arXiv:2602.11949v1 [cs.DB] 12 Feb 2026

RPQ语义的设计与比较

Victor Marsault Univ Gustave Eiffel、CNRS、LIGM，法国马恩拉瓦莱 F-77454 **Antoine Meyer** Univ Gustave Eiffel、CNRS、LIGM，法国马恩拉瓦莱 F-77454

摘要

现代属性图数据库查询语言，如 Cypher、PGQL、GSQL 以及标准 GQL，均借鉴了正则路径查询（RPQ）这一形式体系。为了显式输出游走，它们不再采用经典且经过充分研究的同态 语义。然而，由于正则路径查询可能匹配无限多个游走，向用户呈现结果便变得困难。上述语言采用特定标准，从这些匹配项中选取一个有限子集。例如，Cypher 采用迹 语义，舍弃包含重复边的游走；PGQL 和 GSQL 采用最短游走 语义，仅保留所有匹配游走中长度最短的游走；而 GQL 则允许用户从多种语义中进行选择。尽管已有关于这些语义的学术研究，但研究几乎完全聚焦于评估效率。

为了更好地理解、选择和设计正则路径查询语义，我们提出了一个框架，依据其他标准对其进行分类和比较。我们将若干可能的性质形式化，这些性质涉及将正则路径查询语义视为数学函数的研究：该函数将数据库和查询映射到一个有限 游走集合。我们证明，某些性质彼此排斥，或无法满足。我们还给出若干新的正则路径查询语义作为示例。其中一些或可为未来图数据库查询语言中新语义的设计提供思路。

2012 年 ACM 学科分类 计算理论 → 数据库查询语言（原理）；信息系统 → 非关系型引擎的查询语言；信息系统 → 基于图的数据库模型；计算理论 →正则语言

关键词与短语 正则路径查询；正则路径查询；语义；图数据库；模式匹配；正则表达式

相关版本 本工作的简短版本发表于 [MM26]。

致谢 本工作源于与许多同事的讨论，其中包括 Claire David、Amélie Gheerbrant、Steven Sailly 和 Cristina Sirangelo。我们特别感谢 Thomas Colcombet 提出了定义 ShVC 语义的想法，也感谢 Nadime Francis 和 Sara Khichane 向我们介绍了命题 62 中使用的手电筒搜索技术。最后，感谢 Yann Strozecki 在枚举复杂度框架方面给予的帮助。

1 引言

过去几十年来，人们对图数据库管理系统（DBMS）的兴趣与使用日益增加。使用此类数据库时，典型任务是遍历图。为此，大多数现代查询语言都包含受正则路径查询（RPQ）[CMW87]启发的功能。学术文献对正则路径查询及其推广形式进行了深入研究。本质上，正则路径查询是一种用于查询边标记图的正则表达式。游走（即由顶点和边交替组成的序列）R 若其标签（即其边标签的连接）符合 R，则匹配正则路径查询。

在工业界，正则路径查询被加入标准语言 SPARQL [SPARQL13]，用于资源描述框架。它们也是随成功的属性图数据库管理系统 Neo4j 一同推出的语言 Cypher [Cypher24; FG+18a], 中 `MATCH` 子句的基础。正则路径查询还启发了其他面向图数据库管理系统的查询语言 [GSQL23; PGQL21]。部分由于 Cypher 和 Neo4j 的成功，人们设计了面向属性图的标准查询语言 [GQL24]。

© Victor Marsault 和 Antoine Meyer；依据知识共享许可协议 CC-BY 4.0 授权

其导航部分深受正则路径查询的影响，并被设计为与 SQL[PGQ23] 兼容。

正则路径查询通常采用 同态语义 [AA+17]进行求值。在这种设定下，查询 R 的答案是所有满足以下条件的有序对 (s,t) 的集合：存在一条匹配 R、从 s到达 t的游走。换言之，答案由所有匹配游走的端点 组成。然而，在某些实际场景中，端点语义并不够用：用户有时需要获取完整的匹配游走，而不仅仅是它们的端点。

正则路径查询通常会匹配无限多条游走，因此，如果查询要返回完整游走，将它们全部呈现给用户会带来问题。为了解决这一问题，许多用于属性图的实际查询语言会以各种方式选取匹配游走的一个有限子集，并将其返回给用户。Cypher 使用迹语义：只有不包含边重复的匹配游走才会被返回。GSQL 或 PGQL 等其他语言则使用最短语义：只有不存在端点相同的更短匹配时，才会返回该匹配游走。另一个广为人知的例子是无环语义：只有不包含顶点重复的匹配游走才会被返回。GQL 允许用户在多种语义中进行选择。本文沿用这一现代方法：我们所说的 RPQ 语义，是指任何一种规范，用于从可能无限的匹配游走集合中选取一个有限 子集并返回给用户。

近期关于不同语义下正则路径查询的研究主要关注求值过程。在非循环 [BBG20]或迹[MNP23] 语义下，与求值相关的计算问题难以处理。另一方面，在最短语义下存在高效的求值算法 [MT18; DFM24]。查询求值的一项挑战是结果数量庞大，已有研究致力于更好、更简洁地表示结果集 [MN+23; MN+24]。这描绘出一幅非常片面的图景：最短语义总能带来效率高得多的算法。然而，迄今最流行的系统 Neo4j 使用的是迹语义，我们的假设是，这可以归因于其他一些理想性质。

贡献。 为开展这项研究，我们提出了一个用于定义和分类正则路径查询语义的形式化框架。我们将迹语义或非循环语义称为基于筛选的语义，因为它们充当后置过滤器：每条游走都会被单独判定为符合条件（例如，没有重复）或不符合条件（例如，存在重复），而不受其他匹配项影响。我们将最短语义称为基于序的语义，因为它们依据某种全局准则选出最优的匹配游走。更准确地说，这类语义通过游走上的（偏）排序关系来定义，并返回相对于该排序而言极小的所有游走（在匹配项中）。

随后，我们定义了正则路径查询语义可能具备或不具备的诸多性质，并探讨这些性质为何值得追求。举例来说，我们引入了与单调性、连续性、使用有理算子的可组合性以及覆盖相关的性质。由于我们以抽象方式定义正则路径查询语义，因此很容易构造出不实用甚至毫无意义的语义。为此，我们提出了一系列任何合理语义都应具备的性质。我们还给出了若干不可能性定理，形式为一组组彼此排斥的性质。这些结果旨在促进新颖且有趣的正则路径查询语义的设计。我们常用新的正则路径查询语义示例来阐释定义和论述（这些示例汇编在索引中，见第 30 页）。其中一些似乎是有望应用于实践的合理候选方案。

概要。 第2节介绍基础词汇和定义，尤其是我们用于数据（标记图）和查询（正则表达式）的模型。接着，第3节介绍

我们的框架：什么是正则路径查询语义、如何定义一种语义，以及我们关注哪些语义。第 4 节介绍正则路径查询语义可能具备的若干性质。我们总结了几种常见语义的性质，并给出一些一般性结论（例如，哪些性质彼此互斥）。我们还探讨了这些性质是否值得追求。第 5 节回顾一些关于计算复杂性的已知结果。

本文介绍了许多正则路径查询语义，每种语义都有一个缩写，并介绍了许多相关性质。第 30 页提供了索引。

相关工作。 据我们所知，除端点语义、无环语义、迹语义和最短语义外，学术研究中很少探讨其他正则路径查询语义。SPARQL 这一资源描述框架查询语言中与正则路径查询相关的片段称为属性路径 [SPARQL13]，而在形成其定义的过程中，有一种候选语义确实会计算游走集合。为了只呈现有限数量的结果，人们决定在这种语义下，不能使用 Kleene 星号重复同一游走。由于会返回过多结果，这种候选语义最终被弃用 [ACP12]；这种现象有时被称为*Yotabyte* 问题。最终决定，SPARQL 中的属性路径采用端点语义求值，但有所变通，最容易通过改写来说明：属性路径会自动改写为其他 SPARQL 构造，以得到一条查询，其中所有属性路径的最外层运算符都是 Kleene 星号（参见[SPARQL13]中的第 *9.3* 节，属性路径与等价模式）。这一看似奇怪的决定源于对重数进行一致处理的需要。

在 GQL[GQL24] 和 SQL/GPQ[PGQ23], 中，关键字 `SIMPLE` 会启用一种无环语义变体，该变体返回的不仅有简单游走，也有简单环。理论文章通常不会考虑这种变体，因为它在计算上等价于无环语义。

在 [DFM23], 中，作者提出了两种新的语义（简单运行和绑定轨迹），它们根据底层匹配过程而非仅根据游走的拓扑结构来过滤游走。这类语义不在本文讨论范围内（见第 3.4 节），但 [DFM23]的引入与我们的工作尤为相关。实际上，与迹相比，它们的语义在计算表现上更好，但不如最短语义。为了说明最短语义并非总是最佳语义，他们采用了非正式的评判标准，而非基于理论算法效率的标准。本文旨在提供更系统的论证。

2 预备知识

我们假设存在 $\mathbb{V}$、$\mathbb{E}$和 $\mathbb{L}$，三个两两不相交的可数无限集，分别由顶点标识符、边标识符和标签/字母组成。稍后我们将在这些集合上定义所有数据库和正则表达式。这样，我们就能独立于底层数据库定义若干概念。我们将标识符称为 $\mathbb{V}\cup\mathbb{E}$中的任意元素；将重命名称为 ν上的 $\mathbb{V}\cup\mathbb{E}$ 的任意排列，使得 $\nu(\mathbb{V}) = \mathbb{V}$ 且 $\nu(\mathbb{E}) = \mathbb{E}$。我们将重新标记称为 $\mathbb{L}$上的任意排列。

2.1 词、语言、表达式

字母表是 Σ $\mathbb{L}$ 的有限子集。 Σ 上的一个字是 Σ 中元素组成的有限序列；空字记作 ε。 Σ 上的所有字构成的集合记作 Σ^*。 Σ^* 的子集称为 Σ 上的语言。字 $u = (a_1, \ldots\ldots, a_k)$ 通常写作 $a_1 \cdots a_k$，而 k 称为 u 的长度。两个字 $u = a_1 \cdots a_k$ 和 $v = b_1 \cdots b_\ell$ 的连接为 $u \cdot v = a_1 \cdots a_k b_1 \cdots b_\ell$，通常简写为 uv。连接可按如下方式提升到语言： $L_1 \cdot L_2 = \{w_1 \cdot w_2 \mid w_1 \in L_1, \quad w_2 \in L_2\}$。

一个正则表达式 R 在 Σ 上是关于字母集合的任意公式，以及一元函数 *和二元函数 $+$ 与$\cdot$ 所构成的、符合以下文法的表达式。

$$R := \varepsilon \mid a \mid R^* \mid R \cdot R \mid R + R \qquad \text{with } a \in \Sigma \tag{1}$$

任何此类 R 表示 一个语言 $L(R)$ 在 Σ之上，按通常方式归纳定义。(R)中的任意字都称为 符合 R。

2.2 游走与数据库

游走是由 $\mathbb{V}$ 中的顶点和 $\mathbb{E}$ 中的边交替组成的有限非空序列，且以顶点开始并结束。通常将这样的游走写作 $w = (v_0, e_1, v_1, \ldots, e_k, v_k)$，即$v_0$ —Ae_1 v_1 —A$e_2 \cdots$ —Ae_k v_k。我们称 w的长度为 k，并记作 Len(w)；分别称 w 的源点和目标点为 v_0 和 v_k ，并记作 Src(w) 和 Tgt(w)。有序对 (v_0, v_k) 是 w 的端点对，记作 EndP(w)。长度为 0 的游走称为平凡游走，由单个顶点构成。我们用 $w[i]$ 表示边 e_i，其中 $i \in \{1, \ldots, k\}$；用 wi表示顶点 v_i，其中 $i \in \{0, \ldots, k\}$。注意，游走没有特定的拓扑结构。因此，游走 w 可能与数据库 D 不一致，甚至自身也可能不一致（见下文）。

我们用 游走 表示所有游走的集合（与任何数据库无关）。两个游走 w和 w' 可连接 当且仅当 $\mathsf{Tgt}(w) = \mathsf{Src}(w')$，且它们的 连接，记作 $w \cdot w'$，是如下游走

$$w \cdot w' = \mathsf{Src}(w) \xrightarrow{w[1]} \cdots \xrightarrow{w[k]} \mathsf{Tgt}(w) \xrightarrow{w'[1]} \cdots \xrightarrow{w'[k']} \mathsf{Tgt}(w')$$

其中 $k = \mathsf{Len}(w)$ 且 $k' = \mathsf{Len}(w')$。注意， $w \cdot w'$ 的长度为 $k + k'$：它有 $k + k'$ 条边和 $k + k' + 1$ 个顶点（顶点 $\mathsf{Tgt}(w) = \mathsf{Src}(w')$ 不会重复）。我们照常将 $\cdot$ 扩展到游走集合： $W \cdot W'$ 包含所有满足以下条件的游走 $w \cdot w'$： $w \in W$且 $w \in W'$，并且两者可连接。我们以类似方式扩展 $*$算子： W^* 包含所有游走 $w_1 \cdot w_2 \cdots w_k$，其中每个 w_i 都属于 W，且每对相邻游走 w_i, w_{i+1} 都可连接。

在本文中，我们将图数据库建模为有向标记多重图，并简称为 数据库。一个 数据库 $\mathcal{D}$ 是一个元组 $(L_\mathcal{D}, V_\mathcal{D}, E_\mathcal{D}, \mathsf{Src}_\mathcal{D}, \mathsf{Tgt}_\mathcal{D}, \mathsf{Lbl}_\mathcal{D})$，其中 $L_\mathcal{D} \subset_{\text{fin}} \mathbb{L}$, $V_\mathcal{D} \subset_{\text{fin}} \mathbb{V}$, $E_\mathcal{D} \subset_{\text{fin}} \mathbb{E}$ 以及 $\mathsf{Src}_\mathcal{D}$、$\mathsf{Tgt}_\mathcal{D}$ 和 $\mathsf{Lbl}_\mathcal{D}$ 分别是函数，将 $E_\mathcal{D}$ 中的每条边映射到其源点、目标点和标签。

如果所有元素 w 都属于 $V_\mathcal{D} \cup E_\mathcal{D}$ ，并且对每个 $0 < i \leq$ ，Len(w)、$\mathsf{Src}_\mathcal{D}(w[i]) = w$ $i - 1$ 和 $\mathsf{Tgt}_\mathcal{D}(w[i]) = wi$，则称游走 w 与 D 一致。所有与 D 一致的游走集合，简称 D 中的游走集合，记作 Walks(D)。如果存在一个数据库 D，使得 n 游走w_1、.....、w_n 属于 $\subseteq$ Walks(D)，则称它们一致。每当 $n = 1$时，我们称 w_1 是自洽的。我们按预期将 $\mathsf{Lbl}_\mathcal{D}$ 扩展到 Walks(D)：$\mathsf{Lbl}_\mathcal{D}$(w) 是 w 中连续边标签的连接。当数据库 D 的含义明确时，我们会滥用术语和记号，将每条边 $e \in E_\mathcal{D}$ 视为游走 ($\mathsf{Src}_\mathcal{D}(e)$ e， $\mathsf{Tgt}_\mathcal{D}(e)$)。特别地，我们可以询问游走 w 是否能与边 e 连接；若能，则使用记号 $w \cdot e$。

