不可行的全局细化（第 6 节）。

我们的工作旨在让 SAT 求解器返回约束的满足赋值，其中模型编码了一个完整的证明：矩阵或表。换言之，我们将一阶证明表示为命题模型。这与 SAT 求解器在定理证明中的大多数其他用途形成对比；在那些用途中，目标是实现基项不可满足性 [59], ，通常通过实例化一阶子句来见证 Herbrand 风格的反驳。

我们通过在 TPTP 一阶问题资料库 [57]上评估本项工作，展示了该方法的实际应用。为此，我们在新的求解器 UPCoP中提供了这些技术的概念验证实现UPCoP ，它使用 CaDiCaL SAT 求解器 [12] 和 Z3 SMT 求解器 [43]中的一种。将 UPCoP 与最先进的 meanCoP 求解器 [28], 进行评估比较后，实验结果表明，我们的方法能够解决 meanCoP 无法解决的 179 个问题（第 7 节）。

贡献总结。

1. 我们将一阶连接演算证明的存在性编码为布尔可满足性问题，采用连接表（第 3 节）、矩阵表示（第 4 节）以及通过不可满足核细化实现的迭代加深（第 5 节）。我们证明了这些编码的可靠性和完备性。2. 我们探讨了减少编码中冗余和对称性的优化方法（第 6 节）。3. 我们介绍了如何在新的原型求解器 UPCoP 中实现这些编码，并在 TPTP 基准测试集上对其进行评估（第 7 节）。

2 预备知识

我们采用经典一阶逻辑的标准语法和语义 [56]。项 t 等逻辑对象可以带索引：t^i_j。我们假设输入问题已经取反，并通过保可满足性变换转换为合取范式（CNF） [4]。不过，理论上这种变换是可选的 [10]。

2.1 可满足性求解

我们假设读者熟悉布尔可满足性（SAT）求解 [14] 以及理论可满足性模理论（SMT） [58]。除了针对布尔公式的基本判定过程之外，许多 SAT 求解器还支持在假设下求解以及提取不可满足核。在假设下求解允许在一次求解运行期间临时固定某些文字：之后，求解器会“忘记”这些文字及其后果。如果求解器检测到问题在这些假设下不可满足，则可以提取用于推导出不一致性的部分假设：即所谓的“不可满足核”。不可满足核可能不是极小的，因此可以使用该核的真子集推导出不一致性。生成极小核需要额外的计算成本 [21,40]。

一些 SAT 和 SMT 求解器（包括 CaDiCaL [30] 和 Z3 [15], ）允许用户通过各种方式在搜索过程中进行干预，这通常称为“用户传播”。这类机制使得求解器能够高效处理一大类问题。就我们的目的而言，我们假设：当某个 SAT 变量被赋值为真或假时，系统可以通知我们；同时，我们可以通过断言额外的约束来响应，而这些约束可能包含新的 SAT 变量。我们用 J_1、.....、 $J_n \Vdash F$表示：在求解器的当前模型满足所有前件文字 J_1、.....、 J_n的情况下，我们向求解器添加（传播）了约束 $J_1 \wedge \ldots \wedge J_n \Rightarrow F$，其中 F 为某个公式。该功能使我们能够避免急切地生成所有可能约束的庞大集合，而只添加编码中当前相关的部分。这种惰性生成方式在我们的场景中很理想。

2.2 连接表

连接表本质上是子句表推演 [25] ，并附加如下约束：添加到某一分支的每个子句必须至少有一个文字与当前叶文字相连 [39]。若两个文字具有相同的原子但极性相反，则它们相连，即互为对偶。回想一下，在一阶情形中，表中的子句会将其变量重命名为与其他任何变量互不相同，并将一个全局代换 σ 作用于整个表，从而使文字相连（合一）。连接表推演算并不合流，为达到完备性，既需要回溯，也需要对表进行公平枚举。

我们称两个文字 L、K 可以连接，并记作 $L \bowtie K$ ，如果存在某个替换 ρ ，使得 $\rho(L)$ 与 $\rho(K)$ 相连接。输入子句的一个子集被视为表 [39] 的潜在根节点：我们假设这些起始子句（猜想）已通过某种方式选定，从而至少有一个猜想

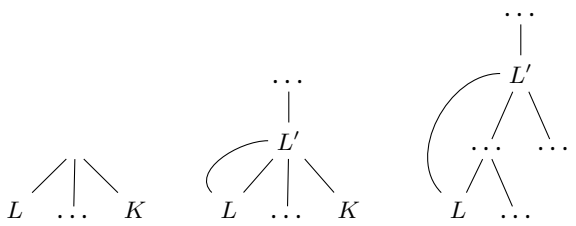


图 1：连接表规则，从左到右依次为：起始、扩展和归约。在起始和扩展操作中，$L \vee \ldots \vee K$ 是输入问题中某个子句的新重命名副本。在扩展和归约操作中，使用 σ 将 L 与 L' 连接起来。

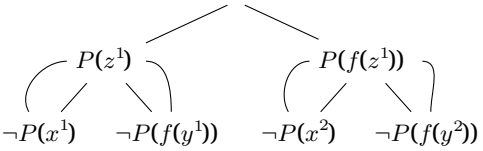
可用于寻找证明。等式无法由基本连接演算处理，因此要么将其公理化 [44]，要么通过 Brand 修改 [18] 的某种变体对其进行预处理消除。我们有时写作 C^k，以区分输入子句 C 的 k^{th} 份副本，并将其变量编为 x^k。

通常，连接表通过图 1 所示的三种操作进行处理。起始操作选取一个起始子句，并将其添加到表的根节点处。所选子句的文字构成表的初始叶节点。扩展操作将输入子句集中的某个子句复制一份，添加到表的某个叶文字下方，并将新添加子句中的至少一个文字与该叶文字连接起来。归约操作将某个叶文字与从该文字沿路径通向表根节点的路径上的另一个文字连接起来。如果某个分支的叶节点与其他文字连接，则该分支闭合。当表的所有分支都闭合时，整个表也就闭合，从而表示一个证明。一般而言，为了实现完备性，必须对所有这些操作进行回溯。

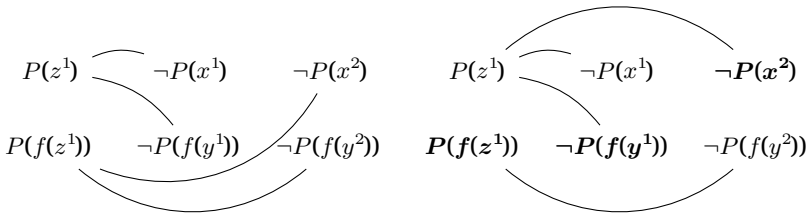
2.3 连接方法、矩阵与生成连接

连接表是连接方法的一种实例 [10]。连接演算系统是一个内容丰富、涉及诸多方面的主题，但我们主要关注以下矩阵表示。我们考虑范式矩阵，因此将矩阵定义为非空的刚性子句集合；也就是说，这些子句包含刚性变量。矩阵可以包含同一输入子句任意多个刚性副本，这些副本中的变量彼此改名以避免冲突，并应用全局代换 σ。矩阵中的一条路径是一个集合，其中恰好包含矩阵中该子句每个副本的一个文字。若该集合至少包含一对连接文字，则称其为闭合路径；否则称为开放路径。矩阵证明，也称为带有连接生成集的矩阵，是指不存在开放路径的矩阵。

