arXiv:2603.01201v1 [cs.AI] 1 Mar 2026

增量 LTL$_f$ 综合

Giuseppe De Giacomo[1,2], **Yves Lesp´erance**[3],**Gianmarco Parretti**[2],**Fabio Patrizi**[2],**Moshe Y. Vardi**[4]

牛津大学[2] 罗马大学“智慧大学”[3] 约克犬学[4] 莱斯大学 giuseppe.degiacomo@cs.ox.ac.uk,
lesperan@yorku.ca, {parretti, patrizi}@diag.uniroma1.it, vardi@cs.rice.edu

摘要

本文研究增量 LTL$_f$ 综合——这是一种反应式综合形式，其中目标在执行过程中逐步给出。换言之，当主导智能体已经针对某个目标执行策略时，它会收到一个新目标：此时，智能体必须放弃当前策略，并综合出一个新策略，既满足最初给定的原目标，也满足从当前时刻开始的新目标。本文对增量综合问题进行形式化定义，并研究其求解方法。我们提出一种解决方案，通过利用基于自动机的合成过程中构建的辅助数据结构，高效地为多个 LTL$_f$目标执行增量综合。我们还考虑了另一种基于 LTL$_f$ 公式推进的解决方案。我们证明，尽管公式推进可能生成比原公式大得多（呈指数级增长）的公式，但这些公式对应的最小自动机大小仍受原公式对应的自动机大小所限。另一方面，实验表明，如果采用朴素实现，即每当收到新目标时，都从头计算推进后 LTL$_f$ 公式的自动机，那么基于公式推进的解决方案并不具备竞争力。

1 引言

本文研究增量式LTL$_f$ 综合——这是一种反应式合成形式，其目标在执行过程中逐步给出。反应式合成最初在形式化方法中针对无限轨迹上的线性时序逻辑（LTL）进行了研究（Pnueli 1977年；Pnueli 和 Rosner 1989年），并采用了源自模型检测的技术。近来，研究者开始针对有限轨迹上的线性时序逻辑开展综合研究（De Giacomo 和 Vardi 2013），在这种情况下，智能体必须在完成目标后最终停止（De Giacomo 和 Vardi 2015）。LTL$_f$ 综合具有一些良好的计算特性，使人们能够利用模型检测中常见的符号技术（Baier 和 Katoen 2008），从而获得可扩展性显著优于 LTL 求解器的符号求解器，参见（Zhu 等，2017；Bansal 等，2020；De Giacomo 和 Favorito 2021；Kankariya 和 Bansal 2024；Zhu 和 Favorito 2025；Duret-Lutz 等人 2025）。

在人工智能领域，反应式合成本质上是完全可观测非确定性（FOND）领域中针对时序扩展目标的强规划的一种变体（Cimatti 等，2003；Bac-chus 和 Kabanza，2000；Baier、Fritz 和 McIlraith，2007；DeGiacomo 和 Rubin，2018；Camacho 和 McIlraith，2019；De Giacomo、Parretti 和 Zhu，2023）。

增量综合处理这样的情况：智能体已经在执行某个目标的策略时，又收到了一个新的目标。此时，智能体必须放弃正在执行的原目标策略，并综合出一个同时满足原目标和新目标的新策略。关键在于，原目标必须从一开始就得到满足，并贯穿当前历史；而新目标则从当前时刻开始生效。一般而言，执行过程中可能需要多次进行增量综合。

增量综合与综合的组合式方法有一些相似之处。在这类方法中，LTL$_f$ 规范是由若干分别处理的子公式合取而成（Bansal 等，2020, 2022）。然而，此处的合取涉及需要在起始时间点不同的轨迹上进行评估的公式。增量综合也与实时综合有一些相似之处：实时综合是在经历一段历史之后，而非在初始状态下进行综合（Finkbeiner、Klein 和 Metzger 2022；Zhu 和 De Giacomo 2022）。

显然，在采纳新目标时，我们必须确保新目标与先前目标共同具备可实现性。如果发生冲突，我们需要决定是放弃采纳新目标，还是舍弃一些旧目标，以确保已采纳的目标仍然可实现。本文不讨论如何解决这一冲突，而将其留待未来工作；不过，我们的方法确实能够判断是否存在冲突。

本文对增量综合问题进行了形式化定义，并研究了其解决方案。具体而言，我们提出了一种解决技术，利用基于自动机的合成过程中构建的辅助数据结构。我们证明了该方法的可靠性与完备性，并利用它刻画了增量 LTL$_f$ 综合的最坏情况计算复杂度为 2EXPTIME 完全，与标准 LTL$_f$ 综合相同。随后，我们使用原型实现，通过实验评估了该方法的有效性。

我们还研究了另一种求解增量综合的方法，即利用 LTL$_f$ 公式推进，也就是时序公式的不动点刻画（Gabbay 等人 1980；Manna 1982；Emerson 1990；Bacchus 与 Kabanza 2000年；De Giacomo 等人，2022年），从而可以递

归地将 LTL$_f$ 公式拆分为一个关于当前 时刻的命题公式，以及一个需在下一时刻 检验的时序公式。通过应用公式推进，我们可以将增量综合问题归约为针对已推进的原始目标与新目标之合取的标准综合。推进后的公式可能会比初始公式呈指数级变大，因此人们可能会认为，采用该技术的增量综合是 3EXPTIME 的。相反，我们在此证明，推进后公式的自动机（在最小化后）以其相应原始公式的自动机为界。因此，同样使用 LTL$_f$ 推进，增量综合也可以在双指数时间内求解。然而，基于 LTL$_f$ 推进的技术如果以朴素方式实现，即每当有新目标到来时都实际去计算推进后公式的自动机，则并不真正具有竞争力，正如我们在第7节中通过实验所展示的那样。事实上，我们在此提出的基于自动机的解决方案技术，可以看作是基于公式推进的解决方案的一种巧妙实现，它利用了自动机缓存。

2 预备知识

我们考虑有限轨迹 上的线性时序逻辑（LTL$_f$）中的规约（又称公式）（De Giacomo 和 Vardi 2013）。LTL$_f$ 是人工智能中广泛使用的一种形式体系，用于刻画有限时间范围上的时序性质，例如规划中的时序扩展目标（Camacho 等，2017；De Giacomo 和 Rubin，2018；Camacho 和 McIlraith，2019），以及状态/动作轨迹约束（Bonassi 等，2021；Bonassi、Gerevini 和 Scala，2022）。

我们要求 LTL$_f$ 规约采用否定范式（否定范式）：否定只能出现在原子命题之前。这不会损失一般性：每个 LTL$_f$ 公式都可以在线性时间内转换为否定范式。形式上，给定一组原子命题 （又称原子命题） P，否定范式中 LTL$_f$ 公式 φ 的语法为：

$$\varphi ::= p \mid \neg p \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \vee \varphi_2 \mid \bigcirc\varphi \mid \bullet\varphi \mid \varphi_1 \,\mathcal{U}\, \varphi_2 \mid \varphi_1 \,\mathcal{R}\, \varphi_2$$

其中 $p \in P$。布尔算子包括：否定 ($\neg$)；析取 ($\vee$)；以及合取 ($\wedge$)。时序算子包括：下一时刻 ($\bigcirc$)；弱下一时刻 ($\bullet$)；*until* ($\mathcal{U}$)；以及 释放 ($\mathcal{R}$)。其他算子包括：命题逻辑中的标准算子，例如 $true$、 $false$、蕴含 ($\supset$) 和等价 ($\equiv$)；以及作为缩写定义的时序算子，例如最终 ($\Diamond\varphi \doteq true\,\mathcal{U}\,\varphi$)和总是($\Box\varphi \doteq false\,\mathcal{R}\,\varphi$)。 φ的大小记作 $|\varphi|$，即其抽象语法树中的子公式数量。

LTL$_f$ 公式定义在有限轨迹 $\tau \in (2^P)^*$之上，这些轨迹由 P上的命题解释构成。空迹为 ϵ。我们用 $\tau_i \in 2^P$表示 τ 在第i个时间步的命题解释，并用 $|\tau|$ 表示其长度。给定轨迹$\tau = \tau_0 \cdots \tau_n$和 $\tau' = \tau'_0 \cdots \tau'_m$，它们的拼接是轨迹 $\tau \cdot \tau' = \tau_0 \cdots \tau_n \cdot \tau'_0 \cdots \tau'_m$。 给定一个 LTL$_f$公式 φ、一条轨迹 τ以及满足 $0 \le i < |\tau|$的索引 i，以下归纳定义形式化地描述了 τ 满足 φ于 i的条件：

- $\tau,\ i \models p$当且仅当 $p \in \tau_i$，对于 $p \in P$；
- $\tau,\ i \models \neg p$当且仅当 $p \notin \tau_i$，对于 $p \in P$；
- $\tau,\ i \models \varphi_1 \vee \varphi_2$当且仅当 $\tau,\ i \models \varphi_1$或 $\tau,\ i \models \varphi_2$；
- $\tau,\ i \models \varphi_1 \wedge \varphi_2$当且仅当 $\tau,\ i \models \varphi_1$并且 $\tau,\ i \models \varphi_2$；
- $\tau,\ i \models \bigcirc\varphi$当且仅当 $i < |\tau| - 1$并且 $\tau,\ i+1 \models \varphi$；
- $\tau,\ i \models \bullet\varphi$当且仅当 $i < |\tau| - 1$蕴含 $\tau,\ i+1 \models \varphi$；
- $\tau,\ i \models \varphi_1 \,\mathcal{U}\, \varphi_2$当且仅当存在 j，使得 $i \le j < |\tau|$且 $\tau,\ j \models \varphi_2$，并且对于每个满足 $i \le k < j$ 的 k，都有 $\tau,\ k \models \varphi_1$；

- $\tau,\ i \models \varphi_1 \,\mathcal{R}\, \varphi_2$当且仅当以下任一情况成立：(1) 存在 j，使得 $i \le j < |\tau|$且 $\tau,\ j \models \varphi_1$，并且对于每个满足 $i \le k \le j$的 k，都有 $\tau,\ k \models \varphi_2$；或 (2) 对于每个满足$i \le k < |\tau|$的 k，都有 $\tau,\ k \models \varphi_2$。

有限轨迹 τ满足 φ，记作 $\tau \models \varphi$，当且仅当 $\tau, 0 \models \varphi$。

LTL$_f$ （反应式）综合旨在寻找一种策略，以满足 LTL$_f$ 目标规范。目标用 LTL$_f$公式表示，定义在 $P = \mathcal{Y} \cup \mathcal{X}$上，其中 $\mathcal{Y}$ 和 $\mathcal{X}$是互不相交的原子命题集合，分别由智能体和环境控制。定义在 $\mathcal{Y} \cup \mathcal{X}$ $\tau = Y_0 \cup$上的 $X_0\ Y_1 \cup X_1 \cdots$ $Y_i \in 2^{\mathcal{Y}}$ $X_i \in 2^{\mathcal{X}}$ 轨迹记作 ( )( )，其中 且对每个$i \ge 0$都成立。这种形式的无限轨迹称为对局；有限轨迹也称为历史。

（智能体）策略是一个函数 $\sigma : (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$，它将（可能为空的）环境动作序列映射为一个智能体动作。由于我们假设智能体先行动（如同规划中一样）， σ 的定义域包含空迹ϵ。若对局 $\tau = (Y_0 \cup X_0)(Y_1 \cup X_1) \cdots$满足以下条件，则称其为 σ-一致的：(*i*)$Y_0 = \sigma(\epsilon)$；并且 (*ii*) 对每个 $i > 0$， $Y_i = \sigma(X_0 \cdots X_{i-1})$。历史 h 为 σ-一致的定义类似。 σ-一致对局的集合记作$Play(\sigma)$。