任意重命名函数 ν 都可以通过将 ν 应用于其中的所有标识符来扩展到游走。它也可以按如下方式扩展到整个数据库： ν(D) $= (L_\mathcal{D}, \nu(V_\mathcal{D})$ $\nu(E_\mathcal{D})$, $\mathsf{Src}'_\mathcal{D}$, $\mathsf{Tgt}'_\mathcal{D}$, $\mathsf{Lbl}'_\mathcal{D})$，其中 $\mathsf{Src}'_\mathcal{D}(x) = \nu(\mathsf{Src}_\mathcal{D}(\nu^{-1}(x))$, $\mathsf{Tgt}'_\mathcal{D} = \nu(\mathsf{Tgt}_\mathcal{D}(\nu^{-1}(x))$，且 $\mathsf{Lbl}'_\mathcal{D}(x) = \mathsf{Lbl}_\mathcal{D}(\nu^{-1}(x))$。类似地，任意重标记函数 λ: $\mathbb{L} \to \mathbb{L}$ 都可以通过将 λ应用于所有原子来扩展到正则表达式，并按如下方式扩展到数据库： λ(D) $= (\lambda(L_\mathcal{D})$ $V_\mathcal{D}, E_\mathcal{D}$, $\mathsf{Src}_\mathcal{D}$, $\mathsf{Tgt}_\mathcal{D}$, $(\lambda \circ \mathsf{Lbl}_\mathcal{D}))$。

3 正则路径查询（RPQ）及其语义

正则路径查询（RPQ）的概念是借助边标签上的正则表达式来查询图数据库 $\mathcal{D}$ 的内容。

▶定义 *1*（匹配项）。给定一个正则表达式 R 和一个数据库 *D*，我们用 *Matches(D R)* 表示 R 在 *D* 中的匹配项集合，即 *D* 中所有标签符合 R 的游走所构成的集合：*Matches(D R)* $=\{ w \in$ *Walks(D)* $\mid$ *Lbl*$_{\mathcal{D}}$*(w)* $\in$ *L(R)* $\}$。给定 v、$v' \in V_{\mathcal{D}}$，我们还将使用记号 *Matches(D R, v, v')* 表示 *Matches(D R)* 中满足 *EndP(w)* $= (v, v')$ 的游走 w 所构成的集合。

3.1 任意语义

▶**定义 2**。一种 语义 S 是一个函数，它将数据库 $\mathcal{D}$ 和查询 R 映射为有限子集 $S(\mathcal{D}\ R)$ 中的 匹配项$(\mathcal{D}\ R)$。

除了不符合我们框架的同态语义（因为它不返回游走）之外，最流行的两种 RPQ 语义是最短游走语义和迹语义。我们将在下文定义它们。

▶ 示例 3（迹语义 Tr）。迹语义是 Cypher [FG+18b] 的语义，也可在 GQL [DF+22; GQL24] 中使用，这要归功于关键字 `TRAIL`。其定义如下。若游走 w 不包含重复边，即对所有 $i \neq j$，$w[i] \neq w[j]$ 均成立，则它是迹。迹语义定义为：Tr(D R) $=\{ w \in$ 匹配项(D R) $\mid w$ 是迹 $\}$。

▶ **示例 4**（最短游走语义 Sh）。最短语义是 GSQL [GSQL23] 和 PGQL [PGQL21]的原始语义；借助关键字 `SHORTEST`，也可在 GQL [DF+22; GQL24] 中使用。设$\mathcal{D}$ 为一个数据库，R 为一个查询。对于每个 $s, t \in V_{\mathcal{D}}$，我们将 Matches($\mathcal{D}$ R, s, t) 中最短游走的长度记作$\ell_{(s,t)}$。于是，最短语义定义为：Sh($\mathcal{D}$ R) $= \bigcup_{s,t \in V_{\mathcal{D}}} \{ w \in$ Matches($\mathcal{D}$ R, s, t) $\mid$ Len(w) $= \ell_{(s,t)} \}$。

我们关注如下文所定义的基于游走的语义。直观地说，这类语义充当匹配项的后置过滤器：它们只能访问已匹配游走的集合本身，而无法访问数据库、正则表达式，也无法获知某条给定游走是因何种原因或以何种方式与查询匹配的。

▶ **定义 5.** 若语义 S 满足 基于游走的，则存在一个函数 r_S：$2^{\mathsf{Walks}} \to 2^{\mathsf{Walks}}$ 使得 $S(D, R) = r_S($匹配项$(D, R))$。

尽管前面的定义看起来非常一般，但根据定义 2，基于游走的语义仍必须输出与查询匹配的游走。基于游走的语义的实例包括上文给出的 Tr 和 Sh，以及下文定义的最短迹语义。另一方面，我们将在第 3.4 节给出几个非基于游走的语义示例，例如绑定轨迹语义和最低成本游走语义。

▶ **示例 6**（最短迹语义 ShT）。最短迹语义可用于 GQL [DF+22; GQL24] 得益于关键短语 `SHORTEST TRAIL`。对于每个 $W \subseteq 2^{\mathsf{Walks}}$，以及任意顶点 s，$t \in \mathbb{V}$，我们令 $\ell_{(s,t)}(W)$ 表示 W 中满足 EndP(w) $= (s, t)$ 的最短迹 w 的长度（若存在）。于是，最短迹语义定义如下：ShT($\mathcal{D}$ R) $= r_{\mathsf{ShT}}($匹配项$(D, R))$，其中 $r_{\mathsf{ShT}}(M)$ 包含 M 中满足 Len(w) $= \ell_{\mathsf{EndP}(w)}(M)$ 的所有迹 w。

对于每个 $\mathcal{D}$和 R，ShT($\mathcal{D}$ R) 显然是 Tr($\mathcal{D}$ R) 的子集，因为它只返回与 R 在 $\mathcal{D}$中匹配的迹。然而，它未必是 Sh($\mathcal{D}$ R) 的子集，因为 $\mathcal{D}$ 中两个顶点之间与 R 匹配的最短游走可能是迹，也可能不是。

3.2 基于过滤器的语义

▶ **定义 7**。 如果存在一个函数 S，则称这种语义为 基于筛选的，该函数 f_S: 游走 $\rightarrow \{\top, \bot\}$，满足 $S(D\ R) = \{\ w \in$ 匹配项$(D\ R) \mid f_S(w) = \top\}$。

请注意，每种基于过滤器的语义也都是基于游走的。迹语义（例 3）是基于过滤器的语义的典型示例。它由函数 f_{Tr} 定义，该函数将游走 w 映射为 $\bot$，当且仅当存在 $i < j$ 使得 $w[i] = w[j]$；否则映射为 $\top$。然而，很容易看出，最短游走语义（例 4）和最短迹语义（例 6）都不是基于过滤器的。下面给出三个基于过滤器的语义示例。

▶ **示例 8** (无环语义 Ac). 无环语义（对应于 GQL 中的 `ACYCLIC`关键字）是由过滤器 f_{Ac} 定义的语义；该过滤器将游走 w 映射为 $\bot$，当且仅当存在 $i < j$ 使得 $w\langle i\rangle = w\langle j\rangle$；否则映射为 $\top$。

▶ **示例 9** （简单环或游走语义 SWC）。简单环或游走语义是 Ac 的一个变体，对应于 GQL 中的关键字 `SIMPLE` 。区别在于，它也会返回简单环。它由过滤器 f_{SWC} 定义：若存在$i < j$使得 $(i,\ j) \neq (0,\ \mathrm{Len}(w))$，且 $w\langle i\rangle = w\langle j\rangle$，则将游走 w 映射为 $\bot$；否则映射为 $\top$。

▶ **示例 10** （二重无环语义 2Ac）。二重无环语义是 Ac 的一个变体，其中每个顶点最多允许出现两次。它由下述过滤器定义。

$$f_{\mathsf{2Ac}}\colon w \mapsto \begin{cases} \bot & \text{if there are } i < j < k \text{ such that } w\langle i\rangle = w\langle j\rangle = w\langle k\rangle \\ \top & \text{otherwise} \end{cases}$$

3.3 基于序的语义

首先，回顾一下关于偏序的几个定义。偏序 是一种二元、传递、反对称且自反的关系。对于任意偏序 $\preceq$，我们用 $\prec$表示其对应的严格偏序。良拟序 定义在集合 X上，也称为良基偏序，是指满足以下条件的偏序 $\preceq$：对于其中取自 x_1的任意无限序列 x_2、……、x_k、……，都存在两个位置 X，使得 $i < j$。也就是说，不存在无限严格递减序列，也不存在相对于 $x_i \preceq x_j$中所有元素两两不可比的无限序列 $\preceq$。众所周知， X 的任意子集相对于其上的良拟序都只有有限个极小元素 X；并且，定义在 X 上的任意良拟序限制到其子集 X 后仍是良拟序。

▶**定义 11** （基于序的语义）。 设 $\preceq$ 是游走上的偏序，使得对每个数据库 $\mathcal{D}$， $\preceq$ 是游走$(\mathcal{D})$上的良基偏序。这样的序称为适用的。基于 $\preceq$的语义，记为 $S_{\preceq}$，对每个数据库 D 和正则表达式 R 定义如下： $S_{\preceq}(D\ R) = \bigcup_{s,t \in V_{\mathcal{D}}} \min_{\preceq}($匹配项$(D\ R, s, t))$

注意，适用序 $\preceq$ 是匹配项$(\mathcal{D}\ R, s, t)$ 上的良基偏序，因为匹配项$(\mathcal{D}\ R, s, t)$ 是游走$(\mathcal{D})$ 的子集。由此可知， $S_{\preceq}(\mathcal{D}\ R)$ 总是有限的。还需注意，只要 w 与 w' 的端点不同，两条游走 w, w' 相对于 $\preceq$ 的排序便无关紧要。对于不一致的游走 w 与 w'，情况同样如此。有时，从所考虑的序中剔除这类无关对是有用的。

▶**定义 12**。对于每个适用的序， $\preceq$ 我们令 $\preceq_t$ 表示由以下规则定义的序： $w \preceq_t w'$当且仅当 w 且 w' 是端点相同且满足 $w \preceq w'$的相容游走。我们称 $\preceq$是修剪的，如果它等于 $\preceq_t$

▶**引理 13**。 每个基于序的语义 $S_{\preceq}$ 都与 $S_{\preceq_t}$相同。

基于序的语义的一个简单示例是最短游走语义 Sh（例 4）。它基于偏序 $\preceq_{\mathsf{Sh}}$，即由 $w \prec_{\mathsf{Sh}} w'$ （当 Len(w) < Len(w') 时定义）的严格偏序的自反闭包。它在游走上不是 良基偏序：由于 $\mathbb{E}$ 是无限的，给定长度的游走有无穷多个，并且相对于 $\prec_{\mathsf{Sh}}$它们彼此都不可比较。然而，可以验证，对于每个数据库$\mathcal{D}$，它确实是 Walks($\mathcal{D}$) 上的良基偏序，因此是适用的。现在让我们定义另一种基于序的语义。

▶ **定义 14** （子游走序）。 设 F 为以下因子插入二元关系：$w\ F\ w'$若存在三个游走 w_1、w_2、w_3使得 $w = w_1 \cdot w_3$且 $w' = w_1 \cdot w_2 \cdot w_3$。我们将 子游走序 $\sqsubseteq$ 定义为 F的自反传递闭包。

不难验证 $\sqsubseteq$ 是 游走上的偏序。此外，可以证明 $\sqsubseteq$ 是一个适用的序。我们称 子游走极小，并用 SM 表示基于 $\sqsubseteq$的语义。

▶**引理 15**。 对于每个数据库 D，$\sqsubseteq$ 是游走（D）上的良基偏序

引理 15 可由希格曼引理的一个推论得出 [Hig52] （另见 [Kar15，第 2.2] 节）。两者陈述如下。

▶**引理 16** （希格曼引理 [Hig52]）。 若 $\preceq$ 是某个集合 X上的良基偏序，则 $\preceq$ 在 X^*上 *是一个良基偏序，其中 $\preceq$ 是 $\preceq$在序列上的扩展，也就是说，$a_1 \cdots a_k \preceq b_1 \cdots b_\ell$ 当且仅当存在一个严格递增函数 φ: $\{1, \ldots, k\} \to \{1, \ldots, \ell\}$，使得对于每个 $\{1, \ldots, k\}$，都有 $a_i \preceq \varphi(a_i)$。

请注意，希格曼引理通常针对良拟序来表述，这类序不要求是反对称的。该结论对于良基偏序同样成立，且证明完全相同。我们关注引理 16 的一个特殊情形，表述如下：即 X 是有限集，且 $\preceq$ 是 X上的相等关系。

▶**推论 17**。 设 X 为有限集，则 X^* 上的子序列关系是良基偏序。

引理 15 的证明。设 D 为一个数据库。固定 $X = V_{\mathcal{D}} \cup E_{\mathcal{D}}$，它是一个有限集，并令 $\preceq$表示 X^*上的子序列关系。推论 17 表明，$\preceq$ 在 X^*上是一个良基偏序。请注意，Walks(D) $\subseteq X^+$，并且可以验证 $\sqsubseteq$ 实际上是 $\preceq$的限制，即等于 $\{(u, v) \in \text{Walks(D)} \times \text{Walks(D)} \mid u \preceq v\}$。特别地，这意味着子游走序 $\sqsubseteq$ 在 Walks(D) 上是一个良基偏序。◀

考虑子游走序 $\sqsubseteq$ 及其相关语义 SM 的原因，源于子游走覆盖保证这一性质，我们将在后文对其进行定义。关于 SM 下查询求值复杂度的初步结果，可见 [Khi24], ，并在图 2（第 25 页）中重述。作为本节的结尾，我们来提及另一种有趣的基于序的语义。

▶ **例 18** （短字典序语义 ShL）。 令 $\preceq_{\text{lex}}$ 为基于 $\mathbb{V} \cup \mathbb{E}$上某个任意序的游走字典序。令 $\preceq_{\text{shortlex}}$ 为相应的长度字典序：若 $|w| < |w'|$，或（$|w| = |w'|$ 且 $w \preceq_{\text{lex}} w'$），则 $w \preceq_{\text{shortlex}} w'$。尽管 $\preceq_{\text{lex}}$不适用，但可以验证 $\preceq_{\text{shortlex}}$ 适用。相应的短字典序最小语义 ShL，是 GQL [DF$^+$22 中 `ANY SHORTEST` 关键短语的一种可能实现；GQL24]。

