arXiv:2603.01799v1 [cs.AI] 2 Mar 2026

面向描述逻辑 ABox 数据流的增量式、抗不一致性推理

Cas Proost[0009−0007−8419−7363] 和 Pieter Bonte[0000−0002−8931−8343]

鲁汶大学 Kulak 校区计算机科学系，比利时
pieter.bonte@kuleuven.be

摘要。 数据正越来越多地以流式方式产生，因此，人们对如何通过流推理从数据流中实时提取可操作的洞见愈发关注。对数据流进行推理带来了多重挑战，尤其是数据的高速率、推理的实时性要求，以及数据流的噪声多、易变等特性。为应对这些挑战，本文提出了针对描述逻辑 ABox 数据流的增量推理新语义。为解决前两项挑战，我们提出的数据流滑动窗口推理语义，允许基于前一窗口的物化结果，增量计算当前窗口的物化结果。此外，为应对数据流易变的特性，我们基于优选修复语义，提出了适用于此类窗口的不一致性修复新语义。随后，我们详细介绍了所提出的 OWL2 RL 半朴素算法，用于增量式物化维护，既涵盖存在不一致性的情况，也涵盖不存在不一致性的情况。

关键词： 描述逻辑 · 流推理 · 不一致性修复

1 引言

当今的数字环境以数据流持续不断且日益扩大的生成规模为特征。社交媒体、股票市场、在线购物平台等，其生成信息的规模都达到了 20 年前难以想象的程度 [12]。由于这些数据流的体量和速度极大，任何试图从流中提取可操作见解的参与方都会面临一项艰巨任务。正是这一问题促成了流推理这一领域的诞生。它将流处理与知识表示与推理结合起来。流处理长期以来一直致力于从流中提取信息，但所处理的层面相对表面。尽管能够从数据中直接读取的表层见解可能颇有价值，但大多数数据流都蕴藏着大量隐含且可推导的信息。推导这些隐含信息需要经过推理过程，而推理依赖于一组给定的逻辑规则，用以表示领域知识。需要提取的见解越详细或复杂，逻辑规则的表达能力就越强，推理问题也越困难。这种表达能力与效率之间的权衡尤其

在对数据流进行推理时至关重要，因为数据的动态特性带来了实时性要求：必须在底层数据流发生（过多）变化之前推导出见解并据此采取行动。在管理表达能力与效率之间的权衡时，还要考虑对更复杂见解日益增长的需求，这是流推理领域面临的一项重大挑战 [7]。因此，流推理研究往往聚焦于优化推理算法，从而在满足实时性要求的同时实现表达能力更强的推理。

流推理研究中最广泛使用的工具之一是所谓的窗口算子：数据流没有边界，理论上可以永远持续下去。因此，不可能等到整个数据流都生成完毕后再对其进行整体处理。窗口算子的思路是将数据流切分成有限且可处理的片段。窗口机制无需永无止境地等待整个数据流到达，而是让推理器能够逐段处理数据流。

窗口算子有许多种，其中最基本的是基于时间的窗口。最基本形式的基于时间的窗口算子会指定一个开始时间和一个结束时间，并返回数据流中这两个时间点之间的所有事实。在实时应用中，窗口的结束时间通常是（接近）当前时间，而开始时间则设定为过去某个固定时长之前，例如五分钟前。这样的窗口使推理器能够仅考虑最近五分钟的数据，以推导出逻辑结论。根据具体应用，窗口可以按不同的步长移动。例如，上述五分钟窗口的起始时间和结束时间可以每隔一两分钟向前移动一次；此时，下一个窗口与当前窗口重叠，这种窗口称为滑动窗口。每隔五分钟才移动一次的五分钟窗口是一种特殊情况，相邻窗口之间没有重叠，称为滚动窗口。

给定一个窗口，推理器随后可以执行一项称为物化的任务，即推导出窗口中数据所蕴含的所有事实 [21]。随后可以对这种物化结果执行查询，以找出某个查询的所有显式和隐式答案。然而，由于流在持续变化，窗口滑动时，旧的窗口物化结果便不再准确。新的事实进入了窗口，旧的事实离开了窗口，因此需要重新计算物化结果。大多数流推理器会从头开始执行这一过程，但有些推理器会采用增量窗口维护技术 [3, 13, 4]。这些技术利用旧的物化结果来计算新的物化结果：当一个五分钟的窗口向前滑动一分钟时，连续两个窗口之间有四分钟的重叠。与其从头开始计算新的窗口物化结果，不如采用增量方式处理：移除已经离开窗口的数据及由这些数据推导出的任何事实，然后添加新进入窗口的数据及新数据所蕴含的任何事实。这一方法能够实现更高效的推理，尤其是在连续窗口之间存在较大重叠时。

流推理面临的另一项重大挑战是数据流具有易变且嘈杂的特性。现实中的数据流通常由大量传感器生成，

这些传感器偶尔会将错误读数传入数据流，从而导致矛盾，这类矛盾通常称为不一致。标准的符号推理技术无法处理不一致，这意味着任何传入推理引擎的错误信息都可能导致整个系统关闭。此外，不一致也可能源于推理引擎处理数据流的方式：

为简便起见，许多流推理方法（如 C-SPARQL[2], C-QELS[19],$\text{SPARQL}_{\text{stream}}$[11]、IMaRS[13], ）使用窗口来选择所需数据，然后对窗口内的数据进行推理，就如同它是一个静态知识库。这一思想被称为图蕴涵下的推理 [14], ，它是一个强有力的假设，能够带来更高的可扩展性和效率。然而，它也有缺点，如下例所示：假设我们考虑一个由核反应堆传感器读数组成的数据流。反应堆中的不同传感器可能以不同的区间产生数据，例如，假设温度传感器每 30 秒指示一次反应堆温度是过低、正常还是过高。现在考虑一个从该流中选取最近一分钟数据的窗口。该窗口将包含两条温度读数。如果这些读数相同，将窗口内的数据视为静态知识库时不会出现问题。然而，如果温度读数不同，对窗口的静态视图将产生 2 条相互矛盾的事实，例如反应堆温度过低且反应堆温度过高。不具备不一致性修复能力的推理器会指出该窗口是不一致的，任何依赖该推理器结果来调节反应堆温度的自动化系统都将无法采取行动，从而可能造成灾难性后果。

本例表明，数据流中的不一致可能源于多种不同来源，而不仅仅来自错误的传感器读数。处理不一致是流推理领域中一项重要且迄今尚未解决的挑战，正如 [7] 和 [14]中所指出的那样。

本文旨在应对上述流推理领域的两大主要挑战。为此，我们关注一类具有较强表达能力的逻辑，即描述逻辑。描述逻辑已得到广泛研究，但主要是在静态数据的背景下，其中并不存在实时性要求。由于其表达能力，它们非常适合需要从数据中提取复杂洞见的应用，而这正是流推理的一项基本目标。例如，它们构成了网络本体语言（OWL2）[16]的基础，该语言在语义网和流推理社区中被广泛使用。

自然地，由于表达能力与效率的权衡，在（尤其是表达能力最强的）描述逻辑下进行推理十分困难 [17, 9]。因此，我们试图修订描述逻辑语义，以支持滑动窗口上的增量推理，从而在保留描述逻辑表达能力的同时满足实时性能要求，进而应对第一项挑战。
此外，修复不一致性已在描述逻辑文献中得到广泛研究

[20, 22]。通过借鉴优选修复语义的思想 [6] 并将其调整为我们提出的新型语义，我们应对了第二项挑战：数据流易变且嘈杂的特性。

本文结构如下：首先概述增量推理和不一致性修复领域相关研究中的问题。随后介绍描述逻辑和数据流的基础知识。接着详细阐述我们针对窗口推理提出的增量语义，并介绍不一致性修复语义。最后一节概述在这些语义下、针对描述逻辑 RL（对应 OWL 2 RL 配置文件）进行推理的半朴素算法。

2 相关工作

在描述逻辑文献中，增量推理或物化维护鲜有涉及，因为数据库和知识表示领域对其研究更为广泛 [23, 15, 10],，而这些领域中的推理形式有所不同。不过，Datalog 形式体系下的推理研究对此已有一定关注；该形式体系与描述逻辑相似（甚至与某些描述逻辑等价）。在 [21],中，作者全面概述并比较了维护 Datalog 知识库物化的不同算法。他们考察了三种不同方法：计数算法、删除/重推导算法（DRed）以及前向-后向-前向算法（FBF）。

Counting 算法会跟踪知识库中的某条事实被推导/证明的次数；删除事实时，则会递减所有由被删除事实推导出的事实的计数器。当某条事实的计数器归零时，它也会从知识库中消失。在这三种算法中，这种算法的表现似乎最好，但它需要额外开销在内存中维护这些计数器，而遗憾的是，论文并未对此进行基准测试。

DRed 算法则首先删除所有由待删除事实推导出的事实，这一步称为过度删除，然后重新推导所有被过度删除的事实。FBF 算法是一种参数化算法，其行为与 DRed 类似，会先执行删除步骤，但它通过对加入删除池的事实进行反向链，尽可能减少过度删除。反向链会查找删除后仍然成立的证明；如果找到这样的证明，就会将该事实从删除池中移除。一个参数决定反向链的深度，也就是说，反向链接深度为零时，FBF 的极端情况就是 DRed。

请注意，这些算法需要通过推理来删除事实，因为它们必须确定物化中的哪些事实依赖于待删除的事实，以便一并删除这些事实或减少其计数。在延迟至关重要的流处理场景中，这并不理想。因此，IMaRS[13] 和LASER[4] 开发了用于数据流增量物化维护的算法，这些算法在删除事实时无需进行推理。

