航空应用中的混合 MKNF：用法与启发式方法

Arun RaveendranNair Sheela FlorenceDeGrancey ChristopheRey克莱蒙奥弗涅大学、LIMOS 泰雷兹 克莱蒙奥弗涅大学、LIMOS 泰雷兹 法国国家科学研究中心，法国

Victor Charpenay圣艾蒂安矿业学院、LIMOS 法国国家科学研究中心，法国

在航空应用中部署知识表示与推理技术面临两项主要挑战：实现足够的表达能力以刻画复杂的领域知识，以及高效执行推理任务，同时尽量减少内存使用和计算开销。实现所需表达能力的一项有效策略，是整合知识表示领域的两个基本概念：规则与本体。本研究采用成熟的知识表示语言混合 MKNF，借助其语义和查询回答能力，实现规则与本体的无缝集成。我们通过一项具体案例研究评估混合 MKNF，以考察其在航空领域的适用性。我们确定了开发航空应用所必需的额外表达能力特性，并提出一套启发式方法，以支持将这些特性整合到混合 MKNF 框架中。

1 引言

航空业被公认为安全关键领域，系统故障可能给利益相关方带来严重的安全影响。因此，集成到该行业中的任何软件系统都必须具备可靠性、鲁棒性和可解释性等可信属性，以防止此类后果。此外，该领域的许多软件系统运行在微控制器等资源受限硬件上。在此背景下，知识表示与推理（KRR）展现出巨大潜力，因为它能够使系统做出透明且可解释的决策。在领域专家的协助下，人们使用本体等形式化表示语言，对航空应用的专门知识进行了表述。这些知识涵盖关键的运行信息，并可与实时数据相结合，通过逻辑推理促进及时决策。然而，将知识表示与推理应用于航空领域也面临若干重大挑战，包括确保表达能力足以涵盖所有复杂的领域知识，以及在计算资源有限、延迟极低的条件下有效执行必要的推理任务。

解决表达能力挑战的一种有效方法是整合规则与本体，这一领域已得到充分研究。本体通常基于描述逻辑（DL），采用开放世界假设（OWA），即信息缺失并不等同于否定。它们通过概念及其关系来封装一般领域知识 [5]。与之相对，逻辑编程（LP）规则采用封闭世界假设（CWA），假定凡是未被确认为真的内容均为假。这些规则通过如果-那么关系来表示知识 [42]。由于二者的特性互补，将描述逻辑与逻辑程序规则结合起来，可以增强由此形成的知识表示与推理语言的表达能力；这种语言称为混合知识库。各种

M. Gebser、D. Inclezan、F. Ricca、M. Carro 和 M. Truszczynski：© A.R.N. Sheela、F. De Grancey、C. Rey 和 V. Charpenay 第41届逻辑编程国际会议（ICLP 2025） 本文采用《知识共享署名许可协议》授权。EPTCS 439，2026，第349–366页，doi:10.4204/EPTCS.439.24

创建混合知识库的方法已被提出，每种方法都有其独特的优点与权衡 [28], [10], [13], [37]。在本研究中，我们探讨了混合 MKNF 语言，它提供了一种统一逻辑，可整合描述逻辑与逻辑程序的语义 [38]。针对混合 MKNF，已有研究提出了良基语义[29]和答案集语义 [39] 。鉴于复杂查询应答是航空领域的主要推理任务，且良基语义在处理该任务时具有更高的计算效率，我们在本方法中采用了这种语义。

本研究的主要贡献在于探讨良基语义下的混合 MKNF 及其在航空领域的实现，并通过全面评估考察其实际适用性。为此，我们开展了一系列实验，重点关注预处理延迟、查询应答性能和资源利用率。我们还指出，良基语义下的混合 MKNF 并不天然支持建模复杂航空场景所需的所有表达能力特性。为解决这一局限，我们借助一个半真实用例定义了一组核心表达需求，并提出启发式方法，展示如何将这些特性有效集成到混合 MKNF 框架中。此外，我们还探讨了未来的研究方向，以进一步增强混合 MKNF 在航空领域中的特性。

第 2 节介绍航空领域中的一个半真实用例，以说明本研究讨论的概念。第 3 节概述混合 MKNF 框架，介绍 NoHr 推理器，并说明其在这一背景下的高效用法。第 4 节提出通过经典否定和完整性约束增强表达能力的启发式方法。第 5 节介绍这些特性的实现。第 6 节回顾相关工作，并阐述选择混合 MKNF 的理由。最后，第 7 节总结本研究并展望未来方向。

