用于结合知识库中的实体解析与查询回答的框架

Ronald Fagin[1] , **Phokion G. Kolaitis**[1,2] , **Domenico Lembo**[3] , **Lucian Popa**[1] , **Federico Scafoglieri**[3]IBM Research，Almaden，美国[2]UC Santa Cruz，美国[3]罗马第一大学，意大利 fagin@us.ibm.com, kolaitis@ucsc.edu, lembo@diag.uniroma1.it, lpopa@us.ibm.com, scafoglieri@diag.uniroma1.it

摘要

我们提出一个新框架，用于结合知识库（KB）中的实体解析与查询回答，并将元组生成依赖（tgds）和等式生成依赖（egds）作为规则。我们将知识库的语义定义为涉及实体等价类和值集的特殊实例。直观而言，前者汇集所有指代同一现实世界对象的实体，而后者汇集某个属性的所有备选值。这种方法使我们既能解析实体，也能绕过数据中可能存在的不一致。随后，我们设计了一种专为这一新框架定制的追赶过程，该过程具有永不失败的特点；此外，当追赶过程终止时，它会生成一个通用解，进而可用于获得合取查询的确定答案。最后，我们讨论追赶过程不终止时所面临的挑战。

1 引言

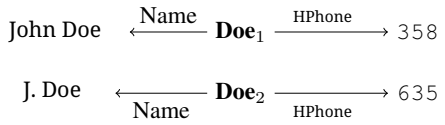
实体解析旨在确定不同的数据记录是否指向同一个现实世界对象，例如同一个个体或同一个组织等（Benjelloun 等人，2009；Papadakis 等人，2021）。本文研究知识库（KB）语境下实体解析与查询回答的结合，其中知识库由基原子以及以元组生成依赖（tgds）和相等生成依赖（egds）形式指定的规则构成。这些规则已在数据库和知识表示领域得到广泛研究，例如（Beeri 和 Vardi 1984；Cal`ı 等人，2004；Fagin 等人，2005；Baget 等人，2011；Cuenca Grau 等人，2013；Kr¨otzsch、Marx 和 Rudolph，2019）；特别是，它们既能表达数据转换，也能表达描述逻辑（DL）（Baader 等人，2007；Artale 等人，2009）和 Datalog +/- 本体（Cal`ı、Gottlob 和 Lukasiewicz，2009）中使用的经典公理。此外，egds 还用于表达实践中常见的实体解析规则，即强制两个实体相等的规则，例如（Bienvenu、Cima 和 Guti´errez-Basulto，2022）。本文考虑的知识库将实体与值区分开来，并涉及表示实体和/或来自预定义数据类型的值上的 n 元关系的 n 元谓词，$n \geq 1$。与本体领域的惯例一致，知识库的*TBox* 是其内涵组件（即规则），而知识库的数据库是其外延组件（即基原子）。

例如，图 1a 展示了一组基原子，其中 $\mathbf{Doe}_1$ 和 $\mathbf{Doe}_2$ 是实体，其余则是值（表示姓名和家庭固定电话号码）。图 2 展示了一个小型 TBox（为简单起见，仅包含相等生成依赖）。为表示实体消歧规则，我们允许相等生成依赖包含涉及内置谓词的原子公式，例如$JaccSim$。在此示例中，规则 s_1 指出，杰卡德相似度高于 0.6 的两个姓名（即字符串）必须属于同一个体。规则 s_2 和 s_3 分别规定，一个体至多有一个姓名和一个家庭固定电话号码。我们将对两个实体施加相等约束的规则称为实体等值生成依赖 规则（例如 s_1），将对两个值施加相等约束的规则称为值等值生成依赖规则（例如 s_2 和 s_3）。

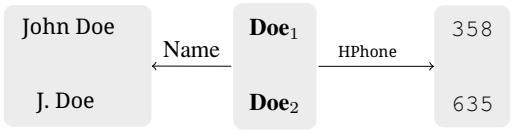
现在很容易看出，按照标准语义，图 1a 中的数据库不满足实体-egd s_1（注意，John Doe 和 J. Doe 的杰卡德相似度为 0.625）。因此，实践中的主要挑战是以一致的方式补全或修改原始知识库，同时遵守其中的所有规则。

本文提出了一个用于知识库中的实体解析和查询回答的框架，其中有效模型（称为知识库解）必须满足所有实体消歧规则以及其他所有知识库规则；此外，这些解必须包含所有原始数据（即绝不丢弃或更改任何信息）。我们的方法遵循一个直观原则：对于每个现实世界对象，解中都必须有一个节点来表示所有指代该对象的“等价”实体。为此，我们使用实体的等价类，并将其作为框架中的一等公民。此外，我们放宽了满足值 egd 的标准方式，允许解使用值集，从而汇集给定属性的所有可能值（例如，姓名或$HPhone$的第二个参数）。继续前面的示例，图 1b 展示了一组新的基原子，其中使用了等价类和值集。

我们指出，在知识库解中使用实体等价类和值集，要求对元组生成依赖和相等生成依赖的经典满足概念进行彻底修订。直观地说，我们将相等生成依赖解释为匹配依赖（Bertossi、Kolahi 和 Lakshmanan，2013年；Fan，2008年）。也就是说，当（体所表达的）条件



(a) 实体解析前。



（b）实体解析之后。

图 1：知识库的基原子

$$(s_1)\quad Name(p_1, n_1) \wedge Name(p_2, n_2) \wedge JaccSim(n_1, n_2, 0.6) \rightarrow p_1 = p_2 (s_2)\quad Name(p, n_1) \wedge Name(p, n_2) \rightarrow n_1 = n_2 (s_3)\quad HPhone(p, f_1) \wedge HPhone(p, f_2) \rightarrow f_1 = f_2$$

图 2：实体等值生成依赖与值等值生成依赖。

当数据中满足实体-egd 或值等值生成依赖（因而必须使两个实体或两个值相等）时，我们通过合并函数将它们合并。我们采用一种通用且常见的合并函数，该函数取实体的并集或值的并集。此函数实际上属于 并集类的匹配与合并函数，相关分析见（Benjelloun 等，2009；Bertossi、Kolahi 和 Lakshmanan，2013）。这样，我们便将表示某个对象的所有可能选项归为唯一的实体集合。类似地，当存在两个值，而根据 TBox 只允许一个值时，我们会显式地将它们合并。请注意，我们将实体并集视为解的“全局”特性，即实体 e 只能属于一个等价类；相反，值的并集是相对于值出现的上下文而言的“局部”特性，也就是说，某个特定值可以属于多个值集。例如，在本例的知识库中添加 $HPhone(\mathbf{Doe}_3, 358)$。那么，在满足该知识库的一个解中，358 属于两个不同的集合：图 1b 所示的集合（即 $\{358, 635\}$）以及单元素集合 $\{358\}$，后者是$\mathbf{Doe}_3$的家庭固定电话号码集合。

值得注意的是，我们为值等值生成依赖采用的语义使知识库始终有解（即模型），即使按照标准的一阶语义并不存在模型；例如，如果某个值等值生成依赖强制两个不同的值相等，那么一阶逻辑就会得出知识库不一致的结论。因此，我们可以说，对于违反值等值生成依赖的情况，我们的语义能够容忍不一致。实际上，我们会将个体属性的所有可能备选值汇集起来，而数据清洗的任务则是从中选出一个备选值。

我们还要指出，值等值生成依赖可能会对实体解析产生重大影响。举例来说，考虑图 1a 中的数据库，以及以下在该数据库上定义的新规则集

$$(s_1')\quad Name(p_1, n_1) \wedge HPhone(p_1, f) \wedge Name(p_2, n_2) \wedge HPhone(p_2, f) \wedge JaccSim(n_1, n_2, 0.6) \rightarrow p_1 = p_2 (s_2')\quad Name(p_1, n_1) \wedge Name(p_2, n_2) \wedge HPhone(p_1, f_1) \wedge HPhone(p_2, f_2) \wedge JaccSim(n_1, n_2, 0.6) \rightarrow f_1 = f_2$$

规则 s_1' 指出，姓名相似且拥有相同家庭固定电话号码的两个实体表示同一个现实世界中的个体，而规则 s_2' 指出，姓名相似的两个实体必须拥有相同的电话号码。显然，如果 TBox 仅包含规则 s_1'，就无法推断出$\mathbf{Doe}_1$ 和$\mathbf{Doe}_2$ 表示同一个个体。然而，根据规则 s_2'，一个解必须将这两个电话号码归入同一集合，这意味着$\mathbf{Doe}_1$ 和$\mathbf{Doe}_2$ 都拥有家庭固定电话号码 $\{635, 358\}$。这个合并会“触发”规则 s_1'，从而解析这些实体，也就是说，将它们归入该解的同一个等价类。

最后需要指出的是，为了最大限度地进行实体消解，我们的语义允许通过某种赋值来满足规则体：同一值变量的不同出现位置可以被不同的值集替换，只要这些值集的交集非空即可（而不要求所有出现位置都由同一个值集替换）。

例如，如果我们在图 2 的规则中加入元组生成依赖

$$HPhone(p_1, f) \wedge HPhone(p_2, f) \rightarrow SameHouse(p_1, p_2, f)$$

该依赖表明，具有相同家庭电话号码的两个实体住在同一房屋中；再将原子 $HPhone(\mathbf{Doe}_3, 358)$ 加入图 1a 的数据库中，我们便可以得出结论：由 $\mathbf{Doe}_3$ 表示的个体与由$\mathbf{Doe}_1$ 和 $\mathbf{Doe}_2$ 表示的个体住在同一房屋中（因为他们共享一个电话号码）。与这一选择一致，我们还要求元组生成依赖“传播”这类交集。于是，通过上述元组生成依赖，我们还可以推断出该房屋的电话号码是 $\{358\}$，并用形如$SameHouse([\mathbf{Doe}_1, \mathbf{Doe}_2], [\mathbf{Doe}_3], \{358\})$ 的事实表示。请注意，我们所描述的行为仅在值变量方面不同于标准行为，因为两个实体等价类要么互不相交，要么是同一个等价类。

本文将上述观点形式化，并研究查询回答，重点关注合取查询（CQ）。主要贡献如下。

- 我们提出了一个新的知识库实体解析框架，该框架由元组生成依赖和等式生成依赖组成，并为其给出了严格的语义。
- 我们定义了普适解，并证明，与标准的元组生成依赖和等式生成依赖语义一样，普适解可用于求得合取查询的确定答案。

•我们提出了一种经典追赶算法过程的变体（Beeri 和 Vardi 1984；Fagin 等人 2005），专为我们的方法设计。我们的追赶算法过程的一个重要特点是，即使存在相等生成依赖，也永远不会失败。与此同时，与其他框架一样，我们的追赶算法过

程可能不会终止。

• 我们证明，当追赶过程终止时，它会返回一个通用解（因此我们有一种算法，可用于计算合取查询的确定答案）。

• 当追赶过程不终止时，定义追赶算法的结果就变得更加复杂。我们指出，（Beeri 和 Vardi 1984）提出的策略在我们的框架中行不通。因此，需要采用一种不同的追赶算法结果定义，这留待未来研究。

本文其余部分的组织结构如下。第 2 节介绍一些有助于后续讨论的背景概念。第 3 节提出我们的框架，并给出知识库的实例、解和通用解的概念。第 4 节定义查询回答，并证明对于合取查询，可以通过在通用解上进行查询求值来求解。第 5 节首先定义我们设定下的追赶过程，并讨论追赶序列终止的情形；此时追赶过程会返回一个通用解。随后，我们探讨追赶序列不终止这一棘手情形，并证明先前提出的方法无法直接适用于我们的设定。第 6 节讨论一些相关工作，第 7 节总结全文。

2 基本概念

等价关系。若 $\mathcal{Z}$ 是一个集合，则 $\mathcal{Z}$ 上的等价关系是一个二元关系 $\theta \subseteq \mathcal{Z} \times \mathcal{Z}$，它满足自反性（即，对于每个 $x \in \mathcal{Z}$，都有 $x\,\theta\,x$）、对称性（即，对于每个 $x,\ y \in \mathcal{Z},\ x\,\theta\,y$ 意味着 $y\,\theta\,x$），以及传递性（即，对于每个 $x,\ y,\ z \in \mathcal{Z},\ x\,\theta\,y$ 和 $y\,\theta\,z$ 意味着 $x\,\theta\,z$）。

若 $x \in \mathcal{Z}$，则的等价类相对于 θ 而言是集合 $[x]_\theta = \{y \in \mathcal{Z} \mid y\,\theta\,x\}$。有时，如果上下文中 θ 的含义明确，我们会用 $[x]$ 代替 $[x]_\theta$。此外，我们也可以写作，例如 $[a,b,c]_\theta$（或简写为 $[a,b,c]$），表示由元素 a、b 和 c 组成的等价类。

等价关系 θ 的等价类具有以下众所周知的性质：(i) 每个等价类都非空；(ii) 对所有 x、$y \in \mathcal{Z}$，有 $x\theta y$ 当且仅当 $[x]_\theta = [y]_\theta$；(iii) 两个等价类要么相等，要么不相交。因此，θ 的等价类构成了 $\mathcal{Z}$ 的一个划分。$\mathcal{Z}$ 上的商集 $\mathcal{Z}/\theta$ θ 是关于 θ 的 $\mathcal{Z}$ 上所有等价类的集合，即 $\mathcal{Z}/\theta = \{[x]_\theta \mid x \in \mathcal{Z}\}$。显然，给定一个商集 $\mathcal{Z}/\theta$，有 $\mathcal{Z} = \bigcup_{[x]_\theta \in \mathcal{Z}/\theta} [x]_\theta$.

字母表与原子合取。我们考虑四个两两不相交的字母表 $\mathcal{S}_P$、$\mathcal{S}_E$、$\mathcal{S}_V$ 和 $\mathcal{S}_\mathcal{V}$。集合 $\mathcal{S}_P$ 是谓词的有限字母表；它划分为集合 $\mathcal{S}_O$ 和 $\mathcal{S}_B$，分别作为知识库谓词和内置谓词的字母表。集合 $\mathcal{S}_E$、$\mathcal{S}_V$ 和 $\mathcal{S}_\mathcal{V}$ 分别是实体、值和变量的可数无限字母表。为便于阐述，我们不区分不同的数据类型，因此 $\mathcal{S}_V$ 是包含所有可能值的单一集合。

谓词 $P \in \mathcal{S}_P$ 的参数个数就是 P的元数，记作元数(P)。对于每个 n元谓词 P，我们关联一个元组类型(P) $= \langle \rho_1, \ldots \rho_n \rangle$，使得对于每个 $1 \le i \le n$，要么是 $\rho_i = \mathbf{e}$，要么是 $\rho_i = \mathbf{v}$。该元组指定了的参数类型，即 P的每个参数是取值于实体（e）还是值（v）。我们也用类型(P, i)表示 P的第 i 个参数的类型 ρ_i。

请注意，$\mathcal{S}_B$ 中的内置谓词是特殊的预解释谓词，其参数的取值范围仅限于值；也就是说，如果 B 是一个 n元内置谓词，那么类型(B) 是 n元组 $\langle \mathbf{v} \ldots, \mathbf{v} \rangle$。第 1 节中的示例将杰卡德相似度$JaccSim$ 用作内置谓词。

一个原子是形如 $P(t_1, \ldots, t_n)$ 的表达式，其中 $P \in \mathcal{S}_P$，且每个 t_i 要么是来自 $\mathcal{S}_\mathcal{V}$ 的变量，要么是常量；常量又可以是来自 $\mathcal{S}_E$的实体（如果类型(P, i) $=$ e），也可以是来自 $\mathcal{S}_V$的值（如果类型(P, i) $=$v）。当 $t_i \in \mathcal{S}_\mathcal{V}$时，如果类型($P, i$) $=$ e，我们称其为实体变量；如果类型(P, i) $=$ v，则称其为值变量。基元是指不含变量的原子。

原子合取式 ϕ(**x**) 是由原子公式组成的表达式 $P_1(\mathbf{t}_1) \wedge \ldots \wedge P_m(\mathbf{t}_m)$，其中每个 $P_j(\mathbf{t}j)$ 都是一个原子，并且$\mathbf{t}j$ 中的每个变量都属于**x**中的变量。我们还要求，**x** 中的每个变量 x 要么是实体变量，要么是值变量；此外，如果 x 出现在谓词为内置谓词的原子中（因此 x 是值变量），那么在 ϕ(**x**) 中存在另一个原子 $P_j(\mathbf{t}j)$，使得 $P_j \in \mathcal{S}_O$，且 x 属于$\mathbf{t}j$。如果合取式不含内置谓词，则称其为无内置谓词；如果不含实体或值，则称其为无常量。

记号说明。与本节一样，在论文中出现的公式里，我们用 e 表示来自 $\mathcal{S}_E$的实体，用 v 表示来自 $\mathcal{S}_V$的值，用 c 表示常量（即 $c \in \mathcal{S}_E \cup \mathcal{S}_V$），用 x、y、w和 z 表示来自 $\mathcal{S}_\mathcal{V}$的变量，并用 t 表示项（即 $t \in \mathcal{S}_\mathcal{V} \cup \mathcal{S}_E \cup \mathcal{S}_V$）。通常，我们用$P$ 表示来自 $\mathcal{S}_P$的谓词。以上所有符号都可以带下标。在示例中，我们使用含义直观的符号，并用粗体表示实体、用等宽字体表示值。

3 框架

本节介绍用于知识库实体解析的框架的语法和语义。

语法。一个知识库 (KB) $\mathcal{K}$ 是一个二元组($\mathcal{T}$,**D**)，由一个TBox $\mathcal{T}$和一个数据库**D**组成。TBox是由有限多个元组生成依赖 (tgds)和等式生成依赖(egds)构成的集合。一个元组生成依赖 （tgd）是一个公式

$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y}\psi(\mathbf{x}, \mathbf{y})), \tag{1}$$

