变换逻辑

Alessandro Ronca 牛津大学
alessandro.ronca
@cs.ox.ac.uk

arXiv:2304.09639v3 [cs.LO] 6 Sep 2024

摘要

我们引入了一类新的时序逻辑，旨在精细地平衡表达能力与复杂度之间的权衡。它们的关键特点是能够定义一种新型算子，我们称之为变换算子。其中一些涵盖了现有的时序算子，另一些则完全新颖。特别值得关注的是基于半群的变换算子。它们使逻辑能够利用半群理论的丰富内涵；我们还表明，这类算子可以构造出表达能力和复杂度逐步提升的层级，而这些层级在现有逻辑中难以刻画。由此形成了一个真正新颖且尚未探索的时序逻辑领域，其中每种逻辑都有潜力满足特定应用所要求的表达能力与复杂度之间的权衡。

1 引言

我们介绍变换逻辑，这是一类新的时序逻辑，旨在细致地平衡表达能力与复杂度之间的权衡。其关键特征是可以定义一种我们称为变换算子的全新算子。它们捕捉序列上的模式，也可以被视为时序算子的一般化。基于有限半群的变换算子子类尤其值得关注。这类半群类算子足以捕捉所有正规语言；更引人注目的是，它们能够构造出表达能力和复杂度不断提升的层级，而这些层级在现有逻辑中难以非平凡地定义。这类层级的基础层级使用由触发器幺半群定义的算子获得。其他层级则通过引入基于单群的算子获得，而单群是所有群的构建模块。单群已被系统地分类为有限个族，参见 [Gorenstein 等人， 2018]。这一分类为群的景观提供了指南，也为时序逻辑的探索提供了路线图，正如本文的结果所明确展示的那样。

我们的动机源于时序逻辑在人工智能中的应用。它们被用于强化学习中，以指定奖励和动力学函数 [Bacchus 等人，1996；Brafman 等人，2018年；Icarte 等人，2018年；Camacho 等人，2019年；De Giacomo 等人，2020年b；De Giacomo 等人，2020年a]；在规划中用于描述时序扩展目标 [Torres 和 Baier，2015年；Camacho 等人，2017年；De Giacomo 和 Rubin，2018年；Braf- man 和 De Giacomo，2019年；Bonassi 等人， 2023]；在流推理中，用于表达能够引用数据流不同位置的程序 [Beck 等人，2018年；Ronca 等人，2022年；Walega 等人， 2023]。

在上述应用中，所需的表达能力与复杂度之间的权衡取决于具体情况。当星自由正则语言的基本表达能力足以满足需求时，可以采用过去时线性时序逻辑 [Manna 和 Pnueli, 1991] 以及 LTLf [De Giacomo and Vardi, 2013]等逻辑。在其他情况下，则需要诉诸表达能力更强的逻辑。上述逻辑现有的扩展具有全部正则语言的表达能力，参见 ETL [Wolper, 1983]和 LDLf [De Giacomo and Vardi, 2013]。这在表达能力上是一次巨大的飞跃，可能导致不必要的高计算复杂度。接下来，我们展示两个例子，其中所需的表达能力介于星自由正则语言与全部正则语言之间。这些中间片段可以在 Transformation Logics 中得到精确刻画。

例1。某个智能体被分配了一项可以多次完成的任务。我们每分钟都会收到一条更新，告知智能体在刚刚过去的这一分钟内是否完成了该任务。我们需要判断智能体是否在过去的每一天都至少完成过一次该任务。

这个例子描述了一种周期模式，超出了无星正则语言的表达能力。为了确定每一天的结束，它需要对分钟数按 $24 * 60 = 1440$ 取模计数。在变换逻辑中，可以用由循环群 C_{1440} 定义的变换算子来表达。或者，也可以分别使用由循环群 C_2、C_3 和 C_5 定义的三个变换算子。循环群算子能够刻画许多有用的周期模式。同时，它们属于可解群算子这一特殊类别；与更大的算子类别相比，这类算子具有计算复杂度更优等良好性质。

下一个例子中，仅使用可解群算子是不够的，我们需要借助对称群算子。

示例2。一场有 n 名参与者参加的自行车赛正在进行，我们需要持续跟踪实时排名。在每个步骤中，都可能发生一次超车，此时我们会收到形式为（$i, i+1$）的通知，表示原本处于第 i位的自行车手超越了原本处于第 $i+1$位的自行车手。我们已知初始排序，需要持续跟踪实时排名。

示例中的排名对应于对称群 S_n，当且仅当 $n \geq 5$ 时，该群不可解。该示例可以用一种变换逻辑来描述，其中包含一个由群 S_n定义的变换算子。

贡献摘要。我们引入变换逻辑，提供形式语法和语义。其主要特征是变换算子。正如我们通过一系列具体示例所展示的，这些算子具有很强的通用性。我们基于半群理论和代数自动机理论，建立了一种定义算子的系统方法，参见 [Ginzburg，1968年；Arbib，1969年；D¨om¨osi 和 Nehaniv， 2005]。通过这种方法，我们能够找出可以涵盖所有有限算子的素算子。对于这些算子，我们证明了一系列关于表达能力和复杂度的结果。在表达能力方面，我们证明，存在一个由触发器幺半群定义的算子，能够实现无星正则语言的表达能力；随着不断加入基于素数阶循环群的算子，表达能力逐步增强，直至涵盖所有可由可解群算子表达的语言；通过加入其他素算子即可达到所有正则语言的表达能力，这些算子可通过从有限单群分类中选取群来定义，参见 [Gorenstein 等人，2018]。在复杂度方面，我们聚焦于求值问题，并展示三组结果。首先，只要变换逻辑中的算子能够在多项式时间内求值，任何变换逻辑都可以在多项式时间内求值。其次，对于两个值得关注的算子族，即使采用紧凑表示，也可以在多项式时间内求值。第三，我们关注求值的数据复杂度，并证明，当我们分别纳入 (i) 仅触发器算子、(ii) 触发器算子和循环算子，以及 (iii) 所有算子时，其数据复杂度分别对应三个电路复杂度类 $\mathsf{AC}^0 \subsetneq \mathsf{ACC}^0 \subseteq \mathsf{NC}^1$ 。最后，我们展示了如何将过去时线性时序逻辑公式轻松转换为包含触发器算子的核心变换逻辑。

附录。本文附有一个附录，其中包含：所有结果的证明；示例的详细说明；关于如何通过变换算子扩展过去时线性时序逻辑的讨论。

第2卷 预备知识

对于 X 集合，变换是一个函数 $f\colon\ X \to X$。我们将任意域上的恒等函数记作 id。我们用 $\mathbb{B}$表示布尔域 $\{0,1\}$，用 $\mathbb{N}$表示自然数，用 $\mathbb{Z}$表示整数。

第2卷.1 形式语言

字母表 Σ 是由称为字母的元素组成的非空有限集。字符串是由来自 Σ的字母有限次连接而成的序列。形式语言是由 Σ 上的字符串构成的集合 Σ。正则语言是由正则表达式定义的语言，或者等价地，由有限自动机定义的语言 [Kleene， 1956]。无星正则语言是由不含 Kleene 星号但含补运算的正则表达式定义的语言，或者等价地，由无群有限自动机定义；参见 [Ginzburg， 1968]。

第2卷.2 命题逻辑

语法。命题变量是来自集合 $\mathcal{V}$ 的元素，我们将其视为已给定。通常，我们用小写拉丁字母表示命题变量。命题公式由命题变量和布尔算子 $\{\neg, \wedge, \vee\}$ 构成。它归纳定义为命题变量，或以下表达式之一： $\neg\alpha$、 $\alpha \wedge \beta$、$\alpha \vee \beta$，其中 α 和 β 是命题公式。可以定义其他布尔算子，但这并无必要，因为前述算子具有普遍性，足以表示所有布尔函数。

语义。命题公式的解释 I 是该公式中出现的命题变量的子集。直观地说，某个变量出现在解释中，意味着该变量表示一个为真的命题。解释也可以看作赋值。赋值是一个函数 $\nu\colon\ V \to \mathbb{B}$ 从一组命题变量 V到布尔域 $\mathbb{B} = \{0,1\}$。当 $V = \{v_1, \ldots, v_n\}$ 时，我们也可以将赋值ν写成映射 $\langle v_1、\ldots\ldots、\ v_n\rangle \mapsto \langle b_1、\ldots\ldots、\ b_n\rangle$，其含义是 $\nu(v_i) = b_i$。解释 I 对应于赋值 ν，使得 $\nu(a) = 1$ 当且仅当 $a \in I$。于是，公式的语义根据下面的可满足关系来定义。给定一个公式 α 以及针对 α 的解释 I，可满足关系 $I \models \alpha$ 按照公式的结构归纳定义；对于变量，则为

- $I \models a$ iff $a \in I$,

其他公式则归纳定义如下

- $I \models \neg\alpha$当且仅当 $I \not\models \alpha$,
- $I \models \alpha \vee \beta$ 当且仅当 $I \models \alpha$ 或 $I \models \beta$,
- $I \models \alpha \wedge \beta$ 当且仅当 $I \models \alpha$ 且 $I \models \beta$。

变量 $\{a_1, \ldots, a_m\}$ 的赋值 ν可以看作合取式 $l_1 \wedge \cdots \wedge l_m$，其中如果 ν，则 $l_i = a_i (a_i) = 1$，否则 $l_i = \neg a_i$。这使我们能够写出 $I \models \nu$。

2.3 半群与群

半群是一个非空集合，并配备一个结合二元运算，该运算将集合中的任意两个元素 a 和 b 结合起来，形成集合中的第三个元素 c，记作 $c = (a \cdot b)$。幺半群是具有单位元 e的半群，即对于每个元素 a，都有 $(a \cdot e) = (e \cdot a) = a$。单位元存在时是唯一的。群是这样一种幺半群，其中每个元素 a 都有一个逆元 b，即 $(a \cdot b) = (b \cdot a) = e$，其中 e 是单位元。对于群中的每个元素 a，其逆元都是唯一的，记作 a^{-1}。半群 S 的子半群（子群）是 S的一个子集，并且本身是一个半群（群）。半群的阶是其元素的数量。对于 S和 T半群，我们记作 $ST = \{s \cdot t \mid s \in S, t \in T\}$；我们也记作 $S^1 = S$ 和 $S^n = SS^{n-1}$。半群 S

由半群 T生成，当且仅当 $S = \bigcup_n T^n$。从半群 S 到半群 T 的同态是一个映射ψ： $S \to T$，满足对于任意s_1、 $s_2 \in S$，都有 $\psi(s_1 \cdot s_2) = \psi(s_1) \cdot \psi(s_2)$。如果 ψ 是双射，则称 S和 T 同构。同构半群被视为相同。设 G为一个群，且 H 为 G的子群。 G的右陪集是 $g \cdot H = \{g \cdot h \mid h \in H\}$（其中 $g \in G$），而 G 的左陪集是$H \cdot g = \{h \cdot g \mid h \in H\}$（其中 $g \in G$）。如果子群 H的左陪集和右陪集重合，即对于每个$g \in G$都有 $g \cdot H = H \cdot g$，则称其为正规子群。如果群是单元素集合 $\{e\}$，则称其为平凡的。单群是满足以下条件的群 G： G 的每个正规子群要么是平凡的，要么就是 G 本身。对于 g、 $h \in G$， g 和 h 的交换子为 $g^{-1} \cdot h^{-1} \cdot g \cdot h$。 G 的导出子群是由其交换子生成的子群。令 $G^{(0)} = G$，则 n阶导出子群 $G^{(n)}$ 是$G^{(n-1)}$ 的导出子群。如果存在 n 使得 $G^{(n)}$是平凡的，则称群为可解群。

触发器幺半群是一个有三个元素的幺半群 $\{s, r, e\}$，其中 $(r \cdot s) = s$、 $(s \cdot s) = s$、 $(r \cdot r) = r$，且 $(s \cdot r) = r$。所有触发器幺半群都彼此同构，因此通常统称为触发器幺半群。循环群是与整数群 C_n同构的群，该整数群 $\{0, \ldots, n-1\}$ 采用模加法 $i \cdot j = i + j \bmod n$。同样，阶为 n 的任意循环群都称为循环群 C_n。循环群的两个相关性质是：(i) 循环群 C_n 是单群，当且仅当 n 是素数；(ii) 每个循环群都是可解群。

3 变换逻辑

我们介绍变换逻辑。它是一种命题形式体系，其中原子命题是代表命题的变量，命题可以为真或假。某些变量的真值作为输入给出，而其他变量的含义则通过定义给出。定义使我们能够避免嵌套表达式，从而有助于提升此处内容的可读性。定义可以采用布尔表达式、延迟算子或变换算子。延迟算子类似于过去时线性时序逻辑中的之前算子。变换算子具有一个元素域以及一组作用于该域的变换——变换是将域中的元素映射到同一域中元素的映射。在每个步骤，变换算子都有一个相关的域元素，并根据操作数的真值对该元素施加变换。随后，求值结果由当前域元素决定。每种变换都对应一项特定功能，例如将一位设为 1，或增加计数。更多直观说明见下文“解释”段落以及后文第 3.1 节。

语法。静态定义是

$$p \coloneqq \alpha$$

其中， p 是一个命题变量， α 是一个命题公式。延迟定义是

$$p \coloneqq \mathcal{D}\, q$$

其中， p 和 q 是命题变量， $\mathcal{D}$ 称为延迟算子。变换算子 $\mathcal{T}$ 是一个元组 $\langle X, T, \phi, \psi \rangle$，由一个非空集合 X、一个由变换 τ 组成的非空集合 T： $X \to X$，以及一个函数组成 ϕ： $\mathbb{B}^m \to T$，以及一个函数 ψ： $X \to \mathbb{B}^n$。我们称 X 为变换域；分别称 m 和 n 为输入元数和输出元数。如果一个算子的变换域是有限的，则称该算子是有限的；在这种情况下，其所有其他分量也必然是有限的。变换定义是

$$p_1, \ldots, p_n \coloneqq \mathcal{T}(q_1, \ldots, q_m \mid x_0)$$

其中， $\mathcal{T}$是一个输入元数为 m 、输出元数为 n的变换算子； p_1、……、 p_n和 q_1、……、 q_m是变量； $x_0 \in X$ 是初始域元素。定义可以是静态定义、延迟定义或变换定义。在一个定义中，‘: =’左侧的表达式称为头部，右侧的表达式称为体部。程序是定义的有限集。查询是一个二元组（P、 q），由程序 P 和其中出现的变量 q 组成。程序必须是非递归的，即具有无环依赖图。程序的依赖图为程序中的每个变量设置一个节点；如果存在一个定义，其中 a位于体部而 b 是头部，则依赖图中存在一条从 a 指向 b 的有向边。程序还必须保证每个变量至多被定义一次，即每个变量 p 在定义的头部中至多出现一次；如果 p 出现在定义 d的头部中，我们称 d 定义了 p。在给定的程序P中，如果 P中存在定义定义了某个变量，则该变量是已定义变量，否则是输入变量。

对于 T 为一组变换算子，变换逻辑 $\mathcal{L}$(T) 是由所有静态定义、所有延迟定义以及算子来自 T 的所有变换定义所组成的程序集合。

语义。程序 P 的输入是由 P 的输入变量的子集组成的有限非空序列。对变换定义 d 的赋值是一个表达式 $d \mapsto x$，其中 x 是 d 的算子的一个域元素。程序的语义通过以下可满足关系来定义。给定程序 P、从 $I = I_1$ 到 P 的输入 I_ℓ 以及一个索引 $t \in [1, \ell]$，我们定义可满足关系 $(P, I, t) \models E$，其中 E 可以是命题变量、命题公式或对变换定义的赋值。我们假设定义采用上述形式；这样便可以引用其中提到的符号，例如，符号 p_i 表示变换定义的第 i 个头部变量。

1. 对于 a 一个输入变量，
   - $(P, I, t) \models a$ iff $a \in I_t$;
2. 对于 α 和 β 公式，
   - $(P, I, t) \models \neg\alpha$ 当且仅当 $(P, I, t) \not\models \alpha$;
   - $(P, I, t) \models \alpha \vee \beta$ 当且仅当 $(P, I, t) \models \alpha$ 或 $(P, I, t) \models \beta$;
   - $(P, I, t) \models \alpha \wedge \beta$ 当且仅当 $(P, I, t) \models \alpha$ 且 $(P, I, t) \models \beta$;
3. 对于 p 由静态定义定义的变量，
   - $(P, I, t) \models p$ iff $(P, I, t) \models \alpha$;
4. 对于 p 由延迟定义定义的变量，
   - $(P, I, t) \models p$ iff $(P, I, t-1) \models q$;
5. 对于 $d \mapsto x$ 对变换定义 d 的赋值，
   - $(P, I, 0) \models (d \mapsto x)$ 当且仅当 $x = x_0$;
   - $(P, I, t) \models (d \mapsto x)$ 当且仅当

– $(P, I, t-1) \models (d \mapsto x')$,
– $(P, I, t) \models (\langle q_1, \ldots, q_m \rangle \mapsto \mu)$, and
– $\tau(x') = x$ with $\tau = \phi(\mu)$;

6. 对于由变换定义 d 定义的变量 p_i ,

- $(P, I, t) \models p_i$ iff
  – $(P, I, t) \models (d \mapsto x)$, and
  – $\psi(x) = \langle b_1, \ldots, b_i, \ldots, b_n \rangle$ with $b_i = 1$.

