arXiv:2603.13603v1 [cs.DB] 13 Mar 2026

等价定理：结构完整数据库系统中的一等关系

Matthew Alford*Network*
Service
*Group*malford@nsgdv.comORCID:
0009-0007-3856-8362

2026年2月

摘要

我们证明了**等价定理**：结构完整的知识表示必须恰好具备四种相互蕴含的能力——带属性的 n 元关系、时间有效性、不确定性量化，以及关系之间的因果关系；这些能力共同等价于将关系视为一等对象。任何实现其中一种能力的系统都必然需要具备全部四种能力；任何缺少其中一种能力的系统都无法实现结构完整性。这一结果具有构造性：我们展示了一个属性化时序因果超图 （ATCH）框架，可同时满足全部四个条件。

该定理给出了严格的表达能力层级——SQL $\subsetneq$ LPG $\subsetneq$ TypeDB $\subsetneq$ ATCH——并提供了见证查询，证明在每个较低层级中，这些查询在结构上都无法表达。我们建立了计算复杂度界，表明一般查询是 NP 完全的，而实用的查询类别（无环模式、有界深度因果链、窗口化时态查询）可在多项式时间内处理。作为直接推论，我们为经典人工智能中的问题推导出解决方案：框架问题（由时间有效性实现默认持久性）、冲突解决（将矛盾视为尚未解决的元数据，并发现隐变量），以及常识推理（带有因果抑制因素的默认值）。

一个以 C 语言编写的 PostgreSQL 扩展原型，在已确立的 - 已确立的复杂度界。

关键词： 知识表示、等价定理、超图、时序数据库、因果推断、一等关系、框架问题

目录

1 引言 **4**
1.1 核心问题.4 1.2 “完整”意味着什么.4 1.3 贡献.5

2 Moti应用示例：从简单到完整 **5**
2.1 示例 1：团队会议（支柱 1 和 2）.6 2.2 示例 2：供应链（支柱 1–3）.6 2.3 示例 3：IT基础设施事件（全部 4 个支柱）.6 2.4 示例 4：旅行中断（全部 4 个支柱）.7

3 现有系统为何会失败：具体示例 **7**
2.5 可视化：投影下的信息丢失.8 3.1 SQL（关系数据库）.8

2.6 可视化：因果链.9 3.2 带标签属性图（Neo4j 及类似系统）.9 3.3 语义三元组存储（RDF/OWL）.10 3.4 TypeDB：最接近但仍不完整.10

4 形式化框架 **11**
4.1 属性化时序因果超图.11 4.2 四大支柱：形式化定义.11 4.3 一等对象：操作性定义.12

5 相关工作 **12**
5.1 因果推理：Pearl 的 Do演算.12 5.2 概率数据库.12 5.3 时序数据库.13 5.4 默认与非单调推理.13 5.5 情境演算与事件演算.13 5.6 描述逻辑与时序扩展.14 5.7 超图数据库与理论.14 5.8 数据模型中的一等关系.14

6 等价定理 **15**
6.1 形式化预备知识.15 6.2 主定理.15

7 经典人工智能问题的应用 **17**
7.1 框架问题.17 7.2 不确定性传播.18 7.3 冲突解决.19 7.3.1 非破坏性存储.19 7.3.2 解决层级.19 7.3.3 上下文坍缩原理.19 7.3.4 因果审计.20 7.3.5 自动隐藏变量发现.20 7.4 常识推理.21 7.4.1 基于不确定性的默认逻辑.21 7.4.2 抑制关系.22

8 计算复杂度 23
8.1 困难性结果.23 8.2 可处理片段.23

9 原型实现 24
9.1 存储模型.24 9.2 API 接口.25 9.3 复杂度验证.25

10 表达能力基准 **25**

11 信息 信息论分析 2 **7**

12 讨论 **28**
12.1 不同知识类型的表征覆盖.28 12.2 对人工智能的启示.29 12.3 局限性.30

13 结论 **31**

一种模态逻辑扩展 **33**

1 引言

请考虑以下现实世界中的知识：

> **旅行：**“航空公司在三月风暴期间更改123号航班的时刻表，**导致**在亚特兰大错过转机，**因此需要**在酒店过夜，费用由航空公司承担。”
>
> **IT 基础设施：**“AcmeCorp 的域控制器在夜间更新 Windows 后，会计团队有一半的人周一早上无法打印——这次更新**重置了**一项组策略，而该策略与三周前为解决另一问题而推送的打印驱动程序**发生冲突**。”

这些简单陈述包含了现有数据库系统都无法自然表示的知识（可视化示例见图2）。每个陈述都涉及：具有属性的 n元关系（航空公司/更新、受影响的系统、上下文）；时间有效区间；不确定性（风暴是真正的原因吗？GPO 冲突是实际的作用机制吗？）；以及关系影响其他关系的因果链。

1.1 核心问题

本文探讨一个基础性问题：

> 数据库系统要在不产生结构性损失的情况下表示任意现实世界知识，必须满足哪些条件？

我们证明，答案恰好涉及四种能力，我们称之为四大支柱：

1. **P1**：带属性的 N 元关系 2. **P2**：时间有效性
3. **P3**：不确定性量化 4. **P4**：关系之间的因果关系

1.2 “完整”的含义

我们必须准确说明此处“完整”的含义（另请参见备注 4.8，其中区分了系统性的一等地位与特设的一等地位）。

定义 1.1（表达完备性）。 知识表示系统 $\mathcal{K}$ 具有表达完备性，当且仅当对于任何能够用自然语言无歧义地表述的事实 f，都存在一种表示 $r \in \mathcal{K}$，使得：

1. r 保留 f中的结构关系*2.* r 保留 f的时间范围*3.* r 保留 f的确定性级别*4.* r 保留 f中的因果联系

注释 1.2 （“完整”并不意味着什么）。 “完整”并不意味着：

- 全知：系统知晓一切
- 推理完备：系统能够推导出所有逻辑后果
- 计算高效：所有查询都很快

这意味着：任何已存在的知识都可以在不造成结构失真的情况下编码。特定领域的推理可能需要额外的机制，但表示本身不会丢失任何信息。

示例 1.3 （语言类比）。 英语对于人类交流来说是“完备”的——你可以表达任何想法。摩尔斯电码对于传输英语来说是“完备”的——不会丢失任何信息。仅靠表情符号则不完备——有些概念无法表达。

同样，该框架对于知识表示来说是完备的；*SQL* 则不是。

1.3 贡献

本文做出了以下贡献：

1. **等价定理：** 证明四大支柱彼此蕴含，且等价于一等关系地位（第 6 节）。 2. **形式化框架：** 构建能够涵盖四大支柱的属性化时序因果超图（第 4 节）。 3. **严格表达能力层级：** 证明 SQL $\subsetneq$LPG $\subsetneq$TypeDB $\subsetneq$ATCH 的表达能力依次递增，并为每个层级差异提供见证查询（第 10 节）。 4. **经典人工智能应用：** 将框架问题、冲突解决、不确定性传播和常识推理的解决方案作为四大支柱的直接推论（第 7 节）。 5. **复杂性分析：** 给出计算复杂性界，并为实际查询确定可处理片段（第 8 节）。 6. **信息论分析：** 形式化量化投影到低表达能力模型时的信息损失（第 11 节）。 7. **具体失败分析：** 展示关系数据库、带标签属性图、语义三元组存储和高元系统中的结构性失败（第 3 节）。 8. **渐进式示例：** 通过四个示例说明每个支柱为何不可或缺（第 2 节）。 9. **原型实现：** 用 C 语言开发 PostgreSQL 扩展，验证其实际可行性（第 9 节）。

2 个动机示例：从简单到完整

我们介绍nt 四个复杂程度逐步增加的示例，展示每个支柱如何变得不可或缺 y.

2.1 示例 1：团队会议（支柱 1 和 2）

知识： “Alice、Bob 和 Carol 于 1 月 15 日在 101 号房间开会。”**尝试用 SQL 表示：**

```
-- 二元方法：丢失会议身份CREATE TABLE attends ( person_id INT , meeting_i
d INT ); CREATE TABLE meetings ( id INT , room VARCHAR , date DATE );
```

问题： 会议作为一种关系并没有身份。若不创建一个独立实体，我们就无法说“那场会议 很有成效”。

在我们的框架中：

超边：{ Alice , Bob , Carol , Room101 } 属性：{ 类型："会议"，有成效：true } 有效_时间： [2024-01 -15 09:00 , 2024年 -01 -15 10:00]

四元关系是一个一等对象，拥有自己的属性和时间界限。

2.2 示例 2：供应链（支柱 1–3）

知识： “工厂 X 向制造商 Z 供应零件 Y（可靠性为 85%），有效期至 2025 年 3 月。”

尝试使用 SQL：

```
CREATE TABLE supplies ( factory_id INT , part_id INT ,
manufacturer_id INT , reliability FLOAT , -- 这应附加在哪里？ valid_
until DATE ) ;
```

问题： 可靠性是一个列，而不是关系真值状态的元属性。

在我们的框架中：

超边：{ FactoryX，PartY，ManufacturerZ } 属性：{合同_编号：“C -2024 -001 ”} 有效_时间： [2024-01 -01，2025 -03 -31]置信度：0.85 -- 真值的元属性

2.3 示例 3：IT基础设施事件（全部 4 个支柱）

知识： “Acme Corp 的域控制器在夜间完成 Windows 更新后，会计团队有一半的人周一早上无法打印——这次更新**重置了** 一项**与**三周前为修复另一个问题而推送的打印驱动程序相冲突的组策略。”

支柱	为何必需
P1	每个事件都涉及多个实体（更新、DC，GPO，驱动程序、子网、团队），并具有-属性
P2	驱动程序推送 3 周前 → 更新周日夜间 → 失败，周一早上
P3	GPO 冲突是真正的原因吗？重新-需要调查才能确定
P4	驱动程序推动关系 与向上日期 关系导致失败关系

在我们的框架中：

e1：{ TechJones，打印驱动程序，2 ndFloorPrinters，工单T -4021} 属性：{ change_类型：“驱动程序_更新”，原因：“纸盒_选择_错误”} 有效_时间： [2024-02 -26 10:00，无限）

e2 : { Windows 更新 , 域控制器 , 组策略 } 属性 : { 更新_编号 : " KB5034763 " , 方法 : " scheduled "} 有效_时间 : [2024-03 -18 01:00 , 2024年 -03 -18 03:00]

e3 : { 会计团队 , 打印故障 , 2 ndFloorSubnet } 属性 : { 范围 : " partial " , 受影响 : "2 nd_floor_only " } 有效_时间 : [2024-03 -18 08:00 , 2024年 -03 -18 15:30]置信度 : 0.78 -- 调查后确认 *GPO* 冲突

-- 关系之间的因果链接
e1 -- 与_冲突 -- > *e2* -- 导致 - -> *e3*

借助该框架的实现，诊断这一事件只需调用一个函数：

```
SELECT*FROMatch.trace_因果_链('print_失败_e3', depth:= 3);-- 返回：e1（驱动程序推送，2月26日）->e2（更新，3月18日）->e3（失败）-- 含每条边的置信度分数
```

系统会自动遍历因果链，揭示那项发生在三周前的驱动程序变更；文本搜索或工单查询都无法将其与周一的服务中断联系起来。

2.4 示例 4：旅行中断（全部 4 个支柱）

知识：“风暴期间航空公司的时刻表变更**导致了**在亚特兰大的转机未能衔接，**因此需要**在当地过夜，入住由航空公司承担费用的酒店。”

在我们的框架中：

e1：{ 航空公司, ScheduleChange, Flight123, MarchStorm } 属性：{ 原因：" weather "，提前_通知：" 2 hrs " } 有效_时间： [2024-03 -15 14:00，2024 -03 -15 14:00]

e2：{ 乘客, 错过转机, 亚特兰大 } 属性：{ 原_航班：" UA456 "} 有效_时间： [2024-03 -15 18:00，2024 -03 -15 18:00]

e3：{ 乘客, HotelStay, AirportHilton } 属性：{ 由_承保：" 航空公司 "，晚数：1} 有效_时间： [2024-03 -15 22:00，2024 -03 -16 06:00]