其中，ϕ(**x**) 和 ψ(**x**，**y**) 是原子合取式，且**x** 与**y** 没有共同变量，ψ(**x**，**y**) 是内建自由的，并且为简便起见，假设其无常量。与（Fagin 等人，2005年）一样，我们假设**x** 中的所有变量都出现在 ϕ(**x**) 中，但不一定出现在 ψ(**x**，**y**) 中。我们称 ϕ(**x**) 为该元组生成依赖的体，并称 ψ(**x**，**y**) 为其头。

相等生成依赖是一种公式

$$\forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow y = z), \tag{2}$$

其中 ϕ(**x**) 是由原子公式构成的合取式，且 y 和 z 是出现在 ϕ(**x**) 中的不同变量，使得 y 和 z 要么都是实体变量（此时得到一个 实体-*egd*），要么都是值变量（此时得到一个

值等值生成依赖）。我们称 $\phi(\mathbf{x})$ 为该相等生成依赖的体。对于 值等值生成依赖，我们要求 y 和 z 均不出现在以内置谓词为谓词的原子公式中。这可确保内置谓词的含义保持不变。我们用 body(r) 表示元组生成依赖或相等生成依赖 r 的体；此外，也可以写作 r，而不带任何量词。

示例 1. 令 $\mathcal{T}$为由以下规则组成的 TBox

(r_1) $CI(p_1, name_1, phone_1) \wedge CI(p_2, name_2, phone_2) \wedge (, ,) (, ,) (, ,) JaccSim(name_1, name_2, 0.6) \rightarrow p_1 = p_2$ (r_2) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow name_1 = name_2$ (r_3) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow phone_1 = phone_2$ (r_4) $CI(p, name, phone) \rightarrow Emp(p, comp) \wedge CEO(comp, dir)$ (r_5) $Emp(p, comp_1) \wedge Emp(p, comp_2) \rightarrow comp_1 = comp_2$ (r_6) $CI(p_1, name_1, phone) \wedge CI(p_2, name_2, phone) \rightarrow SameHouse(p_1, p_2, phone)$

这里，类型(CI)= ⟨e，v，v⟩、类型(Emp)= ⟨e，e⟩、类型(首席执行官) = ⟨e，e⟩，以及类型($SameHouse$) = ⟨e，e，v⟩。这些谓词的含义顾名思义。具体而言，谓词CI 记录个人的联系信息，而Emp 则将员工与其任职的公司关联起来。这六条规则分别对这些谓词作出断言。具体而言，规则 r_1 指出，如果两个姓名的杰卡德相似度高于 0.6，那么这两个姓名属于同一个个体。TBox 还规定，每个人只能有一个姓名（r_2）和一个固定电话号码（r_3）；每个拥有联系信息的人也都是某家公司的员工，而该公司又有一位首席执行官（r_4）；每个人只能为一家公司工作（r_5）；拥有相同固定电话号码的两个人住在同一栋房屋里（r_6）。

□

知识库 $\mathcal{K}$ 的数据库 $\mathbf{D}$ 是由以下形式的基原子构成的有限集合：$P(c_1, \ldots, c_n)$，这些基原子取自字母表 $\mathcal{S}_P$、$\mathcal{S}_E$ 和$\mathcal{S}_V$；其中，$P \in \mathcal{S}_P$，且每个 c_i 都是来自 $\mathcal{S}_E$ 的实体（如果类型$(P, i) = \mathsf{e}$），或者是来自 $\mathcal{S}_V$ 的值（如果类型 $(P, i) = \mathsf{v}$）。

示例 2. 设数据库 $\mathbf{D}$ 由以下原子构成：

(g_1) $CI(\mathbf{Doe}_1, \text{J. Doe}, 358)$ (g_4) $Emp(\mathbf{Doe}_2, \mathbf{雅虎})$ (g_2) $CI(\mathbf{Doe}_2, \text{John Doe}, 635)$ (g_5) $Emp(\mathbf{Doe}_3, \mathbf{IBM})$ (g_3) $CI(\mathbf{Doe}_3, \text{Mary Doe}, 358)$ (g_6) $CEO(\mathbf{雅虎}, \mathbf{Doe}_1)$。

具体而言，$\mathbf{D}$ 中的原子 g_1 表示$\mathbf{Doe}_1$ 的姓名是 J.Doe，电话号码是 358。此外，数据库 $\mathbf{D}$还指出$\mathbf{Doe}_1$ 的电话号码是 358（g_1），$\mathbf{Doe}_2$的姓名是 John Doe，电话号码是 635（g_2），$\mathbf{Doe}_3$的姓名是 Mary Doe，电话号码是 358（g_3），$\mathbf{Doe}_2$是**雅虎** 的员工（g_4），$\mathbf{Doe}_3$ 是**IBM** 的员工（g_5），并且$\mathbf{Doe}_1$是**雅虎** 的首席执行官（g_6）。 □

为确保内置谓词在每个知识库中具有相同的语义，我们假设有一个固定的（无限且可数的）基原子集合 GB，其形式为 $B(v_1, \ldots v_n)$，其中 B 属于 $\mathcal{S}_B$，且 $v_1, \ldots v_n$ 属于 $\mathcal{S}_V$。直观而言，GB包含所有整体上成立的内置谓词事实。给定一个知识库 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$，我们假设 $\mathbf{D} = \mathbf{D}_O \cup \mathbf{D}_B$，其中 $\mathbf{D}_O$ 仅包含谓词属于$\mathcal{S}_O$ 的基原子，而 $\mathbf{D}_B$ 是 GB 中所有原子的（有限）集合，这些原子的内置谓词和值在 $\mathcal{T}$及 $\mathbf{D}_O$中均有提及。

语义。设 $\mathcal{S}_{EN}$ 和 $\mathcal{S}_{VN}$ 是两个无限、可数且互不相交的集合，并且也与第 2 节引入的字母表互不相交——$\mathcal{S}_{EN}$ 称为实体空值集合，$\mathcal{S}_{VN}$ 称为值空值集合；二者的并集称为空值集合。我们用 签名($\mathcal{K}$) 表示知识库$\mathcal{K}$的签名，即 $\mathcal{K}$中出现的 $\mathcal{S}_P$、$\mathcal{S}_E$ 和 $\mathcal{S}_V$的符号集合。

知识库的语义通过特殊数据库来定义，这些数据库称为知识库实例，其基原子的分量要么是实体和实体空值的等价类，要么是值和非空值空值集合。

定义1. 设 $\mathcal{K}$为一个知识库，$\mathcal{S}$为 $(\mathcal{S}_E \cap \text{sig}(\mathcal{K})) \cup \mathcal{S}_{EN}$ 的子集，且 $\sim$ 为 $\mathcal{S}$ 上的等价关系，称为与 $\mathcal{K}$ 关联的等价关系。对于 $\mathcal{K}$，相对于$\sim$的一个实例 $\mathcal{I}$ 是事实集合 $P(T_1, \ldots, T_n)$，满足 $P \in \mathcal{S}_P \cap \text{sig}(\mathcal{K})$，arity($P$) $= n$，并且对每个 $1 \leq i \leq n$，都有$T_i \neq \emptyset$，且满足以下条件之一：若 type$(P, i) = \mathsf{e}$，则 $T_i \in \mathcal{S}/\sim$；若 type$(P, i) = \mathsf{v}$，则$T_i \subseteq (\mathcal{S}_V \cap \text{sig}(\mathcal{K})) \cup \mathcal{S}_{VN}$。

为了表示 $\mathcal{I}$中的一个等价类，我们可以使用符号 E （也可以带下标）。请注意，$\mathcal{I}$ 中的等价类 E 可能同时包含实体和实体空值。下文中，我们可能会直接将 E 称为实体等价类。同样，$(\mathcal{S}_V \cap sig(\mathcal{K})) \cup \mathcal{S}_{VN}$中的一个非空子集也可以简称为值集，并用符号 V 表示（也可以带下标）。我们将使用 T（也可以带下标）表示一个集合，它既可以是实体等价类，也可以是值集。

定义 2. 设 $\mathcal{K}$ 是一个知识库，$\mathcal{I}$ 是关于等价关系 $\sim$的 $\mathcal{K}$的一个实例。

$\mathcal{I}$的活动域 记作活动域($\mathcal{I}$)，是集合{T |，其中有 P(T_1、.....、T_n) $\in \mathcal{I}$且 $i \leq n$ 满足 $T = T_i$}。

我们用活动域$_E$($\mathcal{I}$)和活动域$_V$($\mathcal{I}$)分别表示实体的所有等价类构成的集合，以及活动域($\mathcal{I}$)中包含的所有值集构成的集合。显然，活动域($\mathcal{I}$) = 活动域$_E$($\mathcal{I}$) ∪ 活动域$_V$($\mathcal{I}$)。

$\mathcal{I}$的底层域记为底层($\mathcal{I}$)，它是集合底层 $_E$($\mathcal{I}$) ∪底层($_V$)($\mathcal{I}$)，其中

底层 $_E$($\mathcal{I}$) = {e | 存在 $E \in$ 活动域($_E$)($\mathcal{I}$)，且 $e \in E$}和

$$under_V(\mathcal{I}) = \{v \mid \text{there is } V \in active_V(\mathcal{I}) \text{ and } v \in V\}.$$

如果$\mathcal{I}$是$\sim$下的$\mathcal{K}$实例，那么很容易看出，$\mathcal{I}$是$\sim'$在底层($_E$)($\mathcal{I}$)上诱导出的等价关系下的$\mathcal{K}$实例。下文中，我们只考虑相对于 $\mathcal{S}$ = 底层($_E$)($\mathcal{I}$)上的等价关系的实例。此外，我们可以直接称 $\mathcal{I}$ 为 $\mathcal{K}$ 的实例，而将与 $\mathcal{I}$ 关联的等价关系留作隐含条件。

示例 3. 令 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 为一个知识库，其中 $\mathcal{T}$和$\mathbf{D}$分别如例 1 和例 2 所示。然后，考虑以下事实集合 $\mathcal{I}$：

$(d_1)CI([\text{Doe}_1], \{\text{J. Doe}\}, \{358\})$ $(d_4)Emp([\text{Doe}_2], [\text{雅虎}])$ $(d_2)CI([\text{Doe}_2], \{\text{John Doe}\}, \{635\})$ $(d_5)Emp([\text{Doe}_3], [\text{IBM}])$ $(d_3)CI([\text{Doe}_3], \{\text{Mary Doe}\}, \{358\})$ $(d_6)CEO([\text{雅虎}], [\text{Doe}_1])$$\mathcal{I}$ 是关于集合 $\mathcal{S} = \{\text{Doe}_1, \text{Doe}_2, \text{Doe}_3, \text{雅虎}, \text{IBM}\}$ 上恒等关系的一个实例。进一步，设 $\mathsf{e}_1^\perp$和 $\mathsf{e}_2^\perp$是实体空值，$\sim'$ 是定义在 $\mathcal{S} \cup \{\mathsf{e}_1^\perp, \mathsf{e}_2^\perp\}$上的等价关系，其中 $\text{Doe}_1 \sim' \text{Doe}_2$, IBM $\sim' \mathsf{e}_1^\perp$及其对称形式是 $\sim'$中除恒等关系外仅有的等价关系。以下事实集合 $\mathcal{I}'$ 是

是关于 $\mathcal{K}$ 的一个实例。

CI([**Doe**$_1$, **Doe**$_2$], {**J. Doe, John Doe**}, {358, 635}),CI([**Doe**$_3$], {**Mary Doe**}, {358}),Emp([**Doe**$_1$, **Doe**$_2$], [**雅虎**]) Emp([**Doe**$_3$], [**IBM, e**$_1^{\perp}$])CEO([**雅虎**], [**Doe**$_1$,**Doe**$_2$]) CEO([**IBM,e**$_1^{\perp}$], [**e**$_2^{\perp}$]) $SameHouse$([**Doe**$_1$, **Doe**$_2$], [**Doe**$_3$], {358}) $SameHouse$([**Doe**$_3$], [**Doe**$_1$, **Doe**$_2$], {358})。 □

为了定义实例对元组生成依赖和等式生成依赖的满足概念，我们首先引入从原子公式合取式 ϕ(x) 到知识库 $\mathcal{I}$ 的实例 $\mathcal{K}$ 的赋值概念。为了形式化这一概念，我们需要先对 ϕ(x) 进行转换，将其中每个值变量都替换为一个新变量。我们将这些新变量称为集合变量，并将转换结果记为 $\tau(\phi(\mathbf{x}))$。如果 x 是 x 中的一个值变量，我们用 SetVar$(x, \tau(\phi(\mathbf{x})))$ 表示在 $\tau(\phi(\mathbf{x}))$ 中用于替换 x 在 ϕ(x) 中各次出现的新变量集合。例如，若 $\phi(\mathbf{x}) = P_1(x, y, z) \wedge P_2(y, z) \wedge P_3(x, w)$，其中 type$(P_1)$ = ⟨e, e, v⟩、 type(P_2) = ⟨e, v⟩，且 type(P_3) = ⟨e, v⟩，则 $\tau(\phi(\mathbf{x})) = P_1(x, y, S_1^z) \wedge P_2(y, S_2^z) \wedge P_3(x, S_1^w)$，其中 S_1^z、 S_2^z、 S_1^w 是由 τ 引入的新集合变量。此外，SetVar $(z, \tau(\phi(\mathbf{x})))$ $=\{S_1^z, S_2^z\}$，且 SetVar $(w, \tau(\phi(\mathbf{x})))$ $= \{S_1^w\}$。在下文中，如果 x 是一个值变量，那么 SetVar $(x, \tau(\phi(\mathbf{x})))$ 中的每个集合变量都将以上标 x 表示。

现在我们可以正式定义赋值了。

定义 3. 设 ϕ(x)是一个原子公式的合取式，且$\mathcal{I}$ 是关于 ∼的知识库 $\mathcal{K}$ 的一个实例。从ϕ(x)到 $\mathcal{I}$ 的一个赋值是一个映射 μ，它将$\tau(\phi(\mathbf{x}))$中的变量和值映射到活动域($\mathcal{I}$)，定义如下：

1. 对每个实体变量 x， $\mu(x)$是活动域$_E(\mathcal{I})$中的一个等价类；
2. 对每个值 v， $\mu(v) = V$，满足 $V \in$ 活动域$_V(\mathcal{I})$且 $v \in V$；
3. 对每个集合变量 S， $\mu(S)$是活动域$_V(\mathcal{I})$中的一个值集；4. 对每个值变量 x，若SetVar$(x, \tau(\phi(\mathbf{x})))$ $= \{S_1^x, \ldots, S_k^x\}$，则 $\bigcap_{i=1}^{k} \mu(S_i^x) \neq \emptyset$；5. 对 $\tau(\phi(\mathbf{x}))$中的每个原子 $P(x, S, e, v)$， $\mathcal{I}$ 包含一个形式为 $P(\mu(x)\ \mu(S)\ [e]_{\sim}, \mu(v))$的事实，其中 x 是一个实体变量， S 是一个集合变量， e 是 $\mathcal{S}_E$中的一个实体，且 v 是 $\mathcal{S}_V$ 中的一个值（对于不同形式的原子公式，定义可按显然的方式推广）。

换言之，上述定义表示，赋值将每个实体变量映射到一个实体等价类（条件 1），将每个值映射到一个包含该值的值集（条件 2），并将值变量的每次出现映射到一个值集（条件 3），同时确保同一值变量的多次出现所映射到的值集之间存在非空交集（条件 4）。这体现了这样一种直觉：由于值域上的谓词参数是通过值集来解释的，因此，当这些值集存在非空交集时，这些参数之间的连接成立。最后，条件 5 表明， $\tau(\phi(\mathbf{x}))$ 在 $\mathcal{I}$ 中得到“实现”。注意，赋值也会将值映射到值集（条件 2）。因此，赋值可以将 ϕ(x) 中的原子公式映射到 $\mathcal{I}$ 中的事实，因为值域上的谓词参数在 $\mathcal{I}$ 中通过值集进行实例化。

如果 P(t) 是 ϕ(x) 的一个原子公式，我们将条件 5 中 $\mathcal{I}$ 的事实记作 $\mu(P(\mathbf{t}))$，并称其为 P(t) 的 μ-像（在 $\mathcal{I}$中）。我们将集合 $\{\mu(P(\mathbf{t}))$（其中 | P(t) 出现在 ϕ(x) 中）记作 $\mu(\phi(\mathbf{x}))\}$，并称其为 ϕ(x) 的 μ-像（在 $\mathcal{I}$中）。

例 4。 考虑例 1 中的元组生成依赖 r_6 以及例 3 中给出的实例$\mathcal{I}'$，并将 τ 应用于 r_6 的体，得到$CI(p_1, S^{name_1}, S_1^{phone}) \wedge CI(p_2, S^{name_2}, S_2^{phone})$。令 μ为以下映射：

$\mu(p_1)$ = [**Doe**$_1$,**Doe**$_2$], $\mu(S^{name_1})$ = {J. Doe,John Doe}
$\mu\ S_1^{phone} = \{358, 635\}$ $\mu\ p_2$ = [$_3$] ( ) ,( ) **Doe**( ) Mary Doe,( ) = {358}.
$\mu\ S^{name_2}$ = { } $\mu\ S_2^{phone}$

很容易看出， μ 是从 r_6 的体到$\mathcal{I}'$ 的一个赋值（注意 $\mu(S_1^{phone})$ 与$\mu(S_2^{phone})$之间的非空交集）。现在将 τ 应用于 r_6 的头，得到$SameHouse(p_1, p_2, R^{phone})$。以下映射 μ' 是从 $SameHouse(p_1, p_2,$ 电话)到 $\mathcal{I}'$的一个赋值：

$$\mu'(p_1) = [\mathbf{Doe}_1, \mathbf{Doe}_2], \mu'(p_2) = [\mathbf{Doe}_3], \mu'(R^{phone}) = \{358\}.$$

We are now ready to define the semantics of tgds and egds.