2 用例

为解决航空领域中需要整合描述逻辑与逻辑程序规则的用例，我们提出了 NOTAM 感知推理系统（NARS）。起飞前，飞行员需要手动审阅所有航行通告（NOTAM）消息。这些官方通知用于传达空域状态、环境条件、跑道关闭及其他相关因素的时效性变化。审阅旨在确保所有航行通告均与飞行计划一致。以下是一个经过预处理的航行通告和飞行计划示例：

```
Notam001:hasaffectedByrw1;Flight:AF123:hasDepartureAirportLFBO;:hasRun
wayStatus:关闭;:hasDestinationAirportLFPG;:hasAirportLFPG;:hasDepartureTime"
2025-04-18T06:00:00Z";:haStartTime"2025-04-18T06:00:00Z";:hasArrivalTime"
2025-04-18T07:30:00Z";:hasValidEndTime"2025-04-18T12:00:00Z";:hasIssuedBy:ATC_
Agency1.:landingRunway rw1;:hasAlternateAirportex:LFOB.
```

例如，如果一则航行通告指出，在航班的计划到达时间，指定跑道不可用，飞行员就必须申请修订飞行计划或延误航班。无论由飞行员还是支持人员手动分析每则航行通告以提取此类信息并作出适当决策，都既耗时又容易出错。为应对这一挑战，我们提出了 NARS，通过根据相关航行通告自动提供建议来协助飞行员。NARS 执行的一些关键操作包括：

1. 决策应仅依据有效航行通告作出。任何因缺少关键数据或不同主管部门提供冲突信息而无效或相互矛盾的航行通告，都必须予以忽略，并应通知飞行员其无效性。

2. 如果发现当前备降机场因航行通告中报告的约束而无法使用，则有必要建议指定新的备降机场。3. 在紧急着陆情况下，应推荐附近的机场及符合航空器安全着陆所需参数的跑道。其中一个关键参数是跑道长度，必须大于航空器所需的着陆距离。

NARS 的知识库可以使用混合 MKNF 构建。在描述逻辑方面，可以表示静态且具有层次结构的领域知识，例如机场基础设施、不同类型的跑道以及空域区域。这种结构化知识提供了一套形式化词汇，用于描述实体及其关系。例如，声明“跑道是机场的一部分”或“跑道配备仪表着陆系统（*ILS*）”。相比之下，逻辑程序处理动态知识，这些知识以条件式如果—那么规则编码。例如，一条规则可以规定：“如果跑道在预定着陆时间段内关闭，则应建议使用备用跑道”，或“如果机场的关键设备停止运行，则必须通知飞行运行团队”。具体实例，例如单条航行通告或飞行计划，可以形式化为断言。

3 混合 MKNF

混合 MKNF（如 [38], 中所引入）基于最小知识与否定即失败逻辑（MKNF）。该逻辑是一阶逻辑的扩展，包含两个模态算子，即 **K** 和**not** ，最初由 [34] 描述。**K** 算子表示系统已知的知识，而**not** 算子表示系统未知的知识，其作用类似于逻辑程序中的默认否定，并支持封闭世界推理。在良基语义下，混合 MKNF 定义为 $KB = (O,P)$，其中 O 是描述逻辑本体，P 是由以下形式的 MKNF 规则构成的集合 [30]:

$$\mathbf{K}\, H \leftarrow \mathbf{K}\, A_1, \ldots, \mathbf{K}\, A_m, \mathbf{not}\, B_1, \ldots, \mathbf{not}\, B_n. \tag{1}$$

其中，H、A_i（$1 \leq i \leq m$）以及 B_j（$1 \leq j \leq n$）均为一阶原子。规则（1）可读作：“如果所有 A_i均已知成立，且所有 B_j 均未知是否成立，那么 H 已知成立”。我们考虑该用例中的一个简化场景：根据航行通告数据推断机场和跑道的可用性，并使用混合 MKNF 知识库推导飞行运行建议，$KB = (O,P)$:

$$\text{机场} \sqcap \forall \text{具有跑道.CldRwy} \sqsubseteq \text{关闭机场} \tag{2}$$

$$\text{机场} \sqcap \exists \text{具有跑道.OpnRwy} \sqsubseteq \text{开放机场} \tag{3}$$

$$\text{OpnRwy} \sqsubseteq \text{Rwy} \tag{4}$$

$$\text{CldRwy} \sqcap \text{OpnRwy} \sqsubseteq \bot \tag{5}$$

$$\mathbf{K}\,\text{CldRwy}(X) \leftarrow \mathbf{K}\,\text{航行通告}(X),\ \mathbf{K}\,\text{affdBy}(X,\ Y),\ \mathbf{K}\,\text{Rwy}(Y),\ \mathbf{K}\,\text{opStat}(X,\ \text{关闭})。 \tag{6}$$