-- 关系之间的因果链接e1-- 导致_ -->*e2* -- 需要 - - >
e3

3 现有系统为何失效：具体示例

现在，我们展示现有系统在表达能力方面的具体失败案例。我们的比较目标是架构类别，而非单个供应商：关系数据库（SQL）、 [5], 标注属性图（以 Neo4j 为代表，也包括 Amazon Neptune、TigerGraph、Memgraph 等）、语义三元组存储（RDF/OWL，包括 RDF-star 等近期扩展），以及高元关系系统（TypeDB）。

2.5 可视化：投影下的信息丢失

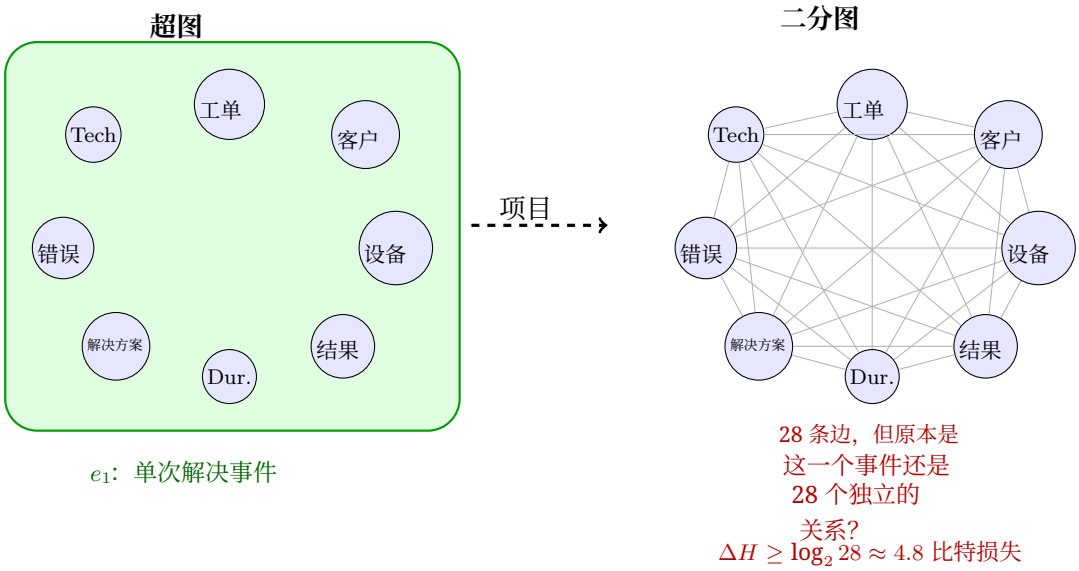

图 1：将一个包含 8 个实体的超边投影为二元边时的信息损失。简洁的单事件结构（左）变成了由 28 条成对边组成的含义模糊的网络（右），并造成无法恢复的关系结构损失。

这些系统在其设计用途方面并无不足；只有当表示目标要求同时具备四大支柱时，它们才会失败。下述失败属于结构性问题——根植于各自架构的数据模型之中——而不是实现方面的缺陷，无法通过未来版本进行修补。

3.1 SQL（关系数据库）

```
CREATE TABLE 处方 (
    编号 INT PRIMARY KEY     ,
    医生_编号 INT ,       药物_编号 INT, 患者_编号 INT ,
    置信度 FLOAT          ,  有效    _自 DATE 起
) ;
CREATE TABLE 反应 (编号 INT , 患者_编号 INT ,                    类型 VARCHAR );

-- 我们该如何表达        " 处方 42 CAUSED                  反应 17"?
创建表 致因 (
    原因    _编号 整数 ,      -- 点          指向哪个表?
    结果    _编号 INT  ,      -- 点          关联到哪个表?
    置信度 FLOAT
) ;
```

失败： 原因表无法将外键指向多个表。标准的变通方案是通过 (entity_type, entity_id) 对实现多态引用，但这会牺牲类型安全、索引效率和规范化保证，实际上是在系统不提供支持的情况下，以特设方式实现一等关系。其他变通方案也好不到哪去：(1) 联合类型会破坏规范化；(2) 使用带有判别列的单一事件表，会将异构关系扁平化为稀疏模式；或者 (3) 按类型划分的链接表

2.6 可视化：因果链

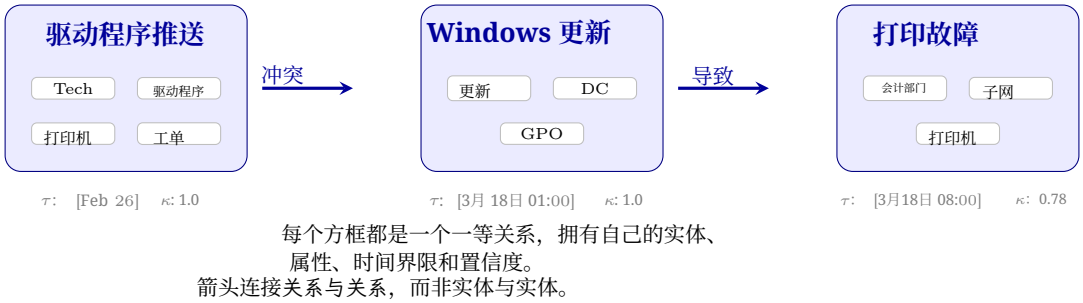

图2：IT 基础设施场景中一等关系之间的因果链。三周前的驱动程序推送（左侧）与夜间 Windows 更新（中间）相互作用，导致打印故障（右侧）。每个关系都包含自己的参与实体，并带有时间和置信度元数据。因果链接连接的是关系，而非实体。

(处方_致因_反应，反应_致因_发现等)会随着新增关系类型而呈组合式增长。

SQL:2011 应用时间段表在一定程度上解决了**P2**，但时间谓词作用于元组，而非作为一等对象的关系。SQL/PGQ 扩展（ISO 9075-16:2023） [12] 为 SQL 增添了属性图查询语法，但沿用了 LPG 数据模型：仅支持二元边，不支持超边，也不支持关系之间的关系。人们无法询问“时间 t的因果结构是什么？”，因为因果结构并非 SQL 或 SQL/PGQ 中的原生构造。每种变通方案都会以特设且依赖具体模式的方式，重新引入一等关系所通用提供的那套机制。

3.2 标记属性图（Neo4j 及类似系统）

Neo4j 是标记属性图（LPG）家族的代表，该家族还包括 Amazon Neptune（OpenCypher 模式）、TigerGraph 和 Memgraph。它们都采用相同的基础数据模型，由 Angles [2] 进行了形式化定义：节点通过携带键值属性的二元有向边相连。

近期发布的 GQL 标准（ISO/IEC 39075:2024） [11]——自 SQL 以来首个新的 ISO 数据库语言——以及 SQL/PGQ 扩展（ISO 9075-16:2023） [12]，将这一模型编纂为国际标准，其形式化模式匹配语义由 Deutsch 等人 [25]和 Francis 等人 [24]进行了分析。关键在于，GQL 和 SQL/PGQ 均未通过超边或关系之间的关系扩展 LPG 数据模型；标准化模型仍然是二元的。这意味着下文指出的结构局限性并非未来版本可以弥补的实现缺陷，而是嵌入标准化数据模型中的架构约束 [3]。

```
// Neo4j Cypher - 关系仅为二元 CREATE (d: 医生 ) -[ rx : PRESCRIBED ]-
>( 药物 : 药物 ) // 患者该放在哪里？必须创建虚假节点。


// “关系之间的关系”——不可能匹配（r1）-[: CAUSED ]- >（r2）// r1、r2 必须是
节点
```

失败：处方涉及 4 个实体，但 LPG 的边恰好连接 2 个节点。对处方建模需要将关系具体化为节点，这会丢失关系语义：具体化后的节点没有元数约束，无法强制规定参与角色，而且在图中与任何其他节点都无法区分。元关系（“该处方导致了该反应”）需要进行第二次具体化，从而进一步加剧语义损失。

3.3 语义三元组存储（RDF/OWL）

RDF 将知识表示为主语–谓语–宾语三元组，因此其本质上是二元的。有几种扩展可部分解决这一局限性：

- **经典再化**会为每条元语句创建四个三元组（包括该语句本身，以及分别表示其主语、谓语和宾语的三元组）。这是一种变通方案，而非原生支持：再化后的语句与原始三元组之间没有强制关联，查询必须手动重建这种关系。

- **命名图**允许将三元组归组到一个可携带元数据的图标识符之下，从而提供一种部分机制，用于将时间或溯源信息附加到一组三元组上。

- **RDF-star** （W3C 社区组 2023年报告；作为 RDF 1.2 一部分的正式 W3C 标准化工作仍在进行中）支持嵌入式三元组——一个三元组可以成为另一个三元组的主语或宾语——从而直接解决元关系问题。

即使有 RDF-star，结构局限性依然存在。N 元关系必须拆解为多个由空白节点或辅助资源连接的三元组；多元模式是有文档记录的变通方案，而非原生构造。时间有效性需要依赖词汇约定（例如 OWL-Time），却没有查询语言集成——SPARQL 没有 AT TIME 运算符。置信度分数可以通过 RDF-star 注释附加，但缺少传播语义。因此，我们将 RDF/OWL 对 **P1** 和 **P4**的支持评为 $\circ$（变通方案），对 **P2** 和**P3**的支持评为 $\times$。

3.4 TypeDB：最接近，但仍不完整

TypeDB 支持 n元关系（**P1**）和部分关系间引用（**P4**）。在所考察的系统中，TypeDB 最接近结构完整性；它对 n元关系的明确支持也印证了支柱 1 的重要性。

```
// TypeDB - 可以建模结构：定义处方子关系，关联医生、药物和患者；//
但无法回答：match $p isa prescription ； AT TIME '2024 -07 -01 ' --
不支持（没有 P2 ） $p has confidence $c; -- 不会传播（没有 P3 ）
```

失败： TypeDB 原生支持**P1** ，并部分支持**P4** （关系可以参与其他关系，但不具备因果语义或传播规则）。然而，它将时间有效性与不确定性视为用户定义的元数据——关系上的可选属性——而非具有查询语言集成和传播语义的内在结构约束。在 TypeDB 中，valid_from 属性在语法上与其他任何属性都无法区分；如果没有应用逻辑来解释这一约定，系统就无法回答“在时间 t，什么是真实的？”。同样，置信度属性只是被动属性，不会沿因果链传播。这正是等价定理所指出的“结构性损失”

：如果系统不原生支持全部四大支柱，那么无论用户采用何种元数据约定，其数据模型都无法保留知识的结构维度。

4 形式化框架

4.1 属性化时序因果超图

定义 4.1（属性化时序因果超图）。 一个 属性化时序因果超图（*ATCH*）是一个元组：

$$\mathcal{H} = (V, \mathcal{E}, \mathcal{A}_V, \mathcal{A}_E, \tau_v, \tau_t, \kappa, \prec)$$

其中：

- V：有限顶点集（实体）
- $\mathcal{E} \subseteq 2^{V \cup \mathcal{E}} \setminus \{\emptyset\}$：超边（可包含其他超边）
- $\mathcal{A}_V$： $V \to \mathcal{P}$：顶点属性函数
- $\mathcal{A}_E$：$\mathcal{E} \to \mathcal{P}$：超边属性函数
- τ_v： $\mathcal{E} \to \mathcal{I}$：有效时间赋值
- τ_t： $\mathcal{E} \to \mathcal{I}$：事务时间赋值
- κ：$\mathcal{E} \to [0,1]$：置信度赋值
- $\prec \subseteq \mathcal{E} \times \mathcal{E}$：因果偏序

