论天使与恶魔：动态模型的策略（解）构造

Davide Catta LIPN，法国国家科学研究中心，巴黎北索邦大学 Villetenause，法国
catta@lipn.univ-paris13.fr

Rustam Galimullin 卑尔根大学 挪威卑尔根
rusta
m.galimullin@uib.no

Munyque Mittelmann LIPN，法国国家科学研究中心，巴黎北索邦大学 Villetenause，法国
mitt
elmann@lipn.univ-paris13.fr

arXiv:2601.07690v1 [cs.LO] 12 Jan 2026

摘要
近年来，人们对形式化描述智能体如何通过修改给定模型的结构来进行策略推理的逻辑日益关注。这一研究方向受到多种应用场景的推动：在这些场景中，建模系统会随时间演化，例如通信网络、安全协议和多智能体规划。本文介绍三种用于推理如何修改加权图拓扑的策略逻辑。在策略解构逻辑中，破坏型代理（恶魔）会在一定代价范围内移除边。在策略构造逻辑中，构造型代理（天使）会在代价上限内添加边。最后，策略更新逻辑 结合了这两种代理，他们可以合作或竞争。我们研究了这些逻辑的表达能力，以及它们的模型检测问题的复杂度。

关键词
策略推理、模型改变、表达力、模型检测、模态逻辑

1 引言

自动驾驶车辆 [22] 和网络安全系统 [21], 等安全关键应用日益广泛地采用并依赖自主系统，因此需要可靠的验证方法。模型检测 [34] 是验证给定系统的安全性和可达性等时态性质的事实标准方法之一。在多智能体系统（MAS）的背景下，这种方法得到了扩展，以捕捉自主智能体之间的交互和策略行为。通常，人们会用时态逻辑或战略逻辑来表达希望验证的系统性质，例如线性时态逻辑（LTL） [68], 、计算树逻辑（CTL） [33], 和交替时态逻辑（ATL） [11]。

在模型检测中，通常使用静态模型（例如带标记的状态转移模型或并发博弈结构）来表示给定系统，该模型描述系统的一组固定配置（即状态）及其之间的转移方式。这类模型无法捕捉系统结构可能动态变化的情形，例如由智能体的行动或移除易受攻击的组件所引起的变化。静态模型的假设限制了模型检测方法的适用性，因为许多现实应用本质上都是动态的。例如，在地铁系统中新增一条线路，可能会导致换乘站出现瓶颈。另一个例子是实施防御策略，以防止网络攻击利用系统漏洞 [28]。

受这些局限性的推动，近年来，人们对用于动态系统规范描述与验证的基于逻辑的方法日益关注。一条颇具影响力的研究路线从破坏游戏 [72], 中汲取灵感。这是一种在图上的可达性游戏，其中一名玩家（称为恶魔）可以删除边，以阻止另一名玩家（旅行者）到达其目标。障碍逻辑 （OL） [28] 近期被提出，用于分析加权图上的类破坏游戏；在这类游戏中，恶魔可以暂时禁用权重不超过指定代价的边。然而，OL 只能捕捉一种特定类型的图修改：被移除的边会在旅行者行动后立即恢复。回到网络安全的例子，虽然它可以表示暂时阻止攻击者访问敏感模块的防御措施，但却无法捕捉移除易受攻击的执行路径等措施。

我们的贡献。我们提出了三种用于推理加权图策略性永久变化的新型逻辑。第一种是策略解构逻辑 （SDL），其风格与 OL 类似：破坏型代理，即恶魔，会在一定代价范围内永久移除边。在网络安全场景中，永久移除边可以在攻击者到达易受攻击的路径之前很久就将其阻断，而这是 OL 无法实现的。第二种逻辑策略构造逻辑 （SCL）则考虑构造型代理，即天使，它可以在代价上限内添加新的边。最后，策略更新逻辑 （SUL）结合了两个按照 ATL 方式并发行动的代理。在 SUL 中，天使和恶魔可以为实现同一目标而合作，其行为甚至可以包括联合策略，以模拟类似 OL 的策略。此外，SUL 还刻画了双方彼此对抗的情形，从而增强了问题的策略维度。所提出逻辑的主要优势在于，它们能够对动态系统进行建模和验证，在这些系统的执行过程中可以采用访问控制和防御机制。

对于这些新逻辑，我们首先证明，它们中的每一种都比计算树逻辑严格更具表达力。其次，我们展示 SDL 和 SCL 在表达力方面不可比较，而 SUL 包含它们二者。接下来，我们研究并讨论它们与 OL 的微妙关系。我们还研究了这些逻辑的模型检验问题，并证明对于 SDL、SCL 以及 SUL 的一个片段，该问题是 PSPACE 完全的；而对于完整的 SUL，该问题属于 EXPSPACE。

2 天使与恶魔的逻辑

我们为在具有串行转移关系的加权有向图上进行的博弈引入逻辑。这些博弈涉及能够

本文是同名论文的扩展版本，该论文将发表于 AAMAS 2026；扩展版本包含提供证明细节的技术附录。

修改图的玩家，以及沿图的边移动的旅行者 。在第一种情形中，修改图的玩家称为恶魔，她可以从图中移除一个边子集，该子集可以为空，但代价不得超过一定限度。随后，旅行者在修改后的图中移动。在第二种情形中，修改图的玩家称为天使[1]，，她能够在一定代价限度内向图中添加边。最后，我们考虑恶魔和天使同时参与的情形，由她们同时选择要添加和移除哪些边。在所有这些逻辑中，我们都能够推理修改图的玩家如何制定策略，无论旅行者如何移动，都能达成时序目标。

2.1 模型

令 At $=\{p,q,r...\}$ 为一个可数的原子命题集合。

定义 1（模型）。一个模型是一个元组 $\mathfrak{M}=(S,\to,\mathrm{V},\mathrm{C})$，其中：

- S是一个非空状态集合。• $\to\subseteq S\times S$是 S 上的串行二元关系。我们将 $s\to s'$ 记作 $(s,s')\in\to$。
- $\mathrm{V}:\mathrm{At}\to 2^S$ 是一个赋值函数，用于指定哪些原子命题在哪些状态下成立。
- $\mathrm{C}:S\times S\to\mathbb{N}^+$ 是一个代价函数，为任意一对状态赋予一个正自然数。直观而言，这个数表示移除或添加给定两个状态之间一条边的代价。

给定一个模型 $\mathfrak{M}$，其状态集记作 $S^{\mathfrak{M}}$，边集记作 $\xrightarrow{\mathfrak{M}}$。带指定状态的模型是一个有序对 $(\mathfrak{M},s)$，其中 $\mathfrak{M}$ 是一个模型，且 $s\in S^{\mathfrak{M}}$。我们将 $\mathfrak{M}\setminus A$ 记作 $(S,\to\setminus A,\mathrm{V},\mathrm{C})$，其中 $A\subseteq\xrightarrow{\mathfrak{M}}$。我们将 $\mathfrak{M}\cup A$ 记作 $(S,\to\cup A,\mathrm{V},\mathrm{C})$，其中 $A\subseteq S^{\mathfrak{M}}\times S^{\mathfrak{M}}$。

给定一个模型 $\mathfrak{M}$，我们令 $True(s)=\{p\in\mathrm{At}\mid s\in\mathrm{V}(p)\}$ 表示在状态 s 为真的原子命题集。然后，我们将 $\mathfrak{M}$ 的大小定义为 $|\mathfrak{M}|=|S|+|\to|+\sum_{s\in S}|True(s)|+\sum_{s,\,t\in S}\mathrm{C}(s,\,t)$，其中整数以二进制编码。最后，如果 $|\mathfrak{M}|$ 是有限的，我们称 $\mathfrak{M}$ 有限。

注 1。请注意，我们为移除边和添加边定义了相同的单一代价。我们本可以放宽这一假设，而论文中的所有结果仍然成立。为简便起见，我们沿用当前定义。

定义 2（（解）构造与更新）。给定两个带指定状态的模型（$\mathfrak{M},\ s$）和（$\mathfrak{M}',\ s'$），如果（$\mathfrak{M}',\ s'$）从（$\mathfrak{M},\ s$）以至多 n 的代价可解构到达，记作（$\mathfrak{M},\ s$）$\overset{n}{\rightsquigarrow}$（$\mathfrak{M}',\ s'$），当且仅当 $\mathfrak{M}'=\mathfrak{M}\setminus A$对某个 $A\subset\xrightarrow{\mathfrak{M}}$、$s\overset{\mathfrak{M}}{\longrightarrow}s'$，且 $\mathrm{C}(A)\leqslant n$，其中 $\mathrm{C}(A)$ 为 $\sum_{x\in A}\mathrm{C}(x)$。我们将称 $\mathfrak{M}'$为 $\mathfrak{M}$的一个 n-子模型 。

我们称 $(\mathfrak{M}',\ s')$ 可通过至多 n 的代价从 $(\mathfrak{M},\ s)$ 构造可达，记作 $(\mathfrak{M},\ s)\overset{n}{\triangleright}(\mathfrak{M}',\ s')$，当且仅当 $\mathfrak{M}'=\mathfrak{M}\cup A$存在某个 $A\subseteq((S^{\mathfrak{M}}\times S^{\mathfrak{M}})\setminus\to^{\mathfrak{M}})$，使得 $\mathrm{C}(A)\leqslant n$，且 $s\overset{\mathfrak{M}}{\longrightarrow}s'$。我们还将 $\mathfrak{M}'$ 称为 $\mathfrak{M}$的一个 n-超模型。

[1]移除边的玩家“恶魔”这一名称源自关于破坏博弈的文献 [15]。因此，作为恶魔的对应者，我们将添加边的智能体称为“天使”。为了减轻这些名称所带来的联想，我们本可以将恶魔称为移除者或破坏者，将天使称为构造者。事实上，在添加和移除边的设定中，如果恶魔移除通往糖糕或不理想状态的边，那么可以将她视为仁慈的；而如果天使向不理想状态添加边，则可以将她视为恶意的。或许，为了配合这一隐喻，我们可以把天界生物的行动理解为赋予或剥夺选择，而凡人——旅行者——则必须自行做出选择。

给定模型 $\mathfrak{M}$，将边集 A添加后得到其 n-超模型 $\mathfrak{M}_1$，将边集 B移除后得到其 m-子模型 $\mathfrak{M}_2$。我们将移除边 B并添加边 A后得到的模型记为 $\mathfrak{M}_1\star\mathfrak{M}_2=(\mathfrak{M}\setminus B)\cup A$。我们将 $\mathfrak{M}_1\star\mathfrak{M}_2$称为一个 n-m-更新。注意，由于边集 A和 B互不相交，$\mathfrak{M}_1\star\mathfrak{M}_2$是定义良好的。

对于两个带标记模型 $(\mathfrak{M},s)$ 和 $(\mathfrak{M}',s')$，我们称 $(\mathfrak{M}',s')$ 可通过代价 n 和 m 从 $(\mathfrak{M},s)$更新可达，记作 $(\mathfrak{M},s)\overset{n}{\Rightarrow}_m(\mathfrak{M}',s')$，当且仅当 $\mathfrak{M}'$ 是 $\mathfrak{M}$ 和 $s\overset{\mathfrak{M}}{\longrightarrow}s'$的一次 n-m-更新。

在解构可达、构造可达或更新可达这三种情境中的任意一种情境下，执行移动 $s\overset{\mathfrak{M}}{\longrightarrow}s'$ 的智能体称为旅行者。

直观地说，如果可以通过从 $\mathfrak{M}$ 中移除边来得到模型 $\mathfrak{M}'$，且移除边的总代价不超过 n，那么模型 $\mathfrak{M}'$ 就可以通过至多为 n 的代价从 $\mathfrak{M}$ 解构到达。n-超模型 $\mathfrak{M}'$ 是通过向 $\mathfrak{M}$ 添加一组新的边得到的，这些边原本不存在于 $\mathfrak{M}$ 中，且总代价不超过 n。注意，由于上述所有可达性概念都是在两个（带标记）模型之间定义的，因此所有操作都会保持序列性。

2.2 策略解构逻辑

我们将能够移除边的智能体称为恶魔。与破坏博弈不同 [72]，我们关注的不只是恶魔的一步行动，而是恶魔的策略，以确保无论旅行者采取何种行动，某种性质始终成立。

定义 3（恶魔策略）。设 π 为一个（可数）无限的带指定状态的模型序列。那么，当且仅当对于每个 $i\geqslant 0$，都有 $\pi(i)\overset{n}{\rightsquigarrow}\pi(i+1)$ 时，π是一个代价为 n的递减模型路径。请注意，这样的序列确实是无限的，因为我们总是可以取给定模型的一个 0-子模型，即保持当前模型不变。我们用 $\pi^{\mathfrak{M}}$ 表示通过从 π的每个元素中去掉状态而得到的相应非带指定状态模型序列。