定义 4. 知识库 $\mathcal{K}$ 的一个实例 $\mathcal{I}$ 满足：

• 形如 (1) 的元组生成依赖：对于每个从 ϕ(x) 到 $\mathcal{I}$的赋值 μ，都存在一个从 ψ(x, y) 到 $\mathcal{I}$的赋值 μ'，使得对于 x 中同时出现在 ϕ(x) 和 ψ(x, y) 中的每个 x：– 如果 x 是实体变量，则 $\mu(x) = \mu'(x)$；– 如果 x 是值变量，则 $\bigcap_{i=1}^{k} \mu(S_i^x) \subseteq \bigcap_{i=1}^{\ell} \mu'(R_i^x)$，其中 $\{S_1^x, \ldots, S_k^x\}$ =SetVar$(x, \tau(\phi(\mathbf{x})))$ 且 $\{R_1^x, \ldots, R_\ell^x\}$ = SetVar$(x, \tau(\psi(\mathbf{x}, \mathbf{y})))$。每个这样的赋值 μ' 称为从 μ 到 $\mathcal{I}$ 的头兼容元组生成依赖扩展（或简称从 μ 到 $\mathcal{I}$ 的元组生成依赖扩展）。

• 形如 (2) 的实体-egd，当每个赋值 μ 从ϕ(x) 到$\mathcal{I}$均满足 $\mu(y) = \mu(z)$ 时；

• 形如 (2) 的值等值生成依赖，当每个赋值 μ从ϕ(x) 到 $\mathcal{I}$均满足 $\mu(S^y) = \mu(S^z)$，且对所有集合变量S^y， $S^z \in SetVar(y, \tau(\phi(\mathbf{x}))) \cup SetVar(z, \tau(\phi(\mathbf{x})))$ 时。

换言之，定义 4 对规则 r 的体中的每个赋值 μ 从 $\mathcal{I}$作出如下规定：

(1) 如果 r 是元组生成依赖，该定义针对边界变量（即同时出现在该元组生成依赖的体和头中的变量）规定了两种不同的行为。具体而言，要使该元组生成依赖得到满足，必须能够扩展 μ，使得 (i) 每个边界实体变量在规则的体和头中都被赋值为完全相同的等价类（这与元组生成依赖中边界变量的标准语义一致）；并且 (ii) 元组生成依赖的体中各个边界值变量 x 出现位置所赋值集的交集，包含于该元组生成依赖的头中 x 各出现位置所赋值集的交集。请注意，尽管 μ 会将 ϕ(x) 中实体变量的所有出现位置映射到同一个等价类，但它可能会将值变量的多个出现位置映射到具有非空交集的不同值集；因此，将这个交集“传递”到头中是自然的。

(2) 如果 r 是一个实体-egd，那么满足的概念是标准的，因为它要求 μ 将 y 和 z 赋给同一个实体等价类。

(3) 如果 r 是值等值生成依赖，我们再次必须考虑规则体中 y 的每次出现都可以由 r 赋予不同的值集

（前提是所有这些值集的交集非空），z 的每次出现也同理。根据定义，要满足值等值生成依赖，所有这些值集必须相等。

示例 5. $\mathcal{I}'$ 满足例 3 中的事实以及例 1 中的元组生成依赖 r_6，因为 μ' 是从 r_6 的体中唯一赋值 μ 得到的头兼容的元组生成依赖扩展。事实上，$\mu'(p_1) = \mu(p_1)$，$\mu'(p_2) = \mu(p_2)$，且 $\mu'(S^{phone}) = \mu(S_1^{phone}) \cap \mu(S_2^{phone})$（参见例 4）。也容易看出，$\mathcal{I}'$ 满足例 1 中的所有规则。□

最后，我们定义什么情况下一个实例是知识库的解。

定义 5. 设 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 是一个知识库，$\mathcal{I}$ 是一个实例。
- $\mathcal{I}$ 满足 $\mathcal{T}$，当且仅当 $\mathcal{I}$ 满足 $\mathcal{T}$ 中的每个元组生成依赖和相等生成依赖。

若存在 $\mathcal{I}$ 中的事实 $P(T_1, \ldots, T_n)$，使得 $c_i \in T_i$，对于 $1 \leq i \leq n$，则
- $\mathcal{I}$ 满足 $\mathbf{D}$ 中的基原子 $P(c_1, \ldots, c_n)$。
- $\mathcal{I}$ 满足 $\mathbf{D}$，当且仅当 $\mathcal{I}$ 满足 $\mathbf{D}$ 中的所有基原子。
- $\mathcal{I}$ 是 $\mathcal{K}$ 的一个解，记作 $\mathcal{I} \models \mathcal{K}$，当且仅当 $\mathcal{I}$ 满足 $\mathcal{T}$ 和 $\mathbf{D}$。知识库 $\mathcal{K}$ 的所有解构成的集合记作 $Sol(\mathcal{K})$，即 $Sol(\mathcal{K}) = \{\mathcal{I} \mid \mathcal{I} \models \mathcal{K}\}$。

示例6。考虑例3中给出的实例 $\mathcal{I}$ 和 $\mathcal{I}'$，它们对应于例1和例2中的 $\mathcal{K}$。容易看出，$\mathcal{I}$ 不是 $\mathcal{K}$ 的解，而 $\mathcal{I}'$ 是 $\mathcal{K}$ 的解。□

通用解。众所周知，通用解具有良好的性质，因此是首选解（参见例如 (Fagin 等人 2005年；Cal`i、Gottlob 和 Kifer 2013年；Calvanese 等人 2007年)）。为了在我们的框架中引入通用解的概念，我们首先需要调整同态的概念。我们先介绍一些辅助定义和记号。

定义 6. 设 $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ 和 $\mathbf{T}' = \langle T_1', \ldots, T_n' \rangle$ 是两个元组，其中每个 T_i 和每个 T_i' 要么是由实体和实体空值构成的集合，要么是由值和值空值构成的集合。
- $\mathbf{T}'$ 支配 $\mathbf{T}$，记作 $\mathbf{T} \preceq \mathbf{T}'$，若对于所有 i，$T_i \subseteq T_i'$。
- $\mathbf{T}'$ 严格支配 $\mathbf{T}$，记作 $\mathbf{T} \prec \mathbf{T}'$，若 $\mathbf{T} \preceq \mathbf{T}'$ 且 $\mathbf{T} \neq \mathbf{T}'$。

- 设 $P(\mathbf{T})$ 和 $P(\mathbf{T}')$ 是事实。$P(\mathbf{T}')$ 支配 $P(\mathbf{T})$，记作 $P(\mathbf{T}) \preceq P(\mathbf{T}')$，若 $\mathbf{T} \preceq \mathbf{T}'$；$P(\mathbf{T}')$ 严格支配 $P(\mathbf{T})$，记作 $P(\mathbf{T}) \prec P(\mathbf{T}')$，若 $\mathbf{T} \prec \mathbf{T}'$。

定义 7. 设 $\mathcal{I}_1$ 和 $\mathcal{I}_2$ 是知识库 $\mathcal{K}$ 的两个实例。同态 $h: \mathcal{I}_1 \to \mathcal{I}_2$ 是一个从 $under(\mathcal{I}_1)$ 的元素到 $under(\mathcal{I}_2)$ 的元素的映射，满足：

1. 对于 $\text{under}_E(\mathcal{I}_1) \cap \mathcal{S}_E$ 中的每个实体 e，$h(e) = e$；2. 对于 $\text{under}_E(\mathcal{I}_1) \cap \mathcal{S}_{EN}$ 中的每个实体空值 $e_\perp$，$h(e_\perp)$ 属于 $\text{under}_E(\mathcal{I}_2)$；3. 对于 $\text{under}_V(\mathcal{I}_1) \cap \mathcal{S}_V$ 中的每个值 v，$h(v) = v$；4. 对于 $\text{under}_V(\mathcal{I}_1) \cap \mathcal{S}_{VN}$ 中的每个值空值 $v_\perp$，$h(v_\perp)$ 属于 $\text{under}_V(\mathcal{I}_2)$；5. 对于 $\mathcal{I}_1$ 中的每个 $P(T_1, \ldots, T_n)$，都存在 $\mathcal{I}_2$ 中的一个 $P(U_1, \ldots, U_n)$，使得 $P(h(T_1) \ldots, h(T_n)) \preceq P(U_1, \ldots, U_n)$，其中，对于 $1 \leq i \leq n$，$h(T_i) = \{h(x) \mid x \in T_i\}$。

在下文中，我们可以用 $h(P(T_1, \ldots, T_n))$ 代替 $P(h(T_1), \ldots, h(T_n))$，并可以用 $h(\langle T_1, \ldots, T_n \rangle)$ 表示元组 $\langle h(T_1), \ldots, h(T_n) \rangle$。

现在我们定义通用解这一关键概念。

定义 8. 知识库 $\mathcal{K}$ 的一个解 $\mathcal{U}$ 称为通用的，若对于每个 $\mathcal{I} \in Sol(\mathcal{K})$，都存在一个同态 $h: \mathcal{U} \to \mathcal{I}$。

例3中的实例 $\mathcal{I}'$ 是 $\mathcal{K}$ 的一个通用解，从 $\mathcal{I}'$ 中消去 $\mathbf{e}_1^\perp$ 后得到的实例也是如此。

两个实例 $\mathcal{I}_1$ 和 $\mathcal{I}_2$ 是同态等价的，当且仅当存在同态 $h: \mathcal{I}_1 \to \mathcal{I}_2$ 和 $h': \mathcal{I}_2 \to \mathcal{I}_1$。所有通用解都是同态等价的。

4 查询回答

合取查询（CQ）q 是一个公式 $\exists \mathbf{y} \phi(\mathbf{x}, \mathbf{y})$，其中 $\phi(\mathbf{x}, \mathbf{y})$ 是由原子公式构成的无内置谓词合取式。q 的元数是其在 $\mathbf{x}$ 中的自由变量数量；我们通常写作 $q(\mathbf{x})$，而不只写 q，以指明 q 的自由变量。

令 $q(\mathbf{x})$：$\exists \mathbf{x} \phi(\mathbf{x}, \mathbf{y})$ 为一个合取查询，其中 $\mathbf{x} = x_1, \ldots, x_n$。给定一个知识库 $\mathcal{K}$ 和 $\mathcal{K}$ 的一个实例 $\mathcal{I}$，q 在 $\mathcal{I}$ 上的答案记作 $q^{\mathcal{I}}$，它是所有满足以下条件的元组 $\langle T_1, \ldots, T_n \rangle$ 的集合：存在一个从 $\phi(\mathbf{x}, \mathbf{y})$ 到 $\mathcal{I}$ 的赋值 μ，使得

- $T_i = \mu(x_i)$，如果 x_i 是实体变量；
- $T_i = \bigcup_{j=1}^{k} \mu(S_j^{x_i})$，如果 x_i 是值变量，其中 $\{S_1^{x_i}, \ldots, S_k^{x_i}\} = SetVar(x_i, \tau(\phi(\mathbf{x}, \mathbf{y})))$.

我们也称 μ 是一个从 $q(\mathbf{x})$ 到 $\mathcal{I}$ 的赋值。

例7。令 $q(x)$ 为合取查询

$\exists p_1, p_2\ CI(p_1, \text{J. Doe}\ x) \wedge CI(p_2, \text{Mary Doe}\ x)$，查询 J. Doe 和 Mary Doe（即名为这两者的实体）共有的电话号码。例3中的实例 $\mathcal{I}'$ 上，$q(x)$ 的答案是 $q^{\mathcal{I}'} = \{\langle \{358\} \rangle\}$。这是通过赋值 μ_q 得到的

$\mu_q(p_1) = [\mathbf{Doe}_1, \mathbf{Doe}_2], \mu_q(\text{J. Doe}) = \{\text{J. Doe}, \text{John Doe}\}$
$\mu_q(S_1^x) = \{358, 635\}$，$\mu_q(p_2) = [\mathbf{Doe}_3]$
$\mu_q(\text{Mary Doe}) = \{\text{Mary Doe}\}$，$\mu_q(S_2^x) = \{358\}$
如果 $q_1(x)$：$\exists z$ 首席执行官(z, x) 是询问首席执行官的查询，那么 $q_1^{\mathcal{I}'} = \{\langle [\mathbf{Doe}_1, \mathbf{Doe}_2] \rangle, \langle [\mathbf{e}_2^\perp] \rangle\}$。□

下一个结果说明，在我们的框架中，合取查询在同态下保持不变。

命题1. 设 q 是一个合取查询，$\mathcal{K}$ 是一个知识库。若 $\mathcal{I}_1$ 和 $\mathcal{I}_2$ 是 $\mathcal{K}$ 的两个实例，且 $h: \mathcal{I}_1 \to \mathcal{I}_2$ 是从 $\mathcal{I}_1$ 到 $\mathcal{I}_2$ 的同态，那么对于每个 $\mathbf{T} \in q^{\mathcal{I}_1}$，都存在 $\mathbf{U} \in q^{\mathcal{I}_2}$，使得 $h(\mathbf{T}) \preceq \mathbf{U}$。

证明。不失一般性，考虑查询 $q(x, y)$，其定义为 $\exists w\, P_1(x, y, e) \wedge P_2(x, y, w, v)$，其中 x 是一个实体变量，y 和 w 是值变量，e 是来自 $\mathcal{S}_E$ 的一个实体，而 v 是来自 $\mathcal{S}_V$ 的一个值。将 τ 算子应用于 $q(x, y)$（其中的原子公式合取式），得到 $P_1(x, S_1^y, e) \wedge P_2(x, S_2^y, S^w, v)$。根据实例上查询答案的定义，对于每个元组 $\langle E, V \rangle \in q^{\mathcal{I}_1}$，存在一个从 $q(x, y)$ 到 $\mathcal{I}_1$ 的赋值 μ，使得 $\mu(x) = E$，$\mu(S_1^y) = V_1$，$\mu(S_2^y) = V_2$，$\mu(S^w) = V_3$，$\mu(v) = V_4$，$V = V_1 \cap V_2 = \emptyset$，$v \in V_4$，并且 $\mathcal{I}_1$ 包含事实 $P_1(E, V_1, [e]_{\sim_1})$ 和 $P_2(E, V_2, V_3, V_4)$（其中 $\sim_1$ 是与 $\mathcal{I}_1$ 相关联的等价关系）。由于 h 是从 $\mathcal{I}_1$ 到 $\mathcal{I}_2$ 的同态，实例 $\mathcal{I}_2$ 必须包含事实 $P_1(\hat{}E, \hat{}V_1, \hat{}E_1)$ 和 $P_2(\hat{}E, \hat{}V_2, \hat{}V_3, \hat{}V_4)$，使得 $P_1(h(E)\ h(V_1)\ h([e]_{\sim_1})) \preceq P_1(E, V_1, E_1)$，并且 $P_2(h(E)\ h(V_2)\ h(V_3)\ h(V_4)) \preceq P_2(E, V_2, V_3, V_4)$。然后，我们构造一个从 $P_1(x, S_1^y, e) \wedge P_2(x, S_2^y, S^w, v)$ 中的变量和值到 $_E(\mathcal{I}_2)$ 的活动域的映射 μ'，如下所示：$\mu'(x) = \hat{}E$，$\mu'(S_1^y) = \hat{}V_1$，$\mu'(S_2^y) = \hat{}V_2$，$\mu'(S^w) = \hat{}V_3$，并且

$\mu'(v) = \hat{V}_4$。不难看出，μ' 是一个从 $q(x, y)$到 $\mathcal{I}_2$的赋值。具体而言：

• 由于 $e \in [e]_{\sim^1}$ 且 $h(e) = e$，因此 $e \in h([e]_{\sim^1})$；又由于 $\hat{} \hat{} h([e]_{\sim^1}) \subseteq E_1$，可推出 $e \in E_1$，这意味着 $\hat{} E_1 = [e]_{\sim^2}$（其中 $\sim^2$ 是与 $\mathcal{I}_2$相关联的等价关系）；

• 由于 $V_1 \cap V_2 \neq \emptyset$ 且由于 $h(V_1 \cap V_2) \subseteq h(V_1) \cap h(V_2) \subseteq \hat{V}_1 \cap \hat{V}_2$，可推出 $\hat{} V_1 \cap \hat{} V_2 \neq \emptyset$；

• 由于 $v \in V_4$ 且 $h(v) = v$，因此 $v \in h(V_4)$；又由于 $\hat{} \hat{} h(V_4) \subseteq V_4$，可推出 $v \in V_4$。

上述赋值 μ' 证明了 $\langle \hat{} E, \hat{} V \rangle \in q^{\mathcal{I}_2}$，其中 $\hat{} \hat{} \hat{} V = V_1 \cap V_2$。这证明了该命题，因为如前所述，$h(E) \subseteq \hat{} \hat{} E$，且 $h(V) \subseteq V$。 □

查询知识库 $\mathcal{K}$ 时，我们关注对 $\mathcal{K}$ 的所有解进行推理。我们将确定答案的经典概念调整为适用于我们的框架。若元组 $\mathbf{T} = \langle T_1, \ldots, T_n \rangle$ 中的每个 T_i 都非空且不含空值，则称其为无空值元组。

定义 9。设 q 是一个合取查询，$\mathcal{K}$ 是一个知识库。若无空值元组 $\mathbf{T}$ 满足以下条件，则称其为关于 $\mathcal{K}$ 的 q 的确定答案：1. 对于 $\mathcal{K}$ 的每个解 $\mathcal{I}$，都存在一个元组 $\mathbf{T}' \in q^{\mathcal{I}}$，使得$\mathbf{T} \leq \mathbf{T}'$；2. 不存在满足条件 1。且 $< \mathbf{T}'$ 的无空值元组 $\mathbf{T}'$。我们用 $cert(q, \mathcal{K})$ 表示 q 的确定答案集合。