若对每一个对局 $\tau \in Play(\sigma)$，都存在一个满足 φ的有限前缀 τ^k，即$\tau^k \models \varphi$，则策略 σ 对 φ 是获胜的。LTL$_f$综合是寻找 φ的获胜策略（若存在）的问题。若存在 φ的获胜策略，我们称综合是 可实现的；否则，它是 不可实现的。LTL$_f$ 综合关于 φ 是 2EXPTIME 完全的（De Giacomo 和 Vardi 2015）。

3 增量综合

本文研究增量LTL$_f$ 综合。在增量综合中，智能体并不像标准综合那样预先知道目标，而是在执行过程中在线接收这些目标。

具体而言，考虑一个正在执行策略 σ 的智能体——该策略先前是为目标 φ_{org} 综合的——迄今已生成历史 h。假设在 h结束时到达一个新目标 φ_{new}。一方面，智能体不能依赖原始策略 σ，因为它是为 φ_{org}综合的，甚至不知道 φ_{new}；另一方面，智能体也不能简单地为 φ_{org} 和 φ_{new}综合一个新策略 σ'，因为 σ' 会从当前时刻重新开始满足 φ_{org}，而没有考虑到在历史 h 之后，原始目标 φ_{org} 只剩下一部分有待实现。相反，我们希望 σ'：（R1）保证 φ_{new}的满足；以及（R2）从 σ 停下的地方继续满足 φ_{org}，即考虑到历史 h。我们在下文形式化这一直觉。

要求 (R1) 可以通过要求新策略对 φ_{new}获胜来轻松表达。关于要求 (R2)：我们形式化定义一个新概念，用以说明在考虑到历史 h 已发生的情况下，策略 σ 对 φ 获胜意味着什么。也就是说，策略 σ 在 h 执行时，只会生成从起始处就满足 φ 的对局。形式化地说：

定义 1 设 φ 是一个 LTLf 目标，h 是一段历史。若对于每个 $\tau \in Play(\sigma)$，$\tau' = h \cdot \tau$ 都有一个满足 φ 的有限前缀，则称策略 σ 在假设 h 的条件下对 φ 获胜。

有了能够刻画要求 (R1) 和 (R2) 的概念，我们可以正式定义增量综合问题：

定义 2 设：φ_{org} 为一个原始的 LTLf 目标；h 为一段历史；φ_{new} 为一个新目标。增量综合问题是计算一个策略 σ，使得：*1.* σ 在假设 h的情况下对 φ_{org} 是制胜的；且 *2.* σ 对 φ_{new}是制胜的；如果这样的策略存在。在后一种情况下，我们称增量综合是可实现的；否则，它是不可实现的。

增量综合是标准综合的推广。事实上，我们有如下结论：针对 φ 的综合等价于针对 $\varphi_{org} = \varphi$、$h = \epsilon$ 和 $\varphi_{new} = true$的增量综合（或者，针对 $\varphi_{org} = true$、$h = \epsilon$ 和$\varphi_{new} = \varphi_{org}$的增量综合）。鉴于标准综合是双指数时间困难的（De Giacomo 和 Vardi 2015），我们可得出增量综合的困难性：

定理 1 增量综合相对于φ_{org} 和 φ_{new}是 *2*指数时间-难的。

事实上，增量综合也保留了原目标的可实现性。设 σ_{org} 是智能体在 h 发生且 φ_{new}到达时所采用的、针对 φ_{org} 的获胜策略；再设 σ_{new} 是满足定义 2 的策略。考虑通过组合σ_{org} 和 σ_{new}的执行而得到的整体策略：我们有，该策略是针对 φ_{org} 的获胜策略——也就是说，的可实现性得以保留。直观来说：在 h 发生之前，智能体采用σ_{org}，它总是保证满足 φ_{org}；在 h 发生之后，智能体采用 σ_{new}，它保证在 h的每种扩展中都满足 φ_{org}。形式化地说：记 $\sigma_{org}[h \leftarrow \sigma_{new}]$为一种策略，它在所有地方都与$\sigma_{org}$ 一致，但在 h 及其所有扩展中与 σ_{new}一致。那么我们有：

定理 2 策略 $\sigma_{org}[h \leftarrow \sigma_{new}]$，其中 σ_{org} 对 φ_{org} 是制胜的，且与 h 一致；而 σ_{new} 是满足定义 *2* 的策略，因此对 φ_{org} 制胜。

最后，我们指出，定义 2 可以轻松推广到任意数量的目标。例如，考虑原始目标 φ_{org}，以及两个依次到来的新目标 φ_{new_1}，它们分别在历史 h_1 和历史 $h_1 \cdot h_2$ 到达，其中 φ_{new_2} 到达历史 $h_1 \cdot h_2$。在这种情况下，增量综合需要计算一个策略，该策略：（i）在假设 $h_1 \cdot h_2$ 的情况下对 φ_{org} 制胜；（ii）在假设 h_2 的情况下对 φ_{new_1} 制胜；以及（iii）对 φ_{new_2} 制胜。

4 基于自动机的综合技术

本节中，我们介绍一种基于自动机的技术，用于解决增量综合。该技术基于在适当的确定性有限自动机上进行的可达性博弈求解，下面将对此作简要回顾。

一个确定有限自动机 (DFA)是一个元组 $\mathcal{A} = (\Sigma, S, \iota, \delta, F)$，其中：$\Sigma$ 是有限输入字母表；S 是有限状态集；$\iota \in S$ 是初始状态；$\delta: S \times \Sigma \to S$ 是转移函数；$F \subseteq S$ 是终态集。$\mathcal{A}$ 的大小为 $|S|$。为方便起见，我们将 δ 扩展为一个函数 $\delta: S \times \Sigma^* \to S$，其定义在有限轨迹 $\alpha = \alpha_0 \cdots \alpha_n$上，递归定义如下：(i) $\delta(s, \epsilon) = s$；以及 (ii)$\delta(s, \alpha_0 \cdots \alpha_n) = \delta(\delta(s, \alpha_0 \cdots \alpha_{n-1})\ \alpha_n)$。若轨迹 α 满足 $\delta(\iota, \alpha) \in F$，则称其被接受。$\mathcal{A}$的语言记作 $\mathcal{L}(\mathcal{A})$，是所有被 $\mathcal{A}$接受的轨迹组成的集合。若不存在其他 DFA $\mathcal{A}'$ 满足 $\mathcal{L}(\mathcal{A}) = \mathcal{L}(\mathcal{A}')$ 且 $|\mathcal{A}'| < |\mathcal{A}|$，则称 DFA $\mathcal{A}$是最小的。可以通过一种算法在多项式时间内将 DFA 最小化，该算法反复将那些对于所有可能的未来输入都不可区分的状态合并为一个状态。对于本文所考虑的语言，即正则语言，存在唯一的最小 DFA (Sipser 1997)。

定理 3 *(De Giacomo* 和 *Vardi 2015)* 给定一个LTLf公式 φ，我们可以构造一个确定有限自动机 $\mathcal{A}_\varphi$，其大小至多为 φ 的双指数级，并且其语言是满足 φ 的有限轨迹集合。

不过我们注意到，在实践中，构造 LTLf 公式的确定有限自动机时，最坏情况下的双指数级膨胀并不常见，而且这种构造具有良好的可扩展性，参见例如 (De Giacomo 和 Vardi 2015；Zhu 等人 2017年；Bansal 等，2020；De Giacomo 和 Favorito 2021)。

对于综合，我们将这样的确定有限自动机视为一个博弈。具体而言，确定有限自动机博弈是一个确定有限自动机 $\mathcal{G} = (\Sigma, S, \iota, \delta, F)$，其输入字母表为 $\Sigma = 2^{\mathcal{Y} \cup \mathcal{X}}$，其中 $\mathcal{Y}$ 和 $\mathcal{X}$ 是互不相交的原子命题集合，分别由智能体和环境控制。博弈策略是一个函数 $\kappa: S \to 2^{\mathcal{Y}}$，它将博弈状态映射到智能体动作。第 2 节中的对局概念同样适用于确定有限自动机博弈。若对局 $\tau = (Y_0 \cup X_0)(Y_1 \cup X_1) \cdots$ 满足以下条件，则称其为 κ-一致的：(i) $Y_0 = \kappa(\iota)$；以及 (ii) 对每个 $i > 0$，$Y_i = \kappa(\delta(\iota, \tau^{i-1}))$，其中 $\tau^{i-1} = (Y_0 \cup X_0) \cdots (Y_{i-1} \cup X_{i-1})$。$\kappa$- 一致对局的集合为 Play($\kappa$)。若对于 Play($\kappa$) 中的每个对局 $\tau \in$，都存在一个被 $\mathcal{G}$ 接受的有限前缀 τ^k，则称博弈策略 κ 在 $\mathcal{G}$ 中是获胜的。也就是说，在确定有限自动机博弈 $\mathcal{G}$ 中，智能体必须至少访问一次终态集合，从而只生成被 $\mathcal{G}$ 接受的轨迹。获胜区域 W 是状态 s 的集合，在这些状态下，智能体在博弈 $\mathcal{G}' = (\Sigma, S, s, \delta, F)$ 中拥有获胜策略；该博弈与 $\mathcal{G}$ 相同，但初始状态改为 s。在 $\mathcal{G}$ 中存在获胜博弈策略，当且仅当 $\iota \in W$。求解确定有限自动机博弈 $\mathcal{G}$，就是计算获胜区域 W，并在存在时求出获胜博弈策略 κ。确定有限自动机博弈可以在多项式时间内通过逆向归纳算法求解，该算法在博弈的状态空间上执行最小不动点计算 (Apt 和 Gr¨adel 2011)。给定博弈 $\mathcal{G}$ 和获胜博弈策略 $\kappa: S \to 2^{\mathcal{Y}}$，我们可以在多项式时间内构造相应 LTL$f$ 公式的获胜策略 $\sigma_{(\mathcal{G},\kappa)}: (2^{\mathcal{X}})^* \to 2^{\mathcal{Y}}$，方法如下：(i) $\sigma_{(\mathcal{G},\kappa)}(\epsilon) = \kappa(\iota)$；以及 (ii) 对每个 $\tau = (Y_0 \cup X_0) \cdots (Y_n \cup X_n)$，定义$\sigma_{(\mathcal{G},\kappa)}(X_0 \cdots X_n) = \kappa(\delta(\iota, \tau))$。有序对 $(\mathcal{G}, \kappa)$ 是实现策略$\sigma_{(\mathcal{G},\kappa)}$ 的转换器。

现在，我们介绍增量综合算法。直观地说，该算法由四个步骤组成，具体如下：步骤（1）构造接受满足原目标和新目标的轨迹的确定有限自动机；步骤（2）和（3）对这些确定有限自动机进行处理并将其组合起来，从而得到一个确定有限自动机博弈，当且仅当增量综合可实现时，该博弈才存在获胜策略；步骤（4）求解该博弈，以判断增量综合是否可实现——若可实现，则返回一个

同时满足原目标和新目标的策略。

该算法使用两个多项式算子来操作确定有限自动机：乘积和推进。前者将多个确定有限自动机合并为一个，其语言是各操作数语言的交集；后者使确定有限自动机的初始状态沿着历史 h推进，从而使推进后的确定有限自动机恰好接受输入确定有限自动机所接受且以 h为前缀的轨迹。形式化地说：

定义 *3*（确定有限自动机乘积）设 $\mathcal{A}_1$、$\cdots$、$\mathcal{A}_n$ 是确定有限自动机，其中对每个 i，$\mathcal{A}_i = (\Sigma, S_i, \iota_i, \delta_i, F_i)$。它们的乘积是确定有限自动机 $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n = (\Sigma, S, \iota, \delta, F)$，其：字母表为 Σ；状态空间为 $S = S_1 \times \cdots \times S_n$；初始状态为 $\iota = (\iota_1, \cdots, \iota_n)$；转移函数 $\delta((s_1, \cdots, s_n), w) = (\delta(s_1, w), \cdots, \delta(s_n, w))$，对每个 $w \in \Sigma$均成立；终态为 $F = F_1 \times \cdots \times F_n$；语言 $\mathcal{L}(\mathcal{A}) = \bigcap_i \mathcal{L}(\mathcal{A}_i)$。