一个恶魔策略 是一个函数 $\Subset$，其输入为一个带指定状态的模型（$\mathfrak{M},s$），输出为一个边集 $A\subset\xrightarrow{\mathfrak{M}}$。

递减模型路径 π 与恶魔策略$\Subset$相容，当且仅当对所有 $i\in\mathbb{N}$，都有 $\Subset(\pi^{\mathfrak{M}}\ i\quad A\qquad Out\ \Subset\ \mathfrak{M}\ s\qquad\qquad(i))=A$ 意味着 $\pi^{\mathfrak{M}}(i+1)(\,)\setminus$。我们用 $(,(,))$ 表示与 $\Subset$相容且其第一分量为 $(\mathfrak{M},s)$ 的路径集合。恶魔策略 $\Subset$ 的代价为 n（在这种情形下，它称为 n-策略 $\Subset$），当且仅当每条与该策略相容的模型路径的代价均为 n。

定义 4（策略解构逻辑）。状态（φ）公式和路径（ψ）公式通过相互递归定义：

$$\varphi:=p\mid\neg\varphi\mid(\varphi\wedge\varphi)\mid\langle\text{⛧}^n\rangle\psi\qquad\psi:=\mathrm{X}\varphi\mid\varphi\,\mathrm{U}\,\varphi\mid\varphi\,\mathrm{R}\,\varphi$$

其中 $p\in\mathrm{At}$ 和 $n\in\mathbb{N}$。策略解构逻辑（SDL）的公式恰好是所有状态公式。构式 $\ ^n\psi$读作“存在一个恶魔 n-策略⛧，使得对于旅行者的所有行动，ψ都成立。”时态模态 $\mathrm{X}\varphi$读作“φ在下一步为真”；模态 $\varphi\,\mathrm{U}\,\psi$表示“$\varphi$成立，直到 ψ为真”；模态 $\varphi\,\mathrm{R}\,\psi$读作“ φ为真就解除了对 ψ为真的要求”。

我们将 有时 算子定义为 $\mathrm{F}\varphi:=\top\,\mathrm{U}\,\varphi$，将 总是 算子定义为 $\mathrm{G}\varphi:=\bot\,\mathrm{R}\,\varphi$。恶魔算子的对偶定义为$[\text{⛧}^n]\psi:=\neg\ \ ^n\neg\psi$，读作“对于所有恶魔 n-策略，⛧

旅行者都有一步操作使得 ψ 成立。”其他命题联结词（如蕴含）按通常方式定义，省略括号的约定也同样适用。我们有时将策略恶魔算子 $\langle\langle n\rangle\rangle\psi$ 中的 n 称为一个资源界。

给定一个 SDL 公式 φ，其大小记为 $|\varphi|$，即 φ 中符号的数量，其中整数以二进制编码。

定义 5（SDL 语义）。设 $(\mathfrak{M}, s)$ 是一个带指定状态的模型，φ 是一个 SDL 公式。我们通过对 φ 进行归纳来定义满足关系 $(\mathfrak{M}, s) \models \varphi$；为简洁起见，省略布尔情形：

$(\mathfrak{M}, s) \models \langle\langle n\rangle\rangle\psi$ 当且仅当存在一个 n-策略 $\mathfrak{S}$，使得对于所有 $\pi \in Out(\mathfrak{S}, (\mathfrak{M}, s))$，都有 $\pi \models \psi$

给定一个路径公式 ψ 和一条路径 π，满足关系 $\pi \models \psi$ 通过对 ψ 进行归纳来定义：

$\pi \models \mathrm{X}\varphi_1$ 当且仅当 $\pi(1) \models \varphi_1$

$\pi \models \varphi_1 \mathrm{U}\ \varphi_2$ 当且仅当存在 $j \in \mathbb{N}$，使得 $\pi(j) \models \varphi_2$ 且 $\pi(i) \models \varphi_1$ 对每个 $0 \leqslant i < j$

$\pi \models \varphi_1 \mathrm{R}\ \varphi_2$ 当且仅当以下情形之一成立：对于每个 $j \in \mathbb{N}$，$\pi(j) \models \varphi_2$；或者存在 $k \in \mathbb{N}$，使得 $\pi(k) \models \varphi_1$ 且对于所有 $0 \leqslant i \leqslant k$，$\pi(i) \models \varphi_2$

注意，我们可以使用资源为 0 的恶魔策略来定义标准的模态方框和模态菱形算子。事实上，取 $\Box\varphi := \langle\langle 0\rangle\rangle \mathrm{X}\varphi$ 和 $\Diamond\varphi := [\![0]\!]\mathrm{X}\varphi$。不难验证，方框和菱形算子具有完全符合预期的语义，因为恶魔能够采取的唯一策略就是保持模型不变。

例1。（访问控制）设图 1 中的模型 $\mathfrak{M}_1$ 描绘了一个由安全工程师管理的计算系统。系统用户可以视为旅行者，而安全工程师控制着其访问权限（即恶魔）。此外，设 *error*、*admin* 和 *server* 为原子命题。状态 s_0 表示身份验证阶段：如果用户输入错误密码，则进入状态 s_1，否则进入 s_2。节点 s_1 是失败状态，会阻止用户访问系统的任何其他状态，包括返回 s_0。状态 s_2 和 s_3 是服务器内的系统模块，其中 s_3 表示管理模块。赋值如下：$\mathrm{V}(s_0) = \emptyset$，$\mathrm{V}(s_1) = \{error\}$，$\mathrm{V}(s_2) = \{server\}$，以及 $\mathrm{V}(s_3) = \{server, admin\}$。

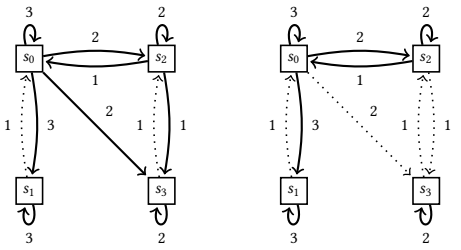

图 1：模型 $\mathfrak{M}_1$（左）和 $\mathfrak{M}_2$（右）。实线箭头表示系统中的实际转移，而虚线箭头表示代价不超过 3 的可能新增转移。

假设用户被信任可以访问系统，但不能访问管理员模块。在这种情形下，安全工程师拥有一个代价为 2 的策略，该策略移除从 s_2 到 s_3 以及从 s_0 到 s_3 的转移，并阻止从其他状态全局访问管理状态。移除转移的顺序取决于当前状态。

博弈进行两步后得到的模型 $\mathfrak{M}_2$ 如图 1 所示。因此我们可以看到，对于任意 $s \neq s_3$，$(\mathfrak{M}_1, s) \models \langle\langle 2\rangle\rangle \mathrm{G}\, admin$。

2.3 策略构造逻辑

现在我们介绍策略构造逻辑，从某种意义上说，它与 SDL 对偶。在 SDL 中，恶魔可以移除边，而在策略构造逻辑中，天使 可以策略性地添加边。

定义 6（天使策略）。给定一个带标记模型的可数无限序列 π，π 是一个代价为 n 的递增模型路径，当且仅当对于每个 $i \geqslant 0$，都有 $\pi(i) \overset{n}{\triangleright} \pi(i+1)$。与定义 3 中一样，我们用 $\pi^{\mathfrak{M}}$ 表示相应的模型序列。

天使策略是一个函数 S，它将给定的带指定状态的模型 $(\mathfrak{M}, s)$ 映射为边集 $A \subseteq ((S^{\mathfrak{M}} \times S^{\mathfrak{M}}) \setminus \rightarrow^{\mathfrak{M}})$。

递增模型路径 π 与 S 相容，当且仅当对于所有 $i \in \mathbb{N}$，都有：$\mathrm{S}(\pi(i)) = A$ 蕴含 $\pi^{\mathfrak{M}}(i+1) = \pi^{\mathfrak{M}}(i) \cup A$。如果与天使策略相容的每条递增模型路径的代价均为 n，则称天使策略 S 为 n-策略。我们用 $Out(\mathrm{S}, (\mathfrak{M}, s))$ 表示从 $(\mathfrak{M}, s)$ 开始且与 S 相容的递增模型路径集合。

定义 7（策略构造逻辑）。策略构造逻辑（SCL）的公式定义方式与定义 4 中策略解构逻辑的公式类似，不同之处在于，形式为 $\langle\langle n\rangle\rangle\psi$ 的状态公式被策略天使算子的状态公式 $\langle\langle n\rangle\rangle\psi$ 所替代。构式 $\langle\langle n\rangle\rangle\psi$ 读作“存在一个天使式的 n 策略，使得对于旅行者的所有行动，ψ 都成立”。该算子的对偶定义为 $[\![n]\!]\psi$：$[\![n]\!]\psi = \neg\langle\langle n\rangle\rangle\neg\psi$。

SCL 的语义定义方式与 SDL 的语义（定义 5）类似，唯一区别在于策略算子的情形。

定义 8（SCL 语义）。设 $(\mathfrak{M}, s)$ 是一个带指定状态的模型。$(\mathfrak{M}, s) \models \langle\langle n\rangle\rangle\psi$ 当且仅当存在 n 个策略 S，使得对于所有 $\pi \in Out(\mathrm{S}, (\mathfrak{M}, s))$，都有 $\pi \models \psi$

与 SDL 类似，我们可以在 SCL 中将标准模态方框和菱形算子定义为 $\Box\varphi := \langle\langle 0\rangle\rangle \mathrm{X}\varphi$ 和 $\Diamond\varphi := [\![0]\!]\mathrm{X}\varphi$。

例2。（访问控制，续）我们再次考虑例1中的模型 $\mathfrak{M}_1$。用户一旦进入失败状态 s_1，例如输入错误密码，就会被永久阻挡在那里。现在假设我们希望允许用户一步之内离开失败状态。添加一条从 s_1 到 s_0、代价为 1 的转移，就能让用户再次尝试进入服务器模块。应用这一天使策略后得到的模型 $\mathfrak{M}_3$ 如图2所示。请注意，天使并不能确保旅行者会离开错误状态，即 $(\mathfrak{M}_3, s_1) \not\models \langle\langle 1\rangle\rangle \mathrm{X}\neg error$。这是因为天使策略可以为旅行者创造新的可能性，却无法强迫她采取这些行动。不过，我们可以轻易验证，在模型 $\mathfrak{M}_1$ 的所有状态中，$\langle\langle 1\rangle\rangle \mathrm{G}(error \rightarrow \Diamond\neg error)$ 都成立；也就是说，天使有一种策略，可以确保旅行者每次进入失败状态后，都能在一步之内离开。事实上，天使只需恢复从 s_1 和 s_0 出发的转移，得到模型 $\mathfrak{M}_3$，即可满足 $\mathrm{G}(error \rightarrow \Diamond\neg error)$。

2.4 策略更新逻辑

在分别定义了恶魔和天使的逻辑之后，我们现在将二者结合起来，使它们能够相互合作或竞争

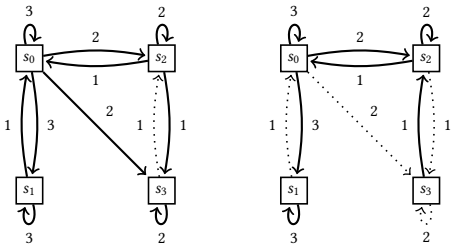

图 2：分别通过改变 $\mathfrak{M}_1$ 和 $\mathfrak{M}_2$ （图 1）得到的模型 $\mathfrak{M}_3$（左）和 $\mathfrak{M}_4$（右）。

。为此，我们从多智能体系统逻辑（例如交替时序逻辑（ATL）[11],联盟逻辑（CL）[66], 和策略逻辑（SL）[64]）中汲取灵感，并假设天使和恶魔同时执行各自的行动。我们的新模态受到 ATL 和 CL 中 $[C]\varphi$ 的启发，其含义是：“存在一个由联盟中的智能体组成的联盟策略 C，使得无论联盟外的智能体采取什么行动，φ 都成立”。在我们的情形中，拥有策略的智能体就是天使和恶魔。

定义 9（更新模型路径）。对于一个由带指定状态的模型组成的可数无限序列 π，我们称 π 为一条代价为 n 和 m 的更新模型路径，当且仅当对所有 $i \in \mathbb{N}$，都有 $\pi(i)_n, \stackrel{m}{\Rightarrow} \pi(i+1)$。给定一个天使式 n-策略 S 和一个恶魔式 m-策略 $\mathfrak{S}$，代价为 n 和 m 的更新模型路径 π 与这些策略相容，当且仅当对所有 $i \in \mathbb{N}$，都有 $\pi^{\mathfrak{M}}(i+1) = (\pi^{\mathfrak{M}}(i) \backslash \mathfrak{S}(\pi(i))) \cup \mathsf{S}(\pi(i))$。最后，$Out(\mathsf{S}, \mathfrak{S}, (\mathfrak{M}, s))$ 是从 $(\mathfrak{M}, s)$ 开始且与策略 S 和 $\mathfrak{S}$ 相容的更新模型路径集合。