请注意，上述定义中的第二个条件断言，确定答案必须是在满足条件 1 的元组中，关于定义 6 $\leq$给出的元组支配序最大的无空值元组。下面的示例说明了这一条件的重要性。

例 8. 在我们当前示例所用知识库 $\mathcal{K}$ 的每个解中，**Doe**$_1$ 和 **Doe**$_2$ 都属于同一个表示首席执行官的等价类。因此，无空值元组$\langle$**{Doe**$_1$**,Doe**$_2$**}**$\rangle$ 是查询 $q_1(x)$:$\exists z$首席执行官(z, x) 的确定答案。注意，无空值元组 $\langle$**{Doe**$_1$**}**$\rangle$满足定义 9 中的第一个条件（但不满足第二个条件）。然而，我们不希望该元组成为确定答案，因为它在任何解中都不是一个等价类。 □

接下来的结果表明，如果两个实体集合出现在同一个确定答案中，或出现在不同的确定答案中，那么它们要么是同一个集合，要么互不相交。具体而言，出现在确定答案中的实体集合可以视为某个等价关系的等价类。

命题 2. </style}设 是一个元数为的合取查询，是一个知识库，且$\mathbf{T} = \langle T_1, \ldots, T_n \rangle$和$\mathbf{T}' = \langle T'_1, \ldots, T'_n \rangle$是关于 $\mathcal{K}$的 q 的两个确定答案。如果 T_i 和 T'_j 是实体集合，那么 $T_i = T'_j$或 $T_i \cap T'_j = \emptyset$，其中 $1 \leq i, \ j \leq n$。

证明。反设 T_i、T'_j 是两个非空实体集合，且 $T_i \neq T'_j$和 $T_i \cap T'_j \neq \emptyset$。那么 $T_i \subsetneq T_i \cup T'_j$或 $T'_j \subsetneq T_i \cup T'_j$。不妨假设$T_i \subsetneq T_i \cup T'_j$。由于 $\langle T_1, \ldots, T_n \rangle$和 $\langle T'_1, \ldots, T'_n \rangle$是 q 的确定答案，对于每个解 $\mathcal{I}$，都存在元组 $\langle L_1, \ldots, L_n \rangle$、$\langle L'_1, \ldots, L'_n \rangle$属于 $q^{\mathcal{I}}$，使得对于每个k，都有 $T_k \subseteq L_k$ 和 $T'_k \subseteq L'_k$。因此我们有 $T_i \subseteq L_i$ 和 $T'_j \subseteq L'_j$。由于 $T_i \cap T'_j \neq \emptyset$，可知$L_i \cap L'_j \neq \emptyset$。然而，$L_i$、$L'_j$ 是解 $\mathcal{I}$中的等价类，因此它们必定相同，即 $L_i = L'_j$。由此可得 $T_i \cup T'_j \subseteq L_i = L'_j$。现在考虑元组 $\langle T_1, \ldots, T_{i-1}, T_i \cup T_j, T_{i+1}, \ldots, T_n \rangle$，它由 $\langle T_1, \ldots, T_n \rangle$替换 T_i为 $T_i \cup T_j$得到。注意，$\langle T_1, \ldots, T_{i-1}, T_i, T_{i+1}, \ldots, T_n \rangle < \langle T_1, \ldots, T_{i-1}, T_i \cup T_j, \ T_{i+1}, \ldots\ldots, \ T_n \rangle$，因为 $T_i \subsetneq T_i \cup T'_j$。然而，对于 $\mathcal{K}$的每个解 $\mathcal{I}$，都有 $T_i \cup T'_j \subseteq L_i$；对于所有满足 $k \neq i$的 $1 \leq k \leq n$，都有 $T_k \subseteq L_k$。这意味着 $\langle T_1, \ldots, T_{i-1}, T_i \cup T_j, T_{i+1}, \ldots, T_n \rangle$满足定义 9 的条件（$i$），因此 $\langle T_1, \ldots, T_n \rangle$不满足定义 9 的条件（$ii$），所以它不是确定答案，这就产生了矛盾。最后请注意，如果假设 $T'_j \subsetneq T_i \cup T'_j$，那么用类似的论证即可推出$\langle T'_1, \ldots, T'_n \rangle$不是确定答案，与此矛盾。

□

以下性质可由上述命题轻易推出。

推论 1. 设 $\mathcal{K}$是一个知识库，q是一个合取查询，$\langle T_1, \ldots, T_n \rangle$是关于 $\mathcal{K}$的 q 的一个确定答案。如果 T_i 和 T_j是满足 $i \neq j$的实体集合，那么 $T_i = T_j$或$T_i \cap T_j = \emptyset$。

证明。当$\mathbf{T} = \mathbf{T}'$时，由命题 2 可得。 □

在数据交换及相关领域中，可以通过在一个万能解上对合取查询求值，然后应用算子 $\downarrow$ 来获得合取查询的确定答案；该算子会消除包含空值的元组（例如，参见（Fagin 等人 2005年；Calvanese 等人 2007年））。算子 $\downarrow$可以很容易地按如下方式调整，以适用于我们的框架。若 E 是实体和实体空值的集合，则 $E\downarrow = \{e \mid e \in \mathcal{S}_E \cap E\}$。类似地，若 V 是非空的值和数值空值集合，则$V\downarrow = \{v \mid v \in \mathcal{S}_V \cap V\}$。换言之，$\downarrow$ 会从E 和 V 中移除所有空值。若 $\langle T_1, \ldots, T_n \rangle$ 是一个元组，且其中每个 T_i 都是实体和实体空值的集合，或值和数值空值的集合，则我们定义 $\langle T_1, \ldots, T_n \rangle\downarrow = \langle T_1\downarrow, \ldots, T_n\downarrow \rangle$。最后，给定一个由上述形式的元组组成的集合 Θ，$\Theta\downarrow$ 是从 Θ 中移除 Θ 中所有包含某个 T_i 的元组（其中 $T_i\downarrow = \emptyset$），并将 Θ 中每个其他元组 $\langle T_1, \ldots, T_n \rangle$替换为元组 $\langle T_1, \ldots, T_n \rangle\downarrow$后得到的集合。

下一个示例表明，在我们的框架中，CQ q 的确定答案并不总能通过在通用解上计算 q，再应用 $\downarrow$得到。

示例 9. 设 P_1 和 P_2 满足 类型(P_1) =类型$(P_2) = \langle$e,v$\rangle$，$\mathcal{T}'$ 是由以下规则组成的 TBox

$$P_1(x, y) \rightarrow P_2(x, y), \ P_1(x, y) \wedge P_1(x, z) \rightarrow y = z$$

并且 $\mathbf{D}' = \{P_1$**(e,1)** P_1**(e,2)**$\}$。考虑 $\mathcal{K}' = (\mathcal{T}'$,$\mathbf{D}')$ 的以下通用解

$$\mathcal{I}_1 = \{P_1([\mathbf{e}], \{1, 2\}), P_2([\mathbf{e}], \{1, 2\})\}$$
$$\mathcal{I}_2 = \{P_1([\mathbf{e}], \{1, 2\}), P_2([\mathbf{e}], \{1\}), P_2([\mathbf{e}], \{1, 2\})\}$$

以及查询 $q\ x\ y \quad P_2\ x\ y$ $q^{\mathcal{I}_1}\downarrow = q^{\mathcal{I}_1} = \{\langle [\ ]\ \{1, 2\}\rangle\}$ $q^{\mathcal{I}_2}\downarrow = q^{\mathcal{I}_2} = \{\langle [\ ]\ \{1, 2\}\rangle\ \langle [\ ]\ \{1\}\rangle\}$ **(,) : (,)**。那么 **e**, , 且 **e**, , **e**, 。因此，$q^{\mathcal{I}_2}\downarrow$与关于 $\mathcal{K}'$ 的 q 的确定答案集合并不一致，因为元组 $\langle$[**e**], $\{1\}\rangle$ 不满足定义 9 中的第二个条件。 □

直观地说，在上述示例中，通用解 $\mathcal{I}_2$并非“最小”，因为事实 $P_2([\mathbf{e}], \{1\})$ 在 $\mathcal{I}_2$ 中受到 $\mathcal{I}_2$ 的另一个事实支配，即 $P_2([\mathbf{e}], \{1,2\})$。这种行为导致对 $\mathcal{I}_2$ 进行查询时，答案中还包含相对于元组支配而言并非极大的元组（参见定义 9）。还需注意的是，存在空值时，情况会更加复杂。例如，我们可能有一个实例，其中同时包含事实 $Q([\mathbf{e}], \{1,2,n\})$ 和 $Q([\mathbf{e}], \{1,n'\})$，其中 n 和 n' 是两个不同的值空值。在这种情况下，解中没有任何事实受到另一事实支配，但消除空值后仍会出现“非最小性”。这表明，除了消除空值之外，我们还需要进行一些额外处理。我们通过进一步执行 约简步骤来修改算子 $\downarrow$。

定义 10. 给定一个查询 q 和知识库 $\mathcal{K}$ 的一个实例 $\mathcal{I}$，我们用 $q^{\mathcal{I}}{\downarrow_\rho}$ 表示通过从 $q^{\mathcal{I}}{\downarrow}$ 中移除被集合中其他元组严格支配的所有元组而得到的无空值元组集合。也就是说，如果 $q^{\mathcal{I}}{\downarrow}$ 中有两个元组 $\mathbf{T}$ 和 $\mathbf{T}'$，满足$\mathbf{T} < \mathbf{T}'$，则从 $q^{\mathcal{I}}{\downarrow}$ 中移除 $\mathbf{T}$。

证明本节最后一个定理需要以下性质。它可由命题1和定义 7 直接推出。

推论 2. 设 $\mathcal{K}$ 是一个知识库，q 是一个合取查询，$\mathcal{U}$ 是 $\mathcal{K}$ 的一个通用解，且 $\mathbf{T} \in q^{\mathcal{U}}$。那么，对于每个 $\mathcal{I} \in Sol(\mathcal{K})$，都存在一个元组 $\mathbf{T}' \in q^{\mathcal{I}}$，使得 $\mathbf{T} \downarrow \leq \mathbf{T}'$。

证明。 显然，$\mathbf{T}' \downarrow \leq \mathbf{T}'$。由于 $\mathcal{U}$ 是通用的，因此存在一个从 $\mathcal{U}$ 到 $\mathcal{I}$ 的同态 h。由命题1可知，存在 $\mathbf{T}' \in q^{\mathcal{I}}$，使得 $h(\mathbf{T}) \leq \mathbf{T}'$。由定义 7（同态）可知，$\mathbf{T} \downarrow \leq h(\mathbf{T})$，因此 $\mathbf{T} \downarrow \leq \mathbf{T}'$。 □

现在我们可以给出本节的主要结果，它指出，在我们的框架中，可以利用通用解计算合取查询的确定答案。

定理 1. 设 q 是一个合取查询，$\mathcal{K}$ 是一个知识库，$\mathcal{U}$ 是 $\mathcal{K}$ 的一个通用解。那么 $cert(q, \mathcal{K}) = q^{\mathcal{U}}{\downarrow_\rho}$。

证明。我们首先证明 $q^{\mathcal{U}}{\downarrow_\rho} \subseteq$cert(q $\in q^{\mathcal{U}}$ $q^{\mathcal{U}}{\downarrow_\rho}$ 2 $\in$,$\mathcal{K}$)。

设 T，这显然意味着 T。根据推论，对于每个 $\mathcal{I} \in$Sol($\mathcal{K}$)，都存在一个元组 $\mathbf{T}' \in q^{\mathcal{I}}$，使得 T↓ ≤T′。由于 $\mathbf{T} \in q^{\mathcal{U}}{\downarrow_\rho}$，它没有空值，因此 T↓ = T。于是，T 满足定义 9（确定答案）的条件（i）。现在反设 T 不满足定义 9 的条件（ii）。这意味着存在一个无空值元组 T′，满足定义 9 的条件（i），且 $\mathbf{T} < \mathbf{T}'$。由于 T′ 满足定义 9 的条件（i），必定存在一个元组 $\mathbf{U} \in q^{\mathcal{U}}$，使得 $\mathbf{T}' \leq \mathbf{U}$，因此显然 $\mathbf{T} < \mathbf{U}$。我们可以将 $\downarrow$ 算子应用于 T 和 U，得到 $\mathbf{T}{\downarrow} < \mathbf{U}{\downarrow}$；并且如前所述，$\mathbf{T} = \mathbf{T}{\downarrow}$，因此有 $\mathbf{T} < \mathbf{U}{\downarrow}$，这与假设矛盾。事实上，$\mathbf{T} \in q^{\mathcal{U}}{\downarrow_\rho}$（根据假设），而 $\mathbf{U} \downarrow \in q^{\mathcal{U}}{\downarrow}$，因此 U 不可能支配 T（参见 $\downarrow_\rho$ 算子的定义中的步骤 3）。

现在我们证明 $cert(q, \mathcal{K}) \subseteq q^{\mathcal{U}}{\downarrow_\rho}$。令 $\mathbf{T} \in cert(q, \mathcal{K})$。根据定义 9 的条件 ($i$)，存在一个元组 $\mathbf{U} \in q^{\mathcal{U}}$，使得 $\mathbf{T} \leq \mathbf{U}$，这意味着 $\mathbf{T} \leq \mathbf{U}{\downarrow}$（因为 $\mathbf{T}$ 是无空值的）。根据推论 2，对于每个$\mathcal{I} \in Sol(\mathcal{K})$，存在一个元组 $\mathbf{T}' \in q^{\mathcal{I}}$，使得 $\mathbf{U}{\downarrow} \leq \mathbf{T}'$。这一事实意味着 $\mathbf{U}{\downarrow}$ 是一个满足以下条件的无空值元组定义 9 的条件 (i)。因此，如果$\mathbf{T} < \mathbf{U}{\downarrow}$，那么 $\mathbf{T}$不是一个确定答案，这与假设矛盾。因此，必有 $\mathbf{T} = \mathbf{U}{\downarrow}$。这表明 $\mathbf{T} \in q^{\mathcal{U}}{\downarrow}$。反设 $\mathbf{T} \notin q^{\mathcal{U}}{\downarrow_\rho}$。根据 $\downarrow_\rho$ 算子的定义，上述假设意味着存在一个元组 $\mathbf{X} \in q^{\mathcal{U}}{\downarrow_\rho}$，使得 $\mathbf{T} < \mathbf{X}$；又因为 $q^{\mathcal{U}}{\downarrow_\rho} \subseteq cert(q, \mathcal{K})$，如本证明第一部分所示，$\mathbf{X}$满足定义 9 的条件 ($i$)，进而意味着 $\mathbf{T} \notin cert(q, \mathcal{K})$，这就产生了矛盾。

□

示例 10。 如例 6 中所见，$\mathcal{I}'$ 是 $\mathcal{K}$的一个通用解。因此，对于查询 $q_1(x)$：$\exists z CEO(z, x)$，我们有 $cert(q_1, \mathcal{K}) = q_1^{\mathcal{I}'}{\downarrow_\rho} = \{\langle[\mathbf{Doe}_1, \mathbf{Doe}_2]\rangle\}$。 □

5 计算通用模型

在本节中，我们将著名的受限追赶概念（Beeri 和 Vardi 1984；Johnson 和 Klug 1984；Fagin 等人 2005；Calvanese 等人 2007）改编到我们的框架中。有趣的是，我们定义的追赶过程永远不会产生失败；这不同于例如标准数据交换场景中的受限追赶过程，在那里应用相等生成依赖可能会导致失败，因为必须将两个不同的常量设为相等。相反，在我们的框架中，当实体相等生成依赖要求将两个不同的实体等价类合并时，我们会将它们合并为一个更大的等价类。同样，当值等值生成依赖要求将两个不同的值集合并时，我们会取这两个集合的并集，并相应地修改实例。然而，追赶过程可能包含无穷多个步骤，每一步都会生成一个新实例。因此，定义将这一（可能无限的）过程应用于知识库 $\mathcal{K}$ 后的结果时需要谨慎，以便得到一个可用于（至少原则上）查询回答的实例。我们将这样的实例称为知识库追赶过程的结果 $\mathcal{K}$，并区分追赶终止与未终止的情况。前一种情况下，$\mathcal{K}$的追赶结果就是追赶过程最后一步生成的实例，我们证明该实例是 $\mathcal{K}$的一个通用解。后一种情况下，我们指出，文献中针对元组生成依赖和相等生成依赖下无限标准追赶序列提出的既有方法，无法顺利改编到我们的框架中；至于如何定义这种情况下的追赶结果，使其成为一个通用解，则留待后续研究。

我们首先介绍知识库的基础实例 这一概念。给定知识库 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$，我们定义集合

$$\mathcal{I}^{\mathbf{D}} = \{P(\{c_1\}, \ldots, \{c_n\}) \mid P(c_1, \ldots, c_n) \in \mathbf{D}\}.$$

$\mathcal{I}^{\mathbf{D}}$ 是相对于集合 $\mathcal{S} = \mathcal{S}_E \cap sig(\mathcal{K})$ 上恒等关系 id 而言，$\mathcal{K}$ 的一个实例。我们称 $\mathcal{I}^{\mathbf{D}}$ 为 $\mathcal{K}$的基础实例 。请注意，$\mathcal{K}$ 的基础实例也是所有以$\mathbf{D}$ 为数据库的知识库的基础实例，并且是 $(\emptyset, \mathbf{D})$ 的一个解。例如，请注意，例 3 中的实例 $\mathcal{I}$ 是例 1 和例2 中定义的知识库 $\mathcal{K}$ 的基础实例。

接下来我们定义三种追赶步骤，分别用于元组生成依赖、实体-egd 和值 egd。

定义 11. 令 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 为一个知识库，$\mathcal{I}_1$ 是相对于 $\mathcal{K}$ 的一个实例，其中等价关系 $\sim^1$ 定义在 $under_E(\mathcal{I}_1)$ 上。