定义 *4*（*DFA* 进展）设 $\mathcal{A} = (\Sigma, S, \iota, \delta, F)$ 是一个确定有限自动机，h 是一段历史。$\mathcal{A}$ 经 h 推进后得到的 *DFA* 是 $\mathcal{A}_h = (\Sigma, S, \delta(\iota, h), \delta, F)$——也就是说，它与 $\mathcal{A}$ 相同，但初始状态为 $\delta(\iota, h)$，因此其语言为 $\mathcal{L}(\mathcal{A}_h) = \{h \cdot \tau \mid \tau \in \mathcal{L}(\mathcal{A})\}$。

下面给出增量综合的算法。

算法 1。考虑一个增量综合实例：φ_{org} 是原始 LTL$_f$ 目标；h 是历史；φ_{new} 是新的 LTL$_f$ 目标。

(1) 构造确定有限自动机 $\mathcal{A}_{org}$ 和 $\mathcal{A}_{new}$，分别接受满足 φ_{org} 和 φ_{new} 的轨迹；(2) 构造确定有限自动机 $\mathcal{A}_{(org,h)}$，即 $\mathcal{A}_{org}$ 经 h 推进后得到的 DFA；(3) 构造确定有限自动机博弈 $\mathcal{G} = \mathcal{A}_{(org,h)} \times \mathcal{A}_{new}$。设 ι 是 $\mathcal{G}$ 的初始状态；(4) 求解 $\mathcal{G}$。令 W 为获胜区域，κ 为获胜博弈策略（若存在）：若 $\iota \notin W$，则返回增量综合不可实现；否则，返回策略 $\sigma_{(\mathcal{G},\kappa)}$，将其作为换能器 $(\mathcal{G}, \kappa)$。

下面我们证明算法 1 的正确性。回想一下，我们希望构造一个博弈，当且仅当增量综合可实现时，该博弈才存在获胜策略。

确定有限自动机博弈由 $\mathcal{G} = \mathcal{A}_{(org,h)} \times \mathcal{A}_{new}$ 得到；根据定义 3，$\mathcal{G}$ 接受的轨迹是同时被 $\mathcal{A}_{(org,h)}$ 和 $\mathcal{A}_{new}$ 接受的轨迹。因此，$\mathcal{G}$ 接受满足以下条件的轨迹 τ：(P1) $\tau \models \varphi_{new}$，根据定理 3；(P2) $h \cdot \tau \models \varphi_{org}$，根据定理 3 和定义 4。

接下来，我们证明 $\mathcal{G}$ 存在获胜策略，当且仅当增量综合可实现。假设 $\mathcal{G}$ 存在获胜策略 $\sigma_{(\mathcal{G},\kappa)}$：由 (P1) 和 (P2) 可知，$\sigma_{(\mathcal{G},\kappa)}$ 只生成满足以下条件的对局 τ：τ 有一个满足 φ_{new} 的有限前缀，并且 $\tau' = h \cdot \tau$ 有一个满足 φ_{org} 的有限前缀——也就是说，$\sigma_{(\mathcal{G},\kappa)}$ 满足定义 2，且增量综合可实现。反过来，假设 $\mathcal{G}$ 不存在获胜策略：由 (P1) 和 (P2) 可知，不存在满足定义 2 的策略——也就是说，增量综合不可实现。因此，我们证明了算法 1 是可靠且完备的，正式结论如下：

定理 4 当且仅当算法 *1* 返回一个策略时，增量综合才是可实现的。

关于复杂度：算法 1 的运行时间关于 h 的长度为多项式，因为它通过遍历 h 中的所有内容来执行 $\mathcal{A}_{org}$ 的转移函数，从而更新其初始状态。关于 φ_{org} 和 φ_{new}：算法 1 求解一个最坏情况下规模呈双指数增长的博弈；该博弈可在多项式时间内求解，因此算法 1 证明了增量综合关于 φ_{org} 和 φ_{new} 属于双指数时间复杂度类。根据定理 1 中已证明的增量综合的困难性，我们得出了增量综合的计算复杂度：

定理 5 φ_{org}、h和 φ_{new} 的增量综合为：

- *2*关于 φ_{org} 和 φ_{new}是指数时间完全的；
- 关于 h是多项式的。

最后，我们证明算法 1 可以扩展到任意数量的目标。令 φ_{org} 为原始目标，$h = h_1 \cdot \ldots \cdot h_n$ 为一段历史，φ_i 为一个新目标，它会在每个 $i \leq n$的 $h_1 \cdot \ldots \cdot h_i$末尾到达。算法 1 的主要修改如下：(1) 构造确定有限自动机 $\mathcal{A}_{org}$，它接受满足 φ_{org}的轨迹；并对每个 $i \leq n$构造 $\mathcal{A}_i$，它接受满足 φ_i 的轨迹；(2) 对每个 $i < n$计算确定有限自动机推进 $\mathcal{A}_{(org,h_1 \cdot \ldots \cdot h_n)}$和$\mathcal{A}_{(i,h_{i+1} \cdot \ldots \cdot h_n)}$（注意，最后一个新目标 φ_n 对应的确定有限自动机不进行推进）；(3) 构造并求解确定有限自动机博弈 $\mathcal{G} = \mathcal{A}_{(org,h_1 \cdot \ldots \cdot h_n)} \times \mathcal{A}_{(1,h_2 \cdot \ldots \cdot h_n)} \times \cdots \times \mathcal{A}_{(n-1,h_n)} \times \mathcal{A}_{n}$。上述关于可靠性和完备性的论证可通过归纳法扩展到考虑多个目标的情形；关于目标和历史的复杂度保持不变。

显然，采用增量综合的智能体无需每次收到添加新目标的请求时，都从头计算原始目标的确定有限自动机——这一操作对每个目标都要耗费双指数时间。相反，智能体可以将原始目标的确定有限自动机存储在内存中，并在执行过程中每个时间步推进其初始状态；当智能体收到添加新目标的请求时，只需计算新目标的确定有限自动机。回顾一下：*LTL$_f$* 综合的瓶颈在于确定有限自动机的构造（Zhu 等，2017；Bansal 等，2020；De Giacomo 和 Favorito 2021；De Giacomo 等人，2022）；因此，将目标的确定有限自动机存储在内存中，有望显著提升增量综合的性能。

这些特性使算法1非常适合高效实现，第7节的实证结果也证实了这一点。

5 公式推进

在本节中，我们表明增量综合可以通过LTL$_f$ 公式推进（又称 LTL$_f$ 推进）来求解（De Giacomo 等人，2022年；Gabbay 等人1980；Manna 1982；Emerson 1990；Bacchus 和 Kabanza 2000年）。直观地说，一个LTL$_f$ 公式的推进是一种句法重写，它在给定一段历史的情况下，刻画该公式在这段历史发生之后仍需满足的内容。

我们首先回顾 LTL$_f$ 推进。首先：LTL$_f$ 是在有限轨迹上解释的；因此有必要明确轨迹何时结束。为此，我们引入两个公式 $\Box(false)$ 和 $\Diamond(true)$，直观而言，它们分别表示有限轨迹结束和有限轨迹未结束。设 φ 是关于一组原子命题 P 的 NNF LTL$_f$ 公式，且 $w \in 2^P$ 是一个命题

解释。φ 经过 w 的推进，记作 $prog(\varphi, w)$，如下所示（De Giacomo 等人，2022年）：

$prog(p, w) \doteq true$ 如果 $p \in w$ 且 $false$ 否则 $prog(\neg p, w) \doteq true$ 如果 $p \notin w$ 且 $false$ 否则 $prog(\varphi_1 \vee \varphi_2, w) \doteq prog(\varphi_1, w) \vee prog(\varphi_2, w) prog(\varphi_1 \wedge \varphi_2, w) \doteq prog(\varphi_1, w) \wedge prog(\varphi_2, w) prog(\bigcirc\varphi, w) \doteq \varphi \wedge \Diamond(true) prog(\bullet\varphi, w) \doteq \varphi \vee \Box(false) prog(\varphi_1 \mathcal{U} \varphi_2, w) \doteq prog(\varphi_2, w) \vee (prog(\varphi_1, w) \wedge prog(\bigcirc(\varphi_1 \mathcal{U} \varphi_2) w) prog(\varphi_1 \mathcal{R} \varphi_2, w) \doteq prog(\varphi_2, w) \wedge (prog(\varphi_1, w) \vee prog(\bullet(\varphi_1 \mathcal{R} \varphi_2) w)$

设 $h = \alpha_0 \cdots \alpha_n \in (2^P)^*$ 为一个历史。我们将 LTL$_f$ 推进从 φ 扩展到 h，记为 $prog(\varphi, h)$，定义如下：(*i*) $prog(\varphi, \epsilon) = \varphi$；以及 (*ii*) $prog(\varphi, \alpha_0 \cdots \alpha_n) = prog(prog(\varphi, \alpha_0 \cdots \alpha_{n-1}) \alpha_n)$。直观而言，$\varphi$ 经由 h 的 LTL$_f$ 推进刻画了在 h 发生之后，φ 中仍有待满足的部分。

考虑一个增量综合实例：φ_{org} 是原始的 LTL$_f$ 目标；h 是历史；φ_{new} 是新的 LTL$_f$ 目标。利用 LTL$_f$ 推进，我们可以通过标准综合来求解增量综合：

定理6 策略 σ 能够求解针对 φ_{org}、h 和 φ_{new} 的增量综合，当且仅当 σ 能够求解针对 $prog(\varphi_{org}, h) \wedge \varphi_{new}$ 的标准综合。

接下来，我们研究通过 LTL$_f$ 推进来解决增量综合的复杂度。我们注意到，经过 φ_{org} 的推进 h 会生成一个 LTL$_f$ 公式 $prog(\varphi_{org}, h)$，其大小在最坏情况下呈指数级增长（De Giacomo 等人，2022年）；这一观察结合 LTL$_f$ 综合的复杂度，或许暗示通过 LTL$_f$ 推进来解决增量综合的复杂度为：相对于 φ_{org} 是 3EXPTIME，相对于 φ_{new} 是双指数时间。然而，相对于 φ_{org} 的界过于悲观。事实上，我们将在下文证明，任何经过推进的公式的确定有限自动机大小确实有上界，且不超过原公式对应的确定有限自动机大小——因此，通过 LTL$_f$ 推进来解决增量综合，相对于 φ_{org} 也属于双指数时间复杂度。

为证明这一点，我们分析 LTL$_f$ 推进与确定有限自动机推进之间的关系，见定义 4。设 $\mathcal{A}_\varphi$ 是 LTL$_f$ 目标 φ 对应的确定有限自动机；令 ι 和 δ 分别为 $\mathcal{A}_\varphi$ 的初始状态和转移函数。如果我们通过 w 将 φ 推进一步，这相当于在 $\mathcal{A}_\varphi$ 中执行相同的一步：对应的确定有限自动机为 $\mathcal{A}_\varphi$，但其初始状态已推进至 $\delta(\iota, w)$。通过归纳法，这一结果可推广到任意长度的历史：如果我们通过 h 将 φ 推进，那么它对应的确定有限自动机为 $\mathcal{A}_{(\varphi,h)}$ ——也就是说，与 $\mathcal{A}_\varphi$ 相同，但其初始状态已推进至 $\delta(\iota, h)$，如定义 4 所述。因此，我们得到：

定理 7 设：φ 是一个 LTL$_f$ 目标；$\mathcal{A}_\varphi$ 是其对应的确定有限自动机；h 是一段历史。该确定有限自动机在 $\mathcal{A}_{(\varphi,h)}$ 上对 $\mathcal{A}_\varphi$ 进行 h 推进，恰好接受满足 $prog(\varphi, h)$ 的轨迹。