定义 10（策略更新逻辑）。策略更新逻辑（SUL）的定义与 SDL 和 SCL 类似，不同之处在于，策略性恶魔或天使算子的模态被替换为公式 $[C]^{n,m}\psi$，其中 n、$m \geqslant 0$ 和 $C \subseteq \{\bigcirc, \otimes\}$。特别地，公式有四种形式：1） $C = \{\}$，我们将写作 $[\otimes^m]\psi$，表示“存在一个恶魔式 n-策略，使得对于所有天使式 m-策略以及旅行者的所有行动，ψ 成立”；2） $C = \{\bigcirc\}$，我们将写作 $[\bigcirc^n]^m\psi$，其含义与前一种情况相同，只是恶魔策略和天使策略互换；3） $C = \{\bigcirc, \otimes\}$，我们将写作 $[\bigcirc^n, \otimes^m]\psi$，表示“存在一个天使式 n-策略和一个恶魔式 m-策略，使得对于旅行者的所有行动，ψ 成立”；最后，4） $C = \emptyset$，记作 $[\emptyset]^{n,m}\psi$，其含义与前一种情况相同，只是将存在量词换成全称量词。按照惯例，我们将组合策略算子的对偶记作 $[C]^{n,m}\psi := \neg[C]^{n,m}\neg\psi$。

定义 11（SUL 语义）。设 $(\mathfrak{M}, s)$ 是一个带指定状态的模型。我们仅给出 $[\bigcirc^n]^m\psi$ 和 $[\bigcirc^n, \otimes^m]\psi$ 的情形，而 $[\otimes^m]^n$ 和 $[\emptyset]^{n,m}\psi$ 的语义则可通过交换恶魔策略和天使策略以及量词来类似地定义。

$(\mathfrak{M}, s) \models [\bigcirc^n]^m\psi$ 当且仅当存在一个 n-策略 S，使得对于所有 m-策略 $\mathfrak{S}$ 以及所有 $\pi \in Out(\mathsf{S}, \mathfrak{S}, (\mathfrak{M}, s))$，都有 $\pi \models \psi$

$(\mathfrak{M}, s) \models [\bigcirc^n, \otimes^m]\psi$ 当且仅当存在一个 n-策略 S，并且存在一个 m-策略 $\mathfrak{S}$，使得对于所有 $\pi \in Out(\mathsf{S}, \mathfrak{S}, (\mathfrak{M}, s))$，都有 $\pi \models \psi$

示例 3.（访问控制，续）天使与恶魔的合作能够实现对系统进行动态、同步的访问控制。考虑图 1 中的模型 $\mathfrak{M}_2$。安全工程师（即恶魔）独自无法让用户离开管理员模块，也就是说，$(\mathfrak{M}_2, s_3) \models [\otimes^2]^2\mathsf{F}\neg admin$，因为天使可以采取不修改模型的策略。然而，恶魔与天使合作时便能实现这一目标。具体来说，第一步中，天使创建转移 $s_3 \to s_2$，恶魔则选择一个 0-子模型（即什么也不做）；在博弈的下一步中，恶魔移除 s_3 处的自环。这一策略得到图 2 中的模型 $\mathfrak{M}_4$。因此，我们有 $(\mathfrak{M}_2, s_3) \models [\bigcirc^2, \otimes^2]\mathsf{F}\neg admin$，因为模型更新为 $(\mathfrak{M}_4, s_3)$ 后，用户唯一能做的就是转移到状态 s_2。

例如，天使可以在（不）与恶意攻击者合作，从而进入未授权状态，给系统造成漏洞。例如，$(\mathfrak{M}_2, s_1) \models [\otimes^2]^2\mathsf{G}\neg admin$。事实上，为了阻止在天使协助下的攻击者从 s_1 到达状态 s_3，恶魔可以移除转移 $s_0 \to s_2$，与此同时，天使搭建一座桥梁 $s_1 \to s_0$。现在，如果攻击者处于状态 s_0，天使可以例如添加一座桥梁 $s_0 \to s_3$；无论恶魔同时采取什么行动，攻击者都能访问管理员状态。不过，如果天使拥有的资源较少，安全工程师就能阻止这次攻击。我们可以轻松验证，$(\mathfrak{M}_2, s_1) \models [\otimes^2]^1\mathsf{G}\neg admin$。事实上，恶魔可以移除从 s_0 到 s_2 的转移；无论天使如何使用 1 个资源，她都无法恢复一条通往状态 s_3 的路径。

3 表达力

本节将我们的新逻辑相互比较，并将它们与文献中已有的逻辑进行比较。

定义 12（表达力）。设 L_1 和 L_2 是两种语言，且 $\varphi \in \mathrm{L}_1$ 和 $\psi \in \mathrm{L}_2$。如果对于所有模型 $(\mathfrak{M}, s)$，都有：$(\mathfrak{M}, s) \models \varphi$ 当且仅当 $(\mathfrak{M}, s) \models \psi$，则称 φ 和 ψ 等价。

如果对于每个 $\varphi \in \mathrm{L}_1$，都存在一个与之等价的 $\psi \in \mathrm{L}_2$，我们记作 $\mathrm{L}_1 \preccurlyeq \mathrm{L}_2$，并称 L_2 至少与 L_1 同样具有表达力。若 $\mathrm{L}_1 \preccurlyeq \mathrm{L}_2$ 当且仅当 $\mathrm{L}_1 \preccurlyeq \mathrm{L}_2$ 且 $\mathrm{L}_2 \not\preccurlyeq \mathrm{L}_1$，我们记作 $\mathrm{L}_1 \prec \mathrm{L}_2$，并称 L_2 严格比 L_1 更具表达力。最后，若 $\mathrm{L}_1 \not\preccurlyeq \mathrm{L}_2$ 且 $\mathrm{L}_2 \not\preccurlyeq \mathrm{L}_1$，则称 L_1 和 L_2 不可比较，并记作 $\mathrm{L}_1 \nsim \mathrm{L}_2$。

计算树逻辑。我们首先证明，SDL、SCL 和 SUL 的表达力都严格强于经典的计算树逻辑(CTL) [33]。为了证明我们的逻辑至少同样具有 CTL 的表达力，我们借鉴关于 OL [28], 的论证，指出 CTL 是 SDL、SCL 和 SUL 的片段。为了说明我们的逻辑能够表达而 CTL 无法表达的性质，我们使用模型变化。

定理 1. CTL $\prec$ SDL、CTL $\prec$ SCL 和 CTL $\prec$ SUL。

证明。要看出 CTL 是我们这些逻辑的一个片段，只需定义一个从 CTL 到我们这些逻辑的保真翻译函数 t。布尔情形显而易见，故略去。现在，为了处理 CTL路径量词，我们可以在 0 个资源上采用恶魔策略和天使策略。具体而言，$t(\mathrm{AX}\varphi) = \blacktriangledown \mathrm{X}\, t(\varphi)$，以及 $t(\mathrm{A}(\varphi \mathrm{U} \psi)) = \blacktriangledown(t(\varphi)\, \mathrm{U}\, t(\psi))$，其中 $\blacktriangledown \in \{\Diamond, \Box, [\bigcirc^0, \otimes^0]\}$，具体取决于所讨论的逻辑。根据语义的定义可立即得知，这种递归翻译是保真的。

现在，对于 SDL、SCL 和 SUL，我们证明存在一些它们能够区分而 CTL 无法区分的模型。首先从 SDL 开始。考虑一个 SDL公式 $\langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, p$，并反设存在一个与之等价的 φ CTL公式。然后，考虑图3所示的模型 $\mathfrak{M}_1$ 和 $\mathfrak{M}_2$。

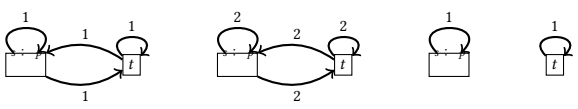

图3：模型 $\mathfrak{M}_1$（左）、$\mathfrak{M}_2$（中）和 $\mathfrak{M}_3$（右）。原子命题 p 在状态 s 中为真。$\mathfrak{M}_3$ 中的所有可能边的代价均为1。

这两个模型同构，唯一的区别是，$\mathfrak{M}_1$ 中每条边的代价均为1，而 $\mathfrak{M}_2$ 中每条边的代价均为2。由于在 CTL 中无法访问边的代价，因此 $(\mathfrak{M}_1, s) \models \varphi$ 当且仅当 $(\mathfrak{M}_2, s) \models \varphi$。

同样容易验证，$(\mathfrak{M}_1, s) \models \langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, p$。然而实上，恶魔可以移除边 $s \to t$，因此旅行者永远无法到达状态 t，在那里 p 为假。此外，显然 $(\mathfrak{M}_2, s) \not\models \langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, \neg p$，因为模型中每条边的代价都是2，恶魔无法对模型进行任何修改。

现在转向 SCL 和 SUL，并考虑以下两个公式：SCL 的 $\langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, [\bigcirc^0] \mathrm{X}\, p$，以及 SUL 的 $[\bigcirc^1]^0 \mathrm{X}\, [0, 0] \mathrm{X}\, \neg p$。这些公式意味着，天使可以添加一条箭头，使得在新的更新后的模型中，旅行者能够到达一个 $\neg p$-状态。反设对于这两个公式中的每一个，都存在某个与之等价的 CTL 公式 φ。取图3中的模型 $\mathfrak{M}_3$，以及模型 $\mathfrak{M}_4$；后者与 $\mathfrak{M}_3$ 类似，但包含单个状态 s。

模型 $\mathfrak{M}_4$ 仅由一个状态 s 和一条自反箭头构成。模型 $\mathfrak{M}_3$ 则由状态 s 和状态 t 构成，二者都只有自反边。显然，$(\mathfrak{M}_3, s) \models \varphi$ 当且仅当 $(\mathfrak{M}_4, s) \models \varphi$。

同时，我们有 $(\mathfrak{M}_4, s) \not\models \langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, [\bigcirc^0] \mathrm{X}\, \neg p$，而且 $(\mathfrak{M}_3, s) \not\models [\bigcirc^1]^0 \mathrm{X}\, [0, 0] \mathrm{X}\, \neg p$，因为在模型 $\mathfrak{M}_4$ 中根本没有满足 $\neg p$ 的状态。另一方面，$(\mathfrak{M}_3, s) \models \langle\!\langle 1 \rangle\!\rangle \mathrm{X}\, [\bigcirc^0] \mathrm{X}\, \neg p$，因为天使可以添加边 $s \to t$，使旅行者能够访问 $\neg p$-状态 t。特别是，天使添加边 $s \to t$ 后，我们会检查旅行者可以采取的所有下一时刻路径。她可以留在 s，也可以移动到 t。两种情况下，$[\bigcirc^0] \mathrm{X}\, \neg p$ 都成立，因为天使没有资源后便无法再修改模型，而且旅行者存在一条下一时刻路径可以到达 $\neg p$-状态 t。通过类似的推理可以看出，$(\mathfrak{M}_3, s) \models [\bigcirc^1]^0 \mathrm{X}\, [0, 0] \mathrm{X}\, \neg p$。□

天使与恶魔。 我们表明，当这些新提出的天使与恶魔逻辑彼此较量时，它们确实各不相同。具体而言，我们认为 SDL 和 SCL 不可比较，而且 SUL 严格比二者都更具表达力。

定理 2. SDL $\not\asymp$ SCL、SDL $\prec$ SUL，且 SCL $\prec$ SUL。

证明。我们可以沿用定理 1 证明中的论证。要说明 SDL $\not\asymp$ SCL，我们回想一下，模型 $\mathfrak{M}_1$ 和 $\mathfrak{M}_2$ 可由一个 SDL 公式区分。没有任何 SCL 公式能够区分这两个模型，这是因为它们的差异仅在于边的成本；而且，由于两个模型中的关系都是全关系，天使无法修改模型。

为了证明 SCL $\not\asymp$ SDL，我们回顾定理 1 的证明中提到的模型 $\mathfrak{M}_3$ 和 $\mathfrak{M}_4$，它们可以由一个 SCL 公式区分。根据恶魔算子的语义，显然不存在 SDL 公式能够区分 $(\mathfrak{M}_3, s)$ 和 $(\mathfrak{M}_4, s)$（恶魔无法移除更多边）。因此，SDL $\not\asymp$ SCL。