3.4 一些范围外语义的示例

本文关注输出游走（定义 3）且基于游走（定义 5）的语义，但这当然不是定义有意义的正则路径查询语义的唯一可能方法。科学文献中对正则路径查询研究最多的语义是同态语义 [AA+17]，它只输出匹配项的端点，因此不 符合我们对语义的理解。另一方面， [DFM23] 提出了基于运行的 语义，即语义 S 需要了解游走与查询匹配的方式，才能判断该游走是否属于 S（$\mathcal{D}$ R）。下面我们将介绍一种这样的基于运行的语义。之后，我们将定义另一种非基于游走的语义。它大致受到关键词 `CHEAPEST`的启发；该关键词曾在 GQL 制定期间的内部文档中讨论过，但未被纳入第 1 版（参见 [PGQ23 中的语言机会 052；GQL24]）。

▶ 示例 19（绑定轨迹语义 BT [DFM23]）。设 D 为一个数据库， R 为一个查询， k为 R中原子的数量。绑定游走（在 D 中匹配 R）是一个游走 $w \in$ Matches(D R)，以及一个函数 {1, . . . , Len(w)} $\to$ {1, . . . , k}，该函数将 w 中的每个边位置映射到 R 中与该边匹配的原子的位置。绑定轨迹（在 D 中匹配 R）是一个绑定游走（w, α），满足不存在互不相同的 i, j 使得 $\alpha(i) = \alpha(j)$且 $w[i] = w[j]$。绑定轨迹语义记作 BT，定义如下。

$$\text{BT} : (\mathcal{D}\ R) \mapsto \{\ w \mid \exists \alpha \text{ 满足条件：} (w, f) \text{ 是 D 中匹配 } R\} \text{的绑定轨迹} \tag{2}$$

更正式的定义见 [DFM23]。同一篇论文还定义了一种类似的语义，称为简单运行语义，它基于表示查询的有限自动机的状态。

▶ **例 20** (最低成本游走语义 ChW$_c$)。 设 c: $\mathbb{L} \to (\mathbb{N} - \{0\})$ 是一个固定的成本函数，并按以下方式将其扩展到词： $c(a_1 \cdots a_k) = \sum_{i=1}^{k} c(a_i)$。于是，数据库 $\mathcal{D}$ 和查询 R 中的游走，其成本就是该游走在$\mathcal{D}$中的标签的成本。最低成本游走语义 ChW$_c$的定义与最短游走语义类似，只是使用匹配项的成本，而非其长度。

请注意，ChW$_c$ 并非基于游走的（除非成本函数 c 为常数），因为游走无法提供其边所带标签的信息。

4 性质

本节定义并探讨 RPQ 语义的若干可能性质，以帮助对其进行比较和分类。在列出一些基本性质后，我们将讨论与单调性、连续性、有理算子的兼容性以及覆盖相关的性质。接着，我们系统地考察同一查询在不同语义下结果集之间的包含关系。本节最后将一般性地讨论任何“合理”的语义应具备的性质。

4.1 基础知识

我们首先列出一些基本性质，这些性质几乎总是我们的模型中一种语义被认为有用所必需的。

▶ 定义 *21*。若对每一个重命名 ν、数据库 *D* 以及正则表达式 R，都有 $\nu(S(D\ R)) = S(\nu(D)\ R)$，则称语义 S 为标识符无关。类似地，若对每一个重新标记 λ、数据库 *D* 以及正则表达式 R，都有 $S(D\ R) = S(\lambda(D)\ \lambda(R))$，则称其为标签无关。

例如，一种根据顶点和边标识符上的某种任意序来选择结果的语义（例如，例 18 中的 ShL），或基于类似任意标准的存在性来选择结果的语义，就不会是标识符无关的。标识符无关性与经典数据库环境中的泛型性相关（例如参见 [AHV95]）。事实上，若考虑一个由正则表达式以及用于解释它的语义 S 构成的查询 q，则 S 是标识符无关的，当且仅当 q 是 $\emptyset$-泛型的。

同样，标签依赖语义会根据有关标签的任意信息来选择其结果。此外，正如下文所述，不存在基于游走的标签依赖语义。

▶**引理 22.** 每种基于游走的语义都是标签无关的。

证明。根据定义，对于任意基于游走的语义 S，都有 $S(\mathrm{D}\ R) = r_S(\text{匹配项}(\mathrm{D}\ R))$，其中某个函数 r_S 满足此关系。然而，由于匹配项(D R) 不保留游走标签的任何信息，对于任意重新标记 λ，匹配项(λ(D) $\lambda(R)$) = 匹配项(D R)。因此，$S(\mathrm{D}\ R) = S(\lambda(\mathrm{D})\ \lambda(R))$。◀

除非另有明确说明，本文始终讨论标识符与标签无关语义。

▶**定义 23.** 若一种语义 S 是 表达式无关 的，则对于每个数据库 $\mathcal{D}$ 以及每个表达式 R_1，R_2 满足 $L(R_1) = L(R_2)$时，都有 $S(\mathcal{D}\ R_1) = S(\mathcal{D}\ R_2)$。

表达式无关语义仅根据正则表达式所表示的语言来选择结果。在某些情况下，让结果的选择取决于表达式本身的结构或许更合理。绑定轨迹语义（例 19）便是这样一个例子。不过，这超出了本文的讨论范围，因为如下所述，所有基于游走的语义都是表达式无关的。

▶**引理 24.** 每一种基于游走的语义都是表达式无关的。

证明。 与引理 22 类似。若 $L(R_1) = L(R_2)$ 则根据 匹配项$(\mathcal{D}\ R_1)$ = 匹配项$(\mathcal{D}\ R_2)$ 的定义，匹配项。因此 $S(\mathcal{D}\ R_1) = S(\mathcal{D}\ R_2)$。◀

▶**定义 25.** 若语义 S 称为无界，则对于每个整数 n，都存在一个数据库 D 和一个正则表达式 R，使得 $S(D, R)$ 包含一条长度至少为 n 的游走。否则，S 称为有界。

当然，我们预计任何合理语义都是无界的，因为有界语义永远不会返回长度超过某个常数的游走。

4.2 单调性

▶ 定义 *26*（单调性、余单调性和可限制性）。如果对于满足 $D \subseteq D'$ 的任意正则表达式 R 以及数据库 D、D'，语义 S 满足 $S(D\ R) \subseteq S(D', R)$，则称其为单调的。如果对于满足 $D \subseteq D'$ 的任意正则表达式 R 以及数据库 D、D'，其满足 $S(D\ R) \supseteq (S(D', R) \cap Walks(D))$，则称其为协单调。如果二者均成立，则称其为可限制的。

这些概念描述的是数据库添加新顶点和边后，语义的行为。单调语义不会在数据库扩充后移除结果。协单调语义不会添加与新元素无关的结果。最后，可限制的语义意味着，给定数据库的一部分，就能够计算出结果的一部分。简单验证可知，以上定义的所有语义都是协单调的。事实上，所有基于过滤器的语义和基于序的语义都具有这一性质。此外，基于过滤器的语义也是单调的，因此也是可限制的。

10 设计与比较 RPQ 语义

▶**引理 27.** 每种基于过滤器的语义都是可限制的，每种基于序的语义都是协单调的。

基于过滤器的语义的证明。设 S 是由过滤函数f_S 定义的基于过滤器的语义（见定义 7）。设 R 是一个正则表达式，且$\mathcal{D}$，$\mathcal{D}'$ 是两个数据库，满足$\mathcal{D} \subseteq \mathcal{D}'$。我们有：

$$S(\mathcal{D}', R) = \{\, w \in \mathsf{Matches}(\mathcal{D}', R) \mid f_S(w) = \top \,\}$$

因此：

$$S(\mathcal{D}', R) \cap \mathsf{Walks}(\mathcal{D}) = \{\, w \in \mathsf{Matches}(\mathcal{D}, R) \mid f_S(w) = \top \,\} = S(\mathcal{D}, R)$$ ◀

基于序的语义的证明。设 S 是由序$\preceq$ 定义的基于序的语义（见定义 11）。设 R 是一个正则表达式，且 D、D′ 是两个数据库，满足 D $\subseteq$ D′。设 w 是 S(D′, R) $\cap$ Walks(D) 中某些顶点 s 和 t 之间的一条游走。根据定义， $w \in$min$\preceq$(匹配项 (D′, R, s, t))。但由于 $w \in$ Walks(D)，我们也有 $w \in$ 匹配项(D R)。匹配项(D R) 中不可能存在 $w' \prec w$，否则 w 在匹配项(D′, R, s, t) 中也不会是极小的。因此 $w \in$ min$\preceq$(匹配项(D R, s, t))，换言之 $w \in S$(D R)。◀

我们认为，所有“合理”的语义都应当是协单调的。事实上，那些 e 不具备协单调性的语义通常相当奇怪，如下文所示。

▶ **例 28** （对数长度语义）。对数长度语义 LL 返回所有长度小于 log$_2$(|$\mathcal{D}$|) 的匹配项。该语义具有这样的性质：LL($\mathcal{D}$ R) 的大小始终是$\mathcal{D}$ 大小的多项式。它不具备协单调性，因为将 $\mathcal{D}$ 的大小加倍（例如，通过添加足够多的孤立顶点）可能会返回更长的游走，而这些游走未必经过新增的孤立顶点。不过请注意，LL 是单调的，因为扩充数据库时，先前的结果不会消失。

虽然 LL 不是基于游走的语义，但我们可以考虑一个基于游走的变体。具体而言，它可以返回长度小于 log$_2$(|X($\mathcal{D}$ R)|) 的游走，其中 X($\mathcal{D}$ R) 是 Matches($\mathcal{D}$ R) 中某个游走所包含的所有边的集合。

至于单调性，情况则更加有趣。我们已经看到，所有基于过滤器的语义都是单调的（引理 27）。不过，值得注意的是，最短语义并非如此。

▶**引理 29.** Sh 不是单调的。

^{a}证明。设 D 是一个由三个顶点 v_1、 v_2 和 v_3 以及两条边 v_1 —A$_{e_1}$ v_2 和 v_2 —^{a}A$_{e_2}$ v_3 组成的数据库，并设 $R = aa + b$，其中 a、$b \in \mathbb{L}$。因此 S(D R) ={v_1 —A$_{e_1}$ v_2 —A$_{e_2}$ v_3}。接着，令 D′ 为数据库 D′ ，它是在 D 中添加一条新的边 v_1 —^{b}A$_{e_2}$ v_3 后得到的。因此 S(D′, R) ={v_1 —^{b}A$_{e_2}$ v_3}，这并不是 S(D R) 的超集。◀

乍看之下，所有基于序的语义 $S_\preceq$ 似乎都必然是非单调的，因为只需使用新的顶点和边，就能构造一条捷径（相对于 $\preceq$）。然而，可以证明，（基于序的）子游走最小语义 SM（定义 14）确实是单调的。更一般地，下面我们将利用元素包含这一概念来刻画单调的基于序的语义（命题 30）。下文中，我们用 VertSet(w)（分别用 EdgeSet(w)）表示 w 中出现的顶点（分别是边）的集合。我们还用 ElemSet(w)（分别用 ElemBag(w)）表示 w 中出现的顶点和边的集合（分别是多重集）。如果 $w \preceq w'$，则称游走上的序 $\preceq$ 尊重元素包含关系：ElemSet(w) $\subseteq$ElemSet(w') 。

▶**命题 30**。设 $S_{\preceq}$ 是基于修剪序 $\preceq$ 的语义。那么， $\preceq$ 尊重元素包含关系，当且仅当 $S_{\preceq}$ 是单调的。

命题 30 正向方向的证明。为导出矛盾，假设存在一个查询 R、两个数据库 D $\subseteq$ D′，以及一个属于 S(D R) 但不属于 S(D′, R) 的游走 w。这意味着存在一个满足 $w' \prec w$ 的游走 $w' \in S(\mathrm{D}', R)$。根据假设可得 ElemSet(w') $\subseteq$ ElemSet(w)，因此 $w' \in$ Walks(D)。由于 S 是协单调的（引理 27）， $w' \in S$(D R)。因此 $w \notin S$(D R)，矛盾。◀

反向方向的证明需要一些额外的工具（定义 31），这些工具在今后的证明中也会派上用场。

▶ **定义 31** （特征数据库、特征表达式）。 为简便起见，以下我们假定 $\mathbb{E} \subseteq \mathbb{L}$，即边标识符可以作为标签。若要严格地表述，我们需要使用某个任意的双射 $\mathbb{E} \to \mathbb{L}$，但我们倾向于将其隐去。

1. 设 $w_1, \ldots, w_k$ 是相容游走。设 $\mathcal{D}$ 是任意一个数据库，使得 $\{w_1, \ldots, w_k\}$ $\subseteq$Walks($\mathcal{D}$)，并且

$$V = \mathsf{VertSet}(w_1) \cup \cdots \cup \mathsf{VertSet}(w_k)$$
$$E = \mathsf{EdgeSet}(w_1) \cup \cdots \cup \mathsf{EdgeSet}(w_k)$$

注意 $V \subseteq V_{\mathcal{D}}$和 $E \subseteq E_{\mathcal{D}}$。该特征数据库由 $\{w_1, \ldots, w_k\}$ 构成，是数据库

$$\mathcal{C}_{w_1,\ldots,w_k} = (L, V, E, \mathsf{Src}, \mathsf{Tgt}, \mathsf{Lbl})$$

其中 $L = E$，Lbl 是恒等函数，而Src （分别为Tgt）是将Src$_{\mathcal{D}}$（分别为Tgt$_{\mathcal{D}}$）限制在 E 上所得的函数。
（可以验证，$\mathcal{C}_{w_1,\ldots,w_k}$ 与所选数据库 D 无关。）

2。 设 w 是一个长度为正的游走 k。我们将 P_w 称为 w的特征正则表达式，定义为 $P_w = w[1] \cdots w[k]$。

直观而言，有限个相容游走的特征数据库就是包含它们的最小数据库，即$\{w_1, \ldots, w_k\}$ $\subseteq$ Walks($\mathcal{C}_{w_1,\ldots,w_k}$)，其中边以其自身的标识符作为标签。游走的特征表达式就是其边标识符序列。

这些定义的选取方式保证：每个正长度的 w_i 的特征表达式都恰好匹配一次（匹配项 ($\mathcal{C}_{w_1,\ldots,w_k}, P_{w_i}$) $= \{w_i\}$），而任何其他正长度游走的特征表达式要么匹配一次，要么完全不匹配：匹配项($\mathcal{C}_{w_1,\ldots,w_k}, P_w$) 要么为空，要么等于 $\{w\}$。

