使用答案集编程处理可废止条件句

Racquel Dennison开普敦大学和 CAIR 开普敦，南非
dnnrac003@myuct.ac.za

Jesse Heyninck
开放大学 海尔伦，林堡，荷兰 开普敦大学和 CAIR 开普敦，南非 jesse.heyninck@ou.nl

Thomas Meyer开普敦大学和 CAIR 开普敦，南非
tmeyer@cair.org.za

可废止蕴涵关注的是如何从不完整信息中得出合理的结论。KLM 框架是对可废止蕴涵进行建模的基础框架。KLM 框架由 Kraus、Lehmann 和 Magidor 提出，概述了可废止蕴涵的若干关键性质。该框架中最著名的算法之一是理性闭包（RC）。本文提出一种基于答案集编程（ASP）计算理性闭包的声明式定义。我们的方法能够根据给定的知识库自动构建最小排序模型，并支持对指定查询进行蕴涵检查。我们从形式上证明了 ASP 编码的正确性，并通过实证评估，将我们的实现与现有的命令式实现（特别是 InfOCF 求解器）的性能进行比较。结果表明，基于 ASP 的方法符合理性闭包的理论基础，并能提高计算效率。

1 引言

形式逻辑在应对人工智能中的推理挑战方面发挥着至关重要的作用 [14]。它能够以精确且结构化的方式表示信息，从而使推理过程能够以算法方式建模 [25]。

传统上，推理算法具有单调性，这意味着向一个领域添加新信息不会推翻此前得出的结论 [15]。然而，在对领域特征进行建模时，单调性会带来挑战 [22] ，因为新信息可能与现有规则相矛盾，或构成例外。

非单调推理（NMR）旨在通过纳入处理例外情况的机制来解决这些局限 [2]。这涉及在获知新信息后，对先前作出的假设进行调整和修订。NMR 基于典型性的概念，即给定对象的属性并非必然全部成立 [7]。例如，我们不说“所有鸟都会飞”，而是说“鸟通常会飞”，因为也存在鸵鸟和企鹅这样的例外情况。通过将对象的特征定义为“通常”成立，我们便为可能出现的例外留出了余地。

可废止推理为这类推理提供了一个建模框架。在可废止推理的各种方法中，由 Kraus、Lehmann 和 Magidor 开发的 KLM 框架[16] 因界定了任何可废止推理框架在处理不完整信息时都应遵循的基本性质而广为人知 [18]。

满足 KLM 性质的一种著名推理模式是理性闭包（RC）。理性闭包是一种保守的可废止推理形式，这意味着它从知识库 [4] 中推导出最小假设。理性闭包既有语义定义，也有算法定义。

尽管 KLM 和答案集编程都用于对非单调推理建模，但它们在建模非单调推理时采用了不同的范式。KLM 框架采用偏好语义方法，

M. Gebser、D. Inclezan、F. Ricca、M. Carro 和 M. Truszczynski：© Racquel Dennison、Jesse Heyninck 和 Thomas Meyer 第41届国际逻辑编程会议（ICLP 2025）本作品采用知识共享署名许可协议授权。EPTCS439，2026年，第206–223页，
doi:10.4204/EPTCS.439.15

依据偏好关系对所有可能世界进行排序。若公式 α在所有最优模型 [27]中都成立，则称其可满足。答案集编程采用失败即否定，这是一种默认推理形式：如果给定程序中的规则无法证明公式 α为真，就假定它为假。这一区别凸显了两种系统处理不确定性和不完整信息的关键差异。

现有的大多数用于计算满足 KLM性质的蕴涵关系的方法都是命令式的 [4], ，也有一些工作探索了声明式编程技术 [12]。

本文提出一种使用答案集编程对理性闭包进行声明式定义的替代方案。我们给出理性闭包的编码，并证明这些定义的正确性。此外，我们还报告了对理性闭包定义的计算运行时间进行测试所得的结果。通过大量实验，我们表明，基于答案集编程的实现优于 InfOCF 求解器 [1]。

2 背景

本节介绍相关背景知识。第 2.1 节介绍命题逻辑。第 2.2 节介绍可废止推理和 KLM方法。第 2.3 节和第 2.4 节分别介绍理性闭包和答案集编程。

2.1 命题逻辑

所有命题原子的有限集合记为 P。命题原子被赋予一个真值，即真或假 [23], ，并通过布尔运算符组合成命题语句，也称为公式。公式用希腊字母表示，例如 α、 β 和 γ。语言记为 L，是由公式构成的集合。对于任意给定的 $p \in P$ 和 τ、$\alpha \in L$， τ 可具有以下形式：$p \neg\tau$、$\tau \wedge \alpha$、$\tau \rightarrow \alpha$、$\tau \leftrightarrow \alpha$、$\tau \vee \alpha$ [23]。知识库 $K \subseteq L$ 是由命题公式构成的集合，可以是有限的，也可以是无限的。本文仅考虑有限知识库。我们将解释 ω定义为从命题原子到真值的映射。解释 ω 定义为 $\omega : P \rightarrow \{T, F\}$ [23]。我们将公式在解释下的取值记为 I(α)。如果对于某个公式 α，I(α) 为真，则称 I 满足 α，记为 $I \Vdash \alpha$。如果至少存在一个解释 I 满足 K，那么 I 就是 K 的模型。K 的所有模型构成的集合记为 mod(K)。如果不存在 $I \in U$，使得 $I \Vdash K$，那么 K 中不存在能够满足 K 中所有 $\alpha \in$ 的解释。如果不存在满足 $I \Vdash K$ 的 I，则称 K 不可满足。从知识库中进行推理称为蕴涵，记为 $\models$。

2.2 可废止推理

可废止推理是一种非单调框架，当新的矛盾信息被引入知识库时，该框架允许撤回先前确立的规则 [1]。为了对可废止推理进行建模，人们提出了多种形式，包括信念修正 [24], 默认逻辑 [11],和偏好方法 [26]。偏好方法后来扩展为现称的 KLM 框架 [16]。由于该框架具有定义明确的性质和计算效率高的推理算法，本文主要关注这一框架，其中一种算法是理性闭包（简称 RC） [4]。

我们的工作立足于 KLM 框架，该框架通过引入可废止蕴含 [18] 扩展了命题语言。可废止蕴含记作 $\mid\sim$，表示为 $\alpha \mid\sim \beta$ 其中 $\alpha, \beta \in$

[1] https://github.com/jonasphilipp/InfOCF

L。因此，我们将语言 $L^{p} = L \cup \{\alpha \mid\sim \beta \mid \alpha,\beta \in L\}$ 定义为命题语言 L 的扩展，并在其中添加可废止联结词。这样，我们便可以使用 $\mid\sim$ 构造公式。请注意，我们不允许嵌套使用 $\mid\sim$。