最后，注意到 SDL 和 SCL 都是 SUL 的片段，可通过翻译 $t(\langle\!\langle n \rangle\!\rangle \varphi) = [\bigcirc^0, \otimes^n] t(\varphi)$ 和 $t([\![n]\!] \varphi) = [\bigcirc^n, \otimes^0] t(\varphi)$ 得到；因此 SDL $\preccurlyeq$ SUL，且 SCL $\preccurlyeq$ SUL。由此我们还可得，若 SCL $\not\asymp$ SDL，则 SUL $\not\asymp$ SDL；若 SDL $\not\asymp$ SCL，则 SUL $\not\asymp$ SCL。综上，我们得出 SDL $\prec$ SUL，且 SCL $\prec$ SUL。□

与障碍逻辑的关系。 障碍逻辑（OL）中策略算子 $\dagger^n \psi$ 和 $[\dagger^n] \psi$ 的语义，与 SDL 模态 $\langle\!\langle n \rangle\!\rangle \psi$ 和 $[\![n]\!] \psi$ 的语义相似，关键区别在于：恶魔禁用某些边、旅行者采取行动后，OL 中的边会恢复。因此，在 OL 中，给定模型中的变化并非永久性的。然而，在定理 2 中证明 SDL $\not\asymp$ SCL 时，我们只在 SDL 公式中使用了下一时刻时态模态。因此，我们可以使用相同的论证和相同的一对模型 $\mathfrak{M}_1$ 和 $\mathfrak{M}_2$，来证明 OL $\not\asymp$ SCL。反过来，即 SCL $\not\asymp$ OL，论证与 SCL $\not\asymp$ SDL 的论证类似，后者使用模型 $\mathfrak{M}_3$ 和 $\mathfrak{M}_4$。同样的论证还表明，SUL $\not\asymp$ OL。

定理 3. OL $\not\asymp$ SCL 和 SUL $\not\asymp$ OL。

OL 与 SDL 之间的关系更加微妙。在证明 SDL $\not\asymp$ OL 时，我们利用了 SDL 中的模型改变是永久性的这一事实，因此可以“记住”被移除的边；而在 OL 中这是不可能的，因为每次恶魔采取新行动之前，边都会恢复。

定理 4. SDL $\not\asymp$ OL。

证明。我们给出证明的思路，完整证明见技术附录。考虑一个 SDL 公式 $\langle\!\langle 1 \rangle\!\rangle \mathrm{F}\, p$，并反设存在一个等价的 OL 公式 φ。由于 OL 公式是有限的，且每个 OL 模态都有有限的资源界，我们可以假设 n 是 φ 中出现的最大的 $n \in \mathbb{N}$。现在考虑图 4 中的模型 $\mathfrak{M}_{n+2}$ 和 $\mathfrak{M}_{n+3}$。这些模型十分相似，唯一的区别是 $\mathfrak{M}_{n+2}$ 少了一个 t-状态。对于这两个模型，p 仅在状态 t_1 中为真。

为了证明 $(\mathfrak{M}_{n+2}, s_1) \models \langle\!\langle 1 \rangle\!\rangle \mathrm{F}\, p$ 和 $(\mathfrak{M}_{n+3}, s_1) \not\models \langle\!\langle 1 \rangle\!\rangle \mathrm{F}\, p$，我们这样论证：当旅行者沿着 s-状态移动时，在 $\mathfrak{M}_{n+2}$ 中的每一步，恶魔都可以移除一条带有 $i > 1$ 的 $s_{n+1} \to t_i$ 边。逐步移除这些边后，就会出现这样的情况：在状态 s_{n+1}，旅行者只剩下一个选择，即进入状态 t_1，从而满足 $\mathrm{F}p$。在 $\mathfrak{M}_{n+3}$ 中，我们多了一个 t-状态，因此恶魔没有足够的步骤来确保只有

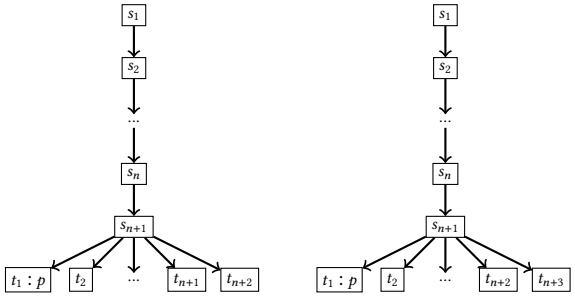

图 4：模型 $\mathfrak{M}_{n+2}$（左）和 $\mathfrak{M}_{n+3}$（右），省略了带有 t-状态的自反箭头。所有边的代价均为 1。

可以进入 p-状态 t_1。因此，旅行者可以进入一个不满足 p 的 t-状态，从而违反 Fp。接着我们论证，φ 无法区分这些模型，因为在 OL 中，边会在每个游戏步骤后恢复，因此恶魔应当在其中一个模型的状态 s_{n+1} 采取行动，而这一行动无法在另一个模型中复现。这是不可能的，因为资源界 n 过低。□

表达能力格局。 图5呈现了表达能力结果的概览。

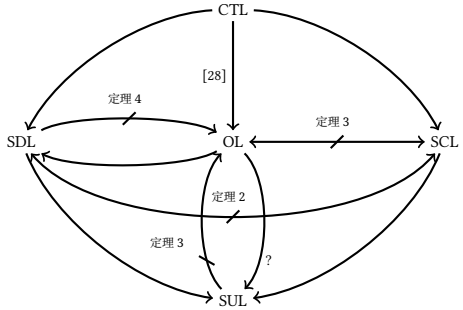

图5：表达能力结果。从$\mathcal{L}_1$到$\mathcal{L}_2$的箭头表示$\mathcal{L}_1 \prec \mathcal{L}_2$。从$\mathcal{L}_1$到$\mathcal{L}_2$的删除线箭头表示$\mathcal{L}_1 \not\prec \mathcal{L}_2$。开放问题用问号表示。

关于 OL $\not\prec$ SDL 和 OL $\not\prec$ SUL 是否成立的问题，我们留待今后研究，并猜想它们确实成立。最后，我们还想指出 OL 与 SUL 之间一个有趣的事实。给定 OL 的 $(\mathfrak{M}, s)$ 和 φ，如果 $(\mathfrak{M}, s) \models \varphi$，那么存在 SUL 的一个公式 ψ，使得 $(\mathfrak{M}, s) \models \psi$。论证直接基于这两种逻辑的语义，例如，$(\mathfrak{M}, s) \models \dagger^n \chi$意味着 $(\mathfrak{M}, s) \models [\bigcirc^n, \textcircled{\scriptsize$\pm$}^n]\chi$。后者意味着，对于 OL 中的每一种恶魔策略，在相同资源界限下，SUL 中都存在一种恶魔与天使联合策略，能够到达同一目标。直观地说，为了对 OL 算子建模，天使和恶魔可以合作：恶魔移除边，而天使在下一回合恢复全部或部分边。

4 模型检测

本节研究 SDL、SCL 和 SUL 的模型检验问题。特别地，我们将证明，对于前两种逻辑，该问题是 PSPACE 完全的；而对于 SUL，该问题属于 EXPSPACE。我们还将提到，SUL 的下一时刻片段的模型检验问题是 PSPACE 完全的。

定义13（模型检测）。给定一个带指定状态的模型（$\mathfrak{M}$，s）和一个公式 φ，模型检验问题 就是计算（$\mathfrak{M}$，s）是否 $\models \varphi$。

必要时，我们假设 φ 处于否定范式（NNF），这意味着在 φ中，否定只出现在原子命题前。显然，SDL、SCL 和 SUL 中的每个公式 φ 都可以利用命题等价式、对偶式（例如 SDL 情形中的 $[\textcircled{\scriptsize$\pm$}^n]\psi$）以及时序等价式

$$\neg \varphi \leftrightarrow \neg\varphi \quad \neg\varphi_1 \quad \varphi_2 \leftrightarrow \neg\varphi_1 \quad \neg\varphi_2 \qquad \neg\varphi_1 \quad \varphi_2 \leftrightarrow \neg\varphi_1 \quad \neg\varphi_2 \qquad (\mathrm{X})$$

X，（U）R，以及（R）U，等价地改写为 φ' 的否定范式。否定范式公式 φ'的大小至多与原始公式 φ的大小呈线性关系。

定理 5. SDL 的模型检测是 PSPACE 完全的。

证明。我们先证明下界，并通过从 PSPACE 完全的量化布尔公式 (QBF) 问题进行归约来实现。给定一个 QBF 实例 $\Psi := Q_1 p_1 \ldots Q_n p_n \psi(p_1, \ldots\ldots, p_n)$，其中 $Q_i \in \{\exists, \forall\}$，该问题要求判定 Ψ 是否为真。不失一般性，我们假设 Ψ 中没有自由变量，且每个变量只用于一次量化。

对于给定的 QBF 实例 Ψ，我们构造一个模型 $(\mathfrak{M}^\Psi, s)$ 和一个 SDL 公式 φ，使得 $(\mathfrak{M}^\Psi, s) \models \varphi$ 当且仅当 Ψ 为真。先从模型入手，考虑 $\mathfrak{M}^\Psi = (S, \to, \mathrm{V}, \mathrm{C})$，其中 $S = \{s, s_1, \ldots, s_{2n}\}$，$s \to s$，$s \to s_i$，并且对于所有 $s_i \in S$，$\mathrm{V}(p_i^1) = \{s_i\}$，而 $\mathrm{V}(p_i^0) = \{s_{n+i}\}$（对于 $1 \leqslant i \leqslant n$），$\mathrm{C}(s, s) = 2$，其他任意状态对的值均为 1。直观地说，对于 Ψ 中的每个 p_i，我们都有两个状态，其对应的原子命题为 p_i^1 和 p_i^0，用于表示 p_i 被设为真还是假。含有两个变量 p_1 和 p_2 的 QBF 实例 Ψ 所对应的模型如图 6 所示。

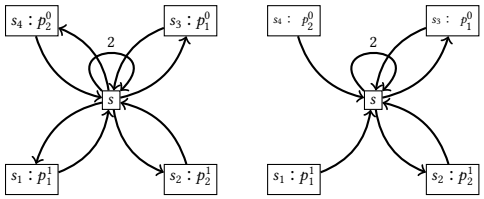

图 6：模型 $\mathfrak{M}^\Psi$（左）及其一次更新（右）。除具有代价 2 的 $s \to s$ 边外，所有边的代价均为 1。

现在，我们将构造 SDL 中对应的公式 φ。首先，回顾一下，n 是 QBF 实例 Ψ中的原子命题数量。然后，我们定义公式 $chosen_k$，以表示仅设置了 Ψ 中前 k 个原子命题（共 n 个）的真值这一事实：

$$chosen_k = \bigwedge_{1 \leqslant i \leqslant k} (\Diamond\Diamond p_i^0 \leftrightarrow \neg\Diamond\Diamond p_i^1) \wedge \bigwedge_{k < i \leqslant n} (\Diamond\Diamond p_i^0 \wedge \Diamond\Diamond p_i^1).$$

回想一下，$\Diamond\varphi := [\textcircled{\scriptsize$\pm$}^0]\mathrm{X}\varphi$。直观地说，若恶魔切断了通往前 k 个原子命题 p_i^0和 p_i^1中某些原子命题的转移，并且切断的方式保证：如果通往 p_i^0的转移被移除，那么就必须还保留一条通往 p_i^1的转移，那么 $chosen_k$ 就成立。这模拟了在 Ψ中对原子命题 p_i 真值的无歧义选择。前 k 个以外的所有其他原子命题仍应是可访问的。公式中使用双重菱形算子，是因为恶魔的每个动作之后，旅行者都会立即移动

。如果旅行者决定前往某个可访问的s_l状态，我们仍应能够在两步之内验证某个$p_i^{j\in\{0,1\}}$是否可访问，并在途中经过 s。

现在，我们已经准备好着手构造 φ。

$$\varphi_0 := \psi(\Diamond\Diamond p_1^1, ..., \Diamond\Diamond p_n^1)$$

$$\varphi_k := \begin{cases} [\textcircled{\scriptsize 1}]\mathrm{X}\,(chosen_k \rightarrow \varphi_{k-1}) & \text{if } Q_k = \forall \\ \langle\textcircled{\scriptsize 1}\rangle\mathrm{X}\,(chosen_k \wedge \varphi_{k-1}) & \text{if } Q_k = \exists \end{cases}$$

$$\varphi := \varphi_n.$$

接下来要证明的是，当且仅当$(\mathfrak{M}^\Psi, s) \models \varphi$时，$\Psi := Q_1 p_1 ... Q_n p_n \psi(p_1, ..., p_n)$为真。首先注意，我们考虑代价至多为 1 的恶魔策略，即在每一步中，恶魔都可以从 s移除至多一条通往某个 s_i[2]的边。移除一条通往 p_i^0的转移，意味着将 p_i的真值设为真。例如，见图 6 右侧的模型，其中 p_1 的值被设为假，而 p_2 被设为真。守卫 $chosen_k$ 确保命题的真值选择明确无误。因此，连同守卫一起，翻译子句 φ_k 中的构造 $[\textcircled{\scriptsize 1}]\mathrm{X}$和 $\langle\textcircled{\scriptsize 1}\rangle\mathrm{X}$模拟了量词 $\forall$ 和 $\exists$。这样设定所有原子命题 p_i 的真值后，量化布尔公式的求值就对应于相应原子命题的可达性。特别地，在对 Ψ 求值时，当且仅当p_i^1在相应的子模型中可达，我们才将 p_i 设为真 。