说明。索引 t 可以视为一个时间点，遍历给定输入 I中的各个位置。第1点和第2点遵循命题公式的标准语义，并根据赋值 I_t进行求值。第3至第5点定义了定义的语义，并依赖辅助的第6点。第3点是静态定义的语义。它指出，由静态定义定义的变量 p 的真值，等于定义体部所对应的命题公式 α 在同一时间点求得的真值。第4点是延迟定义的语义。它指出，由延迟定义定义的变量 p 的真值，等于定义体部中出现的命题变量 q 在前一个时间点求得的真值。第5点描述当前与变换定义相关联的元素 x 。对于 $t = 0$，该元素是定义中指定的初始元素 x_0。对于 $t > 0$，元素 x 按如下方式确定。我们为定义体部中的变量选取一个赋值 $\langle q_1, \ldots\ldots, q_m \rangle \mapsto \mu$。具体来说，$(P, I, t) \models (\langle q_1, \ldots\ldots, q_m \rangle \mapsto \mu)$ 且 $\mu = \langle b_1, \ldots\ldots, b_m \rangle$ 意味着：若 $b_i = 1$，则 $(P, I, t) \models q_i$；否则 $(P, I, t) \not\models q_i$。该赋值确定变换 $\tau = \phi(\mu)$，进而根据前一个元素 x'确定下一个元素 $x = \tau(x')$。第6点定义了变换定义的语义。它规定了由变换定义 d 定义的变量 $p_1, \ldots\ldots, p_n$ 在何种情况下为真。语义通过每次考察一个变量 p_i 来定义，该变量视为前述列表的一部分。变换域中有一个元素 x ，它当前与定义 d相关联，且该定义为 $\mathcal{T} = \langle X, T, \phi, \psi \rangle$。也就是说，条件 $(P, I, t) \models (d \mapsto x)$ 成立。因此，变量$p_1, \ldots\ldots, p_n$的赋值 $\langle b_1, \ldots\ldots, b_n \rangle$ 由 $\psi(x)$ 给出。

3.1 算子示例

定义变换算子的机制允许构造出种类繁多的算子。本节将介绍几个变换算子的示例，展示如何轻松地从文献中已有的算子中提取算子，或设计新的算子来捕捉序列中的已知模式。

时序算子。我们可以按照过去时线性时序逻辑中的时序算子的形式来定义算子，参见 [Manna 和 Pnueli, 1991]。例如，我们可以定义算子

$$\diamondsuit = \langle \mathbb{B}, T, \phi, id \rangle,$$

其中 T由变换集合$(x) = 1$和 id 组成，并且函数 ϕ 定义为 $\phi(0) =$ id 和 $\phi(1) =$ set。然后我们可以写出一个定义

$$p \coloneqq \diamondsuit(a \mid 0),$$

其定义 p 为：当 a 已经发生时为真。在这里，初始变换元素唯一有意义的选择是0，因此可以省略。因此，我们可以将相同的定义写成

$$p \coloneqq \diamondsuit a,$$

并理解为它与上面的算子相对应。其他时序算子也可以用类似的方式引入。

阈值计数算子。阈值为 n 的阈值计数算子为

$$\mathcal{T}_n = \langle \mathbb{N}, T, \phi, \psi \rangle,$$

其中， T 由变换 inc 和 id 组成，且 inc$(x) = x + 1$；函数 ϕ 定义为 $\phi(1) =$ inc 和 $\phi(0) =$ id；函数 ψ 定义为：当 $x \geq n$ 时为 $\psi(x) = 1$，否则为 $\psi(x) = 0$。该算子可以检查给定条件是否至少发生了 n 次。值得注意的是，将所有自然数 $\mathbb{N}$ 替换为有限集 $[0, n]$ ，并将递增变换修改为 inc$(x) =$ min$(n, x + 1)$，我们可以等价地将该算子定义为有限算子。二者等价，因为 ψ 无法区分大于 n 的整数。

示例 3。一个智能体必须收集至少 30 单位的石料，并收集至少 115 单位的铁（已有 13 单位）。收集到足够数量后，它便可以交付并获得奖励。奖励函数由查询 (P, reward) 描述，其中 P 由以下定义组成：

$$\begin{aligned}
enoughStone &\coloneqq \mathcal{T}_{30}(stone \mid 0),\\
\text{enoughIron} &\coloneqq \mathcal{T}_{115}(\text{铁} \mid 13),\\
\text{successfulDelivery} &\coloneqq \text{足够石料} \wedge \text{足够铁} \\
&\qquad \wedge delivery,\\
\text{已交付}: = &\quad \text{交付成功},\\
\text{notAlreadyDelivered} &\coloneqq \neg \text{alreadyDelivered},\\
\text{reward} &\coloneqq \text{交付} \wedge \text{notAlreadyDelivered}。
\end{aligned}$$

当智能体成功完成一次交付时，命题 reward 在第一个时间点成立。

奇偶校验算子。奇偶校验算子是

$$\mathcal{P} = \langle \mathbb{B}, T, \phi, id \rangle,$$

其中， T 由恒等函数 id 和布尔否定函数 ¬组成；函数 ϕ 定义为 $\phi(0) =$id 和 $\phi(1) = \neg$。该算子用于检查输入变量为真的次数是否为偶数。程序示例如下：

$$\begin{aligned}
even &\coloneqq \mathcal{P}(a \mid 0),\\
odd &\coloneqq \neg even.
\end{aligned}$$

它用于定义 a 发生的次数是偶数还是奇数。

度量时序算子。按照度量时序逻辑 [Koymans, 1990], 的风格，我们可以定义这样的算子：检查某件事是否发生在最近 k步内，其中时间与自然数同构，而不是像原始度量时序逻辑那样与

实数同构。上述算子定义为

$$\diamondsuit_k = \langle \mathbb{Z}, T, \phi, \psi \rangle,$$

其中，集合 T 的变换集合和 dec 定义为 set(x) $= k$ 以及 dec(x) $= x - 1$；函数 ϕ 定义为ϕ(1) $=$ set 和 ϕ(0) $=$ dec；函数 ψ 定义为ψ(x) $= 1$ if $x > 0$ 以及 ψ(x) $= 0$ otherwise。通过将$\mathbb{Z}$ 替换为 $[0, k]$，并将 dec(x) $=$ max(0, $x - 1$)，该算子也可以等价地定义为有限算子。具有无限变换域的算子。虽然上述所有算子都可以定义为有限算子，但也可以纳入要求无限变换域的算子。例如，算子

$$\text{sameNum} = \langle \mathbb{Z}, T, \phi, \psi \rangle,$$

其中，T 由 inc(x) $= x + 1$、id (x) $= x$和 dec(x) $= x - 1$组成；函数 ϕ 定义为 ϕ(0, 0) $=$$\phi$(1, 1) $=$ id、ϕ(1, 0) $=$ inc、ϕ(0, 1) $=$ dec；函数 ψ 定义为 ψ(x) $= 1$ iff $x = 0$。当该算子用于如下定义时

$$p \coloneqq \text{sameNum}(a, b \mid 0),$$

我们有，变量 p 为真，当且仅当 a 和 b 为真的次数相同。该算子可用于识别平衡括号的戴克语言，而该语言不是正则语言。若存在一个等价的有限算子，就意味着存在一个有限自动机，从而该语言也将是正则语言。

3.2 有限半群型算子

我们提出一种基于有限半群来定义算子的系统方法。

定义1. 设有限半群为 $S = (X, \cdot)$，满射函数为 ϕ：$\mathbb{B}^m \to X$，单射函数为 ψ： $X \to \mathbb{B}^n$。它们定义变换算子 $\langle X, T, \phi, \psi \rangle$，其中 T 由每个变换 y(x) $= x \cdot y$（$y \in X$）组成。我们称之为半群型算子。

直观地说，ϕ 是集合 X 的二进制解码，而 ψ 是二进制编码。请注意，y 既被视为 X 的一个元素，也被视为函数 y: $X \to X$。从现在起，我们假设对每个集合 X，都固定一种这样的编码和解码。任何选择都适用于我们的目的。因此，我们只说半群定义了一个半群型算子，而不明确提及函数 ϕ 和 ψ。

定义 2（素算子）。触发器算子是由触发器幺半群定义的变换算子。若一个算子由（单）有限群定义，则它是一个（单）群算子。素算子包括触发器算子和单群算子。

我们将特别关注触发器算子以及由循环群定义的算子。对于这两个算子，我们给出一个明确的定义，这要求我们确定编码/解码函数的选择。与直接应用定义1所得的定义相比，我们给出的定义有所改进。具体而言，我们从触发器算子的变换域中省略了一个元素，因为该元素是多余的。在附录中，我们讨论如何通过直接应用定义1来推导这些算子。

定义 3。触发器算子是

$$\mathcal{F} = \langle \mathbb{B}, T, \phi, id \rangle,$$

其中 T由变换集合 set、reset、read 组成，定义如下

$$set(x) = 1, \quad reset(x) = 0, \quad read(x) = x,$$

且函数 ϕ 定义为

$$\phi(0,0) = read,\ \phi(1,0) = \phi(1,1) = set,\ \phi(0,1) = reset.$$

触发器算子对应于数字电路中的触发器——它对应于一个 SR锁存器，但不允许输入 11，参见 Roth 等人，2004]。该算子使我们能够用如下定义来描述一个触发器

$$storedBit \coloneqq \mathcal{F}(writeOne, writeZero).$$

因此，逻辑 $\mathcal{L}(\mathcal{F})$ 可用作包含逻辑门和触发器的数字电路的规范语言。接下来我们介绍循环算子。

定义 4. 阶为 n 的循环算子为

$$\mathcal{C}_n = \langle [0, n-1], T, \phi, \psi \rangle,$$

其中，变换 $T = \{\text{inc}_i \mid i \in [0, n-1]\}$定义为

$$inc_i(x) = (x + i) \bmod n,$$

函数 ϕ 定义为

$$\phi(b_1, \ldots, b_m) = inc_i,$$

其中，m 是表示n所需的最少位数，i是 $n-1$与二进制表示为 $b_1 \ldots b_m$的数之间的较小值；最后，函数 ψ(x)返回 x的二进制表示。

当循环算子用于如下定义时

$$p_1, \ldots, p_m \coloneqq \mathcal{C}_n(0, \ldots, 0, a \mid 0),$$

变量 p_1、...、 p_m 给出 a 为真的次数对 n取模后的数值的二进制表示。请注意，前文引入的奇偶算子与 $\mathcal{C}_2$一致。还请注意，我们选择了高效的二进制编码/解码方案。另一种做法是选择函数 ψ，使其返回计数器值的独热编码。这样需要 $m = n$，而不是 n的对数。接下来我们将刻画素循环算子。

命题1. 循环算子 $\mathcal{C}_n$ 是素算子，当且仅当 n是素数。

其他素算子由有限单群定义，这些有限单群见于有限单群分类，参见 [Gorenstein 等人， 2018]。除素数阶循环群的无限族外，该分类还包括两个无限族的交错群和李型群，以及 27 个散在群。

4 表达能力结果

我们展示了具有有限算子的变换逻辑的表达能力结果，重点关注类半群算子。我们首先从形式语言的角度定义变换逻辑的表达能力这一概念。

定义 5. 考虑一个程序 P，其输入变量为 $V = \{p_1, \ldots, p_n\}$。字母表 $\mathbb{B}^n$ 中的每个字母 $\langle b_1, \ldots\ldots, b_n \rangle$ 都定义了对 V 中变量的一项赋值 ν，即 $\nu(p_i) = b_i$。于是，$\mathbb{B}^n$ 上的每个词 w 都定义了程序 P 的一个输入 I_w。查询 (P, q) 接受 $\mathbb{B}^n$ 上的一个词 w，当且仅当 $(P, I_w, |w|) \models q$；若查询恰好接受 L 中的词，则它识别 $\mathbb{B}^n$ 上的一个语言 L。

定义 第2卷. 逻辑 $\mathcal{L}$ 的表达能力是其查询所识别的语言集合。若逻辑 $\mathcal{L}_1$ 的表达能力真包含于逻辑 $\mathcal{L}_2$ 的表达能力中，则称逻辑 $\mathcal{L}_1$ 的表达能力弱于逻辑 $\mathcal{L}_2$，记作 $\mathcal{L}_1 \sqsubset \mathcal{L}_2$；其中 $\mathcal{L}_1$ 和 $\mathcal{L}_2$ 分别对应这两个逻辑。

首先，我们确定包含所有有限算子时的表达能力。

定理 1. 对于 A 为所有有限算子的集合，$\mathcal{L}$(A) 的表达能力是正则语言。

这一结果源于有限算子能够表征有限自动机这一事实；反过来，采用有限算子的程序可以映射为有限自动机的复合。具体而言，可以映射为所谓的自动机级联复合。这一特征使我们能够运用代数自动机理论（参见 [Ginzburg, 1968；Arbib，1969；D¨om¨osi 和 Nehaniv，2005]，），从而证明素算子足以表征正则语言。

定理 2. 对于 P 为所有素算子的集合，逻辑 $\mathcal{L}$(P) 的表达能力是正则语言。

该结果通过证明每个具有有限变换算子的变换定义都可以仅由含素算子的程序捕获而获得。所使用的素算子由有限自动机的素分解定理 [Krohn 和 Rhodes, 1965]所启发。对于无星正则语言，我们得到以下专门结果。

定理 3。对于 $\mathcal{F}$ 触发器算子，逻辑 $\mathcal{L}(\mathcal{F})$ 的表达能力为无星正则语言。

超越无星类，我们还有正则语言的无名片段。现在我们开始探索这些片段。首先，要超越无星类，只需引入群算子。

定理 4。对于任意非空的群算子集合 G，逻辑 $\mathcal{L}(\mathcal{F}$,G) 的表达能力严格强于 $\mathcal{L}(\mathcal{F})$。

接下来我们聚焦于循环算子。我们将证明，它们构成了变换逻辑的一个核心，并具有良好的性质。首先，循环算子与触发器算子共同构成逻辑 $\mathcal{L}(\mathcal{F}$, S) 的一组典范且普适的算子，其中 S 是所有可解群算子的集合。它们对于 $\mathcal{L}(\mathcal{F}$, S) 也以同样的方式具有典范性和普适性，正如算子 $\{\wedge, \neg\}$ 对命题逻辑具有典范性和普适性一样。我们先陈述普适性，再陈述典范性。

定理 5（普适性）。设 C 为所有素数阶循环算子的集合，S 为任意可解算子的集合，则 $\mathcal{L}(\mathcal{F}$,C) 的表达能力包含 $\mathcal{L}(\mathcal{F}$,S) 的表达能力。

定理 6（典范性）。给定两个素数阶循环算子的集合 C_1 和 C_2，逻辑 $\mathcal{L}(\mathcal{F}, \mathsf{C}_1)$ 和 $\mathcal{L}(\mathcal{F}, \mathsf{C}_2)$ 具有相同的表达能力，当且仅当 $\mathsf{C}_1 = \mathsf{C}_2$。

规范性结果意味着存在无限的表达能力层级，例如以下层级，注意到当 p 是素数时，每个 $\mathcal{C}_p$ 都是素算子。

推论 1. $\mathcal{L}(\mathcal{C}_2) \sqsubset \mathcal{L}(\mathcal{C}_2, \mathcal{C}_3) \sqsubset \mathcal{L}(\mathcal{C}_2, \mathcal{C}_3, \mathcal{C}_5) \sqsubset \cdots$。

值得注意的是，这种形式的典范性一般并不适用于素算子。

定理7. 存在两个素算子集合 $\mathsf{P}_1 \subset \mathsf{P}_2$，使得 $\mathcal{L}(\mathsf{P}_1)$ 和 $\mathcal{L}(\mathsf{P}_2)$ 具有相同的表达能力。

结合我们的表达能力结果，可得以下层级定理。

定理 8. 对于包含所有可解群算子的集合 S 和包含所有素算子的集合 P，存在如下无限表达能力层级：

$$\mathcal{L}(\mathcal{F}) \sqsubset \mathcal{L}(\mathcal{F}, \mathcal{C}_2) \sqsubset \mathcal{L}(\mathcal{F}, \mathcal{C}_2, \mathcal{C}_3) \sqsubset \cdots \sqsubset \mathcal{L}(\mathcal{F}, \mathbf{S}) \sqsubset \mathcal{L}(\mathbf{P}).$$

该定理给出了我们目前对变换逻辑表达能力的认识。在层级底端，$\mathcal{L}(\mathcal{F})$的表达能力与无星正则语言相同。在层级顶端，$\mathcal{L}$(P)的表达能力涵盖所有正则语言。介于两者之间的还有无穷多个逻辑，它们刻画了正则语言中各不相同的片段。

5 复杂度结果

我们研究变换逻辑求值问题的复杂度。

定义 7。变换逻辑 $\mathcal{L}$ 的求值问题是指：给定一个带有 $P \in \mathcal{L}$的查询 (P, q)，以及一个针对 P的输入 $I = I_1, \ldots, I_\ell$，判断 $(P, I, \ell) \models q$ 是否成立。

为了研究复杂度，我们需要假设算子的表示方式，这决定了算子的大小。一般而言，选择算子的表示方式超出了本文的讨论范围。下面我们讨论两个具体情形。

定义 第8卷. 一族算子 T 是多项式时间的，如果对于每个算子 $\langle X, T, \phi, \psi \rangle \in$ T，其 ϕ：$\mathbb{B}^m \to T$和ψ：$X \to \mathbb{B}^n$，对于每个 $x \in X$ 和每个$\mu \in \mathbb{B}^m$，值 $\psi(\tau(x))$ 且满足 $\tau = \phi(\mu)$ 可在算子大小的多项式时间内计算。

定理 9. 对于任意（可能是无限的）多项式时间算子集合 T，$\mathcal{L}$(T) 的求值问题属于 PTime。此外，存在一个多项式时间算子集合 H，使得 $\mathcal{L}$(H) 的求值问题是 PTime-完全的。

我们论证了该上界适用于每个有限算子有限集，包括素算子。相反，无限的有限算子集可能不是多项式时间。

引理 1. 每个由有限算子组成的有限集都是多项式时间的。存在一个不是多项式时间的有限算子集。

定理 10. 对于任意有限个算子组成的有限集 T，$\mathcal{L}$(T) 的求值问题属于 PTime。

接下来我们重点讨论阈值计数反算子 $\mathcal{T}_n$ 和循环算子 $\mathcal{C}_n$。我们将完整族分别记为 $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$。值得注意的是，这类算子可以用紧凑的形式表示。

定义 9. 如果阈值计数算子族 $\{\mathcal{T}_n\}$ 的表示大小为 $O(\log n)$，则称该表示为紧凑表示。循环算子族 $\{\mathcal{C}_n\}$ 也类似。

通过使用固定数量的位对符号 $\mathcal{T}$ 和 $\mathcal{C}$ 进行编码，并使用对数数量的二进制位对索引 n 进行编码，可以获得紧凑表示。