4.2 四大支柱：形式化定义

定义 4.2 (**P1**：带属性的 N 元关系)。 系统 $\mathcal{K}$ 满足 ***P1***，当且仅当对于任意$n \geq 2$ 以及任意属性集 A，$\mathcal{K}$ 都能表示一种恰好连接 n 个实体、并具有属性 A 的关系。实体域在关系构造下封闭：$\mathcal{K}$能够表示的任何关系本身都是一个实体，可以参与其他关系。

注释 4.3 (关系构造闭包)。 定义 *4.2* 中的闭包要求并非可有可无的附加条件，而是将关系一视同仁的自然结果。一个系统若能基于任意实体构造关系，却将关系排除在实体域之外，就施加了特设的类型级限制：系统中的某些对象是一等参与者，另一些则不是。这正是注释 *4.8* 所指出的非统一性，也是特设（而非系统性）关系支持的标志。闭包要求确保“任意 n 个实体”这一量词确实涵盖所有可表示的对象。

定义 4.4（**P2**：时间有效性）。 系统 $\mathcal{K}$ 满足 ***P2*** 当且仅当每个关系 r 都具有关联的有效时间区间 $\tau_v(r)$，$\mathcal{K}$ 支持时间旅行查询，并且保留历史记录（无破坏性更新）。

定义 4.5（**P3**：不确定性量化）。 系统 $\mathcal{K}$ 满足 ***P3*** 当且仅当每个关系r 都可以携带置信度值 $\kappa(r) \in [0,1]$，置信度按照定义明确的规则通过推理传播，并且每个置信度值的证据基础（支持性或削弱性证据以及评估方法）本身是可表示且可查询的。

定义 4.6 (**P4**：因果关系)。 系统 $\mathcal{K}$ 满足 ***P4*** 当且仅当关系可以参与其他关系，并且对于任意 r_1、r_2，都能表示“r_1 导致了 r_2”。

4.3 一等对象：操作性定义

De义 4.7 (一等对象)。 当且仅当 $\mathcal{K}$ 提供以下内容时，关系 r 在其中是一等对象：

1. 存储： r 可分配持久标识符 *2.* 检索： r 可仅凭编号获取 *3.* 查询： r 可作为查询谓词的主体 *4.* 组合： r 可参与其他关系 *5.* 返回：查询可将 r 作为一等结果返回

备注 4.8 (系统化与临时性一等状态)。 定义 *4.7* 要求，系统中的所有关系都通过单一机制以统一 的方式支持这五种操作，而不是针对每种类型采用变通方案。带有代理键的 *SQL* 关联表，对于某一种特定关系类型而言，满足存储 和查询 要求，但在三个方面不满足系统化的一等地位：*(1)* 跨类型组合失败——当处方和反应分别存储在不同的表中时，若不使用会放弃类型安全的多态引用，就无法表达“处方 *42* 导致了反应 *17*”。*(2)* 跨类型查询谓词的查询 失败——“查找涉及实体 *X* 的所有关系”需要逐一枚举每个关联表。*(3)* 动态元数下的存储失败——添加参与者需要修改模式。两者的区别在于：一次处理一个表，且每个表都采用定制模式 （特设的），与为所有关系提供统一 API，无论其类型或元数如何 （系统化）。我们的定理要求的是后者。

5 相关工作

5.1 因果推理：Pearl 的 Do演算

Pearl 关于因果关系的研究 [8] 确立了基于随机变量有向无环图（DAG）的干预推理 Do演算。我们的框架与 Pearl 的工作有三点不同。第一，范围：Pearl 着重于从观测数据中推断 因果效应；我们的框架则着重于将已知因果关系作为一等数据库对象存储和查询 。第二，结构：Pearl 的因果有向无环图连接变量；我们的因果排序 $\prec$连接超边（关系），从而能够表达元因果陈述，例如“这一处方关系导致了那一不良反应关系”。第三，不确定性语义：Pearl 的框架使用条件概率分布；我们的框架则为每种关系使用置信度分数，并配有明确的传播规则（定理 7.5）。

这些方法是互补的，而非相互竞争。我们的框架可以持久化因果发现算法的输出，为学得的因果结构提供可查询性和时间版本管理。反过来，Pearl 的推理机制可以在 ATCH 存储的关系上运行，从而推导出存储层无法原生计算的干预性结论。

5.2 概率数据库

MayBMS [10], Trio 和 ProbLog 等系统率先将不确定性引入数据库，采用可能世界语义：概率数据库表示一组确定性数据库实例上的概率分布，而查询求值则对可能世界进行边缘化。这种方法建模了存在性不确定性——“哪些元组存在？”——并支持有原则的概率推断。

我们的框架模拟的是另一种不确定性：认知性 置信度——“我们有多确定这种关系成立？”每种关系都带有一个标量 $\kappa \in [0, 1]$ ，反映系统的信念程度，而非可能世界上的边缘概率。关键差异

源于这一区别。概率数据库通常需要元组独立性（或有界相关性），才能实现可处理的查询求值；在一般情况下，其查询复杂度为 #P-完全 [10]。我们的框架不要求独立性：置信度分数是各个关系的属性，沿因果链的传播（定理 7.5）可在多项式时间内计算。

两种方法互不包含。对于需要对组合不确定性进行边缘化的查询（例如“节点 A 和 B 相连的概率是多少？”），可能世界语义的表达能力更强。对于结合置信度、时间有效性和因果结构的查询（例如“是什么导致了这种关系？我们对因果链中的每条边有多大把握？”），我们的框架表达能力更强。将可能世界推理应用于 ATCH 存储的关系，是一个很有前景的方向。

5.3 时序数据库

TSQL2 项目 [23]以及 Snodgrass 后续的工作 [9] 确立了双时态模型，其基础是 Allen 区间代数 [1]，而 SQL:2011 部分采用了这一代数。我们的框架采用了这些语义（τ_v、τ_t），但将时间属性扩展到了超边，而不只是元组。这样便能表达“时间 t的因果结构是什么？”之类的查询，而 SQL:2011 无法表达此类查询。

5.4 默认与非单调推理

Reiter 默认逻辑 [7] 使用以下形式的默认规则对默认值和例外进行形式化推理：$\frac{\alpha : \beta}{\gamma}$（“如果 α 被认为成立，且 β 保持一致，则得出 γ”）。我们的框架通过不同的机制处理同样的现象——带有例外的默认值、常识推理：默认值是置信度较高但小于 1 的关系（$\kappa < 1$），而例外则是因果抑制因素或置信度更高、可覆盖默认值的特定关系（第 7.4 节）。这样可以避免默认逻辑中出现的多重扩展问题，因为冲突是通过置信度与特异性层级（第 7.3 节）解决的，而不是通过在最大一致扩展之间进行选择来解决。

5.5 情境演算与事件演算

麦卡锡和海斯 [6] 提出了情境演算，Reiter [14]，随后将其形式化为一种用于对动态世界进行推理的一阶逻辑框架。情境是动作序列；流变项则是在不同情境中变化的属性。情境演算通过后继状态公理，为框架问题提供了简洁的解决方案，并已扩展出概率变体 [16]。Kowalski 和 Sergot [15]，提出的事件演算采取了互补的方法：事件会随时间启动和终止流变项，无需显式的情境项即可实现原生时间推理。

ATCH 与这两种形式体系的区别在于：它们解决的是推理 问题（对未来状态可以推断出什么？），而 ATCH 解决的是表示 问题（哪些内容可以作为数据库存储和查询？）。情境演算中的流变项对应于我们的有时间界限的关系，但流变项是逻辑项，而非具有身份的可查询对象——无法询问“返回在时间 t 为真且置信度高于 0.8 的所有流变项”。事件演算中的事件可以嵌入为带有时间界限的 ATCH 超边，但事件演算原生不支持带属性的 n 元事件（P1）、不确定性量化（P3）以及关于事件的事件（P4）。ATCH 可视为在这些推理形式体系之下提供了一个数据库层：将事件和流变项存储为一等关系，然后基于存储的表示应用情境演算或事件演算进行推理。

5.6 描述逻辑与时间扩展

描述逻辑（DL） [17] 为 OWL 系列本体语言奠定基础，并提供一阶逻辑中具有良好特征复杂度的可判定片段。诸如 $\mathcal{ALCQI}$ 带有时态算子的 [18]时序扩展，以及用于时序本体数据访问的 DL-Lite 系列 [19] ，都将时间引入描述逻辑推理。这些形式体系擅长术语推理（类层次结构、角色限制），并提供严格的复杂度界。

然而，描述逻辑从根本上说是二元的：角色恰好连接两个个体。N元关系需要采用再化模式，这会失去描述逻辑框架的结构性保障 [17]。时态描述逻辑为角色断言添加了时间，却没有为再化的N元结构添加时间。概率描述逻辑 [20], 处理了不确定性，但它们处理的是概率类成员关系，而非带有因果传播的逐关系置信度。ATCH 处理的是另一个层面：描述逻辑提供本体推理（存在哪些类，以及它们之间有何关系？），而 ATCH 提供断言存储（成立的特定关系是什么、何时成立、置信度如何，以及原因是什么？）。

5.7 超图数据库与理论

Berge 的奠基性工作 [21] 确立了超图作为图的自然推广形式，适用于表示多实体关系。在数据库领域，Abiteboul 和 Hull [22]研究了允许嵌套结构和值为集合的结构的复杂对象模型，提供了 ATCH 所需的部分表达机制。近年来，基于超图的知识表示受到人工智能社区的关注：HyperGraphRAG [13] 表明，在检索增强生成中，超图结构化检索的表现优于二元知识图谱。

近期在结构层面最具雄心的形式体系是超关系时态知识广义超图 (HTKGH) [27], ，它将超关系时态知识图谱扩展为支持任意数量的主要实体，以及带有时态注释的二阶边。HTKGH 实现了四大支柱中的三项——N元结构（P1）、时间有效性（P2）和部分元关系（P4）——但缺少不确定性量化（P3）以及关系之间的因果排序。它独立验证了 ATCH 框架大部分内容的必要性，同时也展示了等价定理所弥补的缺口：若不具备全部四大支柱，该表示仍不完整。

ATCH 在此研究脉络上进一步扩展，通过为超图增添结构完备表示所需的能力：时间界限、置信度分数，以及超边之间的因果排序。若没有这些扩展，普通超图只能满足 P1，而无法满足 P2–P4。等价定理证明，这些扩展并非可有可无的附加项，而是将关系视为一等对象所必然带来的结果。

5.8 数据模型中的一等关系

一等关系地位这一概念已从多个角度得到探讨，但尚未正式统一。Sadoughi、Yakovets 和 Fletcher [28] 提出了元属性图，通过将标签和属性作为可查询对象进行一等处理，扩展了 ISO属性图模型，并支持将子结构（子图、边、属性集）具体化为节点。他们明确将具体化描述为“让某物成为一等公民”。然而，元属性图并未将一等地位与时间、因果或不确定性要求联系起来——它在操作层面阐明了什么是一等对象，却没有证明为什么由此必然具备特定能力。等价定理填补了这一空白：它证明，操作层面的一等地位概念（定义 4.7）必然蕴含四大支柱，反之亦然。

6 等价定理

6.1 形式化预备知识

定义 6.1（属性附着）。 当且仅当满足以下条件时，属性 p 附着于系统 $\mathcal{K}$ 中的对象 o：

1. $\mathcal{K}$ 为这对 （o、 p）提供存储。*2.* $\mathcal{K}$ 支持以下形式的查询：“返回 p，针对 o”*3.* 该属性是 o的内在 特性，无法从 o的组成部分推导得出

引理 6.2（二元分解失败）。 对于任何具有 $n \geq 3$ 元关系和属性 a 的关系 r，将 r 分解为二元关系会丢失 a的连接点。

证明。 设 $r = \{e_1, e_2, e_3\}$ 具有属性 a。二元分解会产生以下边：

$$\{(e_1, e_2),\ (e_2, e_3),\ (e_1, e_3)\}.$$

a 连接在哪里?

