arXiv:2305.03832v1 [cs.LO] 5 May 2023

图变换线性时态逻辑的规范与验证*

Fabio Gadducci、Andrea Laretto 和 Davide Trotta

意大利比萨大学计算机科学系，fabio.gadducci@unipi.it、trottadavide92@gmail.com；爱沙尼亚塔林理工大学软件科学系，andrea.laretto@taltech.ee

摘要。 我们提出一种一阶线性时序逻辑，用于推理有向图的演化。其语义基于对应物范式，因此该逻辑能够表示图元素的创建、复制、合并和删除，以及图的拓扑结构如何随时间变化。随后，我们介绍正范式的表达形式，从而简化实际的验证过程。我们借助证明助手 Agda，以计算机辅助的形式化方式给出该逻辑的语法和语义；最后，重点阐述形式化工作的关键方面，以及在构造性证明助手中实际使用量化时序逻辑的方式。

关键词： 对应体语义 · 线性时序逻辑 · Agda 形式化。

1 引言

在形式化方法提供的众多工具中，时序逻辑已被证明是验证大型程序和独立程序最有效的技术之一。多年来，对这些逻辑的研究主要集中于改进验证过程中的算法过程，以及寻找这些逻辑中具有足够表达能力的片段，以规范复杂的多组件系统。人们已经为时序逻辑开发了多种模型，其中最典型的例子是迁移系统，也称为克里普克结构。在迁移系统中，每个状态表示系统的一种配置，而每个迁移则标识一种可能的状态演化。人们通常希望为模型所给出的状态和迁移赋予更多结构，例如将状态视为代数，将迁移视为代数同态。一个重要的应用场景是图逻辑 [9,10,12]，其中，状态被具体化为图，而迁移则是（部分）图态射的族。这些逻辑结合了时序推理与空间推理，能够表达图拓扑随时间发生变化的可能方式。

* 本研究部分受意大利教育、大学与科研部项目 PRIN 2017FTXR7S “IT-MaTTerS”以及比萨大学项目 PRA_2022_99 “FM4HD”的资助。

量化时序逻辑。在通常的时序逻辑中，例如 线性时序逻辑 和CTL [13], 模型的状态被视为原子的，命题对完整的状态成立；另一方面，图逻辑的定义性特征之一是，它们允许针对图中的单个元素进行推理并表达其性质。尽管存在不可判定性 [15,25],量化时序逻辑仍因其表达能力，以及量化可以遍历模型状态中元素的可能性，而被倡导用于这一场景。

然而，这些逻辑的语义模型需要一些巧思。考虑一个简单模型，其中有两个状态 s_0， s_1、两个迁移 $s_0 \to s_1$ 和 $s_1 \to s_0$，以及一个只出现在 s_0中的项 i 。这个项 i 是被销毁后重新创建的，还是只是一个被多次重复使用的标识符？文献中将这一问题称为跨世界同一性问题 [24,2]。一种解决方案是固定一个由全域项构成的集合，为模型各状态中出现的每个个体赋予同一性。由于每个项 i 都属于这个全域，因此在每次时间演化之后，它都恰好是同一个个体 s_1。然而，这意味着迁移基本上表现为状态中各项之间的单射；从概念上说，这种观点很难与上述简单模型相协调，因为在该模型中，我们描述了某个项的销毁与重新创建。同样，这也排除了克隆项的可能性，因为这无法与将演化步视为单射的观念相容。

对应体语义。刘易斯 [30]提出了这一问题的一种解决方案，即对应物范式：每个状态不再对应一个全局的元素集合，而是标识一个局部元素集合，并由（可能是部分的）态射将各个状态连接起来，把元素从一个状态映射到另一个状态。这样，我们就能形式化地描述实体被销毁、复制、（重新）创建或合并的情形，并妥善处理不同图之间个体的同一性问题。

在 [16], 中，采用基于对应关系的语义来引入带二阶量词的 μ-演算的集合论语义。这种模态逻辑提供了一种形式体系，能够为状态增添代数，并为迁移增添部分同态，从而涵盖图逻辑的情形。在 [17]中，借助关系预层并基于 [19,20]中提出的思想，将这些模型推广到范畴化框架。模型以范畴和关系预层（族）的形式表示，为基于部分同态的“状态即代数”方法提供范畴化表示。通过为模型中的范畴赋予单步 箭头的概念，可以捕捉系统的时间演进；随后利用这一范畴化框架引入二阶线性时序逻辑 QLTL。

经典语义和正范式。 我们首先在第 2 节介绍对应模型的概念，并在第 3 节中，采用标准的集合论视角，介绍我们的时序逻辑QLTL（诚然，其语法相当直接）及其基于对应物的语义，并将可满足性归纳定义为一个逻辑谓词。不同于 [16,17] 中使用偏函数的模型，我们将其推广为关系，从而通过允许元素具有多个对应物来对元素的复制进行建模。在第 4 节中，我们给出关于正范式的一些结果，其中模型可以使用

部分态射或关系，并突出说明二者的差异。正范式（即仅对原子公式定义否定）是时序逻辑的标准工具，因为它们既简化了时序逻辑的理论处理，也有助于实现模型检测 [26,5]算法。使用关系而非（可能是部分的）函数会削弱这类正范式的表达能力，并要求为逻辑引入附加算子。然而，个体复制是诸如 Sequi-Pushout 等 [7], 图变换形式系统的核心特征，因此值得研究。

Agda 中的时态逻辑。 我们工作的另一项贡献是使用依赖类型证明助手 Agda 对本文所介绍逻辑的 [36]模型、语义和正范式进行计算机辅助的形式化。我们在第 5 节中介绍机械化的主要方面；该方法可适用于对应物模型，其中的世界是任意多排序签名上的代数，尽管为便于本文阐述，我们将讨论范围限定于图签名。在证明助手中对时序逻辑进行形式化呈现有若干优点：它巩固了本文所阐述数学思想的正确性与连贯性，因为这些思想可以借助软件工具接受独立的检查和具体验证；此外，机械化还有效地提供了一个试验平台，可在其中表达、测试和实验这些逻辑的机制与有效性。

据我们所知，很少有时序逻辑的形式化借助证明助手完成，而这些形式化都没有配备基于对应关系的语义。本工作通过在交互式证明助手中嵌入线性时序逻辑（LTL）的一个量化扩展，使其能够对状态中的个体元素进行推理，向机器验证的时序逻辑应用迈出了一步。

2 对应模型

本节介绍我们用于系统演化的模型。我们考虑将对应模型具体化到这样一种情形：每个世界并非与一个单纯的（无结构的）个体集合相关联，而是与一个有向图相关联，其随时间的演化由保持图结构的适当关系表示。

定义 2.1。 一个（有向）图 是一个四元组 $G := \langle N, E, s, t\rangle$，满足$N$ 是一个节点集合，E 是一个边集合，并且 s、t： $E \to N$ 是两个函数，分别为每条边 $e \in E$指定一个源点 $s(e) \in N$和一个目标 节点 $t(e) \in N$。所有有向图的集合记为 *Graphs*。

定义 *2.2.* 两个图 $G = \langle N, E, s, t\rangle$ 和 $G' = \langle N', E', s', t'\rangle$ 之间的图（关系）态射是一个对 $R := \langle R_N, R_E\rangle$，使得 $R_N \subseteq N \times N'$ 和 $R_E \subseteq E \times E'$ 是图的节点和边之间的关系，并且 $e_1 R_E e_2$ 蕴含 $s(e_1) R_N s'(e_2)$ 和 $t(e_1) R_N t'(e_2)$。给定图 G、G'，图态射的集合记为 $\mathit{GraphRel}(G, G') \subseteq \mathscr{P}((N \times N') \times (E \times E'))$。

定义 2.3。 一个对应模型 是一个三元组 $\mathfrak{M} := \langle W, D, \mathcal{C}\rangle$，满足

– W 是一个非空元素集合，称为世界，

– D： $W \to \mathit{Graphs}$ 是一个函数，为每个世界指定一个有向图，– $\mathcal{C}$： $W \times W \to \mathscr{P}(\mathit{GraphRel}(D(\omega)\ D(\omega')))$ 是一个函数，为每一对 $\langle \omega, \omega' \rangle$ 指定一个图态射集合 $\mathcal{C}\langle \omega, \omega' \rangle \subseteq \mathit{GraphRel}(D(\omega)\ D(\omega'))$，其中每个 $C \in \mathcal{C}\langle \omega, \omega' \rangle$ 都是两个世界所关联图之间的图态射。我们将其称为原子对应关系。

给定两个世界 ω 和 ω'，集合 $\mathcal{C}\langle \omega, \omega' \rangle$ 是从 ω 到 ω'的原子迁移集合，定义了我们通过系统中的一步迁移 访问世界的可能方式。当集合 $\mathcal{C}\langle \omega, \omega' \rangle$ 为空时，不存在从 ω 到 ω'的原子跃迁。每个原子关系 $C \in \mathcal{C}\langle \omega, \omega' \rangle$ 连接两个世界 ω 和 ω'之间的节点和边，直观地将它们识别为模型经过一次演化后的同一组件。例如，如果我们考虑两个节点 $n \in D(\omega)_N$ 和 $n' \in D(\omega')_N$ 以及一个关系 $C \in \mathcal{C}\langle \omega, \omega' \rangle$，如果$\langle n, n' \rangle \in C_N$，那么 n' 表示节点 n 通过 C的未来发展。

定义 2.4. 节点 $s' \in D(\omega')_N$ 是 $s \in D(\omega)_N$通过对应关系 C 的对应节点，当且仅当 $\langle s, s' \rangle \in C_N$；边也类似。

示例 *2.1*（对应模型）。我们通过指明世界集合 $\{\omega_0, \omega_1, \omega_2\}$ 以及关系集合的基数 $\mathcal{C}\langle \omega, \omega' \rangle$（见图 1）给出一个对应模型的示例；与每个世界相关联的图结构以及连接它们的图态射如图 2所示。存在两个对应关系 C_1、$C_2 \in \mathcal{C}\langle \omega_1, \omega_2 \rangle$ 分别位于 ω_1和 ω_2之间，我们使用蓝色虚线和绿色点线来区分 C_1 和 C_2。 附录A 对该模型及其对应关系进行了明确描述。

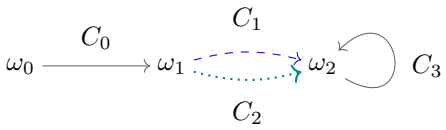

图 1. 模型的世界及其可达关系的图示。

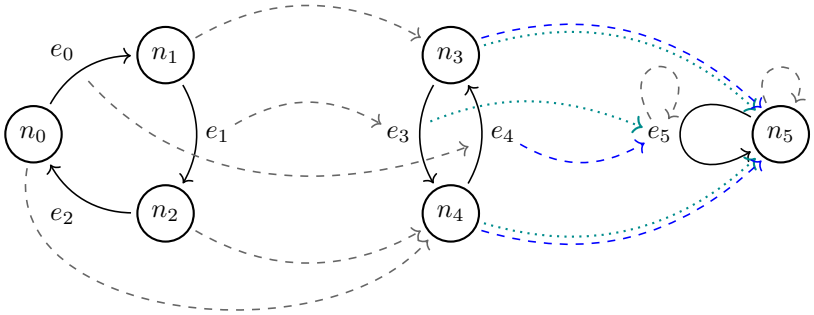

图 2. 模型的图示。

使用关系作为迁移，可以通过在下一状态中不为图的边和节点设置对应项，来模拟它们的移除。例如，如果不存在满足 $\langle e, e'\rangle \in C_E$ 的边 $e' \in D(\omega')_E$，我们便可得出结论：边 e 已被 C 移除。同样，节点的复制可表示为将其与对应关系的两个实例相连接，例如存在元素 n'_1、$n'_2 \in D(\omega')_N$，使得 $\langle n, n'_1\rangle \in C_N$ 和 $\langle n, n'_2\rangle \in C_N$。