为了证明 SDL 的模型检验问题属于 PSPACE，我们给出交替算法 1。设 $(\mathfrak{M}, s)$ 是一个有限模型，φ 是 SDL 的一个公式。不失一般性，我们假设公式 φ 处于否定范式。为简洁起见，算法中只展示情形 $\langle\textcircled{\scriptsize 1}\rangle^n\mathrm{X}\,\psi$ 和 $\langle\textcircled{\scriptsize 1}\rangle^n\psi_1\mathrm{U}\,\psi_2$，完整算法见技术附录。

算法 1 用于 SDL 模型检测的算法

过程 MC($(\mathfrak{M}, s)$ φ)
  情形 $\varphi = \langle\textcircled{\scriptsize 1}\rangle\mathrm{X}\,\psi$
    存在性地选择 n-子模型 $\mathfrak{M}'$
    全称地选择 s' 使得 $s \xrightarrow{\mathfrak{M}'} s'$
    返回 模型检查($(\mathfrak{M}', s')$ ψ)
  情形 $\varphi = \langle\textcircled{\scriptsize 1}\rangle^n\psi_1\,\mathrm{U}\,\psi_2$
    $X \leftarrow (\mathfrak{M}, s)$
    $i \leftarrow 0$
    当.....时 非 模型检查(X, ψ_2) 且 $i \leqslant brDepth$ **执行**
      如果 非模型检查(X, ψ_1) **则**
        返回 假
      存在地选择 n-子模型 $\mathfrak{M}'$ 的 X
      全称地选择 s' 使得 $s \xrightarrow{\mathfrak{M}'} s'$
      $X \leftarrow (\mathfrak{M}', s')$
      $i \leftarrow i + 1$
    结束当.....时
    如果 $i > brDepth$ **那么**
      返回 假
    else
      返回 真
结束过程

在该算法中，我们使用全称选择和存在选择，以深度优先的方式遍历计算树。该算法

[2]请记住，根据恶魔策略的定义，如果某个状态只有一条出边，恶魔就不能移除这条边。

该算法紧密遵循语义的定义，其正确性可通过对 φ 进行归纳法证明。算法能够终止，是因为每种情形都会将一个子公式分解为更简单的子公式，而且树的深度有限。

每个分支的深度 $brDepth$ 至多为 $O(|\rightarrow|\cdot|S|)$，也就是说，恶魔至多有 $|\rightarrow|+1$ 种方式连续修改模型（最坏情况下，逐条移除边，另加一种不修改模型的选择），且每个这样的 n 子模型至多有 $|S|$ 个旅行者可以转移到的状态。换言之，对于每个 n 子模型，只需检查该子模型中至多 $|S|$ 个状态。因此，给定分支上的唯一博弈局势总数，从而树中每个分支的深度，都有上界 $O(|\rightarrow|\cdot|S|) \leqslant O(|\mathfrak{M}|^2)$，并且可以由我们的交替算法在多项式时间内遍历。由于需要考虑的子公式至多有 $|\varphi|$ 个，算法的总运行时间有上界 $O(|\varphi|\cdot|\mathfrak{M}|^2)$。由 APTIME（即交替多项式时间）= PSPACE [31],这一事实，我们得出 SDL 的模型检验问题是 PSPACE 完全的。□

熟悉相关内容的读者可能会指出，尽管 SDL 的模型检测是 PSPACE 完全的，但对其思想先驱 OL 进行模型检测可以在多项式时间内完成 [28]。这是因为两种逻辑的语义存在显著差异。在 OL 中，旅行者移动后，边会恢复；而在 SDL 中，边删除是永久性的，也就是说，为了验证恶魔的策略能力，我们必须在内存中保留多项式数量的修改后模型。不过请注意，我们关于 PSPACE 完全性的结果与那些允许对永久性 模型改变进行量化的逻辑的模型检测结果一致（例如，参见 [36, 43, 50, 61, 76]）。

通过一个相对类似的论证，也可以证明 SCL 的模型检测是 PSPACE 完全的；为简洁起见，这里省略论证，证明概要见《技术附录》。

定理 6。 SCL 的模型检测是 PSPACE 完全的。

最后，我们考察 SUL 的模型检测问题的复杂度，并证明其属于 EXPSPACE。不过，我们还注意到，SUL 的下一时刻片段仍然是 PSPACE 完全的。

定理 7。 SUL 的模型检测属于 EXPSPACE，而 SUL 的下一时刻片段的模型检测是 PSPACE 完全的。

证明。为证明 SUL 的模型检测属于 EXPSPACE，我们可以提供一种类似于算法 1 的交替算法。这里我们给出一个总体思路，完整论证及算法见技术附录。

与算法 1 的主要区别在于，对于涉及 $[C]^{n,\,m}$ 的情形，我们必须根据 C 考虑 n-m-更新。每个 n-m-更新的大小均有 $O(|\mathfrak{M}|^2)$ 的上界，并且可以在多项式时间内计算。对于 SUL 的下一时刻片段，我们至多需要检查 $|\varphi|$ 个此类更新，因此总运行时间属于 APTIME = PSPACE。下界则由以下事实得出：SUL 包含 SDL 和 SCL。

对于完整语言，与算法 1 类似，我们构造一棵分支深度受限于 $brDepth$的计算树。不过，这一次，$brDepth$ 是指数级的，因为在最坏情况下，恶魔和天使联手可以迫使完全图的大小为 $|S|^2$ 的子模型数量呈指数增长，也就是说，$brDepth$ 受限于 $O(2^{|\mathfrak{M}|^2})$，因此该算法的运行时间为

指数时间。由于 AEXPTIME=EXPSPACE， [31], SUL 的模型检测属于 EXPSPACE。□

5 相关工作

障碍逻辑。 我们的研究最初受到关于障碍逻辑 (OL)的研究启发 [28],: 在这种逻辑中，人们可以推理恶魔的策略，即停用某些边，使目标性质无论旅行者如何行动都成立。此类博弈的每一步结束后，所有边都会恢复；这与我们的 SDL 设定不同，在后者中，我们假设边一经停用便永久停用。OL 的扩展包括障碍 ATL [29],、定时 OL [57], 和联盟 OL [30]。所有这些扩展都只涉及临时停用边。与 OL 的大量研究相比，我们还引入了一种形式体系 SCL，用于推理边的策略性激活 ； 此外，我们也在 SUL 中研究恶魔与天使在合作和竞争情境下的相互作用。

破坏类逻辑。 OL 本身的研究受到破坏博弈的启发 [72], : 在这类博弈中，恶魔可以停用任意一条边；与之对应的还有破坏模态逻辑 (SML) [15, 61] ，以及将其推广到边的子集的形式 [27]。与 SML 相关的是一种可定义的边删除方法 [58]；在 SML 中，恶魔可以移除模型中的任意一条边。同样，也有一些逻辑用于推理向模型中添加一条边、交换两条边、复制并移除边等操作（例如，参见 [12–14]）。与所有这些方法相比，我们考虑加权图，以及天使和恶魔的扩展策略，而不是所引文献中的下一时刻结果。

动态认知逻辑。 在动态认知逻辑 （DEL）中，添加和删除箭头尤其是如此，而模型更新总体而言更是基础操作。 [75], 这类逻辑建立在刻画智能体知识的认知模型之上，其中模型的更新对应于各种改变信息的事件。关于边删除，可以提及箭头更新 [55, 76],，以及智能体彼此分享知识的各种模式[3, 17, 50]。也可以将添加箭头视为特定智能体进行内省的一种努力 [43]。此外，已有一些工作致力于将策略推理纳入 DEL，特别是将其作为认知模型上的可达性博弈 [62, 63], 、并发公共通信 [2, 44], 以及交替时间动态认知逻辑 [36]。尽管我们可以借鉴 DEL 中关于模型更新的一些直觉，但不言而喻，我们的设定不同于智能体知识与学习的设定。

社交网络动态逻辑。受 DEL 启发，已有大量研究关注社交网络（SNs）中的动态（概述见 [67] ）。在社交网络的情境中，添加或移除边已被用于对多种变化进行建模，例如友谊关系的变化 [16, 51], 、关注或取消关注其他智能体 [77], 、在社交网络中获取知识 [32], 、社交网络中帖子的可见性 [49], 、网络平衡 [74], , 以及扩散拍卖中卖家的策略 [48]。

策略推理。 在策略推理领域中，除 OL 情形外，我们主要在规范推理中看到模型修改：给定一个多智能体系统，社会法则（或规范）会将给定智能体在给定状态下的行动集合划分为期望的（或允许的）行动，以及非期望的（或禁止的）行动。这通常通过从模型中移除某些转移来实现（例如，参见 [1, 4, 7, 10, 24, 47, 54]）。近期，文献 [46] 提出了一种修改策略性多智能体模型的通用方法。添加和移除边也被解释为赋予或剥夺智能体能力 [45]。还可以提及有关模块检查的研究 [53, 56], , 其中会从执行树中剪去某些转移，以此对环境的各种行为进行建模。分离逻辑中也有一些突出的模型动态类型（例如，参见 [18, 37, 38, 69]），这类逻辑允许对计算机程序的执行进行推理。据我们所知，这些方法都没有针对能够修改底层模型拓扑的智能体，对其扩展时序目标进行推理。

资源逻辑。 在认知逻辑的背景下，研究者利用资源和成本来解决逻辑全知问题（例如，见 [5, 6, 41, 42]），将其作为动态认知行动的约束 [35, 40, 71], , 以及用于认知规划 [20, 23],, 这些只是其中几种应用。在策略推理的背景下，针对 CL 和 ATL 情境中的资源受限智能体，已有大量研究（例如，见 [8, 9, 19, 25, 26, 39, 65]）。请注意，在后一种情形中，模型是静态的，而我们的模型是动态的，因此两种情境截然不同。

图上的博弈。 最后，破坏博弈并非唯一涉及旅行者、且使用模态逻辑工具进行分析的图上博弈（概述见 [73] ）。此类分析还包括捉迷藏逻辑、 [60], 警察与强盗博弈 [59], 以及投毒博弈 [52, 78]。

6 讨论

我们提出了三种新颖的逻辑，用于推理能够修改给定模型的智能体的策略。在 SDL 中，恶魔采用删边策略，使得无论旅行者选择哪条路径，目标条件都成立。类似地，为了推理天使的加边策略，我们提出了 SCL。最后，为了以类似 ATL 的方式表达恶魔与天使之间的互动，我们引入了 SUL。对于所有这些逻辑，我们都研究了其表达力，并提供了模型检验算法。

由于我们对智能体策略性模型变化的研究才刚刚起步，未来还有大量工作可做。首先，我们希望研究这三种逻辑的可满足性问题。此外，请回想一下，我们以无记忆方式定义了恶魔策略和天使策略，即智能体的行动取决于当前的带指定状态的模型。我们还希望研究它们的完美回忆策略；在这类策略中，行动取决于由带指定状态的模型构成的历史。我们推测，对于这两类策略，所有逻辑的语义都是等价的，这可以通过类似于 OL [28] 中论证的论证来证明。

定义 SDL 时，我们受到了 OL 的启发。不过，引入加边策略后，我们很想将 OL 的一个变体形式化并加以研究：在这个变体中，不再由恶魔行动，而是由天使仅在一个回合内添加边。此外，与 [30], 相关，我们觉得将 SUL 扩展到天使与恶魔联盟尤其引人入胜。最后，我们还希望考虑沿着 [70] 的思路，为我们的逻辑引入最大与最小不动点扩展。

参考文献

[1] Thomas Ågotnes、Wiebe van der Hoek 和 Michael J. Wooldridge。2010。稳健规范系统与规范遵从逻辑。*IGPL*逻辑期刊 18，1（2010），4–30，101093 070（），–。https://doi.org/ . /jigpal/jzp