定理 7 表明，尽管 $prog(\varphi, h)$ 在最坏情况下会随 φ 的大小呈指数级增长，但其对应的确定有限自动机可以在双指数时间内构造出来，而非 3EXPTIME——这与 $\mathcal{A}_{(\varphi,h)}$ 对 $\mathcal{A}_\varphi$ 经 h 推进得到确定有限自动机的情况相同。换句话说，最坏情况的

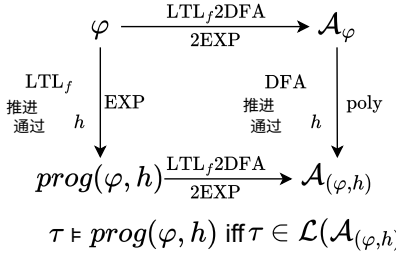

图 1：LTL$_f$ 推进与确定有限自动机在一段历史 h 上的推进之间的关系

由 LTL$_f$ 推进 $prog(\varphi, h)$ 引起的指数级膨胀，并未反映在其对应的最小确定有限自动机中；后者本质上与 $\mathcal{A}_\varphi$ 相同。图 1 总结了这一结果。

事实上，对 $\mathcal{A}_\varphi$ 的初始状态进行推进以得到 $\mathcal{A}_{(\varphi,h)}$，可能会导致 $\mathcal{A}_\varphi$ 的某些部分变得不可达，从而为进一步最小化创造机会。因此：

定理 8 设 $\mathcal{A}_\varphi$ 是公式 φ 的LTL$_f$ 确定有限自动机，且h是一段历史。我们有 $|\mathcal{A}_{(\varphi,h)}| \leq |\mathcal{A}_\varphi|$。

最后，我们说明，算法 1 可以看作是通过 LTL$_f$ 推进实现增量综合的一种巧妙方式。事实上，这可由定理 1、6 和 7 以及定义 3 和 4 得出：

推论 1 考虑一个增量综合实例：φ_{org} 是原始LTL$_f$ 目标；h 是历史；φ_{new} 是新的LTL$_f$ 目标。算法 *1* 的步骤 *(3)* 中构造的确定有限自动机DFA $\mathcal{G} = \mathcal{A}_{(org,h)} \times \mathcal{A}_{new}$（在多项式时间内）恰好接受满足 $prog(\varphi_{org}, h) \wedge \varphi_{new}$ 的轨迹。

6 实现

我们在一个名为 ISABEL-DP[1]（***I***增量***S***综合**D**FA ***P***进展）的工具中实现了扩展到多个目标的基于自动机的技术（算法 1）。ISABEL-DP 使用了文献（Zhu 等，2017）中的符号综合框架；该框架已集成到最先进的 LTL$_f$ 综合工具中（Zhu 和 Favorito 2025；Duret-Lutz 等人 2025）。我们使用 LYDIA（De Giacomo 和 Favorito 2021）从 LTL$_f$ 公式生成确定有限自动机；LYDIA 是性能最佳的 LTL 到确定有限自动机转换工具之一。进而，LYDIA 使用 MONA（Henriksen 等人 1995）表示确定有限自动机，并执行相关操作，例如乘积和最小化。MONA 使用确定有限自动机的半符号表示：状态采用显式表示，转移采用符号化表示。我们使用 SYFT 对确定有限自动机进行全符号化表示，并求解符号化确定有限自动机博弈（Zhu 等，2017）。SYFT 使用二元决策图（Bryant 1992）表示符号化确定有限自动机的状态空间和转移函数，底层的二元决策图库为 CUDD-3.0.0（Somenzi 1998）。

我们的实现同时采用 MONA 和 SYFT 对确定有限自动机的表示，以最大限度地发挥其优势

[1]https://github.com/GianmarcoDIAG/ISabel/

用于增量综合。具体而言：（1）在博弈构造和确定有限自动机推进过程中，采用 MONA 确定有限自动机表示，以最大限度地发挥确定有限自动机最小化的优势；（2）在博弈求解过程中，采用 SYFT 确定有限自动机表示，以最大限度地发挥符号化博弈求解技术的优势。

按照第4节概述的方法，ISABEL-DP 维护一个数据结构，用于存储集合 V 中新增目标的 MONA 确定有限自动机；在执行过程中，这些自动机的初始状态会逐步推进。当收到一个新目标 φ_{new} 时，实现会从 V 中取出先前新增目标的确定有限自动机；这样可以尽可能减少 LTL$_f$ 到确定有限自动机的转换次数——从而为每个已存储的目标节省双指数时间。具体而言，实现分为以下步骤：(1) 将 φ_{new} 转换为 MONA 确定有限自动机 $\mathcal{A}_{new}$；(2) 将 V 和 $\mathcal{A}_{new}$ 中的确定有限自动机构成乘积，构造最小 MONA 确定有限自动机博弈 $\mathcal{G}$；(3) 将 $\mathcal{G}$ 转换为 SYFT 符号化确定有限自动机博弈 $\mathcal{G}^s$；(4) 求解 $\mathcal{G}^s$，若增量综合可实现，则返回一个新策略，此时还会将 $\mathcal{A}_{new}$ 添加到 V。

7 实证分析

在本节中，我们对增量综合进行实证分析。分析旨在确定：在标准综合可行的情况下，增量综合是否也可行。此外，我们还希望确定，直接实现时，基于自动机的解决方案是否比基于公式推进的解决方案更有效。

为此，我们在一些可扩展的基准测试上，将 ISABEL-DP 的性能与一种基线实现进行比较。该实现通过 LTL$_f$ 推进来求解增量综合，名为 ISABEL-FP（增量综合（LTL$_f$ 公式推进））。为确保公平比较，它也使用 LYDIA 进行 LTL$_f$到确定有限自动机的转换，并使用 SYFT 进行符号博弈求解。我们将展示，ISABEL-DP的表现始终显著优于ISABEL-FP，从而证明基于自动机的方法是有效的。

实验方法。 我们考察了四个基准：两个由我们自行设计，即 PLANTS 和 REQUESTS；另外两个改编自用于 LTL$_f$ 综合和强完全可观测非确定性（FOND）规划的现有基准，即 COUNTER 和 TIREWORLD。[2] 这些基准之所以被选中，是因为它们允许智能体在执行过程中添加许多目标，因此适合用于评估增量综合实现。在每个基准中，一个实例 包含以下内容：实现最多运行 3600 秒，超时后停止；我们评估 ISABEL 的两种实现超时前添加的目标数量，以及添加每个目标所需的时间。为了复现增量设置，新目标的添加与执行针对先前目标综合出的策略交替进行：每个时间步，智能体执行一个动作并添加一个新目标。所有实验均在一台笔记本电脑上运行，其操作系统为 64 位 Ubuntu 22.04，配备 3.6 GHz CPU 和 8 GB 内存。

基准。 我们对每个基准进行说明。我们提供 TIREWORLD 和 PLANTS 基准的完整规范；REQUESTS 和 COUNTER 的完整规范请参阅补充材料。回顾第 2 节和第 3 节中的综合设置：原子命题分为智能体原子 $\mathcal{Y}$ 和环境原子 $\mathcal{X}$。

在我们的 TIREWORLD 基准中（Muise、McIlraith 和 Beck, 2012），智能体必须在若干位置之间移动，并应对移动时出现的非确定性结果。具体来说：智能体可以从位置 ℓ 移动到任意位置 ℓ'；智能体移动时，其中一个轮胎可能会以非确定性方式漏气（用 ℓ'表示）；轮胎漏气后，智能体无法移动，但可以更换漏气的轮胎（我们假设备用轮胎总是可用）。形式上，TIREWORLD 基准建模如下。智能体原子为：(*i*) at_i，表示智能体当前所在的位置；(*ii*)$move_i$，表示智能体移动到位置 i；以及 (*iii*)$change\text{-}tire$，表示智能体更换漏气的轮胎。环境原子只有一个：(*iv*) $make\text{-}flat$，表示环境导致智能体的一个轮胎漏气。TIREWORLD 基准中的实例由位置数量 ℓ参数化。我们假设环境只能在智能体移动时使轮胎漏气（而非在智能体更换漏气轮胎时），记作 LTL$f\alpha \doteq \Box(make\text{-}flat \supset \bigvee_{i<\ell} move_i)$。在每个实例中，我们生成目标 φ_n，其中 $n \leq 30$，具体如下：如果假设 α 成立，智能体必须依次经过从 0到 $\ell-1$的各个位置，总共访问 $2n+1$ 次，记作 LTL$_f$ $\varphi_n \doteq$

$$\alpha \supset (\Diamond(at_0 \wedge \bigcirc(\Diamond at_1 \wedge \cdots \wedge \bigcirc(\Diamond at_{\ell-1} \wedge \bigcirc(\Diamond at_0 \wedge \ \cdots \wedge \bigcirc(\Diamond at_{2n+1 \bmod \ell}) \ \cdot)))))$$

其中 n 是目标参数。原始目标（$n = 0$）还要求智能体满足其运行所在领域的规范——在本例中，该领域是一个全观察非确定性规划域，其中非确定性由环境原子 $make\text{-}flat$ 表示。领域规范包括**(C1)**初始状态：智能体位于位置 ℓ_0，写作：

$$\varphi_{init} \doteq at_0 \wedge \bigwedge_{i.0<i<\ell} \neg at_i$$

(C2)动作前提条件：在每个时间步，只有在没有轮胎漏气时，智能体才能移动；只有在轮胎漏气时，才能更换轮胎，写作 $\varphi_{pre} \doteq$

$\Box(\bigwedge_{i<\ell}(move_i \supset \neg flat\text{-}tire)) \wedge \Box(change\text{-}tire \supset flat\text{-}tire)$**(C3)** 互斥公理：在每个时间步，智能体必须恰好执行一个动作，写作

$$\varphi_{mutex} = \Box(change\text{-}tire \vee \bigvee_{i<\ell} move_i) \wedge \Box(change\text{-}tire \supset \neg \bigvee_{i<\ell} move_i) \wedge$$

[2]在许多人工智能应用中（包括规划），智能体拥有其运行环境的模型，例如一个 FOND 领域。世界模型通常被视为环境规范，且往往可以用 LTL$_f$表示。例如，关于 FOND 领域的情况，参见（Aminof 等人，2019）。为避免引入显式的环境规范，我们利用以下事实：智能体动作执行完毕且观察到环境反应后（例如，PDDL 互斥条件的某个效果已经发生），描述后继领域状态的流利符便完全确定（De Giacomo、Parretti 和 Zhu，2023）。因此，我们可以将流利符指定为智能体原子，并允许环境反应完全自由（但可能需要谨慎处理所有环境反应）。由此，我们可以将世界模型表示为原始智能体目标的一部分φ_{org}。这就是我们在 TIREWORLD 基准测试中采用的方法。

$\Box(\bigwedge_{i<\ell} move_i \supset \neg change\text{-}tire \wedge \neg \bigvee_{j\neq i} move_j)$**(C4)**转移函数：在每个时间步，智能体必须遵循领域的转移函数，该函数写作

$$\varphi_{trans} \doteq \Box(\bigwedge_{i\leq \ell}\bigwedge_{j\neq i} at_i \wedge move_j \wedge make\text{-}flat \supset \bullet(at_j \wedge flat\text{-}tire \wedge \bigwedge_{k\neq j} \neg at_k)) \wedge \Box(\bigwedge_{i\leq \ell}\bigwedge_{j\neq i} at_i \wedge move_j \wedge \neg make\text{-}flat \supset \bullet(at_j \wedge \neg flat\text{-}tire \wedge \bigwedge_{k\neq j} \neg at_k)) \wedge \Box(\bigwedge_{i\leq \ell} at_i \wedge change\text{-}tire \supset \bullet(at_i \wedge \neg flat\text{-}tire \wedge \bigwedge_{j\neq i} \neg at_j))$$