• (元组生成依赖) 令 r 为形式如 (1) 的元组生成依赖。不失一般性，我们假设 ψ(x, y) 中的所有原子公式均具有如下形式： $P(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$，其中 $x_1, \ldots, x_k$ 属于 x， $y_1, \ldots, y_\ell$ 属于 y；我们分别用 ye $= y_e^1, \ldots y_e^h$ 和 yv $= y_v^1, \ldots, y_v^j$ 表示 y 中的实体变量和值变量。令 μ 为从 ϕ(x) 到 $\mathcal{I}_1$ 的一个赋值，使得不存在与 ψ(x, y) 相容、且将 μ 扩展到 $\mathcal{I}_1$ 的元组生成依赖扩展。我们称 r 可应用于带有 μ 的 $\mathcal{I}_1$ （或者称 μ 在 $\mathcal{I}_1$ 中触发 r），并通过以下过程构造 $\mathcal{I}_2$：

> **设** $\{f_e^1, \ldots, f_e^h\} \subseteq \mathcal{S}_{EN}$和 $\{f_v^1, \ldots, f_v^j\} \subseteq \mathcal{S}_{VN}$是两个由新空值组成的集合（即未出现在 $\mathcal{I}_1$中），且二者互不相交**置入** $\mathcal{I}_2 := \mathcal{I}_1$**对于每个**原子 $P(x_1, \ldots, x_k, y_1, \ldots, y_\ell)$，属于 ψ(x， y) **执行**$\mathcal{I}_2 := \mathcal{I}_2 \cup \{P(T_1, \ldots, T_k, U_1, \ldots, U_\ell)\}$

其中，
– f 或者$\leq i \leq k$，若 x_i 是实体变量，则有 $T_i = \mu(x_i)$；若 x_i 是值变量且 $\{S_1^{x_i}, \ldots, S_m^{x_i}\} = SetVar(x_i, \tau(\phi(\mathbf{x})))$，则有 $T_i = \bigcap_{p=1}^m \mu(S_p^{x_i})$；–对于 $1 \leq i \leq \ell$，若 $y_i = y_e^s$且$1 \leq s \leq h$，则有 $U_i = \{f_e^s\}$；若 $y_i = y_v^q$且 $1 \leq q \leq j$，则有 $U_i = \{f_v^q\}$。（因此，每个单元素集合 $\{f_e^s\}$ 都是一个新的等价类。）

• (实体-egd) 设 r 是形式为 (2) 的实体-egd，且 μ是从 ϕ(x) 到 $\mathcal{I}_1$ 的一个赋值，使得 $\mu(y) \neq \mu(z)$。我们称 r 对 $\mathcal{I}_1$ 在 μ 下是适用的（或者称 μ 在 $\mathcal{I}_1$ 中触发了 r ），并通过在 $\mathcal{I}_1$ 中将 $\mu(y)$ 和 $\mu(z)$ 的所有出现替换为 $\mu(y) \cup \mu(z)$，从 $\mathcal{I}_1$ 构造出 $\mathcal{I}_2$。

（因此，我们将两个等价类合并为一个新的等价类。）
• (值等值生成依赖) 设 r 是形式为 (2) 的值等值生成依赖。令 $\{S_1^y, \ldots, S_m^y\}$ =SetVar($y, \tau(\phi(\mathbf{x}))$)、 $\{S_1^z, \ldots, S_k^z\}$ =SetVar($z, \tau(\phi(\mathbf{x}))$)、 $1 \leq i \leq m$、 $1 \leq j \leq k$和 μ 为从 ϕ(x) 到 $\mathcal{I}_1$ 的一个赋值，使得 $\mu(S_i^y) \neq \mu(S_j^z)$。我们称 r 在 μ 下对 $\mathcal{I}_1$ 适用（或者称 μ 在 $\mathcal{I}_1$ 中触发了 r），并通过在图像 $\mu(\phi(\mathbf{x}))$ 中，将每个集合 $\mu(S_1^y) \ldots$、 $\mu(S_m^y)$ $\mu(S_1^z) \ldots$ 以及 $\mu(S_k^z)$ 替换为 $\mu(S_1^y) \cup \ldots \cup \mu(S_m^y) \cup \mu(S_1^z) \cup \ldots \cup \mu(S_k^z)$，从 $\mathcal{I}_1$ 构造出 $\mathcal{I}_2$。
如果 r 是一个可与 μ一起应用于 $\mathcal{I}_1$ 的元组生成依赖或相等生成依赖，我们称 $\mathcal{I}_2$ 为将 r 与 μ 一起应用于 $\mathcal{I}_1$ 的结果，并记作$\mathcal{I}_1 \xrightarrow{r, \mu} \mathcal{I}_2$。我们称 $\mathcal{I}_1 \xrightarrow{r, \mu} \mathcal{I}_2$ 为一个追赶步骤。

对于实体相等生成依赖步骤和值相等生成依赖步骤，追赶过程都会通过替换$\mathcal{I}_1$中的某些事实来构造 $\mathcal{I}_2$。然而，对于实体等值生成依赖，替换是“全局的”（即该步骤中合并的两个等价类在 $\mathcal{I}_1$中的所有位置都替换为它们的并集）；而对于值等值生成依赖，替换则是“局部的”，也就是说，该步骤中合并的两个集合仅在出现在像 $\mu(\phi(\mathbf{x}))$中的事实里才替换为它们的并集，而这个像是 $\mathcal{I}_1$ 的子集。

例11。再次考虑例 1 和例2中的知识库 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$。例3中的解释 $\mathcal{I}$ 是 $\mathcal{K}$的基础实例$\mathcal{I}^{\mathbf{D}}$。下文展示从实例 $\mathcal{I} = \mathcal{I}^{\mathbf{D}}$开始应用定义11中的规则。

元组生成依赖应用： $\mathcal{I}^{\mathbf{D}} \xrightarrow{r_4, \mu_0} \mathcal{I}_1$，其中 μ_0 满足μ_0(体(r_4)) $= \{d_3\}$。所得实例 $\mathcal{I}_1$ 包含事实 d_1-d_6 （见例 3 ），以及以下事实：

$$(d_7) Emp([\mathbf{Doe}_3], [\mathbf{e}_1^\perp]),\ (d_8) CEO([\mathbf{e}_1^\perp], [\mathbf{e}_2^\perp]).$$

实体-egd 应用： $\mathcal{I}_1 \xrightarrow{r_1, \mu_1} \mathcal{I}_2$，其中赋值 μ_1 满足 μ_1(体(r_1)) $=\{d_1, d_2, JaccSim(\{\text{J. Doe}\}, \{\text{John Doe}\}, \{0.6\})\}$。

所得实例 $\mathcal{I}_2$ 包含事实 d_3, d_5, d_7, d_8，以及以下事实：

> (d_1) →(d_9) CI([$\mathbf{Doe}_1, \mathbf{Doe}_2$], {**J. Doe**}, {358}) (d_2) →(d_{10}) CI([$\mathbf{Doe}_1, \mathbf{Doe}_2$], {**John Doe**}, {635}) (d_4) →(d_{11}) Emp([$\mathbf{Doe}_1, \mathbf{Doe}_2$], [**雅虎**]) ($d_6$) →($d_{12}$) CEO([**雅虎**], [$\mathbf{Doe}_1, \mathbf{Doe}_2$])。

值等值生成依赖应用： $\mathcal{I}_2 \xrightarrow{r_2, \mu_2} \mathcal{I}_3$，其中 μ_2 满足条件μ_2(体(r_2)) $= \{d_9, d_{10}\}$。所得实例 $\mathcal{I}_3$ 包含事实 d_3, d_{11}, d_5, d_{12}, d_7, d_8, 以及以下事实：(d_9) →(d_{13}) CI([$\text{Doe}_1, \text{Doe}_2$], {J. Doe,John Doe}, {358}) (d_{10}) →(d_{14}) CI([$\text{Doe}_1, \text{Doe}_2$], {J. Doe,John Doe}, {635})。现在我们定义追赶序列的概念。

定义12。令 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 为一个知识库。 $\mathcal{K}$ 的追赶序列 是一个（有限或无限）序列 $\sigma = \mathcal{I}_0, \mathcal{I}_1, \mathcal{I}_2 \ldots$，满足 $\mathcal{I}_0 = \mathcal{I}^{\mathbf{D}}$，且对序列 σ 中每一对相邻项 ($\mathcal{I}_i, \mathcal{I}_{i+1}$)， $\mathcal{I}_i \xrightarrow{r_i, \mu_i} \mathcal{I}_{i+1}$ 是一个追赶步骤。

在附录中，我们为示例中的知识库 $\mathcal{K}$ 给出了一个完整的追赶序列；该序列的前三个实例已在例 11 中描述。

现在我们介绍追赶序列中实例的一些基本性质。

命题 3。令 $\mathcal{K}$ 为一个知识库。
• 若 $\mathcal{I}$ 是关于 在 $_E(\mathcal{I})$ 上的等价关系 $\sim$、针对 $\mathcal{K}$ 的一个实例，且 $\mathcal{I}'$ 满足 $\mathcal{I} \xrightarrow{r, \mu} \mathcal{I}'$，则存在一个等价关系 $\sim'$，使得 $\mathcal{I}'$ 是关于 $\sim'$ 在 在 $_E(\mathcal{I}')$ 上、针对 $\mathcal{K}$ 的一个实例。

• 知识库 $\mathcal{K}$ =($\mathcal{T}$,**D**) 的追赶序列中出现的每个 $\mathcal{I}_i$ 都是 $\mathcal{K}$ 的实例。

证明。我们通过考察规则 r 的三种可能情形，以及定义 11 中相应的追逐步骤，来证明第一条中的论断。

tgd 追赶步骤的情形：设 r 是形式为 (1) 的元组生成依赖。设 $\mathcal{S}$ = 在 $_E(\mathcal{I})$ 下为 $\cup \mathbf{N}^{f_e}$，其中 $\mathbf{N}^{f_e} = \{f_e^1, \ldots, f_e^h\}$，即 $\mathbf{N}^{f_e}$ 是定义 11 中 tgd 情形所用的新鲜空值符号集合。设 $\sim' = \sim \cup \{\langle f, f\rangle \mid f \in \mathbf{N}^{f_e}\}$（注意，$\mathbf{N}^{f_e}$ 甚至可能为空，这意味着 $\sim' = \sim$）。容易看出， $\sim'$ 是 $\mathcal{S}$ 上的等价关系，且在 $_E(\mathcal{I})$ 下有 $/\sim \subseteq \mathcal{S}/\sim'$。于是，根据定义 11，对于每个 $P(T_1, \ldots, T_n) \in \mathcal{I}'$ 以及每个满足 type(P, i) =e 的 $1 \leq i \leq n$，要么 $T_i \in$ 在 $_E(\mathcal{I})$ 下满足 $/\sim$，因此 $T_i \in \mathcal{S}/\sim'$；要么 $T_i = \{f\}$，其中 $f \in \mathbf{N}^{f_e}$。根据 $\sim'$ 的定义，显然有 $\{f\} \in \mathcal{S}/\sim'$。另请注意，在 $_E(\mathcal{I}')$ 下有 $\mathcal{S} =$。此外，对于每个满足 type(P, i) =v 的 $1 \leq i \leq n$，要么 $T_i \subseteq$ 在 $_V(\mathcal{I})$ 下成立，要么 $T_i = \{f'\}$，其中 $f' \in \{f_v^1, \ldots, f_v^j\} \subseteq \mathcal{S}_{VN}$。因此，$T_i$ 是 $(\mathcal{S}_V \cap \text{sig}(\mathcal{K})) \cup \mathcal{S}_{VN}$ 的一个非空子集。以上事实表明， $\mathcal{I}'$ 是相对于 $\sim'$、在 $\mathcal{S} =$ 上且处于 $_E(\mathcal{I}')$ 下的 $\mathcal{K}$ 的一个实例。

实体 *EGD* 追赶步骤的情形：设 r 是形式为 (2) 的实体-egd。在此情形下，我们有 在 $_E(\mathcal{I}')$ =在 $_E(\mathcal{I})$，且 $\sim' = \sim \cup \sim^e$，其中 $\sim^e = \{\langle f, g\rangle \mid f, g \in \mu(y) \cup \mu(z)\}$。显然， $\sim^e$ 是一个等价关系

在 $\mu(y) \cup \mu(z)$ 上。因此， $\sim'$ 是一个等价关系与定义在 $\text{under}_E(\mathcal{I})$ 上的等价关系的并集，以及定义在$\mu(y) \cup \mu(z)$ 上的等价关系的并集。因此， $\sim'$ 是在 $\text{under}_E(\mathcal{I}) \cup \mu(y) \cup \mu(z)$ 上的自反且对称关系；并且由于$\mu(y) \cup \mu(z) \subseteq \text{under}_E(\mathcal{I})$，因此 $\sim'$ 是在 $\text{under}_E(\mathcal{I}) = \text{under}_E(\mathcal{I}')$ 上的自反且对称关系。接下来我们需要证明 $\sim'$ 也是传递的。为此，我们考虑所有可能的情形。设 a、b、c是 $\mathcal{S}$中的元素。可能出现以下几种情况:

•(a, b) $\in\sim$且(b, c) $\in\sim$，但由于 $\sim$是 $_E(\mathcal{I})$ 上的等价关系，因此(a, c) $\in\sim$，并且由于 $\sim\subseteq\sim'$，所以(a, c) $\in\sim'$。

•(a, b) $\in\sim^e$且(b, c) $\in\sim^e$，但由于 $\sim^e$是 $\mu(y) \cup \mu(z)$ 上的等价关系，因此(a, c) $\in\sim^e$，并且由于$\sim^e\subseteq\sim'$，所以(a, c) $\in\sim'$。

•(a, b) $\in\sim$且(b, c) $\in\sim^e$。由于(b, c) $\in\sim^e$，根据追赶步骤的定义，b, $c \in \mu(y) \cup \mu(z)$，并且 $[b]_\sim \subseteq \mu(y) \cup \mu(z)$。由于($a$, b) $\in\sim$，因此 $a \in [b]_\sim$也成立，从而 $a \in \mu(y) \cup \mu(z)$。这意味着(a, b) $\in\sim^e$ 也成立，这与前一种情况一致（我们已经证明该情况下传递性成立）。

然后，我们可以证明， $\mathcal{I}'$ 的每个事实 $P(T_1, \ldots\ldots, T_n)$ 都满足定义1中的条件，论证方式与 tgd 情形证明最后部分所用的论证类似。唯一的区别是，如果某个 T_i 也不出现在 $\mathcal{I}$ 的任何事实中，那么它必定是 $T_i = \mu(z) \cup \mu(y)$，这确实是 在 $_E(\mathcal{I}')/\sim'$ 下的一个等价类。

值 *EGD* 追逐步骤的情形：设 r 是形式为 (2) 的值等值生成依赖。在这种情况下，在 $_E(\mathcal{I}')$ =在 $_E(\mathcal{I})$ 且 $\sim'=\sim$。证明的其余部分与前述情形相同。不过我们注意到，在此情形下，每个未出现在 $\mathcal{I}$ 的事实中的 T_i 都必须具有 $\mu(S_1^y) \cup \ldots \cup \mu(S_m^y) \cup \mu(S_1^z) \cup \ldots \cup \mu(S_k^z)$ 的形式，这显然是 $(\mathcal{S}_V \cap$ 签名 $(\mathcal{K})) \cup \mathcal{S}_{VN}$ 的一个非空子集。

第二个要点中的性质很容易由第一个要点中的性质以及以下事实推出：根据构造，在每个追赶序列中， $\mathcal{I}_0 = \mathcal{I}^{\mathbf{D}}$ 和 $\mathcal{I}^{\mathbf{D}}$ 都是 $\mathcal{K}$ 的实例。

□

命题 4. 设 $\mathcal{K}$ 是一个知识库， σ 是 $\mathcal{K}$的一个追赶序列， $\mathcal{I}_i$ 和 $\mathcal{I}_j$是$\mathcal{K}$ 的两个实例，分别对应等价关系 $\sim^i$ 和 $\sim^j$，其中 $i \leq j$、 $\mathcal{I}_i$ 和 $\mathcal{I}_j$ 均属于 σ。则有:

• $\sim^i\subseteq\sim^j$且在$(\mathcal{I}_i) \subseteq$下，在$(\mathcal{I}_j)$;
• 若 $P(\mathbf{T})$是 $\mathcal{I}_i$中的一个事实，则存在 $\mathcal{I}_j$中的一个事实 $P(\mathbf{T}')$，使得 $P(\mathbf{T}) \leq P(\mathbf{T}')$。

证明。 我们回顾一下， $\mathcal{K} = (\mathcal{T}, \mathbf{D})$，其中 $\mathcal{T}$是$\mathcal{K}$ 的TBox，而$\mathbf{D}$ 是 $\mathcal{K}$的数据库。这些性质可以通过归纳法轻松证明。基例。 设 $i = 0$和 $j = 1$，即考虑 $\mathcal{I}_0$ 和 $\mathcal{I}_1$，其中 $\mathcal{I}_0$ 是 $\mathcal{K}$ 的基础实例$\mathcal{I}^{\mathbf{D}}$ ，而 $\mathcal{I}_1$ 是对 $\mathcal{I}^{\mathbf{D}}$应用一个追赶步骤后生成的实例。包含关系 $\sim^0\subseteq\sim^1$ 和在 $(\mathcal{I}_0) \subseteq$在$(\mathcal{I}_1)$ 下，可以使用命题 3 的证明中采用的相同构造来证明，其中 $\mathcal{I} = \mathcal{I}_0$ 和 $\mathcal{I}' = \mathcal{I}_1$。特别地，对于 r 是元组生成依赖的情况，根据构造可知 $\sim_0=\sim\subseteq\sim'=\sim_1$。此外，在 $_V(\mathcal{I}')$ =在 $_V(\mathcal{I}) \cup \{f_v^1, \ldots, f_v^j\}$，因此在 $(\mathcal{I}) \subseteq$在$(\mathcal{I}')$（即，在 $(\mathcal{I}_0) \subseteq$在 $(\mathcal{I}_1)$。 r 是值等值生成依赖或实体-egd的情况则更加简单。