$$\mathbf{K}\,\text{OpnRwy}(X) \leftarrow \mathbf{K}\,\text{Rwy}(X),\ \text{不}\ \text{CldRwy}(X)。 \tag{7}$$

$$\mathbf{K}\,\text{建议延误}(X) \leftarrow \mathbf{K}\,\text{航班}(X),\ \mathbf{K}\,\text{具有出发机场}(X,\ Y),\ \mathbf{not}\,\text{开放机场}(Y)。 \tag{8}$$

规则（7）推断*rwy(rw1)* 默认情况下被视为*OpnRwy(rw1)* ，除非有明确证据表明*rwy(rw1)* 是*CldRwy(rw1)*。当存在影响特定跑道的航行通告时，可以推导出谓词*CldRwy(rw1)* ，如规则（6）所述。这可以用**not** 算子表示：**not***CldRwy(X)* 表示尚未知晓某条跑道已关闭。因此，在没有表明跑道关闭的航行通告消息时，可以推断跑道仍可运行。此外，断言以下附加事实：*hasRwy(lfbo,rw1)* 和*Airport(lfbo)*，并结合描述逻辑公理（3）应用规则（7），即可得出*Airport(lfbo)* 符合*OpnAirport*的条件。

混合 MKNF 的语义可以通过将知识库转换为带有模态算子的 一阶公式来形式化 [30]。考虑$KB_1 = (O_1, P_1)$，其中O_1 包含描述逻辑公理（4）和断言*Rwy(rw1)*，而P_1 包含规则（7）。KB_1，的一阶翻译记作 $\pi(KB_1)$，其形式为$\mathbf{K}\pi(O_1) \wedge \pi(P_1)$：

$$\forall X(\mathbf{K}\ \mathrm{Rwy}(X) \leftarrow \mathbf{K}\ \mathrm{OpnRwy}(X)). \tag{9}$$

$$\forall X(\mathbf{K}\ \mathrm{OpnRwy}(X) \leftarrow \mathbf{K}\ \mathrm{Rwy}(X),\ \mathbf{not}\ \mathrm{ClsRwy}(X)). \tag{10}$$

MKNF 公式的模型定义为满足该公式的解释的极大集合，且该集合的任何真超集都不满足该公式 [34]。知识库$_1$ 可满足，当且仅当存在一个 MKNF 模型满足 π(知识库$_1$)，并且 知识库$_1$ 蕴含一阶公式 ψ，记作 知识库$_1 \models \psi$ 当且仅当 π(知识库$_1$) $\models_{MKNF} \psi$。 π(知识库$_1$) 的语义可通过答案集 [39] 或良基语义 [30]来解释。在航空领域，良基语义至关重要，因为它能建立一个包含真、假和未定义 原子的单一三值模型，从而便于高效地进行自顶向下查询应答。为了解释混合 MKNF 的良基语义，三值 MKNF 解释定义为一对 (M, N)，其中 M 和N 均为一阶解释的集合，且$N \subseteq M$。为避免在集成描述逻辑与逻辑程序时产生意外后果，采用标准名称假设（SNA），确保每个常量唯一标识一个个体，并将等式显式地作为同余关系处理 [39]。

带有模态算子的 一阶原子相对于三值MKNF模型的真值评估如下。若元组 $(t_1^I, \ldots, t_n^I)$ 属于一阶解释 I 中谓词 P 的解释，记作 P^I；形式上，即 $(t_1^I, \ldots, t_n^I) \in P^I$，则原子公式 P(t1,. ..,t$_n$) 在一阶解释 I 下评估为真。否则，当 $(t_1^I, \ldots, t_n^I) \notin P^I$ 时，该公式评估为假。若 φ_2 的真值相对于序关系 f $<$ u $<$t 大于或等于 φ_1 的真值，则蕴含 $\varphi_1 \supset \varphi_2$ 评估为真；否则评估为假。若 φ 在 M 中的所有解释下都评估为真，则 K φ 的真值为真；若 φ 在 N $\subseteq$M 中的某个解释下评估为假，则其真值为假；否则为未定义。同样，若 not φ 在 N $\subseteq$M 中的某个解释下为假，则 φ 的真值为真；若 φ 在 M 中的所有解释下都为真，则其真值为假；否则为未定义 [30]。