A每个目标的获胜策略1： l i

$$\varphi_{org} \doteq \varphi_{init} \wedge \varphi_{pre} \wedge \varphi_{mutex} \wedge \varphi_{trans} \wedge \varphi_0$$

对于每个目标，智能体的获胜策略如下：依次导航位置 0 到 $\ell-1$，以完成总计 $2n+1$ 次访问；如果环境导致轮胎爆胎，就更换轮胎并继续导航。我们考虑位置数量各异的 TIREWORLD 实例 ℓ，满足 $1 \leq \ell \leq 10$，记作 TIREWORLD-ℓ；在每个实例中，我们考虑满足 $n \leq 30$ 的目标，共计 300 个目标。

我们指出，生成 FOND 领域的确定有限自动机还有更有效的方法，而不必将其 LTL$_f$ 规范转换为确定有限自动机。例如，(De Giacomo、Di Stasio 和 Parretti 2025) 展示了一种 FOND 到 DFA 转换技术，不会产生 LTL$_f$ 到 DFA 转换带来的双指数时间膨胀，其代价仅为领域规模上的指数时间。尽管如此，为了与第 2 节和第 3 节介绍的综合设置保持一致，我们仍将领域规范表示为 LTL$_f$ 公式。

在 COUNTER 基准（Zhu 等人 2020年）中，规范如下：智能体维护一个 k位计数器，所有位初始均设为 0；在每个时间步，环境选择是否发出计数器递增请求，该请求由环境原子 add表示；智能体选择是否接受请求，然后递增计数器。COUNTER 基准中的实例以计数器的位数 k 为参数。对于每个实例，我们按如下方式生成目标 φ_n：如果环境总是发出递增请求，智能体就必须最终将计数器的值设为 $2n+1$，其中 n是目标参数。一个这样的目标在 LTL$_f$ 中写作 $\varphi_n \doteq (\Box add) \supset \Diamond\varphi_{2n+1}^{counter}$，其中 $\varphi_{2n+1}^{counter}$是将计数器值设为 $2n+1$的智能体原子的合取。原始目标还要求智能体满足计数器规范。对于每个目标，一种获胜策略是接受所有递增请求，并相应地递增计数器。我们考虑位数为k 且满足 $6 \leq k \leq 10$的 COUNTER 实例，记作 COUNTER-k；在每个实例中，我们考虑满足 $n \leq 30$的目标，共计 150 个目标。

在 PLANTS 基准中，规格如下：智能体必须照料若干植物；智能体可以给植物浇水，这由智能体原子 $water$ 表示；植物 i 存活由环境原子 $alive_i$ 表示；环境还可以降雨，由环境原子 $rain$ 表示。PLANTS 基准中的实例以植物数量 p 参数化。在每个时间步，植物 i 存活，当且仅当它在上一个时间步存活，且智能体给它浇了水或环境下了雨，记作：

$$\alpha_i \doteq \Box(\bullet(alive_i) \equiv alive_i \wedge (water \vee rain))$$

在每个实例中，我们按如下方式生成目标：如果假设 α_i 成立，智能体必须照料植物 i，持续

实例	新增目标 平均运行时间（秒） ISABEL-DP	ISABEL-FP	ISABEL-DP	ISABEL-FP
PLANTS-1	**30**/30	19/30	**9.12**	155.12
PLANTS-2	**25**/30	16/30	**15.68**	172.00
PLANTS-3	**22**/30	14/30	**18.70**	179.43
PLANTS-4	**20**/30	13/30	**19.60**	173.17
PLANTS-5	**19**/30	12/30	**29.94**	265.45
PLANTS-6	**18**/30	12/30	**24.85**	216.06
PLANTS-7	**17**/30	11/30	**35.37**	268.26
植物-8	**16**/30	11/30	**35.37**	268.26
植物-9	**15**/30	10/30	**32.49**	190.29
PLANTS-10	**15**/30	10/30	**42.09**	237.39
COUNTER-6	**25**/30	20/30	**25.02**	29.00
计数器-7	**21**/30	12/30	**3.98**	13.18
COUNTER-8	**19**/30	10/30	**14.57**	41.11
COUNTER-9	**0**/30	**0**/30	-	-
COUNTER-10	**0**/30	**0**/30	-	-
REQUESTS-1-1	**36**/40	28/40	**27.53**	107.01
请求-1-2	**36**/40	29/40	**31.91**	117.05
请求-1-3	**36**/40	29/40	**31.41**	115.76
请求-1-4	**36**/40	29/40	**31.80**	111.20
请求-2-1	**37**/40	29/40	**24.39**	112.03
请求-2-2	**37**/40	29/40	**23.83**	112.27
请求-2-3	**36**/40	29/40	**24.49**	105.88
请求-2-4	**36**/40	29/40	**27.97**	105.64
REQUESTS-3-1	**37**/40	29/40	**24.12**	113.95
REQUESTS-3-2	**36**/40	29/40	**24.85**	108.55
REQUESTS-3-3	**35**/40	29/40	**33.81**	111.29
REQUESTS-3-4	**6**/40	**6**/40	7.09	**6.43**
TIREWORLD-1	**30**/30	26/30	**5.52**	118.14
轮胎世界-2	**30**/30	26/30	**5.42**	121.49
轮胎世界-3	**30**/30	25/30	**5.06**	141.56
轮胎世界-4	**30**/30	24/30	**9.73**	117.57
轮胎世界-5	**28**/30	21/30	**17.89**	152.42
TIREWORLD-6	**21**/30	19/30	**32.04**	151.25
TIREWORLD-7	**17**/30	**17**/30	**41.77**	157.20
轮胎世界-8	**15**/30	**15**/30	**34.05**	98.96
轮胎世界-9	**13**/30	**13**/30	**18.95**	47.23
TIREWORLD-10	**12**/30	**12**/30	**15.97**	36.65
总计	**892**/1230	692/1230		

表 1：ISABEL-DP 和 ISABEL-FP 添加的目标数量以及实现一个目标所需的平均时间比较。表现最佳的实现结果以粗体显示。

3($n+1$) 天，每天对应一个独立的时间步，其中 n是目标参数。这样的一个目标写作

$$\varphi_n \doteq \bigwedge_{i<p} \alpha_i \supset \varphi^{alive}{}_{3\ n+1\ ()}^{\ i} \text{ 与 } \varphi^{alive}{}_{i3(n+1)} \doteq \Diamond(\bigwedge_{j<3(n+1)} \bigcirc^{\ j} alive_i)$$

其中，$\bigcirc^k$ 表示 k 嵌套 $\bigcirc$ 算子。对于每个目标，获胜策略都是总是给植物浇水。我们考虑植物数量为 ps.t. $1 \leq p \leq 10$的PLANTS实例，记为PLANTS-p；在每个实例中，我们考虑$n \leq 30$个目标，总计300个目标。

在 REQUESTS基准中，规范如下：智能体提供一组服务，环境可以在每个时间步请求这些服务；要满足请求，智能体必须执行特定的动作序列；环境会发出有限个连续请求，智能体必须满足所有请求。REQUESTS基准中的实例由以下参数决定：(1) 智能体提供的服务数量为 i s_i；(2) 智能体必须执行的动作数量为 j $a_{i,\ j}$ 以便在收到对服务 i的请求时提供该服务。对于每个实例，我们按如下方式生成目标：智能体必须满足 $2n+1$ 个环境请求，其中 n 为目标参数。对于每个目标，获胜策略是执行所有动作，以满足环境发出的请求。我们考虑满足以下条件的 REQUESTS实例：(*i*) 服务数量为 i，且 $1 \leq i \leq 3$；(*ii*) 动作数量为 j，且 $1 \leq j \leq 4$；此类实例记为 REQUESTS-*i*-*j*；在每个实例中，我们考虑满足 $n \leq 40$的目标，共计 480 个目标。

实证结果。表 1 比较了 ISABEL-DP 和 ISABEL-FP 添加的目标数量，以及它们添加新目标的平均运行时间；为确保公平比较，每个实例的平均耗时均根据两者共同添加的目标计算。首先，我们注意到，ISABEL-DP 和 ISABEL-FP 在 COUNTER-9 和 COUNTER-10 中均未能添加任何目标：这是因为两者在构造计数器规范的确定有限自动机时都超时了，而该自动机的规模会随位数呈指数增长。对于 REQUESTS-3-4，我们也观察到类似结果：ISABEL-DP 和 ISABEL-FP 均在构造第 7 个目标的确定有限自动机时超时。这些结果证实，确定有限自动机构造是增量综合的瓶颈，与标准综合中的情况相同。

在其余实例中，我们得到如下结果：（*i*）就新增目标的数量而言，ISABEL-DP 的表现始终优于或等同于 ISABEL-FP；（*ii*）就添加一个新目标的平均时间而言，ISABEL-DP 显著优于 ISABEL-FP——平均性能最多可提升一到两个数量级。这在 PLANTS 和 REQUESTS 中十分明显：在它们的实例中，已推进的 LTL$_f$ 目标规模可能呈指数增长。要添加一个新目标，ISABEL-FP 会对由已推进的 LTL$_f$ 目标合取得到的 LTL$_f$ 公式执行 LTL$_f$到确定有限自动机的转换——这可能造成不必要的开销；另一方面，ISABEL-DP 仅对新的 LTL$_f$ 目标执行 LTL$_f$到确定有限自动机的转换，并利用 DFA 进展在多项式时间内生成已推进的 LTL$_f$ 目标的确定有限自动机——从而避免产生任何不必要的开销。

我们在 COUNTER 实例中也得到了类似的结果。不过，需要注意以下几点：在 COUNTER 实例中，推进后的 LTL$_f$ 目标的大小与未推进目标的大小几乎相同。尽管如此，无论是新增目标数量还是新增一个目标所需的平均时间，IS- ABEL-DP 的表现都优于 ISABEL-FP。原因如下：ISABEL-DP 可以从内存中取回各目标的确定有限自动机，而 ISABEL-FP 则需要对由推进后的 LTL$_f$ 目标合取而成的 LTL$_f$ 公式执行 LTL$_f$-到确定有限自动机的转换，因此产生了不必要的开销。因此，将各目标的确定有限自动机存储在内存中，可以提升增量综合的性能。

关于 TIREWORLD 基准测试，我们有以下结论。(1) 从 TIREWORLD-1 到 TIREWORLD-6 的实例表明，无论是新增目标的数量，还是新增一个目标所需的平均时间，ISABEL-DP 的表现都优于 ISABEL-FP——平均性能最多可提升一个或两个数量级——因为 ISABEL-DP 避免了使用 LTL$_f$ 推进所产生的不必要开销。(2) 从 TIREWORLD-7 到 TIREWORLD-10 的实例表明，ISABEL-DP 和 ISABEL-FP 新增的目标数量相同；尽管 ISABEL-DP 仍比 ISABEL-DP 快得多，但 ISABEL-DP 与 ISABEL-FP 之间的性能差距有所缩小。后一结果是由于运行实验的机器受到内存限制（8 GB 随机存取存储器）。事实上，随着内存用量接近上限，内存管理开销对 ISABEL-DP 和 ISABEL-FP 都变得显著，直到二者在构建确定有限自动机博弈竞技场以计算新策略的过程中都耗尽内存。请注意，ISABEL-DP 和 ISABEL-FP 构建相同的确定有限自动机博弈竞技场（见定理 6 和推论 1）：因此，观察到的性能差异源于 ISABEL-DP 采用了效率更高的确定有限自动机博弈竞技场构建方法；这也表明，在更强大的硬件上运行时，面对大型 TIREWORLD 实例，ISABEL-DP 的扩展性会优于 ISABEL-FP。