简单对应模型例 2.1展示了合并与删除的影响。第一个对应关系 C_0 合并节点 n_0 和 n_2的 ω_0，但这一操作并未生成环：事实上，e_2 被删除了，因为它在 ω_1中没有连接到对应项。类似地，两个对应关系C_1 和 C_2 合并节点 n_3 和 n_4 的 ω_1，但它们从 ω_1 迁移到 ω_2时删除的边不同：分别删除边 e_3或 e_4。请注意，在这两种情况下，节点 n_3 和 n_4 都必须保留（尽管可能会合并），以确保所考虑的关系确实是图的（关系）态射。

2.1 对应关系与轨迹

以下我们假设一个固定的对应模型 $\mathfrak{M} := \langle W, D, \mathcal{C}\rangle$，并正式介绍对应关系背后的思想。我们按照图示顺序表示图态射的复合：例如，给定 $C \in \mathsf{GraphRel}(G_1, G_2)$ 和 $C' \in \mathsf{GraphRel}(G_2, G_3)$，复合图态射记作 $C; C' \in \mathsf{GraphRel}(G_1, G_3)$，并满足 $(C; C')_N = \{(a, c) \mid \exists b \in G_{2N}.\ \langle a, b\rangle \in C_N \wedge \langle b, c\rangle \in C'_N\}$，边的情况也类似。

定义 2.5。图态射 $C \in \mathit{GraphRel}(D(\omega)\ D(\omega'))$ 是模型的对应关系，如果满足以下任一情形

– C 是恒等图态射；– $C \in \mathcal{C}\langle\omega, \omega'\rangle$ 是由模型 $\mathfrak{M}$给定的原子图态射；– C 是由一系列对应关系 $C_0; \cdots; C_n$ 复合而成，其中$C_i \in \mathcal{C}\langle\omega_i, \omega_{i+1}\rangle$。

请注意，两个原子对应关系 $C \in \mathcal{C}\langle\omega_1, \omega_2\rangle$ 和 $C' \in \mathcal{C}\langle\omega_2, \omega_3\rangle$ 的复合 $C; C' \in \mathsf{GraphRel}(D(\omega_1)\ D(\omega_3))$ 可能不是原子的，而模型只定义原子迁移。直接在两张图之间迁移，可能不同于经由一张中间图进行迁移，因为直接迁移不一定是这两个对应关系的复合。此外，前者需要一个演化步，后者则需要两个。

正如 线性时序逻辑中可以识别连接线性演化状态的轨迹，例如参见 [3],我们可以考虑对应关系的线性序列，它提供了一系列可按顺序访问且关联有图的世界。

定义 2.6。一条轨迹 σ 是由原子对应关系构成的无限序列$(C_0, C_1, \ldots)$，满足：$C_i \in \mathcal{C}\langle\omega_i, \omega_{i+1}\rangle$ 对于任意 $i \geq 0$。

换言之，轨迹标识了由对应模型所诱导的图中的一条路径，其中单词是节点，原子对应关系是边。给定一条轨迹 $\sigma = (C_0,\ C_1,\ \dots)$，我们用 $\sigma_i := (C_i,\ C_{i+1},\ \dots)$ 表示排除前 i 个对应关系后得到的轨迹。只要上下文明确，我们用 $\omega_0,\ \omega_1,\ \dots$ 和ω_i 表示该轨迹中的世界 σ 。类似地，我们用 $C_{\leq i}$ 表示从第一个世界 ω_0 到第 i个世界 ω_i ，沿轨迹 σ所给关系构成的复合关系 $C_0; \cdots; C_{i-1}$。若 $i = 0$，则关系 $C_{\leq 0}$ 是 ω_0 上的恒等图态射。

3 量化线性时态逻辑

我们采用标准的集合论表示，介绍我们的一阶量化线性时态逻辑 QLTL 的语法和语义。

3.1 QLTL 的语法和语义

由于自由变量可能出现在公式中，我们回顾一下上下文和上下文中的项的常见表示。为简单起见，我们考虑其项表示有向图的代数，如定义 2.1所述。更确切地说，该签名有两种类型E 和N ，以及两个函数符号 s 和 t，分别显然表示边上的源点函数和目标函数。我们假定有一组带类型变量 $\mathcal{X} = \mathcal{X}_N \uplus \mathcal{X}_E$，并将带类型上下文 Γ 定义为这类变量的有限子集，使用记法 $[\Gamma]\ n :\ \mathsf{N}$ 表示项 n的类型为N ，且它是在带类型上下文 Γ中构造的；也就是说，它的自由变量包含于 Γ 中出现的变量之中（ $[\Gamma]\ e :\ \mathsf{E}$的情形也类似）。

为了更简洁地介绍时态逻辑的语义，通常会排除那些能够用其他算子表示的基本构造。因此，我们用一组最小的标准算子来介绍 QLTL，并通过否定导出合取和全称量化。

定义 3.1（QLTL）。设 Γ 是变量集合 $\mathcal{X}$ 上的一个带类型上下文。*QLTL* 公式的集合 $\mathcal{F}_\Gamma^{QLTL}$ 由以下规则生成

$$\psi := \mathsf{true} \mid e_1 =_E e_2 \mid n_1 =_N n_2$$

$$\phi := \psi \mid \neg\phi \mid \phi \vee \phi \mid \exists_N x.\phi \mid \exists_E x.\phi \mid \mathsf{O}\phi \mid \phi \mathsf{U} \phi \mid \phi \mathsf{W} \phi.$$

其中，对于 $i = 1,\ [\Gamma]\ e_i :\ E$ 且 $[\Gamma]\ n_i :\ N,\ 2$。

上述定义实际上给出了上下文中的公式：我们用记号 $[\Gamma]\phi$ 表示公式 ϕ 属于带类型上下文 Γ 下的 $\mathcal{F}_\Gamma^{\mathsf{QLTL}}$ 。显然，说 $[\Gamma]\phi$ 等同于断言 $\mathsf{fv}(\phi) \subseteq \Gamma$，也就是说， ϕ 的自由变量包含在 Γ 中出现的变量之中。对于项和公式，只要上下文无需明确说明，我们就省略方括号括起的上下文。

字母 ψ 表示由两种类型的等价谓词构成的原子公式集合。给定两个边项 $e_1, e_2 : \mathsf{E}$，公式 $e_1 =_E e_2$ 表示这两条边在与当前世界相关联的图中重合；对于两个节点项 $n_1, n_2 : \mathsf{N}$ 和公式 $n_1 =_N n_2$，情况也类似。

存在量词算子 $\exists_N x.\phi$ 和 $\exists_E x.\phi$ 可用于表达当前图中存在满足某种性质 ϕ 的节点（分别为边），其中变量 x 允许作为 ϕ的自由变量出现。

下一个 算子 Oϕ 表示某个性质 ϕ 必须在下一个状态为真。直到 算子 ϕ_1 Uϕ_2 表示性质ϕ_1 必须至少保持到性质 ϕ_2 变为真为止，而后者必须在当前或未来时刻为真。最后，弱直到 算子 ϕ_1Wϕ_2 与 ϕ_1Uϕ_2 算子类似，但允许对应物无限期存在，而永远不会到达 ϕ_2 成立的时刻，前提是 ϕ_1 也无限期保持成立。

对偶算子的语法表达式为 假：$:= \neg$真，$\phi_1 \wedge \phi_2 :=\neg(\neg\phi_1 \vee \neg\phi_2)$，以及 $\forall_N x.\phi :=\neg\exists_N x.\neg\phi$，边的情况也类似。需要注意的是，与经典的线性时序逻辑不同，直到算子和弱直到算子并非自对偶：这一事实将在注 3.3中进行讨论并明确说明。

3.2 可满足性

为介绍公式相对于对应模型的可满足性概念，我们先定义世界中上下文的赋值 。

定义 3.2（赋值）。世界 $\omega \in W$ 中带类型上下文 Γ的赋值 是由两个函数 $\mu:=\langle\mu_N,\mu_E\rangle$ 构成的有序对，满足 $\mu_N\colon \Gamma_N \to D(\omega)_N$ 且 $\mu_E : \Gamma_E \to D(\omega)_E$。我们用记号 $\mathcal{A}^{\Gamma}_{\omega}$ 表示带类型上下文 Γ在 ω 中定义的赋值集合 μ 。

此外，我们用 $\mu[x \mapsto_\tau n] \in \mathcal{A}^{\Gamma,\,(x:\tau)}_{\omega}$表示通过在 μ 的定义域中加入 $n \in D(\omega)_\tau$ 并将其赋给变量 $x \notin \Gamma$而得到的赋值；上下文明确时，省略类型 $\tau \in \{\mathrm{N}, \mathrm{E}\}$。我们用 $\Gamma, (x\colon \tau)$表示在上下文 Γ 中增加一个类型为 τ的变量 x 后得到的上下文。

赋值 μ 针对带类型上下文 Γ 为上下文属于（包含于） Γ的上下文中的项提供解释：它允许对项的自由变量求值，从而定义其语义（相对于该赋值）。这将在定义 3.5中分别为等式$n_1 =_N n_2$ 和 $e_1 =_E e_2$ 提供含义。

定义 *3.3*（项上的赋值）。给定赋值 $\mu \in \mathcal{A}^{\Gamma}_{\omega}$，我们用 $\mu^*:=\langle\mu^*_N,\mu^*_E\rangle$ 表示 μ 在上下文中的项 $[\Gamma]\, n\colon N$ 或 $[\Gamma]\, e\colon E$ 上的解释，其归纳定义规则如下：$\mu^*_E(x) := \mu_E(x)$，$\mu^*_N(y) := \mu_N(y)$，$\mu^*_N(s(e)) := D(\omega)_s(\mu^*_E(e))$，以及 $\mu^*_N(t(e)) := D(\omega)_t(\mu^*_E(e))$。

现在我们来看看如何通过对应关系传递赋值。直观来说，当世界之间存在对应关系，且底层图中的元素逐点相关时，我们需要连接两个赋值所映射到的节点和边。

定义 3.4（赋值上的对应关系）。给定一个对应关系 $C \in \mathit{GraphRel}(D(\omega_1)\ D(\omega_2))$ 以及上下文 Γ上的两个赋值 $\mu_1 \in \mathcal{A}^{\Gamma}_{\omega_1}$ 和 $\mu_2 \in \mathcal{A}^{\Gamma}_{\omega_2}$，如果对于任意 $x \in \Gamma_N$和 $x \in \Gamma_E$，均有 $\langle\mu_{1N}(x)\ \mu_{2N}(x)\rangle \in C_N$和 $\langle\mu_{1E}(x)\ \mu_{2E}(x)\rangle \in C_E$，则称赋值 μ_1和 μ_2是对应相关的。我们用符号 $\langle\mu_1,\mu_2\rangle \in C$表示这一关系。

现在，我们可以引入关于轨迹 σ 和赋值 μ的 QLTL 公式的可满足性这一概念。

定义 3.5（QLTL 可满足性）。给定一个上下文中的 *QLTL* 公式 $[\Gamma]\phi$、一条轨迹 $\sigma = (C_0, C_1, \dots)$，以及 σ 的第一个世界中的一个赋值 $\mu \in \mathcal{A}^{\Gamma}_{\omega_0}$，我们按归纳方式定义如下的可满足关系