[2] Thomas Ågotnes 和 Hans van Ditmarsch。2008。联盟与公告。载于第7届 AAMAS 会议论文集，Lin Padgham、David C. Parkes、Jörg P. Müller 和 Simon Parsons（编）。IFAAMAS，673–680。[3] Thomas Ågotnes 和 Yì N. Wáng。2017。解决分布式知识。《人工智能》 252 2017 1 21 10 1016 2017 07 002 （），–。[4] https://doi.org/ . /J.ARTINT. . . Natasha Alechina、Giuseppe De Giacomo、Brian Logan 和 Giuseppe Perelli。2022。多智能体系统动态规范的自动综合。载于第19届 KR 会议论文集，Gabriele Kern-Isberner、Gerhard Lakemeyer 和 Thomas Meyer（编）。[5] Natasha Alechina 和 Brian Logan。2002。向资源受限智能体归属信念。载于第1届 AAMAS 会议论文集。ACM，881–888。https://doi.org/10.1145/ 544862.544948[6] Natasha Alechina 和 Brian Logan。2009。情境化资源受限智能体逻辑。《逻辑、语言与信息期刊》18，1（2009），79–95。https: //doi.org/ 10 1007 10849 008 9073 6 。 /S - - - Natasha Alechina、Brian Logan 和 Mehdi Dastani。2018。多智能体系统中的规范描述与验证。FLAP 5，2（2018），457–490。[8] Natasha Alechina、Brian Logan、Nguyen Hoang Nga 和 Abdur Rakib。2010。资源受限交替时序逻辑。载于第9届 AAMAS 会议论文集，Wiebe van der Hoek、Gal A. Kaminka、Yves Lespérance、Michael Luck 和 Sandip Sen（编）。IFAAMAS，481–488。[9] Natasha Alechina、Brian Logan、Nguyen Hoang Nga 和 Abdur Rakib。2011。有界资源联盟逻辑。《逻辑与计算杂志》 21, 6 2011 907 937 10 1093 032 （），–。https://doi.org/ . /LOGCOM/EXQ Natasha Alechina、Brian Logan 和 Giuseppe Perelli。2025。综合最小代价动态规范。载于第34届 IJCAI 2025 会议论文集。ijcai.org，3–11。https://doi.org/ 10 24963 2025 1 。 /IJCAI. / Rajeev Alur、Thomas A. Henzinger 和 Orna Kupferman。2002。交替时态逻辑。《ACM期刊》 49 2002 672 713 10 1145 585265 585270 （），–。https://doi.org/ . / 。Carlos Areces、Raul Fervari 和 Guillaume Hoffmann。2015。关系变更模态算子。《IGPL逻辑期刊》23，4（2015），601–627。https://doi.org/ 10.1093/JIGPAL/JZV020 [13] Carlos Areces、Hans van Ditmarsch、Raul Fervari、Bastien Maubert 和 François Schwarzentruber。2021。复制与移除作为动态算子。《应用非经典逻辑期刊》31，3-4（2021），181–220。https://doi.org/10.1080/ 11663081.2021.1964327[14] Carlos Areces、Hans van Ditmarsch、Raul Fervari 和 François Schwarzentruber。2017。复制与移除的模态逻辑。《信息与计算》255（ 2017 243 261 10 1016 2017 01 004 ），–。https://doi.org/ . /J.IC. . . Guillaume Aucher、Johan van Benthem 和 Davide Grossi。2018。再探破坏模态逻辑。《逻辑与计算杂志》28，2（2018），269–303。https://doi.org/ 10 1093 034 。 /LOGCOM/EXX Edoardo Baccini 和 Zoé Christoff。。比较社交网络动态算子。载于第19届 TARK 会议论文集（EPTCS，第379卷），Rineke Verbrugge（编）。 66 81 10 4204 379 8 –。https://doi.org/ . /EPTCS. . Alexandru Baltag 和 Sonja Smets。。学习他人所知。载于第23届 LPAR 会议论文集（计算领域EPiC系列，第73卷），Elvira Albert 和 Laura Kovács（编）。EasyChair， 90 119 10 29007 4 –。https://doi.org/ . /PLM Bartosz Bednarczyk、Stéphane Demri、Raul Fervari 和 Alessio Mansutti。2023。用模态逻辑组合有限森林。《ACM计算逻辑汇刊》 24, 2 2023 12 1 12 46 10 1145 3569954 （），：–：。https://doi.org/ . / Francesco Belardinelli 和 Stéphane Demri。。基于有界数量资源的策略推理：寻求可处理性。《人工智能》 300 2021 103557 10 1016 2021 103557 （），。https://doi.org/ . /J.ARTINT. . Gaia Belardinelli 和 Rasmus K. Rendsvig。。将注意力作为有界资源的认知规划。载于第8届 LORI 会议论文集（LNCS，第13039卷），Sujata Ghosh 和 Thomas Icard （编）。Springer，14–30。https://doi.org/10.1007/978- 3-030-88708-7 2_[21] Deval Bhamare、Maede Zolanvari、Aiman Erbad、Raj Jain、Khaled Khan 和 Nader Meskin。2020。工业控制系统的网络安全：综述。《计算机与安全》89（2020），101677。[22] Cem Bila、Fikret Sivrikaya、Manzoor A Khan 和 Sahin Albayrak。2016。未来的车辆：安全问题研究综述。《IEEE智能交通系统汇刊》18，5（2016），1046–1065。[23] Thomas Bolander、Lasse Dissing 和 Nicolai Herrmann。2021。基于 DEL 的人机协作认知规划：理论与实现。载于第18届 KR 会议论文集，Meghyn Bienvenu、Gerhard Lakemeyer 和 Esra Erdem（编）。 120 129 10 24963 2021 12 –。https://doi.org/ . / KR. / Nils Bulling 和 Mehdi Dastani。。基于规范的机制设计。《人工智能》239（2016），97–142。[25] Rui Cao 和 Pavel Naumov。2017。多智能体系统中的预算约束动态。载于第26届 IJCAI 会议论文集，Carles Sierra（编）。ijcai.org，915–921。https://doi.org/ 10 24963 2017 127 。 /IJCAI. / Davide Catta、Angelo Ferrando 和 Vadim Malvone。2024。基于资源行动的有界 ATL：一种用于多智能体系统、以表达行动成本的新逻辑。载于第25届 PRIMA 会议论文集（LNCS，第15395卷），Ryuta Arisaka、Víctor Sánchez- A nguix, Sebastian Stein, Reyhan Ayd奥根, L伊恩·范 der T奥尔，安d Takayu基·伊特o （编者）。Springer， 206 223 10 1007 978 3 031 77367 9_16 – 。https://doi.org/ 。 / - - - - Davide Catta、Jean Leneutre 和 Vadim Malvone。。攻击图与子集破坏博弈。《Intelligenza Artificiale》17，1（2023），77–88。https://doi.org/10.3233/IA-221080[28] Davide Catta、Jean Leneutre 和 Vadim Malvone。2023。《障碍逻辑：一种用于推理动态博弈模型的战略时序逻辑》。载于第26届ECAI论文集（《人工智能与应用前沿》，第372卷），Kobi Gal、Ann Nowé、Grzegorz J. Nalepa、Roy Fairstein 和 Roxana Radulescu（编）。IOS Press， 365 372 10 3233 230292 – 。https://doi.org/ 。 / FAIA Davide Catta、Jean Leneutre、Vadim Malvone 和 Aniello Murano。2024。《障碍交替时态逻辑：一种用于推理动态模型的战略逻辑》。载于第23届AAMAS论文集，Mehdi Dastani、Jaime Simão Sichman、Natasha Alechina 和 Virginia Dignum（编）。IFAAMAS / ACM，271–280。[30] Davide Catta、Jean Leneutre、Vadim Malvone 和 James Ortiz。2025。《联盟阻碍时序逻辑：一种用于推理恶魔联盟的新型障碍逻辑》。载于第34届IJCAI论文集。ijcai.org，21–28。https://doi.org/10. 24963 2025 3 /IJCAI。 / Ashok K. Chandra 、 Dexter Kozen 和 Larry J. Stockmeyer。1981。交替。《 ACM 学报》 28, 1 1981 114 133 10 1145 322234 322243 （），– 。https://doi.org/ 。 / 。Zoé Christoff、Jens Ulrik Hansen 和 Carlo Proietti。。《反思网络中的社会影响》。《逻辑、语言与信息期刊》25，3-4（2016），299–333。https://doi.org/10.1007/S10849-016-9242-Y [33] Edmund M. Clarke 和 E. Allen E merson。1981。《使用分支时序逻辑设计和综合同步骨架》。载于《程序逻辑》（LNCS，第131卷），Dexter Kozen（编）。Springer，52–71。https: //doi.org/ 10 1007 0025774 。 /BFB Edmund M. Clarke、Thomas A. Henzinger、Helmut Veith 和 Roderick Bloem（编）。2018。《模型检验手册》。Springer。[35] Stefania Costantini、Andrea Formisano 和 Valentina Pitoni。2021。《一种用于具有预算和成本的多智能体系统的认知逻辑》。载于第17届JELIA论文集（LNCS，第12678卷），Wolfgang Faber、Gerhard Friedrich、Martin Gebser 和 Michael Morak（编）。Springer，101–115。https://doi.org/10.1007/978-3-030- 75775-5 8[36] Tiago de Lima。2014。《交替时序动态认知逻辑》。《逻辑与计算杂志》24，6（2014），1145–1178。https://doi.org/10.1093/ LOGCOM/EXS061[37] Stéphane Demri 和 Morgan Deters。2015。《分离逻辑与模态：综述》。《应用非经典逻辑期刊》25，1（2015），50–99。https: //doi.org/ 10 1080 11663081 2015 1018801 。 / 。 。Stéphane Demri 和 Raul Fervari。。《模态分离逻辑的力量》。《逻辑与计算杂志》29，8（2019），1139–1184。https://doi.org/10. 1093/LOGCOM/EXZ019[39] Stéphane Demri 和 Raine Rönnholm。2023。《如何利用ATL管理预算》。载于第20届KR论文集，Pierre Marquis、Tran Cao Son 和 Gabriele Kern-Isberner（编）。 188 197 10 24963 2023 19 – 。https://doi.org/ 。 / KR。 / Vitaliy Dolgorukov、Rustam Galimullin 和 Maksim Gladyshev。。《资源受限信息挖掘智能体的动态认知逻辑》。载于第23届AAMAS论文集，Mehdi Dastani、Jaime Simão Sichman、Natasha Alechina 和 Virginia Dignum（编）。IFAAMAS / ACM，481–489。https://doi.org/10.5555/ 3635637.3662898[41] Ho Ngoc Duc。1997。《关于理性但并非逻辑全知的智能体的推理》。《逻辑与计算杂志》7，5（1997），633–648。https://doi.org/ 10.1093/LOGCOM/7.5.633[42] Ronald Fagin 和 Joseph Y. Halpern。1987。《信念、觉知与有限推理》。《人工智能》34，1（1987），39–76。https://doi.org/10.1016/0004- 3702 87 90003 8 （） - Raul Fervari 和 Fernando R. Velázquez-Quesada。2019。《关系模型中的内省作为一种行动》。《逻辑与代数编程方法杂志》 108 2019 1 23 10 1016 2019 06 005 （），– 。https://doi.org/ 。 /J.JLAMP. 。。Rustam Galimullin。。《联盟与相对化群体公告逻辑》。《逻辑、语言与信息期刊》30，3（2021），451–489。https://doi. org/ 10 1007 10849 020 09327 2 。 /S - - - Rustam Galimullin 和 Thomas Ågotnes。2021。《动态联盟逻辑：授予与撤销独裁权力》。载于第8届LORI论文集（LNCS，第13039卷），Sujata Ghosh 和 Thomas Icard（编）。Springer，88–101。https://doi.org/ 10 1007 978 3 030 88708 7_7 。 / - - - - Rustam Galimullin、Maksim Gladyshev、Munyque Mittelmann 和 Nima Motamed。2025。《改变博弈规则：多智能体系统中动态现象的推理》。载于第24届AAMAS论文集，Sanmay Das、Ann Nowé 和 Yevgeniy Vorobeychik（编）。IFAAMAS / ACM，829–838。https://doi.org/ 10 5555 3709347 3743601 。 / 。Rustam Gali mullin 和 Louwe B. Kuijer。2024。《利用ATL条件综合社会法则》。载于第23届AAMAS论文集，Mehdi Dastani、Jaime Simão Sichman、Natasha Alechina 和 Virginia Dignum（编）。IFAAMAS / ACM，2270–2272。https://doi.org/10.5555/3635637.3663130