从一组可废止蕴含式中得出推理，称为可废止蕴涵。可废止蕴涵（$\mid\approx$）的概念并不唯一；人们已经提出了许多不同的可废止蕴涵定义方式。本文关注 KLM 定义的可废止蕴涵，这是因为它具有 KLM性质 [16], 以及与之相关的定义良好的算法。

2.3 理性闭包

RC 是一种保守的可废止推理形式，这意味着推理算法从知识库 [15]中尽可能少地进行推断，同时允许处理例外信息。该框架既有语义定义，也有算法定义 [4]。本文将重点讨论 Casini 等人 [4]所概述的 RC 算法定义。使用 RC 计算可废止蕴涵需要两个算法：BaseRank 和 RC。

算法 1 BaseRank

1：输入：知识库 K 2：输出：有序元组 $(R_0,\ldots, R_{n-1}, R_\infty, n)$。其中每个 R_i（$i \in [0,n]$）表示每个陈述所属的等级。3：$i = 0$；Ð→4：$E_0 = K$；5：当 $E_{i-1} \neq E_i$ 时 6：$E_{i+1} = \{ \alpha \rightarrow \beta \in E_i \mid E_i \models \neg\alpha\}$；7：$R_i = E_i \setminus E_{i+1}$；8：$i = i + 1$；9：结束循环 10：$R_\infty = E_{i-1}$；11：如果 $E_{i-1} = \varnothing$ 则 12：$n = i - 1$；13：否则 14：$n = i$；15：结束条件判断 16：返回 $(R_0,\ldots, R_{n-1}, R_\infty, n)$；

BaseRank 是一种根据陈述的例外性为知识库中的陈述赋予数值秩的过程。适用于某一特定主题广泛范围的一般陈述会被赋予较低的秩，而违反知识库中既定规则的更具例外性的陈述则会被赋予较高的秩。例如，给定知识库 $K = \{man \mid\sim mortal, Socrates \rightarrow man, Socrates \rightarrow \neg mortal\}$，陈述 $Socrates \rightarrow \neg mortal$ 是例外的，而 $man \mid\sim mortal$ 是适用于人这一类别的一般陈述。

BaseRank 过程对已转换为仅包含经典实质蕴涵的知识库进行排序。我们将这一过程称为物化，并在定义 1 中对其进行形式化。算法 1 概述了 BaseRank 过程。

定义 1. 知识库 K 的物化定义为 $Ð{\rightarrow}K := \{\alpha \rightarrow \beta \mid \alpha \mid\sim \beta \in K\}$ [15]。

定义 2. 如果 $K \models \neg\alpha$ [15]，则公式 α 在集合 K 中是例外的。

理性闭包判定知识库是否蕴涵一个蕴含式。对于给定的查询 $\alpha \mid\sim \beta$， α 是前件， β 是后件。为了计算给定查询 $\alpha \mid\sim \beta$ 的可废止蕴涵，理性闭包程序接收一个可废止知识库，计算 BaseRank，然后反复消除知识库中被分配到各个秩的陈述，只要这些陈述会使前件变得异常，最终确定 Ð→K′，即 Ð→K 的极大子集，且该子集不会使 α 变得异常。如果 K′ 经典蕴涵 $\alpha \rightarrow \beta$ Ð→，理性闭包程序就对该查询返回 true。理性闭包下的蕴涵记作 $K \mid\approx_{RC}$。算法 2 概述了理性闭包程序。

算法 2 理性闭包

1: 输入：知识库 K 和可废止蕴含式 $\alpha \mid\sim \beta$。
2: 输出：如果查询可由知识库蕴含，则为 true；否则为 false。
3: $(R_0,\ldots,R_{n-1},R_\infty,n)$ = BaseRank(K);
4: i = 0;
5: $R = \bigcup^{j<n-1}_{j=0} R_j$;
6: 当 $R_\infty \cup R \models \neg\alpha$ 且 $R \neq \varnothing$ 时
7: $R = R \setminus R_i$;
8: i = i + 1;
9: 结束当
10: 返回 $R_\infty \cup R \models \alpha \rightarrow \beta$;

示例 1 给定知识库 K_1 = {人 $\mid\sim$必死的，苏格拉底 → 人，苏格拉底 →¬必死的} 和查询：苏格拉底 →人。查询的前件是苏格拉底。K_1 的物化结果为 {人 → 必死的，苏格拉底 → 人，苏格拉底 → ¬必死的}，K_1 的基础等级见表1。

表1：知识库 K_1的 BaseRank

$\mathcal{R}_\infty$	苏格拉底 →人，苏格拉底 → ¬必死的
$\mathcal{R}_0$	人 → 必死的

表2：知识库 K 的 $BaseRank_1$（已删除最低等级）。

$\mathcal{R}_\infty$	苏格拉底 →人，苏格拉底 → ¬必死的
$\mathcal{R}_0$	~~人 →必死的~~

为确定 $R \cup R_\infty \models$ ¬苏格拉底；我们计算 K_1 的模型。我们用 s 表示苏格拉底，m 表示人，w 表示必死的。K_1 的模型为 mod(K_1) ={smw, smw, smw}。我们注意到，对于 K_1 ¬的所有解释，苏格拉底都成立，$R \cup R_\infty \models$ ¬苏格拉底。因此，理性闭包将排除秩 0。修订后的 K_1 模型为 {smw, smw,smw, smw}。我们看到 $R \cup R_\infty \not\models$ ¬苏格拉底，因为存在一种解释使苏格拉底为真。对 $R \cup R_\infty$进行经典蕴涵计算后，我们发现剩余秩中至少有一个模型使陈述“苏格拉底 →人”为真，因此在理性闭包下，$K \mid\approx_{RC}$ 苏格拉底 → 人。这个例子凸显了理性闭包的用途。在经典逻辑下，我们会得出苏格拉底并不存在的结论。理性闭包使我们能够捕捉苏格拉底这一例外情况，并在知识库中表示这一情况。

2.4 答案集编程

答案集编程是一种用于解决复杂问题的现代声明式编程方法 [21]。它以易于使用、表达能力强的构造以及计算效率高而著称 [8]。答案集编程由类似于 Prolog 中规则的规则组成 [5]，并采用利用高效可满足性求解器背后概念的计算方法 [8]。求解答案集编程程序需要确定稳定模型 [20]。稳定模型是满足答案集编程程序中所规定规则的一组原子。问题会被编码为程序，并由 gringo [10] 和 clasp [9]等答案集编程工具处理。本文使用的求解器是 clasp [9]。Clingo [8] 将 gringo 和 clasp 整合为一个单体系统，是一种基于 ASP 编程范式的编程工具。答案集编程中的规则形式如下：$a_0 \leftarrow a_1, a2, ..., a_m, \sim a_{m+1}, .., \sim a_n$。我们将其头部记为 $head(r) = \{a_0\}$。体部定义为 $body(r) = \{a_1, a2, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}$。规则的头部和体部都由文字组成。文字是正的或负的原子 [8]。体部中的正文字集合记为 $body^+(r) = \{a_1, a2, ..., a_m\}$，负文字集合记为 $body^-(r) = \{a_{m+1}, ..., a_n\}$。直观地说，若有一个集合 X，由程序中的文字构成，那么如果 $body^+(r) \subseteq X$ 且 $body^-(r) \cap X = \varnothing$，则 $head(r) \in X$ [8]。答案集是答案集编程程序的极小支持模型。完整定义见 [8]。