情况 1： 对于一条边 $\Rightarrow$， r 中其他边的参与信息会丢失。

情况 2： 对于所有边 $\Rightarrow$，会产生冗余和更新异常。

情况 3： 对于以 (e_1， e_2， e_3) 为键的独立表 $\Rightarrow$，这就是 具体化。表中的行将该关系表示为具有自身身份的对象，从而满足存储和查询的要求。

“独立表”方案并非一等地位的替代方案；它实现了 一等地位。任何无损解决方案都要求 r 具有一等地位。 □ □

6.2 主定理

定理 6.3（等价定理）。 对于任意知识表示系统 $\mathcal{K}$，以下各项等价：

$$(\boldsymbol{P1})\ \textit{N-ary relationships with attributes} \tag{1}$$
$$(\boldsymbol{P2})\ \textit{Temporal validity} \tag{2}$$
$$(\boldsymbol{P3})\ \textit{Uncertainty quantification} \tag{3}$$
$$(\boldsymbol{P4})\ \textit{Causal relationships between relationships} \tag{4}$$
$$(FC)\ \textit{Relationships are first-class objects} \tag{5}$$

证明。 我们证明：一等公民 $\Rightarrow$ **P1..4**，且每个 **Pi** $\Rightarrow$ 一等公民。

引理 1： 一等公民 $\Rightarrow$**(P1**$\wedge$**P2**$\wedge$**P3**$\wedge$**P4)**。若关系是一等公民（定义 4.7），则存储+查询支持属性（**P1**）、按时间索引的检索（**P2**）、将置信度作为属性（**P3**），而组合支持元关系（**P4**）。 □

引理 2：P1 $\Rightarrow$ 一等公民。设 r 为 n 元关系，具有 $n \geq 3$ 和属性 a。根据引理 6.2，a 必须关联到其自身。这要求 r 可存储（存储（r， a））且可查询（给定 r检索 a ）。由此可得返回：如果 r 可以成为查询谓词（“返回 $a > x$的关系”）的主体，那么结果集必须包含作为可识别对象的 r，否则该谓词便毫无意义。至于组合：根据定义 4.2 的闭包性质，实体域包含所有可表示的关系。由于 r 可存储（如上所述）， r 属于实体域。因此，P1 允许构建一种新关系，其参与者包括r——例如，具有属性 {的 $\{r_1, r_2$,机制} }。这正是组合：关系参与其他关系。闭包性质（注释 4.3）是关键桥梁：没有它，系统可以支持基础实体之间的 n 元关系，却排除关系参与其中；这正是系统性的一等地位所不允许的特设类型限制。 □

引理 3：P2 $\Rightarrow$一等公民。时间有效性 τ_v（r）是 r的真值随时间变化的属性。它不能附属于端点（端点的生命周期各不相同），也不能无所附属（它必须被存储）。根据定义 6.1，r 必须可存储且可查询。关键检验是时间身份：当同一现实关系在时间区间 $[t_1, t_2]$ 和 $[t_3, t_4]$ 内成立时（例如，某员工调离后又回来），不具备一等关系的系统会将其表示为两条独立的行，且没有共同的指称对象。关于该关系时间历史的查询（“这一关系曾成立过多少次？”）会变成针对隐式分组的聚合查询，而不是对已识别对象进行属性查找。请注意，P2 明确要求支持时间旅行查询并保留历史（不得进行破坏性更新）；这些要求超出了使用复合键的行版本控制，因为保留历史要求能够将该关系完整的时间轨迹重构为单个已识别对象的属性。一旦时间历史可查询，该关系就必须具有持久身份——这就是存储。时间旅行查询则要求返回。至于组合：P2 要求保留历史并支持时间旅行查询。考虑以下查询：“显示所有在1月1日有效、但到3月1日已终止的关系，以及导致各关系终止的事件。”此查询询问的是元关系：它要查找关系 r、其时间边界以及改变这些边界的事件之间的联系。无法将 r 表示为此类元关系参与者的系统无法回答该查询；它只能列出时间戳匹配的行，却会丢失关系、其时间轨迹与推动该轨迹变化的事件之间的结构联系。由于 P2 要求针对保留的历史进行时间旅行查询，而此类查询本质上涉及以 r 为参与者的元关系，因此组合是必然的。因此，r 具有一等地位。 □

引理 4：P3 $\Rightarrow$ 一等公民。置信度 $\kappa(r)$ 是一种元属性——它描述的是 r 的真值状态，而非关于参与实体的事实。在关联表上设置置信度列在语法上是可行的，但请考虑：如果同一关系被多次断言，且置信度值各不相同（例如，一个传感器报告 0.6，另一个报告 0.9），那么哪一行才“是”该关系？系统必须维护一个置信度会更新的单一关系对象（这要求存储 $=$ 具有持久身份），或者将每一行视为独立记录（从而失去对同一关系的各项评估之间的联系）。此外，基于置信度的查询（“查找 $\kappa > 0.9$ 的关系”）需要查询。对于组合：定义 4.5 要求，每个置信度值的证据基础——支持或削弱该值的证据以及评估方法——本身必须可表示、可查询。证据基础是被评估关系 r、证据与方法之间的一种关系：例如，“传感器 s_1 使用方法 m 评估了关系 r，并报告 $\kappa = 0.9$。”这种证据关系的参与者包括 r 本身。因此，要表示这种关系，就要求 r 参与另一种关系，这正是组合的含义。这并非偶然的设计选择，而是定义 4.5 的要求：如果证据基础不可表示，置信度值就只是含义不明的浮点数，无法以有原则的方式解决冲突、整合证据或通过推理传播。返回由查询推出，理由与引理 2 相同。 □

引理 5：P4 $\Rightarrow$ 一等公民。如果“r_1 导致 r_2”可以表示，那么 r_1，r_2 就必须能作为参数被引用。可引用的 $\Leftrightarrow$ 持久标识符 $\Leftrightarrow$ 存储。参与因果关系$\Leftrightarrow$ 组合。这正是一等公民的特性。 □

根据传递性：**P1** $\Rightarrow$ 一等公民 $\Rightarrow$**P2**，等等。所有条件彼此等价。 □ □

推论 6.4（全有或全无）。 缺少任何一个支柱的系统，都无法在没有变通方案的情况下满足其他任何支柱；而这些变通方案实际上实现了缺失的能力。

相互蕴涵结构如图 3 所示。推论 6.4 具有直接的实践意义：这意味着，除非一次添加就能同时实现全部四大支柱，否则逐步向现有系统添加支柱是徒劳的。

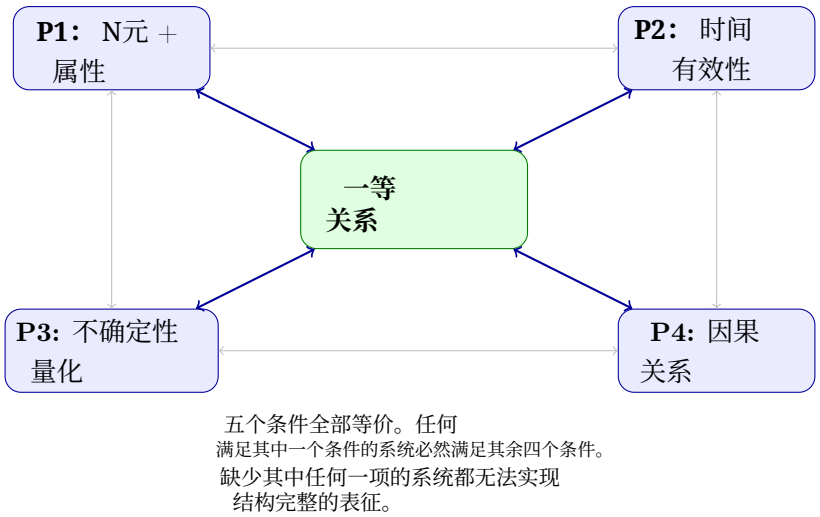

图 3：等价定理：一等关系地位与四大支柱之间相互蕴涵。每一项支柱单独都蕴涵一等地位，而一等地位反过来蕴涵全部四大支柱。

7 经典人工智能问题的应用

等价定理确立了一等关系蕴涵全部四大支柱。由此自然会产生一个问题：除了表征完备性，这四大支柱还能带来什么？本节将展示，四大支柱结合起来，能够自然地解决四个经典人工智能问题——框架问题、冲突解决、常识推理和不确定性传播；在缺少一个或多个支柱的系统中，这些问题一直难以得到简洁的解决方案。各小节分别讨论一个问题；贯穿其中的主线是，每种解决方案都直接源自支柱之间的相互作用，而非特设的扩展。时间持久性解决了框架问题（定理 7.1）；第 7.2 节将因果链中的不确定性传播形式化；隐变量发现（定理 7.17）则实现了自动化的冲突解决。

7.1 框架问题

框架问题是经典人工智能面临的一项挑战：如何表征动作发生时哪些事物不会发生变化。在传统逻辑中，要陈述“Bob 重启了虚拟专用网络”，还必须说明办公室地址没有变化、天空依然是蓝色等——这需要一组无限的“框架公理”。麦卡锡和海斯 [6]提出了这个问题；经典解决方案包括 STRIPS 式的显式框架公理、情境演算和流演算。每种方案都要求知识工程师针对每个动作，明确指定哪些事物保持不变——这项负担会以组合方式增长。

定理 7.1 (默认持久性)。 在我们的框架中，在时间 t_1创建的关系 r 会一直有效，直到被明确终止。无需框架公理。

证明。 根据定义 4.1，每条超边 e 都有一个有效时间区间 $\tau_v(e) = [t_{start}, t_{end}]$。如果$t_{end} = \infty$ （或为 NULL），则 e 当前有效。

当某个操作创建新关系 e时，只添加 e' 。现有关系 e_1， e_2，.....及其 $t_{end} = \infty$ 保持不变。

每个 e_i 的“框架公理”都是隐含的： e_i 之所以持续存在，是因为没有发生终止事件。□

□

定义 7.2（因果影响范围）。 当因果事件 e_c 发生时，只有 e_c 的因果闭包 中的关系可能受到影响：

$$\mathit{Affected}(e_c) = \{e : e_c \prec^* e\} \cup \{e : e \prec^* e_c\}$$

其中 $\prec^*$ 是 $\prec$ 的传递闭包。

推论 7.3（自动隔离）。 不在受影响（e_c） 中的关系经证明不会因 e_c而改变，无需显式的框架公理。

相较于以往方法，其关键优势在于持久性是存储模型的默认行为，而不是额外添加的推理规则。传统数据库基于快照（要了解 T_2 时的状态，需要一幅完整的新图景）；该框架基于增量（仅记录变化，未受影响的关系会自动持久存在）。

7.2 不确定性传播

通过因果联系进行推理时，三个不确定性分量会合并：

定义 7.4（链接不确定性分量）。 对于因果联系 $e_1 \prec e_2$：

- $\kappa(e_1)$：原因 e_1 发生的置信度
- $\kappa_\prec(e_1, e_2)$：对因果机制的置信度
- $\kappa(e_2|e_1)$：给定 e_1时的条件置信度

定理 7.5 (不确定性传播**)**。对于因果链 $e_1 \prec e_2 \prec \cdots \prec e_n$，联合置信度为：

$$\kappa_{chain} = \kappa(e_1) \cdot \prod_{i=1}^{n-1} \kappa_\prec(e_i, e_{i+1}) \cdot \mathit{Context}(e_i, e_{i+1})$$