– $\sigma, \mu \vDash$ 为真；– $\sigma, \mu \vDash e_1 =_E e_2$ 当 $\mu^*_E(e_1)$ 时，$= \mu^*_E(e_2)$；– $\sigma, \mu \vDash n_1 =_N n_2$ 当 $\mu^*_N(n_1)$ 时，$= \mu^*_N(n_2)$；– $\sigma, \mu \vDash \neg\phi$ 当 $\sigma, \mu \nvDash \phi$；– $\sigma, \mu \vDash \phi_1 \vee \phi_2$ 当 $\sigma, \mu \vDash \phi_1$ 或 $\sigma, \mu \vDash \phi_2$；– $\sigma, \mu \vDash \exists_N x.\phi$ 当存在一个节点 $n \in D(\omega_0)_N$，使得 $\sigma, \mu[x \mapsto n] \vDash \phi$；– $\sigma, \mu \vDash \exists_E x.\phi$ 当存在一条边 $e \in D(\omega_0)_E$，使得 $\sigma, \mu[x \mapsto e] \vDash \phi$；– $\sigma, \mu \vDash \mathsf{O}\phi$ 当存在 $\mu_1 \in \mathcal{A}^{\Gamma}_{\omega_1}$，使得 $\langle \mu, \mu_1 \rangle \in C_0$ 且 $\sigma_1, \mu_1 \vDash \phi$；– $\sigma, \mu \vDash \phi_1 \mathsf{U} \phi_2$ 当存在一个 $\bar{n} \geq 0$，使得 *1.* 对于任意 $i < \bar{n}$，存在 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，使得 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 且 $\sigma_i, \mu_i \vDash \phi_1$；*2.* 存在 $\mu_{\bar{n}} \in \mathcal{A}^{\Gamma}_{\omega_{\bar{n}}}$，使得 $\langle \mu, \mu_{\bar{n}} \rangle \in C_{\leq \bar{n}}$ 且 $\sigma_{\bar{n}}, \mu_{\bar{n}} \vDash \phi_2$；– $\sigma, \mu \vDash \phi_1 W \phi_2$ 当以下条件之一成立

- 对于 $\phi_1 \mathsf{U} \phi_2$，同样的条件也适用；或者

对于任意 i，都存在 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，使得 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 且 σ_i，$\mu_i \vDash \phi_1$。

3.3 示例

我们提供一些运行示例中 QLTL公式可满足性的示例，以 图 2 为例，说明我们的对应体语义在实践中如何运作。例如，考虑由以下元素给出的轨迹 $\sigma := (C_0, C_1, C_3, C_3, \dots)$，即考虑从 ω_1迁移时只有 e_4 这条边得到保留的情况。

示例 *3.1*（分配与释放）。正如第 2 节所预期的那样，对应体语义的主要优势之一在于，它能够对系统及其演化过程中元素的存在、释放、复制和合并进行推理。例如，考虑以下简写公式

$$\begin{aligned} \mathbf{present}_\tau(x) &:= \exists_\tau y.x =_\tau y, \\ \mathbf{nextPreserved}_\tau(x) &:= \mathbf{present}_\tau(x) \wedge \mathsf{O}\mathbf{present}_\tau(x), \\ \mathbf{nextDealloc}_\tau(x) &:= \mathbf{present}_\tau(x) \wedge \neg\mathsf{O}\mathbf{present}_\tau(x). \end{aligned}$$

公式 **当前存在**$_\tau(x)$ 表示当前时刻存在一个节点或一条边。我们可以将此谓词与 下一步 算子结合起来，讨论当前世界中存在且在下一步仍然存在的元素，并用公式 **下一步保留**$_\tau(x)$ 简洁地表示。类似地，考虑公式 **下一步释放**$_\tau(x)$，便可指代当前存在、但将在下一步被释放的元素。事实上，我们有

$\sigma_0, \{x : \mathbf{E} \mapsto e_0\} \vDash \mathbf{nextPreserved}_E(x)$; $\sigma_1, \{x : \mathbf{N} \mapsto n_3\} \nvDash \mathbf{nextDealloc}_N(x)\sigma_0, \{x : \mathbf{N} \mapsto n_1\} \vDash \mathbf{nextPreserved}_N(x)$; $\sigma_1, \{x : \mathbf{E} \mapsto e_3\} \vDash \mathbf{nextDealloc}_E(x)\sigma_0, \{x : \mathbf{E} \mapsto e_2\} \nvDash \mathbf{nextPreserved}_E(x)$; $\sigma_1, \{x : \mathbf{E} \mapsto e_4\} \nvDash \mathbf{nextDealloc}_E(x)$

示例 *3.2*（图结构）。此外，我们的语法允许我们定义能够利用图的代数结构的公式，并将它们与时序算子结合起来，以描述图如何随时间演变。我们通过给出以下公式来说明这一点

$$\text{环}(e) := s(e) =_N t(e), \text{有环}(n) := \exists_E e.s(e) =_N n \wedge \text{环}(e), \text{可复合}(x,y) := t(x) =_N s(y) \text{具有复合}(x,y) := \text{可复合}(x,y) \wedge \exists_E e.(s(e) =_N s(x) \wedge t(e) =_N t(y)) \text{相邻}(x,y) := \exists_E e.((s(e) =_N x \wedge t(e) =_N y) \vee (t(e) =_N x \wedge s(e) =_N y))$$

它们分别描述以下情形：我们可以使用 **环**(x) 检查图中给定的边是否为环，或者使用 **有环**(x) 验证唯一具有环的节点是 n_5；或者，我们也可以通过使用 **有环** 来表达这一事实，即该环属于整个世界

$$\sigma_0, \{x \mapsto e_0\} \nvDash \textbf{环}(x); \ \sigma_0, \{x \mapsto n_0\} \nvDash \textbf{有环}(x); \ \sigma_0, \{\} \nvDash \exists_N x.\textbf{有环}(x) \sigma_1, \{x \mapsto e_3\} \nvDash \textbf{环}(x); \ \sigma_1, \{x \mapsto n_3\} \nvDash \textbf{有环}(x); \ \sigma_1, \{\} \nvDash \exists_N x.\textbf{有环}(x) \sigma_2, \{x \mapsto e_5\} \vDash \textbf{环}(x); \ \sigma_2, \{x \mapsto n_5\} \vDash \textbf{有环}(x); \ \sigma_2, \{\} \vDash \exists_N x.\textbf{有环}(x)$$

由于 $\exists_N x.\textbf{有环}(x)$ 是一个闭公式，因此该公式唯一可取的赋值是空赋值。由此，我们得到公式的经典概念：它只提供真或假的二元信息，无需选择任何个体来满足它。最后，我们可以表达图中有关中间节点存在性以及边的可组合性的某些性质

$$\sigma_0, \{x \mapsto n_0, y \mapsto n_1\} \vDash \textbf{相邻}(x,y); \sigma_0, \{x \mapsto e_0, y \mapsto e_1\} \vDash \textbf{可复合}(x,y); \sigma_1, \{x \mapsto n_3, y \mapsto n_4\} \vDash \textbf{相邻}(x,y); \sigma_0, \{x \mapsto e_0, y \mapsto e_1\} \vDash \textbf{O可复合}(x,y); \sigma_1, \{x \mapsto n_3, y \mapsto n_4\} \vDash \textbf{O相邻}(x,y); \ \sigma_1, \{x \mapsto e_3, y \mapsto e_4\} \nvDash \textbf{O可复合}(x,y);$$

$$\begin{aligned} &\sigma_0, \ \{x \mapsto e_0, y \mapsto e_1\} \nvDash \textbf{具有复合}(x, \ y); \sigma_0, \\ &\{x \mapsto e_0, y \mapsto e_1\} \nvDash \mathsf{O}\textbf{具有复合}(x, \ y); \sigma_2, \\ &\{x \mapsto e_5, y \mapsto e_5\} \vDash \mathsf{O}\textbf{具有复合}(x, \ y); \end{aligned}$$

注释 *3.1*（最终和始终算子）。与线性时序逻辑中一样，我们可以将附加的最终 $\Diamond\phi$ 和始终 $\Box\phi$ 算子定义为 $\Diamond\phi :=$trueUϕ 和 $\Box\phi :=\phi$Wfalse。或者，也可以直接将它们的语义表示为

– $\sigma, \ \mu \vDash \Diamond\phi$ 若存在 $i \geq 0$ 和 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，使得 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 且 $\sigma_i, \ \mu_i \vDash \phi_\circ$ – $\sigma, \ \mu \vDash \Box\phi$ 若对任意 $i \geq 0$，存在 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，使得 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 且 $\sigma_i, \ \mu_i \vDash \phi_\circ$

例 *3.3*（时间演化）。我们可以使用这些算子来表示图经过任意多个步骤后的演化

$$\begin{aligned}\mathbf{willMerge}_\tau(x,y) &:= x \neq_\tau y \wedge \Diamond(x =_\tau y),\\ \mathbf{alwaysPreserved}_\tau(x) &:= \Box\mathbf{present}_\tau(x).\\ \mathbf{willBecomeLoop}(e) &:= \neg\mathbf{loop}(e) \wedge \Diamond\mathbf{loop}(e)\end{aligned}$$

<escape>在 图1 中，对于相同的轨迹 $\sigma = (C_0,\ C_1,\ C_3,\ \ldots)$ 我们有</escape>

$\sigma_0, \{\} \vDash \exists n.\exists m.$**将合并**$_N(n, m)$　　$\sigma_0, \{\} \nvDash \forall e.\Diamond$**环(**$e$**),**$\sigma_0, \{\} \nvDash \forall e.$**始终保留**$_E(e)$

$\sigma_0, \{\} \vDash \exists e.\Diamond\Box$**环(**$e$**),**$\sigma_0, \{\} \vDash \exists e.$**将变为环**$e($　　$\sigma_0), \{\} \vDash \exists x.\exists y.\neg\Diamond$**可复合(**$x, y$**),**$\sigma_0, \{\} \vDash$**(**$\exists e.$**s(**$e$**)** $\neq$ **t(**e**))U(**$\exists x.\forall y.x = y$**)** $\sigma_0, \{\} \vDash$**(**$\exists e.$**s(**e**)** $=$ **t(**e**))W¬(**$\exists x.$**环(**e**)).**

注*3.2*（未绑定变量的量词省略）。与标准量词逻辑的一个重要区别是，在 QLTL 中，如果引入的变量没有出现在子公式中，我们就不能省略相应的量化。假设 $\equiv$表示语义等价，并取任意满足 $x \notin$fv(ϕ) 的 ϕ，一般而言有 $\exists x.\phi \not\equiv \phi$，类似地也有 $\forall x.\phi \not\equiv \phi$。更准确地说，当 ϕ 不含任何时态算子，且所考虑的当前世界 $D(\omega)$ 非空时，上述等价关系成立。

考虑一个世界 ω，其中有一个节点 $D(\omega)_N = \{s\}$、没有边，并且只有一个自环对应关系 $\mathcal{C}\langle\omega,\omega\rangle = \{C\}$，其中 $C = \emptyset$ 为空对应关系。轨迹为 $\sigma = (C,\ C,\ \ldots\ldots)$。取空赋值 $\{\}$ 和闭公式 $\phi =$ O(true)，容易验证 $\sigma,\ \{\} \vDash$ O(true)，但 $\sigma,\ \{\} \nvDash \exists_N x.$O(true)。原因在于，一旦赋值扩展了某个元素，从一个世界转到下一个世界时，就要求赋值中的每个个体都得到保留，并且在下一个世界中有对应物。或者，我们也可以在语义中限制赋值，使得只有公式中出现的自由变量才需要有对应物。例如，下一个Oϕ 算子的定义将变为

– $\sigma,\ \mu \vDash$ Oϕ 当存在 $\mu_1 \in A^{\mathrm{fv}(\phi)}_{\omega_1}$ 使得 $\langle\mu_{|\mathrm{fv}(\phi)}, \mu_1\rangle \in C_0$ 且 $\sigma_1,\ \mu_1 \vDash \phi$为便于本文阐述并结合我们的 Agda 实现，我们考虑上下文中的所有元素都有对应项的情况。