现在我们可以证明，如果一种基于序的语义是单调的，那么相应的序会保留元素包含关系。

命题 30 的反向证明。 设 $S_{\preceq}$ 是基于修剪序 $\preceq$的单调语义。

设 w 和 w' 是两个游走，且 $w \prec w'$。首先，考虑 Len(w) =0 的情况，此时 ElemSet(w) = $\{v\}$，其中 v是某个顶点。由于 $\preceq$ 是修剪的， w 和 w' 具有相同的端点。因此， v = Src(w') = Tgt(w')，故 ElemSet(w) $\subseteq$ ElemSet(w')。

其次，假设 Len(w) > 0。由于 $\preceq$ 是修剪的，w 和 w' 是一致的，这意味着特别是两个数据库$\mathcal{C}_{w'}$ 和 $\mathcal{C}_{w,w'}$ 是定义良好的。此外，我们考虑以下查询

$$R = \begin{cases} P_w + P_{w'} & \text{if } \mathsf{Len}(w') > 0 \\ P_w + \varepsilon & \text{if } \mathsf{Len}(w') = 0 \end{cases} \tag{3}$$

注意，匹配项($\mathcal{C}_{w,w'}, R$) $\supseteq \{w, w'\}$，并且当 Len(w') =0 时，这个包含关系可能是严格的，因为 ε 会匹配每个长度为 0 的游走。由于 $w \prec w'$，这意味着 $w' \notin S$($\mathcal{C}_{w,w'}, R$)，因此，根据 S的单调性，

$$w' \notin S_{\preceq}(\mathcal{C}_{w'}, R)\,. \tag{4}$$

如果 w' 的长度为 0，那么 $\mathsf{C}_{w'}$ 将只包含一个孤立顶点。因此，由于 Len(w) > 0，w 不是 $\mathsf{C}_{w'}$中的游走。因此，我们有匹配项($\mathsf{C}_{w'}, R$) $=\{w'\}$，进而$w' \in S_{\preceq}$($\mathsf{C}_{w'}, R$)，这与 (4) 矛盾。由此可知，w' 必须具有正长度，这意味着匹配项($\mathsf{C}_{w,w'}, R$) $=\{w, w'\}$。由于匹配项($\mathsf{C}_{w'}$, R) $\subseteq$匹配项($\mathsf{C}_{w,w'}, R$) $=\{w, w'\}$，方程 (4) 表明 w必须属于匹配项($\mathsf{C}_{w'}, R$)，因而属于 Walks($\mathsf{C}_{w'}$)。根据特征数据库的定义，仅当 ElemSet(w) $\subseteq$ ElemSet(w')时，这才成立。◀

考虑关系 $\preceq_1$，其定义如下：$w \preceq_1 w'$ 当且仅当 ElemSet(w) $\subseteq$ElemSet(w')。它不是反对称的，因此不是偏序。现在考虑关系 $\preceq_2$，其定义如下：$w \prec_2 w'$当且仅当 ElemSet(w) $\subsetneq$ ElemSet(w')。虽然 $\preceq_2$ 是偏序，但它**不** 适用：只要$\mathcal{D}$ 包含一个环，$\preceq_2$ 在 Walks(D) 上就不是良基偏序。下面给出两个基于保留元素包含关系的序的语义示例。例32考虑了重数。例33通过为元素集相同的游走添加决胜规则，细化了 $\preceq_2$ 。

▶ **例32**（最小多重集语义 MM）。令 MM 为基于序 $\preceq$ 的语义，其定义如下：$w \prec w'$ 当且仅当 ElemBag(w) $\subsetneq$ ElemBag(w')。回顾一下，$\sqsubseteq$ 表示游走上的子游走序（定义14）。简单验证可得 $w \sqsubseteq w' \implies w \preceq w'$。于是，对每个数据库$\mathcal{D}$，$\preceq$ 都是 Walks($\mathcal{D}$) 上的良基偏序，因为 $\sqsubseteq$ 也是良基偏序（引理15）。确实，设 $(w_1, w_2, \ldots)$ 是$\mathcal{D}$中的一个无限游走序列。由于 $\sqsubseteq$ 在$\mathcal{D}$上是良基偏序，存在 $i < j$使得 $w_i \sqsubseteq w_j$，这意味着 $w_i \preceq w_j$。

▶ **例33**（最短极小集语义 ShMS）。令 ShMS 为基于由下式定义的序 $\preceq$ 的语义：

$$w \prec w' \iff \begin{cases} \text{either } \mathsf{ElemSet}(w) \subsetneq \mathsf{ElemSet}(w') \\ \text{or } \mathsf{ElemSet}(w) = \mathsf{ElemSet}(w') \text{ and } \mathsf{Len}(w) < \mathsf{Len}(w') \end{cases}$$

语义 MM 和 ShMS 是有限的、标识符无关的且可限制的。在它们下的查询求值复杂度已在 [Mar24] 中探讨，并在图 2、第 25 页中回顾。

4.3 连续性

在本节中，我们尝试刻画一种语义 S 在极限处表现得“合理”的性质。本质上，当 S趋近于 ∞ 时，序列 $\left(S(\mathcal{D}\ a + a^2 + \cdots + a^k)\right)_{k\in\mathbb{N}}$应趋近于 $S(\mathcal{D}\ a^*)$。更准确地说，正则路径查询的非递减序列（相对于语言包含关系）的结果最终应稳定为某个固定集合 W；此外，只要整个序列可以由单个正则路径查询表示，那么该正则路径查询在该

正则路径查询上的结果集合就应当是 W 本身。这些条件可以防止 S 在趋近极限时突然改变方向，任意改变结果集合。正如我们将在下文展示的，所有基于筛选的语义和所有基于序的语义都具有这些性质。

▶**定义34.** 若一种语义 S 满足以下条件，则称其为 稳定化的：对于每个数据库 $\mathcal{D}$ 以及每个正则表达式序列 $(R_i)_{i\in\mathbb{N}}$，若 $(L(R_i))_{i\in\mathbb{N}}$ 是非递减的，则 $(S(\mathcal{D}\ R_i))_{i\in\mathbb{N}}$最终等于某个集合 W；并称 $(S(\mathcal{D}\ R_i))_{i\in\mathbb{N}}$稳定化至 W。

此外，当 S 是稳定化的时，若满足以下条件，则称其为 连续的：只要存在一个如上所述的正则表达式序列 $(R_i)_{i\in\mathbb{N}}$ 以及一个正则表达式 R_∞，使得 $L(R_\infty) = \bigcup_{i\in\mathbb{N}} L(R_i)$，就有 $W = S(\mathcal{D}\ R_\infty)$。

▶**命题35.** 所有基于序的语义和基于过滤器的语义都是连续的。

基于过滤器的语义的证明。设 S 为基于过滤器的语义 f，并设$\mathcal{D}$ 为一个数据库。将每个 F 映射到 $W \subseteq$ 游走$(\mathcal{D})$ 的$\{w \in W \mid f(w) = \top\}$ 的函数是单调的。另请注意， $F($游走$(\mathcal{D}))$ 是有限的，否则 $S(\mathcal{D}\ \Sigma^*)$ 就不会是有限的。因此， S 是稳定化的且连续的。◀

基于序的语义的证明需要更加谨慎。我们首先证明所有基于序的语义都是稳定化的（引理36）。

▶**引理36**。所有基于序的语义都是稳定化的。

证明。考虑一个基于序的语义 $S_{\preceq}$、一个序列 $(R_i)_{i\in\mathbb{N}}$ （如定义34的陈述所述）以及一个数据库$\mathcal{D}$，并设 M_i = 匹配项$(\mathcal{D}\ R_i)$ 和 $S_i = S_{\preceq}(\mathcal{D}\ R_i)$。由于$(M_i)_{i\in\mathbb{N}}$ 是非递减的，且根据 S_i的定义，我们有

$$\begin{array}{ccccccccc} M_0 & \subseteq & M_1 & \subseteq & \cdots & \subseteq & M_i & \subseteq & \cdots \\ \cup| & & \cup| & & & & \cup| & & \\ S_0 & & S_1 & & & & S_i & & \end{array} \tag{5}$$

$\bigcup_{j=0}^{i} S_j \subseteq \bigcup_{j=0}^{i} M_j \subseteq M_i$这尤其意味着，对于每个 $i \in \mathbb{N}$，都有 $\cdot$

首先，我们断言，对于每个满足 $S_{i+1} \neq S_i$ 的 i，都存在一条游走 $w \in (S_{i+1} \setminus S_i)$。由于 $S_{i+1} \neq S_i$，我们有 $(S_{i+1} \setminus S_i) \neq \emptyset$，或者 $(S_i \setminus S_{i+1}) \neq \emptyset$。在第一种情况下，断言立即成立。在第二种情况下，设 w' 是 $S_i \setminus S_{i+1}$ 中的一条游走。由于 $w' \in S_i \subseteq M_i \subseteq M_{i+1}$，但 $w' \notin S_{i+1}$，因此必定存在一条游走 w，它属于 S_{i+1} 且满足 $w \prec w'$。这条游走不可能属于 S_i，否则 w' 也不会属于其中。至此，断言得证。

其次，我们断言，对于满足 $w \in (S_{i+1} \setminus S_i)$ 的每个整数 i 和游走 w，不存在满足 $w' \preceq w$ 的 $w' \in \bigcup_{j=0}^{i} S_j$。根据定义，由于 $w \in S_{i+1}$，因此 w 是 M_{i+1} 中的极小元素。由 (5) 可知，对于每个 $j \leq i$ （$*$），w 与 M_j（分别与 S_j）中的每个元素都不可比较或小于该元素。由此可知，游走 w 不可能属于 M_i，否则它也会属于 S_i，这与其定义矛盾。接着，由 (5) 可推出，对于所有 $j \leq i$， $w \notin S_j$ (**)。至此，断言得证：任何符合断言条件的 w'，要么等于 w，这与 (**) 矛盾；要么不同于 w，这与 (*) 矛盾。

最后，如果 $(S_i)_{i\in\mathbb{N}}$ 并非最终为常量，我们就可以对满足 $S_{i+1} \neq S_i$ 的指标 i 应用上述两个论断。由此可得游走序列 $(w_k)_{k\in\mathbb{N}}$，对于每个 $k < k'$，都有**不**成立 $w_k \preceq w_{k'}$，这与 $\preceq$ 在 Walks($\mathcal{D}$) 上是良拟序这一事实矛盾。

◀

基于序的语义下命题35的证明。设 $S_{\preceq}$ 是一种基于序的语义，D 是一个数据库，$\big(R_i\big)_{i\in\mathbb{N}}$ 是正则表达式序列，使得 $\big(L(R_i)\big)_{i\in\mathbb{N}}$ 是非递减的，并且 R_∞ 和表达式满足 $L(R_\infty) = \bigcup_{i\in\mathbb{N}} L(R_i)$。由于 $S_{\preceq}$ 是稳定化的，因此存在集合 W 和上界 b，使得对每个 $n > b$，都有 $S(\mathrm{D}\ R_n) = W$。我们需要证明 $S(\mathrm{D}\ R_\infty) = W$；这可由以下两个陈述对每个游走 w 均等价这一事实推出：

- $w \in$ 匹配项$(\mathcal{D}\ R_\infty)$，并且存在 $w' \in$ 匹配项$(\mathcal{D}\ R_\infty)$，使得 $w' \prec w$；存在一个索引 $n > b$
- 和一条游走 w'，使得 $w \in$ 匹配项$(\mathcal{D}\ R_n)$、$w' \in$ 匹配项$(\mathcal{D}\ R_n)$，并且 $w' \prec w$。

这一等价性可由 R_i's 和 R_∞ 的定义推出。 ◀

我们认为，任何"合理的"（基于游走的）语义都应当是连续的。事实上，非连续的基于游走的语义必须区别对待有限集和无限集。为了让读者对非连续语义可能呈现的形式有所了解，下面给出两个玩具示例：一个非稳定化的语义，以及一个虽稳定化但非连续的语义。

▶ **示例 37**（放弃语义）。本文介绍的大多数语义都旨在：对于每一对存在匹配的顶点，至少返回一条游走，即使实际上存在无限多个这样的匹配项。与此相反，令 GU$(\mathcal{D}\ R)$ 表示这样一种语义：如果集合 Matches$(\mathcal{D}\ R)$ 是有限集，则返回整个集合；否则返回空游走集。这意味着 GU$(\mathcal{D}\ R)$ 始终是有限集。

给定表达式 $\big(R_i = a^{\leq i}\big)_{i\in\mathbb{N}}$（每个表达式都匹配由某个满足 $j \leq i$ 的词 a^j 标记的所有游走），如果 $\mathcal{D}$ 包含一条从某个顶点 v 到其自身、带有 a 标签的环 e，那么每个 GU$(\mathcal{D}\ R_i)$ 都包含这条新游走 $(ve)^i v$。因此，GU 显然不是稳定化的（因而也不是连续的：注意，在上述情形下，GU$(\mathcal{D}\ a^*) = \emptyset$）。

GU 似乎过于古怪，难以令人感兴趣。不过，可以定义一些仍具有有趣性质的变体。例如，考虑语义 GU′，其定义如下。给定一组元素 X 和两个顶点 s 与 t，我们将 Matches$_X(\mathcal{D}\ R,\ s,\ t)$ 表示为 $\mathcal{D}$ 中满足 EndP$(w) = (s,\ t)$ 且 ElemSet$(w) \subseteq X$ 的 R 的匹配项集合。令 $\mathbb{T}$ 为所有满足 Matches$_X(\mathcal{D}\ R,\ s,\ t)$ 有限的三元组 $(X,\ s,\ t)$ 所构成的集合。换言之，如果某游走的元素集支持无穷多个其他匹配项 R，则不会将其作为结果返回。于是，

$$\mathsf{GU}'(\mathcal{D}, R) = \bigcup_{(X,s,t)\in\mathbb{T}} \mathsf{Matches}_X(\mathcal{D}, R, s, t)\,. \tag{6}$$

由于 $\mathbf{T}$ 是有限的，按照定义，这种语义是有限的。可以证明，它实际上也是基于游走的且可限制的。

稳定但非连续的语义更加奇特。典型的例子是一种在存在无限集合时会改变行为的语义。

▶ **例 38**（稳定的非连续语义）。设 S 为如下定义的语义：

$$S(\mathcal{D}, R) = \begin{cases} \mathsf{Sh}(\mathcal{D}, R) & \text{if } \mathsf{Matches}(\mathcal{D}, R) \text{ is finite,} \\ \mathsf{Tr}(\mathcal{D}, R) & \text{if } \mathsf{Matches}(\mathcal{D}, R) \text{ is infinite.} \end{cases} \tag{7}$$