还需证明，对于每个事实 $P(\mathbf{T})$，都存在 $P(\mathbf{T}')$，使得$\mathbf{T} \leq \mathbf{T}'$。考虑三种可能的追赶步骤。(元组生成依赖)：此步骤不会修改 $\mathcal{I}_0$中的事实，因此$\mathbf{T} = \mathbf{T}'$。(实体-egd)：active$(\mathcal{I}_0)$中的每个等价类 E，即 $\mathcal{I}_0$的活动域，都保持不变，或者被替换为一个满足 $E \subseteq E'$的类 E' 。此外，$\text{active}_V(\mathcal{I}_0) = \text{active}_V(\mathcal{I}_1)$。这些事实很容易表明本情形下结论成立。(值等值生成依赖)：论证与前一种情形类似，也就是说，$\text{active}_V($ $\mathcal{I}_0$)中的每个值集 V都保持不变，或者被替换为一个满足 $V \subseteq V'$的值集 V' 。此外，$\text{active}_E(\mathcal{I}_0)$ =$\text{active}_E(\mathcal{I}_1)$。

归纳情形。 我们假设对于某个整数 ℓ，命题成立，即性质 (*i*)、(*ii*) 和 (*iii*) 对每个满足 $i \leq j$ 的 $i, j \leq \ell$ 都成立，并证明它们对每个满足 $i \leq j$ 的 $i, j \leq \ell + 1$ 也成立。需要证明的唯一情形是 $j = \ell + 1$ 的情形，因为其他情形已由归纳假设成立。当然，如果 $i = \ell + 1$，则该条件显然成立。对于 $i = \ell$，这三个性质可以完全按照基例中的证明方法证明。于是，所有满足 $i < \ell$ 的情形都可由前一种情形和归纳假设推出。 □

命题 4 意味着，对于每个 $i \leq j$ 以及 在 $_E(\mathcal{I}_i)/\sim^i$ 中的每个等价类 E，都存在 在 $_E(\mathcal{I}_j)/\sim^j$ 中的 E'，使得 $E \subseteq E'$；特别地，对于每个实体 $e \in \mathcal{S}_E \cap sig(\mathcal{K})$，我们有 $[e]_{\sim^i} \subseteq [e]_{\sim^j}$。

当知识库的追赶序列是无限的时，序列中某个实例 $\mathcal{I}_i$ 上可能存在适用的规则，这些规则永远不会被应用，尽管它们在后续追赶步骤中仍然适用。不过，我们始终可以为规则的应用建立一个合适的顺序，从而避免上述情况。具有这一性质的追赶序列称为公平的。下文中，我们假设所考虑的所有追赶序列都是公平的。

现在，我们来考察知识库 $\mathcal{K}$进行追赶算法时可能出现的结果。首先，给出有限追赶序列的追赶结果的定义。

定义 13。令 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 为一个知识库。
•对 $\mathcal{K}$ 的有限追赶是一个有限追赶序列 $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_m$，其中不存在规则 r属于 $\mathcal{T}$且赋值 μ，使得 r 能够以 μ应用于 $\mathcal{I}_m$。

• 我们称 $\mathcal{I}_m$ 为 $\mathcal{K}$的有限追赶的结果 σ ，并将其记为追赶算法$(\mathcal{K}, \sigma)$。

命题 3 表明，追赶算法$(\mathcal{K}, \sigma)$ 是 $\mathcal{K}$ 的一个实例。实际上，它还是 $\mathcal{K}$ 的一个解。

引理 1。 设 $\mathcal{K}$ 是一个知识库，且 $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots\ldots, \mathcal{I}_m$ 是 $\mathcal{K}$ 的一个有限追赶。则 追赶算法$(\mathcal{K}, \sigma)$ 是 $\mathcal{K}$ 的一个解。

证明。 我们需要证明，追赶算法$(\mathcal{K}, \sigma)$ 满足 $\mathbf{D}$ 中的所有事实，以及 $\mathcal{T}$ 中的所有元组生成依赖、实体等值生成依赖和值等值生成依赖。我们分别说明每一点:

(i) 追赶算法$(\mathcal{K}, \sigma)$ 满足$\mathbf{D}$ 中的所有事实这一性质， $\mathcal{I}_0 = \mathcal{I}^{\mathbf{D}}$ 可推出这一点，而命题 4 也给出了这一结论。事实上，根据基础实例的定义，$\mathcal{I}^{\mathbf{D}}$ 满足$\mathbf{D}$中的所有事实，这意味着，对于其中的每个基原子 $P(c_1, \ldots, c_n)$,

D 中都存在一个事实 $P(T_1, \ldots, T_n)$，使得 $\mathcal{I}_0$ 满足 $c_i \in T_i$ 对于 $1 \leq i \leq n$。根据命题 4 可知，在 $\mathcal{I}_m$中存在一个元组 $\langle T_1', \ldots, T_n' \rangle$，使得 $T_1 \subseteq T_1', \ldots, T_n \subseteq T_n'$，因此 $P(c_1, \ldots, c_n)$ 在 $\mathcal{I}_m$ =追赶算法 $(\mathcal{K}, \sigma)$ 中也成立。

(ii) 为证明 chase($\mathcal{K}, \sigma$) 满足 $\mathcal{T}$ 中的所有元组生成依赖，我们反设存在一个不满足的元组生成依赖 $d = \forall \mathbf{x}(\phi(\mathbf{x}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y}))$。这意味着存在一个从 $\phi(\mathbf{x})$ 到 chase($\mathcal{K}, \sigma$) 的赋值 μ，使得不存在从 μ 到 chase($\mathcal{K}, \sigma$) 的、与 $\psi(\mathbf{x},\mathbf{y})$ 兼容的元组生成依赖扩展 μ'。但这样一来，凭借赋值 μ，d 就可以应用于 chase($\mathcal{K}, \sigma$)，这与 "σ 是有限追赶 " 的假设矛盾。

(iii) 追赶算法($\mathcal{K}$，σ) 满足所有实体等值生成依赖和值等值生成依赖这一事实，可以按照 (ii) 中证明 $\mathcal{T}$ 中的元组生成依赖得到满足的方法来证明。

□

下面的引理用于证明：如果 σ 是有限追赶，那么 追赶算法($\mathcal{K}$，σ) 不仅是 $\mathcal{K}$ 的一个解，而且还是 $\mathcal{K}$ 的一个通用解。一个类似的结果，即三角引理，曾由 Fagin 等人于 2005年在数据交换的背景下使用。

引理 2. 设 $\mathcal{I}_1 \xrightarrow{r, \mu} \mathcal{I}_2$ 是一个追赶步骤。设 $\mathcal{I}$ 是 $\mathcal{K}$ 的一个实例，使得 $\mathcal{I}$ 满足 r，且存在一个同态 $h_1 : \mathcal{I}_1 \rightarrow \mathcal{I}$。那么，存在一个同态$h_2 : \mathcal{I}_2 \rightarrow \mathcal{I}$，使得 h_2 扩展了 h_1。

证明。 证明思路沿用了 Fagin 等人（2005年）中引理 3.4 的证明。然而，由于我们的结构基于等价类和值集，证明中的各个步骤更为复杂。具体而言，与 Fagin 等人（2005年）不同，在我们的框架中，将赋值与同态复合并不能直接得到新的赋值。尽管如此，仍可基于这种复合构造出一个赋值。

我们通过考察规则 d 的三种可能情况，以及定义 11 中相应的追赶步骤，来证明该命题。

tgd 追赶步骤的情况：设 r 是形如 (1) 的元组生成依赖。为便于说明，我们假设 $\phi(\mathbf{x}) = P_1(x_1, x_2) \wedge P_2(x_1, x_2)$，且 $\psi(\mathbf{x}, \mathbf{y}) = P_3(x_1, x_2, y_1, y_2) \wedge P_4(x_2, y_1, y_2)$，其中 x_1 和 y_2 是实体变量，x_2和 y_1 是值变量。一般元组生成依赖的情况也可以用类似的方式处理。若将算子 τ 同时应用于元组生成依赖的体和头，就得到以下公式：

$\tau(\phi(\mathbf{x})) = P_1(x_1, S_1^{x_2}) \wedge P_2(x_1, S_2^{x_2})\tau(\psi(\mathbf{x}, \mathbf{y})) = P_3(x_1, R_1^{x_2}, R_1^{y_1}, y_2) \wedge P_4(R_2^{x_2}, R_2^{y_1}, y_2)$ 我们首先构造一个从 $\phi(\mathbf{x})$ 到 $\mathcal{I}$ 的赋值 $\hat{}\mu$。根据追赶步骤的定义，μ 是一个从 $\phi(\mathbf{x})$ 到 $\mathcal{I}_1$ 的赋值。因此，存在一个等价类 $E_1 \in$ 活动域$_E(\mathcal{I}_1)$ 以及两个值集 $V_1, V_2 \in$活动域$_V(\mathcal{I}_1)$，使得$V_1 \cap V_2 \neq \emptyset$, $\mu(x_1) = E_1$, $\mu(S_1^{x_2}) = V_1$, $\mu(S_2^{x_2}) = V_2$,$P_1(E_1, V_1)$ 和 $P_2(E_1, V_2)$ 是 $\mathcal{I}_1$ 中的事实。由于 h_1 是从 $\mathcal{I}_1$ 到 $\mathcal{I}$ 的同态，存在事实 $P_1(E_1', V_1')$,$P_2(E_2', V_2')$ 属于 $\mathcal{I}$，使得 $h_1(P_1(E_1, V_1)) \leq P_1(E_1', V_1')$ 且 $h_1(P_2(E_1, V_2)) \leq P_2(E_1', V_2')$，也就是说，$h_1(E_1) \subseteq E_1'$,$h_1(E_1) \subseteq E_2'$, $h_1(V_1) \subseteq V_1'$, $h_1(V_2) \subseteq V_2'$。由于 $V_1 \cap V_2 \neq \emptyset$，那么 $V_1' \cap V2' \neq \emptyset$ 也成立。此外，由于 E_1' 和 E_2' 是包含同一非空子集的等价类，它们必定是同一个等价类，即 $E_1' = E_2'$。接下来，我们将映射 $\hat{}\mu$ 定义如下：$\hat{}\mu(x_1) = E_1'$, $\hat{}\mu(S_1^{x_2}) = V_1'$，以及 $\hat{}\mu(S_2^{x_2}) = V_2'$。显然，$\hat{}\mu$ 满足定义 3 中的条件，因此它是一个从 $\phi(\mathbf{x})$ 到 $\mathcal{I}$ 的赋值。

由于 $\mathcal{I}$ 满足 r，因此存在一个赋值 μ'，它是 $\hat{}\mu$ 到 $\mathcal{I}$ 的 $\psi(\mathbf{x}, \mathbf{y})$- 相容扩展。根据定义，我们有：

- $\mu'(x_1) = \hat{}\mu(x_1) = E_1'$
- $\mu'(R_1^{x_2}) = V_3'$，$\mu'(R_2^{x_2}) = V_4'$，其中 V_3'，$V_4' \in$活动域$_V(\mathcal{I})$ 满足以下条件：$V_1' \cap V_2' \subseteq V_3' \cap V_4'$,
- $\mu'(R_1^{y_1}) = V_5'$, $\mu'(R_2^{y_1}) = V_6'$，其中 V_5'，$V_6' \in$活动域$_V(\mathcal{I})$ 满足 $V_5' \cap V_6' \neq \emptyset$,
- $\mu'(y_2) = E_2'$，其中 $E_2' \in$ 活动域$_E(\mathcal{I})$,
- $\mathcal{I}$包含事实 $P_3(E_1', V_3', V_5', E_2')$ 和$P_4(V_4', V_6', E_2')$。

然后，根据 tgd 追赶步骤的定义可知，还存在一个从 $\psi(\mathbf{x}, \mathbf{y})$ 到 $\mathcal{I}_2$ 的赋值 μ_2，定义如下

- $\mu_2(x_1) = \mu(x_1) = E_1$
- $\mu_2(R_1^{x_2}) = \mu_2(R_2^{x_2}) = V_1 \cap V_2$
- $\mu_2(R_1^{y_1}) = \mu_2(R_2^{y_1}) = \{f_v^{y_1}\}$
- $\mu_2(y_2) = [f_e^{y_2}]$

其中，$f_v^{y_1}$ and $f_e^{y_2}$ are 分别是由追赶步骤引入的值空值和实体空值。我们指出，μ_2 是从 μ 到 $\mathcal{I}_2$的一个 $\psi(\mathbf{x}, \mathbf{y})$-兼容扩展，并且在 $(\mathcal{I}_2)$ =在 $(\mathcal{I}_1) \cup \{f_v^{y_1}, f_e^{y_2}\}$。此外，$\mathcal{I}_2 = \mathcal{I}_1 \cup \{P_3(E_1, V_1 \cap V_2, \{f_v^{y_1}\}, [f_e^{y_2}])\ P_4(V_1 \cap V_2, \{f_v^{y_1}\}, [f_e^{y_2}])\}$。

现在我们构造一个映射 h_2，并证明它是从 $\mathcal{I}_2$ 到 $\mathcal{I}$的同态。我们定义 h_2 如下:

- $h_2(o) = h_1(o)$，对每个 $o \in$在$(\mathcal{I}_1)$;
- $h_2(f_v^{y_1}) = v$，其中 v是 $V_5' \cap V_6'$中的任意元素;
- $h_2(f_e^{y_2}) = e$，其中 e 是 E_2'中的任意元素。

注意，如前所述，$V_5' \cap V_6' \neq \emptyset$。此外，$E_2' \neq \emptyset$，因为 E_2' 是 活动域$(\mathcal{I})$中的一个等价类。因此，上文中的 v 和 e始终存在。另请注意，h_2 扩展了 h_1。

根据构造，h_2 是一个映射，它将 under $(\mathcal{I}_2)$中的元素映射到 under$(\mathcal{I})$中的元素。此外，定义 7（同态）中的条件 1-4显然都满足。要证明 h_2 是从 $\mathcal{I}_2$到 $\mathcal{I}$ 的同态，还需证明定义 7中的条件5也成立。 $\mathcal{I}_1$ 中的所有事实（它包含于 $\mathcal{I}_2$）显然都满足该条件，因为 h_2 在这些事实上的行为与 h_1相同，且 h_1 是到$\mathcal{I}$的同态。现在考虑事实 $P_3(E_1, V_1 \cap V_2, \{f_v^{y_1}\}, [f_e^{y_2}])$，它属于 $\mathcal{I}_2 \setminus \mathcal{I}_1$。我们需要证明，$\mathcal{I}$ 中存在一个形如 $P_3(\hat{}E_1, \hat{}V_1, \hat{}V_2, \hat{}E_2)$ 的事实，其中 $\hat{}\hat{}\hat{}\hat{}E_1, E_2 \in$是活动域$_E(\mathcal{I})$中的元素，且 $V_1, V_2 \in$是活动域$_V(\mathcal{I})$中的元素，并且 $\hat{}\hat{}\hat{}\hat{}h_2(P_3(E_1, V_1 \cap V_2, \{f_v^{y_1}\}, [f_e^{y_2}])) \leq P_3(E_1, V_1, V_2, E_2)$，$\hat{}\hat{}\hat{}$即 $h_2(E_1) \subseteq E_1$，$h_2(V_1 \cap V_2) \subseteq V_1$，$h_2(\{f_v^{y_1}\}) \subseteq V_2$, $h_2([f_e^{y_2}]) \subseteq \hat{}E_2$。为此，我们令 $\hat{}E_1 = E_1'$、$\hat{}V_1 = V_3'$、$\hat{}\hat{}V_2 = V_5'$、$E_2 = E_2'$，并证明上述包含关系确实成立。

- $h_2(E_1) \subseteq E_1'$ 可由 $h_2(E_1) = h_1(E_1)$ 和$h_1(E_1) \subseteq E_1'$推出;

- $h_2(V_1 \cap V_2) \subseteq V_3'$是 $h_2(V_1 \cap V_2) =h_1(V_1 \cap V_2) \subseteq h_1(V_1) \cap h_1(V_2)$ 的结果 $\subseteq V_1' \cap V_2' \subseteq V_3' \cap V_4' \subseteq$

V'_3。
由于 $h_2(f_v^{y_1}) = v$和 $v \in V'_5 \cap V'_6$，因此• $h_2(\{f_v^{y_1}\}) \subseteq V'_5$成立，进而 $h_2(\{f_v^{y_1}\}) = \{v\} \subseteq V'_5$。
由于 $h_2(f_v^{y_1}) = e$和 $e \in E'_2$，因此• $h_2([f_e^{y_2}]) \subseteq E'_2$成立，进而 $h_2([f_e^{y_2}]) = [e] \subseteq E'_2$。
我们可以用类似的方法证明，定义 7 的条件5对于另一个事实 $P_4(V_1 \cap V_2, \{f_v^{y_1}\}, [f_e^{y_2}])$ 也成立，该事实包含在 $\mathcal{I}_2 \setminus \mathcal{I}_1$ 中。