为此，它们利用了数据流的时间特性。在流处理场景中，物化维护通常基于窗口进行，因此事实的时间戳会指明窗口滑动时哪些 ABox断言将离开窗口。通过考虑用于推导某个事实的 ABox断言的时间戳，就可以确定该事实离开 ABox 的“临界时间”。推导出该事实时便会确定这一时间；每当发现该事实的另一种证明时，都会更新这一时间。因此，窗口滑动、需要从物化中删除事实时，无需通过推理来判断哪些派生事实需要删除，只需查找临界时间已过期的事实并将其删除即可。

不过，这两篇论文是唯一针对数据流实现此类算法的工作，而且涵盖的形式体系并不广泛：IMaRS 适用于 RDFS$^+$ ，LASER 适用于 LARS。描述逻辑领域尚未出现此类算法，甚至没有专门针对数据流设计的语义。尽管如此，基于描述逻辑的 OWL2 本体语言在语义网和物联网中被广泛使用，而这两者正是流推理的两个主要应用领域。此外，无论是 IMaRS、LASER，还是增量视图维护算法（Counting、DRed、FBF），都无法在出现不一致时进行推理，尽管易变数据流中经常会出现不一致。

相比之下，描述逻辑领域对不一致修复问题给予了广泛关注。不同的语义旨在通过“修复”不一致，使人们能够在不一致知识库上执行不同的推理任务。大多数研究只考虑不一致的ABox，因为不一致的 TBox 通常被视为建模错误 [6, 20]。对于流推理而言，不一致的 TBox 可以且应当在系统部署前修正，而 ABox（数据流）中的不一致则必须在运行过程中修复。

当前的描述逻辑不一致性修复语义源自 ABox 修复（AR） [20] 语义的思想。该语义考察 ABox 的极大一致子集，即修复。若某个事实可由该 ABox 的所有修复推出，则认为该事实可由不一致的 ABox 推出。这一思想的变体包括交集 AR（IAR）语义和勇敢语义：它们考察相同的极大一致子集，但分别将所有修复的交集中成立的事实，以及任一修复中成立的事实，定义为可推出的事实。这些语义的问题在于，它们需要考察所有这些修复，因此必须计算出所有修复才能进行物化 [5]。此外，目前计算这些修复的方式是“猜测”ABox 的子集，并检查它们是否为修复。由于含有 n 条断言的 ABox 有 2^n 个可能的子集，这一过程在某些情况下会极其低效。

AR 语义的一种有趣变体是优选修复语义[6], ，它是 AR 或 IAR 语义的推广。在这种语义下，修复同样是 ABox 的极大一致子集，但极大性是相对于子集上的某种一般预序来定义的，而不是相对于 $\subseteq$预序。

最著名的优选修复语义是优先级语义和加权修复语义：前者假设 ABox 中存在不同的优先级层级，后者则为 ABox 中的所有断言赋予权重。在 [6], 中，作者研究了这些优选修复语义下查询回答的复杂度。他们发现，这些语义会产生更多答案，但性能有所下降。不过我们注意到，他们的实现似乎与 AR 或 IAR 算法的运行方式相似，只是增加了一些额外步骤，这也解释了性能下降的原因。

尽管 AR、IAR 和勇敢语义已在流式描述逻辑中得到研究，但相关研究仅涉及 DL-Lite [8],$\mathcal{R}$ $\mathcal{EL}\bot$。此外，大多数关于不一致性修复的工作关注查询回答，而非物化。据我们所知，优选修复语义尚未在数据流上、物化场景中或 OWL 2 RL 的背景下得到研究。

3 预备知识

3.1 描述逻辑与 OWL 2 RL

描述逻辑是一类知识表示语言，因其可判定性和表达能力等特性，常用于推理应用[1]。在标准描述逻辑系统中，信息分为两种不同的结构：ABox 和 TBox。ABox 包含事实信息，即 ABox断言，通常分为两类：概念断言或角色断言。概念断言的形式为 $C(x)$，意为“个体 x 属于概念C”。描述逻辑中的概念对应于其他形式知识表示语言（例如 Datalog）中的一元谓词。相较之下，角色断言的形式为 $R(x,\ y)$，意为“个体 x 与个体 y之间存在 R关系”，因此角色相当于描述逻辑中的二元谓词。

虽然 ABox 包含事实性的显式信息，但 TBox 包含用于对这些事实进行推理的规则。一条描述逻辑规则由头部和体部组成；根据特定描述逻辑的语法，它们是由概念、角色以及诸如 $\sqcap, \sqcup, \exists, \forall, \ldots$ 等逻辑运算符构成的语句。TBox 规则允许对 ABox 中的数据进行推理：如果某条规则的体部/被包含项对某个个体（或一对个体）成立，则其头部/包含项也必须成立。

描述逻辑语义基于解释这一思想，解释通过将知识库映射到某个具体集合来表示该知识库：一个解释 $\mathcal{I}$ 由一个论域 $\Delta_{\mathcal{I}}$ 和一个映射 $\cdot^{\mathcal{I}}$组成。该映射将知识库中的每个个体映射到论域中的一个元素，将每个概念映射到论域的一个子集，并将每个角色映射到 $\Delta_{\mathcal{I}} \times \Delta_{\mathcal{I}}$的一个子集。在本文中，我们将坚持唯一名称假设（UNA），这意味着解释的论域恰好是知识库中个体的集合，并且解释将每个个体映射到其自身。不过请注意，在一般情况下研究描述逻辑时，这一点未必成立。

如果解释 $\mathcal{I}$ 满足 ABox 概念断言 $C(x)$，则 $x^{\mathcal{I}} \in C^{\mathcal{I}}$,

如果满足角色断言 $R(x,\ y)$，则 $(x^{\mathcal{I}},\ y^{\mathcal{I}}) \in R^{\mathcal{I}}$。自然，如果某个解释满足 ABox $\mathcal{A}$中的所有断言，就说它满足 $\mathcal{A}$。类似地，如果解释满足规则 $B \sqsubset H$，则称其满足该规则；如果它满足 TBox $\mathcal{T}$中的每条规则，则称其满足该 TBox。请注意，$B^{\mathcal{I}}$ 和 $H^{\mathcal{I}}$ 在此处的含义有些模糊，但在讨论本文所关注的 RL 描述逻辑时会予以明确。
知识库的一个解释 $\mathcal{I}$（$\mathcal{A},\ \mathcal{T}$）如果同时满足 $\mathcal{A}$ 和 $\mathcal{T}$，就是该知识库的一个模型。在描述逻辑中，当一个断言或查询在知识库的每个模型中都成立时，当且仅当它被该知识库蕴含。这意味着，描述逻辑中的推理问题归结为计算哪些事实在每个模型中都成立。通常这并不容易，但对于某些描述逻辑而言，极小模型的存在（即一个包含于所有其他模型中的模型）使这个问题容易处理得多。

其中一种描述逻辑是 OWL 2 RL 配置文件所基于的描述逻辑；我们考虑的是其中一种简化版本，定义见 [18]。构造概念或角色时，它使用由概念名 C_1、.....、C_n、角色名 P_1、.....、P_n 以及逻辑运算符 $\sqcap$、$\exists$ 和逆角色运算符 $^-$组成的字母表。使用这种语法，规则具有以下形式之一：

$$B \sqsubseteq A, B \sqsubseteq \bot, R_1 \sqsubseteq R_2$$

其中，A 是概念名，R_1、R_2 是角色名或其逆，$\bot$ 是底概念，即每个模型都将其映射到空集的概念；B 是由上述运算符构造的概念。这种语法最值得注意的性质是，规则头部不允许出现存在量词，因此任何 RL 知识库都具有一个极小模型或典范模型，这是一项实用性质。

该模型的构造从一种解释开始，该解释仅表示 ABox 中存在的事实，然后通过迭代添加事实来满足 TBox 中的规则。

定义 1. 对于 *RL* 知识库 $(\mathcal{A},\mathcal{T})$，标准解释 $\mathcal{I}_{\mathcal{A}}$定义如下：

1. $A^{\mathcal{I}_{\mathcal{A}}} = \left\{a^{\mathcal{I}_{\mathcal{A}}} | A(a) \in \mathcal{A}\right\}$
2. $P^{\mathcal{I}_{\mathcal{A}}} = \left\{(a^{\mathcal{I}_{\mathcal{A}}}, b^{\mathcal{I}_{\mathcal{A}}}) | P(a,b) \in \mathcal{A}\right\}$

随后，利用标准解释，通过一种前向链推理过程构造一系列解释：设置 $\mathcal{I}_0 = \mathcal{I}_{\mathcal{A}}$。要从 $\mathcal{I}_j$得到 $\mathcal{I}_{j+1}$，应用以下规则：

1. 对任意 $B \sqsubseteq A \in \mathcal{T}$，若 $a \in B^{\mathcal{I}_j}$ 和 $a \notin A^{\mathcal{I}_j}$，则将 a 添加到 $A^{\mathcal{I}_{j+1}}$。2. 对任意 $P_1 \sqsubseteq P_2 \in \mathcal{T}$，若 $(a$、$b) \in P_1^{\mathcal{I}_j}$ 且 $(a$、$b) \notin P_2^{\mathcal{I}_j}$，则将 $(a$、$b)$ 添加到 $P_2^{\mathcal{I}_j}$。
3. 对任意 $B \sqsubseteq \bot \in \mathcal{T}$，若 $a \in B^{\mathcal{I}_j}$，则终止该过程，因为 $(\mathcal{A}$、$\mathcal{T})$ 不一致。

注意，$B^{\mathcal{I}}$ 这里的显然选择取决于 B 的形式：如果$B := A$ 是概念名，则 $B^{\mathcal{I}} = A^{\mathcal{I}}$；如果 $B := B_1 \sqcap B_2$，则 $B^{\mathcal{I}} = B_1^{\mathcal{I}} \cap B_2^{\mathcal{I}}$ 并且

如果 $B := \exists R.B_1$，则 $B^{\mathcal{I}} = \{a | \exists b \in B_1^{\mathcal{I}} s.t. (a, b) \in R^{\mathcal{I}}\}$