MKNF解释对（M,N）满足闭MKNF公式 φ，记作（M,N）$\models \varphi$，当且仅当（M,N）(φ) = t，这意味着在此解释下，φ中的所有规则都为真。若 MKNF解释对（M,N）满足公式 φ的两个关键条件，则称其为该公式的三值MKNF模型。首先，它必须满足公式 φ本身。其次，它必须是极小的，即不能扩展M 或N，使之成为仍满足 φ的更大的解释对。KB_1的唯一三值模型为：*M = {{Rwy(rw1), OpnRwy(rw1)}, {Rwy(rw1), OpnRwy(rw1), ClsRwy(rw1)}}, N = {{Rwy(rw1),OpnRwy(rw1)}, {Rwy(rw1), OpnRwy(rw1),ClsRwy(rw1)}}*。

查询回答： 交替不动点方法为计算混合 MKNF 知识库 [30] 的三值良基模型提供了一种自底向上策略。最初，必须使用所提出的自底向上方法计算该模型。计算完成后，可以将其存储在数据库中，以便进行查询。然而，对于大型知识库而言，这种方法并不实用，尤其是在只有部分特定信息与回答查询相关时。为解决这一问题，研究人员提出了一种名为 SLG(O) 归结的查询回答算法，该算法相对于良基混合MKNF模型而言既可靠又完备。SLG(O) 归结是一种目标导向查询回答方法，它采用 SLG归结来管理逻辑程序规则，其中 O 作为预言机，负责处理与描述逻辑组件 [1] 相关的查询。归结过程从一个查询开始，将其与规则头进行匹配，随后根据该规则的体生成新的子目标。当某个子目标对应于描述逻辑原子时，系统会调用描述逻辑预言机来确定该原子是否由描述逻辑蕴涵

该原子是否由描述逻辑蕴涵。在整个过程中，系统使用表格化存储中间结果，从而避免无限循环并确保终止。在实现 SLG(O) 归结之前，必须执行知识库加倍步骤，使该算法与良基 MKNF 模型的自底向上计算保持一致 [1]。

考虑 知识库 $=(O,P)$ 是一个混合 MKNF 知识库，并在 [1] 为每个出现在知识库中的谓词 L 引入新谓词 L^d和 NL，然后定义：

1. 通过将O 中的每个概念或角色L 替换为L^d来得到O^d；并且
2. 通过将P 中出现的每条形式为 (1) 的规则转换为两条规则来得到P^d：

i K H $\leftarrow$K A1, . .. , K A$_n$, not B$_1^d$,. .. , not B$_m^d$，以及以下二者之一：ii(a) K H $^d \leftarrow$K A$_1^d$, . .. , K A$_n^d$, not B1,. .. , not B$_m$, not NH（若 H 是描述逻辑概念或角色）；或者 ii(b) K H$^d \leftarrow$K A$_1^d$, . .. , K A$_n^d$, not B1,. .. , not B$_m$（其他情况）。

加倍的混合 MKNF 知识库定义为 知识库$^d=(O,O^d,P^d)$。在此变换中，知识库 知识库 L 中的每个原始谓词都与两个谓词配对：L 和 L^d，其中L^d表示L的非假性。此外，还引入一个新谓词 NL，它在语义上等价于经典否定 $\neg L$。该谓词可以阻止在本体中蕴含 $\neg L$ 时推导出L^d。为此，在每条以L^d 为头部的规则体中加入**not** NL，确保只有在本体不支持 $\neg L$ 时才能推断L^d。这一变换过程称为半负变换。它有助于验证 MKNF 一致性，方法是识别L 为真而L^d 不为真的情况。在这种情况下，由于L 为真而L^d 为假，因此 $\neg L$ 可由描述逻辑本体 O 蕴含，从而导致不一致。