总体而言，实证分析表明 ISABEL-DP 优于 ISABEL-FP，这证实了我们提出的增量综合方法的有效性，也表明该方法适用于标准综合同样可行的场景。（进一步的实证结果另见补充材料。）

8 结论

除了具有理论意义和普遍意义外，这项工作还具有直接的实际影响：它为在目标动态变化时修订智能体的策略提供了一种有效方法。我们的研究结果提供了一种高效的增量技术，用于修订在非确定性领域中运行的自主智能体的策略，从而尽可能减少目标动态变化时所需的综合工作量。在此背景下，该方法可以直接集成到目标/意图管理系统中，例如 (De Giacomo 等人 2025) 中所描绘的系统。

请注意，新目标可能与当前目标不一致，从而导致不存在能够保证实现智能体所有目标的策略。尽管本文未探讨这一问题，我们的技术仍可有效检测冲突，这是解决冲突的第一步。接下来，如果智能体的所有目标具有相同的重要性，可以尝试寻找可实现的最大子集，不过这本身就是一个指数级问题，因为可能需要尝试指数多个组合，而每个组合都是一个增量综合问题。一种简单的替代方案是假设采用的目标按照优先级构成全序；在此假设下，保留优先级最高且能够共同实现的目标集合。

需要注意的是，我们的方法能够在 LTL$_f$ 中表达广泛的时序扩展目标/约束。这对于要求对安全/正确行为提供形式化保证的安全关键应用至关重要（Ferrando 等人，2021）。主流的自主智能体开发方法基于 BDI 智能体编程（Rao，1996年；Bordini、H¨ubner 和 Wooldridge，2007），包括支持目标推理的方法（如 Harland 等人，2014），但这些方法无法确保智能体的目标/意图始终保持一致/可实现，也无法提供此类安全性/正确性保证。

未来的工作中，我们计划研究基于综合与规划中采用的前向搜索技术进行增量综合的方法（Camacho 和 McIlraith，2019；Bonassi 等人，2023；Muise、McIlraith 和 Beck，2024；Duret-Lutz 等人，2025）。增量综合可以向多个方向扩展，例如，在不存在获胜策略时综合尽力而为／足够好策略（A minof、De Giacomo 和 Rubin，2021；De Giacomo、Parretti 和 Zhu，2023；Almagor 和 Kupferman，2020），或在环境动态发生变化时进行综合（Aminof 等人，2024）；还可以扩展到部分可观测性情形（Tabajara 和 Vardi，2020）。这些也都是未来值得探讨的课题。

致谢

本研究获得了英国研究与创新署（UKRI）埃尔兰根人工智能数学与计算基础中心（项目编号：EP/Y028872/1）、罗马第一大学意大利国家人工智能博士项目、加拿大国家科学与工程研究委员会以及约克大学的支持。

参考文献

Almagor, S.; and Kupferman, O. 2020年. 尽力综合. 载于*CAV*. Aminof, B.; De Giacomo, G.; Murano, A.; and Rubin, S. 2019. LTL 环境规范下的规划. 载于*ICAPS*. Aminof, B.; De Giacomo, G.; Parretti, G.; and Rubin, S. 2024. 多层环境中的 LTL$_f$ 尽力综合方法. 载于 *IJCAI*. Aminof, B.; De Giacomo, G.; and Rubin, S. 2021. 尽力综合：尽最大努力并不比放弃更难. 载于*IJCAI*. Apt, K. R.; and Gr¨adel, E., 编. 2011. 计算机科学家的博弈论讲义. 剑桥大学出版社. Bacchus, F.; and Kabanza, F. 2000. 使用时序逻辑表达规划中的搜索控制知识. *AIJ*. Baier, C.; and Katoen, J. 2008. 模型检测原理. 麻省理工学院出版社. Baier, J. A.; Fritz, C.; and McIlraith, S. A. 2007. 在先进规划器中利用程序化领域控制知识. 载于*ICAPS*. Bansal, S.; De Giacomo, G.; Di Stasio, A.; Li, Y.; Vardi, M. Y.; and Zhu, S. 2022. 组合式安全 LTL 综合. 载于 *VSTTE*. Bansal, S.; Li, Y.; Tabajara, L. M.; and Vardi, M. Y. 2020. 基于有限时域规范进行反应式合成的混合组合推理. 载于 *AAAI*. Bonassi, L.; De Giacomo, G.; Favorito, M.; Fuggitti, F.; Gerevini, A. E.; and Scala, E. 2023. 面向纯过去线性时序逻辑目标的 FOND 规划. 载于*ECAI*. Bonassi, L.; Gerevini, A. E.; Percassi, F.; and Scala, E. 2021. 通过编译消除，在 PDDL3 中处理定性状态轨迹约束的规划. 载于*ICAPS*. Bonassi, L.; Gerevini, A. E.; and Scala, E. 2022. 在 PDDL 中处理定性动作轨迹约束的规划. 载于*IJCAI*, 4606–4613. Bordini, R. H.; H¨ubner, J. F.; and Wooldridge, M. J. 2007.使用 *Jason* 编程 *AgentSpeak* 多智能体系统. 约翰·威立父子公司有限公司. Bryant, R. E. 1992. 使用有序二叉决策图进行符号布尔运算. *ACM* 计算综述Camacho, A.; and McIlraith, S. A. 2019. 使用 LTL 和 LTL$_f$目标进行强完全可观测非确定性规划. 载于*IJCAI*. Camacho, A.; Triantafillou, E.; Muise, C. J.; Baier, J. A.; and McIlraith, S. A. 2017. 带时序扩展目标的非确定性规划：有限轨迹和无限轨迹上的 LTL. 载于*AAAI*。

Cimatti, A.；Pistore, M.；Roveri, M.；以及 Traverso, P. 2003. 通过符号模型检测实现弱规划、强规划与强循环规划。*AIJ*。De Giacomo, G.；Di Stasio, A.；以及 Parretti, G. 2025. 从PDDL到确定有限自动机：面向有效推理的符号变换。载于*TIME*。De Giacomo, G.；以及 Favorito, M. 2021. 将LTL$_f$/LDL$_f$ 转换为确定有限自动机的组合式方法。载于*ICAPS*。De Giacomo, G.；Favorito, M.；Li, J.；Vardi, M. Y.；Xiao, S.；以及 Zhu, S. 2022. 将LTL$_f$ 综合视为与或图搜索：知识编译的应用。载于 *IJCAI*。De Giacomo, G.；Lesp´erance, Y.；Parretti, G.；Patrizi, F.；以及 Schram, R. 2025. 利用有限轨迹线性时序逻辑综合管理智能体不断变化的意图。载于*AAMAS*。De Giacomo, G.；Parretti, G.；以及 Zhu, S. 2023. 非确定性规划领域中的LTL$_f$ 尽力综合。载于 *ECAI*。De Giacomo, G.；以及 Rubin, S. 2018. 面向 LTL$_f$ 与LDL$_f$ 目标的FOND 规划的自动机理论基础。载于 *IJCAI*。De Giacomo, G. ；以及 Vardi, M. Y. 2013. 有限轨迹上的线性时序逻辑与线性动态逻辑。载于 *IJCAI*，854–860。De Giacomo, G.；以及 Vardi, M. Y. 2015. 有限轨迹上的LTL和LDL综合。载于 *IJCAI*，1558–1564。Duret-Lutz, A.；Zhu, S.；Piterman, N.；De Giacomo, G.；以及 Vardi, M. Y. 2025. 构建LTL$_f$ 综合工具。载于*CIAA*。Emerson, E. A. 1990. 时序逻辑与模态逻辑。载于理论计算机科学手册，第*16*章。Ferrando, A.；Dennis, L. A.；Cardoso, R. C.；Fisher, M.；Ancona, D.；以及 Mascardi, V. 2021. 迈向自主认知系统验证与确认的整体方法。*ACM*软件工程方法学汇刊Finkbeiner, B.；Klein, F.；以及 Metzger, N. 2022. 实时综合。创新系统软件工程Gabbay, D. M.；Pnueli, A.；Shelah, S.；以及 Stavi, J. 1980. 论公平性的时序分析。载于*POPL*。Harland, J.；Morley, D. N.；Thangarajah, J.；以及 Yorke-Smith, N. 2014. BDI智能体中目标生命周期的操作语义。自主智能体与多智能体系统Henriksen, J. G.；Jensen, J. L.；Jørgensen, M. E.；Klarlund, N.；Paige, R.；Rauhe, T.；以及 Sandholm, A. 1995. Mona：实践中的一元二阶逻辑。载于 *TACAS*。Kankariya, Y.；以及 Bansal, S. 2024. 有限轨迹线性时序逻辑到确定有限自动机的组合式转换中的分解。载于*AAAI*。Manna, Z. 1982. 顺序程序验证：时序公理化，53–102。Springer。Muise, C.；McIlraith, S. A.；以及 Beck, J. C. 2024. PRP重启：推进FOND 规划的技术现状。载于 *AAAI*。Muise, C. J.；McIlraith, S. A.；以及 Beck, J. C. 2012. 利用状态相关性改进非确定性规划。载于*ICAPS*。

Pnueli, A. 1977年. 程序的时序逻辑。载于 *SFCS*。Pnueli, A.；以及 Rosner, R. 1989年. 论反应式模块的综合。载于 *POPL*。Rao, A. S. 1996年. AgentSpeak(L)：BDI智能体以逻辑可计算语言发声。载于*MAAMAW*。Sipser, M. 1997年. 计算理论导论。PWS Publishing Company。Somenzi, F. 1998年. CUDD：CU决策图软件包释放2.3.0。载于 科罗拉多大学博尔德分校。Tabajara, L. M.；以及 Vardi, M. Y. 2020年. 部分可观测性下的LTL$_f$ 综合：从理论到实践。载于*GandALF*。Zhu, S.；以及 De Giacomo, G. 2022. 履行你的义务，但维护你的权利。载于 *KR*。Zhu, S.；De Giacomo, G.；Pu, G.；以及 Vardi, M. Y. 2020年. 带公平性与稳定性假设的LTL$_f$ 综合。载于*AAAI*。Zhu, S.；以及 Favorito, M. 2025. LydiaSyft：面向LTL$_f$ 规范的组合式符号综合框架。载于*TACAS*。Zhu, S.；Tabajara, L. M.；Li, J.；Pu, G.；以及 Vardi, M. Y. 2017年. 符号LTL$_f$ 综合。载于 *IJCAI*。

补充材料

补充材料的内容如下：第9节提供了第7节中介绍的基准的更多细节；第10节提供了针对基准中每个实例的更多实证结果，并进行进一步讨论。

9 基准的更多细节

首先，我们给出 COUNTER 和 REQUESTS 基准的完整规范。

我们回顾 COUNTER 基准的规范：智能体维护一个 k 位计数器，其所有位最初都设为 0；在每个时间步，环境选择是否为计数器发出递增请求；智能体选择是否接受该请求，然后递增计数器。COUNTER 基准中的实例由计数器中的位数 k 参数化。形式上，COUNTER 基准建模如下。存在一个环境原子：（*i*） add，表示环境为计数器发出递增请求。智能体原子包括：（*ii*） k 原子命题 $\{b_0, \cdots, b_{k-1}\}$，表示计数器各位的值；以及（*iii*） $k+1$ 原子命题$\{c_0, \cdots, c_k\}$，表示进位位，其中 c_0 表示智能体接受了递增请求。在每个实例中，我们按如下方式生成目标 φ_n：如果环境总是发出递增请求，智能体最终必须将计数器值设为 $2n+1$，其中 n 是目标参数。这样的一个目标在 LTL$_f$ 中写作$\varphi_n \doteq (\Box add) \supset \Diamond \varphi_{2n+1}^{counter}$，其中 $\varphi_{2n+1}^{counter}$是一个合取，由设置计数器值为 $2n+1$的位原子构成。原始目标（$n=0$）还要求智能体满足计数器规范——该规范规定智能体应如何更新各位和进位位的值。计数器规范包括：**(C1)** 初始状态：在初始时间步，每个位和进位位的取值均为 0，记作：

$$\varphi_{init} \doteq \bigwedge_{i<k} \neg b_i \wedge \bigwedge_{i<k} \neg c_i$$