如 [18], 所示，该过程会终止，并且要么证明知识库不一致，要么为知识库构造一个模型，称为典范模型；该模型具有所需的极小性。

现在我们回顾一些关于不一致性修复的语义概念。特别是，我们讨论优先修复。与大多数 ABox 修复语义一样，优选修复语义将修复定义为 ABox 的一个一致子集，且该子集是极大的。决定这种极大性的关系/次序，正是优选修复语义区别于其他语义之处。

定义2（[6]）。给定一个 *ABox* $\mathcal{A}$ 以及一个划分 $\{\mathcal{P}_1, ..., \mathcal{P}_n\}$（称为偏好层级）$\mathcal{A}$，我们在 $\mathcal{A}$的子集上定义一个偏序 $\subseteq_{\mathcal{P}}$，$2^{\mathcal{A}}$如下：

$$\forall \mathcal{A}', \mathcal{A}'' \subseteq \mathcal{A} : \mathcal{A}' \subseteq_{\mathcal{P}} \mathcal{A}'' \textit{ iff:}$$

1. 对所有 $1 \leq i \leq n$， $\mathcal{A}' \cap \mathcal{P}_i = \mathcal{A}'' \cap \mathcal{P}_i$，或

2. $\exists i$、$1 \leq i \leq n$，使得 $\mathcal{A}' \cap \mathcal{P}_i \subsetneq \mathcal{A}'' \cap \mathcal{P}_i$，并且对于所有$i \leq j \leq n$都有 $\mathcal{A}' \cap \mathcal{P}_j = \mathcal{A}'' \cap \mathcal{P}_j$

那么，具有偏好层级 $\mathcal{P}_1$、.....、$\mathcal{P}_n$ 的 ABox $\mathcal{A}$ 的一个优先修复，是满足以下条件的一致子集 $\mathcal{A}' \subset \mathcal{A}$：不存在其他一致子集 $\mathcal{A}$" $\subset \mathcal{A}$，使得 $\mathcal{A}' \subseteq_{\mathcal{P}} \mathcal{A}$"。此外，如果一个事实被每个优先修复蕴含，就称其在优先 AR 语义下被蕴含；如果它被所有优先修复的交集蕴含，就称其在优先 IAR 下被蕴含。下面通过一个例子来说明：

例 1。令 $\mathcal{A} = \mathcal{P}_1 \cup \mathcal{P}_2$ 满足 $\mathcal{P}_1 = \{A(a)\ B(a)\}$，且 $\mathcal{P}_2 = \{C(a)\}$。设 $\mathcal{T} = \{A \sqcap C \sqsubset \bot, B \sqcap C \sqsubset \bot\}$，表示概念 C 与 A 和 B 均不相交。那么，$\mathcal{A}$ 的一致子集恰好为 $\mathcal{P}_2$、$\mathcal{P}_1$ 以及 $\mathcal{P}_1$ 的两个单元素子集。显然，$\mathcal{P}_1 \subseteq_{\mathcal{P}} \mathcal{P}_2$，因为 $\mathcal{P}_1 \cap \mathcal{P}_2 \subsetneq \mathcal{P}_2 \cap \mathcal{P}_2$。因此，$(\mathcal{T}, \mathcal{A})$ 的唯一优先修复是 $\mathcal{P}_2$，这意味着该知识库在优先 AR 和优先 IAR 下蕴含的唯一事实是 $C(a)$。

正如本例所示，ABox $\mathcal{A} = \mathcal{P}_1, ..., \mathcal{P}_n$的优先修复通过尽可能少地移除高优先级断言来解决不一致：如果某个不一致性要求要么移除多个低优先级断言，要么移除单个高优先级断言，则优选修复语义会移除低优先级断言，以保留高优先级断言。

3.2 描述逻辑中的流

一般而言，在流推理中，流是带时间标注的事实的集合。这些事实的形式可能因所采用的形式体系而异，例如，RDF流由四元组（s, p, o, t）组成，它在传统的（主语、谓词、宾语）三元组中加入了时间戳。不过，在描述逻辑中，事实

由ABox断言表示，因此，描述逻辑中的流由带时间标注的ABox断言组成，具体定义如下。流通常被视为无限的，因此需要一种方法来限定流的片段。流上的基于时间的窗口算子由某个区间$[t_1, t_0]$ 定义，并从流中选取所有时间戳落在该区间内的事实。对流进行推理时，通常采用图蕴含机制，即当且仅当根据窗口构建的知识库蕴含某个查询或事实时，才认为该窗口蕴含该查询或事实。一般来说，图蕴含是流推理研究中常见的三种蕴含机制之一。其他机制是窗口级蕴含和流级蕴含。在窗口级蕴含中，仍然使用窗口从流中选取相关部分，因此，窗口范围之外的断言不会用于蕴含判断。不过，窗口级蕴含与图级蕴含的区别在于，窗口内的时间戳不会被忽略；也就是说，窗口内的断言仅在其出现的时间戳上被视为真，而不是在整个窗口期间都为真。这种蕴含机制常用于对流进行更具表达力的时序推理，同时通过窗口限制推理范围，以实现更实用的推理。因此，窗口级蕴含介于高效但表达力较弱的图级蕴含，与表达力最强但不切实际的流级蕴含之间：在流级蕴含中，流中的所有信息都会纳入推理过程，包括流中任意部分的断言及其时间戳。虽然这能实现表达力最强的推理，并允许推导时间距离不受限制的时序关系，但由于内存和计算方面的需求，这种方法在现实应用中往往不切实际 [14]。

在本文中，我们将流建模为一组 $\{\mathcal{A}_t\}_{t\in\mathbb{T}}$ 描述逻辑ABox，其中集合 $\mathbb{T}$ 称为该流的时间线。需要强调的是，流不包含任何 TBox 数据；对这类流进行推理时，所依据的是一个固定的 TBox。流中的所有 ABox 都使用同一组个体名 $a_1, \ldots$、概念名 $C_1, \ldots$ 和角色名 $P_1, \cdots$。它们共享这些名称，意味着某个名称若出现在一个 ABox 中，之后又出现在另一个 ABox 中，那么这些名称就表示同一个个体/概念/角色；但并非每个 ABox 都需要提及每个名称，也并非每个名称都需要在多个 ABox 中出现。请注意，我们的模型等价于一组时间标注ABox断言，因为每个 ABox$\mathcal{A}_t$ 只是用于容纳流中所有时间戳为 t的断言。

本文对每个描述逻辑流作出如下假设：

1. 时间线 $\mathbb{T}$ 是 $\mathbb{R}$ 的可数无穷子集，且存在一个 δ_0 ，使得 $\forall t_1,\ t_2 \in \mathbb{T}$: $|t_1 - t_2| \geq \delta$。2. 流中的每个 ABox $\mathcal{A}_t$ 都包含有限数量的断言

由区间 $I = [t_1, t_2]$ 定义的窗口是集合 $W_I = \{\mathcal{A}_t\}_{t \in I \cap \mathbb{T}^\circ}$ 与窗口 W_I 关联的是其窗口 ABox $\mathcal{A}_{W_I} = \bigcup_{\mathcal{A}_t \in W_I} \mathcal{A}_t$。根据我们的假设……，显然每个窗口和每个窗口 ABox 都是有限的。此外，显然在图蕴涵机制下，查询/事实被窗口蕴涵，当且仅当它在标准描述逻辑语义下被窗口 ABox 蕴涵。

4 增量语义

本节将为描述逻辑流上的基于窗口的推理定义一种图蕴涵语义的增量版本。虽然技术上可以使用标准描述逻辑语义进行图蕴涵推理，但它们忽略了两个重叠窗口之间的联系，因此不足以满足增量推理的需求。

请注意，这些语义旨在适用于一般的描述逻辑。不过，在讨论推理算法时，我们将范围限定为 OWL 2 RL 描述逻辑。

给定如上定义的数据流 $\{\mathcal{A}_t\}_{t \in \mathbb{T}}$ 和描述逻辑 TBox $\mathcal{T}$，我们称区间 $I = [t_1, t_2] \cap \mathbb{T}$ 定义了一个窗口 $W_I = \{\mathcal{A}_t\}_{t \in I^\circ}$。对于 W_I，窗口解释流是一个解释集合 $\{\mathcal{I}_t\}_{t \in I}$，其中每个 $\mathcal{I}_t$ 都是 $\mathcal{A}_t$的一个解释（而非模型）。窗口解释流会在窗口 ABox $\mathcal{A}_{W_I} = \bigcup_{\mathcal{A}_t \in W_I} \mathcal{A}_t$上诱导出一个解释；定义一些内容后，我们将对此进行说明

关于符号：

定义 3。给定两个解释 $\mathcal{I}_1$、$\mathcal{I}_2$，其论域分别为 Δ_1、Δ_2，对应两个 *ABox* $\mathcal{A}_1$、$\mathcal{A}_2$，我们将这两个解释的直和定义为对 $\mathcal{A}_1 \cup \mathcal{A}_2$ 的解释 $\mathcal{I}_1 \oplus \mathcal{I}_2$，其论域为 $\Delta_1 \cup \Delta_2$，解释映射如下：

$$C^{\mathcal{I}_1 \oplus \mathcal{I}_2} := C^{\mathcal{I}_1} \cup C^{\mathcal{I}_2}$$

$$R^{\mathcal{I}_1 \oplus \mathcal{I}_2} := R^{\mathcal{I}_1} \cup R^{\mathcal{I}_2}$$

由于我们采用唯一名称假设，该定义是良定义的，因此同一个体的名称在不同解释中相同。此外，该定义可用于归纳定义某一系列 $\bigoplus_{1 \le i \le n} \mathcal{I}_i$ 的解释 $\mathcal{I}_1$，…，$\mathcal{I}_n$。这使我们能够如下定义由窗口解释流诱导的窗口 ABox 上的解释：