选择规则描述了形成稳定模型的多种替代方式，其中规则的头部由花括号中的表达式 [8] 构成。例如：{a;b}

列出了文字 a 和 b 被纳入稳定模型的所有可能方式。共有四种可能的选择，即 {a,b} 的幂集，也就是 {{∅},{a},{b},{a,b}}。

答案集编程允许表达受最小化和/或最大化约束的代价函数。这在解决优化问题时尤其有用。我们所关注的函数是最小化语句，其表达形式为 $minimize\{l_1 = w_1@p1, \ldots, l_n = w_n@p_n\}$。最小化指令指示求解器计算最优答案集，使元素的加权和 $l_1 = w_1@p1, \ldots, l_n = w_n@p_n$ 最小。

3 使用声明式方法处理 BaseRank 和理性闭包

本节介绍 BaseRank（第3.1节）和 RC（第3.2节）的定义。在这两部分中，我们分别描述问题实例的结构及其在答案集编程（ASP）中的对应问题编码。

3.1 BaseRank

我们通过以下谓词 classical/2 和 defeasible/2 来描述经典蕴含（$\alpha \rightarrow \beta$）和可废止蕴含（$\alpha \mid\sim \beta$）。这些谓词充当 BaseRank 的问题实例。代码清单 1 展示了知识库 K_1（如示例 1 所示）的表示方式。

[1] 可废止 (人，必死的)。[2] 经典 (苏格拉底，
人

代码清单 1：知识库 K_1的编码

BaseRank 的问题编码通过模式规则求解。模式规则包含变量，并允许对逻辑规则进行概括 [8]。我们在清单 2 中对 BaseRank 的定义遵循生成、定义和测试方法 [19]。

第 2 行和第 3 行的初步规则将知识库编码转换为实质蕴涵，由谓词 m implication/2 定义。随后，所有经典规则都被赋予无穷秩，如第 4 行所述。

为确保知识库本身不存在矛盾，我们在第 5、6 和 7 行定义谓词 infer/2。任何能推出 $\top \rightarrow \alpha$ 和 $\neg\alpha \rightarrow \alpha$ 的知识库都将不可满足；因此，infer/2 谓词使我们能够定义约束来捕获这些情况，并定义知识库的全局状态。

第 8 行的生成部分使用选择规则，为每个实质蕴含式任意指定一个数值秩。谓词 rank(m implication(X, Y), N) 表示实质蕴含式 $X \rightarrow Y$（编码为 m implication(X, Y)）被指定了数值秩 N。该选择规则包含一个基数约束，要求基数为 1。这确保每个解都会将一个实质蕴含式置于某个秩上。

在最坏情况下，每个陈述都会被置于一个独立的秩上。因此，第 8 行的选择规则会将每个实质蕴含式分配到任意秩，而陈述可被置于的最大秩取决于知识库中的陈述数量。例如，考虑 K_1,时，在最坏情况下，每个陈述都会被指定一个独立的秩，这意味着排序模型中有 3 个秩。

编码的定义部分位于第 9 至 12 行。该部分会推导知识库中所有可能的后果。derive/3 谓词对传递性、同一律和析取消去的自然演绎规则进行编码。谓词 derive(X,Y,N) 定义为 $X \rightarrow Y$ 在层级 N 上成立。

编码的测试部分位于第 14 至 19 行。第 14 行确保：如果一个答案集包含谓词 derive(X,Y,N) 和 derive(X –Y, N)，则该答案集会被拒绝。第 15 行确保，任何作为重言式的前件都不会在该层级上包含其自身的否定。第 18 和 19 行确保不一致的知识库会被拒绝。

第 21 至 23 行使用 #minimize 优化指令，确保所有蕴含式在保留其分布结构的同时被赋予尽可能低的秩；此外，所有作为重言式的前件都被赋予尽可能低的秩。

---

1 m_蕴含式 (X ,Y) :- defeasible (X ,Y). 2 m_蕴含式 (X ,Y) :- classical (X ,Y). 3 陈述_count (C):- C =#count {X Y) ra: N) P , de N) N1 in

```
#minimize{N,X,Y:rank(m_蕴含式(X,Y),N)}.#minimize{N,X:derive(top,X,N)}.#
mi
sh
```

清单 2：BaseRank 的问题编码

使用清单 2 中定义的 BaseRank 定义计算得到的 K_1 的排序模型如清单 3 所示。

3.2 理性闭包

查询通过谓词 query(X, Y) 定义，而通过计算 BaseRank 得到的陈述的排序层级则由谓词 rank(m_implication(X, Y), N) 描述。我们的理性闭包定义可以处理形如 $p \mid\sim q$ 和 p 的查询，其中 p、q 是文字且不含布尔联结词。

```
rank(m_蕴含式(苏格拉底,人),无穷大).rank(m_蕴含式(苏
格
```

代码清单 3：K_1 的 BaseRank 模型

与 BaseRank 类似，RC 遵循生成、定义与测试方法。

在代码清单 4 的第 1 行，我们通过选择规则生成谓词 guess/1。该规则规定，对于 rank(_,N) 事实中存在的所有秩级 N，恰好选择一个 guess(N)。谓词 guess/1 表示从排序模型中任意选取的秩。

谓词 inference/2 对知识库能够推理出的实质蕴涵进行编码。第 4 至 8 行的规则概述了这些内容。第 4 行假设存在某个解释，使前件 X 为真；这对应于自然演绎中的恒等规则，因此将事实 inference(X,X) 添加到答案集中。第 6 行使用实质蕴涵的传递规则，第 8 和第 9 行则通过逆否规则，涵盖了从实质蕴涵得出的所有逻辑推理。

```
1 { guess(N): rank(_,N) }=1. 2 inference(X,X) :- rank(m_蕴含式(X,Y),N1), guess(N), N<=N1. 3
in
ra
N
to
(X
en
```