3.1 NoHr 推理器

NoHr 推理器采用 SLG(O) 归结，主要是将加倍的混合 MKNF 知识库转换为一组等价的规则 [27]。这种方法通过使用 *XSB Prolog* 进行查询回答，从而实现高效推理。具体而言，加倍的混合 MKNF 知识库表示为 $KB=(O,O^d,P^d)$，将其转换为语义等价的形式：$KB=(/0,P^d\cup P^O)$,其中 P^d 是加倍逻辑程序，而P^O 是转换为规则后再经过知识库加倍的描述逻辑本体。NoHr 推理器采用两种不同的转换方法，将描述逻辑转换为规则。首先，使用分类算法推导包含关系，再将其转换为逻辑程序规则 [26]。在分类过程中，NoHr 集成了三种现有的描述逻辑推理器：ELK、HermiT 和 Konclude，其中 HermiT 和 Konclude 支持表达能力更强的描述逻辑构造子。其次，对于 QL 配置，采用直接转换方法，将描述逻辑公理转换为逻辑程序，而无需预先进行分类 [7]。我们选择重点介绍使用 HermiT 推理器的基于分类的转换，因为与 QL 相比，它支持表达能力更强的描述逻辑构造子。图 1 展示了 NoHr 使用基于分类的转换器时涉及的各个步骤。

考虑描述逻辑公理 (4) 和 (5)，根据 NoHr 命题将其转换为规则，然后进行知识库加倍和半负变换（图 1 中的步骤 1 至步骤 6，详情请参阅 [26]）：

$$\mathrm{Rwy}(X)\leftarrow\mathrm{OpnRwy}(X). \quad (11)$$

$$\mathrm{Rwy}^d(X)\leftarrow\mathrm{OpnRwy}^d(X),\mathbf{not}\ \mathrm{NRwy}(X). \quad (12)$$

$$\mathrm{NCldRwy}(X)\leftarrow\mathrm{OpnRwy}(X). \quad (13)$$

$$\mathrm{NOpnRwy}(X)\leftarrow\mathrm{CldRwy}(X). \quad (14)$$

NoHr 推理器通过知识库加倍与半负变换的结合，表现出次协调行为。这种方法能够在知识库的部分内容存在

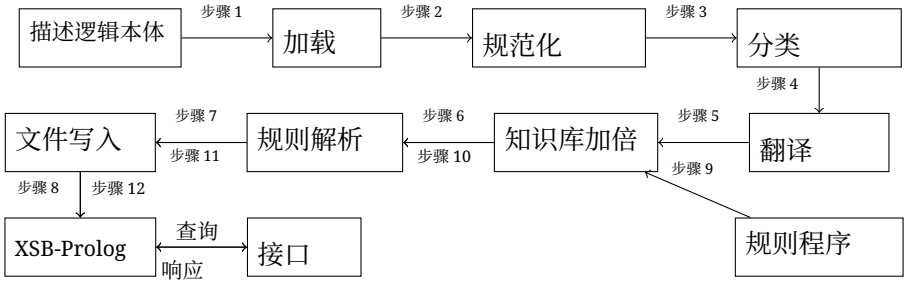


图 1：NoHr 推理器在 EL 配置文件中的不同步骤

矛盾时，通过隔离并聚焦于一致片段，使推理得以持续。在混合 MKNF 知识库中，当原子 H 及其经典否定 $\neg H$ 均可被推导时，就会产生矛盾。例如，考虑添加断言 *OpnRwy(rw1)*和 *cldRwy(rw1)*。根据描述逻辑公理 (5)，这些断言相互矛盾，因为跑道不可能同时处于可运行状态和关闭状态。应用知识库加倍与半负变换（图 1 中的步骤 1 至步骤 6）后，这些断言将转换如下：

$$\text{OpnRwy(跑道1).} \tag{15}$$

$$\text{CldRwy}(Rwy1)。 \tag{16}$$

$$\text{OpnRwy}^d(rw1) \leftarrow \textbf{not}\ \text{NOpnRwy}(rw1). \tag{17}$$

$$\text{CldRwy}^d(rw1) \leftarrow \text{非NCldRwy}(rw1)。 \tag{18}$$

利用规则 (11)，我们可以推断出*Rwy(rw1)*。然而，根据规则 (12)，我们无法推断出 $\text{Rwy}^d(rw1)$，因为*NOpnRwy(rw1)* 成立，这一点已由规则 (14)确立。只有当H 和H^d 均为真时，文字H 才被视为一致为真。在此情况下，尽管 Rwy(*rw*1)为真，但 Rwy^d(*rw*1)为假。因此，该情况被视为不一致。consequently，Rwy(*rw*1) 被排除在一致查询答案集合之外。通过采用这种句法方法，NoHr 推理器能够有效检测矛盾，确保响应查询时仅传达一致的信息，其中一个原子及其重复项被视为真。

下面，我们概述 NoHr 推理器所采用的一致性解决技术。考虑如下形式的合取查询：

$$q(X) \leftarrow A_1, A_2, \ldots, A_n, \textbf{not}\ B_1, \textbf{not}\ B_2, \ldots, \textbf{not}\ B_m \tag{19}$$