它是稳定化的，因为对于每个正则表达式序列 $\big(R_i\big)_{i\in\mathbb{N}}$，只要$\big(L(R_i)\big)_{i\in\mathbb{N}}$ 是非递减的，序列 (匹配项$(\mathcal{D}\ R_i)$)$_{i\in\mathbb{N}}$ 要么只包含有限集，要么存在一个上界 b，使得仅当 $i \leq b$ 时，匹配项$(\mathcal{D}\ R_i)$ 才是有限的。在这两种情况下，$(S(\mathcal{D}\ R_i))_{i\in\mathbb{N}}$ 都会稳定化，因为 Sh 和 Tr 都会稳定化。它不是连续的，这可以通过考察序列 $(a^{\leq i})_{i\in\mathbb{N}}$ 和极限表达式 a^*看出。

其他稳定的非连续语义示例，可以通过将其他语义变体加入关于匹配项集合无限性的条件来构造。例如，给定任意一种基于序的语义 $S_{\preceq}$，下面定义的语义 $S'_{\preceq}$ 只返回那些相对于 $\preceq$而言小于无穷多个游走的游走：

$$S'_{\preceq}(\mathcal{D}, R) = \{w \in S_{\preceq}(\mathcal{D}, R) \mid \exists \text{ infinitely many } w' \in \mathsf{Matches}(\mathcal{D}, R) \text{ such that } w \preceq w'\}$$

由于 $S_{\preceq}$ 具有稳定化性质，因此它也具有稳定化性质；而通常的表达式序列 $(a^{\leq i})_{i\in\mathbb{N}}$ 及其极限表达式 a^* 表明它不是连续的。

4.4 与有理算子的兼容性

本节研究正则路径查询作为正则表达式的结构与结果集之间的相互作用。我们特别定义语义相对于给定正则路径查询算子的可组合性和可分解性。以下以连接为例，说明可组合性背后的直观含义。如果对于 $R \cdot R'$ 的任意结果（即 $S(D, R \cdot R')$ 中的一条游走 w），都可以将其分解为 R 的一个结果和 R' 的一个结果，那么该语义就是·-可分解的。这意味着存在 $w' \in S(D, R)$ 和 $w'' \in S(D, R')$，使得 $w = w' \cdot w''$。反过来，如果语义 S 对 R 和 R' 的结果可以自由组合成 $R \cdot R'$ 的（某些）结果，那么该语义就是·-可组合的。下文将更详细地定义这些概念。

▶**定义 39**。我们称 S 是·-可组合的(分别是·-可分解的），如果下文中的公式(8)(分别是公式(9)）对每个数据库$\mathcal{D}$ 和正则路径查询 R、 R'都成立。

$$S(\mathcal{D}, R \cdot R') \supseteq S(\mathcal{D}, R) \cdot S(\mathcal{D}, R') \tag{8}$$
$$S(\mathcal{D}, R \cdot R') \subseteq S(\mathcal{D}, R) \cdot S(\mathcal{D}, R') \tag{9}$$

如果(8) 和(9) 都成立，我们就称 S 是与·兼容·，简称·兼容的。同样，我们根据下文中的 $+$ 定义可组合性、 $+$可分解性和 $+$兼容性。(10)来定义。

$$S(\mathcal{D}, R + R') \supseteq S(\mathcal{D}, R) \cup S(\mathcal{D}, R') \quad \textit{and/or} \quad S(\mathcal{D}, R + R') \subseteq S(\mathcal{D}, R) \cup S(\mathcal{D}, R') \tag{10}$$

以及 $*$可组合性、 $*$可分解性、 $*$兼容性，依据下文中的(11) 定义。

$$S(\mathcal{D}, R^*) \supseteq (S(\mathcal{D}, R))^* \quad \textit{and/or} \quad S(\mathcal{D}, R^*) \subseteq (S(\mathcal{D}, R))^* \tag{11}$$

最后，我们称一种语义是可分解的，如果它相对于·是可分解的， $+$ 以及 $*$。

▶ **定义 40**。我们称语义 S 是与原子兼容，或称 a-兼容的，如果 $S(\mathcal{D}\ a) =$匹配项$(\mathcal{D}\ a)$，对每个 $a \in \mathbb{L}$ 和数据库$\mathcal{D}$都成立。

下面给出若干关于与有理算子兼容性的结论。首先，引理 41 指出，所有基于筛选的语义和基于序的语义都是 $+$-可分解的，并且 $+$-可组合性本质上等价于基于筛选。

▶**引理 41** （与 $+$ 兼容性）。 1. 每种基于过滤器的语义都与 $+$ 兼容。
2. 每一种基于序的语义都是 $+$-可分解的，但不是 $+$-可组合的。3. 对于每一种 $+$-兼容语义 S，都存在一种基于过滤器的语义 S'，使得对于每个数据库$\mathcal{D}$、正则表达式 R 和正长度游走 w： $w \in S(\mathcal{D}\ R)$ 当且仅当$w \in S'(\mathcal{D}\ R)$。

条目 1 的证明。这直接由定义得出。设 $S_{\preceq}$ 是由过滤器 f 定义的基于过滤器的语义。对于 $+$-可分解性，考虑任意的 R、R' 和 D。若某个由 u 标记的游走 w 属于 $S(\mathrm{D}\ R+R')$，则 $f(w)$ 成立，并且 $u \in L(R)$ 或 $u \in L(R')$ 至少有一个成立。因此，$w \in S(\mathrm{D}\ R)$ 或 $w \in S(\mathrm{D}\ R')$ 至少有一个成立（也可能两者都成立）。

对于 $+$-可组合性，考虑正则路径查询 R 和 R' 以及某个数据库 $\mathcal{D}$。对于任意游走 $w \in S(\mathcal{D}\ R)$，也就是说，满足 $\mathrm{Lbl}(w) \in R$ 且 $f(w)$ 的情况，按定义，w 也属于 $S(\mathcal{D}\ R+R')$，因为 $\mathrm{Lbl}(w) \in L(R+R')$。对于任意 $w' \in S(\mathcal{D}\ R')$，情况也类似。◀

条目 2 的证明。设 $S_{\preceq}$ 是一种基于序的语义。对于 $+$-可分解性，考虑任意 R、R' 和 D。如果某条标记为 u 的游走 w 属于 $S(\mathrm{D}\ R+R')$，那么相对于 $\preceq$，w 是 $\mathrm{Matches}(\mathrm{D}\ R+R')$ 中游走的极小项。由于 $\mathrm{Matches}(\mathrm{D}\ R)$ 和 $\mathrm{Matches}(\mathrm{D}\ R')$ 都是 $\mathrm{Matches}(\mathrm{D}\ R+R')$ 的子集，w 不严格大于这些集合中的任何游走。此外，$u \in L(R)$ 或 $u \in L(R')$。因此，$w \in S(\mathrm{D}\ R)$ 或 $w \in S(\mathrm{D}\ R')$。

为证明 S 不是 $+$-可组合的，考虑一个数据库 $\mathcal{D}$，它由一个顶点 v 和一个标记为某个 $a \in \mathbb{L}$ 的环 $v \text{–} \mathrm{A}_e\ v$ 组成。令 w 表示游走 $v \text{–} \mathrm{A}_e\ v$，因此 w^i 定义良好。由于 $\preceq$ 是 $\mathrm{Walks}(\mathcal{D})$ 上的一个良基偏序，因此存在不同的 i、j，使得 $w^i \prec w^j$。因此，$S(\mathcal{D}\ a^i + a^j) = \{a^i\} \not\supseteq eq\big(S(\mathcal{D}\ a^i) \cup S(\mathcal{D}\ a^j)\big)$。◀

条目 3 的证明。定义过滤器 f 如下。我们使用特征数据库 C_w 和正则表达式 P_w 的概念与记号（定义 31）。每个不自洽的游走 w 都映射到 $\bot$。如果 $S(\mathrm{C}_w, P_w) = \{w\}$，则将每个自洽游走 w 映射到 $\top$，否则映射到 $\bot$。设 D 为一个数据库，R 为一个正则路径查询。下面证明，长度为正且属于 $S(\mathrm{D}\ R)$ 的游走 $w \in \mathrm{Matches}(\mathrm{D}\ R)$ 当且仅当 $f(w) = \top$。

设 w 是 $\mathrm{Matches}(\mathcal{D}\ R)$ 中一条长度为正的游走，并记 $u = \mathrm{Lbl}_{\mathcal{D}}(w)$。语言 $L(R) - \{u\}$ 是正则语言，因此可用正则表达式 R' 表示。由于 S 是基于游走的，我们有如下结论。

$$w \in S(\mathcal{D}, R) \iff w \in S(\mathcal{D}, R' + u) \tag{12}$$

特别注意，$w \notin$ 属于 $\mathrm{Matches}(\mathcal{D}\ R')$，因此 $+$-兼容性蕴含如下结论。

$$w \in S(\mathcal{D}, R' + u) \iff w \in S(\mathcal{D}, u) \tag{13}$$

设 $w_1, \ldots, w_k$ 是 $\mathrm{Matches}(\mathcal{D}\ u)$ 中所有（有限多条）游走；因此，对于某个 i，有 $w_i = w$。所有这些游走的长度都相同，并且由于其中包含 w，它们的长度均为正。现在考虑数据库 $\mathcal{D}'' = \mathcal{C}_{w_1,\ldots,w_k}$ 和正则表达式 $R'' = \sum_{i=0}^{k} P_{w_i}$。特别地，$\mathrm{Matches}(\mathcal{D}'', R'') = \{w_1, \ldots, w_k\}$，因此 $S(\mathcal{D}\ u) = S(\mathcal{D}'', R'')$，这蕴含如下结论。

$$w \in S(\mathcal{D}, u) \iff w \in S(\mathcal{D}'', R'') \tag{14}$$

最后，利用 $+$-兼容性 S 进行简单的归纳法，可得如下结果。

$$w \in S\left(\mathcal{C}_{w_1,\ldots,w_k}, \sum_{i=0}^{i} P_{w_i}\right) \iff w \in S\left(\mathcal{C}_{w_1,\ldots,w_k}, P_w\right) \iff f(w) = \top \tag{15}$$

◀

▶ 注释 42。条目 3 中的等价关系对于长度为 0 的游走并不成立。事实上，考虑如下定义的语义 S。我们用 $\mathrm{Walks}_0(\mathcal{D})$ 表示 $\mathrm{Walks}_0(\mathcal{D})$ 中长度为 0 的游走。然后，我们定义 $S(\mathcal{D}\ R) = \mathrm{Walks}_0(\mathcal{D})$，当且仅当 $\varepsilon \in L(R)$ 且 $\mathcal{D}$ 具有偶数个顶点；否则 $S(\mathcal{D}\ R) = \emptyset$。可以验证，$S$ 是基于游走的且与 $+$兼容，尽管它显然不是基于筛选的。

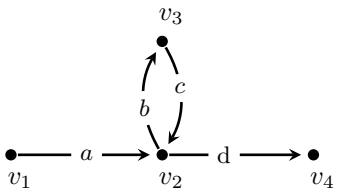
如前所述，基于筛选的语义族与基于序的语义族不相交。这确实可由引理 41 的条目 1 和 2 得出。

▶**推论 43.** 不存在一种语义 S 同时基于序且基于过滤器。

接下来，我们在引理 44 和 45 中指出，在非常宽松的要求下，基于序或基于过滤器的语义都不是 $\cdot$-可组合的，也不是 $*$-可组合的。

▶**引理 44** (与 $\cdot$ 的可组合性)。 **1.** 基于序的语义都不是 $\cdot$-可组合的。**2.** 标识符无关、无界且基于过滤器的语义都不是 $\cdot$-可组合的。

条目 1 的证明。 考虑一种语义 S 基于某个良拟序 $\preceq$ 以及下方所示的数据库 $\mathcal{D}$ （边标识符并不重要）。



考虑正则路径查询 $R = a(bc)^*d$。 R 在 v_1 和 v_4 之间的所有匹配项都标记为$a(bc)^id$ ，其中 i; 并且对于每个 i ，都恰有一个这样的匹配，我们称之为 w_i。根据定义， $S(\mathsf{D}\ R)$ 是 $\{w_i \mid i \geq 0\}$ 中关于 $\preceq$的极小元素构成的有限非空子集。因此，必定存在 $n, k \geq 0$ ，使得 $w_n \prec w_{n+k+1}$。事实上，选择 m ，使得w_m是 $S(\mathsf{D}\ R)$ 中最长的游走，因此 w_{m+1}确实属于 $\mathsf{Matches}(\mathsf{D}\ R)$，但不属于$S(\mathsf{D}\ R)$。因此， $S(\mathsf{D}\ R)$ 必须包含某个 w_n，使得 $n < m$，且 $w_n \prec w_{n+k}$，最后固定 $k = m - n$。

现在我们定义查询 $R_1 = a(bc)^n(b+\varepsilon)$ 和 $R_2 = (c(bc)^k + \varepsilon)d$。 R_1恰有两个匹配：游走 u ，它位于 v_1 和 v_2 之间，标记为 $a(bc)^n$ ；以及游走 u' ，它位于 v_1 和 v_3之间，标记为 $a(bc)^nc$。由于这两条游走的目标点不同，它们都属于 $S(\mathcal{D}\ R_1)$。类似地， R_2 恰有两个匹配：游走 v ，它位于 v_2 和 v_4 之间，标记为 d ；以及游走v' ，它位于 v_3 和 v_4 之间，标记为 $c(bc)^kd$。同样，由于这两条游走的源点不同，它们都属于 $S(\mathcal{D}\ R_2)$。

如果 S是$\cdot$-可组合的，那么 $uv = w_n$和 $u'v' = w_{n+k+1}$都会属于 $S(\mathcal{D}\ R_1 \cdot R_2)$，这与 $w_n \prec w_{n+k+1}$这一事实相矛盾。◀

条目 2 的证明。设 S 是基于过滤函数 f的标识符无关无界语义。设 $\mathsf{D}\ R$满足： $S(\mathsf{D}\ R)$ 包含某个游走 w，且 $\mathsf{Len}(w) > 0$。注意， w 是一个自洽游走，满足 $f(w) = \top$。设 ν 是任意重命名，满足 $\nu(\mathsf{Src}(w)) = \mathsf{Tgt}(w)$、 $\nu(\mathsf{Tgt}(w)) = \mathsf{Src}(w)$，且 $\nu(X) \cap X = \emptyset$，其中：

$$X = (V_{\mathcal{D}} \cup E_{\mathcal{D}}) - \{\mathsf{Src}(w), \mathsf{Tgt}(w)\} \tag{16}$$