定义 4（窗口解释）。对于窗口 W_I 和窗口解释流 $\{\mathcal{I}_t\}_{t \in I}$，解释 $\mathcal{I}_{W_I} = \bigoplus \mathcal{I}_t$ 是由该窗口解释流导出的窗口$_{t \in I}$解释。

因此，该窗口解释是窗口 ABox $\mathcal{A}_W$的一个解释，其中恰好包含窗口解释流中的信息。值得注意的是，即使窗口解释流由窗口中每个 ABox 的模型组成，相关联的窗口解释也未必是 $\mathcal{A}_W$ 的模型，这一点可通过以下简单示例说明：

示例 2. 考虑一个 TBox $\mathcal{T} = \{A \sqcap B \sqsubseteq C\}$ 以及一个包含两个 ABox 的窗口：$\mathcal{A}_1 = \{A(a)\}$，$\mathcal{A}_2 = \{B(a)\}$。假设我们考虑一个窗口解释流，其中包含解释 $\mathcal{I}_1$，其论域为 $\{a\}$，映射由 $A^{\mathcal{I}_1} = \{a\}$给出，$B^{\mathcal{I}_1} = C^{\mathcal{I}_1} = \emptyset$；以及解释 $\mathcal{I}_2$，其论域为 $\{a\}$，映射由$B^{\mathcal{I}_2} = \{a\}$给出，$A^{\mathcal{I}_2} = C^{\mathcal{I}_2} = \emptyset$。显然，这些解释分别是其对应 ABox 的模型。然而，显然，所诱导的窗口解释$\mathcal{I}_W = \mathcal{I}_1 \oplus \mathcal{I}_2$ 不是 $\mathcal{A}_W = \mathcal{A}_1 \cup \mathcal{A}_2$ 的模型，因为它将 C 映射到 $\emptyset$。

现在，我们可以通过引入窗口模型这一概念，准确界定在我们的语义下如何针对窗口进行图蕴含推理。

定义 5（窗口模型）。如果窗口解释流 $\{\mathcal{I}_t\}_{t\in I}$ 中的每个 $\mathcal{I}_t$ 都是 $\mathcal{A}_t$ 的模型，并且与之关联的窗口解释 $\mathcal{I}_{W_I}$ 是$\mathcal{A}_{W_I}$的模型，那么 $\{\mathcal{I}_t\}_{t\in I}$ 就是 W_I 的窗口模型流。

如果某个事实在窗口 W 的所有窗口模型中都成立，即在与窗口模型流相关联的所有窗口解释中都成立，则称该事实在窗口中具有图蕴涵，即所有窗口解释都与窗口模型流相关联。下面的命题表明，这等价于该事实被 $\mathcal{A}_W$所蕴涵。

命题 1。将 W 的一个窗口模型流映射到其相关联的窗口解释 $\mathcal{I}_W$ 的映射，是从 W 的窗口模型流到 $\mathcal{A}_W$的模型的一个满射。

证明。显然，每个窗口模型流都会映射到 $\mathcal{A}_W$的一个模型，因为窗口模型流正是如此定义的。现在假设 $\mathcal{I}_W$ 是 $\mathcal{A}_W$的一个模型。考虑窗口解释流 $\{\mathcal{I}_t\}_{t\in I}$，其中每个 $\mathcal{I}_t = \mathcal{I}_W$。显然，其窗口解释是 $\mathcal{A}_W$的一个模型。此外，由于 $\mathcal{I}_W$ 满足 $\mathcal{A}_W$中的每个断言，因此对于所有 $t \in I$，它也满足 $\mathcal{A}_t$ 中的每个断言，这意味着每个 $\mathcal{I}_t$ 都必须是 $\mathcal{A}_t$的一个模型。因此，该窗口解释流必定是一个窗口模型流。这就证明了满射性。■ ■

这些语义看起来或许过于复杂，因为有人可能会认为，只考虑 $\mathcal{A}_{W_I}$ 的模型就足以对窗口进行图蕴涵推理。然而，考察增量推理问题（即窗口滑动时会发生什么）便能略窥这些语义采用这种形式的原因。

假设只考虑$\mathcal{A}_{W_I}$ 的模型，并通过计算该 ABox 的最小模型/典范模型来完成物化。当窗口

发生变化时，无法根据旧的物化结果计算新的物化结果：时间结构没有任何保留，也无法确定需要从窗口中删除哪些事实。

相较之下，如果物化采用上述窗口模型的形式，那么只需删除对应于已移出窗口的时间戳的模型，就能轻松移除移出窗口的事实。这一点将在增量推理一节中进一步讨论。

文献中很少讨论优选修复语义，可能是因为在静态 ABox 这一标准情形下，偏好层级所增加的结构并不会自然出现。然而，对于 ABox 流，我们有一种自然的方式来设定偏好层级：ABox 的时间戳。考虑在对流中的窗口进行图蕴含推理时，如何修复由此产生的不一致性，显然，窗口划分为各个 ABox 的方式可以作为偏好层级的划分。较新的 ABox 优先于较旧的 ABox，这意味着我们可以将 ABox 的偏好层级设为与其时间戳相同。于是，为了解决不一致性，窗口修复会优先删除任意数量的较旧事实，而不是删除一条新事实。其背后的理念可以简洁地概括为：

$$New\ is\ always\ better.$$

具体而言，当出现不一致性时，为了保留较新的事实，将删除导致该不一致性的最早事实。这是因为较新的事实被认为更可靠，并且可以被解释为否定较早的事实。

示例 3. 假设我们有一个表示汽车踏板运行状态的流，其概念为 $ClutchPressed$, $GasPedalPressed$, $BreaksPressed$。再假设我们有以下 ABox 流（ABox 的下标表示其时间戳，单位为秒），它可能表示一辆汽车 x在交通信号灯前停车：

$$\mathcal{A}_0 = \{GasPedalPressed(x)\}$$

$$\mathcal{A}_1 = \{GasPedalPressed(x)\}$$

$$\mathcal{A}_2 = \{\}$$

$$\mathcal{A}_3 = \{BreaksPressed(x)\}$$

$$\mathcal{A}_4 = \{BreaksPressed(x), ClutchPressed(x)\}$$

如果我们采用一个大小为 2 秒、每秒滑动一次的窗口，第一个窗口将包含 $\mathcal{A}_0$, $\mathcal{A}_1$ 和 $\mathcal{A}_2$。这个窗口只包含断言$GasPedalPressed$(x)，尽管它出现在两个不同的时间点。不过，当窗口滑动时， $\mathcal{A}_0$ 会从窗口中移除，而 $\mathcal{A}_3$ 会加入。现在，窗口中包含两个不同的断言：

$$GasPedalPressed(x), BreaksPressed(x)$$

当然，任何开过车的人都知道，这种情况不应该发生；我们可以用包含规则 $GasPedalPressed \sqcap BreaksPressed \subseteq \bot$ 的 TBox 来建模。根据图蕴含，这条规则会使窗口变得不一致，因此我们无法推理汽车当前的实际状态。为此，我们认为最新的断言可能是正确的，因此会从窗口中移除 $GasPedalPressed(x)$，以解决不一致性。

将由此产生的不一致性解释为旧信息被新信息推翻，这一理念在这里至关重要。具体而言，窗口可以再次滑动，并产生新的不一致性，从而推翻此前用于推翻更早事实的事实。在这种情况下，我们应再次删除导致新不一致性的最旧事实。不过，最初被删除的事实不应再次添加：推翻一次推翻并不等于证明，虽然听起来容易让人困惑。

这种方法也更契合增量推理的理念：窗口滑动时，可以逐个添加新的 ABox，并在每次添加 ABox 后进行修复，而不必一次性考虑添加所有 ABox 时的修复。这一点将在第6节中进一步讨论。现在，我们将优选修复的语义调整为适用于增量窗口维护的情形。

再次假设 $W = \{\mathcal{A}_t\}_{t \in I}$ 是 DL流上的一个窗口，由区间 I 定义。与定义2类似，我们在窗口的子集上定义一个序：

定义6（窗口序）。窗口序 $\subseteq_W$ 是 $\mathcal{A}_W$ 的子集上的偏序，定义如下：

$$\forall U, V \subseteq \mathcal{A}_W : U \subseteq_W V \text{ iff:}$$

1. $U \cap \mathcal{A}_t = V \cap \mathcal{A}_t$ 对于所有 $t \in I$，或者
2. $\exists t_0 \in I \leq n$ 使得 $U \cap \mathcal{A}_{t_0} \subsetneq V \cap \mathcal{A}_{t_0}$ 且对于所有 $t > t_0$ 都有 $U \cap \mathcal{A}_t = V \cap \mathcal{A}_t$

此定义规定了任意窗口的偏好层级，这些层级由该窗口中包含的 ABox 决定。借助这一偏序，我们现在可以定义窗口修复，其基础是优选修复的思想。然而，鉴于上文讨论的细微之处，它与原始概念存在显著差异。

定义7（窗口修复）。给定一个窗口 $W = \{\mathcal{A}_t\}_{t \in I}$，窗口修复是一个集合 $W' = \{\mathcal{A}'_t\}_{t \in I}$，其中每个 $\mathcal{A}'_t \subseteq \mathcal{A}_t$ 都满足以下条件：

1. $\mathcal{A}_{W'}$ 是一致的 2. 对于每个 $t \in I$，$\bigcup \mathcal{A}'_s$ 都是所有极大（相对于 $s \in I, s \leq t \subseteq_W$）一致子集的交集 $\mathcal{A}_t \cup \bigcup \mathcal{A}'_s s \in I, s < t$

与其依据窗口序$\subseteq_W$对 $\mathcal{A}_W$ 进行优先修复，不如通过逐步修复窗口来构建窗口修复：每次添加一个 ABox 后修复窗口，然后再将下一个 ABox 添加到修复结果中。这符合我们之前讨论的理念：在修复过程中从窗口中移除的事实，不会仅仅因为之后某次修复移除了它们的“反证”就被重新添加回来。此外，这种增量式结构使修复算法能够在窗口滑动时利用前一个窗口中已完成的修复，这非常有用。