此外，若矩阵中的每个文字都与另一个不同子句中的至少一个文字相连接 [37]，则称该矩阵相对于某个连接集是完全连接的。如果不存在仅使用 M 的真子集的证明，那么该矩阵证明 M 就是极小的。尽管我们可以要求矩阵中至少有一个猜想作为



(a) 用弯曲的线表示连接的语义树证明。
σ 的计算方式如下，例如， $\sigma(z^1) = f(y^1)$。



(b) 用矩阵形式表示的相同证明

(c) 一个不表示
证明。开放路径以粗体显示。

图：连接演算推导

起始子句，但与连接表证明不同，矩阵证明中并不存在内在的树结构。

例 1。为说明这两种表示，考虑这个不可满足集

$$\{\ \ \forall x \forall y.\ \neg P(x) \vee \neg P(f(y)),\ \ \forall z.\ P(z) \vee P(f(z))\ \ \}$$

并比较图2a和图2b[3]中的矩阵反驳与表反驳。图2c展示了一个全连接矩阵，但它并非证明，因为其中存在一条开放路径。

3 连接表的编码

我们首先将对封闭连接表（第2.2节）的搜索编码到 SAT 求解器中。根据满足赋值，可以显式构造出一个封闭连接表。我们使用 SAT变量 $\langle L$ 对文字 L 是路径 U 处表的叶节点这一事实进行编码；$U\rangle$，其中 U 是标记从 L 指向根节点的各节点的文字集合。 L 和 U 仅用于确定相应的变量。例如，图2a中的 $\neg P$（x^2）由变量 $\langle \neg P$（x^2）表示；$\{P$（f（z^1）$)\}\rangle$。替换 σ 以及连接文字的合一由另一组变量处理，我们将在后文讨论。

[3]每个子句的文字按列垂直书写，子句副本的顺序无关紧要

作为 SAT 的连接表规则。我们首先断言，表中必须至少存在一个起始子句 S。因此，所有文字 $L \in S$ 都必须位于根节点处的表中：

$$\bigvee_{S} \bigwedge_{L \in S} \langle L; \emptyset \rangle \tag{1}$$

SAT 求解器可以自由选择任意起始子句 S，但所选子句 S中的所有文字都必须出现在表的根节点处。当求解器将变量 $\langle L$赋值为真时，我们通过传播额外的要求作出响应。我们要求每个文字都必须应用扩展 $E_{C,\ K}$ 或归约 R_K，以便在最终的表中关闭相应分支：

$$\langle L; U \rangle \Vdash \bigvee_{C,K} E_{C,K} \vee \bigvee_{K \in U} R_K \tag{2}$$

每个公式 $E_{C,\ K}$ 都表示使用子句 C 的一个新副本，在 L 处执行扩展操作；该子句包含文字 $K \bowtie L$，由此得到

$$E_{C,K} := \left(\langle L \sim K \rangle \wedge \bigwedge_{\substack{K' \in C \\ K' \neq K}} \langle K'; \{L\} \cup U \rangle \right) \tag{3}$$

也就是说，若执行扩展步骤 $E_{C,\ K}$，则 L 和 K 相连接，而其他文字 $K' \in C$必须以路径 $\{L\} \cup U$位于表中。我们用 $\langle L \sim K \rangle$ 表示 SAT 变量，它代表 L 和 K 在模 σ意义下相连接。类似地，

$$R_K := \langle L \sim K \rangle \tag{4}$$

其中 $K \bowtie L$ 且 $K \in U$。步骤 $E_{C,\ K}$ 和 R_K 仅根据可能的连接关系 $\bowtie$计算：忽略 SAT 求解器当前维护的全局代换 σ，否则在回溯时可能会遗漏相关情况。照常可以采用迭代加深 [45], ，例如，当路径长度 $|U|$ 超过深度限制时，不提供任何 $E_{C,\ K}$ 选项。

例 2。考虑例 1 中的输入子句，并假设求解器将文字 $\langle \neg P(x^2);\ \{P(f(z^1))\} \rangle$ true 赋值为真，因此求解器认为文字 $\neg P(x^2)$ 位于表中，且在 $P(f(z^1))$ 之下。求解器传播

$$\begin{aligned} &\langle \neg P(x^2); \{P(f(z^1))\} \rangle \Vdash \\ &\qquad \big(\langle P(f(z^2)); \{P(f(z^1)), \neg P(x^2)\} \rangle \wedge \langle \neg P(x^2) \sim P(z^2) \rangle \big) \vee \\ &\qquad \big(\langle P(z^2); \{P(f(z^1)), \neg P(x^2)\} \rangle \wedge \langle \neg P(x^2) \sim P(f(z^2)) \rangle \big) \vee \\ &\qquad \langle \neg P(x^2) \sim P(f(z^1)) \rangle \end{aligned}$$

其中，前两个析取项表示应用扩展规则，最后一个表示应用归约。

合一约束。变量 $\langle L \sim K \rangle$ 约束 σ，使得 $\sigma(L)$ 与 $\sigma(K)$ 相连。当 SAT 求解器将此类文字赋值时，我们会检查这是否与现有约束集一致。这可以通过应用合一算法来完成，也可以使用高效的数据结构（例如变量轨迹 [39] ）来处理回溯。顺便一提，许多 SMT 求解器支持的代数数据类型求解器 [7], 实现了类似的判定过程。如果这些约束不可满足，我们会生成一个冲突子句，其中包含作为布尔赋值的原因。例如，如果我们有 $\langle L \sim K \rangle$、$\langle J \sim K \rangle$ 和 $\langle L \sim J \rangle$，但 $\langle L \sim K \rangle \wedge \langle L \sim J \rangle$ 已经不可满足，我们就添加该冲突

$$\neg\langle L \sim K \rangle \vee \neg\langle L \sim J \rangle \tag{5}$$

导致求解器回溯。该方法还允许对基于不等式约束的精化（如正则性）进行统一处理 [39]。

闭合连接表格的 SAT 编码。我们现在已具备 SAT 编码所需的全部要素，将其记为 $\mathcal{E}_T$。通过断言(i)必须存在一个起始子句(1)，(ii)表中的每个文字都必须对其应用归约或扩展规则(2)，以及(iii)连接必须具有由合一约束强制保证的一致合一子，我们的编码 $\mathcal{E}_T$ 即告完成。$\mathcal{E}_T$ 的每个命题模型都表示一个封闭连接表。

病态行为。我们的 SAT 编码 $\mathcal{E}_T$ 存在缺陷。重要的是，扩展会从子句集向表中添加一个新实例，因此不同 SAT 变量 $\langle L;\ U \rangle$ 的数量会迅速增长。进而，这意味着所得 SAT 问题在变量之间仅具有求解器可利用的有限命题结构。搜索往往会退化为 leanCoP [45]等系统能够高效实现的那种对可能推导的穷举，但额外增加了 SAT 求解器的开销。

4 编码 MatrixProofs

为避免先前所述的 $\mathcal{E}_T$ 问题，我们对矩阵证明进行编码（见第 2.3 节）。我们将基于矩阵的编码记为 $\mathcal{E}_M$。文献中介绍的大多数寻找连接生成集的搜索程序 [10,45] 都会限制连接，使得证明能够或可能由连接表格证明模拟 [36]。在我们的 $\mathcal{E}_M$ 编码中，我们允许矩阵中子句之间存在任意连接。这样，一个矩阵证明可以对应众多语义树证明 [37]。无论如何，我们的新表示法 $\mathcal{E}_M$ 将其转化为一个组合问题：在一组子句之间寻找连接。我们认为，与 $\mathcal{E}_T$ 相比，这种问题更适合 SAT 求解器。