注释 3.3（直到与弱直到不兼容）。在标准 LTL 中，直到 ϕ_1Uϕ_2 和弱直到 ϕ_1 Wϕ_2 算子具有相同的表达能力，并且可以通过等价关系 ϕ_1U$\phi_2 \equiv_{\mathsf{LTL}} \neg(\neg\phi_2$ W$(\neg\phi_1 \wedge \neg\phi_2))$ ϕ_1W$\phi_2 \equiv_{\mathsf{LTL}} \neg(\neg\phi_2U(\neg\phi_1 \wedge \neg\phi_2))$ 互相定义。然而，在 QLTL 中情况并非如此。类似地，乍看之下，似乎可以在 QLTL 中通过 $\Box\phi := \neg\Diamond\neg\phi$ 定义标准的全称算子。然而，该定义与注释 3.1 中给出的语义并不一致。QLTL 的这一特性同样源于我们所处的关系（可能会解除分配）环境；当我们在第 4 节引入 PNF 语义时，将对此进行形式化解释并给出直观说明。通过将模型限制为对应关系是全函数的模型，可以再次获得 LTL 等价关系：这使我们能够考察一条始终有定义的对应物唯一轨迹，从而使我们的模型回到类似标准 LTL 的轨迹概念。

4 QLTL 的正范式

正范式是时序逻辑的一种标准表示形式：它们可用于简化理论和实现方面的构造与算法 [5,26]。它们对于基于不动点的语义至关重要，例如 [16], ，同时仍能保留原始表示形式的表达能力。正如我们在 第 5 节 中指出的，为我们的逻辑提供无否定语义，也确保它能够更容易地在证明助手中进行处理，而其中的定义和证明都是 构造性的。此外，正范式转换还充当一个具体过程，可交互地用于自动将公式转换为其正范式；该过程的正确性已在证明助手中得到证明。本节将介绍 QLTL 的正范式的显式语义，我们将其记为 PNF。

4.1 正范式的语义

正如注 3.3中所观察到的，为了给出正范式，我们需要添加其他算子，以充分表达时态算子的否定。因此，我们引入一种新的下一时刻算子，称为全称下一时刻 Aϕ。类似地，我们还需要为直到 $\phi_1 \mathsf{U} \phi_2$ 和弱直到 $\phi_1 \mathsf{W} \phi_2$引入对偶算子，分别称为然后 $\phi_1 \mathsf{T} \phi_2$ 和全称直到 $\phi_1 \mathsf{F} \phi_2$ 算子。

定义 4.1（QLTL 的正范式）。设 Γ 是变量集 $\mathcal{X}$ 上的带类型上下文。*QLTL* 正范式中的公式集合 $\mathcal{F}_{\Gamma}^{PNF}$ 由以下规则生成

$$\psi := \mathsf{true} \mid e_1 =_E e_2 \mid n_1 =_N n_2$$

$$\phi := \psi \mid \neg\psi \mid \phi \vee \phi \mid \phi \wedge \phi \mid \exists_\tau x.\phi \mid \forall_\tau x.\phi \mid \mathsf{O}\phi \mid \mathsf{A}\phi \mid \phi \mathsf{U} \phi \mid \phi \mathsf{F} \phi \mid \phi \mathsf{W} \phi \mid \phi \mathsf{T} \phi.$$

其中 $[\Gamma]\ e_i :\ E$，$[\Gamma]\ n_i :\ N$，$i = 1$、2 以及 $\tau \in \{N、E\}$。

全称下一步 Aϕ 算子的直观含义是，它使我们能够描述这样一种情况：某个个体在下一步不存在对应物；如果存在任何对应物，则要求其满足公式 ϕ。

与直到 $\phi_1 \mathsf{U} \phi_2$ 算子类似，全称直到 $\phi_1 \mathsf{F} \phi_2$ 算子允许我们取一系列图，在其中 ϕ_1 在若干步骤中成立，直至 ϕ_2 成立。关键在于，每个 满足 ϕ_1 的中间对应物以及最终对应物都必须满足 ϕ_2。这些对应物不要求存在；事实上，由全为空对应关系组成的任何轨迹始终同时满足 $\phi_1 \mathsf{F} \phi_2$ 和 $\phi_1 \mathsf{T} \phi_2$。与弱直到 $\phi_1 \mathsf{W} \phi_2$ 算子类似，然后 $\phi_1 \mathsf{T} \phi_2$ 算子对应于弱全称直到，并且可以通过这样一条轨迹来验证：其中所有对应物都满足 ϕ_1，但始终不满足 ϕ_2。

现在，我们通过仅规定附加算子的语义（省略未发生变化的算子），来给出正范式公式的可满足关系。

定义 4.2（QLTL 正范式可满足性）。给定一个处于正范式的 *QLTL* 公式上下文 $[\Gamma]\phi$、一条轨迹 $\sigma = (C_0, C_1, \dots)$，以及在 σ的第一个世界中的一个赋值 $\mu \in \mathcal{A}^{\Gamma}_{\omega_0}$，我们针对附加算子归纳地定义可满足关系如下

– $\sigma, \mu \vDash \neg\psi$ 若 $\sigma, \mu \nvDash \psi$；– $\sigma, \mu \vDash \phi_1 \wedge \phi_2$ 若 $\sigma, \mu \vDash \phi_1$ 且 $\sigma, \mu \vDash \phi_2$；– $\sigma, \mu \vDash \forall_\tau x.\phi$ 若对于任意 $s \in D(\omega_0)_\tau$，我们有 $\sigma, \mu[x \mapsto s] \vDash \phi$；– $\sigma, \mu \vDash A\phi$ 若对于任意满足 $\langle \mu, \mu_1 \rangle \in C_0$ 的 $\mu_1 \in \mathcal{A}^{\Gamma}_{\omega_1}$，我们有 $\sigma_1, \mu_1 \vDash \phi$；– $\sigma, \mu \vDash \phi_1 F \phi_2$ 若存在一个 $\bar{n} \geq 0$，满足：*1.* 对于任意满足 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 的 $i < \bar{n}$ 和 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，我们有 $\sigma_i, \mu_i \vDash \phi_1$；*2.* 对于任意满足 $\langle \mu, \mu_{\bar{n}} \rangle \in C_{\leq \bar{n}}$ 的 $\mu_{\bar{n}} \in \mathcal{A}^{\Gamma}_{\omega_{\bar{n}}}$，我们有 $\sigma_{\bar{n}}, \mu_{\bar{n}} \vDash \phi_2$；– $\sigma, \mu \vDash \phi_1 T \phi_2$，若以下条件之一成立

- 对于 $\phi_1 \mathsf{F} \phi_2$，同样的条件也适用；或者
  对于任意满足 $\langle \mu, \mu_i \rangle \in C_{\leq i}$ 的 i 和 $\mu_i \in \mathcal{A}^{\Gamma}_{\omega_i}$，我们有 $\sigma_i, \mu_i \vDash \phi_1$。

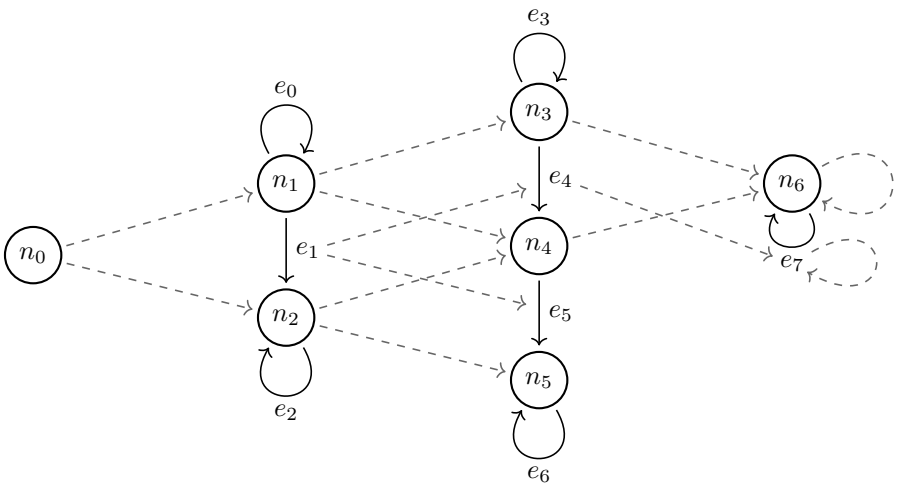

图 3. 对应模型的图示。

例 4.1。我们通过图 3 中的例子说明，对应关系可以复制图中的边和节点，同时也为定义 4.2 中定义的新算子提供了一些具体实例。例如，$\sigma_0, \{x \mapsto n_0\} \vDash$ A(hasLoop(x))成立，但 $\sigma_1, \{x \mapsto n_1\} \nvDash$ A(hasLoop(x))不成立，因为 n_4 是 n_1 的对应节点，却没有环。此外，$\sigma_1, \{x \mapsto e_1\} \vDash$ hasLoop(s(x))U(loop(x))成立，但 $\sigma_1, \{x \mapsto e_1\} \nvDash$hasLoop($s$($x$))F(loop($x$))不成立，因为我们还要求 e_5 的源点处有环，这是因为它是 e_1的对应节点。注意，$\sigma_2, \{x \mapsto e_5\} \vDash$ A(loop(x))成立，因为下一步骤没有对应节点；事实上，我们同样有 σ_2, $\{x \mapsto n_5\} \vDash$ hasLoop(x)F(false)。最后，$\sigma_2, \{x \mapsto e_4\} \vDash$hasLoop($s$($x$))T(¬ loop($x$))成立，因为中间条件始终成立。

4.2 QLTL 的否定与正范式

验证第 4 节所述正范式（PNF）的关键观察是，第 4 节中 *next* $\mathsf{O}\phi$、*until* $\phi_1 \mathsf{U}\phi_2$以及*weak until* $\phi_1\mathsf{W}\phi_2$ 公式的否定现在都可以在该逻辑中表达。我们用 $\vDash_{\mathsf{QLTL}}$ 和 $\vDash_{\mathsf{PNF}}$ 分别表示标准QLTL和正范式下的QLTL中公式的可满足关系。

命题 4.1（否定可用正范式表示） *(Relational.Negation)* 设 ψ，ψ_1，ψ_2 为正范式中的原子公式。那么有

$$\begin{aligned}
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \vDash_{QLTL} \neg \mathsf{O}(\psi) && \iff \sigma, \mu \vDash_{PNF} \mathsf{A}(\neg\psi)\\
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \vDash_{QLTL} \neg(\psi_1 \mathsf{U} \psi_2) && \iff \sigma, \mu \vDash_{PNF} (\neg\psi_2)\mathsf{T}(\neg\psi_1 \wedge \neg\psi_2)\\
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \vDash_{QLTL} \neg(\psi_1 \mathsf{W} \psi_2) && \iff \sigma, \mu \vDash_{PNF} (\neg\psi_2)\mathsf{F}(\neg\psi_1 \wedge \neg\psi_2).
\end{aligned}$$

类似地，在正范式中表示这些新算子的否定的逆命题并不成立：唯一的例外是简单的全称下一步$\mathsf{A}\phi$ 算子，其否定直接对应于下一步$\mathsf{O}\phi$ 算子。

命题 4.2（新算子的否定不在正范式中） 设 ψ，ψ_1，ψ_2为正范式中的原子公式。那么有

$$\begin{aligned}
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \nvDash_{PNF} \mathsf{A}(\psi) && \iff \sigma, \mu \vDash_{PNF} \mathsf{O}(\neg\psi)\\
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \nvDash_{PNF} \psi_1 \mathsf{T} \psi_2 && \not\Longleftarrow\ \sigma, \mu \vDash_{PNF} (\neg\psi_2)\mathsf{U}(\neg\psi_1 \wedge \neg\psi_2)\\
&\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \nvDash_{PNF} \psi_1 \mathsf{F} \psi_2 && \not\Longleftarrow\ \sigma, \mu \vDash_{PNF} (\neg\psi_2)\mathsf{W}(\neg\psi_1 \wedge \neg\psi_2).
\end{aligned}$$