游走 $w' = \nu(w)$ 属于 $\nu(\mathsf{D})$，且由于 S 是标识符无关的， $w' \in S(\nu(\mathsf{D})\ R)$。注意， w 和 w' 是相容的，因为它们没有共同的边，并且二者都是自相容的。还要注意， $w \cdot w'$ 和 $w' \cdot w$ 都是良定义的。因此，特征数据库 $\mathsf{C}_{w,w'}$（定义 31）存在，并且包含对每个 $i \in \mathbb{N}$ 都成立的游走 $(w \cdot w')^i$。如果 P_w 和 $P_{w'}$ 分别是 w 和 w' 的特征表达式（定义 31），那么 $S(\mathsf{C}_{w,w'}, P_w) \ni w$ 且 $S(\mathsf{C}_{w,w'}, P_{w'}) \ni w'$。为导出矛盾，假设 S 是 $\cdot$-可组合的。那么，对于某个由 P_w 和 $P_{w'}$ 的连接构成的表达式 R_i ，有 $S(\mathsf{C}_{w,w'}, R_i) \ni ((w \cdot w')^i)$。因此，对于每个 $i \in \mathbb{N}$，都有 $f((w \cdot w')^i) = \top$，这意味着 $S(\mathsf{C}_{w,w'}, (P_w P_{w'})^*)$ 是无限的，矛盾。◀

►**引理 45** (与 $*$ 的可组合性)。 **1.** 没有基于序的语义是 $*$-可组合的 .
2. 不存在标识符无关、无界且基于过滤器的语义 $*$-可组合。

条目 1 的证明。 考虑数据库 $\mathcal{D}$，它有一个顶点 v 和一条边 e，其标签为 a，起点为 v，终点为 v。那么，根据基于序的语义的定义，$S(\mathcal{D}\ a) = \{v \longrightarrow_e {}^{v}\}$ 并且根据 $*$-可组合性，$S(\mathcal{D}\ a^*)$ 是无限的，矛盾。◄

条目 2 的证明。 为导出矛盾，设 S 是一种满足该命题全部四项性质的语义。根据假设，存在一个数据库 $\mathcal{D}$ 和一个正则路径查询 R，使得 $w \in S(\mathcal{D}\ R)$ 且 $\mathsf{Len}(w) \geq 1$。对于 e D中的每条边 let e 为一个新标识符（即，属于 $\mathbb{E} \setminus E$），对于 v D中的每个顶点，令 v 定义如下。

$$\overline{v} = \begin{cases} s & \text{if } v = t \\ t & \text{if } v = s \\ \text{a fresh identifier in } \mathbb{V} \setminus V & \text{otherwise} \end{cases} \tag{17}$$

接下来，我们将数据库 $\mathcal{D}' = (L_{\mathcal{D}'}, V_{\mathcal{D}'}, E_{\mathcal{D}'}, \mathsf{Src}_{\mathcal{D}'}, \mathsf{Tgt}_{\mathcal{D}'}, \mathsf{Lbl}_{\mathcal{D}'})$ 定义如下：

$$L_{\mathcal{D}'} = L_{\mathcal{D}} \qquad \mathsf{Src}_{\mathcal{D}'} : e \mapsto \begin{cases} \mathsf{Src}_{\mathcal{D}}(e) & \text{if } e \in E_{\mathcal{D}} \\ \overline{\mathsf{Src}_{\mathcal{D}}(e)} & \text{if } e \in \overline{E_{\mathcal{D}}} \end{cases}$$

$$V_{\mathcal{D}'} = V_{\mathcal{D}} \cup \overline{V_{\mathcal{D}}} \qquad \mathsf{Tgt}_{\mathcal{D}'} \text{ is defined similarly}$$

$$E_{\mathcal{D}'} = E_{\mathcal{D}} \cup \overline{E_{\mathcal{D}}} \qquad \mathsf{Lbl}_{\mathcal{D}'} : e \mapsto \begin{cases} \mathsf{Lbl}_{\mathcal{D}}(e) & \text{if } e \in E_{\mathcal{D}} \\ \mathsf{Lbl}_{\mathcal{D}}(f) & \text{if } e = \overline{f} \text{ and } f \in E_{\mathcal{D}} \end{cases}$$

我们将其提升到游走。注意，数据库 D′ 包含游走 w，它：起始于 t,一结束于 s，与 w具有相同的标签，并且在重命名后与 w 相等。因此，$w \cdot w$ 是 D′中围绕 s、标签为 uu 的一个环。由于 w 和 w 在重命名后相等，且 S 是标识符无关的，$f(w) = f(w) = \top$。因此，$\{w, w\} \subseteq S(\mathsf{D}', R)$。由于 S是 $*$-可组合的，且 $w \cdot w$是一个环，集合 $\{w, w\}^*$ 是无限的，并且是 $S(\mathsf{D}', R^*)$的子集，这就构成了矛盾。◄

我们猜想，没有任何“合理”的语义相对于 $\cdot$ 和 $*$是可组合的。然而，要定义能推出 $*$-不可组合性或 $\cdot$-不可组合性的“合理”性质的极小集并不容易。

作为本节的总结，下面的命题46概述了关于前几节所定义的部分语义之间兼容性的性质。

►**命题46**。 在不同的可组合性和可分解性性质中，以下性质仅适用于所述语义：

*1. Ac*仅具有 $+$-兼容性和$(\cdot\ *)$-可分解性。*2. Tr*和*SWC*仅具有$(a, +)$-兼容性和$(\cdot\ *)$-可分解性。*3. Sh*、*SM*、*MM*仅具有 a-兼容性和$(\cdot\ +, *)$-可分解性。*4. BT*仅具有$(a, \cdot\ +)$-兼容性和 $*$-可分解性。

回想一下，BT 不是基于游走的（例 19），因此尽管它是·可组合的（条目 4），却不能作为“合理”的基于游走的语义无法·可组合这一猜想的反例。

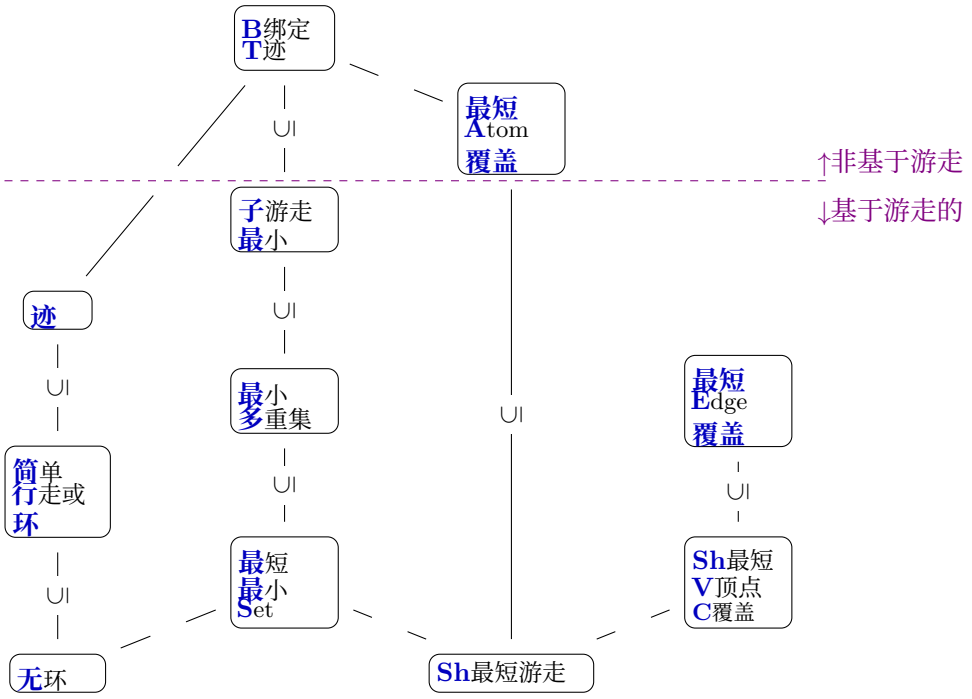


图1 主要语义的包含关系

4.5 语义的包含关系

▶ **定义 47.** 给定两个语义 S_1，S_2，我们称 S_1 包含于 S_2，并记作 $S_1 \subseteq S_2$，若对每个数据库 D 和查询 R，该关系都成立。我们称 S_1 和 S_2 互为 不可比的，如果 $S_1 \subseteq S_2$ 和 $S_1 \supseteq S_2$ 均不成立。

图1展示了本文定义的主要语义之间的包含关系。所有包含关系都是严格的，而不存在路径意味着这些语义不可比较。由于数量众多，我们不提供这些结论的证明。它们都很容易证明，其中一些证明可见于 [DFM23], [Mar24]和 [Khi24]。

▶ 注 48. 很难解释两个语义 S_1, S_2 满足 $S_1 \subseteq S_2$这一事实。差异中的结果究竟大多有意义还是多余，并不总是显而易见。此外，有些情况下差异看起来很小（例如 MM 和 SM），但查询求值的复杂度却有显著差异（见第 5 节）。

4.6 覆盖

回想一下，语义 S 返回一个有限的结果子集（$S(\mathcal{D}\ R)$），而匹配项的集合（Matches($\mathcal{D}\ R$)）可能是无限的。在此过程中，某些信息必然会丢失。于是，一个有趣的问题是如何刻画给定语义 S的覆盖，也就是在多大程度上 $S(\mathcal{D}\ R)$表示Matches($\mathcal{D}\ R$)。本节将介绍几种可能的定义，试图捕捉这一概念。一方面，可以从数据库的顶点（或边）角度来表达覆盖：某个匹配中的所有顶点（或边）也必须出现在某个结果中。另一方面，也可以从子游走角度来表达覆盖：对于 Matches($\mathcal{D}\ R$) 中的每条游走，$S(\mathcal{D}\ R)$都必须至少包含 w的一条子游走。最后，也可以关注表达式的覆盖：如果 R 的某个原子有助于匹配任意游走，那么 $S(\mathcal{D}\ R)$ 中就必须包含一条使用该原子的游走。

4.6.1 顶点覆盖与边覆盖

▶**定义 49**。一条游走 w 覆盖一个顶点 v 当且仅当 v 出现在 w中。一组游走 W 覆盖 v 当 W 中的某条游走覆盖 v时。我们称一种语义 S 为顶点覆盖，如果对于每个$\mathcal{D}$ R，*Matches*Matches($\mathcal{D}$ R)中覆盖的所有顶点也都被 $S(\mathcal{D}$ $R)$覆盖。

可以证明，没有任何基于过滤器的语义是顶点覆盖的，而任何基于序的语义都可以被做成顶点覆盖的，代价是结果集大小的指数级膨胀。

▶**引理 50.** 没有任何基于过滤器的语义是顶点覆盖的。

证明。考虑由过滤函数 f定义的某种语义 S，并设 D 为包含单个顶点 v 和一条标签为 a 、绕 v成环的单条边 e 的数据库。集合 Matches(D a^*) 包含形如 $w_n = v(ev)^n$ 的所有游走，其中 $n \geq 0$。由于 S(D a^*) 必须是有限的，因此必定存在 k，使得 $w_k \notin S$(D a^*)，即使得 $f(w_k) = \bot$。由于该路径是查询 a^k在 D 中的唯一匹配， S(D a^k) 必定为空。 S(D a^k) 并未覆盖v，尽管 Matches(D a^k) 覆盖了它，这表明 S 不是顶点覆盖的。◀

▶**引理 51.** 对于任意基于序的语义 S，若其基于序 $\preceq$，则存在一种顶点覆盖的基于序的语义 S'，它基于某个序 $\preceq'$，且该序是 $\preceq$ 的细化。

证明。设 S 是基于良拟序 $\preceq$的语义。 S 可能无法做到顶点覆盖，因为对于给定的数据库 $\mathcal{D}$、顶点 v 和查询 R，无法保证 v 会出现在 匹配项 $(\mathcal{D}$ $R)$ 中的某个极小游走里。

我们将序 $\preceq'$ 定义如下： $w \preceq' w'$，当且仅当 $w \preceq w'$，并且 w 和 w' 中出现的顶点集合相等。换句话说，我们强制让顶点集合不同的游走彼此不可比较。容易验证，由 $\preceq'$ 定义的基于序的语义 S' 是顶点覆盖的，因为对于 Matches($\mathcal{D}$ R) 中出现的每个顶点，都至少存在一个包含该顶点的极小元素（实际上，对于匹配项中包含该顶点的每一种可能的顶点集合，都至少存在一个这样的极小元素）。◀

根据引理 51 证明中 $\preceq'$ 的构造可知，在某些情况下，结果集的规模会发生指数级膨胀。例如，对于 a标记的 n团，查询 a^*在基于序的语义 Sh 下会对每一对端点返回一个结果，而通过该构造得到的语义则会对每个非空顶点集至少返回一个结果。

现在我们定义一种顶点覆盖语义，它既不是基于筛选的，也不是（严格来说）基于序的，同时还能避免此前结果规模膨胀的问题。

▶**例 52**（最短顶点覆盖语义）。令 $\preceq_{\mathsf{Sh}}$ 为用于定义 Sh 的序。

$$\mathsf{ShVC}(\mathcal{D}, R) = \bigcup_{(s,t,v)\in(V_{\mathcal{D}})^3} \min_{\preceq_{\mathsf{Sh}}}\{w \in \mathsf{Matches}(\mathcal{D}, R) \mid w \text{ starts in } s, \text{ ends in } t \text{ and covers } v\}$$

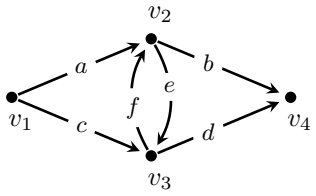
请注意，对于每个顶点，ShVC 都会返回覆盖该顶点的所有 最小长度匹配。 [Khi24] 研究了 ShVC 下查询求值的复杂度，并在图 2（第 25 页）中回顾了相关内容。以下引理总结了它的其他一些性质。

▶**引理 53.** ShVC 是
1. 标识符无关且无界；**2.** 既非基于筛选的，
也非基于序的；**3.** 协单调，但非单调的；

4. 连续的；**5.** a兼容且 θ可分解，但对于任意 θ均不可组合 $\theta \in \{\cdot\ +,\ *\}$。

条目 1 的证明是显然的。条目 4 的证明与第 4.3 节中证明所有基于序的语义都是连续的相似。

条目 2 的证明。 考虑由下图定义的数据库 $\mathcal{D}$：



为简单起见，我们可以将每条边与其标签等同起来。请考虑下面的三个查询和三条游走。

$$R_1 = ab + aefb \qquad R_2 = cd + aefb \qquad R_3 = ab + cd + aefb \tag{18}$$

$$w_1 = v_1 a v_2 b v_4 \qquad w_2 = v_1 c v_3 d v_4 \qquad w_3 = v_1 a v_2 e v_3 f v_2 b v_4 \tag{19}$$