4.1 编码概述

我们在给定资源限制下寻找矩阵证明，并将推理分为：

1. 我们为全连接矩阵（第 4.2 节）编码约束。

2. 我们约束结果包含一组生成连接（第 4.3 节）。

我们使用以下结果来说明编码的动机。

定理 1（全连接矩阵）。设 M 是一个具有连接生成集的极小矩阵证明。那么 M 是全连接的。

证明。首先请注意，这与 Letz 关于配对剪枝 [37] 的工作中的命题 1 相似，但并不完全相同。反设存在一个文字 $L \in C \in M$，它与任何其他 K 都没有连接。现在考虑矩阵 $M' = M \setminus \{C\}$ 的其余部分。由于 M 是极小的，因此存在一条经过 M' 的开放路径 U；否则， M' 就会有一个连接生成集。由于 L 没有与任何文字连接，$U \cup \{L\}$ 对于 M 来说是一条开放路径，这与我们假设 M 是一个证明相冲突。

定理 1 使我们能够将研究范围限定为全连接矩阵。这一限制是合理的近似，因为在实践中，没有连接生成集的全连接矩阵很少（见第 7 节）。我们使用形如 S_C 的 SAT 变量表示子句 C 出现在矩阵中，有时会添加上标 S^k_C，以表示选择 C^k，即 C 的第 k^{th} 个副本。我们称这些变量为选择器，并称当 S_C 被赋值为真时， C 被选中。至少必须选择一个起始子句 C，参见（1）：

$$\bigvee S^1_C. \tag{6}$$

在第 3 节中，我们对分支的最大长度应用迭代加深。这里无法采用这种资源限制，因为分支的概念并不明显，因此我们必须设计替代方案。我们首先对矩阵中的子句数 d 应用迭代加深。我们很快发现，每个子句最多需要引入 d 个选择器。由于我们始终引用相同的副本，求解器更容易从 $\mathcal{E}_M$ 而非 $\mathcal{E}_T$中学习到合理的冲突。我们将在第 5 节稍后讨论一种进一步增强的编码。

4.2 全连接矩阵

根据定理 1，我们可以施加约束，要求矩阵中的每个文字都必须连接到至少一个其他文字。与（2）类似，对于选择 S_C，我们通过枚举所有可能的连接，传播这样一条约束：矩阵中每个文字都必须与另一子句中的某个其他文字相连。另一子句可能已被选中，也可能需要被选中，但扩展与归约之间没有区别。假设 C 已被选中。对于每个 $L \in C$，我们传播

$$S_C \Vdash \bigvee_D \bigvee_{1 \le k \le d} \bigvee_{K \in D^k} S^k_D \wedge \langle L \sim K \rangle \tag{7}$$

其中 $K \bowtie L$ 是我们连接到的输入子句 D 中的文字，而 k 表示使用该子句的哪个副本 D^k。

有几种可选方案可以确保矩阵中被选中的子句至多为 d 个。我们建议使用伪布尔约束 [20] 或直接编码 [5,14,55]，以约束“被赋值的选择变量不超过 d 个”。在应用迭代加深时，我们可以将其加强为恰好 d 个，因此小于 d 的情况已经处理过了。

示例 3。假设情况

$$\{\ S^1_{P(z)\vee P(f(z))},\ \neg S^1_{\neg P(x)\vee\neg P(f(y))},\ \langle P(z^1)\sim\neg P(x^1)\rangle,\ \langle P(z^1)\sim\neg P(f(y^1))\rangle\ \}$$

这对应于图 2c 中去掉 3^{rd} 子句后的情况。假设每个子句最多有两个副本，我们会传播得到 $P(f(z^1))$ 必须以某种方式连接。这意味着，

$$\begin{aligned} S^1_{P(z)\vee P(f(z))} \quad \Vdash \quad & \Big(S^1_{\neg P(x)\vee\neg P(f(y))}\wedge\langle P(f(z^1))\sim\neg P(x^1)\rangle\Big)\vee \\ & \Big(S^1_{\neg P(x)\vee\neg P(f(y))}\wedge\langle P(f(z^1))\sim\neg P(f(y^1))\rangle\Big)\vee \\ & \Big(S^2_{\neg P(x)\vee\neg P(f(y))}\wedge\langle P(f(z^1))\sim\neg P(x^2)\rangle\Big)\vee \\ & \Big(S^2_{\neg P(x)\vee\neg P(f(y))}\wedge\langle P(f(z^1))\sim\neg P(f(y^2))\rangle\Big) \end{aligned}$$

根据可用子句及其副本，这些就是所有可能的选项。

4.3 连接的生成集合

一旦我们得到一个全连接矩阵，就检查是否存在开放路径（见第 2.3 节）。如果不存在，我们就完成了，可以使用所得的 SAT 模型输出证明，其中包括选定的子句、连接和全局代换。反之，假设存在一条开放路径 U ——由文字集合给出——穿过矩阵 M。沿着 U 的至少两个文字必须连接到 $\bar{}$，才能构成 M 证明。令 S 为被赋值为真的选择器集合。传播

$$\bar{S}\Vdash\bigvee_{\{L,K\}\subseteq U}\langle L\sim K\rangle \tag{8}$$

迫使求解器“修正” M，很可能通过回溯，要求 U 不是开放路径。

示例 4。继续示例 3，假设 SAT 求解器进一步赋值

$$\{\ S^2_{\neg P(x)\vee\neg P(f(y))},\ \langle P(f(z^1))\sim\neg P(f(y^2))\rangle,\langle P(z^1)\sim\neg P(x^2)\rangle\ \}$$

真。结果是全连接矩阵，如图 2c 所示。我们无法继续传播尚未被满足的子句。然而，我们可以找到开放路径 $\{\ P(f(z^1))\ \ \neg P(f(y^1))\ \neg P(x^2)\ \}$，我们用其排除

$$\begin{aligned} S^1_{P(z)\vee P(f(z))}, S^1_{\neg P(x)\vee\neg P(f(y))}, S^2_{\neg P(x)\vee\neg P(f(y))} \Vdash \\ \langle P(f(z^1))\sim\neg P(f(y^1))\rangle\vee\langle P(f(z^1))\sim\neg P(x^2)\rangle. \end{aligned}$$

由于 $\langle P(f(z^1))\ \sim\neg P(f(y^1))\rangle$ 和 $\langle P(f(z^1))\ \sim\neg P(x^2)\rangle$ 都无法产生一致合一子，因此 SAT 求解器必须回溯。

4.4 矩阵编码的正确性与复杂度

编码 $\mathcal{E}_M$ 由 (6)、(7)、(8) 以及深度限制的约束构成。其模型表示全连接矩阵证明。对于 $\mathcal{E}_M$ 中给定的大小 d，我们证明其可靠性、完备性和终止性，并说明 $\mathcal{E}_M$ 各自所属的复杂性类。