实体-egd 追赶步骤的情况：设 r 是形式为 (2) 的实体-egd。根据追赶步骤的定义，μ 是从 ϕ(x) 到 $\mathcal{I}_1$ 的一个赋值，特别地，存在等价类 E_1、$E_2 \in$ 活动域$_E(\mathcal{I}_1)$，使得 $\mu(y) = E_1$ 且 $\mu(z) = E_2$。因此，可以构造一个从 ϕ(x) 到 $\mathcal{I}$ 的赋值 $\hat{}\mu$，使得 $\hat{}\mu(y) = E'_1$、$\hat{}\mu(z) = E'_2$、E'_1、$E'_2 \in$ 活动域$_E(\mathcal{I}_1)$、$h_1(E_1) \subseteq E'_1$ 且$h_1(E_2) \subseteq E'_2$。$\hat{}\mu$ 的构造与 tgd 追赶步骤情形中所述的类似构造相似。为完整起见，我们也针对实体-egd 追赶步骤描述这一过程（注意，此时还需要处理规则体 ϕ(x) 中可能出现的实体和值）。为便于说明，且不失一般性，我们假设 ϕ(x) $= P_1(x, y, e, v) \wedge P_2(x, y, z)$，其中 x 是值变量，y、z 是实体变量，$e \in \mathcal{S}_E$，且 $v \in \mathcal{S}_V$。将算子 τ 应用于 ϕ(x)，得到 $\tau(\phi$(x)) $= P_1(S_1^x, y, e, v) \wedge P_2(S_2^x, y, z)$。由于 μ 是从 ϕ(x) 到 $\mathcal{I}_1$ 的赋值，存在 E、E_1、$E_2 \in$活动域$_E(\mathcal{I}_1)$ 以及V、V_1、$V_2 \in$活动域$_V(\mathcal{I}_1)$，使得 $e \in E$、$V_1 \cap V_2 \neq \emptyset$、$v \in V$、$P_1(V_1, E_1, E, V)$ 和 $P_2(V_2, E_1, E_2)$ 是 $\mathcal{I}_1$ 中的两个事实，并且 $\mu(S_1^x) = V_1$、$\mu(y) = E_1$、$\mu(v) = V$、$\mu(S_2^x) = V_2$，以及 $\mu(z) = E_2$。由于 h_1 是从 $\mathcal{I}_1$ 到 $\mathcal{I}$ 的同态，存在 $P_1(V'_1, E'_1, E', V')$ 和 $P_2(V'_2, E'_1, E'_2)$ 属于 $\mathcal{I}$，使得 $h_1(P_1(V_1, E_1, E, V)) \leq P_1(V'_1, E'_1, E', V')$ 且$h_1(P_2(V_2, E_1, E_2)) \leq P_2(V'_2, E'_1, E'_2)$，即 $h_1(E_1) \subseteq E'_1$、$h_1(V_1) \subseteq V'_1$、$h_1(E) \subseteq E'$、$h_1(V) \subseteq V'$、$h_1(V_2) \subseteq V'_2$、$h_1(E_2) \subseteq E'_2$ 以及 。接下来，我们定义 $\hat{}\mu$ 如下：$\hat{}\mu(S_1^x) = V'_1$，$\hat{}\mu(y) = E'_1$，$\hat{}\mu(v) = V$，$\hat{}\mu(S_2^x) = V'_2$，$\hat{}\mu(z) = E'_2$。显然，$\hat{}\mu$ 是从 ϕ(x) 到 $\mathcal{I}$ 的赋值。特别地，由于 $e \in E$ 且 $h(e) = e$，显然有 $e \in h_1(E) \subseteq E'$，由此可得 $e \in E'$。同理，由于 $v \in \mu(v) = V$ 且 $v = h_1(v)$，我们有$v \in h_1(V) \subseteq V'$，由此可得 $v \in V'$。

接下来令 h_2 为 h_1，并证明将 h_1 视为从 $\mathcal{I}_2$ 到 $\mathcal{I}$ 的映射时，它仍是一个同态。我们很快注意到，在 under($\mathcal{I}_1$) 下，$=$ 位于 ($\mathcal{I}_2$) 下，因此 h_1 也是从 under($\mathcal{I}_2$) 出发的映射。此外，由于 h_1 是一个同态，定义 7 的条件1-4显然都满足。剩下要证明定义 7 的条件5也成立，即我们必须证明，对于 $\mathcal{I}_2$ 中的每个事实 $P(T_1, \ldots, T_n)$，都存在 $\mathcal{I}$ 中的一个事实 $P(U_1, \ldots, U_n)$，使得对于 $1 \leq i \leq n$，$h_1(T_i) \subseteq U_i$。回顾一下，$\mathcal{I}_2$ 是通过将 $\mathcal{I}_1$ 中每个 E_1 的出现以及每个 E_2 的出现都替换为 $E_1 \cup E_2$ 得到的。因此，存在两种情况。（a）对于 $1 \leq i \leq n$，$T_i \neq E_1 \cup E_2$。在这种情况下，结论成立，因为 $P(T_1, \ldots, T_n)$ 也属于 $\mathcal{I}_1$，并且 h_1 是从 $\mathcal{I}_1$ 到 $\mathcal{I}$ 的同态。（b）存在 j，使得$1 \leq j \leq n$且 $T_j = E_1 \cup E_2$。不失一般性，我们还可以假设，对于 $1 \leq i \leq n$ 和 $i \neq j$，$T_i \neq E_1 \cup E_2$。由于两个等价类要么不相交，要么是同一个类，因此每个满足 h_1 的 h_2，要么是一个值集，要么是一个与 h_1 不相交的等价类。

于是，存在 $\mathcal{I}_1$ 中的一个事实 $P(T_1, \ldots, T_{j-1}, T'_j, T_{j+1} \ldots, T_n)$，其中要么 $T'_j = E_1$，要么 $T'_j = E_2$；并且对于每个 $i \neq j$，$T_i \neq E_1$ 且 $T_i \neq E_2$。考虑 $T'_j = E_1$（另一种选择可以用类似方式处理）。那么，必定存在 $\mathcal{I}$ 中的一个事实 $P(U_1, \ldots, U_{j-1}, U_j, U_{j+1} \ldots, U_n)$，使得对于 $1 \leq i \leq n$，$h_1(T_i) \subseteq U_i$。特别地，这意味着$h_1(E_1) \subseteq U_j$，其中 U_j 是活动域$_E(\mathcal{I})$中的一个等价类。由于 $h_1(E_1) \subseteq E'_1$，必有 $U_j = E'_1$。事实上，对于这两个等价类 U_j 和 E'_1，要么 $U_j = E'_1$，要么 $U_j \cap E'_1 = \emptyset$；但第二种情况不可能，因为$h_1(E_1)$ 是 U_j 和 E'_1 的一个共同非空子集。接下来，由于 $\mathcal{I}$ 满足实体 -egd d（该依赖在 d 中由 $\hat{}\mu$ 触发），我们有 $E'_1 = E'_2$，因此由 $h_1(E_1) \subseteq E'_1$ 和$h_1(E_2) \subseteq E'_2$ 可推出 $h_1(E_1) \cup h_1(E_2) \subseteq E'_1$，进而得到$h_1(E_1 \cup E_2) \subseteq E'_1$，也就是说 $h_1(T_j) \subseteq E'_1 = U_j$。这说明 $P(U_1, \ldots, U_{j-1}, U_j, U_{j+1} \ldots, U_n)$ 就是我们要找的 $\mathcal{I}$ 中的事实，从而证明了定义 7 的条件5。

值等值生成依赖追赶步骤的情形：证明与实体-egd 追赶步骤的证明类似。设 r 是形式为 (2) 的实体-egd，并且不失一般性，假设 ϕ(x) $= P_1(y, z) \wedge P_2(z)$，其中 y, z 是值变量（可能出现的实体变量、实体和值可按照本证明前面几种情形中的方法处理）。于是，$\tau(\phi$(x)) $= P_1(S_1^y, S_1^z) \wedge P_2(S_2^z)$。由于 μ 是从 ϕ(x) 到 $\mathcal{I}_1$ 的赋值，因此存在$V_1, V_2, V_3 \in$ 个活动域 $_V(\mathcal{I}_1)$，使得 $\mu(S_1^y) = V_1$, $\mu(S_1^z) = V_2$, $\mu(S_2^z) = V_3$, $V_2 \cap V_3 \neq \emptyset$, $P_1(V_1, V_2)$ 和 $P_2(V_3)$ 是 $\mathcal{I}_1$ 中的事实。由于 h_1 是从 $\mathcal{I}_1$ 到 $\mathcal{I}$ 的同态，因此 $P_1(V'_1, V'_2)$ 和 $P_2(V'_3)$ 在 $\mathcal{I}$ 中是事实，且使得 $h_1(P_1(V_1, V_2)) \leq P_1(V'_1, V'_2)$ 并且 $h_1(P_2(V_3)) \leq P_2(V'_3)$，即 $h_1(V_1) \subseteq V'_1$, $h_1(V_2) \subseteq V'_2$, $h_1(V_3) \subseteq V'_3$。由于 $V_2 \cap V_3 \neq \emptyset$，因此 $V'_2 \cap V'_3 \neq \emptyset$。我们定义一个映射 $\hat{}\mu$，使得 $\hat{}\mu(S_1^y) = V'_1$, $\hat{}\mu(S_1^z) = V'_2$, $\hat{}\mu(S_2^z) = V'_3$。容易看出，$\hat{}\mu$ 是从 ϕ(x) 到 $\mathcal{I}$ 的赋值。

接下来令 h_2 为 h_1，并证明将其视为从 $\mathcal{I}_2$ 到 $\mathcal{I}$的映射时，h_1 仍是一个同态。我们很快注意到，在($\mathcal{I}_1$)下有 $=$ 在($\mathcal{I}_2$)下，因此 h_1 也是从 under($\mathcal{I}_2$)出发的映射。此外，由于 h_1 是一个同态，定义 7 的条件1-4显然都满足。对于$\mathcal{I}_2$ 中也属于$\mathcal{I}_1$的所有事实，定义 7 的条件5也已满足。还需证明，对于 $\mathcal{I}_2 \setminus \mathcal{I}_1$中的事实，该条件同样成立。仅有两个这样的事实：$P_1(V_1 \cup V_2 \cup V_3, V_1 \cup V_2 \cup V_3)$和 $P_2(V_1 \cup V_2 \cup V_3)$（见定义 11 中的值 EGD 追逐步骤）。我们必须证明，在 $\mathcal{I}$ 中存在事实 $P_1(V_1, V_2)$和 $P_2(V_3)$，使得 $h_1(P_1(V_1 \cup V_2 \cup V_3, V_1 \cup V_2 \cup V_3)) \leq P_1(V_1, V_2)$ 且 $h_1(P_2(V_1 \cup V_2 \cup V_3)) \leq P_2(V_3)$，即 $h_1(V_1 \cup V_2 \cup V_3) \subseteq V_1$,$h_1(V_1 \cup V_2 \cup V_3) \subseteq V_2$，且 $h_1(V_1 \cup V_2 \cup V_3) \subseteq V_3$。不难看出，这些事实是 $P_1(V'_1, V'_2)$和 $P_2(V'_3)$。事实上，我们知道 $h_1(V_1) \subseteq V'_1$, $h_1(V_2) \subseteq V'_2$且$h_1(V_3) \subseteq V'_3$。然后，由于 $\mathcal{I}$满足 d，且该条件在 $\mathcal{I}$中由μ触发，因此必有 $V'_1 = V'_2 = V'_3$。于是我们有$h_1(V_1) \cup h_1(V_2) \cup h_1(V_3) \subseteq V'_1 = V'_2 = V'_3$，进而$h_1(V_1 \cup V_2 \cup V_3) \subseteq V'_1 = V'_2 = V'_3$，这就证明了该论点。

□

以下定理是本节的主要结果。

定理 2. 如果 $\mathcal{K}$ 是一个知识库，且 $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots, \mathcal{I}_m$ 是针对 $\mathcal{K}$的有限追赶，那么 追赶算法$(\mathcal{K}, \sigma)$ 是 $\mathcal{K}$的一个通用解。

证明。 由引理 1 可知，追赶算法$(\mathcal{K}, \sigma)$ 是 $\mathcal{K}$的一个模型。接下来需要证明它是通用的，即存在一个同态，将 追赶算法$(\mathcal{K}, \sigma) = \mathcal{I}_m$ 映射到 $\mathcal{K}$的每一个解。证明基于引理2，以及这样一个观察：恒等映射 id 是一个同态，可将 $\mathcal{I}_0$ 映射到每一个 $\mathcal{I} \in$ 解$(\mathcal{K})$。因此，对 σ中的每个 $\mathcal{I}_i, \mathcal{I}_{i+1}$ 应用引理 2，即可证明该命题；从 $\mathcal{I}_0, \mathcal{I}_1$开始，其中 $h_1 = \mathrm{id}$ 是一个同态，将 $\mathcal{I}_0$ 映射到 $\mathcal{K}$的一个解 $\mathcal{I}$。 □

定理 2 意味着，如果 σ 和 σ' 都是针对 $\mathcal{K}$的有限追赶，那么 追赶算法$(\mathcal{K}, \sigma)$ 和 追赶算法$(\mathcal{K}, \sigma')$ 同态等价。因此，针对 $\mathcal{K}$ 进行有限追赶所得的结果在同态等价意义下是唯一的。

容易验证，若 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$ 是一个知识库，其中 $\mathcal{T}$ 中的每个元组生成依赖都是满的（即，$\mathcal{T}$ 中元组生成依赖的头部没有存在量词），那么 $\mathcal{K}$ 的每个追赶序列都是有限的。特别地，当 $\mathcal{T}$ 仅由相等生成依赖组成时，这一结论也成立，这涵盖了所有实体解析场景。

对于无限追赶序列，定义追赶算法的结果时需要格外谨慎。当仅存在元组生成依赖时（Kr¨otzsch、Marx 和 Rudolph 2019；Grahne 和 Onet 2018），或对于涉及可分离元组生成依赖和相等生成依赖的情形（Johnson 和 Klug 1984；Calvanese 等人 2007；Cal`ı、Gottlob 和 Kifer 2013），通常采用的方法是将无限追赶序列的结果定义为各个追赶步骤中生成的所有事实的并集。对于任意元组生成依赖和相等生成依赖，这种方法并不适用，因为由此得到的实例未必满足所有规则，因此按本文的术语也未必是一个解。此外，在我们的框架中，无限追赶序列中所有实例的并集甚至不是当前知识库的一个实例，因为结果中的实体集合并不对应于某个等价关系下的等价类。

一种替代方法可能更适用于包含任意元组生成依赖和相等生成依赖的情形：将无限追赶序列的结果定义为包含所有持久 事实的实例，即追赶序列中某一步引入、且在后续追赶步骤中从未修改的所有事实。此概念由 Beeri 和 Vardi 1984 提出。在我们的设定中，给定一个知识库 $\mathcal{K}$ 和一个无限追赶序列 $\sigma = \mathcal{I}_0, \mathcal{I}_1, \ldots$，这意味着我们将 σ 对 $\mathcal{K}$ 的无限追赶结果追赶算法$(\mathcal{K}, \sigma)$定义如下：

$$chase(\mathcal{K}, \sigma) = \{f \mid \text{there is some } i \geq 0 \text{ such that } f \in \mathcal{I}_j \text{ for each } j \geq i\}. \quad (3)$$

下面的示例表明，这一定义在我们的框架中行不通，因为上述集合可能为空（即使数据库$\mathbf{D}$ 非空）。

示例 12. 设 $\mathcal{K} = \langle \mathcal{T}, \mathbf{D} \rangle$，其中 $\mathbf{D} = \{P(1, 2)\}$ 和$\mathcal{T}$由以下两条规则组成

$$(r_1)\ P(x, y) \to P(y, z); \quad r_2)\ P(x, y) \wedge P(y, z) \to y = z,$$

其中 类型$(P) = \langle \mathrm{v}, \mathrm{v} \rangle$。我们构造一个无限追赶序列，以 $\mathcal{I}_0 = \{P(\{1\}, \{2\})\}$ 为起点，并通过重复按照以下顺序应用上述规则及适当的赋值：r_1，r_1，r_2，r_1，r_2，r_1，r_2，$r_1 \ldots$。

我们得到无限追赶序 列 $\sigma = \mathcal{I}\,\mathrm{q}\mathcal{I}\ \ldots\ \ 0,\ 1,$

$$\begin{aligned}
\mathcal{I}_0 &= \{P(\{1\}, \{2\})\} \\
\mathcal{I}_1 &= \{P(\{1\}, \{2\}), P(\{2\}, \{v_\perp^1\})\} \\
\mathcal{I}_2 &= \{P(\{1\}, \{2\}), P(\{2\}, \{v_\perp^1\}), P(\{v_\perp^1\}, \{v_\perp^2\})\} \\
\mathcal{I}_3 &= \{P(\{1\}, \{2, v_\perp^1\}), P(\{2, v_\perp^1\}, \{2, v_\perp^1\}), \\
&\quad P(\{v_\perp^1\}, \{v_\perp^2\})\} \\
&\ldots \\
\mathcal{I}_7 &= \{P(\{1\}, \{2, v_\perp^1, v_\perp^2\}), P(\{2, v_\perp^1\}, \{2, v_\perp^1, v_\perp^2\}), \\
&\quad P(\{2, v_\perp^1, v_\perp^2\}, \{2, v_\perp^1, v_\perp^2\}), P(\{v_\perp^2\}, \{v_\perp^3\}), \\
&\quad P(\{v_\perp^3\}, \{v_\perp^4\})\} \\
&\ldots
\end{aligned}$$

不难看出，上文（3）中定义的 追赶算法$(\mathcal{K},\ \sigma)$ 为空，即 σ中不存在持久事实。 □

例 12 中的无限追赶序列 σ 是公平的。同时，此示例中的知识库 $\mathcal{K}$ 也存在有限公平序列。事实上，如果我们先应用规则 r_1，然后应用规则 r_2，就会得到有限追赶序列 $\sigma' = \mathcal{I}_0', \mathcal{I}_1', \mathcal{I}_2'$，其中

$$\begin{aligned}
\mathcal{I}_0' &= \{P(\{1\}, \{2\})\} \\
\mathcal{I}_1' &= \{P(\{1\}, \{2\}), P(\{2\}, \{v_\perp^1\})\} \\
\mathcal{I}_2' &= \{P(\{1\}, \{2, v_\perp\}), P(\{2, v_\perp\}, \{2, v_\perp\})\}.
\end{aligned}$$