5 OWL 2 RL 中针对窗口的增量推理

本节介绍一种算法，用于在上一节定义的修订版描述逻辑语义下进行增量推理。更具体地说，我们关注 OWL 2 RL 配置文件所基于的描述逻辑。为了构建一种也能解决不一致问题的增量算法，我们需要具备三个组成部分：

1. 窗口滑动时，能够添加进入窗口的 ABox。2. 窗口滑动时，能够移除离开窗口的事实。3. 窗口滑动时添加 ABox 所引发的不一致问题能够得到解决。

我们首先介绍如何向已物化的窗口添加单个 ABox，并假设此时不存在不一致问题。该过程还使我们能够从头构建窗口物化：从空物化开始，逐个添加 ABox。

5.1 向窗口添加 ABox

单个 RL ABox 相对于 RL TBox 的物化可以用其典范模型表示：由于 ABox 的每个模型都必须包含其典范模型，因此该模型所表示的每个事实都必定由知识库蕴涵，而典范模型中不存在的事实则不由其蕴涵。因此，要表示窗口的图蕴涵物化，我们需要构造图蕴涵窗口 ABox $\mathcal{A}_W$ 的典范模型。为此，我们将构造一个“最小”窗口模型，使其关联的图蕴涵窗口模型等于 $\mathcal{A}_W$ 的典范模型。请回忆命题 1，这种最小窗口模型不一定唯一。不过，我们会以一种能够使所需的增量推理过程相当自然的方式进行构造。因此，我们将通过该构造得到的模型称为典范窗口模型。

当窗口 W 由单个 ABox $W = \{\mathcal{A}\}$组成时，最小窗口模型只有一种可能：$\mathcal{A}$的典范模型。现在，我们通过归纳法为包含多个 ABox 的窗口构造窗口模型：

设 $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_n}\}$ 是一个由 $n > 1$ 个ABox组成的窗口，其中 $t_i < t_j$

若 $i < j$。令 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$ 为通过对窗口 $W \setminus \mathcal{A}_{t_n}$进行此构造而得到的窗口模型，并令 $\mathcal{I}_n$ 为 $\mathcal{A}_{t_n}$的典范模型。我们按如下方式构造 W 的窗口模型:

1. 将 $\mathcal{I}_n$ 加入窗口模型。2. 对于 $\mathcal{T}$中的每一条包含公理 $B \sqsubseteq A$ （或 $P \sqsubseteq R$）：如果在 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}, \mathcal{I}_n\}$ 中存在某个最小模型集 $\mathcal{I}_{u_1}, ..., \mathcal{I}_{u_m}$，使得对某个 x 有 $x \in B^{\bigoplus_i \mathcal{I}_{u_i}}$（或对某个 x, y有$P(x, y)$），则将 x 加入 $A^{\mathcal{I}_{\min 1 \le i \le m(\ )^{u_i}}}$，若它尚未在该集合中（或将$(x, y)$加入表示该同一解释的 R 的集合）。3. 当不存在任何会使该过程改变窗口模型的规则时，停止此过程（即当每条规则的头部都已被加入每一个适当的模型时）。

请注意，我们所说的最小模型集，是指去掉其中任何一个模型后，x 就不再属于通过这些解释的直和所给出的映射下 B 的像。

定义 8（典范窗口模型）。通过上述构造得到的窗口模型，就是 W的典范窗口模型。

此定义背后的想法是，推导出一个事实时，我们将其添加到推导中所用最早断言对应的模型中。这样一来，如果窗口滑动后，推导中所用的最早断言离开窗口，我们就无需计算由该断言推导出的所有事实再将它们从窗口中删除。相反，只需删除与该最早断言对应的模型，即可一并处理由该断言推导出的所有事实。这意味着，只要巧妙地选择如何处理第一项要求（向窗口添加 ABox），我们也就解决了要求 2（从窗口移除 ABox）。这一点概括如下命题:

命题 2. 设 $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_n}\}$ 是一个窗口模型，$\{\mathcal{I}_1, ..., \mathcal{I}_n\}$ 是它的典范窗口模型。那么，$\{\mathcal{I}_2, ..., \mathcal{I}_n\}$ 是$W' = W \setminus \{\mathcal{A}_{t_1}\}$的典范窗口模型。

证明。 设 $\{\mathcal{I}'_2, ..., \mathcal{I}'_n\}$ 是 W'的典范窗口模型。我们证明 $\mathcal{I}'_i = \mathcal{I}_i$。显然，对于每个概念/角色 $C^{\mathcal{I}_i} \supseteq C^{\mathcal{I}'_i}$：如果在构造 W'的典范模型时，构造典范窗口模型的第二步适用于某条规则，那么在构造 W的模型时，同一步骤也必定适用于同一条规则。反过来，显然，如果在构造 W 时该步骤适用于某条规则，而在构造 W' 时不适用，那么该规则必定涉及 $\mathcal{I}_1$。因此 $\mathcal{I}_i \subseteq \mathcal{I}'_i$ （稍微宽泛地说，这意味着任何概念或角色在 $\mathcal{I}_i$ 下的映射都是该概念或角色在 $\mathcal{I}'_i$下映射的子集），由此得证。■ ■

现在让我们通过一个例子来说明这些概念:

例 4. 设 S 是一个时间线为 $\mathbb{N}$ 的流，其中 $\mathcal{A}_i = \emptyset$ 对于 $i > 4$ 成立，并设该流中的前四个 ABox 如下:

$$\mathcal{A}_1 = \{A(a), B(a)\}$$
$$\mathcal{A}_2 = \{C(a)\}$$
$$\mathcal{A}_3 = \{A(a)\}$$
$$\mathcal{A}_4 = \{B(a)\}$$

考虑 TBox $\mathcal{T} = \{A \sqcap C \sqsubseteq D, B \sqcap D \sqsubseteq E\}$。假设我们要对由 $[1,2]$ 定义的窗口 W_1 进行推理。为了构造 W_1 的典范窗口模型，我们首先分别计算 $\mathcal{I}_1$、 $\mathcal{I}_2$ 和 $\mathcal{A}_1$、$\mathcal{A}_2$ 的典范模型。显然，它们分别是各自 ABox 的标准解释。接着，为了得到 W_1 的典范窗口模型，我们从 $\{\mathcal{I}_1\}$ 开始，并根据定义 8 添加 $\mathcal{I}_2$，得到 $\{\mathcal{I}_1, \mathcal{I}_2\}$。然后，我们看到可以应用规则 $A \sqcap C \sqsubseteq D$，因为 $a \in A^{\mathcal{I}}_1$ 和 $a \in C^{\mathcal{I}_2}$。因此，我们将 a 添加到 $D^{\mathcal{I}_1}$ 中（$\mathcal{I}_1$ 包含此推导中使用的最旧事实）。现在，我们再次检查是否可以应用任何规则，发现可以在 $\mathcal{I}_1$ 内完全应用 $B \sqcap D \sqsubseteq E$。因此，我们将 a 添加到 $E^{\mathcal{I}_1}$ 中；此后不再有规则可以应用，构造过程就此终止。
因此，可以利用 W_1 的模型来构造由 $[1,3]$定义的窗口W_2 的模型：我们从 W_1 的典范模型开始，并向其中添加 $\mathcal{I}_3$，即 $\mathcal{A}_3$的典范模型，而它再次等同于其标准解释。我们再次检查哪些规则可以应用，并发现 $A \sqcap C \sqsubseteq D$ 可以与$a \in A^{\mathcal{I}_3}$ 一起应用于$a \in C^{\mathcal{I}_2}$。请注意，尽管这条规则已应用于该个体，但它并未同时应用于这两个断言，因此我们确实会将 a 添加到 $D^{\mathcal{I}_2}$中。

现在让窗口滑动，这意味着我们现在关注由 $[2,4]$定义的 W_3。要得到 W_3的典范模型，我们首先需要区间 $[2,3]$上窗口 W_4 的典范模型。值得庆幸的是，由于典范窗口模型的构造方式，我们可以从 W_2的典范模型中删除 $\mathcal{I}_1$ 来获得该模型：如果从头计算 W_4 的典范窗口模型，我们会从 $\mathcal{A}_2$的典范模型开始，然后添加 $\mathcal{A}_3$的典范模型，接着应用所有可应用的规则，即$A \sqcap C \sqsubseteq D$。然而，显然这些步骤在构造 W_2的模型时已经完成；此外，从该模型中删除 $\mathcal{I}_1$ 后，我们便不再拥有任何由 $\mathcal{A}_1$推导出的事实。因此，显然，从 W_2 的典范窗口模型中删除 $\mathcal{I}_1$ 所得的结果，恰好就是 W_4的典范窗口模型。

要得到 W_3 的模型，我们将 $\mathcal{A}_4$的典范模型 I_4 加入其中；同样，该模型等于其标准解释，然后应用所有可能的规则。显然，第二条规则 $B \sqcap D \sqsubseteq E$ 可以应用于 $a \in B^{\mathcal{I}_4}$ 和 $a \in D^{\mathcal{I}_2}$，因此我们将 a 加入 $E^{\mathcal{I}_2}$。由于没有更多规则可以应用，由此得到 W_3的典范窗口模型。

我们现在可以描述一种用于将单个ABox添加到窗口物化中的半朴素算法：假设 $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_{n-1}}\}$ 是一个窗口，并且我们拥

算法 1 向窗口物化中添加一个 ABox

要求： 一个 TBox $\mathcal{T}$，一个窗口 $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_{n-1}}\}$，其典范模型 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$ 和一个新的 ABox $\mathcal{A}_{t_n}$

确保： $W' = W \cup \mathcal{A}_{t_n}$的典范窗口模型 $M = \{\mathcal{I}_1, ..., \mathcal{I}_{n-1}, \mathcal{I}_n\}$