定理 2（可靠性）。 $\mathcal{E}_M$ 的命题模型表示一个具有连接生成集的矩阵。

证明。每当 SAT 求解器找到一个命题模型，我们首先检查它是否表示一个证明；如果不是，则添加约束（第 4.3 节）。

定理 3（完备性）。如果存在一个矩阵 M 以及一个连接生成集，那么在深度 $d = |M|$处存在一个 $\mathcal{E}_M$ 的命题模型。

证明。这样的矩阵证明 M 可以通过将 S^k_C 设为真来表示，当且仅当 M中至少有 k 个 C 副本。连接生成集通过将 $L \sim K$ 设为真来表示，当且仅当 L 在证明中与 K 相连。 $\mathcal{E}_M$ 的这个模型及其所有子模型，在遵循 $\sim$ 的语义和（7）的所有可能实例的意义下都是一致的。此外，最终模型满足深度约束，并且包含至少一个起始子句。另外，由于连接集是生成集，我们不会在第 4.3 节的最终检查中阻止该模型。

定理 4（复杂性界）。对于输入规模和矩阵证明规模而言，求解我们的特定编码 $\mathcal{E}_M$ 属于复杂性类 Σ^P_2。

证明。SAT 变量的数量是多项式级的。为此，设 c 是输入中的子句数量，其中总共包含 l 个文字。我们至多有 $d \cdot c$个选择器 S^k_C。此外，我们还有 $O(d^2l^2)$ 个可能的连接文字 $\langle L \sim K \rangle$。因此，(7) 的实例化数量是多项式级的。在最坏情况下将它们全部添加后，该问题属于 NP。我们可以非确定性地猜测所有多项式数量的选择器和合一原子的赋值。显然，可以在确定性多项式时间内检查模型。检查一个矩阵是否表示矩阵证明是 co-NP 完全问题。可以通过独立 SAT 求解器解决该问题，该求解器检查由 SAT 模型表示的矩阵 $\sigma(M)$是否可满足。由于我们可以使用 co-NP 预言机在 NP 中求解 $\mathcal{E}_M$，因此，对于某个固定上限 d，求解我们的编码这一问题属于 Σ^P_2。

由于判断一组关于刚性变量的子句是否可满足是 Σ^P_2-完全的 [33], ，我们方法的复杂度与其理论界限一致。

推论 1（终止性）。在固定 d 下求解 $\mathcal{E}_M$ 的运行会终止。

5 通过不可满足核精化进行迭代加深

我们的编码 $\mathcal{E}_M$ 的一个缺点，尤其是第 4.2 节中的约束，是我们会急于为不需要的子句实例引入并使用选择器。如果输入子句不止一个，且矩阵大小为 d，那么出于算术原因，并非所有子句都能在矩阵中拥有 d 个副本。因此，为每个子句创建 d 个实例实属多余。本节将应对这一挑战，并通过不可满足核改进迭代加深，从而得到一种改进后的编码 $\mathcal{E}_U$。

我们采用抽象-精化 [23] 方法，估算每个子句所需的副本数量。这样可以避免将可能不必要的子句实例引入搜索空间。我们不再使用粗略的全局限制 d，而是通过重数 μ [9]估算每个子句所需的副本数量。初始时，对于起始子句，我们有 $\mu(C) = 1$；其他情况下则有 $\mu(C) = 0$。我们根据以下编码的不可满足核单调地增加重数。我们将约束 (7) 精化为

$$S_C \Vdash \bigvee_{D} \bigvee_{1 \le k \le \boldsymbol{\mu(D)+1}} \bigvee_{K \in D^k} S_D^k \wedge \langle L \sim K \rangle \tag{9}$$

因为我们有 $\mu(D)$ 个 D的副本。请注意，k 的取值范围最大为 $\mu(D) + 1$。我们添加临时断言[4] $\kappa_D := \neg S_D^{\mu(D)+1}$，使求解器无法选择 $D^{\mu(D)+1}$，但可以报告证明查找失败的部分原因是缺少 D的副本。我们修订 (8)，因为现在不能再假设全连接矩阵恰好有 d 个子句。矩阵可以是全连接的，但所需的矩阵证明实际上可能是当前矩阵的真超集。由于 (8) 现在过强，我们将其弱化为

$$\bar{S} \Vdash \left[\bigvee_{\{L,K\} \subseteq U} \langle L \sim K \rangle \right] \vee \bigvee_{\boldsymbol{L \in U}} \boldsymbol{F_L} \tag{10}$$

其中，F_L 是一个公式，表示 L 也可以与尚未纳入矩阵的某个子句中的另一个$\bar{}$文字相连；S 则是与式（8）中相同的一组选择器。

示例 5. 假设我们有 $\mu(P(z) \vee P(f(z))) = \mu(\neg P(x) \vee \neg P(f(y))) := 2$，再来看看示例 3。在这次传播中，我们还需要列出 $S^3_{\neg P(x) \vee \neg P(f(y))}$ 为真的情况。$S^3_{P(z) \vee P(f(z))}$ 的情况则不需要，因为无论如何都无法将其连接起来。由于我们假设 $S^3_{\neg P(x) \vee \neg P(f(y))}$ 为假，SAT 求解器不会将其赋值为真；但如果在更高的深度限制下存在可行的后续步骤，它可能会在不可满足核中报告该原子。在示例 4 中，我们还需要列出以下附加情况：

$$S^2_{P(z) \vee P(f(z))} \wedge \langle \neg P(f(y^1)) \sim P(z^2) \rangle,\ S^2_{P(z) \vee P(f(z))} \wedge \langle \neg P(f(y^1)) \sim P(f(z^2)) \rangle,$$

$$S^2_{P(z) \vee P(f(z))} \wedge \langle \neg P(x^2) \sim P(z^2) \rangle,\ S^2_{P(z) \vee P(f(z))} \wedge \langle \neg P(x^2) \sim P(f(z^2)) \rangle,$$

$$S^3_{P(z) \vee P(f(z))},\ S^3_{\neg P(x) \vee \neg P(f(y))}$$

[4] 之所以命名为 κ，是因为它表示一个子句还需要更多“κ-性”

由于相应的子句选择器无论如何都被假设为假，因此我们无需为超出深度限制的子句副本引入连通性原子。不过，对编码稍作调整并进行优化，也可以将这些方面纳入编码。此外，同时列出子句 $P(z) \vee P(f(z))$ 的第二个和第三个副本，展示了第 6 节讨论的一种对称性来源。

每当 SAT 求解器报告不可满足时，我们就提取不可满足核，它表示一个可能并非最小的 κ 断言子集，足以导致不可满足。如果相应断言出现在不可满足核中，我们可以增加一个或多个 $\mu(C)$。不过，为了保持完备性，我们需要确保最终增加不可满足核中反复出现的每个子句的重数：换句话说，我们要求公平性。如果核为空，我们就可以得出结论：不存在证明。因此，采用改进的迭代加深的 SAT 编码 $\mathcal{E}_U$ 由 (6)、(9) 和 (10) 给出。

例 6。考虑输入问题

$$C := P(a) \qquad D := \forall x.\, \neg P(x) \vee P(f(x)) \qquad E := \forall y.\, \neg P(y)$$

以 C 作为起始子句。无论 κ_D 的重数是多少，它始终都会包含在不可满足核中。然而，公平的枚举最终会包含 κ_E。
通过与定理 2 和定理 4 类似的论证，我们改进后的编码 $\mathcal{E}_U$ 仍然是可靠且可终止的。完备性则需要调整论证。
定理 5（完备性）。如果存在一个矩阵 M 以及一个连接生成集，那么就存在一个与 $\mathcal{E}_U$对应的命题模型。
证明。除了定理 3 之外，我们还证明：如果存在一个使用 M 的证明，而当前的 μ 不允许该证明，那么至少有一个相关的 κ_C 包含在不可满足核中。公平性随后确保我们最终会找到该证明。考虑一个在 μ处可表示的极大子集 $M' \subset M$。