其中，上下文(e_i, e_{i+1}) 是基于 *N*元属性的修正项。

示例 7.6 (上下文修正**)**。考虑：电涌 $\xrightarrow{0.9}$ 电源故障 $\xrightarrow{0.8}$ 主板短路。

如果 *PSU* 具有属性 *surge_protector* = *true*，上下文修饰符会降低第二个链接的有效置信度：

$$\kappa_{effective} = 0.8 \times (1 - 0.7) = 0.24$$

当多条因果路径导向同一结论时：

定理 7.7（路径积分）。 给定通往结论 e 的路径 P_1、P_2、…、P_k，其置信度分别为 κ_1、…、κ_k：

$$\kappa(e) = 1 - \prod_{i=1}^{k} (1 - \kappa_i)$$

假设各路径相互独立（噪声或组合）。

证明概要。 每条路径 P_i 都会独立地以概率 κ_i产生 e，因此路径 P_i 不 产生 e 的概率为（$1 - \kappa_i$）。根据独立性，没有 任何路径产生e 的概率为 $\prod_{i=1}^{k}$（$1 - \kappa_i$）。取补集可得 κ（e） $= 1 - \prod_{i=1}^{k}$（$1 - \kappa_i$），即噪声或公式。独立性假设将在下文的注释中讨论；若该假设不成立，则可采用保守的替代方案。 □

注释 7.8（路径独立性假设）。 *Noisy-OR*组合规则假设路径之间相互独立，即不同路径上的因果机制不存在共同原因。当多条路径共享上游原因时（例如，同一根事件导致的两个结果），独立性便不成立，合并后的置信度可能被高估。在这种情况下，可采用其他组合规则：最大置信度规则（$\kappa(e) = \text{最大值}_i\, \kappa_i$）可提供保守界限，而概率图模型中的连接树方法则可在依赖结构已知时计算精确的联合置信度。该框架会存储足够的信息（完整的因果图），以便检测共享祖先并自动选择适当的组合规则。

示例 7.9. 路径 *A*：硬件故障 $\rightarrow$ 离线（置信度 *65%*）路径 *B*：计划维护 $\rightarrow$离线（置信度 *20%*）综合： $1-(1-0.65)(1-0.20) = 72\%$ 置信度 系统处于离线状态。

7.3 冲突解决

在传统数据库中，矛盾是错误。在我们的框架中，矛盾是尚未解决的元数据。

7.3.1 非破坏性存储

原则 7.10 (逻辑共存)。 如果关系 R_1 断言 A，且关系 R_2 断言$\neg A$，那么二者会同时存在于超图中。冲突解决会使用关系属性。

7.3.2 解决层级

定义 7.11（冲突解决顺序）。 给定相互冲突的关系 R_1、R_2:

1. 时间：其中一个是否更新？*(*τ_v（R_2）$.start > \tau_v$（R_1）$.start$*) 2.* 置信度：其中一个的 κ 是否更高？ *3.* 来源：基于属性的优先级（传感器 >手动录入） *4.* 特异性：更具体的上下文优先

7.3.3 上下文坍缩原理

原理 7.12（隐藏上下文）。 如果关系 R_1 和 R_2 分别在同一系统内断言 A 和 $\neg A$，且二者的置信度均非零，那么在假设系统不包含真正相互矛盾的基本事实的前提下，存在一个上下文变量 z，使二者均有条件地为真：

$$R_1 : A \ given\ z = z_1 \qquad R_2 : \neg A \ given\ z = z_2$$

论证。 如果 A 和 $\neg A$ 均基于观测以非零置信度成立，那么支持二者的观测必然是在不同条件下生成的。令 z 表示区分这些观测上下文的所有条件的合取。则 $A|_{z=z_1}$ 和$\neg A|_{z=z_2}$ 均是一致的，而将隐藏上下文显式表示为相应关系的 N元属性，即可解决表面上的矛盾。该原理为依赖上下文的真值提供了表示能力； 适当上下文变量 z 的发现 由领域专家完成，或由第 7.3.5 节中的自动化程序执行。 □

示例 7.13. 矛盾：“该药物有效”与“该药物无效”。解决方案：“该药物对 *X*组有效”，而“对 *Y*组无效”。*N*元属性（患者群体）消除了表面上的矛盾。

示例 7.14（基础设施监控）。 传感器 *A* 在同一时间戳报告“服务器离线”（$\kappa = 0.95$），而传感器 *B* 报告“服务器在线”（$\kappa = 0.92$）。在标准监控系统中，这是一个需要人工分诊的矛盾。在 *ATCH* 中，两种关系都带有 *N*元属性 *network_segment*：传感器 *A* 从网段 *1* 进行观测（该网段的防火墙规则阻止 *ICMP*），传感器 *B* 则从网段 *2* 进行观测（可直接连接）。隐藏上下文变量是 *network_segment*；两种读数在各自的上下文中都正确，因此无需舍弃任何一项观测，表面上的矛盾便迎刃而解。

7.3.4 因果审计

发生冲突时，我们的框架可以追溯其因果起源：

算法：因果冲突消解

1. 给定： 冲突关系 R_1， $R_2$2. $C_1 \leftarrow$ CausalChain(R_1) 3. $C_2 \leftarrow$ CausalChain(R_2) 4. **如果** $\exists e \in C_1$ 的 κ 较低（故障源）：

- 创建解释：“e 导致了错误的 信念 R_1”
- 首选 R_2

7.3.5 自动隐藏变量发现

上下文坍缩原理（原理 7.12）保证：当相互矛盾的关系同时累积置信度时，必然存在一个隐藏上下文变量。本小节将框架从其自身的存储属性中发现该变量的机械过程形式化。

定义 7.15（矛盾信号）。 当两个关系簇 $\mathcal{C}_R = \{e : \ e$ 支持 R、$\kappa(e) \ > \theta\}$ 和 $\mathcal{C}_{\neg R} = \{e : \ e$ 支持 $\neg R$、$\kappa(e) \ > \theta\}$ 在 *Noisy-OR* 累积后均超过置信度阈值 θ 时，就会触发矛盾信号。

当矛盾信号触发时，通过检查每个簇中关系所附加的 N元属性（P1），即可识别隐藏变量。

定义 7.16（属性划分质量）。 对于取值来自域$\mathcal{A}$的属性 a，以及矛盾簇 $\mathcal{C}_R$、 $\mathcal{C}_{\neg R}$，将 a 的划分质量 定义为信息增益：

$$IG(a) = H(\mathcal{C}) - \sum_{v \in \mathcal{A}} \frac{|\mathcal{C}_{a=v}|}{|\mathcal{C}|} H(\mathcal{C}_{a=v})$$

其中， $\mathcal{C} = \mathcal{C}_R \cup \mathcal{C}_{\neg R}$、 $\mathcal{C}_{a=v}$ 是属性 a 取值为 v时的观测子集，而 H是每个子集内类别标签（R 与 $\neg R$）的香农熵。

定理 7.17（隐藏变量识别）。 给定 $\mathcal{C}_R$ 与$\mathcal{C}_{\neg R}$之间的矛盾信号，原则 *7.12* 中的隐藏上下文变量 z 对应于使划分质量最大化的属性 a^*：

$$a^* = \arg\max_{a \in \mathcal{A}_E} IG(a)$$

其中，$\mathcal{A}_E$ 是附加到 $\mathcal{C}_R \cup \mathcal{C}_{\neg R}$中各超边上的所有属性的集合。

证明。 根据上下文坍缩原理，存在一个变量 z，使得 $R|_{z=z_1}$ 和 $\neg R|_{z=z_2}$均保持一致。能够完美分离各簇的变量 z 实现 $H(\mathcal{C}_{z=z_1}) = H(\mathcal{C}_{z=z_2}) = 0$，从而使 $\mathrm{IG}(z) = H(\mathcal{C})$ 最大化。在实践中，使其最大化的属性 a^* 是利用存储属性所得的 z 的最佳经验近似。如果没有任何单一属性能够实现完美分离，则隐藏上下文可能是多个属性的合取，可通过递归划分发现（即在属性空间上构建决策树）。 □ □

示例 7.18（实践中的隐变量发现）。 有 *40* 张支持工单涉及 *Windows* 更新 *KB5034763*。其中 *20* 张报告了打印失败，另外 *20* 张则报告打印成功。两个簇通过 *Noisy-OR* 都达到了较高的置信度。矛盾信号触发了。

检查 *N* 元属性时，属性 *driver_version* 实现了完美区分：所有失败都对应版本 *6.1*，所有成功都对应版本 *8.3*。系统将原本矛盾的关系拆分为两个特定于上下文且置信度较高的关系：

$$e_1 : \{KB5034763, PrintFailure\} \ with \ \texttt{driver_version} = 6.1, \quad \kappa = 0.997$$
$$e_2 : \{KB5034763, PrintSuccess\} \ with \ \texttt{driver_version} = 8.3, \quad \kappa = 0.991$$

矛盾消解为两个特定于上下文且置信度较高的关系。未来的研究方向之一，是在各种不同的数据集上对这一机制进行实证验证。

关键的促成因素是支柱 1。由于 N 元关系本身就包含丰富的属性，解释变量总是与冲突共置。在缺少 P1 的系统中——例如二元图、传统关系表——隐变量可能存在于数据的某处，但它并未附加到关系本身，因此若不通过临时连接或外部分析，就无法从结构上实现自动发现。

备注 7.19（计算成本）。 对 N 条观测和 m 个属性进行隐变量发现，每个矛盾信号都需要 $O(N \cdot m)$ 次计算——即遍历一次属性矩阵。

7.4 常识推理

常识推理——处理带有例外的默认值——出了名地困难。“鸟类会飞”必须考虑企鹅、受伤的鸟、笼中鸟等情况。在经典逻辑中，“所有鸟都会飞”遇到第一只企鹅时就不成立；限定条件的列表是无穷无尽的。

定义 7.20（上下文关系）。 我们不再表示为 *Bird*（x）$\rightarrow$*Flies*（x），而是表示为：

$$R_{fly} : \{x, flight\} \ with \ attributes \ \{species, health, environment\}$$

遇到企鹅时，我们不会删除飞行关系；我们会观察到，其物种属性值（$Spheniscidae$）会停用默认值。

7.4.1 通过不确定性实现默认逻辑

定义 7.21 (默认推理)。 • 一般规则： $R_{default}$：鸟 $\rightarrow$ 会飞，置信度为 κ =0。95

- 特定事实： $R_{specific}$：翠儿 $\rightarrow$ 不会_飞，置信度为 $\kappa = 1.0$
  系统会进行特异性检查：置信度较高的特定事实会覆盖置信度较低的默认值。

7.4.2 抑制关系

定义 **n 7.22 (**因果抑制)。 关系 $R_{inhibit}$ 可以抑制关系 R_{target}:

$$R_{broken_wing} \xrightarrow{inhibits} R_{can_fly}$$

这刻画了为什么：不仅是翠儿无法飞行，更是因为断裂的翅膀当前正在妨碍其飞行能力。当翅膀痊愈时（时间变化），抑制因素便会失效，飞行能力也会自动恢复。

定理 7.23 (限定完备性)。任何针对默认关系 R 的常识性例外，都可以表示为以下一种或多种情况:

1. 修改该关系的 *N*元属性（***P1***） *2.* 限制有效性的时间边界（***P2***） *3.* 降低置信度的不确定性降低（***P3***） *4.* 抑制默认关系的因果抑制因素（***P4***）

证明。针对默认关系 R 的例外，是一条知识，断言在某些条件下 R 不成立。我们通过穷举所有情况证明，每个此类例外至少会沿四个结构维度之一修改 R 。

例外必须指明什么对 R 而言不成立。任何关系的结构方面包括：（i）其参与者及其角色，（ii）其时间范围，（iii）其真值或信念程度，以及（iv）其因果背景。我们认为这些方面是穷尽的，因为关系作为一种知识主张，断言某些实体在某个时期内基于某些理由以某种置信度处于某种关系中。例外必须否定或限制这些方面中的至少一个:

情形 1: 该例外限制适用的实体或上下文 R（例如，“鸟类会飞，但企鹅除外”）。这改变了参与者／上下文结构——一种划分论域的 N元属性（**P1**）。

情形 2: 该例外限制何时 R 成立（例如，“鸟类会飞，但换羽季除外”）。这是一个时间边界（**P2**）。

情形 3: 该例外降低了对 R 的信念程度（例如，“鸟类通常会飞”，并带有 $\kappa < 1$，以概率方式容纳例外）。这是一种不确定性降低（**P3**）。