命题 2. 阈值计数器算子 $\{\mathcal{T}_n\}$ 和循环算子 $\{\mathcal{C}_n\}$ 具有紧凑表示。

引理 2。算子族 $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$ 是多项式时间的，即使以紧凑形式表示也是如此。

总体而言，我们得出结论：对于包含任意有限个有限算子，以及阈值计数器和循环算子的每一种变换逻辑，多项式时间求值都是可能的。

定理11. 对任意有限算子组成的有限集 A，$\mathcal{L}(\mathrm{A}\ \{\mathcal{T}_n\},\{\mathcal{C}_n\})$ 的求值问题属于 PTIME，即使算子 $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$ 采用紧凑表示也是如此。

5.1 常深度与数据复杂度

我们研究具有常深度的程序的求值复杂度。这会带来数据复杂度方面的结果。具体而言，对于常深度程序，所有成员复杂度结果也都意味着数据复杂度方面的成员资格，前提是程序固定而程序的输入大小可以任意变化。数据复杂度衡量固定程序的求值如何随输入大小变化 [Vardi, 1982]。我们首先定义程序的深度，以及相应的常深度程序类别。

定义10。考虑一个程序 P。P 的输入变量的深度为零。由静态定义 $d \in P$ 定义的变量，其深度等于 d 体部中变量的最大深度，加上 d体部解析树的深度。由延迟或变换定义 $d \in P$ 定义的变量，其深度等于 d 体部中变量的最大深度加一。P 的深度等于 P中变量的最大深度。

定义11. 对任意变换算子集 T，以及任意深度 k，变换逻辑 $\mathcal{L}(\mathrm{T} \mid k)$ 是 $\mathcal{L}(\mathrm{T})$ 的子集，其中包含深度至多为 k 的程序。

我们的结果以三个电路复杂度类来表述，下面列出这些类及其已知的包含关系。

$$\mathrm{AC}^0 \subsetneq \mathrm{ACC}^0 \subseteq \mathrm{NC}^1$$

第一个结果针对触发器算子，它建立在关于无群半群复杂度的结果之上 [Chandra 等人, 1985]。

定理 12. 对于任意 k，$\mathcal{L}(\mathcal{F} \mid k)$ 的求值属于 AC^0。

循环群算子，以及一般的可解群算子，都会增加求值问题的复杂度。

定理 13。对于任意深度 k 以及任意有限集 S（由可解群算子组成），$\mathcal{L}(\mathcal{F}, \mathrm{S} \mid k)$ 的求值属于 ACC^0。此外，存在一个可解群算子 $\mathcal{G}$，使得对于每个深度 $k \geq 1$，$\mathcal{L}(\mathcal{G} \mid k)$ 的求值不属于 AC^0。

上界利用了可解群 [的复杂性结果 1989]Barrington，下界依赖于这样一个事实：循环群（因而是可解群）C_2 能够表示奇偶函数，而已知奇偶函数不属于 AC^0 [Furst 等人, 1984]。

非可解群与更大的电路复杂度类 NC^1 之间存在精确对应关系。我们的结果建立在关于非可解群复杂度的结果之上 [Bar- rington, 1989]。

定理 14. 对于任意深度 k，以及包含非可解群的任意群的有限集 G，$\mathcal{L}(\mathcal{F}, \mathrm{G} \mid k)$ 的求值问题在 AC^0 归约下是 NC^1 完全的。

6 与过去时线性时序逻辑的关系

我们证明，变换逻辑 $\mathcal{L}(\mathcal{F})$ 可以表达过去时线性时序逻辑。附录中包含过去时线性时序逻辑的背景介绍。首先，before 和 since 算子分别对应 delay 和触发器算子。

引理 3。考虑一个过去时线性时序逻辑公式$\varphi = \ominus p$。令 P为由定义 q 组成的单例程序：$= \mathcal{D}\, p$对于 I的每个解释 φ 以及每个时间点 t，有 $(I,\ t) \models \varphi$ 当且仅当 $(P,\ I,\ t) \models q$。

引理 4. 考虑一个过去时线性时序逻辑公式 $\varphi = a\, \mathcal{S}\, b$。令 P 为由两个定义 $c := \neg a$ 和 $p := \mathcal{F}(b, c \mid 0)$ 组成的程序。对于 φ 的每个解释 I 以及每个时间点 t，当且仅当 $(P, I, t) \models p$ 时，$(I, t) \models \varphi$。

根据上述引理，我们可以通过对其解析树进行归纳，为任意给定的过去时线性时序逻辑公式构造一个程序：对每个布尔算子的出现引入一个静态定义，对 before 算子的每次出现引入一个延迟定义，并对 since 算子的每次出现引入一个变换定义。

定理 15. 每个过去时线性时序逻辑公式 φ 都可以转换为一个 $\mathcal{L}(\mathcal{F})$ 的程序。该程序的大小与 φ 的大小呈线性关系，并且可以在对数空间内从 φ 计算得到。

7 相关工作

变换逻辑是人工智能中所采用的丰富时序逻辑和动态逻辑体系的重要补充。这类逻辑包括过去的命题时态逻辑，例如过去时线性时序逻辑，参见 [Manna 和 Pnueli, 1991]；在有限轨迹和无限轨迹上均有解释的未来时序逻辑 [Pnueli，1977年；Wolper，1983年；De Giacomo and Vardi, 2013]；动态逻辑，例如 PDL，参见 [Harel 等人，2000], 以及有限轨迹上的线性动态逻辑 [De Giacomo and Vardi, 2013]；具有时间的稠密解释的逻辑，例如度量时序逻辑 [Koymans, 1990]。它们还包括一阶变体，例如过去 FOTL [Chomicki, 1995]。最后，它们包括若干种基于规则的逻辑：命题模态时态逻辑，例如 Templog [Abadi 和 Manna，1989 年；Baudinet, 1995]；具有

一阶变量，其中时间显式地作为特殊类型的参数出现，例如 Datalog_{1S} [Chomicki 和 Imielinski， 1988], 以及时序 Datalog [Ronca 等人，2018年；Ronca 等人， 2022]；具有度量时序算子的逻辑，例如 DatalogMTL [Brandt 等人，2018年；Walega 等人，2019年；Walega 等人， 2020]；以及专门用于流推理的命题模态逻辑 [Beck 等人， 2018],及其加权变体 [Eiter 和 Kiesel， 2020]这些逻辑与我们的工作具有共同的代数特征。

第8卷 结论

变换逻辑为序列上的逻辑提供了一个通用框架。在群论的指导下，变换算子所带来的灵活性使我们能够定义具有正则语言多个不同片段表达能力的逻辑。无星正则语言因具有许多成熟形式体系的表达能力而广为人知，而超出这些语言的片段则鲜为人知。变换逻辑为探索这些片段提供了一种途径。

致谢

Alessandro Ronca 得到了欧洲研究委员会（ERC）通过欧盟地平线2020研究与创新程序的资助（资助协议编号：852769，ARiAT）。

参考文献

[Abadi 和 Manna， 1989] Mart´ın Abadi 和 Zohar Manna。时序逻辑编程。《符号计算杂志》，第8卷，1989 年。[Arbib， 1969] Michael Arbib。《抽象自动机理论》。自动计算。Prentice-Hall，1969年。[Bacchus 等， 1996] Fahiem Bacchus、Craig Boutilier 和 Adam J. Grove。奖励行为。载于 AAAI，1996年。[Barrington， 1989] David A. Mix Barrington。有界宽度多项式规模的分支程序恰好识别 NC1 中的语言。《计算机与系统科学杂志》，第38卷，1989 年。[Baudinet， 1995] Marianne Baudinet。论时序逻辑编程的表达能力。《信息与计算》，第117卷，1995年。[Beck 等， 2018] Harald Beck、Minh Dao-Tran 和 Thomas Eiter。LARS：一种基于逻辑的流分析推理框架。《人工智能》，第261卷，2018年。[Bonassi 等， 2023] Luigi Bonassi、Giuseppe De Giacomo、Marco Favorito、Francesco Fuggitti、Alfonso Emilio Gerevini 和 Enrico Scala。纯过去线性时序逻辑中时序扩展目标的规划。载于 ICAPS，第33卷，2023年。[Brafman 和 De Giacomo， 2019] Ronen I. Brafman 和 Giuseppe De Giacomo。非马尔可夫完全可观测非确定性领域中针对 LTLf/LDLf 目标的规划。载于 IJCAI，2019年。[Brafman 等， 2018] Ronen I. Brafman、Giuseppe De Giacomo 和 Fabio Patrizi。LTLf/LDLf 非马尔可夫奖励。载于 AAAI，2018年。[Brandt 等人， 2018] Sebastian Brandt、Elem G¨uzel Kalayci、Vladislav Ryzhikov、Guohui Xiao 和 Michael Za- kharyaschev。《使用度量时序逻辑查询日志数据》。人工智能研究杂志，第62卷，2018年。[Camacho 等人， 2017] Alberto Camacho、Eleni Triantafil- lou、Christian J. Muise、Jorge A. Baier 和 Sheila A. McIlraith。《具有时序扩展目标的非确定性规划：基于有限与无限轨迹的 LTL》。载于 AAAI，2017年。[Camacho 等人， 2019] Alberto Camacho、Rodrigo Toro Icarte、Toryn Q. Klassen、Richard Anthony Valenzano 和 Sheila A. McIlraith。《LTL 及其扩展：强化学习中奖励函数规范的形式语言》。载于 IJCAI，2019年。[Chandra 等人， 1985] Ashok K. Chandra、Steve n Fortune 和 Richard J. Lipton。《无界扇入电路与结合函数》。计算机与系统科学杂志，第30卷，1985年。[Chomicki 和 Imielinski， 1988] Jan Chomicki 和 Tomasz Imielinski。《时序演绎数据库与无限对象》。载于 PODS，1988年。[Chomicki， 1995] Jan Chomicki。《使用有界历史编码高效检查时序完整性约束》。ACM 数据库系统汇刊，第20卷，1995年。[Dantsin 等人， 2001] Evgeny Dantsin、Thomas Eiter、Georg Gottlob 和 Andrei Voronkov。《逻辑程序设计的复杂度与表达能力》。ACM 计算调查，第33卷，2001年。[De Giacomo 和 Rubin， 2018] Giuseppe De Giacomo 和 Sasha Rubin。《面向 LTLf 和 LDLf 目标的 FOND 规划的自动机理论基础》。载于 IJCAI，2018 年。[De Giacomo 和 Vardi， 2013] Giuseppe De Giacomo 和 Moshe Y. Vardi。《有限轨迹上的线性时序逻辑与线性动态逻辑》。载于 IJCAI，2013年。[De Giacomo 等人，2020年a] Giuseppe De Giacomo、Marco Favorito、Luca Iocchi、Fabio Patrizi 和 Alessandro Ronca。《通过转换器监测时序逻辑奖励》。载于 KR，2020年。[De Giacomo 等人，2020年b] Giuseppe De Giacomo、Luca Iocchi、Marco Favorito 和 Fabio Patrizi。《强化学习智能体的约束机制》。载于 AAAI，2020年。[D¨om¨osi 和 Nehaniv， 2005] P´al D¨om¨osi 和 Chrystopher L Nehaniv。《自动机网络代数理论：导论》。SIAM，2005年。[Eiter 和 Kiesel， 2020] Thomas Eiter 和 Rafael Kiesel。《用于定量流推理的加权 LARS》。载于 ECAI，2020年。[Furst 等人， 1984] Merrick L. Furst、James B. Saxe 和 Michael Sipser。《奇偶性、电路与多项式时间层次》。数学系统理论，第17卷，1984 年。[Gallian， 2021] Joseph Gallian。《当代抽象代数》。Chapman and Hall/CRC，2021年。[Ginzburg， 1968] Abraham Ginzburg。《自动机代数理论》。Acade mic Press，1968年。

[Gorenstein 等人， 2018] Daniel Gorenstein、Richard Lyons 和 Ronald Solomon。《有限单群分类》第8卷。美国数学会，2018年。

[Groupprops， 2023] Groupprops。《交错群a_5》。https://groupprops.subwiki.org/wiki/Alternating group:A5# _ Subgroups，2023年。访问日期：2023年8月8日。

[Harel 等人， 2000] David Harel、Jerzy Tiuryn 和 Dexter Kozen。《动态逻辑》。麻省理工学院出版社，2000年。

[Hopcroft 和 Ullman， 1979] J. Hopcroft 和 J. Ullman。《自动机理论、形式语言与计算导论》。Addison-Wesley，1979年。

[Icarte 等人， 2018] Rodrigo Toro Icarte、Toryn Q. Klassen、Richard Anthony Valenzano 和 Sheila A. McIlraith。使用线性时序逻辑向强化学习智能体教授多项任务。载于 AAAI-MAS，2018年。[Kleene， 1956] Stephen C. Kleene。神经网络与有限自动机中的事件表示。《自动机研究》，第34卷，1956年。[Koymans， 1990] Ron Koymans。使用度量时序逻辑描述实时性质。《实时系统》，第2卷，1990。[Krohn 和 Rhodes， 1965] Kenneth Krohn 和 John Rhodes。机器代数理论 I：有限半群与机器的素分解定理。《美国数学会汇刊》，第116卷，1965年。[Manna 和 Pnueli， 1991] Zohar Manna 和 Amir Pnueli。完成时序图景。《理论计算机科学》，第83卷，1991年。[Pnueli， 1977] Amir Pnueli。程序时序逻辑。载于 FOCS，1977年。[Ronca 等人， 2018] Alessandro Ronca、Mark Kaminski、Bernardo Cuenca Grau、Boris Motik 和 Ian Horrocks。时序 Datalog 中的流推理。载于 AAAI，2018年。[Ronca 等人， 2022] Alessandro Ronca、Mark Kaminski、Bernardo Cuenca Grau 和 Ian Horrocks。基于规则的流推理中的延迟问题与窗口大小问题。《人工智能》，第306卷，2022年。[Roth 等人， 2004] Charles Roth、Larry Kinney 和 Eugene John。逻辑设计基础。Thomson Brooks/Cole，2004年。[Sch¨utzenberger， 1965] Marcel Paul Sch¨utzenberger。关于仅具有平凡子群的有限幺半群。《信息与控制》，第8卷，1965年。[Torres 和 Baier， 2015] Jorge Torres 和 Jorge A. Baier。将 LTL 时序扩展目标多项式时间重构为终态目标。载于 IJCAI，2015年。[Vardi， 1982] Moshe Y. Vardi。关系查询语言的复杂度（扩展摘要）。载于 STOC，1982年。[Walega 等人， 2019] Przemyslaw Andrzej Walega、Bernardo Cuenca Grau、Mark Kaminski 和 Egor V. Kostylev。DatalogMTL：计算复杂度与表达能力。载于 IJCAI，2019年。

[Walega 等人， 2020] Przemyslaw Andrzej Walega、Bernardo Cuenca Grau、Mark Kaminski 和 Egor V. Kostylev。《整数时间线上的 DatalogMTL》。载于 KR，2020年。[Walega 等人， 2023] Przemyslaw Andrzej Walega、Mark Kaminski、Dingmin Wang 和 Bernardo Cuenca Grau。《基于 DatalogMTL 的流推理》。《网络语义学杂志》，第76卷，2023年。[Wolper， 1983] Pierre Wolper。《时序逻辑可以具有更强的表达能力》。《信息与控制》，第56卷，1983年。

《变换逻辑》附录

摘要。附录由六个部分组成。
A 部分。我们提供引言中示例 1 和示例 2 的完整版本。B 部分。我们说明如何使用变换算子扩展过去时线性时序逻辑。C 部分。我们解释触发器和循环算子的定义是如何推导出来的。D 部分。我们介绍证明所需的其他背景知识。E 部分。我们证明一些用于主要结果证明的初步结果。F 部分。我们按照主要正文中陈述结果的顺序，给出所有结果的证明。

示例的完整版本

A.1 示例 1

我们在引言部分报告的示例 1 基础上提供了完整的详细说明。这是一个基于智能体的场景示例，既可以转化为强化学习应用，也可以转化为规划应用。

让我们考虑一个能够多次完成 assignedTask 任务的智能体。我们想知道该智能体是否在过去的每一天都完成了任务，并且我们每分钟都会收到一次更新。也就是说，我们接收一个输入序列 $I = I_1$、.....、I_ℓ，其中 assignedTask $\in I_t$ 当且仅当 t 是智能体刚刚完成 assignedTask 任务的时刻。该规范可以用一种变换逻辑来表示，其中包含一个标志每一天结束的算子。

运算符定义如下，其中 $n = 24 * 60 = 1440$ 表示一天中的分钟数。

$$\text{endOfDay} = \langle [0, n-1], T, \phi, \psi \rangle$$

变换集合为 $T = \{\text{inc}\}$，其中 $\text{inc}(x) = x + 1 \bmod n$；函数 ϕ：$\mathbb{B}^0 \to T$是映射到 inc 的常值映射，而函数 ψ定义为 $\psi(b_0, \ldots, b_{10}) = 1$ 当且仅当 $b_0, \ldots, b_{10}$是 $n-1$的二进制编码——请注意，编码 n 个数字只需十一位。

以下程序描述了这个示例。

$$end \coloneqq \text{endOfDay}(\mid 0) \tag{1}$$
$$doneTaskToday \coloneqq \mathcal{F}(assignedTask, end \mid 0) \tag{2}$$
$$failedInADay \coloneqq end \wedge \neg doneTaskToday \tag{3}$$
$$failed \coloneqq \mathcal{F}(failedInADay, \bot \mid 0) \tag{4}$$

首先，(1) 将 end 定义为在每天结束时为真。然后，(2) 将 doneTaskToday 定义为：如果任务自前一天结束以来已完成，则为真——请注意，该触发器会在任务完成时置位，但在每天结束时复位。接着，(3) 规定，如果当前是一天结束时，且 doneTaskToday 为假，则智能体当天未能完成任务。在这种情况下，(4) 会永久置位 failed 标志。

endOfDay 的特征半群是循环群 C_{1440}，因此根据下文证明的定理 18，它可以使用素算子$\mathcal{C}_2$、$\mathcal{C}_3$、$\mathcal{C}_5$来表示。这将使我们能够去除 end-of-day 算子，改用素算子。