1. 如果 M' 无法完全连接，其中就至少包含一个没有连接的文字 L。由于 M' 是极大的，且 M 可以完全连接，因此 L 应当连接到子句 D 中的某个文字，而该子句尚不在矩阵中。(9) 提供了这一选项，但由于相应的 κ_D 假设被强制为假，该选项失败。因此，κ_D 属于不可满足核。2. 如果 M' 可以完全连接，那么我们就会未能闭合某条开放路径 U，并传播了 (10) 的某个实例。根据 (10) 的右析取项，某个 $L \in U$ 必须连接到一个尚未包含在 M' 中的子句中的至少一个文字。由于 M' 是极大的，我们无法添加任何子句，因此约束因 κ 假设而失败。

编码 $\mathcal{E}_U$ 的一个副作用是，它也会在某些非定理上终止。

6 冗余消除

在求解第 3–5 节中的 SAT 编码时，限制 SAT 求解器的搜索空间是有益的。除了重言式消除等标准技术 [39], ，我们还提出了一些专门的冗余消除方法。

6.1 多重性对称

我们在第 3–5 节中的编码包含若干对称性 [1], ，现在我们选择避免这些对称性，而非打破它们 [51]。第一种对称性在于，子句的副本可以互换。假设我们将某个文字 L 与尚未加入矩阵的某个副本 C中的文字 K 相连，随后却未能沿该方向找到证明。没有任何因素能阻止 SAT 求解器选择 C 的另一个副本，并因相同原因失败。我们通过传播来避免这一点

$$S_C^{i+1} \Vdash S_C^i, \tag{11}$$

强制仅当选中 C^j 和 $j < i$ 时才选中 C^i 。

6.2 包含与实例对称

饱和系统通常会删除子句 C ，因为它被某个更一般的子句 D所包含， [47] 。连接系统中的动态过程有所不同，因为不会推导出新的一阶子句，但我们可以通过应用一些包含关系而获益。如果当前矩阵中有两个不同的子句 C 和 D ，我们可以强制它们在模 σ[5] 的意义下互不包含。根据 Bibel 引理 6.8 [10]，这一限制保持了完备性。

这一想法的一个显而易见的扩展是从矩阵中移除被输入集中的其他子句所包含的子句。然而，这种做法行不通。

例 7. 考虑以下四个输入子句

$$C := P(a) \quad D := Q(a) \quad E := \forall x.\, \neg P(x) \vee Q(x) \quad F := \forall y.\, \neg Q(y)$$

其中 C 是唯一的起始子句。存在一个不使用包含的证明，依次经过 C、E，最后到达 F；这是使用 C 的唯一极小证明。然而，将 E 与 σ（x）一同放入矩阵后， $= a$ 会使其被 D 包含。

通常意义上的包含，即用较小的子句表示对较大子句的任意使用，并不成立。这种做法行不通，是因为我们可能会失去将某个子句连接到当前矩阵的理由。保留较大的子句也行不通，因为我们可能无法连接较大子句中的所有文字。尽管如此，我们仍可以借此引出额外的对称性规避。定义输入子句上的任意全序 $\prec$ ，使起始子句为最小元素。我们假设矩阵中各子句的顺序与其所复制自的输入集中的子句顺序相同。

引理 1（实例对称）。假设存在一个矩阵证明 M ，其中包含一个带有 $D \succ C$ 的子句 D ，并且存在一个 ρ ，使得 ρ（C） $= \sigma$（D）。那么，将 D 替换为 C 后的 M 也具有一个生成连接集。

证明。由于 C 和 D 中的所有变量都是新引入的，我们可以根据 ρ 调整 σ 。这样一来， C 可能会与 D 连接到相同的文字。由于 $\rho(C)$ 与 $\sigma(D)$ 含有相同的文字，我们既不会添加必须闭合的额外路径，也不会因删除相应的文字而阻止其他子句连接到 C 。

[5] 我们不对任一侧应用额外的替换

推论 2（实例对称完备性）。剪枝包含此类子句 D 的模型仍保持完备。

6.3 代换对称性

对称性出现在应用于同一子句不同副本的代换中。

例 8。考虑同一子句两个副本中的一个文字，$L[x]$ 和 $L[y]$。假设所有采用 $\sigma(x) = a$ 和 $\sigma(y) = b$ 的尝试均失败。没有任何因素阻止再次尝试，将所有连接“翻转”到另一个子句，以及 $\sigma(x) = b$ 和$\sigma(y) = a$，从而在最坏情况下引入指数数量的分支。

我们对施加于同一子句各副本中变量的代换强制一种排序。这种项的排序应在替换下保持稳定，并尽可能多地定向各项，但不必具有子项性质，因此可能不是约化序 [3]。我们建议如下排序。假设函数符号上有任意全序 $\prec$。定义 $f(\bar{t}) \prec g(\bar{s})$ 当且仅当 (i) $f \prec g$ 或 (ii) $f = g$ 且 $\bar{t} \prec \bar{s}$。项序列 $\bar{t} \prec \bar{s}$ 按字典序进行比较。设 $\bar{x}$ 为子句 C中从左到右出现的变量。给定同一子句的两个实例 C^i 和 C^j，且 $i < j$，我们可以强制$\sigma(\bar{x}_i) \not\succeq \sigma(\bar{x}_j)$，以避免子句代换上的对称。

引理 2（生成序）。假设 M 具有一个连接生成集，并且包含同一子句的两个副本 C^i 和 C^j。那么存在一个连接生成集，满足 $\sigma(\bar{x}_i) \not\succeq \sigma(\bar{x}_j)$。

证明。如果这一条件尚未成立，则有 $\sigma(\bar{x}_i) \succeq \sigma(\bar{x}_j)$。重复子句已经被消除，因此实际上 $\sigma(\bar{x}_i) \succ \sigma(\bar{x}_j)$。现在，通过交换 C^i 和C^j 各自的连接来“互换”它们，从而得到一个新的连接生成集和一致代换 σ'。必然有 $\sigma'(\bar{x}_i) \prec \sigma'(\bar{x}_j)$。

通过反复应用引理 2，可以将任意连接生成集“重新排序”为另一个符合该顺序的连接生成集。

推论 3（代换对称完备性）。对同一子句的副本中变量的替换施加排序约束，仍能保持完备性。

7 实现与实验

实现。我们在新原型 UPCoP[6]中实现了第 3–6 节的编码与优化。UPCoP 使用 CaDiCaL SAT 求解器 [30] 的用户传播器，或 Z3 SMT 求解器 [15]的用户传播器。

由于我们通过 SAT 变量进行编码，UPCoP 在做出决策时可以选择任意原子进行赋值。然而，这种任意的原子

[6] 在https://github.com/CEisenhofer/UPCoP

求解器	$\text{UPCoP}_{\text{SAT}}$	$\text{UPCoP}_{\text{SMT}}$	$\text{UPCoP}_{\text{SAT}}$	$\text{UPCoP}_{\text{SMT}}$	$\text{UPCoP}_{\text{SAT}}$	$\text{UPCoP}_{\text{SMT}}$	meanCoP
Enc.	$\mathcal{E}_M$		$\mathcal{E}_U$		$\mathcal{E}_H$		
已求解	928	855	1,152	1,055	1,272	1,264	1,972
独有	27	20	109	93	105	76	551