(C2) 动作前提：在每个时间步，智能体只有在环境于前一个时间步发出递增请求时，才能接受递增请求，记为：

$$\varphi_{pre} \doteq \Box(\bullet c_0 \supset add)$$

(C3)转移函数：在每个时间步，智能体必须根据计数器的规范更新各位的值，记为

$$\varphi_{trans} \doteq \bigwedge_{i<k} \Box(B_i)$$

其中，对于每个 $i<k$，都有 $B_i \doteq$

$$\begin{array}{c}(\neg c_i \wedge \neg b_i) \supset (\bullet(\neg b_i \wedge \neg c_{i+1})) \wedge \\ (\neg c_i \wedge b_i) \supset (\bullet(b_i \wedge \neg c_{i+1})) \wedge \\ (c_i \wedge \neg b_i) \supset (\bullet(b_i \wedge \neg c_{i+1})) \wedge \\ (c_i \wedge b_i) \supset (\bullet(\neg b_i \wedge c_{i+1}))\end{array}$$

因此，原始目标为：

$$\varphi_{org} \doteq \varphi_{init} \wedge \varphi_{pre} \wedge \varphi_{trans} \wedge \varphi_0$$

应注意以下几点：在每个时间步，LTL$_f$ *COUNTER* 实例中目标的推进结果COUNTER 其大小与未推进的目标几乎相同。事实上，在每个时间步，目标的推进结果由以下部分组成：(*i*) 一个需要在当前时间步求值的命题公式——它在推进后将变为后续时间步的 $true$ 或 $false$；以及 (*ii*) 一个将在后续时间步求值的时序公式。例如，观察 φ_n for every w 的推进结果具有以下形式：

$$prog(\varphi_n, w) = (add \wedge prog(\bullet(\Box(add)), w)) \supset (\varphi_{2n+1}^{counter} \vee prog(\bigcirc(\Diamond(\varphi_{2n+1}^{counter})), w))$$

原始目标的推进同样适用。

我们回顾一下REQUESTS基准的规范：智能体提供一组服务，环境可以在每个时间步请求这些服务；为满足一个请求，智能体必须执行特定的动作序列；环境发出有限数量的连续请求，智能体必须满足所有请求。REQUESTS基准中的实例由以下参数确定：(1) 智能体提供的服务数量 $i\, s_i$；以及(2) 智能体为提供服务 i 以响应对该服务的请求而必须执行的动作数量 $j\, a_{i,\, j}$。形式化地说，REQUESTS基准定义如下。其中包括：(*i*) i 个环境原子 r_i，每个原子表示环境请求了服务 i；以及(*ii*) $i \times j$ 个智能体原子$a_{i,\, j}$，每个原子表示智能体为满足对服务 i 的请求所需执行的第 j 个动作。我们考虑以下公式：

(F1) 环境发出一系列 k 请求，记为：

$$\mathcal{E}_k \doteq (\bigwedge_{v<k} \bullet^v (\bigvee_{s<i} r_s)) \wedge \bullet^k (\Box(\neg \bigvee_{s<i} r_s))$$

其中 $\bullet$ 表示 k 嵌套的 $\bullet$ 算子。

(F2) 对于每种请求 i，智能体都必须执行一个特定的动作序列 j，记作：

$$\varphi_{request} \doteq \bigwedge_{s<i} r_s \supset (\Diamond(a_{s,1} \wedge \bigcirc(\Diamond(a_{s,2}) \wedge \cdots \wedge \bigcirc(\Diamond(a_{s,j})) \cdots)$$

(F3) 智能体生成的轨迹长度必须至少为 k，以确保读取环境生成的所有请求；这由以下条件强制实现：

$$\varphi_{2n+1}^{true} \doteq \bigcirc^k (true)$$

其中 $\bigcirc^k$ 表示 k 嵌套的 $\bigcirc$ 算子。在每个请求实例中，我们按如下方式生成目标：如果环境生成一个由 $2n+1$ 个请求组成的序列，智能体必须满足所有请求，记作：$\varphi_n \doteq \mathcal{E}_{2n+1} \supset (\varphi_{request} \wedge \varphi_{2n+1}^{true})$。

需要注意以下几点：在 REQUESTS 基准测试中，已推进的 LTL$_f$ 目标的规模可能相对于未推进的目标呈指数级增长。要理解这一点，请观察：在每个时间步，当环境发出特定请求时，蕴含式 $\varphi_{request}$ 前件的 LTL$_f$ 推进会因 $\Diamond$、$\bigcirc$ 和 $\wedge$的嵌套而呈指数级增长。

事实上，我们也在 PLANTS 和 TIREWORLD 基准测试中观察到了类似的结果。

10 其他实证结果

本节针对基准测试中的每个实例给出更多实证结果。为方便起见，我们按基准测试对实证结果进行分组展示。此外，我们指出，本节中的图表采用对数刻度，以提高可读性。

TIREWORLD。图2比较了ISabel-DP和ISabel-FP在每个TIREWORLD实例中添加一个目标所需的时间。我们观察到，在每一个TIREWORLD实例中，ISabel-DP添加新目标所需的时间都少于ISabel-FP——往往能快一到两个数量级，更

好的性能。事实上，随着位置数量 ℓ的增加，ISabel-DP与ISabel-FP之间的性能差距会变小；这是由于我们在论文正文第7节中讨论的内存管理开销。尽管如此，ISabel-DP所采用的更优DFA博弈构造技术表明，在更强大的硬件上，它在大型TIREWORLD实例中的扩展性也优于ISabel-FP。

关于 ISABEL-DP，需要注意的是，添加第一个（即原始）LTL$_f$ 目标（其中还包括 LTL$_f$ 领域规范）所需的时间，比添加某些后续 LTL$_f$ 目标所需的时间高出几个数量级。原因在于，原始目标的规模主要由构成领域规范的合取项的规模决定；后续新增的目标不包含领域规范，因此其规模通常远小于原始目标。ISABEL-DP 添加原始目标与添加某些后续目标所需时间之间的性能差距，正是因为与原始目标的确定有限自动机相比，后续目标的确定有限自动机可以更高效地构建。事实上，在大型 TIREWORLD 实例中，这种性能差距更加明显，因为领域规范的规模更大。

请注意，ISABEL-FP 并未表现出相同行为：添加原始目标所需的时间与添加某些后续目标所需的时间处于同一数量级。实际上，为了添加新目标，ISABEL-FP 会构建由已推进的 LTL$_f$ 规范合取而成的规范的确定有限自动机——也就是说，每次 ISABEL-FP 添加新目标时，都会从头构建部分 LTL$_f$ 领域规范的确定有限自动机。因此，即使某些后续目标的规模比原始目标小几个数量级，ISABEL-FP 仍会承担不必要的开销，再次构建领域规范的确定有限自动机。

COUNTER。图3比较了 ISABEL-DP 和 ISABEL-FP 在每个 COUNTER 实例中添加一个目标所需的时间。我们观察到，在每个 COUNTER 实例中，ISABEL-DP 添加目标的速度都快于 ISABEL-FP。此外，我们还注意到，在 COUNTER 实例中，ISABEL-DP 添加原始目标与添加某些后续目标所需时间之间的性能差距同样存在，尽管与 TIREWORLD 实例相比，这种差距不那么明显。结合第9节所述 COUNTER 实例中已推进 LTL$_f$ 目标的规模几乎与未推进目标相同这一事实，可以得出结论：ISABEL-DP 与 ISABEL-FP 之间的性能差距主要是因为 ISABEL-DP 能够受益于将先前添加的 LTL$_f$ 目标的确定有限自动机存储在内存中。

PLANTS 和 **REQUESTS**。图4和图5分别比较了在每个 PLANTS 和 REQUESTS 实例中添加一个目标所需的时间，比较对象为 ISABEL-DP 和 ISABEL-FP。结果表明，添加的目标数量较少时，ISABEL-DP 和 ISABEL-FP 的表现可能不相上下；然而，如果执行过程中需要添加许多目标，那么 IS- ABEL-DP 显然更胜一筹（但 REQUESTS-3-4 例外，构建第 7 个目标的确定有限自动机时，ISABEL-DP 和 ISABEL-FP 均超时）。在该实例中，构建第 7 个目标的确定有限自动机时，ISABEL-DP 和 ISABEL-FP 均超时）。这是因为：在最初的几个时间步中，推进后的 LTL$_f$ 目标规模不会激增，因此 ISABEL-DP 和 ISABEL-FP 的性能相近；然而，经过一定数量的时间步后，推进后的 LTL$_f$ 目标规模会大幅增加，因此 ISABEL-FP 所产生的开销开始显现，导致添加许多目标时，ISABEL-DP 比 ISABEL-FP 更具可扩展性。

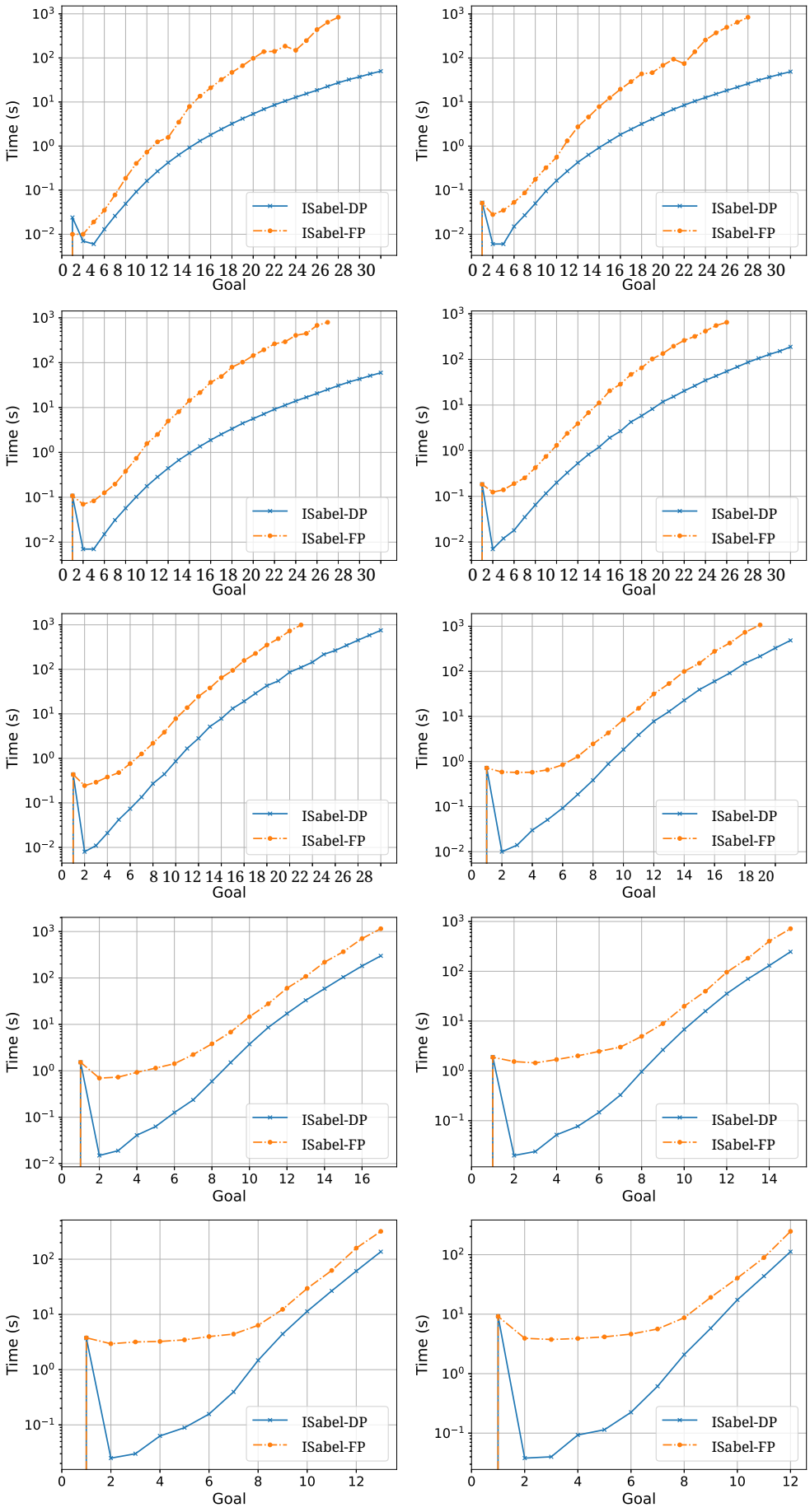

图2：从上到下、从左到右，ISABEL-DP 和 ISABEL-FP 在实例 TIREWORLD-1 到 TIREWORLD-10 中添加新目标所需的时间（以对数刻度表示）。