代码清单 4：理性闭包的问题编码

第 7 行和第 8 行在确定哪些等级满足查询前件方面发挥着重要作用。这两行都对析取消去进行了编码。对于第 7 行，如果我们有某个推理，例如 inference(X,Y) 和 inference(–X, Y)，那么后件 Y 总是成立。第 8 行指出，如果我们能够推出 inference(X, Y) 和 inference(X –Y)，那么 –X 为真，也就是说 X 为假。

推理规则对应于标准的自然演绎规则，即同一律、肯定前件、逆否命题和析取消去。

第 9 行拒绝前件在猜测的等级中属于例外的任何答案集。第 10 行定义了蕴涵。如果 inference(X, Y) 在给定答案集中有定义，并且 query(X, Y) 也成立，则该查询就被知识库蕴涵。当答案集中没有证据表明 entailed(true) 成立时，陈述不被知识库蕴涵。

#minimize 指令确保选择不会导致例外前件的最小等级。

示例 2 我们首先定义一个简单的知识库 K_1，，如代码清单 5 所示。其中包括可撤销条件句和经典条件句，以及一个查询。该知识库编码在名为 knowledge base.lp 的文件中。

```
1 可废止 (人 , 必死的 ).2 经典 ( 苏格拉底 , 人 ). 经典 ( 苏格拉底 ,- 必死的 ).3 查
询
```

代码清单 5：知识库 K_1 的编码

要使用 RC 程序评估查询，我们在终端中运行以下命令：clingo knowledge base.lp base rank.lp rational closure.lp。- 所得的最优答案集见代码清单 6。输出表明，求解器只计算出了一个答案集。我们观察到，猜测的等级为无穷等级，由此得出了答案集中所示的推理。由于使用推理规则得到的答案集中包含原子 inference(socrates, man)，因此知识库蕴涵查询 query(socrates, man)。

```
1 查询( 苏格拉底 , 人 )          猜测 ( inf )
2 推理 ( 苏格拉底              , 苏格拉底 ) 推理 (- 人 ,- 苏格拉底 )
3 推理 ( 必死的 ,- 苏格拉底 ) 推理 ( 苏格拉底 , 人)
4 推理 ( 苏格拉底              , - 必死的 ) 蕴涵 ( 真 )
```

代码清单 6：使用 Clingo 对 K_1 的理性闭包蕴涵结果

3.3 可靠性和正确性

为了证明我们对 BaseRank 和理性闭包的答案集编程定义的正确性，我们分别证明它们的可靠性和完备性。

3.3.1 BaseRank

首先介绍一些记号方面的预备知识。给定知识库 K，我们将 encoding(K) 记为其编码（见第 1 节），并将清单 2 中的程序（不包括第

22–24 行的 minimize 指令）记为 BaseRank。我们将清单 2 第 22–24 行的 minimize 指令记为 baseRankMin。此外，对于 encoding(K)∪baseRank 的给定答案集 S，我们定义 $K_i^S$and Ð→K $_{Si}$如下：

$$\mathcal{K}_i^{\mathcal{S}} = \{A \mid\sim B \in \mathcal{K} \mid \texttt{rank(m_implication(A,B),i)} \in \mathcal{S}\}$$

$$\overrightarrow{\mathcal{K}}_i^{\mathcal{S}} = \{A \rightarrow B \mid A \mid\sim B \in \mathcal{K}_i^{\mathcal{S}}\}$$

以下内容表明，BaseRank 中编码的导出规则与自然演绎中使用的规则完全相同 [6]。

命题 1. 设 i 为固定的非负整数，S 为满足 BaseRank 中所列规则（不含最小化指令）的某个答案集。进一步定义

$$\mathrm{ND}_N = \left\{X \rightarrow Y \;\middle|\; \overrightarrow{\mathcal{K}}_i^{\mathcal{S}} \models X \rightarrow Y\right\}$$

and

$$\mathrm{ASP}_i = \left\{X \rightarrow Y \;\middle|\; \texttt{derive(X,Y,i)} \in \mathcal{S}\right\}.$$

Then

$$\mathrm{ND}_i = \mathrm{ASP}_i.$$

以下命题表明，BaseRank 算法所赋予的等级遵循 BaseRank 算法中使用的容忍或例外性概念：

命题 2. 给定一个知识库 K。对于 encoding(K)∪baseRank 的给定答案集 S，Ð→对于任意 i ≥ 0，以及对于任意 A $\mid\sim$ B $\in K_i^S$, $\cup_{j=i}^{\infty} K_j^S \not\models \neg$A。

此外，BaseRank 是可靠的，也就是说，K 的基础等级的每一种“细化”都会作为 encoding(K)∪baseRank 的一个答案集出现：

命题 3. 设给定一个知识库 K，其 BaseRank(K) 为 =(R0, . . . ,R$_n$, R$_\infty$)。对于 K 的任意划分 (R$'_0$, . . . , R$'_m$, R$_\infty$)，使得对于每个 R$'_i$，都存在一个 j，使得 R$'_i \subseteq$ R $_j$，编码(K)∪baseRank 存在一个答案集 S，使得对于每个 i ≥ 0，K_i^S = R$'_i$。

最后，我们证明，baseRank 编码的最小答案集与 BaseRank 算法生成的划分一致：

命题 4. 设给定一个知识库 K，其 BaseRank(K) 为 =(R0, . . . , R$_n$, R$_\infty$)。对于编码(K) ∪ baseRank∪baseRankMin 的答案集 S，对于 i ≥ 0，K_i^S = R$_i$。

3.3.2 理性闭包

我们将理性闭包 ASP 算法记为 RC，并将 encoding(K)记为 RC 的 BaseRank。此→外，对于任意给定的答案集 S，我们将 $\cup_{j=n}^{\infty} K_j^S$ ={ A $\mid\sim$B |rank(m implication(A, B),i)∈ S} 记为

我们首先证明该编码是可靠的：

命题 5. 给定某个知识库 K 和查询 A|∼ B，若 entailed(true) ∈S，其中 S 是最小答案集，则 K $\mid\approx_{RC}$ A $\mid\sim$ B。

同样，该编码也是完备的：

命题 6。给定某个知识库 K 和查询 A|∼ B，如果 K $\mid\approx_{RC}$ A $\mid\sim$ B，那么存在一个最小答案集 S，使得 Π where entailed(true) ∈ S。

4 实验与结果

为了评估我们的理性闭包定义的计算效率，我们进行了多项实验，以测试从知识库中回答查询的运行时间。所使用的知识库由形如 $\alpha \mid\sim \beta$ 的公式组成。查询是知识库中明确存在的陈述。下一节将概述实验过程以及所进行的测试。我们将实现与 InfOCF 求解器对 System Z 的实现进行了比较。