该程序根据定理 13 属于数据复杂度为 ACC^0 的变换逻辑，因为该程序使用了触发器算子和可解群算子。

这个示例无法用过去时线性时序逻辑（Past LTL）表示，因为查询（P, failed ）所识别的语言不是无星语言。通过验证该语言不是非计数语言即可看出这一点，参见 [Ginzburg, 1968]。

A.2 示例 2

我们在引言部分提供了关于示例第2卷的完整详情。

一场有 n 名参与者参加的自行车赛正在进行，我们需要实时跟踪排名。每个步骤都可能发生一次超车，此时我们会收到以 $(i, i+1)$ 形式传达的信息，表示处于第 i 位的自行车手超越了处于第 $i+1$ 位的自行车手。具体而言，我们会收到一个命题 overtake($i, i+1$)。我们已知初始排序，并需要实时跟踪排名。

排名对应于对称群 S_n。事实上，对称群 S_n 是一个列表中所有 n 个对象的排列所构成的群。它可以看作整数区间 $[1, n]$上的所有变换的集合。例如，对于 $n = 3$，变换 $\{1 \mapsto 2, 2 \mapsto 3, 3 \mapsto 1\}$ 描述了这样一种排列：我们将第一个元素移到第二个位置，将第二个元素移到第三个位置，并将第三个元素移到第一个位置。对称群由相邻换位集合生成。相邻换位是一种交换两个相邻元素且不影响其他任何元素的排列。例如，对于 $n = 3$，变换 $\{1 \mapsto 2, 2 \mapsto 1, 3 \mapsto 3\}$ 是一个相邻换位。我们观察到，该示例中的排名是一个环行者列表，而“超车”就是相邻换位。因此，该示例对应于对称群 S_n。

由 S_n 定义的变换算子包含大量变换；由于排列的数量为 $n!$，因此变换数量就是这个数。然而，我们只关注相邻换位，同时加入恒等排列，以涵盖没有发生超车的情况。这样一来，我们总共得到 n 个变换。因此，我们考虑由 S_n定义的该算子的简化版本，其中只包含由相邻换位以及恒等排列给出的变换。算子定义如下。

$$\text{ranking} = \langle X, T, \phi, \psi \rangle$$

定义域 X 是整数 $[1, n]$ 的列表集合。变换集合为 $T = \{\text{id}\} \cup \{\text{转置}_{(i,i+1)} \mid i \in [1, n-1]\}$其中

$$transpose_{(i,i+1)}(j_1, \ldots, j_n) = j_1, \ldots, j_{i-1}, j_{i+1}, j_i, j_{i+2}, \ldots, j_n.$$

函数 ϕ 将全零向量 $\langle 0, \ldots, 0 \rangle$ 映射为 id，并将每个表示数字 i 的指示向量 $\langle b_1, \ldots, b_m \rangle$ 映射为变换 transpose$^{(i,i+1)}$。例如，指示向量 $\langle 0, 0, 1, 0, 0 \rangle$ 映射为 transpose$^{(3,4)}$。设 m 为编码 n 所需的最少位数。函数 $\psi\colon X \to \mathbb{B}^{nm}$ 将 X 中的每个列表映射为其二进制编码。

该示例由以下单例程序捕获。

$$p_{1,1}, \ldots, p_{1,m}, \ldots, p_{n,1}, \ldots, p_{n,m} \coloneqq \text{排名}\ \big(\text{超车}(1, 2) \ldots, \text{超车}(n-1, n)\ \mid (c_1, \ldots, c_n)\big)$$

列表 (c_1， ....., c_n) 是初始排序。头部变量根据作为输入提供的更新内容，确定实时排名。

B 扩展过去时态线性时序逻辑

我们首先介绍过去时线性时序逻辑的语法和语义，这些内容在本节以及后文的证明中都会用到。然后，我们简要讨论过去时线性时序逻辑语义形式化的一个方面；鉴于变换逻辑的语义，这一方面尤其值得关注。最后，我们讨论本节的主要内容，即通过变换算子对过去时线性时序逻辑进行可能的扩展。

B.1 背景：过去时态线性时序逻辑

我们介绍过去时态线性时序逻辑（过去时线性时序逻辑），参见 [Manna 和 Pnueli， 1991]。

语法。 一个 过去时线性时序逻辑公式 由命题逻辑公式通过 时序算子 $\{\bigcirc, \mathcal{S}\}$构成，分别称为 之前算子 和 since 算子。它被归纳定义为一个命题逻辑公式，或以下两个表达式之一： α， $\alpha \mathcal{S} \beta$ 其中 α 和 β 是过去时线性时序逻辑公式。其他时序算子可以用前述算子定义，例如， 过去某时 α：$= \top \mathcal{S} \alpha$ 以及 过去始终 α：$= \neg \neg \alpha$。

语义。过去时线性时序逻辑公式 α 的一个解释是由命题变量集合组成的有限非空序列——请注意，非空性要求并不排除由空集构成的序列。直观地说，某个变量出现在解释的第 t 个元素中，意味着该变量表示一个在时间 t 为真的命题。请注意，解释也可以看作赋值序列。公式的语义根据下面的可满足关系定义，该关系扩展了命题逻辑公式的可满足关系，并使其同时相对于时间 $t \in \{1, 2, \ldots, \}$。给定一个过去时线性时序逻辑公式 α，以及一个解释 $I = I_1$， ....., I_n for α，可满足关系 I， $t \models \alpha$ 按照公式的结构定义如下：

- I， $t \models a$ 当且仅当 $a \in I_t$,
- I， $t \models \neg\alpha$ 当且仅当 I， $t \not\models \alpha$,
- $I, t \models \alpha \vee \beta$ 当且仅当 $I, t \models \alpha$ 或 $I, t \models \beta$,
- $I, t \models \alpha \wedge \beta$ 当且仅当 $I, t \models \alpha$ 且 $I, t \models \beta$.
- I， $t \models \bigcirc\alpha$ 当且仅当 I， $t - 1 \models \alpha$,
- $I, t \models \alpha \mathcal{S} \beta$ 当且仅当存在 $j \in [1, t]$ 使得 $I, j \models \beta$ 且对每个 $k \in [j + 1, t]$， 都有 $I, k \models \alpha$。

since 算子为其语义提供了一种等价的归纳式刻画，参见 [Chomicki， 1995]。我们有，$I, t \models \alpha \mathcal{S} \beta$ 成立当且仅当以下两个条件之一成立：

1. $I, t \models \beta$, 第2卷. I,

$t - 1 \models \alpha \mathcal{S} \beta$ 和 $I, t \models \alpha$.

B.2 过去时线性时序逻辑语义讨论

值得注意的是，since 算子的语义是通过有限线性序上的一阶逻辑 FOL$[<]$ 来定义的。FOL$[<]$ 的表达能力是无星正则语言。因此，无法使用 FOL$[<]$ 来定义表达能力超出无星正则语言的算子的语义。这也说明了为何对变换算子采用另一种方法。

B.3 扩展过去时线性时序逻辑

我们讨论如何通过加入变换算子来扩展过去时线性时序逻辑。其目标是增强其表达能力，使其超越无星表达能力，并通过阈值计数算子等算子提高其简洁性。每个具有一元输出的变换算子 $\mathcal{T}$ 都可以自然地作为一个算子纳入过去时线性时序逻辑。因此，可以将公式集合扩展为包含如下形式的公式：$\mathcal{T}(\varphi_1, \ldots, \varphi_n \mid x)$，其中 φ_i 是一个公式，且 x 属于 $\mathcal{T}$ 的变换域。由此得到逻辑 PLTL(T)，即通过引入来自 T 的算子对过去时线性时序逻辑进行扩展所形成的逻辑。

例4。 考虑奇偶校验算子 $\mathcal{P}$。那么，公式 $(a \vee \mathcal{P}(\top \mid 0))$ 当且仅当 a 始终在奇数位置发生时为真。请注意，在偶数位置上析取始终成立，此时 $\mathcal{P}(\top \mid 0)$ 的求值结果为真。

输出元数大于一的变换算子似乎无法以自然的方式纳入过去时线性时序逻辑公式的语法中。只要考虑的是有限算子，这并不限制其表达能力。事实上，任何变换域为 $\{x_1, \ldots, x_n\}$ 的有限算子，都可以由 n 个一元输出算子表示，每个算子分别指示 x_i 是否为当前变换元素。尽管如此，变换逻辑基于定义的语法仍更简洁易懂。

C 触发器与循环算子：定义3和定义4的推导方式

我们解释定义是如何　　触发器算子（定义3）和循环算子（定义4）是由定义推导得出的　　1.

触发器算子。回顾一下，触发器幺半群是 $F = \{r, s, e\}$，其中 e 是单位元，并且 $(r \cdot s) = s$，$(s \cdot s) = s$，$(r \cdot r) = r$，$(s \cdot r) = r$。我们推导出由 F定义的算子，其形式为

$$\langle X, T, \phi, \psi \rangle,$$

其分量将在下文描述。首先，我们有 $X = F$。对于这些变换，注意，在 F中任取一个元素，分别与 s 和 r 相乘，总会得到 s 和 r；而与 e 相乘则总是得到该元素本身。这些映射给出了由 F定义的算子的变换。具体而言，变换集合 T由映射 $s(x) = s$、映射 $r(x) = r$和映射 $e(x) = e$组成。因此，$T = F$，这里将 F中的元素视为上述映射。接下来是一个简化步骤。我们注意到，将元素 e 纳入 X 是多余的。注意，元素 e 可以分别变换为 s、r、e，因为 $s(e) = s$、$r(e) = r$和 $e(e) = e$。与此同时，只有 e 本身可以变换为 e。如果使用初始元素不同于 e的算子——例如 $\mathcal{F}(a, b \mid s)$——那么元素 e将永远不会与该算子关联。如果使用初始元素为 e的算子——例如 $\mathcal{F}(a, b \mid e)$——那么关联元素在起初到某个时刻之前可以是 e，之后则可能变为 r 或 s。这两种行为都可以由一个从 X中省略 e 的触发器算子来表示。因此，在触发器算子的定义中，我们从 X中省略 e。然后，我们可以将s、r分别重命名为 X中的元素1、0。还可以将 s、r、e分别重命名为 T中的元素set、reset、id。最后，函数 ϕ 和 ψ 是任意的，但重要的是，它们分别是满射和单射，正如定义所要求的。

循环算子。我们回顾一下，循环群 C_n 是（同构于）整数 $\{0, \ldots, n-1\}$，其运算为模 $i \cdot j = i + j$ 加法 n。我们推导出由 C_n 定义的算子，其形式为

$$\langle X, T, \phi, \psi \rangle,$$

其分量将在下文描述。首先，我们有 $X = \{0, \ldots, n-1\}$。对于这些变换，请注意，将 C_n 中的一个元素 i 与另一个元素 j 相乘，得到 $i + j$ 模 n。这些乘法产生了由 C_n 定义的算子的变换。具体而言，变换集合 T 由映射 i（j）$= i \cdot j = i + j$ 模 n 组成，其中每个 $i \in \{0, \ldots, n-1\}$ 都满足该关系。因此，$T = \{0, \ldots, n-1\}$，其中约定每个整数都被视为上述映射之一。然后，我们可以将每个映射 i 重命名为 inc_i。最后，函数 ϕ 和 ψ 是任意的，但重要的是，按照定义的要求，它们分别是满射和单射。

D 附加背景

我们介绍以下证明所需的其他背景知识。我们介绍：（i）自动机理论的背景知识，（ii）自动机的级联组合，（iii）半群与群论的其他背景知识，以及（iv）代数自动机理论的背景知识。

D.1 自动机

我们引入有限自动机，参见 [Ginzburg，1968年；Arbib，1969年；Hopcroft 和 Ullman，1979年；D¨om¨osi 和 Nehaniv， 2005],本文将其简称为自动机。自动机是对有状态机器的一种数学描述，它以同步顺序的方式处理字符串，即对每个输入字母执行一个基本计算步骤，并将一个输出字母作为该步骤的结果返回。其核心在于一种在每个步骤更新内部状态的机制。这种机制通过半自动机的概念来刻画。

语法。半自动机是一个元组 $D = \langle \Sigma, Q, \delta \rangle$，其中： Σ 是一个称为输入字母表的字母表； Q 是一个有限非空集合，其元素称为状态； δ : $Q \times \Sigma \to Q$ 是一个称为转移函数的函数。如果 $|Q| = 1$，则半自动机是平凡的。

自动机是在半自动机的基础上添加初始状态和输出函数得到的。

自动机是一个元组 $A = \langle \Sigma, Q, \delta, q_{\text{init}}, \Gamma, \theta \rangle$，其中： $D_A = \langle \Sigma, Q, \delta \rangle$ 是一个半自动机，称为A的半自动机； $q_{\text{init}} \in Q$ 称为初始状态； Γ 是一个字母表，称为输出字母表； θ : $Q \to \Gamma$ 称为输出函数。接受器是一种特殊的自动机，其输出函数是一个布尔值函数 θ : $Q \to \mathbb{B}$。在下文中，我们假设自动机均为接受器。

值得注意的是，我们所定义的这种自动机称为 Moore 机，以区别于 Mealy 机。

语义。自动机 A 在输入字符串 $s = \sigma_1 \dots \sigma_n$ 上的执行过程如下。最初，自动机处于状态 $q_0 = q_{\text{init}}$。读取字母 σ_i后，状态更新为 $q_i = \delta(q_{i-1}, \sigma_i)$，并且自动机返回字母 $\theta(q_i)$。我们将自动机在 s上执行时返回的最后一个字母记为 $A(s)$。更形式化地，可以先将转移函数扩展到字符串上。对于空字符串 ε，转移函数扩展为 $\delta(q, \varepsilon) = q$；对于任意非空字符串 $\sigma_1 \dots \sigma_n$，扩展为 $\delta(q, \sigma_1 \dots \sigma_n) = \delta(\delta(q, \sigma_1)\ \sigma_2 \dots \sigma_n)$。于是有 $q_i = \delta(\sigma_1 \dots \sigma_i)$，并且还有$A(s) = \theta(\delta(q_{\text{init}}))$。

自动机 A 识别的语言是满足 $A(s) = 1$ 的字符串 s 的集合。若两个自动机 A_1 和 A_2 识别相同的语言，则它们等价，即它们使用相同的输入字母表 Σ，并且对每个字符串 $s \in \Sigma^*$，等式 $A_1(s) = A_2(s)$ 成立。

典范性。如果存在一个输入字符串 $\sigma_1 \dots \sigma_\ell$，使得对于 $q_0 = q'$ 和 $q_i = \delta(q_{i-1}, \sigma_i)$，有 $q = q_\ell$成立，则自动机 A 中的状态 $q \in Q$ 从状态 $q' \in Q$ 可达。如果自动机 A 中的每个状态 $q \in Q$ 都能从初始状态 q_0到达，则自动机 A 是连通的。对于自动机 A的任意状态 q，自动机 A^q 是将 q 设为初始状态后得到的自动机。若 A 中的两个状态 q 和 q' 满足自动机 A^q 和 $A^{q'}$ 等价，则这两个状态等价。如果一个自动机不存在彼此等价的不同状态，则该自动机处于约简形式。如果一个自动机是连通的且处于约简形式，则该自动机是典范的。

D.2 半自动机级联

我们介绍半自动机级联，参见 [D¨om¨osi 和 Nehaniv， 2005]。

半自动机级联是一个半自动机 $A = \langle \Sigma, Q, \delta \rangle$，由下述半自动机 d、...、 A_d及连接函数 ϕ_1、...、 ϕ_d复合而成。每个半自动机的形式为 $A_i = \langle \Sigma_i, Q_i, \delta_i \rangle$。每个连接函数的形式为 ϕ_i : $\Sigma \times Q_1 \times \cdots \times Q_{i-1} \to \Sigma_i$。该级联的状态空间为

$$Q = Q_1 \times \cdots \times Q_d,$$

其转移函数为

$$\delta(\langle q_1, \dots, q_d \rangle, \sigma) = \langle \delta_1(q_1, \sigma_1), \dots, \delta_d(q_d, \sigma_d) \rangle, \qquad \text{where } \sigma_i = \phi_i(\sigma, q_1, \dots, q_{i-1}).$$

级联 表示为 $(\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$，并称每一对 (ϕ_i, A_i) 为该级联的一个组件。

D.3 关于半群与群论的补充背景

我们补充介绍 半群与群论的背景，参见 [D¨om¨osi 和 Nehaniv，2005年；Gallian，2021年].

如果 ψ 是从半群 S 到半群 T 的同态，并且 ψ 是满射，我们就称 T 是 S 的同态像。如果 ψ 是从半群 S 到半群 T 的同态，并且 ψ 是双射，我们就称 ψ 为同构。若半群 S 有一个子半群 T'，使得 S 是 T' 的同态像，则称半群 S 整除半群 T。

如果 G 是 sim $p$ple，则 ever y 同态 p像 g是 G 的同构 p于 $\{e\}$ 或 G。

对于 G和 H半群，记作 $GH = \{g \cdot h \mid g \in G, h \in H\}$。我们也记作 $H^1 = H$和 $H^k = HH^{k-1}$。若半群S 由半群 H 生成，即 $S = \bigcup_n H^n$，则称 H 为 S的一个生成元。对于 G 一个群，以及 N 为 G的正规子群的情形，称 $G/N = \{g \cdot N \mid g \in G\}$ 为一个商群。注意，每个 $g \cdot N$ 都是一个左陪集。群G 的合成列是一个序列 G_0，.....， G_n，其中 $G_n = G$，并且对 $i \in [0, n-1]$而言，每个 G_i都是 G_{i+1}的极大真正规子群。必然， G_0 是平凡群。按照标准记法，

$$\{e\} = G_0 \triangleleft G_1 \triangleleft \cdots \triangleleft G_{n-1} \triangleleft G_n = G.$$

对于每个 $i \in [1, n]$，因子群 G_i/G_{i-1} 称为该合成列的因子。可解群也可以等价地定义为这样一个群：它具有一个合成列，其中所有因子都是素数阶循环群。

D.4 代数自动机理论

我们介绍 代数自动机理论的必要背景，遵循 [D¨om¨osi 和 Nehaniv， 2005] .