证明。参见附录 B.1。

现在回到命题 4.1。我们利用其中给出的算子之间的对应关系，来定义一个转换 $\cdot : \mathcal{F}^{\mathsf{QLTL}} \to \mathcal{F}^{\mathsf{PNF}}$ 从QLTL 语法（见定义 3.5）转换为当前的正范式。值得注意的是，这种转换保留了公式的等价性。

定理 4.3（正范式等价性）。*(Relational.Conversion)* 设 $\cdot : \mathcal{F}^{QLTL} \to \mathcal{F}^{PNF}$ 为一种语法转换：通过将否定下推至原子公式，把带否定的时态算子替换为其在正范式中的等价算子。对于任意*QLTL* 公式 $[\Gamma]\phi \in \mathcal{F}^{QLTL}$，有

$$\forall \sigma, \mu \in \mathcal{A}^{\Gamma}_{\omega_0}.\ \sigma, \mu \vDash_{QLTL} \phi \iff \sigma, \mu \vDash_{PNF} \overline{\phi}.$$

与线性时序逻辑中的情况不同，对于对应关系，通常的展开律（即用算子自身定义该算子的展开律）在 QLTL 中并不成立，以下结果说明了这一点。

命题 4.4（展开律在 QLTL 中不成立） 我们在正范式下有以下不等式

$$\begin{aligned}
&\phi_1 \mathsf{U} \phi_2 \not\equiv \phi_2 \vee (\phi_1 \wedge \mathsf{O}(\phi_1 \mathsf{U} \phi_2)) && \phi_1 \mathsf{F} \phi_2 \not\equiv \phi_2 \vee (\phi_1 \wedge \mathsf{A}(\phi_1 \mathsf{F} \phi_2))\\
&\phi_1 \mathsf{W} \phi_2 \not\equiv \phi_2 \vee (\phi_1 \wedge \mathsf{O}(\phi_1 \mathsf{W} \phi_2)) && \phi_1 \mathsf{T} \phi_2 \not\equiv \phi_2 \vee (\phi_1 \wedge \mathsf{A}(\phi_1 \mathsf{T} \phi_2)).
\end{aligned}$$

证明。 见附录 B.2。

备注 *4.1*（函数式对应关系）。在每个对应关系都是偏函数的情况下，可以根据 [17,16]中给出的对应模型定义，对先前的结果重新表述。事实证明，在偏函数的假设下，我们可以得到命题 4.2（Functional.Negation）中陈述的所有等价关系，以及命题 4.4（Functional.ExpansionLaws）中的展开律。后者可用于将时序算子表示为最小不动点（$\phi_1 \mathsf{U} \phi_2$, $\phi_1 \mathsf{F} \phi_2$）以及最大不动点（$\phi_1 \mathsf{W} \phi_2$, $\phi_1 \mathsf{T} \phi_2$，其中的算子经过适当定义；这种基于不动点的定义，只有在偏函数（或全函数）的情况下，才与定义 4.2 中给出的算子语义一致。

5 Agda 形式化

本节概述本文的另一项贡献：使用依赖类型编程语言和证明助手 Agda，对 QLTL的语义及其 PNF 进行完整形式化 [36]。我们在附录 C 中简要介绍这项工作的开发成果及其用法，并展示如何在 Agda 中具体建模图 2 所示运行示例的时间演化。逻辑的完整形式化以及 PNF 相关结果可在以下网址获取：
https://github.com/iwilare/algebraic-temporal-logics。

5.1 形式化方面

我们的形式化工作包括对论文中介绍的所有方面进行机械化：首先定义对应关系与迹以及关系态射的迹的概念，并将其作为逻辑的模型；随后为 QLTL 和 PNF 公式的语法（要求类型正确且作用域合法）提供表示，并给出相应的可满足性语义。接着，我们提供从 QLTL到 PNF的转换函数，并证明该过程的正确性与完备性；最后，利用所定义的框架，我们证明了若干等价关系，其中包括第 4.2 节 为函数式情形引入的相关展开律。

为便于本文阐述，我们将讨论范围限定在图签名上，而附录 C中给出的建模示例确实是基于有向图签名实例化的。不过，我们要指出，我们的实现具有足够的通用性，可以对任意通用多类型签名上的代数进行建模。具体而言，这意味着通过指定一个定义恰当的签名，逻辑所考虑的模型（及公式）类可以扩展到任何支持在多类型签名上进行代数表示的图形式系统，例如多重图、无向图，以及 [22]中的带类型属性图。

此外，鉴于该形式化的构造性解释，证明 PNF 相对于 QLTL 的正确性和完备性，也具体体现为一个能够将公式转换为其正范式版本的过程，同时还能提供转换正确性的证明。下面我们将介绍用户如何使用该形式化提供的主要组件，与证明助手进行交互。

签名定义。 利用形式化中给出的定义，用户可以编写自己的代数签名，用于将所关注的系统表示为该签名上的代数。例如，通过定义图的签名 Gr，用户可以对图的时间演化进行推理，并使用（关系型）图同态作为世界之间的对应关系。

公式构建。 在提供感兴趣的签名后，用户可以充分利用 QLTL 的全部表达能力来构建公式，并根据签名构造项，基于项的相等性进行推理。这使用户能够表达结合逻辑量词并利用系统特定结构的性质，还可以组合和重用先前定义的公式。形式化所提供的基础设施能够确保用户构建的公式在本质上经过检查，相对于签名中的类型而言，具有正确的作用域和类型，例如，在图的情况下，即边和节点。用户可以在公式中自由使用否定，也可以（选择性地）使用我们定义的过程，将公式自动转换为其 正范式，正如我们在 第 4 节 中所看到的，相较于标准时序逻辑，这一点在对应体语境下可能尤其违反直觉。

模型定义。 用户可以按照之前提供的签名，构造系统在各个时间点的模型。然后，用户指定一系列符号世界，并通过定义世界上的关系，指出可能发生的迁移。接着，必须为每个世界指定一个该签名的代数，而世界之间的连接则被转换为用户提供的代数之间的态射。Agda 会检查模型的迁移是否保留了所考虑世界的代数结构，从而对应于图态射的概念；这一步相对直接，因为 Agda 提供的自动化功能有助于证明映射保留结构。世界之间的轨迹通过使用带大小类型的归纳定义来定义 [11], ，从而允许用户对无限（重复）轨迹进行建模和定义。

模型中公式的验证。使用该库，用户可以证明特定模型满足给定公式；我们的形式化会自动简化验证该公式所需证明的目标，并由证明助手通过自动构造证明项的骨架来引导用户。

5.2 直觉主义证明助手

在我们的设定中，有几个关键的易用性问题需要说明。Agda 是一种证明助手，其基础是对数学的直觉主义解释 [23]。这意味着，在时态逻辑中常用的一些有用逻辑原理在该系统中无法证明，例如排中律，或在子公式中出现否定时用于转换联结词与量词的德摩根律。因此，若不假设这些逻辑原理，我们的时序逻辑 QLTL 的嵌入实际上会受限于直觉主义片段；不过在实践中，这并非特别棘手，因为只需将经典推理作为公理加以假设，便可得到上述等价关系

。然而，从用户的角度来看，这并不可取，因为他们不得不在证明中显式使用这些经典公理。为了应对这些易用性问题以及直觉主义设定下否定的处理，我们采用以下方法：使用 Agda 将逻辑公式表示为完整的正范式，与第 4 节中介绍的形式类似；我们已在 Agda 中通过公设经典原理证明了其正确性，见定理 4.3。这样一来，处理否定所需的经典推理负担便从用户转移到了实现者，同时也让用户能够充分使用扩展后的正确量词集合。此外，转换为正范式的正确性证明既从理论上保证了逻辑表示的表达能力不会增加或减少，也提供了一种具体算法，用户可用它将公式转换为等价的正范式公式，从而更容易证明其有效性。

5.3 自动化

将时序逻辑嵌入证明助手中，可以让用户利用工具的助手 特性，例如帮助用户证明（甚至自动证明）模型中的某个公式是否满足。

在 Agda 中，自动化能力有限，尤其是与那些将自动化和策略的使用作为软件环境核心功能的证明助手相比，例如 Coq [6],Lean [33], 和 Isabelle [35]。Agda 的综合器 Agsy [31] 是 Agda 中实现某种自动化证明搜索的主要辅助工具。遗憾的是，Agsy 仅提供通用搜索过程，其定理证明能力远不及专门的模型检测算法。不过，Agda 所提供的目标导向交互性，是逐步骤证明定理并在我们的设置中手动验证公式的宝贵工具；而构造器的辅助引入功能，则让用户能够快速生成验证时序公式所需的证明结构。

6 相关工作

截至21世纪10年代初，已有一系列论文致力于研究用于表达图及图演化性质的若干量词逻辑变体。我们的模型受到 [16],中围绕带不动点的 μ演算所研究的基于对应关系的逻辑的启发；关于当时已有方案的概述及比较（例如广为人知的 [1],，它们都倾向于采用基于全域的方法），请参阅该文。上述工作后续的研究包括 [18] 以及相关的 [43,46],，后者进一步探索了图研究领域开发的一种重要工具GROOVE。在某种程度上，本文及其姊妹论文 [17]，（介绍二阶QLTL的范畴语义）总结了此前围绕对应模型（包括其实现）展开的一系列研究。事实上，文献中关于模态逻辑的研究也表明，对应模型的范畴语义本身就颇具意义，相关工作可见 [17]中的综述。

关于在（构造性）证明助手中形式化时序逻辑，这一主题有着悠久的历史，例如参见 [8,44,47]。关于与现代模型检查器的实际应用和比较，可参见 [14], ；而一个完全验证的LTL 模型检查器则在Isabelle定理证明器中实现。在 [37],中，针对标准CTL，在Agda中形式化了一个经过验证的证明搜索程序，并配有用于实现类型正确的证明搜索过程的工具箱；文献LTL在Agda中的类似嵌入见 [28] ，其用于验证函数式响应式程序。我们在Agda中对QLTL 进行的概念验证实现表明，可以进一步推进面向证明助手的量化时序逻辑形式化，而这一问题在文献中仅得到较少关注。

关于图计算模型（GCM），我们在文献中发现了几种形式化方法，它们采用图特定的定义，将关于节点、边、边的源点、边的目标点以及等式的语法陈述作为逻辑中的一等公民，用于表达待分析系统的性质。过去十年间，陆续有论文倡导将量化时序逻辑作为规范 GCM 性质的形式化方法。我们简要回顾了文献中近期出现的一些方案，重点关注全域方法与基于对应物的方法之间的二分差异。

图程序/流图。人们一直倡导使用单子二阶逻辑来证明基于图的编程语言的性质 [45,38]，其中重点是从图变换规则和前置条件公式中提炼出后置条件公式。在 [4,32], 中提出了一种更为抽象的运行时验证元模型，其中控制流图可以实例化为具体模型，而性质则由一阶公式给出。尽管存在差异，但在这两种情况下，所得分析都类似于采用全域方法。

度量逻辑，*I*. 使用轨迹和一阶规范是运行时验证的关键要素。一项相关提议是采用度量一阶时序逻辑（MFOTL） [39,40]，研究适当片段的表达能力 [27]，或探讨与我们的正范式（PNF）类似的对偶性结果 [34]。这些逻辑允许人们对状态的各个组成部分进行推理，并使用（任意）关系集作为模型，从而能够对不同类型的图进行编码。我们这项工作的核心区别在于：与 MFOTL 的标准模型不同，我们允许时序结构具有可变定义域，并允许通过对应关系创建和销毁节点。