我们进行了实验，以确定使用我们的 ASP 理性闭包定义进行蕴涵计算答案集所需的运行时间。测试在一台配备 M1 处理器的 MacBook Air 上进行，该设备搭载八核中央处理器和七核图形处理器。我们通过 Clasp 求解器在进程完备后记录的统计信息，获取使用我们的理性闭包定义计算蕴涵所需的时间。为了计算 InfOCF 求解器中理性闭包的运行时间，我们使用 Python 库中的计时函数，测试了计算推理所需的时间。需要注意的是，参与测试的 InfOCF 求解器版本是使用 Python 3 编写的。

我们的实验包含两个变量：知识库的大小，以及在该知识库上计算推理所需的时间。我们测试的知识库规模从 25 条陈述到 325 条陈述，每次递增 25 条。对于每个知识库，我们都测试了计算指定查询的推理所需的时间。此外，我们进行了 2000 次独立试验，以准确呈现所需时间。由于 InfOCF 求解器出现递归错误，超过 325 条陈述后的测试便停止了。

图 1 展示了在知识库规模为 25 至 325 条陈述的范围内，我们的基于 ASP 的理性闭包求解器与 InfOCF 求解器实现的平均运行时间（±95% 置信区间）。基于 ASP 的理性闭包实现始终在所有知识库规模下都显著缩短运行时间，清楚地体现了其计算效率。值得注意的是，随着知识库规模增大，两个求解器的运行时间差距也随之扩大。

InfOCF 求解器的运行时间随知识库规模呈线性增长，而随着知识库规模增大，我们的方法的运行时间扩展性显著更好。ASP 求解器的最小二乘回归为 $0.00003x + 0.0013$，而 InfOCF 求解器的回归为 $0.00035x + 0.0050$。比较两者的斜率可以发现，我们的方法运行速度快 12 倍。

置信区间（CI）进一步体现了效率差异：即使在测试的最大知识库规模（325 条陈述）下，ASP 的置信区间仍稳定在 ± 1 ms，表明其性能稳定且可预测。相比之下，InfOCF 求解器的置信区间在约 275 条陈述时扩大至约 ± 3 ms，反映出其运行时间对规模更为敏感。

从量化结果来看，在知识库规模最大时，我们的方法运行时间约为 0.011 秒，而 InfOCF 求解器需要 0.12 秒，计算时间减少约 91%。这些结果表明，基于答案集编程的理性闭包求解器在蕴涵推理任务中不仅效率更高，扩展性也更强。

除了计算性能之外，ASP 本身的声明式特性还带来了显著的实际优势。ASP 的语法清晰，由逻辑蕴含构成；一旦理解了基本的逻辑运算，程序就具有很高的可读性。

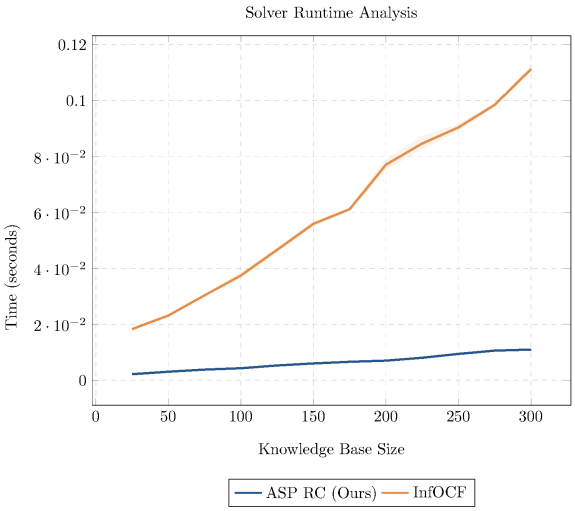


图 1：我们基于答案集编程实现的理性闭包与 InfOCF 求解器的比较

此外，RC 的 ASP 编码便于修改和扩展。例如，将我们的 RC 求解器扩展为支持字典序闭包，只需调整排序模块，而不必彻底重新设计程序的规则。此外，ASP 的透明性使人们能够直观地了解推理过程。由于 ASP 规则直接表示逻辑规范，因此如何根据给定前提推导出答案集一目了然。这有助于简化调试过程，并提高长期可维护性。

5 结论与未来工作

本文提出了一种新颖的声明式定义，使用答案集编程计算理性闭包框架下的蕴涵。我们阐述了该方法背后的基本直觉，并给出了对使用答案集编程定义的 BaseRank 和理性闭包正确性的形式化证明。此外，我们进行了实验评估，将我们的实现与用于计算理性闭包的命令式替代方案 InfOCF 求解器进行了比较。结果表明，在计算运行时间方面，基于答案集编程的定义显著优于 InfOCF 求解器。

除了缩短计算运行时间之外，我们还发现，答案集编程在 KLM 框架中具有其他几项优势。其中一项是其声明式特性，强调问题的“是什么”，而非“如何做”。我们发现，这种抽象有助于加深对理性闭包的概念性理解。在对理性闭包进行建模时，声明式范式不仅使问题的描述更加清晰，也为理解 KLM 蕴涵关系背后的逻辑提供了全新的视角。另一项优势是编码简洁易读。

本文提出的 RC 定义考虑了由如下形式的陈述构成的知识库： $\alpha \mid\sim \beta$ where α and β 是文字。未来的工作可以通过探索涉及合取和析取的更复杂逻辑陈述来扩展这一研究。此外，虽然 RC 是较为知名的可废止推理形式之一，但 KLM 框架还提出了许多其他可废止蕴涵形式，未来可以使用答案集编程对其进行研究。其中包括字典序闭包 [17]和相关闭包 [3]。

致谢

作者谨此感谢匿名审稿人提出的宝贵意见。本研究部分基于南非国家研究基金会资助的研究（参考编号：SAI240823262612）。Jesse Heyninck 部分得到了项目 LogicLM: Combining Logic Programs with Language Model 的资助，该项目编号为 NGF.1609.241.010，隶属于研究计划 NGF AiNed XS Europa 2024-1；该计划由荷兰研究委员会（NWO）部分资助。