连接自动机与半群。半自动机 $\langle \Sigma, Q, \delta \rangle$ 的变换集合由每个变换 $\delta_\sigma(q) = \delta(q, \sigma)$（其中 $\sigma \in \Sigma$）组成。我们考虑由变换构成的半群，其中结合二元运算为函数复合。半自动机 A 的特征半群 S(A) 是由其变换生成的半群。自动机的特征半群就是其半自动机的特征半群。给定一个半群 S，由 S 定义的半自动机为 $A_S = \langle S, S, \delta \rangle$，其中 $\delta(s_1, s_2) = s_1 \cdot s_2$。注意，S($A_S$) $= S$。置换半自动机是指其变换均为排列的半自动机。无群半自动机是指其特征半群中仅有平凡群作为除子的一种半自动机。

同态表示。满射函数 ψ：$Q \to Q'$ 是从 $A = \langle \Sigma, Q, \delta \rangle$ 到 $A' = \langle \Sigma, Q', \delta' \rangle$ 的同态，如果对于每个 $\sigma \in \Sigma, q \in Q$，都有 $\psi(\delta(q, \sigma)) = \delta'(\psi(q), \sigma)$。在这种情况下，我们称 A' 是 A 的同态像。如果 ψ 是双射的，则称 A 和 A' 同构。给定一个半自动机 $A = \langle \Sigma, Q, \delta \rangle$，如果半自动机 $A' = \langle \Sigma', Q', \delta' \rangle$ 是 A 的子半自动机，即 $\Sigma' \subseteq \Sigma, Q' \subseteq Q$，且 δ' 是 δ 在 $\Sigma' \times Q'$ 上的限制，并且对于每个 $q' \in Q', \sigma' \in \Sigma'$ 都满足 $\delta'(q', \sigma') \in Q'$，那么它就是一个子半自动机。如果半自动机 A_1 是半自动机 A_2 的某个子半自动机的同态像，那么我们称 A_1 由 A_2 同态表示。

分解定理。首先，我们陈述 Jordan–H¨older 分解定理，该定理为分解由群定义的半自动机提供了一种方法。这是对 [D¨om¨osi 和 Nehaniv 的定理 1.19 的重新表述， 2005], 该定理以缠绕积的形式表述。

定理 16（Jordan–H¨older 分解定理）。如果 G 是一个群，其合成列为 $G_0 \triangleleft G_1 \triangleleft \cdots \triangleleft G_{n-1} \triangleleft G_n$，那么由 G 定义的半自动机可由半自动机级联 $(\phi_1, A_1) \ltimes \cdots \ltimes (\phi_n, A_n)$ 同态表示，其中 A_i 是由单群 G_i/G_{i-1} 定义的半自动机。

然后，我们陈述 Krohn-Rhodes 分解定理，该定理适用于所有半自动机——参见 [D¨om¨osi 和 Nehaniv 的定理 3.1 2005]。

定理 17（Krohn-Rhodes 分解定理）。设 A 是一个半自动机， G_1、.....、G_n 是整除特征半群 S(A) 的单群。那么 A 可以由一个半自动机级联 $(\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$ 同态表示，其中每个 A_i 都由 $\{F, G_1, \ldots, G_n\}$ 中的某个半群定义，而 F 是触发器幺半群。当 A 是一个置换半自动机时，可以省去 F。反过来，设 $(\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$ 是一个同态表示半自动机 A 的半自动机级联。如果触发器幺半群 F 的子半群 S 或单群 S 整除 S(A)，那么对于某个 $i \in [1, n]$， S 整除 S(A_i)。

E 初步结果

我们证明两个初步结果（命题 3 和命题 7），借此可以利用同态表示这一概念。这些结果具有新颖性，尽管其中也包含一些与文献中已有结果相同的思路，例如，[Ginzburg，1968年；Arbib，第2卷。

E.1 同态表示蕴含等价性

我们陈述并证明命题 3。

命题 3. 如果半自动机 D_1 同态表示半自动机 D_2，那么每个以半自动机 D_2 为基础的自动机都等价于一个以半自动机 D_1 为基础的自动机。

证明。考虑半自动机 D_1 和 D_2。假设 D_1 同态表示 D_2。由此可知，存在 D_1 的一个子半自动机 D_1'，以及一个从 D_1' 映到 D_2 上的同态 ψ。令 $D_1' = \langle \Sigma_1, Q_1, \delta_1 \rangle$ 和 $D_2 = \langle \Sigma_2, Q_2, \delta_2 \rangle$，并考虑一个具有半自动机 D_2 的任意接受器 A_2，

$$A_2 = \langle \Sigma_2, Q_2, \delta_2, q^2_{\text{init}}, \Gamma, \theta_2 \rangle.$$

让我们选择 $q^1_{\text{init}} \in Q_1$，使得 $\psi(q^1_{\text{init}}) = q^2_{\text{init}}$。这之所以存在，是因为 ψ 是满射的。我们构造自动机

$$A_1 = \langle \Sigma_2, Q_1, \delta_1, q^1_{\text{init}}, \Gamma, \theta_1 \rangle,$$

其输出函数为 $\theta_1(q) = \theta_2(\psi(q))$。然后我们证明 A_1 和 A_2 是等价的。我们考虑 A_1 和 A_2 在任意输入字符串 $w = \sigma_1 \ldots \sigma_n$ 上的执行。只需证明 A_1 在 w 上的最后一个输出等于 A_2 在 w 上的最后一个输出。

设 $q^1_0, \ldots, q^1_n$ 和 $q^2_0, \ldots, q^2_n$ 分别是 A_1 和 A_2 在所考虑的关于 w 的执行中的状态序列。也就是说，对于 $1 \le i \le n$， $q^1_0 = q^1_{\text{init}}$ 和 $q^1_i = \delta_1(q^1_{i-1}, \sigma_i)$。类似地，对于 $1 \le i \le n$， $q^2_0 = q^2_{\text{init}}$ 和 $q^2_i = \delta_2(q^2_{i-1}, \sigma_i)$。

作为辅助结果，我们证明对于每个 $0 \le i \le n$ 都有 $q^2_i = \psi_2(q^1_i)$。我们对 i 进行归纳法证明。

在基础情形中， $i = 0$ 和 $q^2_0 = \psi(q^1_0)$ 等价于 $q^2_{\text{init}} = \psi(q^1_{\text{init}})$，根据定义可知该结论成立。

在归纳情形中， $i > 0$，并假设 $q^2_{i-1} = \psi(q^1_{i-1})$。我们需要证明 $q^2_i = \psi(q^1_i)$。根据上述 q^1_i 和 q^2_i 的定义，前者可以改写为

$$\delta_2(q^2_{i-1}, \sigma_i) = \psi(\delta_1(q^1_{i-1}, \sigma_i)).$$

我们交换等式左边和右边，

$$\psi(\delta_1(q^1_{i-1}, \sigma_i)) = \delta_2(q^2_{i-1}, \sigma_i).$$

利用归纳假设 $q_{i-1}^2 = \psi(q_{i-1}^1)$，我们得到

$$\psi(\delta_1(q_{i-1}^1, \sigma_i)) = \delta_2(\psi(q_{i-1}^1), \sigma_i),$$

这成立是因为 ψ 是从 D_1 到 D_2的同态。这证明了辅助结果。

A_1 接受 $\sigma_1 \dots \sigma_n$ 当且仅当 $\theta_1(q_n^1) = 1$，类似地，A_2 接受 $\sigma_1 \dots \sigma_n$ 当且仅当 $\theta_2(q_n^2) = 1$。我们有

$$\theta_1(q_n^1) = \theta_2(\psi(q_n^1)) = \theta_2(q_n^2),$$

其中，第一个等式根据 θ_1的定义成立，第二个等式则根据辅助结果成立。因此，A_1 和 A_2等价。现在，A_1 的半自动机是 D_1'，而不是命题陈述所要求的 D_1。不过，显然可以将 A_1 扩展为一个等价的自动机 A_1'，使其半自动机为 D_1，因为 D_1 中不属于 D_1' 的状态无法从 A_1 的初始状态到达，因此不会影响输出。这证明了该命题。 □

E.2 等价性与典范性蕴含同态表示

本节将陈述并证明命题 7。我们首先给出一些预备定义和命题。

定义12. 考虑一个定义在 Σ上的语言 L。当且仅当对每个字符串 $z \in \Sigma^*$，等价关系$(xz \in L) \Leftrightarrow (yz \in L)$成立时，两个字符串 x、$y \in \Sigma^*$ 具有关系 $x \sim_L y$。由此得到的等价类集合记作 $\Sigma^*/\sim_L$，字符串 $x \in \Sigma^*$ 的等价类记作 $[x]_L$。

定义13。考虑一个自动机 A。两个输入字符串 x, y 当且仅当读取 x 和 y 后 A 的状态相同，才满足关系 $x \sim_A y$。由此得到的等价类集合记作 $\Sigma^*/\sim_A$，字符串 $x \in \Sigma^*$ 的等价类记作 $[x]_A$。

命题　　命题 4。对于每个识别语言 L 的自动机 A，都有 $x \sim_A y$，这意味着 $x \sim_L$　　y.

证明。假设 $x \sim_A y$，即这两个字符串将 q 中的 A 导向同一个状态。对于每个字符串 z，A 在 xz 和 yz 上的输出都是 $A^q(z)$。因此，$A(xz) = A^q(z)$，并且 $A(yz) = A^q(z)$。由于 A 识别 L，可知 $xz \in L \Leftrightarrow yz \in L$，从而有 $x \sim_L y$。 □

命题 5。考虑一个规范自动机 A，它在字母表 Σ 上识别语言 L。A 的状态与 $\Sigma^*/\sim_L$ 的等价类一一对应。对于任意将 q 导向的字符串 x，状态 q 对应于等价类 $[x]_L$。

证明。令 ψ 为上述对应关系。首先，ψ 将每个状态映射到某个等价类，因为每个状态都是可达的，这是因为 A 是规范的。其次，根据命题 4，ψ 将每个状态至多映射到一个等价类。第三，ψ 是满射，因为每个等价类 $[x]_L$ 都由 ψ 分配给通过 x 到达的状态。第四，ψ 是单射，因为不存在不同的状态 q、q'，使得对于 x 导向 q 且 x' 导向 q' 的 $[x]_L = [x']_L$，满足该条件。等式 $[x]_L = [x']_L$ 仅在 $x \sim_L x'$ 时成立，而这仅在对于每个非空 z 都有 $xz \in L \Leftrightarrow x'z \in L$ 时成立；由于 A 识别 L，这仅在 $A^q = A^{q'}$ 时成立。由于 A 是规范的，因此处于约简形式，我们有 $A^q = A^{q'}$ 不成立，从而 ψ 是单射。因此，ψ 是所要求的一一对应。 □

命题 6。考虑一个以 A 为输入字母表的连通自动机。A 的状态与等价类 $\Sigma^*/\sim_A$一一对应。状态 q 对应于等价类 $[x]_A$，其中 x 是任何可到达 q的字符串。

证明。设 ψ 为上述一一对应关系。首先，ψ 将每个状态映射到某个等价类，因为每个状态都是可达的，这是由于 A 是连通的。其次，根据等价类 $\Sigma^*/\sim_A$的定义，ψ 将每个状态至多映射到一个等价类。第三，ψ 是满射，因为每个等价类 $[x]_A$ 都由 ψ 分配给由 x到达的状态。第四，ψ 是单射，因为不存在不同的状态 q、q'，使得 $[x]_A = [x']_A$ 对于 x 通向 q 且 x'通向 q'。等式 $[x]_A = [x']_A$ 仅当 $x \sim_A x'$时成立，而这仅当 x 和 x' 通向同一状态时成立。因此，ψ 是一一对应关系，正如所要求的。 □

命题7。若一个自动机　　n A_1 等价于一个典范的自动机 A_2，则 A_1 同态的半自动机　　*lly*
表示半自动机　　属于 A_2。

证明。考虑一个自动机 A_1 和一个典范的自动机 A_2。

$$A_1 = \langle \Sigma, Q_1, \delta_1, q_1^{\text{init}}, \Gamma, \theta_1 \rangle$$
$$A_2 = \langle \Sigma, Q_2, \delta_2, q_2^{\text{init}}, \Gamma, \theta_2 \rangle$$

假设这两个自动机等价，也就是说，它们识别相同的语言 L。根据命题 5，A_2 的每个状态都可以看作一个等价类 $[x]_L$。设 A_1' 为 A_1的可达子自动机，D_1' 为它的半自动机。

$$D_1' = \langle \Sigma, Q_1', \delta_1' \rangle$$

根据命题 6， A'_1 的每个状态都可以看作一个等价类 $[x]_{A_1}$。定义函数 ψ，它将 $[x]_{A_1}$ 映射到$[x]_L$。根据命题 4，由于 A_1 识别 L， ψ 是一个良定义的函数，也就是说，它不会为同一个输入指定多个值。

我们论证 ψ 是一个满射。满足同态定义所要求的函数。该函数是满射，因为 ce every 状态 q 在 A_2 中可达 e，因此存在 x 能够到达它，从而 ψ 将 $[x]_{A_1}$ 映射到 $[x]_L = q$。

在论证了上述性质之后，为了证明 ψ 是从 D'_1 到 D_2 的同态，只需证明对于每个 $q \in Q'_1$ 和 $\sigma \in \Sigma$，以下等式成立。

$$\psi\big(\delta'_1(q,\sigma)\big) = \delta_2\big(\psi(q),\sigma\big)$$

让我们考虑任意的 $q \in Q'_1$ 和 $\sigma \in \Sigma$。设 $x \in \Sigma^*$ 是一个在 A'_1中到达 q 的字符串。注意， q 可以被视为等价类 $[x]_{A_1}$。我们有以下等价关系：

$$\begin{aligned}
&\psi(\delta'_1(q,\sigma)) = \delta_2(\psi(q),\sigma)\\
&\Leftrightarrow \psi(\delta'_1([x]_{A_1},\sigma)) = \delta_2(\psi([x]_{A'_1}),\sigma)\\
&\Leftrightarrow \psi([x\sigma]_{A_1}) = \delta_2(\psi([x]_{A_1}),\sigma)\\
&\Leftrightarrow [x\sigma]_L = \delta_2(\psi([x]_{A_1}),\sigma)\\
&\Leftrightarrow [x\sigma]_L = \delta_2([x]_L,\sigma)\\
&\Leftrightarrow [x\sigma]_L = [x\sigma]_L.
\end{aligned}$$

最后一个等式显然成立，因此 ψ 是从 D'_1 到 D_2的同态。由于 D'_1 是 D_1的子半自动机，我们得出结论： D_1 同态表示 D_2。命题得证。 □

E.3 变换逻辑的素分解定理

我们沿用 Krohn-Rhodes 分解定理的思路，为变换逻辑引入一个分解定理。该定理分为两部分。第一部分（定理 18）说明如何将变换定义分解为关于素算子的程序。第二部分（定理 19）说明素算子确实是素的，即它们不能分解为其他素算子。

我们首先介绍一些预备定义。

定义14。给定一个算子 $\mathcal{T} = \langle X、\ T、\ \phi、\ \psi\rangle$，其特征半群 S($\mathcal{T}$) 是由 T 生成的半群。

定义15。如果一个算子的所有变换都是双射，则该算子是置换算子。

定义16. 如果程序 P_1 被程序 P_2 捕获，则这两个程序具有相同的输入变量集合，并且对于在 P_1中定义的每个变量 q、到 P_1的每个输入 I 以及遍历 I各位置的每个索引 t，均有（P_1， I， t） $\models q$当且仅当（P_2， I， t） $\models q$。如果单例程序 $\{r\}$ 被 P捕获，则定义 r 被程序 P 捕获。

定理 18。设 r 是一个带有算子 $\mathcal{T}$ 的变换定义，并设 G_1，...， G_n 是整除 S($\mathcal{T}$) 的单群。那么 r 可由一个程序捕获，该程序的素算子定义为 $\{F, G_1, \ldots, G_n\}$，其中 F 是触发器幺半群。此外，如果 $\mathcal{T}$ 是一个置换算子，则可以排除 F。

证明思路。我们首先将给定的定义 r 映射为一个自动机 A。然后，我们使用自动机的 Krohn-Rhodes 分解定理，为 A获得一个素分解 A'。最后，我们将 A' 映射为一个程序，确保级联的每个组件都映射到具有相同特征半群的变换算子。 □

证明。若 P $= \{G_1, \ldots, G_n\}$ 是置换算子，则令 P $= \{F, G_1, \ldots, G_n\}$，否则令 P $= \{F, G_1, \ldots, G_n\}$。令 $\mathcal{T} = \langle X、\ T、\ \phi、\ \psi\rangle$，并令 m 和 n 分别为其输入元数和输出元数。令定义 r 具有如下形式

$$p_1, \ldots, p_n \coloneqq \mathcal{T}(a_1, \ldots, a_m \mid x_0).$$

令 $V = \{a_1, \ldots, a_m\}$ 且令 $U = \{p_1, \ldots, p_n\}$。令 P 为单例程序 $\{r\}$。我们构造一个自动机 A，使其与 P 等价，并且具有与 $\mathcal{T}$ 相同的特征半群。自动机 A 定义为 $\langle \Sigma, Q, \delta, q_{\text{init}}, \Gamma, \theta\rangle$，其中输入字母表为 $\Sigma = 2^V$，状态集合为 $Q = X$，转移函数定义为 $\delta(q, \sigma) = \tau(q)$，其中 $\tau = \phi(b_1, \ldots, b_m)$，且 $\langle b_1, \ldots, b_m\rangle$ 是变量集合 σ 的指示向量，初始状态为 $q_{\text{init}} = x_0$，输出字母表为 $\Gamma = 2^U$，输出函数 $\theta(x)$ 返回由 $\psi(x)$ 所指示的变量集合。我们有， A 与 P 等价，其含义是：对于输入 $I = I_1, \ldots, I_\ell$ 到 P 以及任意 $t \in [1, \ell]$，均有 $(P, I, t) \models p_i$ 当且仅当 $p_i \in A(I_{1:t})$，其中 $I_{1:t} = I_1 \ldots I_t$。

根据 Krohn-Rhodes 分解定理（定理 17），存在一个半自动机级联 $C = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$，它同态表示 A 的半自动机，使得每个 A_i 都由 P 中的半群定义。