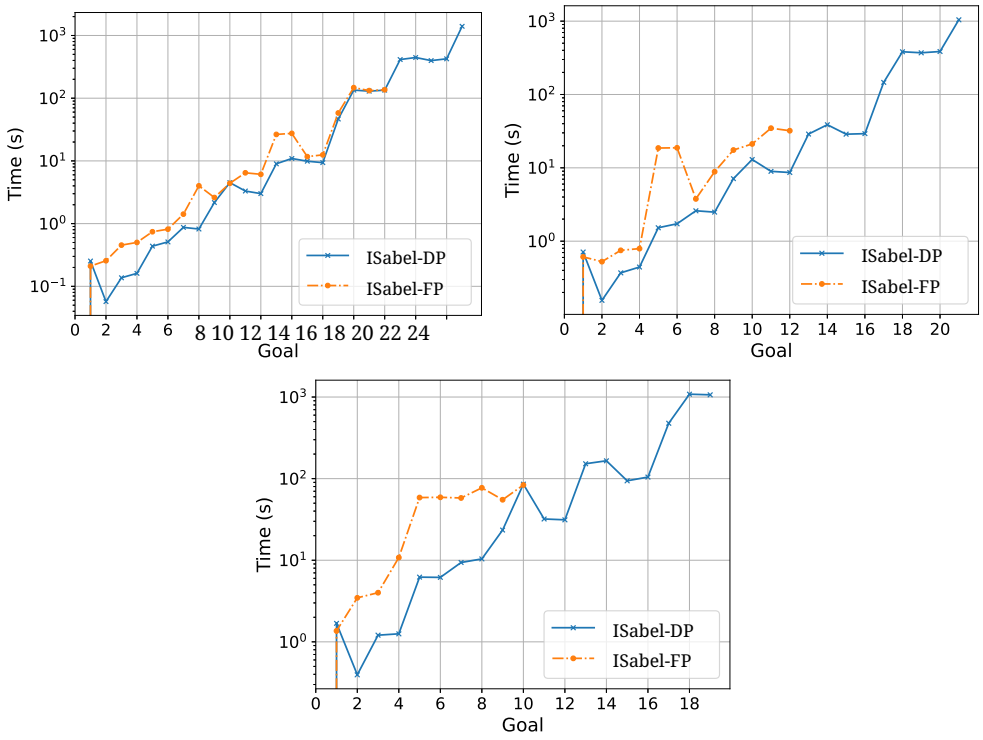

图3：从上到下、从左到右，ISABEL-DP 和 ISABEL-FP 在实例 COUNTER-6 到 COUNTER-8 中添加新目标所需的时间（以对数刻度表示）。

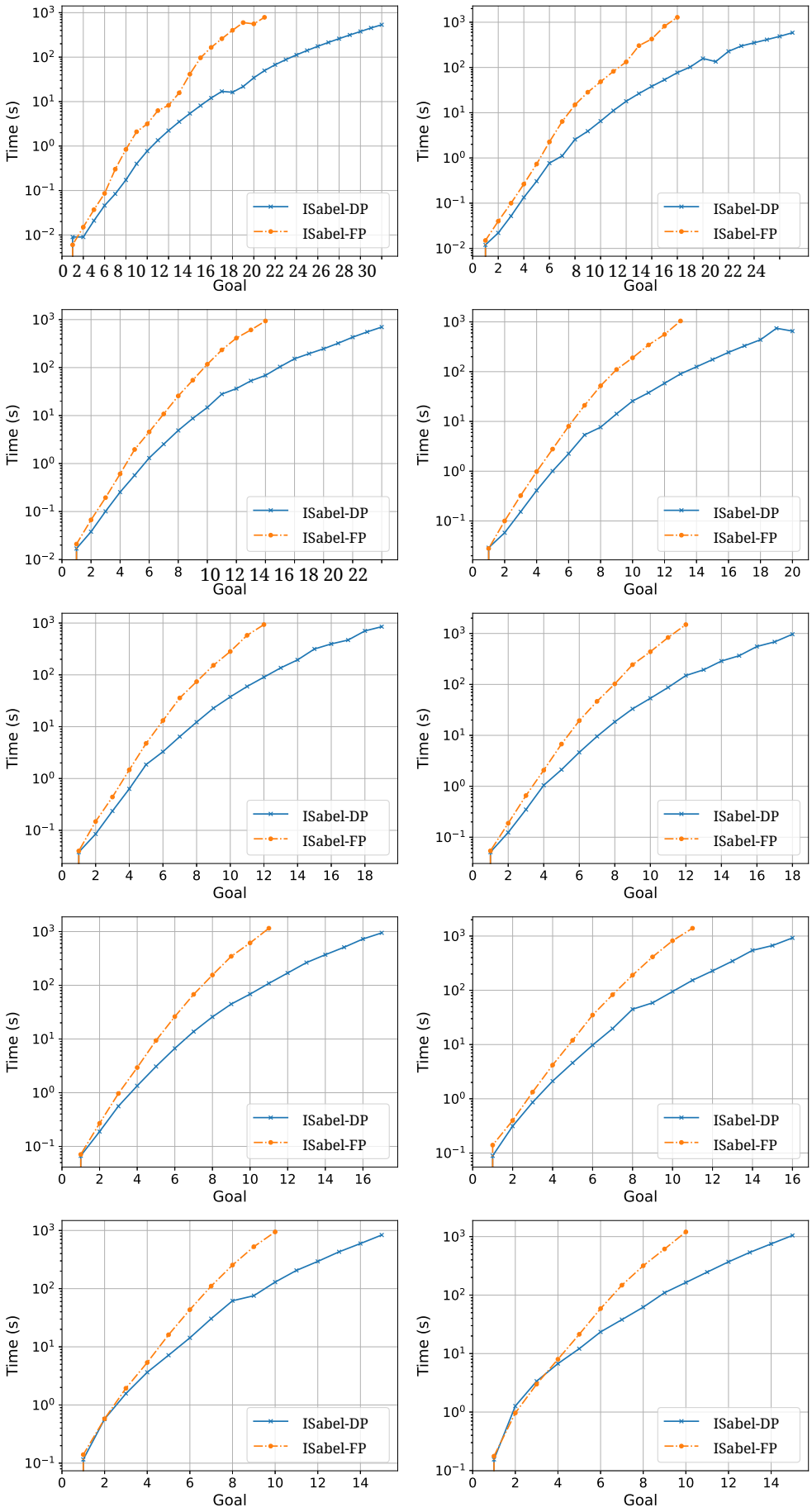

图 4：从上到下、从左到右，ISABEL-DP 和 ISABEL-FP 在实例 PLANTS-1 至 PLANTS-10 中添加新目标所需的时间（以对数刻度表示）。

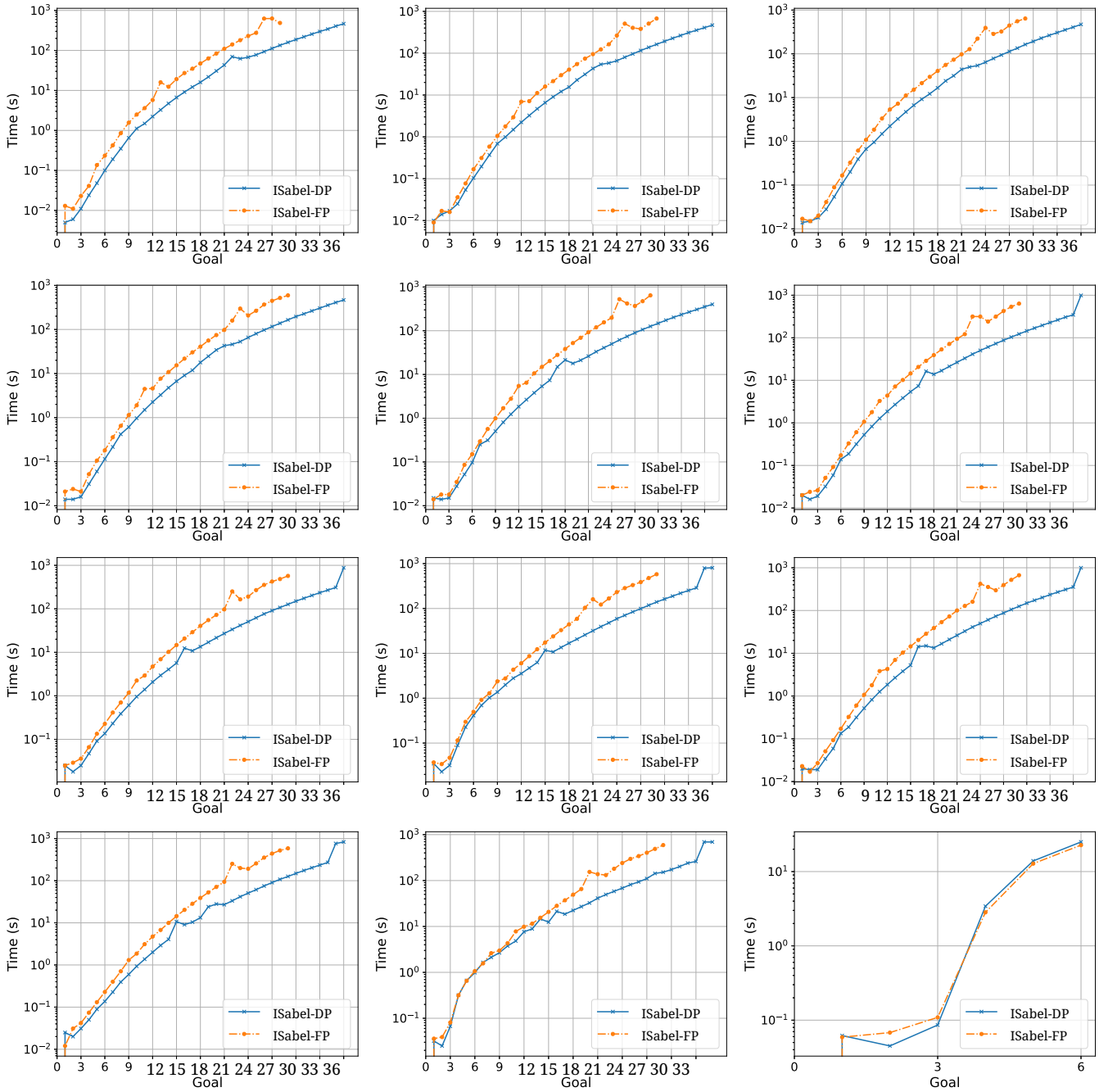

图5：从上到下、从左到右，ISABEL-DP 和 ISABEL-FP 在实例 REQUESTS-1-1 至 REQUESTS-3-4 中添加新目标所需的时间（对数尺度）。