1: $\Delta_1 \leftarrow \mathcal{I}_n$ ▷ $\mathcal{I}_n$ 是 $\mathcal{A}_t$的典范模型 $_n$

2: $M \leftarrow \{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$

3: **当** $\Delta_1 \neq \emptyset$ **时执行**

4: **对所有** $B \sqsubseteq A$ 在 $\mathcal{T}$**中执行**

5: **如果**$\exists a \notin B^{\mathcal{I}_M}$ 且 $\exists S \subset \{1, ..., n\}$ 使得 $a \in B^{\bigoplus_{i \in S} \mathcal{I}_i}$ 但 $a \notin B^{\bigoplus_{i \in S'} \mathcal{I}_i}$ for 任何 $S' \subsetneq S$ **然后**

6: $j \leftarrow$ 最小值 S

7: $A^{\mathcal{I}'_j} \leftarrow A^{\mathcal{I}'_j} \cup \{a\}$ ▷ $\mathcal{I}'_j$ 是一个新的解释，我们向其中添加事实 添加到 $\mathcal{I}_j$

8: **结束 if**

9: **结束 for**

10: $\Delta_2 \leftarrow \{\mathcal{I}'_1, ..., \mathcal{I}'_{n-1}\}$

11: $M \leftarrow \{\mathcal{I}_i \oplus \mathcal{I}'_i | 1 \leq i \leq n, \mathcal{I}_i \in M, \mathcal{I}'_i \in \Delta_1\}$

12: $\Delta_1 \leftarrow \Delta_2$

13: **结束 while**

有其物化，即以其典范窗口模型 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$的形式。我们的目标是计算 $W \cup \mathcal{A}_{t_n}$的典范窗口模型。根据定义 8，我们需要针对每条规则检查是否存在某个个体，该个体依据$\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\} \cup \mathcal{I}_n$中解释的某种组合满足规则体。然而我们知道，对于那些此类组合不包含 $\mathcal{I}_n$的规则，其规则头已经被添加到相应的模型中。因此我们采用如下半朴素方法：我们在第1行设置 $\Delta_1 := \mathcal{I}_n$，并在第2行设置 $M = \{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$。然后，只要增量集非空，我们就寻找所有这样的规则：其规则体由一组解释所满足，且该组合至少包含一个来自 Δ_1 的解释以及至少一个来自 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$ 的解释，如第4行和第5行所示。在第10行，我们将 $\Delta_2 = \{\mathcal{I}'_1, ..., \mathcal{I}'_{n-1}$ 设置为恰好包含这些规则头的解释，这些规则头需要分别添加到 $\mathcal{I}_1, ..., \mathcal{I}_{n-1}$（请回想，规则头必须被添加到其推导中所使用的最旧解释），而这些解释在第6行和第7行计算得出。在计算出 Δ_2之后，我们可以在第11行将 Δ_1 添加到窗口模型中以得到$\{\mathcal{I}_1, ..., \mathcal{I}_n\}$，然后将 Δ_1 设为等于 Δ_2，并通过寻找所有这样的规则来重复该过程：其规则体由一组断言所蕴含，且该组断言至少包含一条来自此新窗口模型的事实以及一条来自新 Δ_1的事实。当 $\Delta_1 = \emptyset$ 时该过程终止，如第3行所述。显然该过程会结束，因为窗口由有限数量的ABox组成，每个ABox都包含有限数量的断言。

然后，我们可以立即应用此算法，获得用于增量维护 RL 流窗口物化（典范窗口模型）的算法：

给定区间 $I = [t_0, t_1]$，其对应的窗口 W_I 及其典范

算法 2 增量窗口滑动

要求： W_I，其典范窗口模型 $\{\mathcal{I}_t\}_{t \in I}$，以及 $W_{I'}$
确保： $M = \{\mathcal{I}_t\}_{t \in I'}$
1: $M \leftarrow \{\mathcal{I}_t\}_{t \in I \cap I'}$
2: **对于所有** $\mathcal{A}_i$ 满足 $i \in I' \setminus I$ **执行**
3: 计算 $\mathcal{I}_i$
4: 通过算法 1 将 $\mathcal{I}_i$ 添加到 M
5: **结束循环**

窗口模型 $\{\mathcal{I}_t\}_{t \in I}$，我们要计算 $W_{I'}$ 的典范窗口模型，其中 $I' = [t_0 + t_s, t_1 + t_s]$，且 $t_s < t_0 - t_1$。为此，我们首先在第 1 行移除 $[t_0, t_0 + t_s]$ 中的物化事实，从而得到 $W_{[t_0+t_s,t_1]}$ 的典范窗口模型：$\{\mathcal{I}_t\}_{t \in [t_0+t_s,t_1]}$。接着，第 3 行为 $W_{]t_1,\ t_1+t_s]}$ 中的每个 ABox 构造典范模型，例如 $\mathcal{I}_1$，...$\mathcal{I}_k$，然后在第 4 行使用上述算法，将它们逐一添加到窗口模型中。最终得到 $W_{I'}$ 的典范窗口模型。

6 具有非递归不一致性的增量窗口

在本节中，我们提出对我们面向 RL 的增量窗口维护算法的一种调整，目的是处理不一致。然而，由于处理不一致和计算修复的困难，我们仅限于所谓的非递归不一致性。

非递归不一致性，通俗而言，是指可直接从 ABox 中推导出的不一致性，无需在运行时进行任何推理或物化。形式化地，我们将非递归否定包含定义如下：

定义 9（非递归否定包含）。当且仅当存在一个有限的概念集 $B_1, ..., B_n$，使得对于任意 *ABox* $\mathcal{A}$：$\mathcal{T}, \mathcal{A} \vDash B \subset \bot$ 当且仅当 $B_1^{\mathcal{I}}, ..., B_n^{\mathcal{I}} = \emptyset$，此时 $\mathcal{I}$ 是 $\mathcal{A}$ 的标准解释，即仅包含 $\mathcal{A}$ 中断言的解释。

直观来说，非递归负包含可以改写为一眼就能从 ABox 中直接看出的形式，无需先进行任何推理。这一点很重要，因为根据该定义，如果 TBox 中的所有否定包含都是非递归的，那么我们就可以改写 TBox，将每个否定包含 $B \subset \bot$ 替换为其对应的集合 $B_1 \subset \bot, ..., B_n \subset \bot$。改写后的 TBox（我们称之为不一致性范式）使我们可以先查找不一致，解决这些不一致，然后才开始推理并

向物化结果中添加事实。这一点很重要，因为这样可以避免将根据需要为修复不一致而移除的断言推导出的事实物化。因此，将 ABox $\mathcal{A}$ 添加到窗口 W时，第一步总是检查会出现哪些不一致。现在，我们引入一些记号来处理这些不一致：

定义 10. 断言集合 $S \subset \mathcal{A}_W \cup \mathcal{A}$ 是一个冲突集，当且仅当它是极小不一致的，即它是不一致的，且其所有真子集 $S' \subsetneq S$ 都是一致的。

注意，如果 S 是一个冲突集，那么不一致规范 TBox 中必定存在某个否定包含$B \subset \bot$，使得 $S \vDash B$。

例 5. 令 $\mathcal{T} = \{\forall R.B \sqsubseteq B, A \cap B \sqsubseteq \bot\}$。否定包含 $A \sqcap B \sqsubseteq \bot$ 在 $\mathcal{T}$中是递归的：由于规则 $\forall R.B \sqsubseteq B$，我们永远无法以某种方式重写 TBox，从而能够一眼判断对于所有可能的 ABox， $A \sqcap B$ 是否成立。考虑 ABox $\mathcal{A} = \{A(a)\ B(b)\ R(a,b)\}$，相对于 $\mathcal{T}$它是不一致的： $a \in \forall R.B$ ，这意味着 $B(a)$，而它与 $A(a)$ 一起导致不一致。可以通过添加否定包含 $A \sqcap \forall R.B \sqsubseteq \bot$来重写 $\mathcal{T}$，但这并不能解决问题：令 $\mathcal{A}' = \{A(a)\ B(c)\ R(a,b)\ R(b,c)\}$。相对于 $\mathcal{T}$，这个 ABox 同样是不一致的，因为 $B(c)$与 $R(b,c)$一起推出 $B(b)$，后者再与 $R(a,b)$一起推出 $B(a)$，进而像之前一样导致不一致。此外，添加的规则 $A \sqcap \forall R.B \sqsubseteq \bot$无法让我们一眼看出 $\mathcal{A}'$ 的不一致性。为此，我们需要一条规则 $A \sqcap \forall R.\forall R.B \sqsubseteq \bot$。显然，我们可以无限地继续这一过程，这意味着需要添加无限多条规则，因此我们得出结论：否定包含 $A \sqcap B \sqsubseteq \bot$ 确实是递归的。

对于冲突集 S，我们用 $\min S$ 表示 S 中最早的断言集合。这个记法很有用，因为通常我们会尝试通过删除 $\min S$ 中的元素来修复包含 S 的窗口。下面的引理以此为基础，帮助我们确定哪些冲突集必须删除 $\min S$：

引理 1. 设 $S_1, ..., S_n$ 是将 $\mathcal{A}$ 添加到 W 时产生的冲突集。那么对于任意 $1 \le i \le n$，若不存在满足 $\min S_j \subset S_i$ 的 $j \neq i$，则有如下结论：对于任意 $a \in \min S_i$， $W \cup \{\mathcal{A}\}$ 存在一个不包含 a 的窗口修复。