图3：实验总结。$\text{UPCoP}_{\text{SAT}}$ 和 $\text{UPCoP}_{\text{SMT}}$ 表示基于 CaDiCaL （SAT）和Z3（SMT）的 UPCoP版本。“独有”列出了可由 UPCoP求解但meanCoP无法求解的问题数量，反之亦然。

选择与目标导向性的概念相冲突。例如，考虑初始约束 (6)：必须选取某个猜想子句的至少一个选择器。然而，这一约束并不要求求解器一开始就选择其中一个子句，只要求在最终模型中选中一个起始子句。因此，求解器可能会选取任意一个选择器，并由此导出一个无法与任何猜想连接的子矩阵。在传播 (7) 中所有可能的候选项时，也可能出现类似情况。因此，我们强制UPCoP 始终选择当前相关的选项之一，而不是将选择留给 UPCoP 的内部启发式。此外，我们让 UPCoP 将当前无关的选择器赋值为假，而不是赋值为真。

我们注意到， UPCoP 可能会在矩阵中任何已选子句都不需要的项等式和排序上耗费大量推理时间。为解决这一问题， UPCoP 将每个非基项唯一地关联到其来源的子句副本。这样，只有在相应的选择器被赋值为真时，我们才会处理这些非基项所报告的（不）等式。

实验设置。我们使用 AMD EPYC 7502 处理器（主频约为 1.8 GHz），将每次运行的内存限制设为 16 GB，超时时间设为 30 秒，在 TPTP 基准测试集 [57] 上评估了 UPCoP 。我们考虑了 TPTP 8.2.0 问题库中全部 6468 个可证明的一阶问题，并预先将其转换为 SMT-LIB 输入格式 [6]。

我们使用以下三种编码评估了 UPCoP 。首先，我们分别采用第 4–5 节中的编码 $\mathcal{E}_M$ 和 $\mathcal{E}_U$ 。此外，我们还考虑了一种混合方法 $\mathcal{E}_H$ ，它以这样的方式结合了 $\mathcal{E}_M$ 和 $\mathcal{E}_U$ ：每次失败时都会增加每个（非基）子句的容量，但矩阵中的子句集受可用副本数量限制，而非受严格上限限制。对于每种编码，我们分别使用 CaDiCaL 和 Z3 的用户传播器评估了 UPCoP 。因此，我们的实验结果报告了 UPCoP 的六种不同设置，如图3所示。此外，我们还将 UPCoP 与 meanCoP [28] 的默认（完备）模式进行了比较。

实验分析。我们的结果汇总于图3， UPCoP 的所有版本总共解决了1,601个问题。虽然 UPCoP 解决的问题少于 meanCoP，但我们强调， UPCoP 解决了179个属于各种不同类别、而 meanCoP 无法解决的问题。这有许多不同的原因。除了连接哪些文字这一选择具有“非确定性”之外，主要还有两个原因。首先， UPCoP 找到的一些矩阵证明实际上明显小于相应的表形式证明——

meanCoP 的表形式证明。其次，基于不可满足核的编码有时确实能推断出某些子句完全无关，因此将其忽略，从而合理地提升了速度。

出乎意料的是， UPCoP 通常不会在消除非生成连接时陷入停滞。 UPCoP 无法终止的情况大多是因为，受前文所述因素影响， UPCoP查找全连接矩阵的速度太慢。一旦找到全连接矩阵，只需拒绝少数几个矩阵，它就会成为生成连接。尽管跨接性检查是 co-NP 完全问题，但我们没有遇到检查耗时显著的情况。

我们注意到，对于规模较小的证明（可能不是树状结构）， $\mathcal{E}_M$ 能够快速终止。 $\mathcal{E}_U$ 最适合处理含有冗余且无法连接子句的大型问题，因为我们不会为冗余子句生成更多副本。对于证明会使用输入中的大多数子句、但每个子句只使用少数几次的问题，$\mathcal{E}_H$ 最能发挥作用。
讨论。分析实验结果时，我们注意到 UPCoP 仍可从以下方面改进：

– UPCoP 使用的 SAT/SMT 求解器会不断学习有关底层证明系统的更多事实，其中大多会用到新的布尔变量。因此，求解器可能会选择对这些布尔变量进行分支。这会拖慢证明搜索，因为求解器可能会花费大量精力学习有关部分证明的事实，而这些部分证明无法扩展为完整证明。此外，其中一些布尔赋值代表的部分证明，朴素的证明枚举甚至无法触及。

– 从这些无果证明空间中学习到的子句也不一定能够组合起来：例如，求解器可以同时考虑多个子矩阵，尽管这些子矩阵永远无法组合。冲突学习和 SAT 启发式方法会引导求解器进行更有针对性的搜索这一假设，并未得到证实。推迟传播那些未连接到主矩阵的子矩阵所产生的可能连接，有助于减少布尔变量的数量以及传播可能未使用约束所带来的开销，但需要额外且代价高昂的簿记工作。

– 调整 SAT/SMT 求解器的决策启发式方法，使其优先选择与当前证明状态相关的布尔变量，或许会有所帮助 [41]。然而，这种调整会损害求解器内部高效的变量选择启发式方法。– 编码 $\mathcal{E}_T$ 会受到影响，因为所学习到的冲突主要取决于表中的具体路径。同样，在矩阵编码 $\mathcal{E}_M$ 和 $\mathcal{E}_U$ 中，针对布尔变量学习到的冲突也无法很好地从一个副本推广到另一个副本，因而同一子句的多个副本会以同构的方式被反复考虑。– 此外，SAT/SMT 求解器在处理子句时，传播循环会带来不可忽视的开销。通过 SAT 模拟单条规则的过程中，朴素证明枚举求解器可以在同一时间内处理多个步骤。

如前所示并讨论过的，解决其中大多数问题的一种方法，是实现一种定制的冲突学习变体，它不具备

现代 SAT 求解器大多固有的那些已讨论行为。基于这些结果，我们进一步构建了系统 hopCoP [49]，该系统为连接演算的语义树形式实现了这样一种定制的冲突学习引擎。

8 相关工作

大多数一阶定理证明器都采用各种基于基项的推理技术，其中主要是 SAT 和 SMT 求解器。这里必须提到基于实例的方法 [8]：将一组一阶子句实例化为基项，期望它们变得不可满足；这种方法可以通过专用演算 [34] 使用，也可以与现有系统 [50,53] 配合使用。反过来，SMT 求解器通常会将量词实例化集成到可满足性例程中 [32,42]。基于基项的推理还可用于一阶定理证明器中的许多其他组合任务 [52],，例如跟踪子句拆分 [59],、检测包含 [48],，或确定推理规则是否适用 [24]。MACE 风格的有限模型构造器 [22] 采用 SAT 求解来确定一组子句是否可满足，假设模型大小固定且有限，并使用对称性破缺 [51]。

除我们采用自定义冲突学习引擎、并将自身限制于证明对象的直接编码这一方法 [49] 之外，ChewTPTP 系统的 SAT [26] 和 SMT [17] 实现都是现有通过可满足性求解进行定理证明的方法中与我们最为接近的。ChewTPTP 完全预先为封闭连接表编码约束，然后将所得约束传递给 SAT 或 SMT 求解器。高阶系统 Satallax [29] 及其分支 Lash [19] 也将证明搜索归约到命题层面。然而，与 UPCoP不同，当 Satallax 的 SAT 求解器报告不可满足性时，就表示已经找到了证明。我们此前已在更一般的环境中发表过这些想法的早期版本 [27]。