参考文献

[1] G. Antoniou、D. Billington、G. Governatori、M. J. Maher 和 A. Rock（2000）：一类可废止推理逻辑及其实现。载于：第14届欧洲人工智能会议论文集，ECAI’00，IOS Press，NLD，第459–463页。[2] Gerhard Brewka、Ilkka Niemel¨a 和 Mirosław Truszczy´nski（2008）：第6章 非单调推理。载于 Frank van Harmelen、Vladimir Lifschitz 和 Bruce Porter 编：《知识表示手册》，人工智能基础第3卷，爱思唯尔，第239–284页，doi：https://doi.org/10.1016/ S1574-6526(07)03006-4。[3] Giovanni Casini、Thomas Meyer、Kodylan Moodley 和 Riku Nortj´e（2014）：相关闭包：一种用于描述逻辑的可废止推理新形式。载于 Eduardo Ferm´e 和 Jo˜ao Leite 编：《人工智能逻辑》，施普林格国际出版，Cham，第92–106页，doi：https://doi.org/10.1007/ 978-3-319-11558-0_7。[4] Giovanni Casini、Thomas Meyer 和 Ivan Varzinczak（2018年10月27–29日）：可废止蕴涵：从理性闭包到字典序闭包及其他。载于：第17届非单调推理国际研讨会（NMR）2018，美国亚利桑那州，第109–118页。获取地址：https://researchspace.csir.co.za/items/438002eb-e5fe-49a7-b46c-183e19697404。[5] Alain Colmerauer（1990）：Prolog III 简介。《ACM通讯》33(7)，第69–90页，doi：10.1145/ 79204.79210。[6]Amr Elnashar 和 Wafik Boulos Lotfallah（2018）：最小完备命题自然演绎系统。《哲学逻辑杂志》47 47(4)，第803–815页，doi：https://doi.org/10.1007/ s10992-017-9450-1。[7] Dov M. Gabbay 和 John Woods（2014年1月2日）：《逻辑史手册》。北荷兰出版社。[8] Martin Gebser、Roland Kaminski、Benjamin Kaufmann 和 Torsten Schaub（2012）：《实践中的答案集求解》。摩根与克莱普尔出版社。[9] Martin Gebser、Benjamin Kaufmann、Andr´e Neumann 和 Torsten Schaub（2007）：Clasp：一种冲突驱动答案集求解器。载于：第9届逻辑程序设计与非单调推理国际会议论文集，LPNMR’07，施普林格出版社，柏林、海德堡，第260–265页，doi：https://doi. org/10.1007/978-3-540-72200-7_23。[10] Martin Gebser、Torsten Schaub 和 Sven Thiele（2007）：GrinGo：一种用于答案集编程的新型实例化器。载于：第9届逻辑程序设计与非单调推理国际会议论文集，LPNMR’07，施普林格出版社，柏林、海德堡，第266–271页，doi：https://doi.org/10.1007/ 978-3-540-72200-7_24。[11] Hector Geffner 和 Judea Pearl（1992）：条件蕴涵：连接默认推理的两种方法。《人工智能》53(2–3)，第209–244页，doi：https://doi.org/10.1016/0004-3702(92)90071-5。[12] Laura Giordano、Daniele Theseider Dupr´e、Camillo Fiorentini、Alberto Momigliano 和 Alberto Pettorossi（2018）：SROEL中的可废止推理：从理性蕴涵到理性闭包。《Fundamenta Informaticae》，第161卷，第1–2期，161(1–2)，第135–161页，doi：https://doi.org/10.3233/FI-2018-1698。[13] Volker Halbach（2010年8月19日）：《逻辑手册》。牛津大学出版社。

[14] Frank van Harmelen、Frank van Harmelen、Vladimir Lifschitz 和 Bruce Porter（2007）：《知识表示手册》。爱思唯尔科学，圣迭戈，加利福尼亚州，美国。[15] Adam Kaliski（2020）：《KLM风格可废止蕴涵概述》。硕士论文。博士论文，开普敦大学理学院，隆德博斯，开普敦，7700。获取地址：http://hdl. handle.net/11427/32743。[16]Sarit Kraus、Daniel Lehmann 和 Menachem Magidor（2002）：非单调推理、偏好模型与累积逻辑。CoRR cs.AI/0202021，doi：https://doi.org/10.1016/ 0004-3702(90)90101-5。[17] Daniel Lehmann（1995）：默认推理的另一种视角。《数学与人工智能年刊》15(1)，第61–82页，doi：https://doi.org/10.1007/BF01535841。[18] Daniel Lehmann 和 Menachem Magidor（1992年5月）：条件知识库蕴涵什么？《人工智能杂志》第55卷第1期，第1–60页，doi：https://doi.org/10.1016/ 0004-3702(92)90041-U。[19] Vladimir Lifschitz（2002）：答案集编程与规划生成。《人工智能》138，第39–54页。获取地址：http://www.cs.utexas.edu/users/ai-lab?lif02。[20] Vladimir Lifschitz（2008）：稳定模型的十二种定义。《逻辑程序设计。ICLP 2008。计算机科学讲义》，第5366卷。施普林格，柏林、海德堡，第37–51页，doi：https://doi.org/10. 1007/978-3-540-89982-2_8。[21] Vladimir Lifschitz（2019）：《答案集编程》。施普林格，Cham，doi：https://doi.org/10.1007/ 978-3-030-24658-7。[22] Drew McDermott 和 Jon Doyle（1980）：非单调逻辑 I。《人工智能》13(1)，第41–72页，doi：https://doi.org/10.1016/0004-3702(80)90012-0。非单调逻辑特刊。[23] Ben-Ari Mordechai（2012）：《计算机科学数理逻辑》，第3版。施普林格，伦敦，doi：https://doi.org/10.1007/978-1-4471-4129-7。[24] C.E. Alchourr ´on、P. G¨a rdenfors 和 D Makinson（1985）：理论变更逻辑：部分交收缩与修正函数。《符号逻辑杂志》50，第510–530页，doi：https://doi.org/10.1007/ 978-3-319-20451-2_13。[25] Stuart Russell 和 Peter Norvig（2021）：《人工智能：现代方法》，全球版。培生德国。获取地址：https://elibrary.pearson.de/book/99.150005/9781292401171。[26] Karl Schlechta（2007）：非单调逻辑：偏好方法。载于 Dov M. Gabbay 和 John Woods 编：《逻辑中的多值与非单调转向》，《逻辑史手册》第8卷，北荷兰出版社，第451–516页，doi：https://doi.org/10.1016/S1874-5857(07)80010-0。[27] Yoav Shoham（1987）：非单调逻辑：意义与效用。载于第10届人工智能国际联合会议论文集——第1卷（意大利米兰）（IJCAI'87）。摩根·考夫曼出版社公司，旧金山，加利福尼亚州，美国，第388–393页。

A 附录

A.1 自然演绎系统

演绎系统用于构造证明，并逐步推导出重言式。我们在 ASP 编码中使用的系统是自然演绎。自然演绎系统允许从一组给定的前提出发，通过中间结论证明最终结论。

自然演绎系统既可靠又完备 [13]。形式上，若将 Γ定义为一组前提，而 ω是待证明的句子，那么当且仅当 Γ ⊨ ω时，可以从前提集合Γ 推导出 ω。此外，该系统是完备的，因为所有有效论证 ω（其中 Γ ⊨ ω）都可以通过系统中定义的证明步骤推导出来。