度量逻辑，*II*。一种面向图的 MFOTL 方法由度量时序图逻辑（MTGL） [22,42], 提出，它允许对结构及状态的属性建模，并已用于形式化测试的上下文 [41]。这里，轨迹是表示规则的单射跨度对，与我们的部分图态射等价。其语法针对这类规则量身定制，因此 ϕ_G 表示关于给定图 G的公式，而单步$\exists$（f， ϕ_H）以单态 f: $G \to H$为索引，大致表示由规则诱导的部分态射。因此，除了我们对关系的使用之外，元素的恒等性及其保持/删除似乎都是隐含的；探索与基于对应物的 QLTL 之间的联系，也是我们未来的研究方向之一。

7 结论与未来工作

我们已经看到，一阶线性时序逻辑 QLTL 的集合论语义可以在对应体语境中加以呈现。我们还看到，其语法和语义可以自然地应用于代数语境，以表达有向图及其随时间的演化所具备的性质；并且，前几节介绍的概念和模型可以在基于依赖类型论的证明助手（如 Agda）中形式化并进行实际试验。我们研究了该逻辑在关系和偏函数情形下正范式的一些结果，并论证了它们在实践中以及在构造性证明助手中的用途。

我们确定了多种可能扩展本项工作的方向。

二阶。 我们的理论阐述和形式化工作聚焦于 QLTL 的一阶方面。其语义 [17,16] 还允许对元素集合进行量化。由于 Agda 通常将子集形式化为谓词，这在 Agda 中并不实用，因为在具体示例中呈现这种形式会十分繁琐，例如表达对元素子集的全称量化和外延相等。一种可能的扩展方向是研究实用的编码和可能的自动化技术，以便为基于对应体的时序逻辑引入二阶量化。

CTL 及其他逻辑。 本文介绍的量化时序逻辑着重于提供一组受限但足够强大的算子和结构。这些逻辑可以扩展到其他构造和模型，例如 CTL [13] 的构造和模型。将我们的逻辑扩展到更复杂的模型似乎是一项直接的任务，但这可能会导致为获得正范式所需的时序算子发生组合爆炸。

自动化与求解器。 我们强调了验证时序公式所需的证明必须由用户手动提供。为了重复利用现有工作和算法，并为证明助手提供的自动化功能提供更高效的替代方案，人们在将证明助手与外部求解器和检查器进行集成方面投入了大量精力。传统的证明自动化方式是通过使用内部 和外部 求解器来实现的：第一种技术利用 Agda 的反射能力，使得可以按照[37,14,29]所体现的思路，直接在 Agda 中编写经过验证的求解器和证明搜索过程。第二种机制则是为外部程序（例如模型检查器、可满足性模理论求解器和布尔可满足性求解器）编写绑定接口，从而将公式证明或反例生成的任务交由效率更高、专业性更强的程序完成。这项工作的一个可能扩展方向是将上述任一机制应用于对应体语义这一领域。

有限轨迹。 人工智能领域的一个当前趋势，是研究有限轨迹上的时序公式 [21], , 这源于其在规划和强化学习中的应用。我们的模型似乎非常适合应对这一发展，因为每条有限轨迹都可以看作一条以空图中的一个环为终点的无限轨迹，因此也继承了我们所指出的关于该逻辑正范式的所有问题。

参考文献

1. Baldan, P., Corradini, A., König, B., Lluch-Lafuente, A.：用于验证图变换系统的时序图逻辑。载于：Fiadeiro, J.L., Schobbens, P.（编）WADT 2006。LNCS，第 4409 卷，第 1–20 页。施普林格（2006） 2. Belardinelli, F.：量化模态逻辑与物理对象本体论（2004–2005） 3. Blackburn, P., van Benthem, J., Wolter, F.（编）：模态逻辑手册，第 3 卷。北荷兰出版社（2007） 4. Búr, M., Marussy, K., Meyer, B.H., Varró, D.：通过见证生成计算基于查询的监视器的最坏情况执行时间。ACM 嵌入式计算系统汇刊 **20**(6)（2021） 5. Bustan, D., Flaisher, A., Grumberg, O., Kupferman, O., Vardi, M.：正则空泛性。载于：Borrione, D., Paul, W.J.（编）CHARME 2005。LNCS，第 3725 卷，第 191–206 页。施普林格（2005） 6. Coq 开发团队：Coq 证明助手参考手册（2016） 7. Corradini, A., Heindel, T., Hermann, F., König, B.：半推挤重写。载于：Corradini, A., Ehrig, H., Montanari, U., Ribeiro, L., Rozenberg, G.（编）ICGT 2006。LNCS，第 4178 卷，第 30–45 页。施普林格（2006） 8. Coupet-Grimal, S.：归纳构造演算中的线性时序逻辑公理化。逻辑与计算杂志 **13**(6)，801–813（2003） 9. Courcelle, B.：图的单子二阶逻辑。I. 有限图的可识别集合。信息与计算 **85**(1)，12–75（1990） 10. Courcelle, B.：图的单子二阶逻辑。XII. 平面图和平面地图。理论计算机科学 **237**(1)，1–32（2000） 11. Danielsson, N.A.：使用带大小类型的 up-to 技术。载于：POPL 2018。第 43:1–43:28 页。ACM（2018） 12. Dawar, A., Gardner, P., Ghelli, G.：图逻辑的表达能力与复杂度。信息与计算 **205**(3)，263–310（2007） 13. Emerson, E.A.：时序逻辑与模态逻辑。载于：van Leeuwen, J.（编）理论计算机科学手册，B 卷：形式模型与语义，第 995–1072 页。爱思唯尔和麻省理工学院出版社（1990） 14. Esparza, J., Lammich, P., Neumann, R., Nipkow, T., Schimpf, A., Smaus, J.：一个经过完全验证的可执行线性时序逻辑模型检查器。载于：Sharygina, N., Veith, H.（编）CAV 2013。LNCS，第 8044 卷，第 463–478 页。施普林格（2013） 15. Franconi, E., Toman, D.：时序描述逻辑的不动点扩展。载于：Calcanese, D., De Giacomo, G., Franconi, E.（编）DL 2003。CEUR 研讨会论文集，第 81 卷（2003） 16. Gadducci, F., Lluch-Lafuente, A., Vandin, A.：二阶 μ 演算的对应体语义。Fundamenta Informaticae **118**(1-2)，177–205（2012） 17. Gadducci, F., Trotta, D.：量化时序逻辑的预层语义。CoRR $\mathbf{abs/2111.03855}$（2021） 18. Ghamarian, A.H., de Mol, M., Rensink, A., Zambon, E., Zimakova, M.：使用 GROOVE 进行建模与分析。《国际软件工具与技术转移杂志》**14**(1)，15–40（2012） 19. Ghilardi, S., Meloni, G.：模态与时态谓词逻辑：预层中的模型与范畴化概念化。载于：Borceux, F.（编）范畴代数及其应用。LNM，第 1348 卷，第 130–142 页。施普林格（1988） 20. Ghilardi, S., Meloni, G.：关系变量集、部分变量集与基本谓词逻辑。符号逻辑杂志 **61**(3)，843–872（1996）

21. Giacomo, G.D., Vardi, M.Y.：有限轨迹上的线性时序逻辑和线性动态逻辑合成。载于：Yang, Q., Wooldridge, M.J.（编）IJCAI 2015。第1558–1564页。AAAI出版社（2015） 22. Giese, H., Maximova, M., Sakizloglou, L., Schneider, S.：带类型属性图上的度量时序图逻辑。载于：Hähnle, R., van der Aalst, W.（编）FASE 2019。LNCS，第11424卷，第282–298页。施普林格（2019） 23. Girard, J., Lafont, Y., Taylor, P.：《证明与类型》，剑桥理论计算机科学论丛，第7卷。剑桥大学出版社（1989） 24. Hazen, A.：模态逻辑的对应物理论语义。《哲学杂志》**76**(6)，319–338（1979） 25. Hodkinson, I.M., Wolter, F., Zakharyaschev, M.：一阶时态逻辑的单元片段：公元2000—2001年。载于：Nieuwenhuis, R., Voronkov, A.（编）LPAR 2001。LNCS，第2250卷，第1–23页。施普林格（2001） 26. Huang, S., Cleaveland, R.：有限线性时序逻辑的表构造。《逻辑与代数编程方法杂志》**125**，100743（2022） 27. Hublet, F., Basin, D., Krstić, S.：使用度量一阶时序逻辑进行实时策略执行。载于：Atluri, V., Di Pietro, R., Jensen, C.D., Meng, W.（编）ESORICS 2022。LNCS，第13555卷，第211–232页。施普林格（2022） 28. Jeffrey, A.：LTL 类型 FRP：将线性时序逻辑命题作为类型，将证明作为函数式响应式程序。载于：Claessen, K., Swamy, N.（编）PLPV 2012。第49–60页。ACM（2012） 29. Kokke, W., Swierstra, W.：Agda 中的 Auto——使用反射编写证明搜索程序。载于：Hinze, R., Voigtländer, J.（编）MPC 2015。LNCS，第9129卷，第276–301页。施普林格（2015） 30. 刘易斯, D.K.：对应物理论与量化模态逻辑。《哲学杂志》**65**(5)，113–126（1968） 31. Lindblad, F., Benke, M.：Agda 自动定理证明工具。载于：Fil- liâtre, J., Paulin-Mohring, C., Werner, B.（编）TYPES 2004。LNCS，第3839卷，第154–169页。施普林格（2006） 32. Marussy, K., Semeráth, O., Babikian, A.A., Varró, D.：基于部分模型的一致模型生成规约语言。《对象技术杂志》**19**，3:1–22（2020） 33. de Moura, L., Ullrich, S.：Lean 4 定理证明器与编程语言。载于：Platzer, A., Sutcliffe, G.（编）CADE 2021。LNCS，第12699卷，第625–635页。施普林格（2021） 34. Huerta y Munive, J.J.：通过动态自由变量放宽度量一阶时序逻辑的安全性。载于：Dang, T., Stolz, V.（编）RV 2022。LNCS，第13498卷，第45–66页。施普林格（2022） 35. Nipkow, T., Paulson, L.C., Wenzel, M.：《Isabelle/HOL——高阶逻辑证明助手》，LNCS，第2283卷。施普林格（2002） 36. Norell, U.：Agda 中的依赖类型编程。载于：Kennedy, A., Ahmed, A.（编）TLDI 2009。第1–2页。ACM（2009） 37. O'Connor, L.：应用式证明搜索的应用。载于：Chapman, J., Swierstra, W.（编）TyDe@ICFP 2016。第43–55页。ACM（2016） 38. Poskitt, C.M., Plump, D.：GP 2 的单子二阶错误逻辑。《逻辑与代数编程方法杂志》**130**，100825（2023） 39. Schneider, J., Basin, D., Krstić, S., Traytel, D.：度量一阶时序逻辑的形式化验证监视器。载于：Finkbeiner, B., Mariani, L.（编）RV 2019。LNCS，第11757卷，第310–328页。施普林格（2019） 40. Schneider, J., Traytel, D.：度量一阶时序逻辑监控算法的形式化。形式化证明档案（2019年7月）