根据命题 3，存在一个自动机 $A' = \langle \Sigma, Q_C, \delta_C, q^C_{\text{init}}, \Gamma, \theta'\rangle$，其半自动机为 C，使得 A' 与 A 等价，因而也与 P 等价。特别地，它的状态为 $Q_C = Q_1 \times \cdots \times Q_d$，其初始状态为 $q^C_{\text{init}} = \langle q^1_{\text{init}}, \ldots, q^d_{\text{init}}\rangle$。我们构造一个程序 P'，使其与 A' 等价（因而也与 P 等价），并且它的算子由 P 定义。

首先，通过对级联 C 的分量数量 d 进行归纳法，我们构造一个程序 P_C，用于跟踪级联 C 的状态，具体含义如下。对于每个 $i \in [1,d]$，令 $Q_i = \{q_{i,1},\dots,q_{i,m_i}\}$。我们还将每个状态 $q_{i,\,1}$ 用作一个命题变量，以表示该状态。然后，P_C 按照以下含义跟踪级联 C 的状态：$(P_C, I, t) \models q_{i,\,j}$ 当且仅当 $q_{i,\,j}$ 是级联 C 在输入 $I_{1:t}$ 上执行时其状态的第 i 个分量。

在基础情形中，我们有 $C = (\phi_1, A_1)$。回顾一下，级联的输入字母表为 $\Sigma = 2^V$。此外，A_1 是半群型的，因此它具有 $A_1 = \langle Q_1, Q_1, \delta_1\rangle$ 这种形式。因此，ϕ_1 具有如下形式 $\phi_1 \colon \Sigma \to Q_1$。令 $\mathcal{A}_1 = \langle Q_1, T_1, \xi_1, \psi_1\rangle$ 为由 A_1 的特征半群定义的半群型算子。我们有，T_1 与 A_1 的变换集合相同。注意，函数 ξ_1 和 ψ_1 如正文所述是给定的。特别地，我们有，$\xi_1 \colon \mathbb{B}^{k_1} \to Q_1$ 是满射，而 $\psi_1 \colon Q_1 \to \mathbb{B}^{\ell_1}$ 是单射。现在，T_1 的每个变换都对应于一个元素 $x \in Q_1$。由于 ξ_1 是满射，对于每个 $x \in Q_1$，都存在一个二进制向量 $\vec{b}$，使得 $\xi_1(\vec{b}) = x$。任意固定一个二进制向量 $\vec{b}$，使其作为 x 的返回值，我们定义“逆”函数 $\xi_1^{-1}(x) = \vec{b}$。由此，我们可以定义复合 $f = \phi_1 \circ \xi_1^{-1}$，它是如下形式的函数：$f \colon \Sigma \to \mathbb{B}^{k_1}$。因此，对于输入 $\sigma \in \Sigma$，半自动机 A_1 接收 $\phi_1(\sigma) = x$，并应用变换 x；算子 $\mathcal{A}_1$ 也应用同一个变换 x，因为 $(\phi_1 \circ \xi^{-1} \circ \xi)(\sigma) = \phi_1(\sigma) = x$。此外，$f$ 是从变量集 $\Sigma = 2^V$ 到布尔元组的函数，因此它的每个投影 f_i 都可以视为布尔函数，并可由关于变量 V 的命题公式 $\varphi_{1,i}$ 表示。令 $a_{1,}\mathbf{1}, \dots, a_{1,k_1}$ 和 $b_{1,}\mathbf{1}, \dots, b_{1,k_1}$ 为新变量。我们引入如下静态定义 $r_{1,i}$

$$a_{1,i} \coloneqq \varphi_{1,i}$$

以及变换定义 r_{A_1} 定义为

$$b_{1,1},\dots,b_{1,k_1} \coloneqq \mathcal{A}_1(a_{1,1},\dots,a_{1,k_1} \mid q^1_{\text{init}}).$$

现在，基于 ψ_1，我们可以为变量 $Q_1 = \{q_{1,1},\dots,q_{1,m_1}\}$ 引入定义。由于 ψ_1 是单射，存在一个映射 $g_1 \colon \mathbb{B}^{\ell_1} \to \mathbb{B}^{m_1}$，使得对于每个 $q_{i,\,j} \in Q_1$，都有 g_1，即 $g_1(\psi_1(q_{i,\,j}))$ 是 $q_{i,\,j}$ 在 Q_1 中的独热编码。我们引入静态定义 $r_{q_{1,\,i}}$ 定义为

$$q_{1,i} \coloneqq \beta_{1,i}$$

其中，$\beta_{1,\,i}$ 是一个命题公式，表示 g_1 的第 i 个投影。于是，程序 $P_C = \{r_{A_1}, r_{a_{1,1}},\dots,r_{a_{1,k_1}}, r_{q_{1,1}},\dots,r_{q_{1,m_1}}\}$ 跟踪级联 C 的状态。

在归纳情形中，我们有 $C = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_{d-1}, A_{d-1}) \ltimes (\phi_d, A_d)$，其中有 $d \geq 2$，并且假设我们有一个程序 $P_{C'}$，用于级联 $C' = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_{d-1}, A_{d-1})$。回顾一下，级联 C' 的状态在 $P_{C'}$ 中由变量 $q_{i,\,j}$ 表示。级联中的每个组件都会在更新之前读取前置组件的状态。为了捕捉这一点，我们引入：

(i) 新变量 $Q'_i = \{q'_{i,1},\dots,q'_{i,m_i}\}$ 和延迟定义

$$q'_{i,j} \coloneqq \mathcal{D}\, q_{i,j}$$

(ii) 新变量 $Q''_i = \{q''_{i,1},\dots,q''_{i,m_i}\}$ 以及满足 $q_{i,\,j} \neq q^i_{\text{init}}$ 的 i、j 的以下静态定义，

$$q''_{i,j} \coloneqq q'_{i,j} \tag{5}$$

(iii) 满足 $q_{i,\,j} = q^i_{\text{init}}$ 的 i、j 的以下静态定义，

$$q''_{i,j} \coloneqq q'_{i,j} \vee (\neg q'_{1,i}, \wedge \cdots \wedge \neg q'_{i,m_i}). \tag{6}$$

注意，在第一个时间点，根据延迟算子的语义，每个 $q'_{i,\,j}$ 均为假，因此定义（5）和（6）确保 $q''_{i,\,j}$ 成立，当且仅当 $q_{i,\,j} = q^i_{\text{init}}$ 成立。在后续时间点，某些 $q'_{i,\,j}$ 成立，因此只有当定义（6）的第一个析取项 $q'_{i,\,j}$ 成立时，定义（6）才会被触发。归纳情形的其余部分与基础情形类似，唯一的区别在于，相关变量现在是 $V \cup Q''_1 \cup \cdots \cup Q''_{d-1}$，其中 V 是输入变量，而 $Q_1 \cup \cdots \cup Q_{d_1}$ 在更新之前跟踪 C' 的状态。附加变量不会影响论证，因为现在可以将它们视为输入变量。因此，按照上述相同的论证，我们引入以下静态定义

$$a_{d,i} \coloneqq \varphi_{d,i}$$

变换定义

$$b_{d,1},\dots,b_{d,k_d} \coloneqq \mathcal{A}_d(a_{d,1},\dots,a_{d,k_d} \mid q^d_{\text{init}}),$$

以及静态定义

$$q_{d,i} \coloneqq \beta_{d,i}$$

然后，我们得到，通过使用上述定义扩展 $P_{C'}$ 所获得的程序 P_C，能够跟踪级联 C 的状态。

最后，为了得到能够捕获初始程序 P' 的程序 P，只需捕获自动机 A'的输出函数 θ'。我们有，该函数具有如下形式 $\theta\colon\ Q_1 \times \cdots \times Q_d \to \Gamma$，其中 $\Gamma = 2^U$ 和 $U = \{p_1, \ldots, p_n\}$。我们为每个 p_i 构造一个静态定义 r_{p_i}，如下所示

$$p_i \coloneqq \alpha_i$$

其中，α_i是关于变量 $Q_1 \cup \cdots \cup Q_d$的命题公式，当且仅当 $p_i \in \theta$（q_1，...，q_d）为真；它之所以存在，是因为每个布尔函数都可表示为一个命题公式。于是，程序 $P' = P_C \cup \{r_{p_1}, \ldots, r_{p_n}\}$ 捕获 P。 □

定理 19。如果带有素算子 $\mathcal{P}$的变换定义 r可由带有算子 $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$的程序捕获，那么 $\mathsf{S}(\mathcal{P})$ 整除 $\{F, \mathsf{S}(\mathcal{T}_1)$ ...，$\mathsf{S}(\mathcal{T}_n)\}$ 中的某个半群，其中 F是触发器幺半群。

证明思路。首先，我们将给定的定义 r 映射为一个自动机 A。其次，我们将给定的程序 P 映射为一个等价的级联 C，确保每个算子都映射到一个具有相同特征半群的组件。特别地，我们有 C捕获了 A，因此，根据克罗恩-罗兹定理的逆命题，A 的半群整除 C的某个单一组件的半群。由此，我们得出结论：$\mathcal{P}$ 的特征半群整除某个算子 $\mathcal{T}_i$的特征半群。 □

证明。设定义 r 的形式为

$$p_1, \ldots, p_n \coloneqq \mathcal{P}(a_1, \ldots, a_m \mid x_0).$$

令 $\mathcal{P} = \langle X, T, \phi, \psi \rangle$，并令 m 和 n 分别为其输入元数和输出元数。令 $V = \{a_1, \ldots, a_m\}$，并令 $U = \{p_1, \ldots, p_n\}$。令 P 为单例程序 $\{r\}$。首先，我们构造一个自动机 A，它与 P 等价，并且与 $\mathcal{P}$ 具有相同的特征半群。自动机 A 定义为 $\langle \Sigma, Q, \delta, q_{\text{init}}, \Gamma, \theta \rangle$，其中输入字母表为 $\Sigma = 2^V$，状态集合为 $Q = X$，转移函数定义为 $\delta(q, \sigma) = \tau(q)$，其中 $\tau = \phi(b_1, \ldots, b_m)$，$\langle b_1, \ldots, b_m \rangle$ 是变量集合 σ 的指示向量，初始状态为 $q_{\text{init}} = x_0$，输出字母表为 $\Gamma = 2^U$，输出函数 $\theta(x)$ 返回由 $\psi(x)$ 指示的变量集合。我们有，A 与 P 等价，即对于从 P 接收的每个输入 $I = I_1, \ldots, I_\ell$ 以及每个 $t \in [1, \ell]$，有 $(P, I, t) \models p_i$ 当且仅当 $p_i \in A(I_{1:t})$，其中 $I_{1:t} = I_1 \ldots I_t$。

令 P' 为捕获 P 的程序，其中包含 $\{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$ 中的算子。我们构造半自动机级联 $C = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$，用于捕获 P' 中由变换定义和延迟定义所定义的变量。

我们根据 P' 中变换定义和延迟定义的数量进行归纳法证明。在基础情形中，不存在延迟定义或变换定义，因此该断言平凡地成立。在归纳情形中，我们考虑程序 P''，它是通过移除一个不被其他变换定义或延迟定义所依赖的变换定义或延迟定义 r' 而得到的。根据归纳法假设，我们有一个级联 $C' = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_{d-1}, A_{d-1})$，用于捕获 P'' 中由变换定义或延迟定义所定义的变量。

我们将其扩展为级联 $C = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_{d-1}, A_{d-1}) \ltimes (\phi_d, A_d)$，用于捕获 P'。我们分别考虑两种情形。在第一种情形中，r' 是延迟定义

$$p \coloneqq \mathcal{D}\, a.$$

令 f 为一个函数，该函数根据 P'' 中由延迟定义或变换定义所定义的变量，返回 a 的真值。于是有：$\phi_d(\sigma, q_1, \ldots, q_{d-1})$ 在 $f(x) = 1$ 时返回 set，否则返回 reset。于是，A_d 是触发器半自动机。因此，我们有 C；特别是，其最后一个组件（ϕ_d, A_d）捕获了由延迟定义所定义的变量 p，因为 q_d 将是变量 a 的前一个真值。

在第二种情况下，我们有 r' 是一个变换定义

$$p_1, \ldots, p_n \coloneqq \mathcal{T}(a_1, \ldots, a_n \mid x_0).$$

选择自动机 A_d 为由 $\mathcal{T}$ 的特征半群定义的半自动机。函数 $\phi_d(\sigma, q_1, \ldots, q_{d-1})$ 的选择如下。首先注意，变量 $a_1, \ldots, a_n$ 的值由输入变量 σ 以及 P'' 中由延迟定义或变换定义所定义的变量 $b_1, \ldots, b_{d-1}$ 的某些函数 $f_1, \ldots, f_n$ 决定。特别地，$f_1, \ldots, f_n$ 由 P'' 中的静态定义定义，并且可以通过对此类定义进行归纳来构造。其次注意，更新前状态 $q_1, \ldots, q_{d-1}$ 捕获变量 $b_1, \ldots, b_{d-1}$ 在前一个时间点的值，因此更新后状态 $q'_1, \ldots, q'_{d-1}$ 捕获变量 $b_1, \ldots, b_{d-1}$ 在当前时间点的值。现在，令 $\phi\colon\ \mathbb{B}^n \to T$ 为所考虑定义中变换算子的解码函数，并将变量 $\sigma, q'_1, \ldots, q'_{d-1}$ 的列表记为 $\mathbf{x}$。于是，从输入 σ 和状态 q'_i 出发，变量 $a_1, \ldots, a_n$ 的值函数式地确定为 $f_1(\mathbf{x}) \ldots, f_n(\mathbf{x})$。反过来，每个状态 q'_i 都是 $\sigma, q_1, \ldots, q_{d-1}$ 的函数。事实上，如果令 δ_i 为 A_i 关于 $i \in [1, d-1]$ 的转移函数，则每个 q'_i 在基础情形中归纳定义为 $q'_1 = \delta_1(q_1, \sigma)$，而在归纳情形 $i \in [2, d-1]$ 中定义为 $q'_i = \delta_i(q_i, \langle \sigma, q'_1, \ldots, q'_{d-1} \rangle)$。因此，我们可以选择 $\phi_d(\sigma, q_1, \ldots, q_{d-1}) = \phi(f_1(\mathbf{x}) \ldots, f_n(\mathbf{x}))$。

我们有，由 A 与恒等输出及初始状态 x_0 得到的自动机是典范的。于是，根据命题 7，我们有 C 同态表示 A。由于 $\mathcal{P}$ 是素的，我们有，$\mathcal{P}$ 的特征半群 S 是触发器幺半群或单群。我们有，S 也是 A的特征半群。根据 Krohn-Rhodes 分解定理（定理 17）的逆定理，可知 S 整除某个 A_i的半群，而该半群是某个 $\mathcal{T}_i$的特征半群。

□

F 证明

我们按照主要体部中各项结果陈述的相同顺序，给出所有结果的证明。

F.1 定理 1 的证明

定理 1. 对于由所有有限算子组成的集合 A， $\mathcal{L}$(A) 的表达能力为正则语言。

证明. 我们分两部分证明该定理。

首先，要证明程序识别的每种语言都是正则语言，只需证明每个程序都存在一个与之等价的自动机。这一点已在定理 19 的证明中得到证明。

接着，要证明每种正则语言都能由变换逻辑表示，只需证明每个自动机接受器都能转换为程序。考虑一个自动机 $A = \langle \Sigma, Q, \delta, q_{\text{init}}, \Gamma, \theta \rangle$。令 T由每个变换 $\delta_\sigma(q) = \delta(q, \sigma)$（其中 $\sigma \in \Sigma$）组成。令 $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$，并令 $Q = \{q_1, \ldots, q_n\}$。再将 $\sigma_1, \ldots, \sigma_m$ 和$q_1, \ldots, q_n$ 也视为命题变量。令 $\phi(\vec{b})$ 为一个函数，它将每个独热向量 $\vec{b} \in \mathbb{B}^m$ 映射到对应的变换 δ_{σ_i}。令 $\psi(\vec{b})$ 为一个函数，它将每个状态 $q \in Q$ 映射到其独热编码。定义变换算子 $\mathcal{T} = \langle Q, T, \phi, \psi \rangle$。我们引入变换定义 r_{t}，定义如下

$$q_1, \ldots, q_n \coloneqq \mathcal{T}(\sigma_1, \ldots, \sigma_m \mid q_{\text{init}})$$

以及静态定义 r_{out} 定义为

$$out \coloneqq \alpha$$

其中 α 是变量 Q 上的命题公式，当且仅当 $\theta(q) = 1$时为真。那么，程序 $P = \{r_{\text{t}}, r_{\text{out}}\}$与给定的自动机接受器 A 等价，具体而言，对于每个字符串 $s = \sigma_{i_1} \ldots \sigma_{i_\ell}$，都有 $A(s) = 1$ 当且仅当 $(P, I, \ell) \models$输出，其中 $I = \{\sigma_{i_1}\}, \ldots, \{\sigma_{i_\ell}\}$。 □

F.2 定理 2 的证明

定理 2。 对于由所有素算子组成的集合 P，逻辑 $\mathcal{L}$(P) 的表达能力是正则语言 .