证明。设 S_i 满足对于所有 S_j，$\min S_j \not\subset S_i$。然后从每个 $\min\ S_j \setminus S_i$ 中取一个元素，再从 $\min\ S_i$ 中取一个元素 $a \in$，并将它们从 $\mathcal{A} \cup \mathcal{A}_W$ 中删除。我们得到 $\mathcal{A} \cup \mathcal{A}_W$ 的一个一致子集 B_0。因此必定存在一个修复 R_0，使得 $B_0 \subseteq_P R_0$。要么 $B_0 = R_0$，此时我们得到所需的修复；要么存在某个 t_0，使得 $B_0 \cap \mathcal{A}_{t_0} \subsetneq R_0 \cap \mathcal{A}_{t_0}$，并且对于所有 $t > t_0$，$B_0 \cap \mathcal{A}_t = R_0 \cap \mathcal{A}_t$。接着我们令 $B_1 = B_0 \cup (R_0 \cap \mathcal{A}_{t_0})$。注意， B_1 是一致的：假设存在一个冲突集 $S_j \subset B_1$。那么，我们删除的 $\min S_j$ 中的元素必定包含在 $R_0 \cap \mathcal{A}_{t_0}$ 中，因此 $S_j \subset \bigcup \mathcal{A}_t$。但由于对于 $t \ge t_0$， $B_1 \cap \mathcal{A}_t = R_0 \cap \mathcal{A}_t$ $t \ge t_0$，这就意味着

ply $S_j \subset R_0$这并非 p可能的。

现在，同样必定存在某个修复 R_1，使得 $B_1 \subseteq_P R_1$。或者 $B_1 = R_1$

或者存在某个 t_1，使得 $B_1 \cap \mathcal{A}_{t_1} \subsetneq R_1 \cap \mathcal{A}_{t_1}$，并且对于所有$t > t_1$都有 $B_1 \cap \mathcal{A}_t = R_1 \cap \mathcal{A}_t$。注意，$t_1 < t_0$，否则 $R_0 \subseteq_P R_1$ 且 $R_1 \neq R_0$，这与 R_0的极大性相矛盾。同样，我们设 $B_2 = B_1 \cup (R_1 \cap \mathcal{A}_{t_1})$。现在注意到 B_2 是一致的：假设存在一个冲突集 $S_j \subset B_2$。那么，我们删除的 $\min S_j$ 中的元素必定属于 $R_1 \cap \mathcal{A}_{t_1}$，因为我们没有向 B_0 添加任何早于 t_1 的元素。因此 $S_j \subset \bigcup \mathcal{A}_t$。但是，由于对于$t \geq t_1$，$B_2 \cap \mathcal{A}_t = R_1 \cap \mathcal{A}_t$ $t \geq t_1$，这将意味着 $S_j \subset R_1$，这是不可能的。
我们可以重复这一过程，但显然，这个过程必定会在某个时刻结束（我们删除了 n 个元素才得到 B_0，因此至多可以将 n 个元素加回去；实际上还要更少，因为把所有 n 都加回去会导致不一致）。因此，我们得到某个 B_k，它是一个修复，而它是通过仅向 B_0 加回元素构造出来的。此外，$a \notin B_k$，因为 $S_i \setminus \{a\} \subset B_0 \subset B_k$。因此，$B_k$ 就是所需的修复。■ ■

利用这一引理，我们可以调整将 ABox 添加到窗口物化中的过程，使其能够解决非递归不一致性。再次假设 $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_{n-1}}\}$ 是一个窗口，并且我们拥有它的物化，即其典范窗口模型 $\{\mathcal{I}_1, ..., \mathcal{I}_{n-1}\}$。还假设 $\mathcal{A}_W$ 是一致的，且 $\mathcal{T}$ 不包含递归负包含，并且处于不一致性范式。在向物化中添加任何事实之前，我们首先检查 $\mathcal{T}$ 中任何否定包含的体是否能由 $\mathcal{A}_W \cup \mathcal{A}$ 推出。如果不存在冲突，则该过程可以像算法 1 中那样继续进行。

现在假设存在冲突集 S_1，$...S_k$。我们将按如下方式构造 $\mathcal{A}_W \cup \mathcal{A}$ 的所有窗口修复的交集：首先，按从最新到最旧的顺序排列冲突集，即 $\min S_i$ 比 $\min S_{i+1}$ 更新（或同样新）。我们将从最新的冲突开始解决，并逐步处理较旧的冲突。这是因为上面的引理中的条件：

$$\min S_j \subset S_i$$

只有当 S_j 比 S_i更新（或同样新）时，该条件才成立。因此，最新的冲突集（其 min 部分属于时间戳最大的 ABox）只需相互检查，而最旧的冲突集则需要与其他所有冲突集进行检查。我们用一个简短的例子来说明这一思路：

示例 6。考虑一个由 ABox $\mathcal{A}_1 = \{A(a)\}, \mathcal{A}_2 = \{B(a)\}$ 组成的窗口，以及一个将要添加到该窗口的 ABox $\mathcal{A}_3 = \{C(a)\ D(a)\}$。设 $\mathcal{T} = \{A \sqcap B \sqcap C \sqsubseteq \bot, B \sqcap D \sqsubseteq \bot\}$。将 $\mathcal{A}_3$ 添加到窗口后，会产生 2 个冲突集：$S_1 = \{A(a)\ B(a)\ C(a)\}$ 和 $S_2 = \{B(a)\ D(a)\}$。为了解决这些冲突，我们必须按照新近性对它们排序，也就是说，先处理最小元素最新的冲突集；在本例中，该冲突集是 S_2。我们看到 $\min S_2 = \{B(a)\}$，并且不存在与 S_2同样新的冲突集，因此可以确定，必须从所有窗口修复中排除 $B(a)$，而

尤其要从所有窗口修复的交集中排除它。现在我们继续考虑 $S_1 = \{A(a)\ B(a)\ C(a)\}$，但 $\min\ S_2 \subset S_1$，因此每个修复都会通过修复 S_2来解决 S_1。这意味着无需对S_1采取任何操作，我们由此得出，只需移除 $B(a)$即可得到所需的所有修复的交集；巧合的是，它也是唯一的修复。

现在令 $S_1, ..., S_{p_1}$ 为最新的冲突集。显然，如果对于 $1 \leq i \leq p_i$，$|\min\ S_i| = 1$ 成立，那么每个修复都必须删除这个元素。令 D 为所有此类元素组成的集合：

$$D = \bigcup_{1 \leq i \leq p-1}^{|\min S_i|=1} \min S_i$$

任何修复与 D 的交集都必须为空。因此，对于任意 $1 \leq i \leq n$，如果 $S_i \cap D \neq \emptyset$，那么只需删除 D，S_i 就已经得到解决。这意味着在修复过程中，我们可以不再考虑此类 S_i。接着考虑所有满足 $1 \leq i \leq p_1$ 且 $S_i \cap D = \emptyset$的 S_i。对于这些情况，我们有 $\min |S_i| > 1$，这使得它们难以处理，因为不同的修复可能会保留或删除不同的元素。不过，前面的引理让我们可以轻松处理这些情况：如果不存在满足 $\min S_j \subset S_i$的 S_j，那么所有修复的交集必定与 $\min S_i$不相交。另一方面，如果存在这样的 S_j，那么我们可以不再考虑 S_i：事实上，如果存在这样的 S_j，那么任何修复都必须删除 $\min S_j$中的某个元素，因为它必定是最新的修复之一，而这总能解决 S_i。因此，我们需要考虑的只有满足 $1 \leq i \leq p_1$ 且 $S_i \cap D = \emptyset$ 的 S_i，其中 $\min S_i$ 关于 $\subseteq$ 预序是极小的。我们将这些元素汇集为以下集合：

$$N = \{\min S_i | 1 \leq i \leq p_1, S_i \cap D = \emptyset, \min S_i \text{ is minimal w.r.t } \subseteq\}$$

我们的引理指出，所有修复的交集与 N 中的任意集合都不相交。此外，我们知道，没有任何修复会完整包含 N 中的任何集合。因此，对于任意与 $1 \leq i \leq n$ 相关的冲突集 S_i，如果存在一个 $M \in N$ 使得 $M \subset S_i$，我们就可以不再考虑 S_i。

现在，我们迭代地重新应用这一思路来解决所有冲突：在仍需考虑的冲突集中，取时间上最新的那些。在这些最新的冲突集中，取最早断言只有一个的那些。将这些断言添加到 D，并从考虑范围中移除与 D 有非空交集的任何冲突集。取仍需考虑且包含多个最早断言的最新冲突集。在这些冲突集中，取其 $\min$ 关于 $\subseteq$最小的那些，并将其余的从考虑范围中移除。将所有 $\min$ 添加到 N，最后从考虑范围中移除所有完整包含 N 中某个集合的冲突集。

显然，应用这一方法最终会从考虑范围中移除所有冲突，留下集合 D 和集合 N。要得到所有修复的交集，剩下要做的就是从 $\mathcal{A}_W \cup \mathcal{A}$ 中移除 $D \cup \bigcup_{M \in N} M$。最后，我们需要从物化中移除由该集合推导出的事实。实际操作中，可以使用 [21] 中讨论的任意算法来完成。

算法3 添加 ABox 时解决不一致要求： $\mathcal{T}$处于递归范式， $W = \{\mathcal{A}_{t_1}, ..., \mathcal{A}_{t_{n-1}}\}$、 $\mathcal{A}$ 待添加确保： D，使得（$\mathcal{A}_W \cup \mathcal{A}$）$\setminus D$是一个窗口修复1： $C \leftarrow [\ ]$2：对于所有 $B \sqsubseteq \bot \in \mathcal{T}$3：如果 $\mathcal{A}_w \cup \mathcal{A} \vDash B$ 4：将所有满足 $S_i \vDash B$、 $\forall S_i' \subsetneq S_i$、 $S_i' \nvDash B$ 的 S_i 添加到 C 5：结束 if6：结束 for7： $D \leftarrow [\ ]$8： $N \leftarrow [\ ]$9：当 $C \neq \emptyset$ 10： $C \triangleright$ 中最新的 S_i，即 min S_i 时间戳最高的那些 S_i t11： $D \leftarrow D \cup \bigcup \{\min\ S_i | S_i \in C_1, \ |\min S_i| = 1\}$12： $C \leftarrow C \setminus \{S_i | S_i \in C, S_i \cap D \neq \emptyset\}$13： $N \leftarrow N \cup \{\min\ S_i | S_i \in C_1, \forall S_j \in C_1\ \min\ S_j \not\subset \min\ S_i\}$14： $C \leftarrow C \setminus \{S_i | S_i \in C, \exists M \in N, M \subseteq S_i\}$15：结束 while16： $D \leftarrow D \cup \bigcup M_{M \in N}$