自然演绎中的证明以一个假设为起点。要得出 A →B 成立，我们先假设 A，然后由此得出 A → B。

在自然演绎中，处理每个联结词时有两个重要概念：引入规则和消去规则。引入规则规定了如何在证明中引入联结词，而消去规则规定了如何从证明中消去联结词。

传递性证明

给定前提：A →B，B →C。

$$\dfrac{\dfrac{A, A \to B}{B} \quad B \to C}{\dfrac{C}{A \to C}}$$

消去假设后，我们得到 A →C。

逆否命题证明

给定前提：A → B。

$$\dfrac{\dfrac{\neg B, \dfrac{A, A \to B}{B} \quad (\to E)}{\dfrac{\bot \quad (\neg E)}{\neg A} \quad (\neg I)} \neg B \to \neg A \quad (\to I)}$$

析取消去证明

给定前提：A ∨¬A, A →C, ¬A →C。

$$\dfrac{A \vee \neg A, \quad \dfrac{A, A \to C}{C}, \quad \dfrac{\neg A, \neg A \to C}{C}}{C}$$

恒等证明

给定前提：X →Y。

$$\dfrac{X \to Y}{X \to Y}$$

第3节的证明（BaseRank 和理性闭包的声明式方法）

命题 1. 设 i 为一个固定的非负整数，S 为满足 BaseRank 中所列规则（不含最小化指令）的某个答案集。进一步定义

$$\mathrm{ND}_N = \left\{ X \to Y \;\middle|\; \vec{\mathcal{K}}_i^{\mathcal{S}} \models X \to Y \right\}$$

and

$$\mathrm{ASP}_i = \left\{ X \to Y \;\middle|\; \texttt{derive(X,Y,i)} \in \mathcal{S} \right\}.$$

Then

$$\mathrm{ND}_i = \mathrm{ASP}_i.$$

证明。首先注意，由于我们的语言不过是文字之间的“如果—那么”关系，因此唯一的非平凡推理步骤是同一律、肯定前件和析取消去。

我们通过证明以下内容来继续：

$$\mathrm{ND}_N \subseteq \mathrm{ASP}_N \quad \text{and} \quad \mathrm{ASP}_N \subseteq \mathrm{ND}_N.$$

(1) $\mathrm{ND}_N \subseteq \mathrm{ASP}_N$ 设

$$X \to Y \in \mathrm{ND}_N.$$

根据定义，存在一个从 K_N 中的前提出发、推导出 X →Y 的自然演绎证明。我们将此证明记作 Π，其长度为 n。我们证明 derive(X ,Y,N) ∈ASP N。我们对演绎证明 Π 的长度进行结构归纳。

基础情形（n=*0*）：如果 n=0，那么这只是恒等规则，并且 X →Y 是 K_N 中的一个前提，根据规则

```
derive(X,Y,N) :- rank(m_implication(X,Y),N) , derive(X,X,N).
```

被加入答案集后，便可推出 derive(X,Y,N) ∈S，X →Y ∈答案集编程$_N$。

归纳假设（n=k）：对于任意长度为 k 的自然演绎证明 Π，若 X →Y ∈ND_N，则 X →Y ∈答案集编程$_N$

归纳步骤（n =k +1）。设 Π 是一个长度为 k +1、证明 X →Y 的自然演绎证明。我们考虑以下情况：

恒等律：在此情况下，最后一步是“重申前提 X →Y。”用自然演绎符号表示：

$$\frac{}{X \to Y \vdash X \to Y} \quad \Longrightarrow \quad X \to Y.$$

由于 X →Y ∈K_N，相应的 ASP 子句

```
derive(X,Y,N) :- rank(m_implication(X,Y),N)
```

fires，因此推导出 derive(X ,Y,N) ∈S。因此 X →Y ∈答案集编程$_N$。

肯定前件 / 传递性：

$$\frac{X \to Z \quad Z \to Y}{X \to Y}$$

根据归纳假设，X $\to$Z 和 Z $\to$Y 均推出 derive(X , Z, N) $\in$S 和 derive(Z,Y,N) $\in$S。于是，ASP 规则

$$\texttt{derive}(X,Y,N) \text{ :- } \texttt{derive}(X,Z,N), \texttt{derive}(Z,Y,N)$$

适用，得到 derive(X ,Y,N)。因此，X $\to$Y $\in$答案集编程$_N$。

析取消去：

$$\frac{[X] \vdash Y \quad [\neg X] \vdash Y}{Y}$$

根据归纳假设，我们有 derive(X ,Y,N) $\in$S 和 derive(–X ,Y,N) $\in$S。然后，ASP 的析取消去子句

$$\texttt{derive}(\top,Y,N) \text{ :- } \texttt{derive}(X,Y,N), \texttt{derive}(-X,Y,N)$$

将 derive($\top$,Y, N) 加入答案集，即 Y $\in$答案集编程$_N$。

在所有情况下，derive(X ,Y,N)（或 derive($\top$,Y, N)）都会出现在 S 中。因此 ND$_N$ $\subseteq$ASP$_N$。

(2) ASP$_N$ $\subseteq$ND$_N$。

反过来，假设 derive(X ,Y,N) $\in$S。检查 BaseRank 编码可知，derive(X ,Y,N) 的推导方式恰好有三种：

1. derive(X,Y,N) 由 rank(m implications(X, Y), N) 直接推导得出，对应代码清单第 10 行 2.
2. deriv 根据清单 2 第 11 行，由 derive(X, Z, N) 和 derive(Z, Y, N) 推导出 e(X,Y,N)。
3. 根据清单 2 第 12 行，由 derive(X, Z, N) 和 derive(–X, Z, N) 推导出 derive(top,Z,N)。

我们通过对 derive 谓词 derive(X,Y,N) 的 ASP 推导深度进行归纳，重构 X $\to$Y 的自然演绎证明。

基础情形 (n=*0*)：如果 derive 谓词的长度为 0，那么 derive(X ,Y,N) 会根据 rank(m implication(X ,Y), N) 加入答案集 S，因此 X $\to$Y $\in$K$_N$，且 X $\to$Y $\in$ND$_N$。
归纳假设 (n=k)：对于长度为 n=k 的情形，若对每个 derive(X,Y,N) $\in$S，且对每个 X $\to$Y $\in$ASP$_N$，则 X $\to$Y $\in$ND$_N$。
归纳步骤 (n=k+*1*)：

- 如果 derive(X, Y, N) 是由 rank(m implications(X, Y), N) 得到的，那么这就是基础情形，因此 X $\to$Y $\in$ND$_N$。
- 如果 derive(X ,Y, N) 是由 derive(X , Z, N) 和 rank(m implication(Z,Y), N) 得到的。根据归纳假设，X $\to$Z 且 Z $\to$Y $\in$ND$_N$，因此，通过自然演绎，我们可以证明 X $\to$Y $\in$ND$_N$。
- 如果 derive(X ,Y,N) 和 derive(–X ,Y, N) 成立，那么 derive($\top$,Y, N) $\in$S；根据归纳假设，X $\to$ Y 且 X $\to \neg$Y $\in$ND$_N$，通过自然演绎 $\top \to$Y $\in$ND$_N$。