情况 4: 该例外指出一种起覆盖作用的机制 R （例如，“折断的翅膀会妨碍飞行”）。这是一种因果抑制因素（**P4**）。

我们通过考察可能看起来不属于情况1–4的例外类型，来验证其穷尽性。模态例外（“鸟类可以飞行”与“鸟类应该飞行”）会改变关系的模态，而模态是关系的一种属性——即参与者/上下文限制（情况 1），其中该属性将模态上下文划分开来。反事实例外（“如果这只鸟的翅膀折断了，它就不能飞”）断言一种因果机制（折断的翅膀会妨碍飞行），并以假设状态为条件——因果抑制因素属于情况 4，而假设性条件则是一种属性（情况 1）。道义例外（“鸟类获准飞行，但限制空域除外”）依据规范语境限制适用性，这同样是一种参与者/上下文属性（情况 1）。

更一般地说，任何不属于情况1–4的例外，都必须修改 R 中除参与者、时间范围、置信度或因果背景之外的某个方面。但这四个方面——谁/什么、何时、确定程度以及为什么/为什么不——穷尽了知识主张的结构维度。不修改这些方面中的任何一个的例外实际上并未施加约束 R ，因此也就不是例外。 □

□

8 计算复杂性

8.1 困难性结果

定理 8.1 （难解性）。*1.* 一般超图模式匹配：*NP* 完全

2. 无界因果链查询：*PSPACE*完全 *3.* 最优整合：*NP*困难

证明概要。 **(1)** 通过子图同构进行归约：给定模式超图 P 和数据超图 D，寻找嵌入 $P \hookrightarrow D$ 将普通图上的 NP 完全子图同构问题推广了。**(2)** 无界因果链需要探索穿过 $\prec$的任意长度路径；带有时间约束和置信度约束的配置可达性问题可从量化布尔公式可满足性归约而来。**(3)** 最优整合——在时间约束下选择置信度最高的一致超边子集——可从加权MAX-SAT归约而来。 □

□

8.2 可处理片段

定理 8.2 (可处理片段**)**。 以下查询类别属于多项式时间：

- α-无环超边模式
- 固定深度因果链（d为常数）
- 有界时间窗口
- 置信度阈值

证明概要。 α-无环超边模式是数据库可处理性中的标准无环性概念， [21], 可通过 Yannakakis 式连接树求值在 $O(n \cdot |E|)$ 时间内处理，其中 n 是模式大小， $|E|$ 是超边数量。固定深度因果链最多需要沿$\prec$进行 d次连接；每次连接都可通过时间范围包含（利用索引支持的区间查询）和置信度阈值（简单的算术比较）进行筛选。有界时间窗口通过对 τ_v应用范围谓词来限制工作集，并由标准 B 树或 GiST 索引提供支持。这些多项式时间操作的组合仍为多项式时间。 □ □

大多数实际查询都满足这些约束，因此我们的框架能够高效处理实际工作负载。

注释 8.3 (谓词下推**)**。 定理 *8.2* 中的多项式界假设时间谓词和置信度谓词在结构连接之前进行计算，而非之后。实际中，这通过谓词下推来实现：由于 τ_v 通过 *GiST* 建立索引，且 κ 是标量列，查询规划器会在任何结构模式匹配开始之前，于连接树的叶级积极筛选工作集。如果不进行下推，即使对于 α-无环模式，在高度连接的超图中，*Yannakakis* 求值期间的中间工作集也可能变得很大；进行下推后，每个连接步骤中的工作集大小都受时间窗口和置信度阈值的选择性限制。

注释 8.4 (置信度驱动的深度终止)。 定理 *8.2* 中的固定深度约束（d 为常量）看起来或许过于严格：在根因分析中，用户很少能预先知道确切的因果深度。然而，不确定性传播规则（定理 *7.5*）提供了自然的终止准则。由于联合置信度沿因果链呈乘法衰减——$\kappa_{chain} = \kappa(e_1) \cdot \prod_i \kappa_{\prec}(e_i, e_{i+1})$——累积置信度最终会低于任意相关性阈值 θ。当 $\kappa_{chain} < \theta$时，遍历引擎可以动态终止，从而将有效深度限制在 $d_{eff} \leq \lceil \log_{\kappa_{min}} \theta \rceil$ ，其中 κ_{min} 为最小单链路置信度。这样，深度界由数据驱动，而非由用户指定，同时仍能保证多项式时间复杂度。

9 原型实现

为证明理论框架能够得到实际实现，我们以 C 语言编写了 ATCH 模型的原型，并将其实现为 PostgreSQL 扩展。该原型集成了数据库引擎的查询执行、索引和类型系统，并验证了第 8 节中的复杂度界可以利用标准数据库基础设施实现。本节介绍该原型的外部接口；一篇包含性能基准测试的详细系统论文正在准备中。

具体而言，该扩展通过提供统一 API，实现定义 4.7 中的五项一等操作，从而满足四大支柱，并因此满足等价定理（定理 6.3）：通过以 UUID 为键的超边实现存储和检索，通过富含谓词的 SQL 运算符实现查询，通过因果_链接关系实现组合，并通过一等结果集实现返回。下面的各小节将详细说明每个支柱是如何具体实现的。

9.1 存储模型

该扩展将每个支柱表示为原生数据库构造，而不是应用层约定。逻辑存储模型将超边公开为一等数据库对象：

```
CREATE TABLE 超边 (
   编号 通用唯一标识符 主键 ,
   节点 TEXT [] 非空,                          -- P1：通过数组实现 N 元
   属性 JSONB ,                                -- P1 :  灵活属性
   有效_时间 带时区时间范围 NOT NULL , -- P2：时间有效性
   置信度 FLOAT                                -- P3 :  不确定性
      CHECK ( 置信度 >= 0 AND 置信度 <= 1)
) ;

CREATE TABLE 因果_链接 (                       -- 关系之间
   原因_编号 UUID REFERENCES 超边 ( 编号) ,          -- 关系
   结果_编号 UUID REFERENCES 超边 ( 编号),
   机制 TEXT
) ;
```

与四大支柱的对应关系是直接的：**P1**通过节点数组（任意元数）和属性 JSONB（灵活属性）实现；**P2**通过有效_时间范围类型及其原生时间包含运算符实现；**P3** 通过带有检查约束 $\kappa \in [0, 1]$ 的置信度标量实现；**P4** 则通过因果_链接表实现，其外键将超边连接到超边。虽然逻辑模式可以用标准 SQL 表达，但 C 扩展为核心操作提供了优化后的实现，其能力超越了 SQL 层面的变通方案所能实现的范围。

9.2 API 接口

该扩展将四项核心操作作为原生 PostgreSQL 函数公开：

- atch.at_time(t)：时间旅行查询，返回截至时间戳 t时的完整知识状态，并解析所有超边的时间有效性。
- atch.trace_causal_chain(target, depth)：从目标超边开始遍历因果链，返回完整的因果历史，并提供截至指定深度的每条链路的置信度分数。
- atch.propagate_confidence(chain)：依据定理 7.5 的传播规则计算因果链上的联合置信度，其中包括 N 元属性带来的上下文修正。
- atch.discover_hidden_context(cluster_a, cluster_b)：实现定理 7.17 的自动隐藏变量发现。给定两个相互矛盾的超边簇（由 UUID 数组标识），该函数会计算所有 JSONB 属性的信息增益，并返回最能划分这些簇的属性——即隐藏上下文变量 z，识别该变量即可消解表面上的矛盾。核心熵计算由 C 语言实现，以提升性能；该函数只需对 N 条观测的 m 个属性进行一次 $O(N \cdot m)$ 遍历。

这些操作可通过标准 SQL 查询访问，使 ATCH 框架能够与现有的 PostgreSQL 工具、对象关系映射器和应用程序代码互操作。

9.3 复杂度验证

原型验证了定理 8.2 中确定的可处理片段，借助标准 PostgreSQL 索引基础设施，可达到预期的复杂度界：

- 点查找：通过超边标识符上的 B 树索引，复杂度为 $O(1)$
- 时间旅行查询：通过时间范围列上的 GiST 索引，复杂度为 $O(\log n)$
- 因果链遍历（深度 d）：通过优化的图遍历，复杂度为 $O(n \cdot d)$
- 置信度传播：在因果遍历过程中内联计算，开销可忽略不计

选择 PostgreSQL 作为宿主系统是经过深思熟虑的：它提供 ACID 保证、成熟的索引（B 树、GiST、GIN）、强大的扩展 API，以及广泛的生态系统兼容性。该扩展补充了 PostgreSQL 的 SQL 层无法原生表达的能力——一等超边操作、时序因果查询和置信度传播——而用户无需放弃现有的数据库基础设施。与现有系统进行详细的性能基准测试，将作为未来工作的一个方向。

10 表达能力基准

✓= 原生， ○ = 变通方案， × = 不可能**我们**
框架中的 Q7（原生）：

表 1：基准查询表达能力

Q	查询	P	SQL	LPG	TypeDB	我们的方法
1	带属性的三方会议	1	×	×	✓	✓
2	X 在日期 D 的状态	2	○	×	×	✓
3	与 $\kappa > 0.8$ 的关系	3	○	○	×	✓
4	关系 R 的成因是什么?	4	×	×	○	✓
5	沿着链传播 κ	3,4	×	×	×	✓
6	在区间 I 内有效的关系	1,2	○	×	×	✓
7	时刻 T 的因果历史（含 κ）	All	×	×	×	✓

```
SELECT atch . trace_因果_链 ( 目标 :=' 失当行为_发现 ' , 截至 :=' 2024 -08 -15 ',
计算 _ 置信度 := 真 ) ;-- 返回：处方 (0.73) - - [0。89] - - > 反应 (0.95) -- - - [0。
78]- - >发现 (0.62) -- 链置信度：0.51
```

定义 10.1（查询可表达性）。当且仅当存在一种系统 $\mathcal{K}$ 的语言中的查询 q，使得对于所有数据库 D，查询 Q 可表达：$q(D) = Q(D)$。

定理 10.2（严格表达能力层次）。

$$SQL \subsetneq LPG \subsetneq TypeDB \subsetneq ATCH$$

证明。我们通过一个见证查询来证明每一种严格分离。

SQL $\subsetneq$ LPG： 见证 $Q_1 =$ “返回连接节点 a 和 b的边，以及该边自身的属性和身份，使结果可用作后续遍历步骤的输入。”在 LPG 中，边是一等对象，具有持久身份：MATCH (a)-[r]->(b) RETURN r 会将 r 作为一个对象返回，该对象带有自身的属性，并可参与进一步的模式匹配。在 SQL 中，最接近的等效方式是通过关联表进行连接，但结果是一个派生行——它没有独立于组成它的外键而存在的持久身份，无法携带关系本身固有的属性（而非恰好位于同一行中的列），也无法在后续连接中作为参数引用，除非重新构造键元组。这种分离并非关乎结果值 是否可计算，而是关乎数据模型是否为边赋予结构身份：SQL 的关系模型将关系视为由元组共现派生而来；LPG 的模型则将它们视为具名、有类型的对象。包含性：LPG 中的每个 SQL 表都可嵌入为一组互不相连、带有属性的节点。

LPG $\subsetneq$ TypeDB： 见证 $Q_2 =$ “带属性的三元关系”。LPG 的边只能恰好连接 2 个节点；三元关系需要通过具体化来实现（这会改变语义）。TypeDB 则通过原生 n 元关系实现。

TypeDB $\subsetneq$ 我们的框架： 见证 $Q_3 =$“在 t 时有效的关系，置信度为 $\kappa > 0.8$”。TypeDB 不支持时间有效性和置信度语义。若将它们作为数据属性，而非元数据，语义便会有所不同，且无法传播。