9 结论

我们将一阶连接演算编码为命题问题，改进了用于矩阵证明的 SAT 编码，并利用不可满足核心引导迭代加深。我们引入了若干优化，以在证明搜索过程中剪除对称并消除不必要的分支。对新求解器 UPCoP 的初步实验表明，我们的编码可用于解决某些问题，而其他连接求解器无法解决这些问题。鉴于求解过程中遇到的上述问题，进一步的改进可能来自应用自定义垃圾回收和变量选择启发式策略UPCoP 。

致谢。 本研究全部或部分由 ERC 协同资助项目 ARTIST 101002685、ERC 概念验证资助项目 LEARN 101213411、维也纳工业大学博士学院 SecInt、FWF SpyCoDe 资助项目 10.55776/F85、WWTF ForSmart 资助项目 10.47379/ICT22007 以及亚马逊研究奖 2023 QuAT 资助。

利益披露。作者声明，他们没有与本文内容相关的竞争性利益。

参考文献

1. Aloul, F.A., Sakallah, K.A., Markov, I.L.：布尔可满足性的高效对称性破缺。IEEE Trans. Computers **55**(5), 549–558 (2006)。https://doi.org/10 75 2 1109 2006 . / TC. . . Andrews, P.B.：关于连接与高阶逻辑。自动推理杂志 **5**(3), 257–291 (1989)。https://doi.org/10.1007/BF00248320，https://doi.org/1000248320 31007./BF. Baader, F., Nipkow, T.：《项重写及相关内容》。剑桥大学出版社 (1998) 4. Baaz, M., Egly, U., Leitsch, A.：范式变换。载于：《自动推理手册》（共2卷），第273–333页。爱思唯尔和麻省理工学院出版社 (2001)。https://doi.org/10 2 5 1016 978 044450813 3 50007 . /B - - / - . Bailleux, O., Boufkhad, Y.：布尔基数约束的高效CNF编码。载于：约束规划。LNCS，第2833卷，第108–122页 (2003)。https://doi.org/10 8 8 6 1007 978 3 540 45193 . / - - - - . Barrett, C., Fontaine, P., Tinelli, C.：《可满足性模理论库（SMT-LIB）》。 www.SMT-LIB.org (2016) 7. Barrett, C.W., Shikanian, I., Tinelli, C.：归纳数据类型理论的抽象判定过程。J. Satisf. Boolean Model. Comput. **3**(1-2), 21–46 (2007 190028 8 10 3233)。https://doi.org/ . /SAT . Baumgartner, P., Thorstensen, E.：基于实例的方法——简要概述。K¨unstliche Intell.**24**(1), 35–42 (2010)。https://doi.org/10.1007/S13218-010-0002-X 9. Bibel, W.：矩阵中的配对。Commun. ACM **26**(11), 844–852 (1983)。https://doi.org/10 183 10 1145 182. / . . Bibel, W.：自动定理证明。Artificial intelligence，Vieweg，第2版修订版 (1987) 11. Bibel, W.：证明方法比较。载于：AReCCa。第119–132页 (2023), https://ceur-ws.org/Vol-3613/ 12. Biere, A., Fazekas, K., Fleury, M., Heisinger, M.：CaDiCaL、Kissat、Paracooba、Plingeling和Treengeling参加2020年SAT竞赛。载于：2020年SAT竞赛论文集——求解器与基准测试描述。第51–53页。赫尔辛基大学计算机科学系报告系列B (2020) 13. Biere, A., Froleyks, N., Wang, W.：CadiBack：使用CaDiCaL提取骨干。载于：SAT。LIPIcs，第271卷，第3:1–3:12页 (2023)。https://doi.org/10 3 14 4230 2023 . /LIPICS.SAT. . . Biere, A., Heule, M., van Maaren, H., Walsh, T.（编）：《可满足性手册》第二版，《人工智能与应用前沿》，第336卷。IOS出版社 (2021 10 3233 336)。https://doi.org/ . /FAIA. Bjørner, N.S., Eisenhofer, C., Kovacs, L.：Z3中的模自定义理论的可满足性。载于：VMCAI。LNCS，第13881卷，第91–105页 (2023)。https://doi.org/10 1 5 16 1007 978 3 031 24950 . / - - - - . Bonacina, M.P.：定理证明策略分类。载于：《今日人工智能：近期趋势与进展》，LNCS，第1600卷，第43–84页。Springer (1999)。https://doi.org/10.1007/3-540-48317-9 3

17. Bongio, J., Katrak, C., Lin, H., Lynch, C., McGregor, R.E.: 在SMT中编码一阶证明。载于: SMT。ENTCS, 第198卷, 第71–84页 (2007)。https://doi.org/10 081 18 1016 2008 04 . /J.ENTCS. . . . Brand, D.: 使用修改方法证明定理。SIAM J. Comput. 40204036 19 4 412 430 1975 10 1137 1 0 ( ), – ( )。https://doi.org/ . / . Brown, C.E., Kaliszyk, C.: Lash（系统说明）。CoRR **abs/2205.06640** (2022 06640 20 10 48550 2205)。https://doi.org/ . /ARXIV. . . Chai, D., Kuehlmann, A.: 快速伪布尔约束求解器。IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **24**(3), 305–317 (2005)。https://doi.org/10 842808 21 1109 2004 . /TCAD. . . Cimatti, A., Griggio, A., Sebastiani, R.: 在可满足性模理论中计算小型不可满足核。J. Artif. Intell. Res. **40**, 701–728 (2011)。https://doi.org/ 10 1613 3196 . /JAIR. . Claessen, K., Sorensson, N.: 改进MACE风格有限模型寻找的新技术。载于: CADE-19 研讨会论文集: 模型计算——原理、算法与应用。第11–27页 (2003) 23. Clarke, E.M., Grumberg, O., Jha, S., Lu, Y., Veith, H.: 用于符号模型检验的反例引导抽象精化。J. ACM **50**(5), 752–794 (2003)。https://doi.org/ 10 1145 876638 876643 . / . . Coutelier, R., Kovacs, L., Rawson, M., Rath, J.: 基于SAT的包含消解。载于: CADE。LNCS, 第14132卷, 第190–206页 (2023)。https://doi.org/ 10 1007 978 3 031 38499 8_11 . / - - - - . D'Agostino, M., Gabbay, D.M., Hahnle, R., Posegga, J.: 《语义树方法手册》。Springer Science & Business Media (2013) 26. Deshane, T., Hu, W., Jablonski, P., Lin, H., Lynch, C., McGregor, R.E.: 在SAT中编码一阶证明。载于: CADE。LNCS, 第4603卷, 第476–491页 (2007)。https://doi.org/ 10 1007 978 3 540 73595 3_35 . / - - - - . Eisenhofer, C., Kovacs, L., Rawson, M.: 将连接演算嵌入可满足性模理论。载于: AReCCa。第54–63页 (2023), https://ceur-ws.org/Vol-3613/ 28. F¨arber, M.: 一种效果出奇出色的连接表回溯策略。载于: AReCCa。第23–40页 (2023), https://ceur-ws.org/Vol-3613/ 29. F¨arber, M., Brown, C.E.: Satallax的内部引导。载于: IJCAR。LNCS, 第9706卷, 第349–361页。Springer (2016)。https://doi.org/10 1 24 30 1007 978 3 319 40229 . / - - - - . Fazekas, K., Niemetz, A., Preiner, M., Kirchweger, M., Szeider, S., Biere, A.: IPASIR-UP: 用于CDCL的用户传播器。载于: SAT。LIPIcs, 第271卷, 第8:1–8:13页 (2023 8 31 10 4230 2023)。https://doi.org/ . /LIPICS.SAT. . . Galmiche, D.: 《线性逻辑中的连接方法与证明网构造》。Theoretical Computer Science **232**(1), 231–272 (2000)。https://doi.org/https://doi.org/10.1016/S0304-3975(99)00176-0, https://www.sciencedirect.com/science/article/pii/S0304397599001760 32. Ge, Y., de Moura, L.M.: 可满足性模理论中量化公式的完备实例化。载于: CAV。LNCS, 第5643卷, 第306–320页 (2009)。https://doi.org/10 4 25 33 1007 978 3 642 02658 . / - - - - . Goubault, J.: 资源有界一阶经典逻辑的复杂性。载于: STACS。第59 8 131 34 70 1994 10 1007 3 540 57785 – . LNCS ( )。https://doi.org/ . / - - - . Korovin, K.: Inst-Gen——一种基于实例化的自动推理模块化方法。载于: 《程序设计逻辑——纪念Harald Ganzinger文集》。LNCS, 第7797卷, 第239–270页 (2013)。https://doi.org/10.1007/978-3-642-37651-1 10