在每种情形下，我们都会为 X $\to$Y 构造一个有效的自然演绎子证明。因此，答案集编程$_N$ $\subseteq$ ND$_N$。

结合 (1) 和 (2)，即完成了 ND$_N$ =ASP$_N$ 的证明。 □

命题 2。给定一个知识库 K。对于编码(K)∪BaseRank 的给定答案集 S，Ð→对于任意 $i \geq 0$，以及对于任意 $A \mid\sim B \in K_i^S$，$\bigcup_{j=i}^{\infty} K_j^S \not\models \neg A$。

Ð→证明。反设对于某个 $A \mid\sim B \in K_i^S$，$\bigcup_{j=i}^{\infty} K_j^S \models \neg A$。这意味着每个满足 $\bigcup_{j=i}^{\infty} K_j^S$ 的解释也满足 ¬A。根据 1 中 ASP 规则与自然演绎规则之间的等价关系，我们可以看出，这意味着 derive(top –A, j) 被加入答案集 S 中。由此可知，使用自然演绎规则，$\bigcup_{j=i}^{\infty} K_j^S \models (A \rightarrow -)$ 成立。然而，由于 $A \mid\sim B \in K_i^S$，这意味着 derive(A, B, j) ∈S。由此可知，derive(A, B, j), derive(top –A, j) ∈S。根据清单 2 第 15 行的完整性约束，这不成立。因此 $\bigcup_{j=i}^{\infty} K_j^S \not\models \neg A$

□

命题 3. 给定一个知识库 K，其 BaseRank(K) 为 =(R0, . . . ,R_n, R_∞)。对于 K 的任意划分 (R'_0, . . . , R'_m, R_∞)，若对每个 R'_i，都存在一个 j，使得 $R'_i \subseteq R_j$，则编码(K)∪baseRank 存在一个答案集 S，使得对于每个 $i \geq 0$，$K_i^S = R'_i$。

证明. 给定 BaseRank 划分 (R0, . . . ,R_n, R_∞) 的一个细化 (R'_0, . . . , R'_m, R_∞)，其中对于每个 R'_i，都存在某个 j，使得 $R'_i \subseteq R_j$。给定 R'_i，我们可以如下构造一个答案集 S：

- 对于每个可废止蕴含 $A \mid\sim B \in R_i$，将原子 rank(m implication(A,B), i) 纳入 S。
- 对于每个经典蕴含式 A →B ∈K，根据编码，将原子 rank(m implication(A,B), inf) 纳入答案集 S。
- 纳入根据编码中的推理和传播规则可由这些排序推导出的所有原子，derive/3、infer/2。

给定这个答案集 S，我们将证明它满足由 BaseRank 定义的完整性约束。

- 第 14 行的完整性约束：
  该约束禁止在秩 N 上同时推导出 X →Y 和 X →–Y。原始的 BaseR-ank 划分 (R0, . . . ,R_n, R_∞) 按构造是一致的。由于每个 R'_i都是某个 R_j 的子集，细化不会引入矛盾。因此，第 18 行成立。
- 第 15—16 行的完整性约束：
  这些约束禁止在层级 N 上同时推出 –X，其中 X 是重言式。同样，由于原始划分是一致的，其任何子集也都是一致的。因此，该完整性约束成立。
- 第 17 行的完整性约束：
  该构造明确地为知识库 K 中的每个蕴含式指定且仅指定一个秩，因此该完整性约束成立。

因此，对每个 i，$K_i^S = R'_i$。 □

命题 4。给定一个知识库 K，其 BaseRank(K) =(R0, . . . , R_n, R_∞)。对于编码(K) ∪baseRank∪ baseRankMin 的答案集 S，$K_i^S = R_i$ 对 $i \geq 0$。

证明。假设某个可废止条件句 $\delta = \alpha \mid\sim \beta$ 被置于最小答案集 S 中的第 j >i 级，而 $\delta \in R_i$ 位于 BaseRank(K) 中。由于 BaseRank 将 δ 分配到最低的一致级别，我们知道，将 δ置于第 i 级仍会得到一个一致解释。

因此，我们可以构造一个替代答案集 S′，其中将 δ重新分配到第 i 级，而其他所有分配保持不变。在 #minimize 指令下，这个新的答案集会具有严格更低的代价，这与 S 是最小的这一假设相矛盾。

反之，假设存在一个可废止条件句 $\delta \in R_i$，它属于 BaseRank(K)，但未在 S 中出现在等级 i。于是，它要么被置于更高等级（这与上文矛盾），要么根本未被分配等级（而选择规则禁止这种情况）。因此，δ必须出现在 K_i^S 中。

由此可知，最小答案集必须将每个可废止条件句分配到与 BaseRank(K) 中相同的等级。因此，对于所有 i，$K_i^S = R_i$。□

命题 5。给定某个知识库 K 和查询 A|~ B，若 entailed(true)∈S，其中 S 是最小答案集，则 $K \mid\approx_{RC} A \mid\sim B$。

证明。给定 entailed(true) ∈ S，根据理性闭包（RC）中 entailed 的定义，这意味着 inference(A, B) ∈ S。Ð→为证明 $K \mid\approx_{RC}$，我们将使用 Π 构造相应的 BaseRank 模型。令 $\bigcup_{j=n}^{\infty} K_j^S$ = {A →B | rank(m implication(A,B),i) ∈ S,guess(N) ∈ S}，其中对于 guess(N) ∈ S，有 n = N。根据命题 Ð→ Ð→3 $\bigcup_{j=n}^{\infty} K_j^S$ 映射到理性闭包（RC）所使用的相应最小基排序模型，$\bigcup_{j=n}^{\infty} K_j^S = R_\infty \cup R$。RC 中列出的推理规则与用于建模蕴涵的规则完全相同，因此 $R_\infty \cup R \models A \rightarrow B$，从而得到 $K \mid\approx_{RC} A \rightarrow B$ □

命题 6. 给定某个知识库 K 和查询 A|~ B，若 $K \mid\approx_{RC} A \mid\sim B$，则存在一个 Π 的最小答案集 S，使得 entailed(true)∈S。

证明. $K \mid\approx_{RC} A \mid\sim B$，因此存在 BaseRank 模型的某个划分 R′，其中 $R_\infty \cup R' \models A \rightarrow B$。命题 3 保证存在一个与该排序相对应的答案集 S。使用 ASP 编码中定义的推理规则，可推导出 inference(A, B) ∈ S，且 query(A,B) ∈ S，从而得到 entailed(true) ∈ S。这是因为所定义的推理规则遵循自然演绎风格的推导规则，因此是可靠且完备的。

□