41. Schneider, S., Maximova, M., Sakizloglou, L., Giese, H.：使用度量时序图逻辑对定时图变换系统进行形式化测试。《International Journal on Software Tools for Technology Transfer》**23**(3)，411–488（2021） 42. Schneider, S., Sakizloglou, L., Maximova, M., Giese, H.：度量时序图逻辑的乐观与悲观即时分析。载于：Gadducci, F., Kehrer, T.（编）ICGT 2020。LNCS，第12150卷。施普林格（2020） 43. Smid, W., Rensink, A.：使用 GROOVE 重构类图。载于：Gorp, P.V., Rose, L.M., Krause, C.（编）TTC 2013。EPTCS，第135卷，第83–87页（2013） 44. Sprenger, C.：Coq 中模态 μ-演算的经验证模型检查器。载于：Steffen, B.（编）TACAS 1998。LNCS，第1384卷，第167–183页。施普林格（1998） 45. Wulandari, G.S., Plump, D.：使用单子二阶逻辑验证图程序。载于：Gadducci, F., Kehrer, T.（编）ICGT 2021。LNCS，第12741卷，第240–261页。施普林格（2021） 46. Zambon, E., Rensink, A.：咖啡配方：在 GROOVE 中组合构造 JAVA 控制流图。载于：Müller, P., Schaefer, I.（编）《原则性软件开发》。第305–323页。LNCS，施普林格（2018） 47. Zanarini, D., Luna, C., Sierra, L.：（余）归纳构造演算中的交替时间时序逻辑。载于：Gheyi, R., Naumann, D.A.（编）SBMF 2012。LNCS，第7498卷，第210–225页。施普林格（2012）

A 附录

我们对 图 2 中给出的模型作出明确描述。分配给每个世界的图$D(\omega_i)$ 以及连接它们的图态射集合 $\mathcal{C}\langle\omega_i,\omega_j\rangle$ 如下所示：

$$\begin{array}{ll} D(\omega_0)_E = \{e_0, e_1, e_2\}, & D(\omega_0)_N = \{n_0, n_1, n_2\}; \\ D(\omega_1)_E = \{e_3, e_4\}, & D(\omega_1)_N = \{n_3, n_4\}; \\ D(\omega_2)_E = \{e_5\}, & D(\omega_2)_N = \{n_5\}. \end{array}$$

连接每对世界的原子对应关系集合如下所示

$$\begin{array}{ll} \mathcal{C}\langle\omega_0,\omega_1\rangle = \{C_0\}, & \mathcal{C}\langle\omega_0,\omega_0\rangle = \emptyset, \\ \mathcal{C}\langle\omega_1,\omega_2\rangle = \{C_1, C_2\}, & \mathcal{C}\langle\omega_1,\omega_1\rangle = \emptyset, \\ \mathcal{C}\langle\omega_2,\omega_2\rangle = \{C_1, C_2\}, & \mathcal{C}\langle\omega_0,\omega_2\rangle = \emptyset, \end{array}$$

以及 $C_0 \in \mathsf{GraphRel}(D(\omega_0)\ D(\omega_1))$，$C_1$，$C_2 \in \mathsf{GraphRel}(D(\omega_1)\ D(\omega_2))$，$C_3 \in \mathsf{GraphRel}(D(\omega_2)\ D(\omega_2))$ 是以下图态射

$$\begin{array}{ll} C_{0N} = \{(n_4, n_0), (n_3, n_1), (n_4, n_2)\}, & C_{0E} = \{(e_4, e_0), (e_3, e_1)\}; \\ C_{1N} = \{(n_5, n_3), (n_5, n_4)\}, & C_{1E} = \{(e_5, e_4)\}; \\ C_{2N} = \{(n_5, n_3), (n_5, n_4)\}, & C_{2E} = \{(e_5, e_3)\}; \\ C_{3N} = \{(n_5, n_5)\}, & C_{3E} = \{(e_5, e_5)\}; \end{array}$$

B 省略的证明

我们给出先前所述定理的显式证明。我们考察一些非常简单的情形：图中仅有节点，并且通过一元谓词 x 来表示节点上的某种区分性性质$\mathsf{B}(x)$ 和 $\mathsf{R}(x)$。

B.1 命题命题 4.2

对于 *then* 和 全称直到 两种情况，我们给出一个反例。以公式 $\mathsf{B}(x)\mathsf{T}\mathsf{R}(x)$为例；我们考虑图4

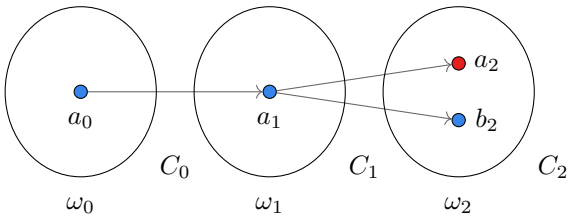

图 4. 一个反例模型：对于对应关系，我们有 $\neg(\mathsf{B}(x)\mathsf{T}\mathsf{R}(x)) \not\equiv (\neg\mathsf{R}(x))\mathsf{U}(\neg\mathsf{B}(x) \wedge \neg\mathsf{R}(x))$。

显然，我们有 $\sigma, \{x \mapsto a_0\} \vDash \neg(\mathrm{B}(x)\mathrm{TR}(x))$。然而， $\sigma, \{x \mapsto a_0\} \nvDash (\neg\mathrm{R}(x))\mathrm{U}(\neg\mathrm{B}(x) \wedge \neg\mathrm{R}(x))$，因为直到算子要求存在一个单一的对应项，使得经过 n 个步骤后，$\neg\mathrm{B}(x)$ 和 $\neg\mathrm{R}(x)$ 都成立。 $\phi_1\mathrm{F}\phi_2$ 及其使用 $\phi_1\mathrm{W}\phi_2$ 表示的否定形式的情况也类似。

B.2 命题 4.4 的证明

我们为直到 $\phi_1\mathrm{U}\phi_2$ 和全称直到 $\phi_1\mathrm{F}\phi_2$的情形给出直接反例，而弱直到 $\phi_1\mathrm{W}\phi_2$ 和然后 $\phi_1\mathrm{T}\phi_2$ 的情形显然也可类似得出。

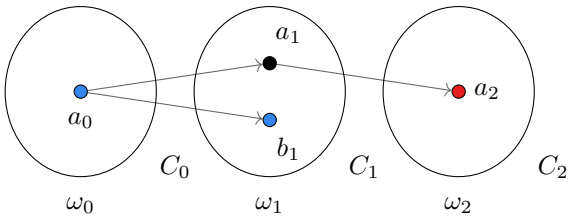

图 5. 一个反例模型：在对应关系的情形下，我们有 $\mathrm{B}(x)\mathrm{U}\mathrm{R}(x) \not\equiv \mathrm{R}(x) \vee (\mathrm{B}(x) \wedge \mathrm{O}(\mathrm{B}(x)\mathrm{U}\mathrm{R}(x)))$。

考虑公式 $\mathrm{B}(x)\mathrm{UR}(x)$ 的直到情形。在图 5 所示的模型中，我们有 $\sigma, \{x \mapsto a_0\} \vDash \mathrm{B}(x)\mathrm{UR}(x)$，因为两步之后存在一个满足 $\mathrm{R}(x)$ 的对应图，而对于它之前的所有图，都存在一个满足 $\mathrm{B}(x)$ 的对应图。然而，显然 a_0 不满足展开公式，因为既不成立 $\sigma_1, \{x \mapsto a_1\} \nvDash \mathrm{B}(x)\mathrm{UR}(x)$，也不成立 $\sigma_1, \{x \mapsto b_1\} \nvDash \mathrm{B}(x)\mathrm{UR}(x)$。

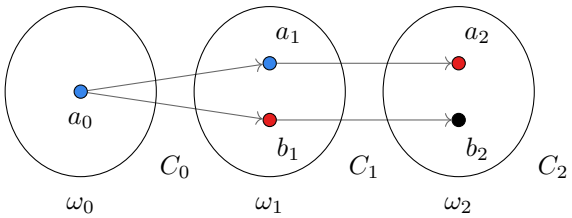

图 6。 一个反例模型：对于对应关系，我们有 $\mathrm{B}(x)\mathrm{FR}(x) \not\equiv \mathrm{R}(x) \vee (\mathrm{B}(x) \wedge \mathrm{A}(\mathrm{B}(x)\mathrm{FR}(x)))$。

考虑公式 $\mathrm{B}(x)\mathrm{FR}(x)$ 的全称直到情形。在图 6 a_0 所示的模型中，展开公式成立，因为对于原子对应物 a_1 和 b_1 σ， $\{x \mapsto a_0\} \vDash \mathrm{B}(x)\mathrm{FR}(x)$，且 σ 和 $\{x \mapsto a_0\} \vDash \mathrm{B}(x)\mathrm{FR}(x)$，其中所有对应物都满足 $\mathrm{R}(x)$ 的世界分别经过两步和一步后到达。然而， $\sigma, \{x \mapsto a_0\} \nvDash \mathrm{A}(\mathrm{B}(x)\mathrm{FR}(x))$ 不成立，因为不存在单一的世界 ω_n ，使得经过 n 步后所有对应物都满足 $\mathrm{R}(x)$。

C Agda 示例

本节将通过 Agda 中的一个简单示例，说明如何使用我们的库来形式化一个对应模型，并证明模型满足某些公式；该示例实现了图 2中展示的贯穿全文的例子。

我们的库能够处理定义在任意多排序签名上的项和代数，其中有向图是此类情况的一个特例。为了形式化本文展示的示例，我们首先定义图的签名 Gr，并展示一个图的示例 G_1，将其视为给定签名上的代数，同时省略模型中的其余图。

```
导入 Data.Unit
模式* = lift Data.Unit.tt

数据 Gr-Sorts: Set 其中
  边 : Gr-类型
  节点 : Gr-类型

数据 Gr-函数: 集合 其中
  s : Gr-函数
  t : Gr-函数

Gr : 签名
Gr = 记录{ 𝒮 = Gr-Sorts
         ; ℱ = Gr-函数
         ; 标记ℱ = λ { s → [ 边 ] ↦ 节点
                     ;t → [ 边 ] ↦ 节点
                     }
         } 其中打开导入 Data.Vec使用 ([_])

数据G₀-边 : 集合 其中 e0 e1 e2 : G₀-边数据G₀-节点 :
集合 其中 n0 n1 n2 : G₀- 节点

G₀ : Σ-代数Gr G₀ = record { S = λ { 边 → G₀-边 ; 节点 →
G₀-节点 } ; F = λ { s → λ { (e0 , *) → n0 ; (e1 , *) → n1 ;
(e2 , *) → n2 } ; t → λ { (e0 , *) → n1 ; (e1 , *) → n2 ;
(e2 , *) → n0 } } }
```

– <省略的 G_1,G_2> 的定义

给定两个图，我们可以按照定义 定义 2.2 中的规定，通过特化代数间关系 Σ-Rel 这一更一般的概念，来刻画图态射的概念，该概念是在我们的图的签名下定义的。我们以关系 C_0 为例，在 图 2中展示了这一情形：请注意，我们要求提供一个证明 ρ-homo ，以说明所给出的定义确实是一个关系图（同）态射。

data C_0-边 : G_0-边 $\to$ G_1-边 $\to$ Set where e0e4 : C_0-边 e0 e4 e1e3 : C_0-边 e1 e3 data C_0-节点 : G_0-节点 $\to$ G_1-节点 $\to$ Set where n0n4 : C_0-节点 n0 n4 n1n3 : C_0-节点 n1 n3 n2n4 : C_0-节点 n2 n4 C_0 : Σ-Rel G_0 G_1C_0 = record{ $\rho = \lambda$ { {Edge} $\to$ C_0-边 ; {Node} $\to$ C_0-节点 } ; ρ-homo $= \lambda$ { s (e 0e4 , *) $\to$ n0n4 ; s (e1e3 , *) $\to$ n1n3 ; t (e0e4 , *) $\to$ n1n3 ; t (e1e3 , *) $\to$ n2n4 } }