证明。我们分两部分证明该定理。

首先，根据定理 1，表达能力属于正则语言。

然后，同样根据定理 1，我们知道每个正则语言都由某个程序 P识别。根据定理 18，我们可以将 P中的每个变换定义替换为一个基于素算子的程序。通过将所有基于非素算子的定义替换掉，我们便得到了所需的程序。 □

F.3 定理 3 的证明

定理 3。 对于 $\mathcal{F}$ 触发器算子，逻辑 $\mathcal{L}(\mathcal{F})$ 的表达能力是星自由正则语言 *s.*

证明。我们分两部分证明该定理。

首先，我们证明 $\mathcal{L}(\mathcal{F})$ 的表达能力包含于无星正则语言中。考虑 $\mathcal{L}(\mathcal{F})$ 的一个查询 (P, q)，并令 L 为其识别的语言。根据定理 19 的证明中给出的构造，我们可以构造一个与 (P, q) 等价的触发器级联A，因此它识别 L。根据 Ginzburg 的第 7.12 节中的引理 C， 1968], 每个由触发器级联识别的语言都是无星的，因此 L 也是无星的。由于 (P, q) 是 $\mathcal{L}(\mathcal{F})$ 的任意查询，我们得出 $\mathcal{L}(\mathcal{F})$ 的表达能力包含于无星正则语言中。

然后，我们证明 $\mathcal{L}(\mathcal{F})$ 的表达能力包含无星正则语言。考虑一个无星正则语言 L。根据 Sch¨utzenberger 关于无星正则语言与非计数正则语言之间的对应关系， 1965], 存在一个非计数自动机。根据 Ginzburg 的第 7.12 节中的引理 A， 1968], 存在一个无群自动机 A，即没有非平凡群整除 A 的特征半群 S(A)。于是存在一个程序 P，其算子 $\mathcal{A}$ 由 S(A) 定义，并且该程序识别L； P的构造过程与定理 1 的证明相同。然后根据定理 18，存在一个程序 P'，其唯一算子是 $\mathcal{F}$。由于 L 是任意的无星正则语言，我们得出 $\mathcal{L}(\mathcal{F})$ 的表达能力包含无星正则语言。 □

F.4 定理 4 的证明

定理 4：对于任意非空的群算子集合 G，逻辑 $\mathcal{L}(\mathcal{F}$,G) 的表达能力严格强于 $\mathcal{L}(\mathcal{F})$。

证明：首先，我们有 $\mathcal{L}(\mathcal{F}$, G) 的表达能力包含 $\mathcal{L}(\mathcal{F})$ 的表达能力。然后，为证明该定理，只需展示一个 $\mathcal{L}(\mathcal{F}$, G) 的程序，且该程序无法由 $\mathcal{L}(\mathcal{F})$ 的任何程序所表达。

考虑一个群算子 $\mathcal{G} \in$ G，以及一个形式为 r 的变换定义

$$p_1, \ldots, p_n \coloneqq \mathcal{G}(a_1, \ldots, a_m \mid x_0).$$

接下来，考虑任意程序 P of $\mathcal{L}(\mathcal{F})$。反设 P 能够表示 $\{r\}$。根据定理 19，有 $\mathsf{S}(\mathcal{G})$整除触发器幺半群 F。我们证明这将导致矛盾。我们有 $\mathsf{S}(\mathcal{G})$是一个群 G。根据定义，当 G 整除 F时，存在一个同态 ψ，它从 F的某个子半群映射到 G。令 $F = \{a, b, e\}$ 具有 e 单位元。我们有 $a \cdot s = s$ 和 $e \cdot a = a$。因此，$\psi(a) \cdot \psi(b) = \psi(a \cdot b) = \psi(b)$，并且 $\psi(e) \cdot \psi(b) = \psi(e \cdot b) = \psi(b)$。这意味着 $\psi(b)$在 G中不具有逆元；但由于 G 是一个群，这就产生了矛盾。我们得出结论：P不能表示 $\{r\}$，因此 $\mathcal{L}(\mathcal{F}, \mathsf{G})$的表达能力严格包含 $\mathcal{L}(\mathcal{F})$的表达能力。 □

F.5 定理 5 的证明

定理 5（普适性）。对于 C 为所有素数阶循环算子的集合，且 S 为任意可解算子集合，$\mathcal{L}(\mathcal{F},\mathsf{C})$ 的表达能力包含 $\mathcal{L}(\mathcal{F},\mathsf{S})$ 的表达能力。

证明。考虑程序 P of $\mathcal{L}(\mathcal{F}, \mathsf{S})$。按照定理 19 证明中的构造，我们可以构造一个自动机A，它捕获 P，且其半自动机是级联 $C = (\phi_1, A_1) \ltimes \cdots \ltimes (\phi_d, A_d)$，其中 A_i 是触发器半自动机，或者是由可解群 $G \in \mathsf{S}$ 定义的类群半自动机。在后一种情况下，我们有 G 具有一个合成列，其因子是素数阶循环群。令 C_i 表示这些因子。根据 Jordan–H¨older 分解定理（定理 16），A_i 被同态表示为半自动机级联 $(\phi_i, 1, A_{i,1}) \ltimes \cdots \ltimes (\phi_i, 1, A_{i,n_1})$，其中每个半自动机 $A_{i,j}$ 都由 $\mathcal{C}_i$中的一个循环群定义。通过替换级联 C中的每个此类组件，我们得到一个级联 C'，它同态表示 C，并且其中每个半自动机要么是触发器半自动机，要么由素数阶循环群定义。根据命题 1，存在一个自动机 A'，其半自动机 C' 等价于 A。按照定理 18 证明中的构造，我们构造一个程序 P'，它等价于 A'，并且其中每个算子要么是触发器算子，要么是素数阶循环算子。特别地，我们有 P'捕获 P。由于 P是 $\mathcal{L}(\mathcal{F}, \mathsf{S})$ 的任意程序，我们得到 $\mathcal{L}(\mathcal{F}, \mathsf{S})$ 的表达能力包含于 $\mathcal{L}(\mathcal{F}, \mathsf{C})$ 的表达能力。 □

F.6 定理 6 的证明

定理 6（典范性）。给定两个由素数阶循环算子组成的集合 C_1 和 C_2，逻辑 $\mathcal{L}(\mathcal{F}, \mathsf{C}_1)$ 与 $\mathcal{L}(\mathcal{F}, \mathsf{C}_2)$ 的表达能力相同，当且仅当 $\mathsf{C}_1 = \mathsf{C}_2$。

证明。我们分两部分证明该定理。

首先，当 $\mathsf{C}_1 = \mathsf{C}_2$，时，$\mathcal{L}(\mathcal{F}, \mathsf{C}_1)$ 的表达能力等于 $\mathcal{L}(\mathcal{F}, \mathsf{C}_2)$ 的表达能力，因为它们是同一逻辑。

对于第二部分，假设 $\mathsf{C}_1 \neq \mathsf{C}_2$。存在一个算子 $\mathcal{C}_p$，其阶为某个素数 p，使得 $\mathsf{C}_1 \setminus \mathsf{C}_2$ 或 $\mathsf{C}_2 \setminus \mathsf{C}_1$。假设 $\mathsf{C}_2 \setminus \mathsf{C}_1$，这一情形；另一种情形的证明是对称的，故略去。具体而言，我们有 $\mathcal{C}_p \in \mathsf{C}_2$ 和 $\mathcal{C}_p \notin \mathsf{C}_1$。令 r 为如下形式的定义

$$p_1, \ldots, p_n \coloneqq \mathcal{C}_p(a_1, \ldots, a_m \mid x_0).$$

只需证明不存在一个能够刻画 r 的 $\mathcal{L}(\mathcal{F}, \mathsf{C}_1)$ 程序。反设存在一个能够刻画 r 的 P' $\mathcal{L}(\mathcal{F}, \mathsf{C}_1)$ 程序。由定理 19 可知，$\mathcal{C}_p$ 的特征半群 C_p 整除触发器幺半群 F，或者整除 C_1 中某个算子的特征半群。正如定理 4 的证明中所述，我们排除 C_p 整除 F 的情况。因此，考虑 C_p 整除某个循环群 C_q 的情形，其中 $q \neq p$。我们将证明这是不可能的。

存在一个从 C_q 的子半群 S 到 C_p 的同态。它是一个满射 $\psi\colon S \to C_p$，满足对于任意 $s_1, s_2 \in S$，都有 $\psi(s_1 \cdot s_2) = \psi(s_1) \cdot \psi(s_2)$。令 g 为 C_p 的一个生成元。由于 ψ 是满射，存在 $s \in S$ 使得 $\psi(s) = g$。我们有，C_p 的每个元素都可表示为 $n \in [0, p]$ 对应的 g^n。等价地，它具有如下形式：$\psi(s)^n = \psi(s) \cdot \psi(s) = \psi(s^n)$。因此，$s$ 是一个循环子群 $C \subseteq S \subseteq C_q$ 的生成元。C 的阶必整除 C_q 的阶，参见 Gallian, 2021] 的定理 4.3（[）。由此可知，要么 $C = \{e\}$，要么 $C = S = C_2$。在第一种情况下，由 $C = \{e\}$ 可知 $C_p = \{e\}$ 也成立，因为 C_p 的每个元素都具有形式 $\psi(s^n) = \psi(e^n) = \psi(e) = e$。这不可能，因为 C_p 的阶为 p，它是一个素数，因此不等于 1。在第二种情况下，我们有 $C = S = C_q$。由此可知，C_p 是 C_q 的同态像。由于 C_p 是单的（即它没有正规子群），因此 C_q 要么是平凡的，要么等于 C_p——参见 D¨om¨osi 和 Nehaniv 的第 7 页（[），或 Ginzburg, 1968] 的第 1.14 节末尾（[）。由于我们已假设 $q \neq 1$ 和 $C_p \neq C_q$，这两种情况都不成立。 □

F.7 定理 7 的证明

定理 7。存在　　两组素算子 $\mathsf{P}_1 \subset \mathsf{P}_2$，使得 $\mathcal{L}(\mathsf{P}_1)$ 和 $\mathcal{L}(\mathsf{P}_2)$ 具有相同的表达能力　　*ity.*

证明。考虑 $\mathsf{P}_1 = \{\mathcal{F}, \mathcal{A}_5\}$ 和 $\mathsf{P}_2 = \{\mathcal{F}, \mathcal{A}_5, \mathcal{C}_2\}$，其中 $\mathcal{A}_5$ 是由交错群 A_5 定义的算子，而 $\mathcal{C}_2$ 是由二阶循环群定义的算子。注意，$\mathcal{C}_2$ 和 $\mathcal{A}_5$ 是素算子，因为 C_2 和 A_5 是单群，参见 [Gorenstein 等人，2018]。还要注意，C_2 是 A_5 的子群，参见 [Groupprops，2023]。也就是说，A_5 的某个子群与 C_2 之间存在同构，这意味着 C_2 整除 A_5。

首先，$\mathcal{L}(\mathsf{P}_1)$ 的表达能力包含于 $\mathcal{L}(\mathsf{P}_2)$ 的表达能力中，因为 $\mathsf{P}_1 \subset \mathsf{P}_1$。为证明 $\mathcal{L}(\mathsf{P}_2)$ 的表达能力包含于 $\mathcal{L}(\mathsf{P}_1)$ 的表达能力中，我们考虑 $\mathcal{L}(\mathsf{P}_2)$ 的任意程序 P。只需证明，某个 $\mathcal{L}(\mathsf{P}_1)$ 的程序能够刻画 P。按照定理 19 的构造，我们可以给出一个级联 A 来刻画 P

其中，每个半自动机都由 $\{F, A_5, C_2\}$ 中的一个半群定义。由于 C_2 整除 A_5，我们可以将每个由 C_2 定义的半自动机替换为由 A_5 定义的半自动机，并得到一个等价的级联 A'。然后，按照定理 18 的构造，我们可以从 A' 得到一个能够刻画 P 且其算子属于 $\mathcal{P}_1$ 的程序。 □

F.8 定理 8 的证明

定理 8. 对于所有可解群算子 S 和所有素算子 P，以下无限表达能力层级成立：

$$\mathcal{L}(\mathcal{F}) \sqsubset \mathcal{L}(\mathcal{F}, \mathcal{C}_2) \sqsubset \mathcal{L}(\mathcal{F}, \mathcal{C}_2, \mathcal{C}_3) \sqsubset \cdots \sqsubset \mathcal{L}(\mathcal{F}, \mathbf{S}) \sqsubset \mathcal{L}(\mathbf{P}).$$

证明。添加素数阶循环算子时的严格包含关系由定理 6 得出。无限层次结构包含于 $\mathcal{L}(\mathcal{F}, \mathrm{S})$，这是因为每个循环群都是可解的。 □

F.9 定理 9 的证明

定理 9. 对于任意（可能无限的）多项式时间算子集合 T， $\mathcal{L}$(T) 的求值问题属于 PT第2卷。此外，存在一个多项式时间算子集合 H，使得 $\mathcal{L}$(H) 的求值问题是 PT第2卷-完全的。

证明。对于第一个结论，我们给出一种多项式时间算法。考虑一个查询 (P, q) with $P \in \mathcal{L}$，以及一个输入$I = I_1, \ldots, I_\ell$ for P。回顾一下，程序 P具有一个关联的无环依赖图。具体而言，每个变量对应一个节点，并且由于依赖图是无环的，每个节点都有一个深度。该算法按照时间 t 递增的顺序以及深度 n递增的顺序执行，计算：(i) 对于深度为 n 的所有变量 p，判断它们在时间 t是否为真；(ii) 对于深度为 n 的所有变换定义 r，计算其关联元素。具体而言，它计算 $(P, I, t) \models p$ 是否成立，以及满足 $(P, I, t) \models (r \mapsto x)$ 的元素x。对于 $n = 0$，变量 p 是一个输入变量，因此，要检查 $(P, I, t) \models p$，只需检查 $p \in I_t$。考虑 $n > 0$ 和 $t \geq 0$。根据 p 是由静态定义、延迟定义还是变换定义给出，我们分别讨论三种情况。在静态定义的情况下，算法已经计算出定义体部中的变量在时间 t 的真值。为了确定 p的值，只需计算体部中的命题公式，这可以在多项式时间内完成。在延迟定义的情况下，如果 $t = 1$，算法就确定 p 在时间 t 的真值为假；如果 $t \geq 1$，算法则将 p 在时间 t 的真值确定为体部变量在时间 $t - 1$的真值，而该值已经计算完毕。在变换定义的情况下，元素 x 满足以下条件：$(P, I, t) \models (r \mapsto x)$ 由 $x = \tau(x')$ 给出，其中 $\tau = \phi(\mu)$，且 $(P, I, t) \models (a_1, \ldots, a_m) \to \mu$ 和 $(P, I, t-1) \models (r \mapsto x')$；我们已经计算出变量 a_i 的真值，以及值 x'；由于所有算子都可以在多项式时间内执行，因此函数 ϕ 和 τ可以在多项式时间内计算。此外，由该定义所定义的变量 p的值由 b_i给出，该值由 $\psi(x) = \langle b_1, \ldots, b_n \rangle$确定。由于所有算子都可以在多项式时间内执行，因此 ψ可以在多项式时间内计算。由于定义的数量至多为 $|P|$，因此步骤数为 $O(\ell \cdot |P| \leq |I| \cdot |P|)$。

对于第二个主张，考虑算子族 $\mathcal{H}_n = \langle \mathbb{B}, T, \phi,$ 其中变换集合 T 是 set$(x) = 1$ 和 read$(x) = x$，函数 $\phi\colon \mathbb{B}^n \to \mathbb{B}$ 当且仅当其输入是长度为 n 的命题逻辑程序设计问题的一个正实例的二进制编码时返回 1。注意，命题逻辑程序设计是多项式时间完全的，参见 [Dantsin 等人， 2001]。令 H $= \{\mathcal{H}_n \mid n \geq 1\}$。我们给出一个从命题逻辑程序设计到逻辑 $\mathcal{L}$(H) 的求值问题的对数空间归约 ψ。由于命题逻辑程序设计是多项式时间完全的，这表明该问题是 PTIME-困难的，参见 [Dantsin 等人， 2001]。考虑由定义 P_n 构成的单例程序族

$$q \coloneqq \mathcal{H}_n(a_1, \ldots, a_n).$$

该归约将逻辑程序设计的一个实例 I 映射到输入为 $\psi(I) = \langle (P, q), I, 1 \rangle$ 的求值问题。我们有： I 是逻辑程序设计的一个正实例，当且仅当 $\psi(I)$ 是逻辑 $\mathcal{L}$(H) 的求值问题的一个正实例。事实上，我们有 $(P, I, 1) \models q$ 当且仅当 I 是逻辑程序设计的一个正实例。 □

F.10 引理 1 的证明

引理 1. 每个有限算子集都是多项式时间的。存在一个不是多项式时间的有限算子集。

证明。两个断言分别证明。

对于第一个断言，我们有，每个有限算子集都可以在常数时间 $O(1)$ 内求值，因为算子的大小有一个常数上界。

对于第二个断言，考虑算子族 $\mathcal{H}_n = \langle \mathbb{B}$、$T$、$\phi$、id$\rangle$，其中变换集 T为 set$(x) = 1$和 read$(x) = x$，函数 ϕ：$\mathbb{B}^n \to T$ 当其输入是长度为 n 的 Datalog 求值问题的正实例的二进制编码时返回 set。由于 Datalog 是 EXPTIME-完全的，参见 [Dantsin 等人，依据时间层次定理， 2001], PTIME $\subsetneq$ EXPTIME，并且在给定 x 和 μ 时，必须对 ϕ 求值才能确定 $\psi(\tau(x))$ 的值，其中 $\tau = \phi(\mu)$，我们得出结论：算子族 $\mathcal{H}_n$ 不是多项式时间的。 □

F.11 定理 10 的证明

定理 10. 对于任意有限算子构成的有限集 T， $\mathcal{L}$(T) 的求值问题属于 PTIME。

证明。根据引理 1，我们有 T 是多项式时间的。然后由定理 9 可得该定理。 □

F.12 命题 2 的证明

第2卷命题。这个 thr　eshold-counter 算子 $\{\mathcal{T}_n\}$ 和循环算子 $\{\mathcal{C}_n\}$ 具有紧凑表示。

证明。每个算子 $\mathcal{T}_n$ 都可以通过使用固定数量的位对符号 $\mathcal{T}$ 进行编码，并使用 $O(\log n)$ 位对索引 n 进行二进制编码来表示。同样的论证也适用于每个算子 $\mathcal{C}_n$。 □

F.13 引理第2卷的证明

引理 2. 算子族 $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$ 即使以紧凑形式表示，也是多项式时间的。

证明。我们分别处理这两类算子。首先讨论 $\{\mathcal{T}_n\}$。令 m 为编码 n所需的最小位数。注意到 $m \in O(\log n)$。我们回顾 $\mathcal{T}_n$的定义。我们有 $\mathcal{T}_n = \langle [0,n], T, \phi, \mathrm{id} \rangle$，其中 T和 ϕ定义如下。首先，我们定义 $T = \{\mathrm{inc}, \mathrm{id}\}$，其中 inc 定义为 $\mathrm{inc}(x) = \min(n, x+1)$，而 id 为恒等函数。其次，我们将 ϕ 定义为 $\phi(0) = \mathrm{id}$ 和 $\phi(1) = \mathrm{inc}$。我们有，函数 ϕ 的运行时间为常数时间，而所有其他函数都可以在元素 $x \in [0,n]$的二进制表示上以多项式时间运行，因此其运行时间关于算子的大小是多项式的，符合要求。