因此，追赶算法$(\mathcal{K},\ \sigma') = \mathcal{I}_2'$。因此，当某个知识库同时存在有限和无限追赶序列时，还需要进一步研究。我们将其留作未来工作的一个课题。

6 相关工作

关于实体解析的文献浩如烟海，我们仅简要评论与本文相关的少部分早期工作。

Swoosh（Benjelloun 等人，2009）是一种通用的实体解析方法，其中用于比较和合并记录的函数都是“黑箱”。在我们的框架中，匹配函数由实体-egd 和值 egd 决定，而合并函数则通过并集运算实现——这是 Swoosh 方法中合并函数的一个重要特例。从这个意义上说，我们的框架不如 Swoosh 通用。从另一方面来看，我们的框架更为通用，因为它支持元组生成依赖，区分实体和值，并纳入了查询回答。

与匹配依赖（MDs）的相关研究（Fan，2008年；Bertossi、Kolahi 和 Lakshmanan，2013年）一样，我们认为，egd 头部中的变量应当进行匹配和合并，而不是设为相等。在（Bertossi、Kolahi 和 Lakshmanan，2013年）中，使用满足某些自然属性的通用函数来计算匹配结果；而我们使用匹配所涉及的值集或实体集合的并集。在匹配依赖框架中，使用一种追赶过程的变体来解决依赖违反。该追赶算法在每一步都局部作用于一对元组，而我们的追赶过程则以全局方式匹配实体，以局部方式匹配值。此外，我们还支持元组生成依赖，而这并不属于匹配依赖框架。

已有一些关于声明式实体解析及其变体的研究，与我们的框架相关；相关参考文献包括 Dedupalog 框架（Arasu、R´e 和 Suciu，2009）、文献（Burdick 等，2016；Burdick 等，2019）中用于实体链接的声明式框架，以及

较新的 LACE 框架（Bienvenu、Cima 和 Guti´errez-Basulto，2022）。在 Dedupalog 和 LACE 中，都区分了硬性和软性实体消歧规则。LACE 支持全局合并，并像我们一样创建等价类，但不支持局部合并；此外，LACE 将实体消歧规则与否定约束结合起来，而我们的框架则将实体消歧规则与元组生成依赖结合起来。文献（Burdick 等，2016；Burdick 等，2019）中用于实体链接的声明式框架使用键依赖和析取约束，并采用“加权”语义来衡量链接的强度。键依赖被解释为解必须按标准意义满足的硬性规则。因此，两个相互冲突的链接不能共存于同一个解中，该方法便使用修复来定义确定链接的概念。相比之下，在我们的框架中，我们通过追赶算法保留所有备选项，形式为值集或实体的等价类。我们的框架的另一个特点是聚焦于查询的确定答案。最后，上述用于实体链接的声明式框架采用了更一般的链接关系，没有为实体的等价关系设定规则，因此对实体解析的关注较少。

7 结论与未来工作

本文的主要贡献是开发了一个将知识库中的实体解析与查询回答相结合的新型声明式框架。这在很大程度上是一项概念性贡献，因为新声明式框架的开发促使我们从第一性原理重新思考了赋值、同态、模型中元组生成依赖和相等生成依赖的满足性，以及通用解等核心概念的定义。在技术层面，我们设计了一个永不失败的追赶过程，并证明了当该过程终止时，它会生成一个通用解，而该通用解又可用于查询回答。

关于未来的发展方向，也许最紧迫的问题是：当追赶过程不终止时，如何确定追赶结果的一个“良好”概念。这可能促使我们将本文提出的框架扩展到所有通用解都是无限的情形。虽然无限通用解无法被物化，但这类解已被用于获得可重写性结果（Calvanese 等人 2007），有时还会结合对无限追赶结果的部分物化（Lutz、Toman 和 Wolter 2009）。与此同时，确定 TBox 的元组生成依赖和相等生成依赖的结构条件也十分重要；这些条件能够保证追赶过程在多项式时间内终止，从而实现易处理的合取查询回答。此外，探索追赶过程的其他变体是否能够适当地适配到我们的框架中也很有价值，例如半无知追赶（又称 Skolem 追赶）（Marnette 2009；Calautti 和 Pieris 2021）或核追逐（Deutsch、Nash 和 Remmel 2008），使其理想性质得以延续。

最后，我们注意到存在若干不同的领域，包括数据交换（Fagin 等人 2005年）、数据集成（Lenzerini 2002年；Doan、Halevy 和 Ives 2012年）、本体中介查询应答（Bienvenu 和 Ortiz 2015），以及基于本体的数据访问（Calvanese 等人 2018年），其中元组生成依赖和相等生成依赖发挥着关键作用。我们相信，此处提出的框架能够以一种有原则的方式将实体解析融入这些领域。

参考文献

Arasu, A.; R´e, C.; and Suciu, D. 2009. 使用约束进行大规模去重的 Dedupalog 方法。载于*IEEE*第*25*届国际数据工程会议（*ICDE*）论文集，952–963。IEEE计算机学会出版社。Artale, A.; Calvanese, D.; Kontchakov, R.; and Zakharyaschev, M. 2009. *DL-Lite* 家族及其关系。《人工智能研究》36:1–69。Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P. F., 编. 2007.《描述逻辑手册：理论、实现与应用》。剑桥大学出版社，第2版。Baget, J.; Le`cl`ere, M.; Mugnier, M.; and Salvat, E. 2011. 关于带存在变量的规则：在可判定性边界上探索。《人工智能》175(9-10):1620–1654。Beeri, C., and Vardi, M. Y. 1984. 数据依赖的证明过程。《*ACM*杂志》31(4):718–741。Benjelloun, O.; Garcia-Molina, H.; Menestrina, D.; Su, Q.; Whang, S. E.; and Widom, J. 2009. Swoosh：一种通用的实体解析方法。《超大型数据库杂志》18(1):255–276。Bertossi, L. E.; Kolahi, S.; and Lakshmanan, L. V. S. 2013. 使用匹配依赖关系和匹配函数进行数据清洗与查询回答。《理论计算机科学》52(3):441–482。Bienvenu, M., and Ortiz, M. 2015. 使用数据可处理描述逻辑进行本体中介查询应答。载于Faber, W., and Paschke, A., 编，*Reasoning Web*：智能数据访问语义技术——第*11*届国际暑期学校教程讲座（*RW*），第9203卷，218–307。Springer。Bienvenu, M.; Cima, G.; and Guti´errez-Basulto, V. 2022. LACE：一种用于集体实体解析的逻辑方法。载于第*41*届*ACM SIGACT SIGMOD SIGAI*数据库系统原理研讨会（*PODS*）论文集，379–391。ACM出版社。Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2016. 用于实体链接的声明式框架。《*ACM*数据库系统汇刊》41(3):17:1–17:38。Burdick, D.; Fagin, R.; Kolaitis, P. G.; Popa, L.; and Tan, W. 2019. 实体链接框架的表达能力。《计算机系统科学杂志》100:44–69。Calautti, M., and Pieris, A. 2021. 半无视追赶终止：黏性情形。《理论计算机科学》65(1):84–121。Cal`ı, A.; Calvanese, D.; De Giacomo, G.; and Lenzerini, M. 2004. 完整性约束下的数据集成。《信息系统》29:147–163。Cal`ı, A.; Gottlob, G.; and Kifer, M. 2013. 驯服无限追赶：表达力丰富的关系约束下的查询回答。《人工智能研究》48:115–174。Cal`ı, A.; Gottlob, G.; and Lukasiewicz, T. 2009. 一种用于本体上可处理查询回答的通用基于Datalog的框架。载于第28届ACM SIGACT *SIGMOD SIGART* 数据库系统原理研讨会（*PODS*），77–86。Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; 和 Rosati, R. 2007年。描述逻辑中的可处理推理与高效查询回答：*DL-Lite* 系列。自动推理杂志39(3):385–429。Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; 和 Rosati, R. 2018年。基于本体的数据访问与集成。见 Liu, L. 和 Ozsu, M. T. 编，数据库系统百科全书，第二版。Springer。Cuenca Grau, B.; Horrocks, I.; Kr¨otzsch, M.; Kupke, C.; Magka, D.; Motik, B.; 和 Wang, Z. 2013年。存在规则的无环性概念及其在本体查询回答中的应用。人工智能研究杂志47:741–808。Deutsch, A.; Nash, A.; 和 Remmel, J. B. 2008年。重新审视追赶算法。见第*27*届*ACM SIGACT SIGMOD SIGART* 数据库系统原理研讨会（*PODS*）论文集，149–158。ACM出版社。Doan, A.; Halevy, A. Y.; 和 Ives, Z. G. 2012年。数据集成原理。Morgan Kaufmann。Fagin, R.; Kolaitis, P. G.; Miller, R. J.; 和 Popa, L. 2005年。数据交换：语义与查询回答。理论计算机科学336(1):89–124。Fan, W. 2008年。重新审视依赖关系以提升数据质量。见 Lenzerini, M. 和 Lembo, D. 编，第*27*届 *ACM SIGACT SIGMOD SIGART* 数据库系统原理研讨会（*PODS*）论文集，159–170。ACM出版社。Grahne, G. 和 Onet, A. 2018年。追赶算法剖析。*Fundamenta Informaticae*157(3):221–270。Johnson, D. S. 和 Klug, A. C. 1984年。函数依赖和包含依赖下合取查询包含关系的测试。计算机与系统科学杂志28(1):167–189。Kr¨otzsch, M.; Marx, M.; 和 Rudolph, S. 2019。终止追赶算法的力量（特邀报告）。见第*22*届国际数据库理论会议（*ICDT*）论文集，第127卷*LIPIcs*，3:1–3:17。Lenzerini, M. 2002年。数据集成：理论视角。见第*21*届 *ACM SIGACT SIGMOD SIGART* 数据库系统原理研讨会（*PODS*）论文集，233–246。Lutz, C.; Toman, D.; 和 Wolter, F. 2009。使用关系数据库系统进行描述逻辑中的合取查询回答。见第*21*届国际人工智能联合会议（*IJCAI*）论文集，2070–2075。Marnette, B. 2009。广义模式映射：从终止性到可处理性。见 Paredaens, J. 和 Su, J. 编，第*28*届 *ACM SIGACT SIGMOD SIGART* 数据库系统原理研讨会（*PODS*）论文集，13–22。ACM出版社。Papadakis, G.; Ioannou, E.; Thanos, E.; 和 Palpanas, T. 2021。《实体解析的四代发展》。数据管理综合讲座。Morgan & Claypool出版社。

知识库中结合实体解析与查询回答的框架

Ronald Fagin、Phokion G. Kolaitis、Domenico Lembo、Lucian Popa、Federico Scafoglieri

附录

终止追赶序列示例

现在，我们展示当前示例中追赶过程的一次完整执行。

示例 13. 考虑示例 1 和 2 中的知识库 $\mathcal{K} = (\mathcal{T}, \mathbf{D})$，为便于展示，我们在下文列出该知识库：

(r_1) $CI\ p_1\ name_1\ phone_1 \wedge CI\ p_2\ name_2\ phone_2 \wedge JaccSim\ name_1\ name_2\ 0.6 \rightarrow p_1 = p_2$ (r_2) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow name_1 = name_2$ (r_3) $CI(p, name_1, phone_1) \wedge CI(p, name_2, phone_2) \rightarrow phone_1 = phone_2$ (r_4) $CI(p, name, phone) \rightarrow Emp(p, comp) \wedge CEO(comp, dir)$ (r_5) $Emp(p, comp_1) \wedge Emp(p, comp_2) \rightarrow comp_1 = comp_2$ (r_6) $CI(p_1, name_1, phone) \wedge CI(p_2, name_2, phone) \rightarrow SameHouse(p_1, p_2, phone)$

(g_1) CI(**Doe**$_1$, J. Doe, 358) (g_4) Emp(**Doe**$_2$, **雅虎**) (g_2) CI(**Doe**$_2$, John Doe, 635) (g_5) Emp(**Doe**$_3$, **IBM**) (g_3) CI(**Doe**$_3$, Mary Doe, 358) (g_6) CEO(**雅虎**, **Doe**$_1$)

如前所述，示例 3 的解释 $\mathcal{I}$ 是$\mathcal{I}^{\mathbf{D}}$ 的基础实例 $\mathcal{K}$。为方便起见，我们再次在下方列出。

(d_1)CI([**Doe**$_1$], {J. Doe}, {358}) (d_4)Emp([**Doe**$_2$], [**雅虎**])
(d_2)CI([**Doe**$_2$], {John Doe}, {635}) (d_5)Emp([**Doe**$_3$], [**IBM**])
(d_3)CI([**Doe**$_3$], {Mary Doe}, {358}) (d_6)CEO([**雅虎**], [**Doe**$_1$])

一个可能的追赶序列由以下追逐步骤组成：

步骤 1: $\mathcal{I}^{\mathbf{D}} \xrightarrow{r_4, \mu_0} \mathcal{I}_1$，其中 μ_0 满足μ_0(体(r_4)) $= \{d_3\}$。生成的实例 $\mathcal{I}_1$ 包含事实 d_1-d_6，以及以下事实：

$$(d_7)\, Emp([\mathbf{Doe}_3], [\mathbf{e}_1^{\perp}]),\ (d_8)\, CEO([\mathbf{e}_1^{\perp}], [\mathbf{e}_2^{\perp}]).$$

步骤2: $\mathcal{I}_1 \xrightarrow{r_1, \mu_1} \mathcal{I}_2$，其中 μ_1 满足 μ_1(体(r_1)) $=\{d_1, d_2, JaccSim(\{\text{J. Doe}\}, \{\text{John Doe}\}, \{0.6\})$。生成的实例 $\mathcal{I}_2$ 包含事实 d_3, d_5, d_7和 d_8，以及以下事实：

(d_1) $\rightarrow$(d_9) CI(**Doe**$_1$, **Doe**$_2$, {**J. Doe**}, {358}) (d_2) $\rightarrow$(d_{10}) CI(**Doe**$_1$, **Doe**$_2$, {**John Doe**}, {635}) (d_4) $\rightarrow$(d_{11}) Emp([**Doe**$_1$,**Doe**$_2$], [**雅虎**]) (d_6) $\rightarrow$(d_{12}) CEO([**雅虎**], [**Doe**$_1$,**Doe**$_2$])

步骤3: $\mathcal{I}_2 \xrightarrow{r_2, \mu_2} \mathcal{I}_3$，其中 μ_2 满足 μ_2(body(r_2)) $= \{d_9, d_{10}\}$。结果实例 $\mathcal{I}_3$ 包含事实 d_3,d_{11}, d_5, d_{12}, d_7, d_8，以及以下事实：
(d_9) $\rightarrow$(d_{13}) CI([**Doe**$_1$, **Doe**$_2$], {J. Doe,John Doe}, {358}) (d_{10}) $\rightarrow$(d_{14}) CI([**Doe**$_1$, **Doe**$_2$], {J. Doe,John Doe}, {635})

步骤 4: $\mathcal{I}_3 \xrightarrow{r_3, \mu_4} \mathcal{I}_4$，其中 μ_3 满足 μ_3(体(r_3)) $=\{d_{13}, d_{14}\}$。由此得到的实例 $\mathcal{I}_4$ 包含事实 d_3,d_{11}, d_5, d_{12}, d_7, d_8，以及以下事实：
(d_{13}), (d_{14}) $\rightarrow$(d_{15}) CI([**Doe**$_1$, **Doe**$_2$], {J. Doe,John Doe}, {358, 635})
请注意， d_{15} 替换了 $\mathcal{I}_3$ 中的事实 d_{13} 和 d_{14} .

步骤5: $\mathcal{I}_4 \xrightarrow{r_5, \mu_5} \mathcal{I}_5$，其中 μ_5 满足 μ_5(body(r_5)) $=\{d_5, d_7\}$。结果实例 $\mathcal{I}_5$ 包含事实 d_{14}, d_3,d_{11}, d_{12} ，以及以下事实：

(d_5), (d_7) $\rightarrow$ (d_{16}) $Emp([\mathbf{Doe}_3], [\mathbf{IBM}, \mathbf{e}_1^{\perp}])$
(d_8) $\rightarrow$ (d_{17}) $CEO([\mathbf{IBM}, \mathbf{e}_1^{\perp}], [\mathbf{e}_2^{\perp}])$

请注意， d_{16} 替换了 $\mathcal{I}_4$ 中的事实 d_5 和 d_7 。

步骤6: $\mathcal{I}_5 \xrightarrow{r_6, \mu_6} \mathcal{I}_6$，其中 μ_6 满足 μ_6(body(r_6)) $= \{d_{14}, d_3\}$。结果实例 $\mathcal{I}_6$ 包含与 $\mathcal{I}_5$ 相同的事实，以及以下事实

$$(d_{18})\ \ SameHouse([\mathbf{Doe}_1, \mathbf{Doe}_2], [\mathbf{Doe}_3], \{358\})$$

步骤7: $\mathcal{I}_6 \xrightarrow{r_6, \mu_7} \mathcal{I}_7$，其中 μ_7 满足 μ_7(体(r_6)) $= \{d_{14}, d_3\}$ （与步骤 6 相同，但在本例中， d_{14} 是体(r_6)中第二个事实的映像，而 d_3 是第一个事实的映像）。由此得到的实例 $\mathcal{I}_7$ 包含与 $\mathcal{I}_6$ 相同的事实，以及以下事实

$$(d_{19})\ \ SameHouse([\mathbf{Doe}_3], [\mathbf{Doe}_1, \mathbf{Doe}_2], \{358\})$$

不难看出， $\mathcal{T}$的规则均无法应用于 $\mathcal{I}_7$，因此追赶过程终止。追赶算法的结果确实是 $\mathcal{I}_7$，与示例 3 中给出的 $\mathcal{I}'$ 一致。

□