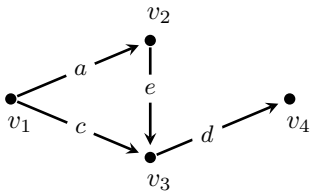
显然，ShVC($\mathcal{D}$ R_1) $= \{w_1, w_3\}$ (∗) 和 ShVC($\mathcal{D}$ R_2) $= \{w_2, w_3\}$ (†)。游走 w_3 必须保留在每个集合中，因为它比 w_1 或 w_2 单独覆盖的顶点都多一个。此外，ShVC($\mathcal{D}$ R_3) $= \{w_1, w_2\}$ (‡) 不包含 w_3，尽管它也是 R_3 的一个匹配，因为它比 w_1 和 w_2 更长，且没有覆盖任何额外的顶点。

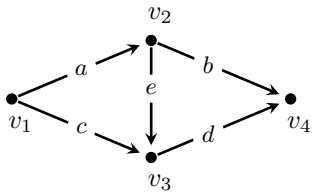
反设 ShVC 是使用某个序 $\preceq$定义的。根据 (∗) 和 (†)，必有 $w_1 \not\preceq w_3$ 和 $w_2 \not\preceq w_3$。然而，根据 (‡)，我们必有 $w_1 \preceq w_3$ 或 $w_2 \preceq w_3$，矛盾。

现在假设 ShVC 是使用过滤函数 f 定义的。根据（∗），我们必须有 $f(w_3) = \top$，但根据（‡），$f(w_3) = \bot$，这同样会导致矛盾。◀

条目 3 的证明。为证明余单调性，假设存在 D $\subseteq$ D′, R，以及 $w \in$ Walks(D)，使得 $w \notin$ ShVC(D R) (∗)，但 $w \in$ ShVC(D′, R) (†)。根据（∗），必定存在$w_1, \ldots, w_k \in$ ShVC(D R)，它们覆盖了 w 的所有顶点，且长度严格小于 w。由于 $w_1, \ldots, w_k$ 是 R 的匹配项，并且属于 Walks(D′)，因此它们属于 Matches(D′, R)。所以，(†) 不可能成立。

为证明非单调性，考虑由下方图定义的数据库 $\mathcal{D}$ 和 $\mathcal{D}'$：





a和查询 $R = ab + cd + aed$。显然，$\mathcal{D} \subseteq \mathcal{D}'$，且 aed 标签的游走位于 v_1 和 v_4之间
b属于 ShVC($\mathcal{D}$ R)，但不属于 ShVC($\mathcal{D}'$, R)。◀

条目 5 的证明。首先，我们证明 ShVC 是 $*$-可分解的。设 D 为一个数据库，R 为一个查询，w 为 ShVC(D R^*) 中的任意游走。根据 ShVC 的定义，存在一个顶点 v，使得 w 是所有覆盖 v 并匹配 R 的游走中最短的。由于 $w \in$ Matches(D R^*)，存在一个因式分解 $w = w_1 \cdots w_k$，使得对于每个 $i \in \{1, \ldots, k\}$，都有 $w_i \in$ Matches(D R)。

为导出矛盾，假设对于某个 i， $w_i \notin \text{ShVC}(\mathcal{D}\ R)$。我们分两种情况讨论。

1. 如果存在一个因子 w_j，其中 $j \neq i$ 经过 v，由于 $w_i \notin \text{ShVC}(\text{D}\ R)$，必定存在一条游走 w_i'，它严格短于 w_i，且在 ShVC(D R) 中具有相同的端点。接着，令 w' 为在 w 的因式分解中用 w_i' 替换 w_i 后得到的游走。游走 w' 严格短于 w'，具有相同的端点，其标签属于 R^*，并覆盖 v。这与 $w \in \text{ShVC}(\text{D}\ R^*)$ 矛盾。2. 否则，由于 $w_i \notin \text{ShVC}(\text{D}\ R)$，必定存在 ShVC(D R) 中的一条游走 w_i'，其端点与 w_i 相同，且覆盖 v。与前面一样，我们通过在 w 中用 w_i' 替换 w_i 得出结论。

证明 ShVC 是 ·-可分解的过程类似但更简单，而证明它是 +-可分解的则显而易见。使用引理 45 的第 1 项中的相同论证可证明其不可 $*$-组合。

利用条目 3 中的数据库 D′，可以证明非-+-可组合性和非-·-可组合性。对于非-+-可组合性，考虑查询 $R_1 = ab + cd$ 和 $R_2 = aed$。v_1 与 v_4 之间标记为 aed 的游走 w 属于 $S(\text{D}', R_2)$，但不属于 $S(\text{D}', R_1 + R_2)$。对于非-+-可组合性，考虑查询 $R_3 = ae + c$ 和 $R_4 = b + d$。v_1 与 v_4 之间标记为 aed 的游走 w 属于 $S(\text{D}', R_1) \cdot S(\text{D}', R_2)$，但不属于 $S(\text{D}', R_1 \cdot R_2)$。◀

注意，在例 52 中将 $\preceq_{\mathsf{Sh}}$ 替换为任意适用的序，都可以得到一个有效的正则路径查询语义。这为定义语义开辟了一种全新的方式，我们将其留待未来研究。

边覆盖。关于边覆盖的情况与顶点覆盖的情况十分相似。它可以用类似的方式定义，并且本文定义的语义中没有一种是边覆盖的。通过调整例 52，可以定义一种最短覆盖语义，即最短边覆盖语义 ShEC。

4.6.2 子游走保证

▶ **定义 54.** 若一种语义 S 满足以下条件，则称其具有子游走保证：对于每个数据库 $\mathcal{D}$、查询 R和游走 w，若满足 $w \in$ 匹配项$(\mathcal{D}\ R)$，则 w 存在一个子游走，属于 $S(\mathcal{D}\ R)$。

这一性质在 [DFM23], 的引理 13 中首次提出，该引理指出 BT 具有这一性质。由此引出了 SM 的定义，按定义，SM 具有这一性质。本文讨论的其他语义均不具有这一性质。

▶**引理55.** 1. SM 和 BT 具有子游走覆盖保证。2. ShVC、ShMS、MM 和 Sh 不具有子游走覆盖保证。3. 没有任何基于过滤器的语义具有子游走保证。

第 1 项的证明。根据 [DFM23，引理 13]，BT 按定义具有子游走覆盖保证。SM 具有该性质，这是定义所规定的。◀

第 2 项的证明。对于 ShMS、MM 和 Sh，其结果源于它们是 SM 的真子集。因此，在这些包含关系不成立的实例中，便构成子游走覆盖保证的反例。

对于 ShVC，考虑下面给出的数据库 $\mathcal{D}$，它由一个 a-环和两个 b-环组成。

$$v_1 \xrightarrow{a} v_2 \xrightarrow{a} v_3 \xrightarrow{a} v_1 \tag{20}$$

$$v_1 \xrightarrow{b} v_2 \xrightarrow{b} v_1 \tag{21}$$

$$v_1 \xrightarrow{b} v_3 \xrightarrow{b} v_1 \tag{22}$$

对于 $R = aaa + bb$，可以验证，式 (20) 中的游走没有任何子游走是 ShVC($\mathcal{D}$ R)。 ◀

条目 3 的证明思路。基于过滤器的语义至少存在一条长度为正的游走 w，但它会将该游走过滤掉。考虑特征数据库$\mathcal{D}$ 以及特征正则表达式 R，它属于 w（定义 31）。那么 $S(\mathcal{D}\ R)$ 为空，而 w 属于匹配项($\mathcal{D}$ R)。◀

4.6.3 原子覆盖

另一种正交的覆盖类型也值得一提。它覆盖的是正则表达式，而非匹配集。直观地说，如果对于每个$\mathcal{D}$和 R，以及其中的每个字母出现位置，只要存在使用该字母出现位置匹配的游走， S ($\mathcal{D}$ R)就至少包含一条这样的游走，我们就称语义 S 具有原子覆盖性质。下面给出正式定义。

▶**定义 56**。设 R 是 *sobre* Σ的正则表达式。我们用 k 表示 R中的原子数量，用 a_i 表示 R中的第 i 个原子，其中 $i \in \{1, \dots, k\}$。
1. R 的线性化是 R 的一个副本 R'，其中 Σ中的每个原子出现都被新字母表 Γ中的不同符号替换。我们用 α 表示相应的双射$\{1, \dots, k\} \to \Gamma$，并用 β表示由下式定义的函数 $\Gamma \to \Sigma$： $\forall i \in \{1, \dots, k\} \beta(\alpha(i)) = a_i$。 *2.* 如果存在某个字 $u' \in L(R')$，其中字母 $\alpha(i)$出现，且对于 R的某个线性化 R'，有 $u = \beta(u')$，则字 u 覆盖 R 的第 i个原子。如果游走 w 在D中的标签覆盖该原子，则称该游走覆盖 R 的第 i个原子。请注意，此概念与所选的线性化无关。*3.* 如果对于每个数据库D、正则表达式 R 以及位置 i（其中 i属于$\{1, \dots, k\}$），只要$Matches(D\ R)$中的某个游走覆盖 R的第 i个原子，就存在一个游走在 $S(D\ R)$中覆盖 R 的第 i个原子，那么语义 S 就是原子覆盖的。

这个定义表达了这样一种思想：设计具有多个备选方案的查询时，每个至少能产生一个匹配的备选方案都应反映在结果集中。可以验证，不存在基于游走的语义能够实现原子覆盖。与顶点覆盖和边覆盖的情况一样，定义实现原子覆盖的语义相当容易（尽管在这种情况下它们并非基于游走），例如下面给出的语义。

▶ **例 57** （最短原子覆盖 ShAC）。 设$\mathcal{D}$ 为一个数据库， R 为一个正则表达式。设$\{1, \dots, k\}$ 为 R 中原子的位置集合。设 $\preceq_{\mathsf{Sh}}$ 为短于 的序（ Sh 所基于的序）。那么：

$$\mathsf{ShAC}(\mathcal{D}, R) = \bigcup_{s,t \in V_{\mathcal{D}}} \bigcup_{i=1}^{k} \min_{\preceq_{\mathsf{Sh}}} \left\{ w \in \mathsf{Matches}(\mathcal{D}, R) \;\middle|\; \begin{array}{l} w \text{ starts in } s \text{ and ends in } t \\ w \text{ covers the } i\text{-th atom of } R \end{array} \right\}$$

不难验证，ShAC 的定义蕴含其实现原子覆盖。由于 BT 包含 ShAC 语义（见第4.5节），它也实现原子覆盖。

▶**命题58**。 ShAC 和BT 实现原子覆盖。

4.7 总结：什么是合理语义?

尽管第4节中描述的许多性质都值得追求，但我们也看到其中有些性质彼此不兼容。下面列出我们认为任何合理语义都应满足的性质。正如我们将看到的，在许多情况下都存在一些注意事项。

标识符独立性 (定义 21) 在数据库系统中，用户并非总能访问标识符，也无法保证标识符始终不随时间变化。因此，让语义依赖标识符似乎很奇怪。不过请注意，依赖标识符的语义仍可能有助于对数据库引擎可以任意选择的情况进行建模，例如 GQL 中的 `ANY SHORTEST` 关键短语或 ShL 语义（例 18）。

标签独立性 (定义 21) 语义是在数据-库模式之前定义的（因此独立于数据库模式）。由于标签是模式中关系名称的抽象，赋予它们含义似乎很奇怪。另一方面，也可以用标签来建模其他方面，例如 `ANY CHEAPEST`GQL 中所述语义里的权重（参见例 20）这一关键短语提案。

无界性 (定义 25) 有界语义只会产生长度受某个常数限制的游走，无论查询或数据库如何，这似乎有悖于使用正则路径查询的初衷。请注意，这里指的是语义本身的一种性质，而不是指查询语言可能的扩展，即允许用户限制返回游走的长度（这在某些场景下可能很有用）。

余单调性 (定义 26) 协单调语义确保：对于一个固定查询，只要通过添加新顶点或边来扩展数据库图，新的结果就应当始终使用其中一些新顶点或边。相反，在非协单调语义下，某个匹配是否属于结果集，取决于一些看似无关的信息。

可分解性 (定义 39) 与余单调性类似，不可分解的语义有时会凭空产生新的结果。具体而言，对于这类语义，无法自底向上地根据表达式计算结果。

与原子式的兼容性 (定义 40) 使用与原子式不兼容的语义，意味着原子查询的返回结果中可能会缺少某些边。如果该语义还具有可分解性，那么这些缺失的边将永远不会出现在任何查询的结果中。请注意，Ac是 GQL 中可用的一种语义，但实际上它与原子式不兼容，因为图中可能存在自环。这可能意味着 Ac假定图中不存在自环。

连续性 (定义 34) 不连续的（基于游走的）语义意味着，这种语义对有限匹配集和无限匹配集的处理方式有所不同。请注意，表达式无关的语义（因此也不是基于游走的语义，见引理 24），例如绑定轨迹语义（例 19），可能是不连续的，但仍可被视为有意义。

复杂度 基于筛选的（定义 7）和基于序的（定义 11）语义的定义，并未对过滤函数或序所属的复杂度类作出要求。合理的语义应基于可处理的过滤函数或序，否则常见查询求值的复杂度类必然无法处理（参见第 5 节）。

5 计算问题

本文档**并非** 关于计算问题的。在第 5 节中，我们汇集了其他来源中关于这些计算问题及其已知复杂性结果的内容。特别是，图 2 总结了本文档提及的最有趣语义的已知结果。

语义	存在性	成员关系	可扩展性	枚举
Tr（迹） [MNP23]	**NP**-c	**NL**	**NP**-c	预处理是**NP-c**
Ac（无环） [BBG20]	**NP**-c	**NL**	**NP**-c	预处理是**NP-c**
Sh（最短行走）	**NL**-c	**P**	**P**	**DelayP**
BT（绑定轨迹） [DFM23]	**NL**-c	**NP**-c	**NP**-c	开放问题/**DelayP**[1]
SM（子游走极小） [Khi24]	**NL**-c	**P**	**NP**-c	Open
MM（极小多重集） [Mar24]	**NL**-c	**coNP**-完全	**coNP**-困难	非 **DelayP** 除非 **P** =**NP**
ShVC（简写：顶点覆盖） [Khi24]	**NL**-c	**P**	**P**	**DelayP**

图 2 主要语义下计算问题（数据）复杂度总结

在 S 下的存在性
数据： 数据库 $\mathcal{D}$ 以及两个顶点 s 和 t，取自 $\mathcal{D}$。 **查询：** 一个正则表达式 R。 **输出：** 是否存在一条游走 w，从 s 到 t，在 $S(\mathcal{D}\ R)$？

▶**定理 59.** 当 S 是基于序的时，在 S 下的存在性属于***NL***。

如图 2 所示，众所周知，在语义 Tr 和 Ac 下，存在性是 **NP**-完全的。很容易证明，在变体 SWC（例 9）或 2Ac（例 10）下，它同样是 **NP**-完全的。实际上，我们猜想，对于任何基于过滤器的语义，存在性都是 **NP**-困难的：