引理2. 给定一个区间 $I = [t_0, t_1]$和一个 $t_i < t_1 \in I$，设 W_I 是由 I 定义的流 S 上的窗口，且设 W_I' 是 W_I 的窗口修复。设 $I' = [t_i, t_1]$，并设 $W_{I'}$ 是由 I' 定义的 S 上的窗口。集合 $W_{I'}' = \{\mathcal{A}_t'\}_{t \in I'} \subset W_I'$ 是 $W_{I'}$ 的窗口修复。

证明. 假设 $W_{I'}'$ 不是 $W_{I'}$ 的窗口修复。那么，由于一致的ABox的子集仍然一致，必定存在最小的 $t \in I'$，使得 $\bigcup \mathcal{A}_s'$ 不是 $\mathcal{A}_t \cup \bigcup \mathcal{A}_s'$ 的极大一致子集。因此，必定存在一个断言 $a \in \mathcal{A}_t \setminus \mathcal{A}_t'$，使得 $\{a\} \cup \bigcup \mathcal{A}_s'$ 是一致的。然而，这意味着 W_I' 不是 W_I 的窗口修复，因为 $\bigcup_{t_0 \leq s \leq t} \mathcal{A}_s' \subseteq_W \{a\} \cup \bigcup_{t_i \leq s \leq t} \mathcal{A}_s'$。这一矛盾完成了证明。■ ■

这个引理表明，当窗口滑动且我们需要丢弃最旧的 ABox 时，可以从修复中丢弃最后面的 ABox，并仍将其作为下一窗口修复的起点。

7 讨论

我们新语义的增量基础确保算法能够应对数据流高速变化这一挑战。

此外，这些理念在设计时考虑了边缘计算：瞬时 ABox 可在边缘端完成物化，然后发送到云端。若不在边缘端处理，则可在中央系统上并行物化瞬时 ABox，再将其添加到窗口中。无论采用哪种方式，这些语义从根本上都适用于对数据流进行高性能推理。

不一致问题通过依循“新的总是更好”这一核心原则得以解决，该原则为处理不一致数据提供了一种明确且合理的方式。这一原则使我们能够以更加确定性的方式执行修复，而不是仅仅猜测需要删除哪些断言。以增量方式执行修复还确保了修复过程具有低延迟，并符合“否定一个否定并不能构成证明”这一理念。

我们提出的语义和算法解决了增量处理和不一致处理方面的挑战，但仍存在一些局限。最大的局限在于，目前只能处理非递归不一致性。不过，在实践中，可以通过将递归否定包含公理重写后的版本纳入其中，并将重写深度限制在某个预先确定的范围内，来弥补这一不足。这样一来，不一致性修复算法便能够处理大多数递归不一致性，具体取决于指定的最大深度，但代价是需要检查更多规则，这很可能会影响性能。另一个局限是，这些语义只考虑数据流；然而在实践中，数据流通常会与某些静态的“背景”知识库结合进行推理。通过设置一个背景 ABox，便可以以相当简单的方式将所提出的语义修改为适用于这一场景：该背景 ABox 是每个窗口的起始 ABox，并且在修复顺序中具有最高优先级，即其时间戳为 $+\infty$。这样可以确保背景知识永远不会被删除，因为在数据流处理期间，背景知识应始终被验证为真实。

8 结论

总之，我们提出了在处理（非递归）不一致的同时，对 DL流进行增量推理的语义。我们针对 OWL2 RL 底层的 RL 描述逻辑，在这些语义下提出了半朴素推理算法。尽管所提出的语义无法直接处理背景知识和递归不一致，但理论上可以将其扩展以应对这两种情况，这或许能为实际应用场景带来价值。此外，这些算法仍有待进行实验评估，因此这是我们未来工作的核心目标。最后，将这些语义扩展到其他描述逻辑，尤其是 OWL2 另外两个配置文件底层的描述逻辑，是一项重大挑战。这两种描述逻辑都不像 RL 知识库那样拥有通用的有限“最小”模型（即典范模型），但它们确实具有类似的性质，我们相信这些性质足以支持对语义进行扩展。

参考文献

1. Baader, F., Horrocks, I., Lutz, C., Sattler, U.：《描述逻辑导论》。剑桥大学出版社（2017） 2. Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.：C-SPARQL：用于连续查询的 SPARQL。载于：第18届万维网国际会议论文集，第1061–1062页。WWW '09，美国计算机协会，纽约，纽约州，美国（2009）。https://doi.org/10.1145/1526709.1526856，https://doi.org/10 1526856 3 1145 1526709 . / . . Barbieri, D.F., Braga, D., Ceri, S., Della Valle, E., Grossniklaus, M.：基于数据流和丰富背景知识的增量推理。载于：Aroyo, L., Antoniou, G., Hyvönen, E., ten Teije, A., Stuckenschmidt, H., Cabral, L., Tudorache, T.（编）《语义网：研究与应用》，第1–15页。Springer Berlin Heidelberg，柏林，海德堡（2010） 4. Bazoobandi, H.R., Beck, H., Urbani, J.：利用 LASER 进行表达性流推理。载于：国际语义网会议，第87–103页。Springer（2017） 5. Benferhat, S., Bouraoui, Z., Tabia, K.：如何从不一致且具有优先级的 DL-Lite 知识库中选择一个优选的断言型修复？载于：第24届国际人工智能会议论文集，第1450–1456页。IJ-CAI'15，AAAI 出版社（2015） 6. Bienvenu, M., Bourgaux, C., Goasdoué, F.：优选修复语义下对不一致描述逻辑知识库的查询。载于：AAAI人工智能会议论文集，第28卷（2014） 7. Bonte, P., Calbimonte, J.P., de Leng, D., Dell'Aglio, D., Della Valle, E., Eiter, T., Giannini, F., Heintz, F., Schekotihin, K., Le-Phuoc, D., 等：为流推理研究奠定基础。《图数据与知识》（2024） 8. Bourgaux, C., Koopmann, P., Turhan, A.Y.：对时序且不一致的数据进行本体介导的查询回答。《语义网》**10**(3), 475–521 (2019) 9. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.：描述逻辑中查询回答的数据复杂度。《人工智能》**195**, 335 360 2013 10 1016 2012 10 003, – ( ). https://doi.org/https://doi.org/ . /j.artint. . . https://www.sciencedirect.com/science/article/pii/S0004370212001257 10. Ceri, S., Widom, J.：推导用于增量视图维护的产生式规则。载于：第17届超大型数据库国际会议论文集，第577–589页。VLDB '91，Morgan Kaufmann 出版社，旧金山，加利福尼亚州，美国（1991） 11. Corcho, O., Calbimonte, J.P., Jeung, H., Aberer, K.：实现语义传感器网的查询技术。《语义网》**8**(1), 43–63（2012年1月）。https://doi.org/10.4018/jswis.2012010103，https://doi.org/10 2012010103 12 4018. / jswis. . Della Valle, E., Ceri, S., Van Harmelen, F., Fensel, D.：这是一个流式世界！对快速变化的信息进行推理。《IEEE Intelligent Systems》**24**(6), 83–89（2009） 13. Dell'Aglio, D., Della Valle, E.：RDF流上的增量推理，第413–435页 22 14 05 2014 10 1201 16859 ( ). https://doi.org/ . /b - . Dell'Aglio, D., Della Valle, E., van Harmelen, F., Bernstein, A.：流推理：综述与展望——十年研究总结及未来十年的愿景。《数据科学》**1**(1-2), 59–83（2017） 15. Griffin, T., Libkin, L.：含重复项视图的增量维护。《SIGMOD Rec.》**24**(2), 328–339（1995年5月）。https://doi.org/10.1145/568271.223849，https://doi.org/10.1145/568271.223849

16. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.：OWL 2 Web 本体语言入门（第二版）（2012），https://api.semanticscholar.org/CorpusID:57542951 17. Hustadt, U., Motik, B., Sattler, U.：超表达能力描述逻辑推理的数据复杂度。载于：第 19 届国际人工智能联合会议论文集。第 466–471 页。IJCAI'05，Morgan Kaufmann 出版社，旧金山，美国加利福尼亚州（2005）18. Kontchakov, R., Zakharyaschev, M.：描述逻辑与查询重写导论，第 195–244 页。Springer 国际出版公司，Cham（2014）。https://doi.org/10 1007 978 3 319 10587 1_5 ./ - - - - 。Le-Phuoc, D., Dao-Tran, M., Xavier Parreira, J., Hauswirth, M.：一种用于统一处理关联流和关联数据的原生自适应方法。载于：Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E.（编）《语义网——ISWC 2011》。第 370–388 页。Springer Berlin Heidelberg，柏林，海德堡（2011）20. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.：描述逻辑的不一致容忍语义。载于：国际 Web 推理与规则系统会议论文集。第 103–117 页。Springer（2010）21. Motik, B., Nenov, Y., Piro, R., Horrocks, I.：重新审视 Datalog 物化结果的维护。《人工智能》**269**，76–136（2019）22. Rosati, R.：描述逻辑本体中处理不一致性的复杂度。载于：第二十二届国际人工智能联合会议论文集——第二卷。第 1057–1062 页。IJCAI'11，AAAI 出版社（2011）23. Salem, K., Beyer, K., Lindsay, B., Cochrane, R.：如何执行连接：异步增量视图维护。载于：2000 年 ACM SIGMOD 国际数据管理会议论文集。第 129–140 页。SIGMOD '00，美国计算机协会，纽约，美国纽约州（2000）。https://doi.org/10.1145/342009.335393，https://doi.org/10.1145/342009.335393