每种区分都十分严格：若不造成结构失真，这些见证便无法表达——目标系统只能通过会改变数据模型的编码方式（具体化、模式复制或丢失元数据语义）来计算等价的结果集。 □

□

图4通过各层级的见证查询和支柱支持，将这一严格包含链可视化呈现出来。

表2：系统比较摘要

系统	P1	P2	P3	P4	*FC*	完整?
SQL	×	○	×	×	×	No
LPG（Neo4j）	×	○	×	×	○	No
RDF/OWL	○	×	×	○	○	No
TypeDB	✓	×	×	○	✓	No
Ours	✓	✓	✓	✓	✓	**Yes**

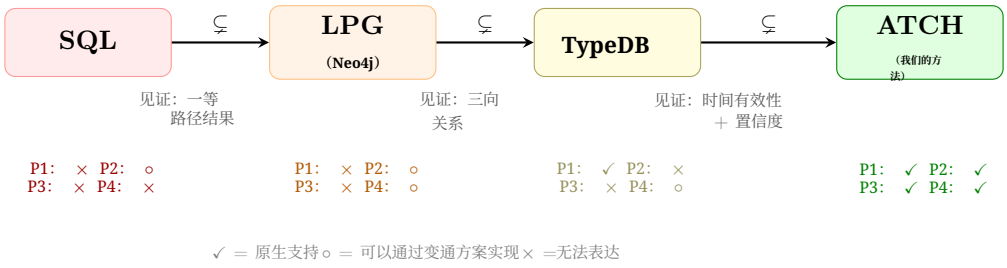

图4：包含见证查询和支柱支持的严格表达能力层次。每个系统的表达能力都严格弱于其后继系统。只有 ATCH 框架原生支持四大支柱。

11 信息论分析

我们量化了将框架投影到低表达能力模型时的信息损失（图1展示了二元投影引入的结构歧义）。

定义11.1 (二元投影)。 π_2： $\mathcal{H} \to G$ 将超边 $\{v_1, \ldots, v_n\}$ 映射为团$\{(v_i, v_j)$： $i < j\}$。

定义11.2 (重构歧义)。 二元投影的重构歧义 $\mathcal{A}(\mathcal{H}, \pi_2)$ 是从二元投影$\pi_2(\mathcal{H})$中恢复原始超图 $\mathcal{H}$ 所需的最少比特数。

定理 11.3 （信息损失界）。对于平均元数为 $\bar{n} > 2$ 的超图 $\mathcal{H}$：

$$\mathcal{A}(\mathcal{H}, \pi_2) \geq |\mathcal{E}| \cdot \binom{\bar{n}}{2} \ bits$$

证明。考虑包含 n 个节点的超边 e。在 π_2 下， e 会映射为由 $\binom{n}{2}$ 条二元边组成的团。逆映射 π_2^{-1} 必须确定：这些边究竟源自 (a) 一条 N元超边，还是 (b) $\binom{n}{2}$ 个独立的二元关系，抑或 (c) 两者的某种组合?

如果没有额外信息，那么所有 2($\binom{n}{2}$) 个二元边子集都可能是其来源。消除这种歧义所需的最小描述长度为每条超边 $\binom{n}{2}$ 比特（每条二元边用一比特标明它是否属于原始超边）。

对所有元数分别为 n_1、 n_2、……、$n_{|\mathcal{E}|}$ 的超边求和：

$$\mathcal{A} \geq \sum_{i=1}^{|\mathcal{E}|} \binom{n_i}{2}$$

由于 $\binom{n}{2} = n(n-1)/2$ 在 n 上对 $n \geq 2$ 是凸函数，根据詹森不等式可得：

$$\frac{1}{|\mathcal{E}|} \sum_{i=1}^{|\mathcal{E}|} \binom{n_i}{2} \geq \binom{\bar{n}}{2}$$

其中 $\bar{n} = \frac{1}{|\mathcal{E}|}\sum_i n_i$ 是平均元数。两边同时乘以 $|\mathcal{E}|$:

$$\mathcal{A} \geq |\mathcal{E}| \cdot \binom{\bar{n}}{2}$$

当超边具有统一元数时，此界是紧的。 □ □

定理 11.3 量化了使用二元系统的最低成本；下文的命题 11.5 将其推广到缺少任意子集支柱的系统。

推论 11.4 (不可逆性**)。** 二元投影并非信息保持：一般而言，$\pi_2^{-1}(\pi_2(\mathcal{H})) \neq \mathcal{H}$。

命题 11.5 (总表达能力差距**)。** 在缺少支柱 S 的系统中表示 $ATCH$ $\mathcal{H}$所丢失的信息有界于：

$$\Delta H(S) \leq \sum_{P_i \in S} H_i(\mathcal{H})$$

其中， H_1 = 表示结构， H_2 =表示时间， H_3 =表示置信度， H_4 =表示因果熵。

注释 11.6。 该界是不等式而非等式，因为等价定理确立了支柱之间的相互蕴涵，这意味着熵分量之间可能存在相关性。当支柱彼此独立地缺失时（例如，系统支持结构但不支持时间），加法界成立；当多个支柱同时缺失时，它们之间的共享信息可能使实际损失低于各项之和。该界量化了使用不完整系统的最坏情况下的“代价”。

12 讨论

12.1 跨知识类型的表征覆盖范围

四大支柱定义了知识的结构维度：涉及什么（P1）、何时成立（P2）、确定程度如何（P3）以及由什么导致（P4）。领域特定语义作为属性存在于这一结构之中。支柱并不限制存储的内容——它们确保容器保留关于该内容的一切信息。

为证实这一论断，我们系统地考察了各种知识类型如何映射到该框架（表 3）。在每种情况下，四大支柱提供结构支架，而属性则承载领域特定含义。

这一模式是一致的：四大支柱提供维度，属性承载语义。模态力量、空间几何、规范权重、模糊程度、类比强度——这些都是关系的属性。支柱确保这些属性拥有存放位置（P1）、时间范围（P2）、置信度等级（P3）以及与其他知识的连接（P4）。

注释 12.1 (表征完备性与计算完备性**)。**该框架声称具有表征完备性：任何知识都可以被忠实地存储和检索，而不会发生结构失真。特定领域的推理——概率推理、空间计算、道义逻辑检查——需要额外的计算机制来处理所存储的表征。该框架是存储层，而非推理引擎。

12.2 对人工智能的启示

等价定理对人工智能研究的多个活跃领域有着直接影响。四大支柱刻画了知识在结构上所需的条件；任何存储、检索或推理知识的人工智能系统，无论是否满足这些条件，都必然受到它们的约束。除了前文各节探讨的经典问题（第 7.1 节的框架问题、第 7.3 节的冲突解决，以及第 7.4 节的常识推理）之外，该框架还针对现代人工智能系统中的结构性局限性。

检索增强生成（RAG）。当前的 RAG 流程会根据语义相似度检索文档，然后将文本块交给语言模型进行推理。这种架构继承了底层存储的局限性：文档是非结构化文本，而相似度搜索无法还原关系结构。查询可能会检索到一张描述打印故障的工单，却无法呈现三周前为其埋下隐患的驱动程序变更——因为因果联系只隐含地存在于不同文档之间。该框架提供了一种结构化的替代方案：人工智能系统无需检索相似文本，而是可以直接查询关系，沿着因果链（P4）追溯、按时间有效性（P2）筛选，并按置信度（P3）排序。检索问题由“查找相似文档”转变为“查找相关关系”，这是截然不同且表达能力更强的操作。近期有关 HyperGraphRAG [13] 的研究独立验证了这一方向，证明在多个领域中，超图结构化知识检索的表现优于基于二元图的方法——这恰恰是因为二元知识图谱无法表示 n 元关系。

神经符号集成。神经符号人工智能社区正积极寻找恰当的符号表示层，以补充神经感知和语言能力。目前大多数方法采用带二元边的知识图谱 [26]，本文证明这类图谱在形式上并不完备（定理 10.2）。无论神经组件的能力有多强，如果神经符号系统的符号组件无法表示 n 元关系、时间有效性、不确定性或关系之间的因果联系，就会在符号边界处丢失信息。等价定理为符号层必须包含的内容提供了形式化规范：具备四大支柱的一等关系，是实现结构完备的符号表示的必要且充分条件。

超图系统。研究文献早已指出，超图是对复杂关系进行建模的恰当数学基础 [4]。然而，实际实现仍然十分少见。现有超图研究主要关注结构属性——社区检测、划分、谱分析——却没有涉及时间有效性、不确定性量化，或超边之间的因果关系。ATCH 框架通过为超图扩展恰好满足结构完备表示所需的能力，弥合了这一差距。这为超图理论研究走向实用知识系统提供了一条路径：为超边增添时间界限、置信度分数和因果排序，由此形成的结构经证明足以表示任何关系知识。

持久化人工智能记忆。目前，大语言模型无法在不同对话之间保留持久记忆。上下文窗口有限，先前交互中的信息会丢失，除非将其明确重新注入。该框架为人工智能记忆提供了具有形式化基础的架构：具有时间有效性的关系能够自然地表示曾在某个时刻为真、但在另一个时刻并非如此的知识（P2）；随着证据不断积累、置信度程度各异的知识（P3）；以及关于不同交互中事物之间因果关系的知识（P4）。尽管与语言模型的集成仍是一项工程挑战，该框架仍提供了持久化记忆系统所需的表征保证，确保存储的知识能够随着时间推移保留其完整的关系结构。

可解释性。人工智能系统面临的一项长期挑战，是回答“为什么会发生这种情况？”目前，大多数可解释性方法都试图在事后重构推理过程——分析注意力权重、计算特征归因，或生成自然语言解释

，而这些都发生在决策作出之后。如果将因果关系存储为一等对象，可解释性就会成为查询，而非重构。因果链 $e_1 \prec e_2 \prec e_3$ 及其相关置信度分数构成了原生审计追踪：系统可以通过遍历 $\prec$ 直接回答“是什么导致了这一结果？”，而非根据不透明的内部状态近似生成解释。

隐私保护知识聚合。当知识系统跨组织边界部署时——例如部署于多家医院、金融机构或托管服务提供商之间——跨部署模式检测需要在不暴露私有数据的情况下聚合知识。四大支柱实现了清晰的结构／实例分离，这是二元图表示无法做到的。具有明确参与者角色（P1）的超边可以通过单一且定义明确的操作投影为结构模式签名——保留关系类型、参与者类型和角色结构，同时舍弃所有实体标识符和属性值。在二分图中，同一关系会分散在多条边上，结构模式与实例数据之间不存在清晰边界，因此匿名化只能依赖启发式方法，而非遵循原则。支柱 3 提供了第二项结构优势：由于每条超边都携带置信度值 $\kappa \in [0, 1]$，因此可以通过聚合置信度统计刻画模式，在不保留单个实例的情况下汇总多个观测所提供的证据。导出这些聚合统计数据而非原始数据，使抽象成为表示形式固有的一部分，而不是事后强加的处理。支柱 2 支持撤回同意：由于每条超边都携带时间界限，退出知识共享的部署方可以按时间窗口从聚合置信度分数中清晰扣除其贡献，而无需根据原始数据重新计算。支柱 4 支持共享因果拓扑——“类型 A 的变更通过类型 C 的组件导致类型 B 的失败”——而不会暴露具体有哪些实体参与其中，因为因果链接连接的是带类型的超边，而非已识别实体。综合来看，这些属性表明，ATCH 框架不仅是联邦知识系统的一种可能基础，更是结构上必需的基础：它提供了同时支持完整知识与有原则的隐私边界所需的最小表示。

12.3 局限性

- 一般超图查询仍然是 NP 完全问题；实际系统必须限制在可处理片段内（第 8 节）

- 特定领域推理（概率推理、空间计算、道义逻辑检查）需要存储层之外的计算扩展——该框架在表示层面是完备的，而非推理完备的

- 复杂的联合概率分布可以作为属性存储，但缺乏原生传播语义；与贝叶斯推理引擎集成则是互补的

- 等价定理确立了四大支柱对于完备性的必要性，但并未声称它们足以应对存储知识上的所有计算 任务

- 从现有系统迁移需要进行数据转换；转换成本属于工程问题，不在理论框架的讨论范围内

13 结论

我们已经证明，结构完整的知识表示需要具备四项能力——带属性的 N 元关系、时间有效性、不确定性量化和因果关系——且这四项能力彼此蕴含：任何实现其中一项的系统都必然需要具备全部四项，而这四项等价于将关系视为一等对象。