▶**猜想 60.** 在 S 下，对每一种“合理的”基于过滤器的语义，存在性在数据复杂度意义下都是 ***NP***-困难的。

我们在猜想 60 中加入了合理的 这一限制，因为否则这个问题很可能毫无意义。事实上，基于过滤器的语义的定义（定义 7）并未对过滤函数的复杂度作出限制。此外，证明很可能还要求语义满足其他性质，例如第 4.7 节中给出的性质。

S 下的成员关系
数据： 数据库$\mathcal{D}$ 以及一个游走 w 在$\mathcal{D}$中。 **查询：** 一个正则表达式 R。 **输出：** 是否 w 属于 $S(\mathcal{D}\ R)$？

再次，基于过滤器的语义的定义不对过滤函数施加任何复杂度限制。因此，基于过滤器的语义下的成员关系在计算上等价于过滤函数。基于序的语义的情况更糟：$S_{\leq}$ 下的成员关系可能比 $\leq$的复杂度更难。例如，它在 MM [Mar24]，下是**coNP**-困难的，而序本身属于 **NL**。这是最坏情形，因为在合理的基于序的语义下，成员关系始终属于 **coNP**，如下所述。

26 设计与比较 RPQ 语义

▶**命题 61.** 设 $S_{\preceq}$ 是基于序 $\preceq$ 的语义，且该序属于 $\boldsymbol{P}$。若存在多项式函数 p，使得对每一游走 $w' \preceq w$，都有 $|w'| < p(|w|)$，则S 下的成员关系属于$\boldsymbol{coNP}$。

证明概要。不属于 w 意味着要找到一个匹配 w' 满足 $w' < w$。根据假设，检查给定的 w' 是否满足这两个条件可在多项式时间内完成。另一方面，任何 w' 满足 $w' < w$ 的长度都必须关于 w的长度为多项式级，因此 w' 必然足够短，可以作为不属于 w的证书。◀

在 S 下进行枚举
数据： 一个数据库$\mathcal{D}$ 和一条游走 w，位于游走集$(\mathcal{D})$中。 **查询：** 一个正则表达式 R。 **输出：** 所有游走 w 属于 $S(\mathcal{D}\ R)$ 从 s 到 t。

枚举的预处理时间下界至少与存在性问题的下界一样高，因为可以使用前者的算法来解决后者。这正是用于证明在通常的基于过滤器的语义（Tr、Ac、SWC 等）下，枚举问题不可处理的论证。MM 的情形十分特别：枚举问题不可处理，但原因在于延迟，而非预处理时间 [Mar24]。

下面我们将介绍一个不太常见的问题：可扩展性。它使我们能够使用广为人知的手电筒搜索技术：只要可扩展性和成员关系问题都可处理，枚举问题就可处理。

在 S 下的可扩展性
数据： 一个数据库 $\mathcal{D}$，一个游走 w 属于游走$(\mathcal{D})$ 以及一个顶点 t 属于 $\mathcal{D}$。 **查询：** 一个正则表达式 R。 **输出：** 是否存在一个游走 w'使得 $w \cdot w'$属于 $S(\mathcal{D}\ R)$并且终止于 t?

可扩展性至少与存在性一样困难，因为存在性相当于将空游走作为输入提供给可扩展性。尽管形式上并非如此，但语义的定义通常意味着，可扩展性通常至少与成员关系一样困难。

▶ **命题 62** (应用于 [Str21，命题 3。5])。如果在某种语义 S 下，可扩展性和成员关系在数据复杂度方面都属于$\boldsymbol{P}$，那么在 S 下的枚举在$\boldsymbol{DelayP}$ 数据复杂度方面属于（且无记忆，参见 *[Str21]*）。

证明概要。令 A 数据库$\mathcal{D}$、一个游走 w 属于 Walks$(\mathcal{D})$ 以及一个顶点 t 属于 $\mathcal{D}$ 作为该问题的输入。对于数据库中的每个顶点 s，我们调用下面的函数 FORWARD，并以平凡游走 (s) 作为参数。

1. 函数 Forward(游走 w) 2. 如果 Tgt(w) $= t$ 且 成员关系 (D R, w) 返回 True 3. 输出 w4. 对于每条从 Tgt(w) 出发的边 e 5. $w' \leftarrow w \cdot$ (Src(e) e, Tgt(e)) 6. 如果 可扩展性(D R, w', t) 返回 True 7. Forward(w') 8. 返回 $\perp$

请注意，栈上对 Forward 的递归调用次数受下一个要输出的游走的长度限制。◀

▶ 注释 63. 存在性与枚举有一些变体，其中端点不作为输入提供。类似地，有时可扩展性的目标点 t 也不会提供。这些变化通常不会影响这些问题的复杂度。

6 结论与未来工作

业界只考虑过少数几种正则路径查询语义，对它们的系统研究通常也是事后进行的。在这项工作中，我们采取了相反的做法：首先定义正则路径查询语义可能具备的性质，以便未来设计出更好的正则路径查询语义。我们区分了两类简单的语义：基于过滤器的语义，依据逐游走判据筛选结果；以及基于序的语义，仅保留相对于某种排序关系而言极小的匹配。我们提出了一系列可能的性质，其中一些可视为所有正则路径查询语义都应满足的要求，另一些则被证明彼此不兼容。由此，我们得到了几种颇有前景的语义，例如子游走极小语义和最短顶点覆盖语义，它们很可能值得进一步研究。

我们的工作绝非面面俱到，仍存在若干局限，而这些局限也为未来研究提供了有趣的方向。首先，我们的一些定义可能过于严格。例如，基于筛选的 语义类别旨在刻画实践中使用的若干语义，但我们的定义似乎几乎没有为新颖的设计留下空间。另一个例子是与有理算子的兼容性。大多数有理算子只有极少数合理语义能够验证其兼容性，因此值得考虑更弱的兼容性概念。例如， $*$-可组合性的一种较弱形式可以是： $S(\mathcal{D}\ R)$始终包含于 $S(\mathcal{D}\ R^*)$。

其次，我们将研究范围限定于基于游走的语义。语义是一种将可能无限的匹配项集合缩减为有限集合的方法，目的是为用户提供具有代表性的结果。从这个角度来看，基于游走的语义必须仅凭非常有限的信息来进行这种筛选。因此，我们无法考虑那些结果取决于正则表达式实际语法的语义，例如 [ACP12; DFM23]中的语义。未来的研究可以探索利用更多信息的语义，例如游走标签、整个数据库、查询的语法，或者查询如何匹配游走。

第三，我们所考虑的数据模型和查询语言都相当简单。添加数据或节点标签等功能，不应影响我们关于基于游走的语义的观察。然而，在考虑非基于游走的语义时（如上文所述），这些变化会扩大设计空间。注意，GQL 数据模型 [DF+22; GQL24]允许边（以及节点）带有多个标签。这引出了结果中游走重数的问题，而我们完全忽略了这一维度。研究<style id='4'>动态语义也会很有趣：扩展语法，使用户能够对语义进行一定程度的控制，类似于`CHEAPEST`在 GQL 中的做法。

7 参考文献

7.1 学术出版物

[AA+17] R. Angles、M. Arenas、P. Barceló、A. Hogan、J. L. Reutter 和 D. Vrgoč。《图数据库现代查询语言基础》。载于：ACM计算调查 50.5（2017）。doi: 10.1145/3104031.[ACP12] M. Arenas、S. Conca 和 J. Pérez。《超越 Yottabyte 的计数，或 SPARQL 1.1属性路径将如何阻碍该标准的采用》。载于：万维网（WWW）。ACM，2012。doi: 10.1145/2187836.2187922.[AHV95] S. Abiteboul、R. Hull 和 V. Vianu。《数据库基础》。英文。Reading，Mass：Addison-Wesley，1995。isbn：978-0-201-53771-0。[BBG20] G. Bagan、A. Bonifati 和 B. Groz。《图上的正则简单路径查询的三分法》。载于：计算机与系统科学杂志 108（2020）。doi: 10.1016/j.jcss.2019.08.006.[CMW87] I. F. Cruz、A. O. Mendelzon 和 P. T. Wood。《支持递归的图形查询语言》。载于：SIGMOD'87。U. Dayal 和 I. L. Traiger 编。ACM Press，1987。doi: 10.1145/38713.38749.[DF+22] 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、H. Voigt、O. van Rest、D. Vrgoč、M. Wu 和 F. Zemke。《GQL 和 SQL/PGQ 中的图模式匹配》。载于：SIGMOD'22。ACM，2022。doi: 10.1145/3514221.3526057.[DFM23] C. David、N. Francis 和 V. Marsault。《RPQ 的基于运行的语义》。载于：知识表示与推理原理（KR'23）。2023 年 8 月。doi: 10.24963/kr.2023/18.[DFM24] C. David、N. Francis 和 V. Marsault。《RPQ 的不同最短游走枚举》。载于：Proc. ACM Manag. Data 2.2（2024）。doi: 10.1145/3651601.[FG+18a] N. Francis、A. Green、P. Guagliardo、L. Libkin、T. Lindaaker、V. Marsault、S. Plantikow、M. Rydberg、M. Schuster、P. Selmer 和 A. Taylor。《Cypher 语言的形式语义》。arXiv预印本。2018。url: https://arxiv.org/abs/1802.09984.[FG+18b] N. Francis、A. Green、P. Guagliardo、L. Libkin、T. Lindaaker、V. Marsault、S. Plantikow、M. Rydberg、P. Selmer 和 A. Taylor。《Cypher：一种不断演化的属性图查询语言》。载于：SIGMOD'18。ACM，2018。doi: 10.1145/3183713.3190657.[Hig52] G. Higman。《抽象代数中的整除序》。载于：《伦敦数学学会论文集》（1952）。url: https://api.semanticscholar.org/CorpusID:123011310.[Kar15] P. Karandikar。《子词：自动机、嵌入问题与验证》。博士论文。卡尚高等师范学院—卡尚高等师范学院；金奈数学研究所，2015。网址: https://theses.hal.science/tel-01157502/file/Karandikar2015.pdf.[Khi24] S. Khichane。《正则路径查询新语义研究》。硕士论文。巴黎西岱大学，2024。[Mar24] V. Marsault。相对于边包含极小的匹配游走。arXiv预印本。2024。网址: http://arxiv.org/abs/2408.14048.[MM26] V. Marsault 和 A. Meyer。《设计与比较 RPQ 语义》。载于：*ICDT*。即将出版。2026。doi: 10.4230/LIPIcs.ICDT.2026.6。

[MN+23] W. Martens、M. Niewerth、T. Popp、C. Rojas、S. Vansummeren 和 D. Vrgoč。《图数据库模式匹配中的路径表示》。载于：*VLDB Endow.* 会议录16.7（2023）。doi：10.14778/3587136.3587151。[MN+24] W. Martens、M. Niewerth、T. Popp、C. Rojas、S. Vansummeren 和 D. Vrgoč。《图数据库模式匹配的紧凑路径表示》。载于：*ICDEW'24*。2024。doi：10.1109/ICDEW61823.2024.00059。[MNP23] W. Martens、M. Niewerth 和 T. Popp。《正则轨迹查询的三分法》。载于：计算机科学逻辑方法19.4（2023）。doi：10.46298/LMCS-19(4:20)2023。[MT18] W. Martens 和 T. Trautner。《正则路径查询的求值与枚举问题》。载于：*ICDT'18*。由 B. Kimelfeld 和 Y. Amsterdamer 编。第 98 卷。LIPIcs。施洛斯·达格施图尔—莱布尼茨信息学中心，2018。doi：10.4230/LIPIcs.ICDT.2018.19。[Str21] Y. Strozecki。枚举复杂性：增量时间、延迟与空间。特许资格论文。凡尔赛大学—圣康坦昂伊夫利讷大学。2021。

7.2 语言文档与规范

[Cypher24] Neo4j。*Cypher* 手册（版本 *5*）。 https://neo4j.com/docs/cypher-manual/5。2024 年。[GQL24] 国际标准化组织。*GQL*。 https://www.iso.org/standard/76120.html。标准 ISO/IEC CD 39075。2024 年。[GSQL23] TigerGraph。*GSQL* 语言参考（版本 *3.9*）。2023 年。网址： https://docs.tigergraph.com/gsql-ref/3.9/intro/。[PGQ23] 国际标准化组织。*SQL*——第 *16* 部分：*SQL* 属性图查询（*SQL/PGQ*）。 https://www.iso.org/standard/79473.html。标准 ISO/IEC CD 9075-16。 [SQL23]的第 16 部分。2023 年。[PGQL21] Oracle。*PGQL 2.0* 规范。 https://pgql-lang.org/spec/2.0/。2021 年。[SPARQL13] 万维网联盟。*SPARQL 1.1* 查询语言，第 *9* 节：属性路径。 https://www.w3.org/TR/sparql11-query/#propertypaths。2013 年。[SQL23] 国际标准化组织。*SQL*。标准 ISO/IEC CD 9075。第 6 版。2023 年。

8 索引与缩略语

#

2Ac，代表 二重无环语义，6
Ac，代表 无环语义，6
BT，代表 绑定轨迹语义，8
ChW，代表 最廉价语义，8
GU，表示放弃语义，14
LL，表示对数长度语义，10
MM，表示最小多重集语义，12
SM，表示子游走极小语义，7
SR，表示简单运行语义，8
SWC，表示简单游走或环语义，6
ShAC，表示最短原子覆盖语义，23
ShL，表示短字典序语义，7
ShMS，表示最短极小集语义，12
ShT，表示 最短迹语义，5
ShVC，表示 最短顶点覆盖语义，20
Sh，表示 最短语义，5
Tr，表示 迹语义，5

A

无环语义（Ac），6
任意语义，5
原子覆盖，23

B

绑定轨迹语义（BT），8
有界，9

C

特征数据库，11
特征表达式，11
最廉价语义（ChW），8
协单调，9
兼容
  与原子，15
  与有理算子，15
可组合的
  关于有理算子，15
连续的，13

D

可分解
  关于有理算子，15

E

表达式无关，9

F

基于筛选的，6

G

放弃语义（GU），14

I

标识符无关，8

L

标签无关，8
对数长度语义（LL），10

M

匹配，5
最小多重集语义（MM），12
单调的，9

O

基于序的，6

R

可限制的，9

S

最短语义（Sh），5
最短原子覆盖语义（ShAC），23
最短极小集语义（ShMS），12
最短迹语义（ShT），5
最短顶点覆盖语义（ShVC），20
短字典序语义（ShL），7
简单运行语义（SR），8
简单游走或环语义（SWC），6
稳定化的，13
子游走保证，22
子游走最小语义（SM），7
适用，6

T

迹语义（Tr），5
修剪，6
二重无环语义（2Ac），6

U

无界，9

V

顶点覆盖，20

W

基于游走的语义，5