[48] Rustam Galimullin、Munyque Mittelmann 和 Laurent Perrussel。2026。《扩散拍卖的形式化验证》。载于第40届AAAI论文集。（即将发表）。[49] Rustam Galimullin 和 Mina Young Pedersen。2024。《社交网络中的可见性与利用》。《计算机科学中的数学结构》34，7（2024），615 644 10 1017 0960129523000397 – 。https://doi.org/ 。 /S
[50] Rustam Galimullin 和 Fernando R. Velázquez-Quesada。2024。《智能体之间基于主题的通信》。《逻辑研究》（2024）。https://doi.org/10.1007/ s11225-024-10119-z[51] Saúl Fernández González。2022。《社交网络中的变化：社会认知逻辑的一些动态扩展》。《逻辑与计算杂志》32，6（2022 1212 1233 10 1093 024），– 。https://doi.org/ 。
[52] /LOGCOM/EXAC Davide Grossi 和 Simon Rey。 。《轻信可接受性、毒药博弈与模态逻辑》。载于第18届AAMAS论文集，Edith Elkind、Manuela Veloso、Noa Agmon 和 Matthew E. Taylor（编）。IFAAMASI，1994–1996。[53] Wojciech Jamroga 和 Aniello Murano。2015。《策略能力的模块检查》。载于第14届AAMAS论文集，Gerhard Weiss、Pinar Yolum、Rafael H. Bordini 和 Edith Elkind（编）。ACM，227–235。[54] Max Knobbout、Mehdi Dastani 和 John-Jules Ch. Meyer。2016。《规范变更的动态逻辑》。载于第22届ECAI论文集（《人工智能与应用前沿》，第285卷），Gal A. Kaminka、Maria Fox、Paolo Bouquet、Eyke Hüllermeier、Virginia Dignum、Frank Dignum 和 Frank van Harmelen（编）。IOS Press，886 894 10 3233 978 1 61499 672 9 886 – 。https://doi.org/ 。 / -
[55] - - - - Barteld Kooi 和 Bryan Renne。 。《箭头更新逻辑》。《符号逻辑评论》4, 4 2011 536 559 10 1017 1755020311000189（），– 。
[56] https://doi.org/ 。 /S Orna Kupferman、Moshe Y Vardi 和 Pierre Wolper。 。《模块检查》。《信息与计算》164，2（2001），322–344。[57] Jean Leneutre、Vadim Malvone 和 James Ortiz。2025。《定时阻碍逻辑：动态博弈推理的定时方法》。载于第24届AAMAS论文集，Sanmay Das、Ann Nowé 和 Yevgeniy Vorobeychik（编）。IFAAMAS / ACM，1272 1281 10 5555 3709347 3743759 – 。https://doi.org/ 。 / 。Dazhu Li。
[58] 。《失去连接：可定义链接删除的模态逻辑》。《逻辑与计算杂志》 30，3 （ 2020 ），715–743。https://doi.org/10.1093/ LOGCOM/EXZ036[59] Dazhu Li、Sujata Ghosh 和 Fenrong Liu。2025。《警察与强盗博弈中的不确定性推理》。《CoRR》abs/2508.00004（2025）。https://doi.org/10. 48550/ARXIV.2508.00004[60] Dazhu Li、Sujata Ghosh、Fenrong Liu 和 Yaxin Tu。2023。《一种简单的捉迷藏博弈逻辑》。《逻辑研究》111，5（2023），821–853。https://doi.org/10. 1007/S11225-023-10039-4[61] Christof Löding 和 Philipp Rohde。2003。《破坏模态逻辑的模型检测与可满足性》。载于第23届FSTTCS论文集（LNCS，第2914卷），Paritosh K. Pandya 和 Jaikumar Radhakrishnan（编）。Springer，302–313。https: // doi.org/ 10 1007 978 3 540 24597 1_26 。 / - - - - Bastien Maubert、Sophie
[62] Pinchinat 和 François Schwarzentruber。2019。《动态认知逻辑中的可达性博弈》。载于第28届IJCAI论文集，Sarit Kraus（编）。ijcai.org，499–505。https://doi.org/10.24963/IJCAI.2019/71[63] Bastien Maubert、Sophie Pinchinat、François Schwarzentruber 和 Silvia Stranieri。2020。《动态认知逻辑中的并发博弈》。载于第29届IJCAI论文集，Christian Bessiere（编）。1877–1883。https://doi.org/10. 24963 2020 260 /IJCAI。 / Fabio Mogavero、Aniello Murano 和 Moshe Y. Vardi。
[64] 2010。《策略推理》。载于第30届FSTTCS论文集（LIPIcs，第8卷），Kamal Lodaya 和 Meena Mahajan（编）。施洛斯·达格施图尔—莱布尼茨信息学中心，133–144 10 4230 2010 133 。https://doi.org/ 。 /LIPICS.FSTTCS. 。
[65] Dario Della Monica、Margherita Napoli 和 Mimmo Parente。2011。《论一种具有定价资源智能体的联盟博弈逻辑》。载于第7届M4M和第4届LAMAS论文集（ENTCS，第278卷），Hans van Ditmarsch、David Fernández-Duque、Valentin Goranko、Wojciech Jamroga 和 Manuel Ojeda-Aciego（编）。爱思唯尔，215 228 10 1016 2011 10 017 – 。https://doi.org/ 。 /J.ENTCS. 。
[66] 。Marc Pauly。 。《博弈中联盟力量的模态逻辑》。《逻辑与计算杂志》12, 1 2002 149 166 10 1093 12 1 149（），– 。
[67] https://doi.org/ 。 /logcom/ 。 。Mina Young Pedersen。 。《恶意智能体与少数者的力量：社交网络中的异常逻辑》。博士论文。卑尔根大学，挪威。https://bora.uib.no/bora-xmlui/handle/11250 3151733 / A. Pnueli。 。《程序时序逻辑》。
[68] 载于计算机科学基础年度研讨会论文集。[69] John C. Reynolds。2002。《分离逻辑：一种用于共享可变数据结构的逻辑》。载于第17届LICS论文集。IEEE计算机学会，55–74。https: //doi.org/ 10 1109 2002 1029817 。 /LICS。 。Philipp Rohde。 。《增添破坏机制的 -演算》。
[70] 载于第9届FOSSACS论文集（LNCS，第3921卷），Luca Aceto 和 Anna Ingólfsdóttir（编）。Springer，142 156 10 1007 11690634_10 – 。https://doi.org/ 。 /
[71] Anthia Solaki。 。《在有界群体中实现分布式知识》。《逻辑与计算杂志》33，6（2023），1497–1525。https://doi.org/10. 1093/LOGCOM/EXAC007[72] Johan van Benthem。2005。《关于破坏与阻碍的论述》。载于《数学推理机械化》（LNCS，第2605卷）。Springer，268–276。
[73] Johan van Benthem 和 Fenrong Liu。2020。《图博弈与逻辑设计》。载于知识、证明与动态，Fenrong Liu、Hiroakira Ono 和 Junhua Yu（编）。Springer，125–146。[74] Wiebe van der Hoek、Louwe B. Kuijer 和 Yì N. Wáng。2020。《盟友与敌人逻辑：社会平衡理论动态的形式化方法》。载于第*29*届 *IJCAI* 论文集，Christian Bessiere（编）。ijcai.org，210–216。https://doi.org/ 10 24963 2020 30 。/IJCAI。/
[75] Hans van Ditmarsch、Wiebe van der Hoek 和 Barteld Kooi。2008。《动态认知逻辑》。Synthese Library，第337卷。Springer。[76] Hans van Ditmarsch、Wiebe van der Hoek、Barteld Kooi 和 Louwe B. Kuijer。2017。任意箭头更新逻辑。人工智能 242 (2017)，80–106。https://doi.org/ 10 1016 2016 10 003 。/
[77] J.ARTINT。... Zuojun Xiong 和 Meiyun Guo。。《一种追随的动态混合逻辑》。载于第7届 *LORI* 论文集（ *LNCS*，第 *11813* 卷），Patrick Blackburn 、 Emiliano Lorini 和 Meiyun Guo（编）。Springer，425–439。https://doi.org/10.1007/978- 3-662-60292-8 31[78] Francesca Zaffora Blando、Krzysztof Mierzewski 和 Carlos Areces。2020。《毒药博弈的模态逻辑》。载于知识、证明与动态，Fenrong Liu、Hiroakira Ono 和 Junhua Yu（编）。Springer，3–23。

技术附录

定理 4. SDL $\not\preccurlyeq$ OL。

证明。考虑一个 SDL公式 $^1\mathrm{F}\varphi$ 并反设存在一个与之等价的 OL 公式 φ。由于 OL 的公式是有限的，且每个 OL 模态都有有限的资源界，我们可以假设 n 是 φ中出现的最大 $n \in \mathbb{N}$。现在考虑下方图7中的模型 $\mathfrak{M}_{n+2}$ 和 $\mathfrak{M}_{n+3}$。

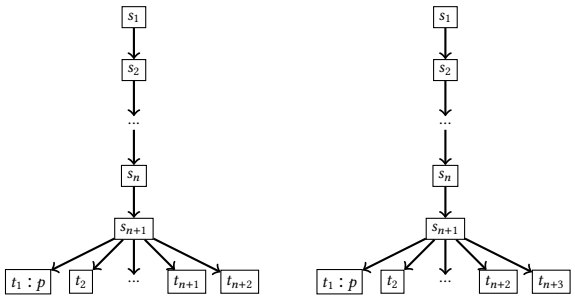

图7：模型 $\mathfrak{M}_{n+2}$（左）和 $\mathfrak{M}_{n+3}$（右），为便于阅读，省略了 t-状态的自反箭头。所有边的代价均为 1。

这两个模型非常相似，唯一的区别是左边的模型少了一个 t-状态。在这两个模型中，p 仅在状态 t_1中为真。

现在，为了证明 $(\mathfrak{M}_{n+2}, s_1) \models \varphi$ 当且仅当 $(\mathfrak{M}_{n+3}, s_1) \models \varphi$，我们对 φ 使用归纳法，并在此给出论证概要。对于基例，两个模型中的状态 s_1 显然满足相同的命题原子。布尔情形也依预期成立。现在，我们来看看构造 $\dagger^m\psi$ 和 $[\dagger^m]\psi$with $m \leqslant n$。首先请注意，这些模型的构造方式保证了恶魔无法移除 s_i-状态之间的任何边。此外，回想一下，在 OL 中，恶魔移除一些边且旅行者完成移动后，所有边都会恢复。因此，只需考虑从状态 s_{n+1} 出发的恶魔策略。于是，剩下需要论证的是：对于 $(\mathfrak{M}_{n+2}, s_{n+1})$ 中的任意恶魔策略，在 $(\mathfrak{M}_{n+3}, s_{n+1})$ 中都存在一个与之等价的恶魔策略，反之亦然。

从左到右来看，显然，在 $(\mathfrak{M}_{n+2}, s_{n+1})$ 中选择移除至多 n 条 $s_{n+1} - t_i$-边，可以在 $(\mathfrak{M}_{n+3}, s_{n+1})$ 中通过完全相同的选择直接复现。因此，如果在 $(\mathfrak{M}_{n+2}, s_{n+1})$ 中存在恶魔策略，那么在 $(\mathfrak{M}_{n+3}, s_{n+1})$ 中就存在一个等价的恶魔策略。此外，$(\mathfrak{M}_{n+2}, s_{n+1})$ 中的任意恶魔策略，都有一个在 $(\mathfrak{M}_{n+3}, s_{n+1})$ 中与之等价的恶魔策略。

在从右到左方向，我们可以运用相当类似的论证，唯一的区别是：如果恶魔在（$\mathfrak{M}_{n+3}$，s_{n+1}）中移除了 s_{n+1}与 t_{n+3}之间的边，那么在（$\mathfrak{M}_{n+2}$，s_{n+1}）中，我们选择第一个满足 $t_i \neq t_1$ 且从s_{n+1}到 t_i的边在（$\mathfrak{M}_{n+3}$，s_{n+1}）中未被移除的 t_i-状态。由于我们假设 n 是 φ中最大的资源界，这样的状态 t_i 总是存在。由于那里分别有 $n+2$ 和 $n+3$ t-状态，恶魔总会至少留下 2 个（分别为 3 个）可访问的 t-状态；也就是说，至少有一个不满足 p 的 t-状态总是可访问的。在两种情形下，无论是（i） t_1 仍然可访问，且其他 t-状态中也有 k，还是（ii）只有 $i > 1$可访问的 $k \leqslant n+2$ t_i-状态，旅行者都必须移入这些状态之一，在那里不可能再采取非平凡的恶魔策略。很容易看出，任何 OL公式都无法区分（$\mathfrak{M}_{n+2}$，t_1）和（$\mathfrak{M}_{n+3}$，t_1），同样也无法区分（$\mathfrak{M}_{n+2}$，t_i）和（$\mathfrak{M}_{n+3}$，t_j），其中 i，$j > 1$。