等价定理确立了 SQL $\subsetneq$ LPG $\subsetneq$TypeDB $\subsetneq$ATCH 之间严格的表达能力层次；在每个较低层级，都存在结构上无法表达的见证查询。作为直接推论，四大支柱为经典人工智能问题提供了解法——框架问题、通过隐变量发现来解决冲突、因果链中的不确定性传播，以及带有例外的常识推理——这些解法并非独立的机制，而是表示结构的自然结果。

对多种知识类型（情态、空间关系、程序性、规范性、反事实、类比的等）的系统性考察证实，四大支柱提供了结构维度，而领域特定语义则存在于属性中。限定完整性定理（定理 7.23）进一步确立，所有常识例外都可以沿着这些维度表达。

完备性的计算成本有界：一般超图查询虽然是 NP 完全问题，但涵盖实用查询类别的可处理片段——无环模式、有界深度因果链、窗口化时态查询——可在多项式时间内运行。一个以 C 语言编写的 PostgreSQL 原型扩展证实，利用标准数据库基础设施即可实现这些界限。

核心洞见在于，一等关系并非诸多功能中的一项，而是一个结构性门槛：低于这一门槛，信息损失不可避免，并且会随着每次操作不断累积；高于这一门槛，知识表示的完整谱系便可用。更广泛地说，等价定理刻画的并非数据库所需的内容，而是知识所需的内容：任何会随着时间推积来自不确定观测的信念的系统——无论是数据库、人工智能智能体、传感器网络还是科学仪器——都需要具备这四种能力，才能使其信念在结构上忠实于它所建模的领域。

配套的 Lean 4 形式化提供了核心结果的机器验证证明：二元分解失败引理、全部五个等价引理（一等公民 $\Rightarrow$ P1–P4 以及每个 Pi $\Rightarrow$ 一等公民）、主要等价定理、全有或全无推论、框架问题持久性定理（包括任意动作序列下的持久性），以及表达能力层级。该形式化在 Lean 4 v4.14.0 下编译时零错误，并可通过 https://github.com/Network-Services-Group/equivalence-theorem-lean4 获取。

未来仍有几个研究方向。首先，实证验证：参考实现（第 9 节）提供了一个起点，但若能在真实世界的数据集上，将 ATCH 的查询性能与 SQL、LPG 和 TypeDB 工作负载进行系统性的基准测试，将进一步增强其实用性论据。其次，领域特定案例研究：将 ATCH 应用于特定领域（临床数据、供应链、IT 基础设施），可以展示该框架的涌现属性，并发现实际的优化机会。参考实现和基准测试套件将在完成后作为开源代码库发布。

致谢

作者在本作品的创作过程中使用 Claude（Anthropic）作为人工智能研究助理。人工智能辅助用于数学形式化、文献综述和稿件准备。所有理论贡献、研究方向、领域专业知识、证明验证和最终编辑决策均由作者作出，作者对内容的准确性和完整性承担全部责任。

参考文献

[1] J. F. Allen. 关于时间区间知识的维护。*ACM* 通讯，26(11):832–843，1983年。[2] R. Angles. 属性图数据库模型。载于*AMW* 论文集，CEUR Workshop Proceedings，第2100卷，2018年。[3] R. Angles, A. Bonifati, S. Dumbrava, G. Fletcher, A. Green, J. Hidders, B. Li, L. Libkin, V. Marsault, W. Martens, F. Murlak, S. Plantikow, O. Savković, M. Schmidt, J. Sequeda, S. Staworko, D. Tomaszuk, H. Voigt, D. Vrgoc, M. Wu, 和 D. Zivkovic. PG-Schema：属性图的模式。*ACM* 数据管理会议论文集，1(2):198:1–198:25，2023年。https: //doi.org/10.1145/3589778[4] R. J. Brachman 和 H. J. Levesque.《知识表示与推理》。Morgan Kaufmann，2004年。[5] E. F. Codd. 面向大型共享数据库的数据关系模型。*ACM* 通讯，13(6):377–387，1970年。[6] J. McCarthy 和 P. J. Hayes. 从人工智能视角看若干哲学问题。*Machine Intelligence*，4:463–502，1969年。[7] R. Reiter. 一种默认推理逻辑。*Artificial Intelligence*，13(1-2):81–132，1980年。[8] J. Pearl.《因果关系：模型、推理与推断》。剑桥大学出版社，2009年。[9] R. T. Snodgrass.《在 SQL 中开发面向时间的数据库应用》。Morgan Kaufmann，2000年。[10] D. Suciu, D. Olteanu, C. Ré, 和 C. Koch.《概率数据库》。Morgan & Claypool，2011年。[11] ISO/IEC. ISO/IEC 39075:2024 — 信息技术 — 数据库语言 — GQL。国际标准化组织，2024年。https://www.iso.org/standard/ 76120.html[12] ISO/IEC. ISO/IEC 9075-16:2023 — 信息技术 — 数据库语言 — SQL — 第16部分：属性图查询（SQL/PGQ）。国际标准化组织，2023年。https://www.iso.org/standard/79473.html[13] H. Luo, H. E, G. Chen, Y. Zheng, X. Wu, Y. Guo, Q. Lin, Y. Feng, Z. Kuang, M. Song, Y. Zhu, 和 A. T. Luu. HyperGraphRAG：通过超图结构化知识表示实现检索增强生成。载于*NeurIPS* 论文集，2025年。https://arxiv.org/abs/ 2503.21322[14] R. Reiter.《行动中的知识：用于规范和实现动态系统的逻辑基础》。麻省理工学院出版社，2001年。[15] R. Kowalski 和 M. Sergot. 基于逻辑的事件演算。*New Generation Computing*，4(1):67–95，1986年。[16] F. Bacchus, J. Y. Halpern, 和 H. J. Levesque. 在情境演算中对噪声传感器和执行器进行推理。*Artificial Intelligence*，111(1-2):171–208，1999年。[17] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, 和 P. F. Patel-Schneider，编者。《描述逻辑手册：理论、实现与应用》。剑桥大学出版社，2003年。

[18] A. Artale、E. Franconi、F. Wolter 和 M. Zakharyaschev。《用于对概念模式和查询进行推理的时序描述逻辑》。载于*JELIA* 会议论文集，LNAI 2424，第 98–110 页。Springer，2002年。[19] A. Artale、R. Kontchakov、A. Kovtunova、V. Ryzhikov、F. Wolter 和 M. Zakharyaschev。《时序数据上的本体介导的查询应答：综述》。载于*TIME* 会议论文集，LIPIcs 第 90 卷，第 1:1–1:37 页，2017年。[20] T. Lukasiewicz。《表达力强的概率描述逻辑》。*Artificial Intelligence*，172(6- 7):852–883，2008年。[21] C. Berge。《超图：有限集组合学》。North-Holland，1989年。[22] S. Abiteboul 和 R. Hull。《数据函数、Datalog 与否定》。载于*ACM SIGMOD* 会议论文集，第 143–153 页，1988年。[23] C. S. Jensen 和 R. T. Snodgrass。《时序数据管理》。*IEEE Trans. Knowl. DataEng.*，11(1):36–44，1999年。[24] N. Francis、A. Gheerbrant、P. Guagliardo、L. Libkin、V. Marsault、W. Martens、F. Murlak、L. Peterfreund、A. Rogova 和 D. Vrgoc。《GPC：属性图的模式演算》。载于*PODS* 会议论文集，第 241–250 页，2023年。https://doi.org/10.1145/3584372.3588662[25] A. Deutsch、N. Francis、A. Green、K. Hare、B. Li、L. Libkin、T. Lindaaker、V. Marsault、W. Martens、J. Michels、F. Murlak、S. Plantikow、P. Selmer、O. van Rest、H. Voigt、D. Vrgoc、M. Wu 和 F. Zemke。《GQL 和 SQL/PGQ 中的图模式匹配》。载于*SIGMOD* 会议论文集，第 2246–2258 页，2022年。https://doi.org/10.1145/3514221.3526057[26] A. Hogan、E. Blomqvist、M. Cochez、C. d' Amato、G. de Melo、C. Gutierrez、S. Kirrane、J. E. L. Gayo、R. Navigli、S. Neu maier、A.-C. Ngonga Ngomo、A. Polleres、S. M. Rashid、A. Rula、L. Schmelzeisen、J. Sequeda、S. Staab 和 A. Zimmermann。《知识图谱》。《*ACM*计算调查》，54(4):71:1–71:37，2021年。https://doi.org/10.1145/3447772[27] K. Ahrabian、E. Boxer 和 J. Pujara。《迈向更适用于地缘政治事件预测的时序结构》。*arXiv* 预印本 *arXiv:2601.00430*，2026年。https://arxiv.org/abs/2601. 00430[28] S. Sadoughi、N. Yakovets 和 G. Fletcher。《元属性图：通过元数据感知和具体化扩展属性图》。*arXiv* 预印本 *arXiv:2410.13813*，2024年。https://arxiv.org/abs/2410.13813

情态逻辑扩展

核心框架可以扩展，以处理模态算子：

定义 A.1 (模态置信度**)**。使用限定符扩展 κ：

- $\kappa^{\Box}(e)$：确信 e 必然为真
- $\kappa^{\Diamond}(e)$：确信 e 可能为真

这保留了等价定理：模态限定符是关系的属性，因此需要赋予其一等地位。

表 3：不同知识类型的表征覆盖

知识 Type	示例	编码
事实性	“技术人员 推送打印 驱动程序”	与参与者和 属性的关系 (P1)
情态	“The 服务器 可能地 失败”	属性 模态：可能；$\kappa = 0.3$ 反映可能性程度 (P1, P3)
空间关系	“服务器 A 与 B 相邻 在机架中”	{A，B，机架}，距离为_米：0.5 (P1)
程序性	“如何回滚组策略对象 更改”	因果链：步骤$_1$ $\prec$ 步骤$_2$ $\prec$ 步骤$_3$, 每个都与之存在 时间顺序关系（P2、P4）
规范性	“推送驱动程序前 先测试组策略对象的影响”	{技术人员, GPO, 驱动程序} with 义务：应当性和时间 有效性（策略 P1、P2）
反事实	“如果我们当时没有推动那件事 驱动程序。. .”	关系 with 类型：反事实 引用 the 因果链 (P1, P4)
类比的	“这次服务中断就像 三月那次”	两个相似的关系之间的关系：0.8（P1, P4）
负向	“未发现恶意软件”	{扫描，设备，恶意软件} with 结果：不存在，$\kappa = 0.99$, 有界 τ (P1, P2, P3)
聚合	“子网上的大多数客户端 具有延迟”	关于某个模式的高阶关系- 具有以下比率的模式：0.73（P1、P4）
身份	“DB-Prod-1 是 Legacy-SQL-3”	{名称$_1$，名称$_2$} 及其关系：身份，.....的时间界限 身份成立的时间（P1、P2）
模糊	“服务器有点 慢”	{服务器，性能} with 程度：0.6—模糊性 as a 属性的知识，not 对此的不确定性（P1）
时间 pattern	“总是在升级后崩溃 日期”	因果联系 更新 $\prec$崩溃 与 频率：始终，计数：12（P1，P4）
义务性	“技术人员已获授权可以 重启”	{技术人员, 服务器，重启} with 权限：已授权，τ 用于 有效期（P1，P2）
机构	“计为严重级别 1”	{事件，分类} with 规则：策略 描述 why (P1, P4)
元认知	“我们不知道备份是否 成功了”	{系统, 知识缺口, 备份} 带有缺口：结果_未知，$\kappa = 0.5$（P1、P3、P4）
指示性	“The 当前 主服务器 服务器”	{服务器, 角色, 主服务器}，带有 $\tau = [$开始，$\infty)$；“当前” 由.....解决 时态查询（P2）