35. Kov´acs, L., Voronkov, A.: 一阶定理证明与 Vampire。载于：CAV。页码 1 8 1 36 35 2013 10 1007 978 3 642 39799 – 。LNCS（）。https://doi.org/ 。 / - - -‿ 。Kreitz, C., Otten, J., Schmitt, S., Pientka, B.: 基于矩阵的构造性定理证明。载于：智能学与计算逻辑（献给Wolfgang Bibel六十岁诞辰）。应用逻辑丛书，第19卷，第189–205页（2000）37. Letz, R.: 利用配对剪枝连接表。载于：CADE。LNCS，第1421卷，0054273 38 381 396 1998 10 1007 页码 – （）。https://doi.org/ 。 /BFB 。Letz, R., Schumann, J., Bayerl, S., Bibel, W.: SETHEO：一种高性能定理证明器。自动推理杂志 8（2），第183–212页（1992）。https://doi.org/10 00244282 39 1007. /BF 。Letz, R., Stenz, G.: 模型消去与连接表程序。载于：《自动推理手册》（两卷本），第2015–2114页。爱思唯尔与麻省理工学院出版社（ 2001 8 40 10 1016 978 044450813 3 50030 ）。https://doi.org/ 。 /B - - / - 。Lynce, I., Marques-Silva, J.: 论最小不可满足核的计算。载于：SAT（2004），http://www.satisfiability.org/SAT04/programme/110.pdf 41. de Moura, L., Bjørner, N.: 相关性传播。技术报告 MSR-TR- 2007-140，微软研究院，技术报告（2007）42. de Moura, L.M., Bjørner, N.S.: 面向SMT 求解器的高效E-匹配。载于：CADE。LNCS，第4603卷，第183–198页（2007）。https://doi.org/10 3 13 43 1007 978 3 540 73595 。 / - - - - 。de Moura, L.M., Bjørner, N.S.: Z3：一种高效的 SMT 求解器。载于：TACAS。LNCS，第4963卷，第337–340页。Springer（2008）。https://doi.org/10 3 24 44 1007 978 3 540 78800 。 / - - - - 。Nieuwenhuis, R., Rubio, A.: 基于参数归结的定理证明。载于：《自动推理手册》（两卷本），第371–443页。爱思唯尔与麻省理工学院出版社（2001 6 45 10 1016 978 044450813 3 50009 2 0 1 2 ）。https://doi.org/ 。 /B - - / - 。Otten, J.: leanCoP 。与 ileanCoP 。：经典逻辑与直觉主义逻辑中的高性能精简定理证明（系统描述）。载于：IJCAR。第283–291页。LNCS（2008 7 23 46 10 1007 978 3 540 71070 ）。https://doi.org/ 。 / - - - - 。Otten, J.: 实现一阶模态逻辑的连接演算。载于：IWIL。计算领域 EPiC 系列，第22卷，第18–32页。EasyChair（2012）。https://doi.org/10 9 47 29007 82. / M 。Ramakrishnan, I.V., Sekar, R., Voronkov, A.: 项索引。载于：《自动推理手册》（两卷本），第1853–1964页。爱思唯尔与麻省理工学院出版社（2001）。https://doi.org/10.1016/b978-044450813-3/50028-x 48. Rath, J., Biere, A., Kov´acs, L.: 通过SAT 求解实现一阶包含。载于：FMCAD。第160–169页（2022）。https://doi.org/ 10 34727 2022 978 3 85448 053 2_22 。 / /ISBN. - - - - 。 49 Rawson, M., Eisenhofer, C., Kovacs, L.: 面向非合流证明搜索的约束学习。载于：TABLEAUX。LNCS，第15980卷，第103–119页（2025）。https://doi.org/10 3 6 50 1007 978 3 032 06085 。 / - - - - 。Rawson, M., Reger, G.: 在模型消去过程中消去模型。载于：TABLEAUX。LNCS，第12842卷，第250–265页（2021）。https://doi.org/10 2 15 51 1007 978 3 030 86059 。 / - - - - 。Reger, G., Riener, M., Suda, M.: MACE风格有限模型寻找中的对称性规避。载于：FroCoS。LNCS，第11715卷，第3–21页（2019）。https://doi.org/10 8 1 52 1007 978 3 030 29007 。 / - - - - 。Reger, G., Suda, M.: SAT 求解器在 Vampire 中的应用。载于：Vampire。计算领域 EPiC 系列，第38卷，68 53 63 69 2015 10 29007 4 页码 – （）。https://doi.org/ 。 / W 。Schulz, S.: 将SAT 求解轻量级集成到一阶推理器中——初步实验。Vampire 第9–19页（2017）

54. Schulz, S., Cruanes, S., Vukmirovic, P.: 更快、更高、更强：E 2.3。载于：CADE。页码 495 6 29 55 507 2019 10 1007 978 3 030 29436 – 。LNCS（）。https://doi.org/ 。 / - - - - 。 Sinz, C.: 迈向布尔基数约束的最优CNF编码。载于：约束规划。LNCS，第3709卷，11564751 73 56 827 831 2005 10 1007 1968 页码 – （）。https://doi.org/ 。 / 。 Smullyan, R.M.: 一阶逻辑。Springer（）57. Sutcliffe, G.: TPTP问题库及相关基础设施。从CNF到TH0，TPTP v6.4.0。自动推理杂志 59（4），第483–502页（2017）58. Tinelli, C.: 面向基元理论可满足性的基于DPLL的演算。载于：JELIA。页码 308 7 26 59 319 2002 10 1007 3 540 45757 – （）。https://doi.org/ 。 / - - - 。Voronkov, A.: AVATAR：一阶定理证明器的架构。载于：CAV。LNCS，第8559卷，第696–710页（2014）。https://doi.org/10 9 46 60 1007 978 3 319 08867 。 / - - - - 。Weidenbach, C., Dimova, D., Fietzke, A., Kumar, R., Suda, M., Wischnewski, P.: SPASS 3.5版。载于：CADE。LNCS，第5663卷，第140–145页（2009）。https://doi.org/10.1007/978-3-642-02959-2 10