现在回到我们的 SDL公式 $^1\mathrm{F}p$，我们可以论证，$(\mathfrak{M}_{n+2}, s) \models {}^1\mathrm{F}p$，且 $(\mathfrak{M}_{n+3}, s) \not\models {}^1\mathrm{F}p$。先来看 $(\mathfrak{M}_{n+2}, s) \models {}^1\mathrm{F}p$。根据语义的定义，这等价于存在一个代价为 $\mathsf{C}(\mathfrak{S}) \leqslant 1$的恶魔策略 $\mathfrak{S}$，使得对于所有路径 $\pi \in Out(\mathfrak{S}, (\mathfrak{M}, s))$，都有 $\pi \models \mathrm{F}p$。恶魔策略是一个从带指定状态的模型到边集的函数；这里我们考虑如下策略 $\mathfrak{S}$：给定一个带有某个 s_i 状态的带指定状态的模型时，就选择移除边$s_{n+1} \to t_{i+1}$；给定一个带有某个 t-状态的带指定状态的模型时，就选择移除空边集。直观而言，该策略会逐一移除从 s_{n+1} 到 t_i 状态的边，每次移除一条，且满足 $i > 1$。这一恶魔策略是在旅行者的所有可能移动面前执行的。请注意，模型的构造方式使得旅行者除了沿着 s-状态移动之外别无选择。因此，当旅行者到达状态 s_{n+1}时，恶魔已经移除了从状态 s_{n+1} 到状态 t_2 至 t_{n+1}的 n 条边。在状态 s_{n+1}，恶魔遵循其策略，移除连接 s_{n+1} 和 t_{n+2}的边，从而迫使旅行者进入 t_1 状态，在那里 p 成立。

要看出 $(\mathfrak{M}_{n+3}, s) \not\models {}^1\mathrm{F}p$，只需注意到，同样的渐进式边删除技巧无法用于模型 $\mathfrak{M}_{n+3}$：当旅行者到达状态 s_{n+1}时，恶魔仍需删除两条边，即从 s_{n+1} 到 t_{n+2} 和 t_{n+3}的边，才能强制满足 p。这是不可能的，因为模型的构造方式保证恶魔每轮最多只能删除一条边。这意味着恶魔无法保证旅行者最终到达满足 p 的状态 t_1。□

定理 5. SDL 的模型检测是 PSPACE 完全。

证明。用于模型检测 SDL 的完整算法如下所示。

算法2一个用于模型检测 SDL 的算法

```
过程 MC((𝔐, s) φ)
  情形 φ = p
    返回 s ∈ V(p)
  情形 φ = ¬p
    返回非 s ∈ 𝒱(p)
  情形 φ = ψ ∨ χ
    存在性地选择 θ ∈ {ψ, χ}
    返回 MC((𝔐, s) θ)
  情形 φ = ψ ∧ χ
    对所有情形选择 θ ∈ {ψ, χ}
    返回 MC((𝔐, s) θ)
  情形 φ = ⟨⟩^n_X ψ
    存在地选择 n-子模型 𝔐′
    对所有情形选择 s′ 使得 s --𝔐′--> s′
    返回 MC((𝔐′, s′) ψ)
  情形 φ = ⟨⟩^n ψ1 U ψ2
    X ← (𝔐, s)
    i ← 0
    当.....时 非 MC(X, ψ2) 并且 i ⩽ brDepth 执行
      如果 非MC(X, ψ1) 则
        返回 false
```

存在性地选择 n-子模型 $\mathfrak{M}'$ 的 X
全称地选择 s' 使得 $s \xrightarrow{\mathfrak{M}'} s'$
$X \leftarrow (\mathfrak{M}', s')$
$i \leftarrow i+1$
结束当……时
如果 $i > brDepth$ **那么**
返回 假
else
返回 真
情形 $\varphi = \langle\!\langle n \rangle\!\rangle \psi_1 \mathrm{R} \psi_2$
$X \leftarrow (\mathfrak{M}, s)$
$i \leftarrow 0$
当……时 $\mathrm{MC}(X, \psi_2)$ 且 $i \leqslant brDepth$ **执行**
如果 $\mathrm{MC}(X, \psi_1)$ **则**
返回 真
存在性地选择 n-子模型 $\mathfrak{M}'$ 的 X
全称地选择 s' 使得 $s \xrightarrow{\mathfrak{M}'} s'$
$X \leftarrow (\mathfrak{M}', s')$
$i \leftarrow i+1$
结束当……时
如果 $i > brDepth$ **那么**
返回 真
else
返回 假
结束过程

$[\![n]\!]\mathrm{X}\psi$、$[\![n]\!]\psi_1 \mathrm{U} \psi_2$和 $[\![n]\!]\psi_1 \mathrm{R} \psi_2$ 的情形，采用与其菱形对应项类似的方式，通过交换存在选择和全称选择来计算。□

定理 6. SCL 的模型检测是 PSPACE 完全。

证明。对于 PSPACE 包含性的论证，我们可以重用算法 2。具体而言，我们将每一处 ^{n} 和 $[\![^{n}]\!]$ 分别替换为 ^{n} 和 $[\![^{n}]\!]$，并将“n-子模型”替换为“n-超模型”。每个分支的深度 $brDepth$ 至多为 $O(|S|^2 \cdot |S|)$，因为天使最多可以通过逐条添加边，连续 $|S|^2+1$ 次修改模型，而旅行者可能需要遍历所得超模型的至多 $|S|$ 个状态。

为了证明 PSPACE 困难性，我们再次采用从 QBF 问题出发的归约。对于给定的 QBF 实例 $\Psi := Q_1 p_1 \ldots Q_n p_n \psi(p_1, \ldots, p_n)$，我们构造模型 $\mathfrak{M}^{\Psi} = (S, \to, \mathrm{V}, \mathrm{C})$，其中 $S = \{s, \ldots, s_{2n}\}$，$s \to s$，且对于所有 $s_i \in S$，$\mathrm{V}(p_i^1) = \{s_i\}$，$\mathrm{V}(p_i^0) = \{s_{n+i}\}$（当 $1 \leqslant i \leqslant n$ 时），任意一对状态之间的边的代价均为 1。图 8 展示了为 $\Psi := \forall p_1 \exists p_2 \; p_1 \to p_2$ 构造的模型 $\mathfrak{M}^{\Psi}$ 的一个示例。

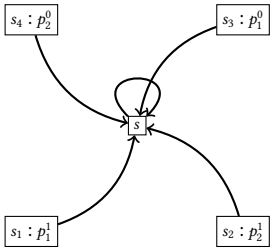

图 8：QBF 实例 Ψ 的模型 $\mathfrak{M}^{\Psi}$。所有边的代价均为 1。

与 SDL 的情形类似，我们将标准模态菱形定义为 $\Diamond\varphi := [\bigcirc^0]\mathrm{X}\varphi$ 的缩写。相应的公式 $chosen_k$ 定义如下：

$$chosen_k = \bigwedge_{1 \leqslant i \leqslant k} (\Diamond\Diamond p_i^0 \leftrightarrow \neg\Diamond\Diamond p_i^1) \wedge \bigwedge_{k < i \leqslant n} (\neg\Diamond\Diamond p_i^0 \wedge \neg\Diamond\Diamond p_i^1).$$

公式 $chosen_k$ 表示，前 k 个原子命题在 Ψ 中的真值已被明确设定（即，对于每个 p_i，对于 $1 \leqslant i \leqslant k$，$p_i^0$和 p_i^1中恰有一个可达），而其他原子命题的真值尚未设定（即，对于每个满足 $k < i \leqslant n$ 的 p_i，p_i^0和 p_i^1均不可达）。与 SDL 情形的主要区别在于，现在我们添加边，而不是移除边。

接下来，φ 的构造如下：

$$\varphi_0 := \psi(\Diamond\Diamond p_1^1, \ldots, \Diamond\Diamond p_n^1)$$

$$\varphi_k := \begin{cases} [\bigcirc^1]\mathrm{X}\,(chosen_k \to \varphi_{k-1}) & \text{if } Q_k = \forall \\ \langle\bigcirc^1\rangle\mathrm{X}\,(chosen_k \wedge \varphi_{k-1}) & \text{if } Q_k = \exists \end{cases}$$

$$\varphi := \varphi_n.$$

其余论证与 SDL 的 PSPACE 困难性证明类似。□

定理 7。 SUL 的模型检测属于 EXPSPACE，而 SUL 的下一时刻片段的模型检测是 PSPACE 完全的。

证明。为了证明 SUL 的模型检测属于 EXPSPACE，我们给出一个交替算法 3，它与算法 1 类似。为简洁起见，我们只考虑情形 $[C]^n.\, {}^m\mathrm{X}\psi$和 $[C]^n.\, {}^m\psi_1 \mathrm{U} \psi_2$，其他情形也可以类似地计算。不失一般性，我们假设公式 φ处于否定范式。给定一个有限模型 $\mathfrak{M} = (S, \to, \mathrm{V}, \mathrm{C})$、$n$-超模型 $\mathfrak{M}_1 = \mathfrak{M} \cup A$以及 m-子模型 $\mathfrak{M}_2 = \mathfrak{M} \setminus B$，其中$A$和 B是边集的子集，则 n-m-更新 $\mathfrak{M}_1 \star \mathfrak{M}_2$ 的大小至多为 $O(|\mathfrak{M}|^2)$（即完全图）。

算法 3SUL 的模型检测算法

过程 $\mathrm{MC}((\mathfrak{M}, s)\ \varphi)$
情形 $\varphi = [C]^n.\, {}^m\mathrm{X}\psi$
返回 $\mathrm{MC}(\mathrm{Update}((\mathfrak{M}, s)\ C, n, m)\ \psi)$
情形 $\varphi = [C]^n.\, {}^m\psi_1 \mathrm{U} \psi_2$
$X \leftarrow (\mathfrak{M}, s)$
$i \leftarrow 0$
当……时 X 不是 $\mathrm{MC}(X, \psi_2)$ 并且 $i \leqslant brDepth$ **执行**
如果 X 不是$\mathrm{MC}(X, \psi_1)$ **那么**
返回 假
$X \leftarrow$ 更新(X, C, n, m)
$i \leftarrow i+1$
结束当……时循环
如果 $i > brDepth$ **那么**
返回 假
else
返回 真
结束过程
过程 更新$((\mathfrak{M}, s)\ C, n, m)$
$X \leftarrow (\mathfrak{M}, s)$
如果 $C = \{\langle\!\langle m \rangle\!\rangle\}$ **那么**
存在性地选择 m-子模型 $\mathfrak{M}_1$ 的 $\mathfrak{M}$
全称地选择 n-超模型 $\mathfrak{M}_2$的 $\mathfrak{M}$ **且** s' 满足 $s \xrightarrow{\mathfrak{M}_1 \star \mathfrak{M}_2} s'$
返回 $(\mathfrak{M}_1 \star \mathfrak{M}_2, s')$

否则，如果 $C = \{\infty\bigcirc\!\!\infty^n\}$ **则**
  存在性地选择 n-超模型 $\mathfrak{M}_1$ 的 $\mathfrak{M}$
  全称地选择 m-子模型 $\mathfrak{M}_2$ 的 $\mathfrak{M}$ **以及** s' 使得 $s \xrightarrow{\mathfrak{M}_1 \star \mathfrak{M}_2} s'$
  返回 $(\mathfrak{M}_1 \star \mathfrak{M}_2, s')$
否则，如果 $C = \{\infty\bigcirc\!\!\infty^n, \text{⛧}^m\}$ **然后**
  存在地选择 n-m-更新 $\mathfrak{M}'$ 的 $\mathfrak{M}$
  全称地选择 s' 使得 $s \xrightarrow{\mathfrak{M}} s'$
  返回$(\mathfrak{M}', s')$
else
  全称地选择 n-m-更新 $\mathfrak{M}'$ 的 $\mathfrak{M}$ **和** s' 使得 $s \xrightarrow{\mathfrak{M}} s'$
  返回$(\mathfrak{M}', s')$
结束过程

在算法中，Update 过程根据给定的模型和集合 C确定要使用的更新后的模型。关于复杂度，Update 过程可在多项式时间内运行。对于 SUL 的下一时刻片段，算法 3 至多检查 $|\varphi|$ 个 φ的子公式，因此总运行时间为多项式时间。从由 APTIME=PSPACE 可知，SUL 的下一时刻片段属于 PSPACE。困难性直接来自以下事实：SDL 和 SCL 都包含于 SUL，且这两种逻辑的模型检验问题都是 PSPACE 完全。

现在，我们来讨论 SUL 的完整语言。该算法与用于 SDL 和 SCL 的算法非常相似。不过，我们的路径树 $brDepth$中每个分支的深度现在是指数级的，因为恶魔和天使在最坏情形下可以共同迫使我们考虑完全图中大小为 $|S|^2$ 的指数多个子模型。对于每个这样的子模型，我们还可能需要检查旅行者可以访问的至多 $|S|$ 个状态。特别地，$brDepth$至多为 $O(2^{|S|^2} \cdot |S|) \leqslant O(2^{|\mathfrak{M}|^2})$。由于至多有$|\varphi|$ 个子公式需要考虑，算法 3 的运行时间有上界$O(|\varphi| \cdot 2^{|\mathfrak{M}|^2})$。根据 AEXPTIME = EXPSPACE [31],这一事实，我们得出 SUL 的模型检测属于 EXPSPACE。□