接下来我们讨论 $\{\mathcal{C}_n\}$。我们给出 $\mathcal{C}_n$ 的定义，从而明确该算子具有多项式时间。令 m 为编码 $n-1$所需的最小位数。注意， $m \in O(\log n)$。考虑函数 φ_n： 定义 $\mathbb{B}^m \to [0, n-1]$ 为

$$\varphi_n(b_1, \ldots, b_m) = \min\left(n-1, \sum_{i=0}^{m-1} b_i \cdot 2^i\right)$$

我们可以将 $\mathcal{C}_n$定义为 $\langle [0,n-1]$、T、φ_n、$\mathrm{id}\ \rangle$，其中 $T = [0, n-1]$，定义 τ为 $x(= x+\tau) \bmod n$。我们知道，函数 φ 在 m 上具有多项式时间，因此相对于算子的大小也具有多项式时间；并且所有其他函数都可以在元素 $x \in [0,n]$的二进制表示上以多项式时间运行，因此其运行时间相对于算子的大小也是多项式的，正如所要求的那样。 □

F.14 定理 11 的证明

定理 11。对于任意由有限算子组成的有限集 A， $\mathcal{L}$(A $\{\mathcal{T}_n\}$、$\{\mathcal{C}_n\}$) 的求值问题属于 PTIME，即使算子采用紧凑表示 $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$。

证明。根据引理 2， $\{\mathcal{T}_n\}$ 和 $\{\mathcal{C}_n\}$ 这两个族均为多项式时间。根据引理 1，族 A 是多项式时间的。因此，族 A $\cup \{\mathcal{T}_n\} \cup \{\mathcal{C}_n\}$ 是多项式时间的。于是，由定理 9 可得该定理。 □

F.15 定理 12 的证明

定理 12。对于任意 k，计算 $\mathcal{L}(\mathcal{F} \mid k)$ 属于 AC^0。

证明。给定一个查询（P, q），其中 $P \in \mathcal{L}(\mathcal{F}, k)$，只需证明存在一族 AC^0 电路，能够对每个输入 $I = I_1, \ldots, I_\ell$ 计算（P, I, ℓ） $\models q$到 P。回顾一下，AC^0 电路具有“非”门、无界扇入的“与”门和无界扇入的“或”门，并且具有常深度和多项式规模。对于带有触发器算子的变换定义， [Chandra 等人在定理 2.6 的证明中给出了一种已有构造，该构造展示了如何构造一个 AC^0 电路，计算触发器在时间 t存储的位。静态定义也对应于 AC^0 电路，因为程序具有常深度；延迟定义则对应于这样一种触发器：当输入为真时置位，否则复位（即存储前一位），因此也可以采用上述构造。因此，考虑到程序 P具有常深度，我们可以合并上述用于 t 从 1 到 ℓ 的电路，得到一个具有常深度和多项式规模的电路；其输出在输入 I 时为 1，当且仅当（P, I, ℓ） $\models q$。下面给出该电路更详细的构造。

电路的详细信息。设 a_1、.....、a_n 是给定程序 P 的输入变量。对于每个时间点 $t \in [1, \ell]$，电路都为输入 $I = I_1$、.....、I_ℓ 所指定的每个变量的真值赋值设置一个输入引脚，因此它具有 $a_{1,1}$、.....、$a_{n,\ell}$ 个输入引脚。构造过程通过引入门来确保：对于 P 中定义的每个变量 p 以及每个时间点 $t \in [1, p]$，都存在一个门 $g_{p,t}$，其输出对应于 $(P, I, t) \models p$。我们考虑一个定义 r，该定义涉及 P，并且 r 的体部中出现的所有变量所对应的门都已包含在电路中——由于程序具有无环依赖图，除非电路已经构造完成，否则这样的定义总是存在。根据定义 r 的类型，我们分别讨论三种情况。

首先，考虑 r 是变换定义的情况。已知 r 是带有触发器算子的变换定义。其形式如下。

$$p := \mathcal{F}(a, b \mid x_0)$$

这里我们采用 [Chandra 等人 1985] 的定理 2.6 中的构造。我们分别考虑两种情况。在第一种情况下，我们有 $x_0 = 0$。对于从 1 到 ℓ 的 t，我们用以下电路扩展目前构建的电路。

$$g_{p,t} = \bigvee_{1 \le i \le t} \left(g_{a,i} \wedge \bigwedge_{i < j \le t} \neg g_{b,j} \right)$$

注意，每个 $g_{a,i}$ 以及每个 $g_{b,j}$ 都是目前构建的电路中已经存在的门。直观地说，p 在时间 t 为真，当且仅当存在某个时间点 i，使得触发器已被置位（a 为真），且在其后的任何时间点都未被复位（b 为假）。情况 $x_0 = 1$ 的处理方式类似，只是要添加的电路如下：

$$g_{p,t} = \left(\bigwedge_{0<j\le t} \neg g_{b,j} \right) \vee \bigvee_{1\le i\le t} \left(g_{a,i} \wedge \bigwedge_{i<j\le t} \neg g_{b,j} \right)$$

其次，考虑 r 是延迟定义的情况。设该延迟定义具有以下形式。

$$p \coloneqq \mathcal{D}\, a$$

我们用以下电路扩展目前构建的电路，其中 $\bot$ 始终取值为假，

$$g_{p,1} = \bot,$$

以及以下电路，其中 t 从第2卷到 ℓ 变化，

$$g_{p,t} = g_{a,t-1}.$$

直观而言，p 在时间 1 为假；对于 $[2,\ell]$ 中的各个时间点，其真假取决于前一时间点上 a 的取值。

第三，考虑 r 是静态定义的情况。设 b_1、……、b_m 是 r 体部中的变量，p 是由 r 定义的变量。r 的体部为我们提供了一个电路，我们可以将其复制第8卷次，其中第 t 个副本的输入为 $g_{b_1,t}$、……、$g_{b_m,t}$。然后，第 t 个电路中有一个门用于定义 $g_{p,t}$。□

F.16 定理13的证明

定义17. 设 S 是一个半群。对于给定的 s_1、s_2、…、$s_n \in S$，计算元素 $s_1 \cdot s_2 \cdots s_n$ 的问题称为 S 的字问题。

以下定理是 [Barrington, 1989] 的定理 5。

定理 20（Barrington, 1989）。任意固定可解群的字问题 G 可 AC^0 归约到函数 $f_g(x) = x$ 模 g，其中 g 是 G 的阶。

定理 13. 对于任意深度 k，以及任意有限集 S（由可解群算子组成），$\mathcal{L}(\mathcal{F}, \mathrm{S} \mid k)$ 的求值属于 ACC^0。此外，存在一个可解群算子 $\mathcal{G}$，使得对于每个深度 $k \ge 1$，$\mathcal{L}(\mathcal{G} \mid k)$ 的求值不属于 AC^0。

证明。我们分两部分证明该定理。

对于第一部分，给定一个带有 $P \in \mathcal{L}(\mathcal{F}, \mathrm{S}\ k)$ 的查询 (P, q)，只需证明存在一族 ACC^0 电路，对于每个输入 I、P 以及每个时间点 t，计算 $(P, I, t) \models q$。我们回顾一下，ACC^0 电路是 AC^0 电路，其中还允许对任意 g 使用无界扇入的模-g 门，这些门计算其输入之和模 g。利用定理 20，每个带有可解群算子的变换定义都可以由 ACC^0 电路捕获。其他变换定义以及其余论证遵循定理 12 的证明。第二部分的证明之后，我们将在下面进一步详细说明电路的构造。

对于第二部分，我们考虑奇偶函数 $f\colon\ \mathbb{B}^* \to \mathbb{B}$ 该函数返回其输入位的奇偶性，即这些位之和模 2 的结果。我们考虑由 2 阶循环群定义的算子 $\mathcal{C}_2$，以及定义 r，其定义为

$$p \coloneqq \mathcal{C}_2(a \mid 1).$$

设 $P = \{r\}$。对于输入 I 到 P，我们有 $(\{r\}, I, t) \models p$ 当且仅当 a 在 I 中出现偶数次。也就是说，程序 P 捕获了奇偶性。由于奇偶性不属于 AC^0 [Furst 等人，1984], 和 $P \in \mathcal{L}(\mathcal{F}, \mathcal{C}_2, 1)$，我们得出，$\mathcal{L}(\mathcal{F}, \mathcal{C}_2, k)$ 的求值问题不属于 AC^0。注意，$\mathcal{C}_2$ 是所要求的可解群算子，因为每个循环群都是可解的。

电路的细节。证明第一部分所用的电路按照定理 12 的证明构造，但包含群算子的变换定义这一情形除外。接下来我们聚焦于这种情形。考虑一个可解群 $G = \langle X$、其对应的可解群算子 $\mathcal{G} = \langle X$、T、ϕ、$\psi \rangle$，以及一个包含该算子的变换定义。

$$p_1, \ldots, p_n \coloneqq \mathcal{G}(q_1, \ldots, q_m \mid x_0)$$

[Barrington 的定理 20 1989] 指出，给定 G 中的元素 $x_0, x_1, \ldots, x_t$，我们可以从 $x_0, x_1, \ldots, x_t$ 的二进制编码出发，使用 AC^0 电路和一个模-$|X|$门，计算乘积元素 $x_0 \cdot x_1 \cdots x_t$ 的二进制编码。请注意，[Barrington 1989] 假设使用固定编码；在我们的情形中确实如此，因为每个算子都有由 ϕ 和 ψ 确定的自身编码，而且算子集合 G 也是固定的。于是，到目前为止构造的电路包含门 $g_{q_1,t}, \ldots, g_{q_m,t}$，对于 $t \in [1,\ell]$，这些门为我们提供了 X 中与时间 t 的变换定义相关联的元素的二进制编码。我们添加常量门 $g_{q_1,0}, \ldots, g_{q_m,0}$，为我们提供初始元素 x_0 的二进制编码。根据定理 20，我们可以使用 AC^0 电路和一个模-$|X|$门，获得对于每个 $t \in [1,\ell]$ 的门 $g_{p_1,t}, \ldots, g_{p_n,t}$；这些就是变量 $p_1, \ldots, p_n$ 所需的门。

□

F.17 定理14的证明

以下定理是 [Barrington 1989]的定理 4。

定理 21（Barrington, 1989）。任意固定的非可解群的字问题在 AC^0 归约下是 NC^1 -完全的。

定理 14. 对于任意深度 k，以及任意包含非可解群的有限集 G，$\mathcal{L}(\mathcal{F}, \mathsf{G} \mid k)$ 的求值问题在 AC^0 归约下是 NC^1 完全的。

证明。根据定理 21，变换定义属于 NC^1, ，其中含有非可解群的变换定义可归约到 NC^1。含有可解群和触发器的变换定义，以及静态定义和延迟定义，如定理 13 和定理 12 的证明所述，可归约到 $\mathsf{ACC}^0 \subseteq \mathsf{NC}^1$ 。

反之亦然，因为我们 能将每个 NC^1 电路 AC^0-归约为一个群的字问题 [Barrington, 1989], ，并且
我们可以将词问题 以及任意 AC^0 电路表示为 $\mathcal{L}(\mathcal{F}, \mathsf{G}\ k)$ 的程序，其中 G 和 k 任意。 □

F.18 引理 3 的证明

引理 3。考虑一个过去时线性时序逻辑公式 $\varphi = \ominus p$。设 P 为由定义 $q := \mathcal{D}\, p$ 构成的单例程序。对于每个 I 的解释 φ 以及每个时间点 t，均有 $(I, t) \models \varphi$ 当且仅当 $(P, I, t) \models q$。

证明。由此可得 延迟定义的语义与之前算子的语义一致这一事实 . □

F.19 引理4的证明

引理 4. 设 $\varphi = a\,\mathcal{S}\,b$ 为一个过去时线性时序逻辑公式。设 P 为由以下两个定义组成的程序：$c :=\ \neg a$ 和 $p :=\ \mathcal{F}\ (b,\ c \mid 0)$。对于 φ 的任意解释 I 和任意时间点 t，均有（I，t）$\models \varphi$ 当且仅当（P，I，t）$\models p$。

证明思路。对 t 进行归纳法，并利用“since”算子语义的另一种递归定义。 □

证明。设 r_1 为定义

$$c := \neg a,$$

并设 r_2 为定义

$$p := \mathcal{F}(b, c \mid 0).$$

设 $I = I_1$，……，I_ℓ。我们对 t 从 1 到 ℓ 进行归纳法。

在基础情形 $t = 1$ 中，我们分别证明这两个蕴含关系。首先，证明 $(I, 1) \models (a\,\mathcal{S}\,b)$ 蕴含 $(P, I, 1) \models p$。我们有 $(I, 1) \models (a\,\mathcal{S}\,b)$ 蕴含 $b \in I_1$，因此 $(P, I, 1) \models (r_2 \mapsto 1)$，进而 $(P, I, 1) \models p$，如所需。其次，证明 $(P, I, 1) \models p$ 蕴含 $(I, 1) \models (a\,\mathcal{S}\,b)$。我们有 $(P, I, 1) \models p$，根据语义的第5点，这蕴含 $(P, I, 1) \models (r_2 \mapsto 1)$；而后者又蕴含（根据语义的第6点和 $\mathcal{F}$ 的定义）$(P, I, 1) \models b$，因此 $b \in \sigma_1$，进而 $(I, 1) \models (a\,\mathcal{S}\,b)$。

在归纳情形 $t \geq 2$ 中，我们假设 $(I, t-1) \models (a\,\mathcal{S}\,b)$ 当且仅当 $(P, I, t-1) \models p$。我们分别证明两个蕴含关系。首先，证明 $(I, t) \models (a\,\mathcal{S}\,b)$ 蕴含 $(P, I, t) \models p$。由 $(I, t) \models (a\,\mathcal{S}\,b)$ 可得 (i) $(I, t) \models b$，或 (ii) $(I, t-1) \models (a\,\mathcal{S}\,b)$ 且 $(I, t) \models a$。第一种情形与基础情形中的情形类似。考虑第二种情形，假设 $(I, t-1) \models (a\,\mathcal{S}\,b)$ 且 $(I, t) \models a$。因此，$a \in I_t$，由归纳假设可得 $(P, I, t-1) \models p$。由 $(P, I, t-1) \models p$ 可得 $(P, I, t-1) \models (r_2 \mapsto 1)$，因此只需证明 $(P, I, t) \not\models c$，而这由 $a \in I_t$ 成立。现在证明另一个蕴含关系，即 $(P, I, t) \models p$ 蕴含 $(I, t) \models (a\,\mathcal{S}\,b)$。假设 $(P, I, t) \models p$。由此可得 $(P, I, t) \models (r_2 \mapsto 1)$。根据 $\mathcal{F}$ 的定义，以下两个条件之一成立：(i) $(P, I, t) \models b$，因此 $b \in I_t$；或 (ii) $(P, I, t-1) \models (r_2 \mapsto 1)$ 且 $(P, I, t) \not\models c$（这等价于 $a \in I_t$）。我们分别考虑这两种可能情形。第一种情形下，$b \in I_t$ 蕴含 $(I, t) \models (a\,\mathcal{S}\,b)$。第二种情形下，$(P, I, t-1) \models p$。由归纳假设可得 $(I, t-1) \models (a\,\mathcal{S}\,b)$，这蕴含 $(I, t) \models (a\,\mathcal{S}\,b)$，因为 $a \in I_t$。

至此，证明完成。 □

F.20 定理 15 的证明

定理 15. 每个过去时线性时序逻辑公式 φ 都可以转换为 $\mathcal{L}(\mathcal{F})$ 的程序。该程序的大小与 φ 的大小呈线性关系，并且可以在对数空间内从 φ 计算得到。

证明。考虑一个过去时线性时序逻辑公式 φ。我们给出一个查询 (P, q)，使得对于 φ 的每种解释 I 以及每个时间点 t，都有 $(I, t) \models \varphi$ 当且仅当 $(P, I, t) \models q$。我们按照 φ 的解析树深度进行归纳。基础情形中，深度为零，解析树仅由根节点构成，且 $\varphi = a$，其中 a 是一个命题变量。相应的查询为 (P, q)，其中 q 是一个新变量，而 P 由定义 $q := a$ 构成。

在归纳情形中，解析树 φ 的深度 d 大于零，并且我们假设可以为每个解析树深度至多为 $d-1$的公式构造一个程序。我们根据 φ的顶层运算符（即与根节点关联的运算符）分情形讨论。在 $\vee$的情形中，给定与根节点关联的公式 α 和 β 的查询（P_α，a_α）和（P_β，a_β），

对于根节点的子节点，对应的查询为（P， q），其中 q是一个新变量， P是扩展了定义$q \coloneqq a_\alpha \vee a_\beta$的 $P_\alpha \cup P_\beta$。对于 $\wedge$同理。在 $\neg$的情形中，给定与根节点子节点关联的公式 α 的查询（P_α， a_α），所得查询为（P， q），其中 q 是一个新变量， P是扩展了定义 $q \coloneqq \neg a_\alpha$的 P_α。在 的情形中，给定与根节点子节点关联的公式 α 的查询（P_α， a_α），所得查询为（P， q），其中 q 是一个新变量， P是扩展了定义 $q \coloneqq \mathcal{D}\, a_\alpha$的 P_α。在 $\mathcal{S}$的情形中，给定与根节点子节点关联的公式 α 和 β 的查询（P_α， a_α）和（P_β， a_β），所得查询为（P， q），其中 q 是一个新变量， P是扩展了定义 $q \coloneqq \mathcal{F}\,(a_\beta,\ c \mid 0)$ 和 $c \coloneqq \neg a_\alpha$的 $P_\alpha \cup P_\beta$。

布尔算子的构造正确性源于它们在变换逻辑中的语义是标准语义，因此与它们在过去时线性时序逻辑中的语义一致。“before”算子的正确性由引理 3 得出。“since”算子的正确性由引理 4 得出。所得程序的规模是线性的，因为我们为 φ 的每个算子引入一个定义，而每个定义都只包含常数个变量。该翻译只需同时保留常数个指针和变量名，因此可以在对数空间内完成。 □