数据 C_1-边：G_1-边 $\to$ G_2-边 $\to$ Set，其中 e4e5_2 : C_1-边 e4 e5 数据 C_1-节点：G_1-节点 $\to$ G_2-节点 $\to$ Set，其中 n3n5_2 : C_1-节点 n3 n5 n4n5_2 : C_1-节点 n4 n5 – $<$省略了 C_1,C_2,$C_3$$>$ 的定义

现在，我们可以定义世界集合 W 以及模型的可达关系 _⇝_，使其结构与 图 1 中给出的结构相对应。最后，我们通过为每个世界分配一个图（其中包含 对应模型 M ），并为每个可达关系分配一个图态射（其中包含 d ），来定义我们的模型 $\mathcal{C}$。

```
data W : Set where
  ω0 ω1 ω2 : W

data _⇝_ : W → W → Set where
  a0 : ω0 ⇝ ω1
  a1 a2 : ω1 ⇝ ω2
  a3 : ω2 ⇝ ω2

d : W → Σ-Algebra Gr
```

d ω_0 = G_0d ω_1 = G_1d ω_2 = $G_2\mathcal{C}$: ∀ {A B} → A ⇝ B → Σ-Rel (d A) (d B)$\mathcal{C}$ a_0 = $C_0\mathcal{C}$ a_1 =$C_1\mathcal{C}$ a_2 =$C_2\mathcal{C}$ a_3 =C_3

M: 对应模型 Gr
M = 记录{ W = W ; d = d ; _⇝_ = _⇝_ ; $\mathcal{C}$ = $\mathcal{C}$ }

一个 对应轨迹 通过指明模型的可达关系来定义，并以其起始的第一个世界作为索引；下面我们通过定义第 $\sigma = (C_0,\ C_1,\ C_3,\ C_3,\ \ldots)$ 节中使用的 第 3.3 节。

~~σ~~在 ω_2~~ω_2~~上重复轨迹，其中具有一个$_3$
σ ω_2 $_2$ -环-:
对应轨迹 M - 环 -σ = $_{\circ_3}$
B -环-ω_2 。关系 aσ = σ -
环-ω_2 。尾部 -环-$\omega_2\sigma$: 对应轨迹 M
$\omega_0\sigma$ = a_0 →a_1 → σ-环-ω_2

我们将第 第 3.3 节中的简写公式形式化。我们在变量中使用德布鲁因指标来处理量词的绑定符并确保公式的作用域良构性，同时使用一些辅助符号 v0、v1、v2 来表示项中的变量。

present: ∀ {τ} → QLTL (τ :: []) present {τ} = ∃< τ > v1 ≡^{t} v0 notPresent: ∀ {τ} → QLTL (τ :: []) notPresent {τ} = ∀< τ > v1 ≢^{t} v0 nextPreserved: ∀ {τ} → QLTL (τ :: []) nextPreserved = present ∧ O present nextDealloc: ∀ {τ} → QLTL (τ :: []) nextDealloc = present ∧ A notPresent loop: ∀ {n} {Γ : Ctx n} → QLTL (Edge :: Γ) loop = s $ (v0 , *) ≡^{t} t $ (v0 , *) worldHasLoop: QLTL [] worldHasLoop = ∃<Edge >loop hasLoop: QLTL(Node :: [])

有环 $= \exists <$ 边 $>$ (s \$ (v0 , *) $\equiv^t$ v1 $\wedge$ 环) 相邻：QLTL(节点 :: 节点 :: []) 相邻 $= \exists <$ 边 $>$ ((s \$ (v0 , *) $\equiv^t$ v1 $\wedge$ t \$ (v0 , *) $\equiv^t$ v2)$\vee$ (s \$ (v0 , *) $\equiv^t$ v2 $\wedge$ t \$ (v0 , *) $\equiv^t$ v1)) 可复合：QLTL(边 :: 边 :: []) 可复合 $= \exists <$ 节点 $>$(s \$ (v2 , *) $\equiv^t$ v0 $\wedge$ t \$ (v1 , *) $\equiv^t$ v0) 具有复合：QLTL(边 :: 边 :: []) 具有复合 $=$可复合 $\wedge \exists <$ 边 $>$ (s \$ (v0 , *) $\equiv^t$ s \$ (v2 , *)$\wedge$ t \$ (v0 , *) $\equiv^t$ t \$ (v1 , *)) 将合并：$\forall$ {τ} $\rightarrow$ QLTL (τ :: τ :: []) 将合并 {τ} $=$ v0 $\not\equiv^t$ v1 $\wedge \Diamond$ v0 $\equiv^t$ v1 始终保留：$\forall$ {τ} $\rightarrow$ QLTL (τ :: []) 始终保留 {τ} $= \Box$ 存在 {τ} 将变为环：QLTL(边 :: []) 将变为环 $=$! 环 $\wedge \Diamond$ 环

我们展示了如何使用我们的框架，在 Agda 中形式化验证 第 3.3 节 中所示全部示例公式的有效性。其中许多示例都足够简单，Agda 可以自动为该公式提供证明项。

```
_ : skip 0 σ , (e0 , *) ⊨ nextPreserved {Edge}
_ = (e0 , refl) , ((e4 , *) , (e0e4 , *) , e4 , refl)
_ : skip 0 σ , (n1 , *) ⊨ nextPreserved {Node}
_ = (n1 , refl) , ((n3 , *) , (n1n3 , *) , n3 , refl)
_ : skip 0 σ , (e2 , *) ⊨ ! nextPreserved {Edge}
_ = λ { ((e2 , refl) , ()) }

_ : skip 1 σ , (n3 , *) ⊨ ! nextDealloc {Node}
_ = λ { ((.n3 , refl) , ¬d) → ¬d (n5 , *) (n3n5₂ , *) n5 refl }
_ : skip 1 σ , (e3 , *) ⊨ nextDealloc {Edge}
_ = (e3 , refl) , (λ { (e5 , *) (() , *) e5 refl })
_ : skip 1 σ , (e4 , *) ⊨ ! nextDealloc {Edge}
_ = λ { ((e4 , refl) , ¬d) → ¬d (e5 , *) (e4e5₂ , *) e5 refl }

_ : skip 0 σ , (e0 , *) ⊨ ! loop
_ = λ ()
_ : skip 1 σ , (e3 , *) ⊨ ! loop
_ = λ ()
_ : skip 2 σ , (e5 , *) ⊨ loop
_ = refl

_ : skip 0 σ , (n0 , *) ⊨ ! hasLoop
```

_ = λ { (e0 , ()) ; (e1 , ()) ; (e2 , ()) }_ : 跳过 1 σ , (n3 , *) ⊨ ! 有环 _ = λ { (e3 , ()) ; (e4 , ()) }_ : 跳过 2 σ , (n5 , *) ⊨ 有环_ = e5 , (refl , refl)_ : 跳过 0 σ , * ⊨ ∃< 节点 > (! 有环)_ = n0 , (λ { (e0 , ()) ; (e1 , ()) ; (e2 , ()) })_ : 跳过 1 σ , * ⊨ ∃< 节点 > (! 有环)_ = n3 , λ { (e3 , ()) ; (e4 , ()) }_ : 跳过 2 σ , * ⊨ ∃< 节点 > 有环_ = n5 , (e5 , (refl , refl))_ : 跳过 0 σ , (n0 , n1 , *) ⊨ 相邻_ = e0 , inj$_1$ (refl , refl)_ : 跳过 1 σ , (n3 , n4 , *) ⊨ 相邻_ = e3 , inj$_1$ (refl , refl)_ : 跳过 1 σ , (n3 , n4 , *) ⊨ O 相邻_ = ((n5 , n5 , *) , (n3n5$_2$, n4n5$_2$, *) , e5 , inj$_1$ (refl , refl))_ : 跳过 0 σ , (e0 , e1 , *) ⊨ 可复合_ = n1 , refl , refl_ : 跳过 0 σ , (e0 , e1 , *) ⊨ O 可复合_ = (e4 , e3 , *) , (e0e4 , e1e3 , *) , n3, refl , refl_ : 跳过 1 σ , (e3 , e4 , *) ⊨ ! O 可复合_ = λ { () }_ : 跳过 0 σ , (e0 , e1 , *) ⊨ ! 具有复合_ = λ { ((n1 , refl , refl) , e2 , ()) }_ : 跳过 0 σ , (e0 , e1 , *) ⊨ ! O 具有复合_ = λ { ((e4 , e3 , *) , (e0e4 , e1e3 , *) , (n3 , refl , refl) , e3 , ()) ; ((e4 , e3 , *) , (e0e4 , e1e3 , *) , (n3 , refl , refl) , e4 , ()) }_ : 跳过 2 σ , (e5 , e5 , *) ⊨ O 具有复合_ = (e5 , (e5 , *)) , (e5e5 , e5e5 , *) , (n5 , refl , refl) , e5 , refl , refl_ : 跳过 1 σ , * ⊨ ∃< 节点 > ∃< 节点 > 将合并_ = n3 , (n4 , (λ ()) , 1 , (λ { zero (s≤s z≤n)→ (n4 , (n3, *)) , ((refl , refl , *) , *) ; (suc _) (s≤s ()) }) , (n5 , (n5 , *)) , ((n5 , (n4n5$_2$, refl)) , (n5 , n3n5$_2$, refl) ,

*)

, refl)_ : 跳过 0 σ , * ⊨ ∃< 节点 > ∃< 节点 > 将合并 _ = n0, （n1, （λ （） ）, 2, （λ { zero (s≤s z≤n) → (n1, (n0 , *)) , ((refl , refl , *) , *)

；(suc zero) (s≤s (s≤s z≤n))→ (n3 , n4 , *) , (((n3 , n1n3 , refl) , ((n4, n0n4 , refl) , *)) , *) }) , (n5 , (n5 , *)) , ((n3 , (n1n3 , (n5 , (n3n5₂ , refl)))) , ((n4 , n0n4 , (n5 , n4n5₂ , refl)) , *)) , refl)_‘: 跳过 0 σ , * ⊨ ! ∀< 边 > 始终保留‘ ¬∀ 与 ¬∀ e2_ ..... | inj₂ 始终与 always 1 ..... | ()_ : 跳过 0 σ , * ⊨ ∃< 边 > 将变为环_ = e0 , ((λ ()) , 2 , (λ { zero (s≤s z≤n) → (e0 , *) , ((refl , *) , *) ; (suc zero) (s≤s (s≤s z≤n)) →(e4 , *) , ((e4 , e0e4 , refl) , *) , * }) , ((e5 , *) , (((e4 , (e0e4 , e5 , (e4e5₂ , refl))) , *) , refl)))_ : 跳过 0 σ , * ⊨ (∃<边 > s $ (v0 , *) ≠ᵗ t $ (v0 , *)) U (∃<节点 > ∀< 节点 > v0 ≡ᵗ v1)_ = 2 , (λ { zero (s≤s z≤n) → * , (* , (e2 , (λ ()))) ; (suc zero) (s≤s (s≤s z≤n)) → * , (* , (e3 , (λ ()))) }) , * , * , n5 , (λ { n5 → refl })∃□环: 跳过 2 σ , * ⊨ ∃< 边 > (□ 环)∃□环 =e5, inj₂ □环 where□环: _□环 zero = (e5 , *) , (refl , *) , refl□环 (suc i) with □环 i ... | (e , *) , (ρ-e5 , *) , isLoop = (e , *) , (((e5 , (e5e5 , ρ-e5)) , *) , isLoop)_ : 跳过 0 σ , * ⊨ ∃< 边 > (◊ □ 环)_ = e0 , 2 , (λ { zero (s≤s z≤n) → (e0 , *) , (refl , *) , * ; (suc zero) (s≤s (s≤s z≤n))→ (e4 , *) , ((e4 , e0e4 , refl) , *) , * }) , (e5 , *) , ((e4 , (e0e4 , (e5 , (e4e5₂ , refl)))) , *) , proj₂ ∃□环