其中，$A_1, A2, \ldots, A_n, \textbf{not}\, B_1, \textbf{not}\, B2, \ldots, \textbf{not}\, B_m$表示在混合 MKNF 框架中所查询谓词的合取。为应对容矛盾特性，该查询被转换为两个不同的查询：

$$q(X) \leftarrow A_1, A_2, \ldots, A_n, \textbf{not}\ B_1^d, \textbf{not}\ B_2^d, \ldots, \textbf{not}\ B_m^d \tag{20}$$

$$q^d(X) \leftarrow A_1^d, A_2^d, \ldots, A_n^d, \textbf{not}\ B_1, \textbf{not}\ B_2, \ldots, \textbf{not}\ B_m \tag{21}$$

通过使用以下语句查询 Prolog 引擎，可以识别一致答案：$q(X) \wedge q^d(X)$；而矛盾或不一致的答案则可通过以下语句确定：$q(X) \wedge \textbf{not}\, q^d(X)$。这种方法使系统能够通过明确区分一致结论与矛盾结论来管理矛盾，从而确保矛盾的存在不会削弱推理过程的可靠性。

评估：要将 NoHr 推理器集成到航空应用中，必须对其进行全面评估，以检验其在真实运行环境中的性能。评估应包括确定 NoHr 支持的描述逻辑构造子，这对参与本体建模的知识工程师尤其有益。此外，还必须测量

描述逻辑本体和逻辑程序各预处理阶段所需的时间。这决定了在任一组件中引入新知识或系统重启后，推理器需要多久才能开始运行。此外，考察 NoHr 推理器查询响应的延迟也至关重要；鉴于航空领域具有安全关键性，查询回答延误可能带来重大风险，因此推理器必须在一秒内给出结果。另一个重要参数是资源利用率；如前所述，推理器可能运行在资源有限的受限设备上。因此，推理器不应占用过多内存或 CPU 资源。

为分析描述逻辑组件的预处理延迟，我们使用了 OWL2Bench 基准测试。该基准测试是 LUBM 数据集的扩展，旨在生成大小各异、符合四种 OWL 配置文件（EL、QL、RL 和 DL）的描述逻辑本体 [43]。我们使用该基准测试，通过 NoHr 推理器评估描述逻辑部分。对于逻辑程序组件，我们采用了 NoHr 推理器提供的基准测试，该测试会根据关联的描述逻辑本体生成数量为 'n' 的规则。对于使用 NoHr 推理器提供的规则基准测试、从 OWL2Benchmark 生成的每个描述逻辑本体，生成的规则数量分别为 10、100、1,000、10,000 和 25,000。规则基准测试可在此处获取：https://github.com/NoHRReasoner/NoHR/tree/master/nohr-benchmark；OWL2Bench 可在此处获取：https://github.com/kracr/owl2bench。对于查询评估，我们使用了 LUBM 基准测试提供的一组查询，并稍作修改，使其与 OWL2Bench 基准测试生成的描述逻辑本体的结构和词汇相匹配。随后，我们将这些查询转换为与 Prolog 引擎兼容的等价查询。查询可在此处查阅：https://swat.cse.lehigh.edu/projects/lubm/queries-sparql.txt。为评估内存消耗和 CPU 利用率，我们使用了包含不同数量公理的本体文件。

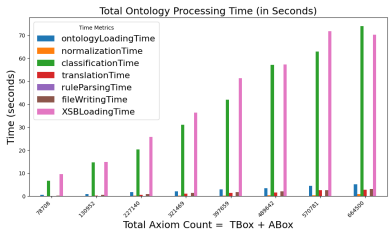


图 2：本体处理

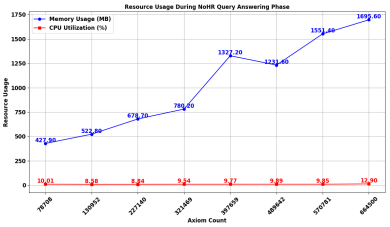


图 3：资源消耗

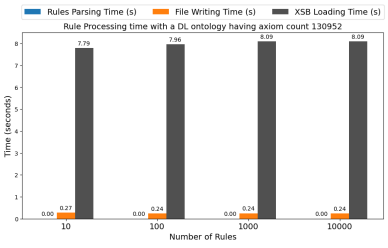


图 4：规则处理（78708 条描述逻辑公理）

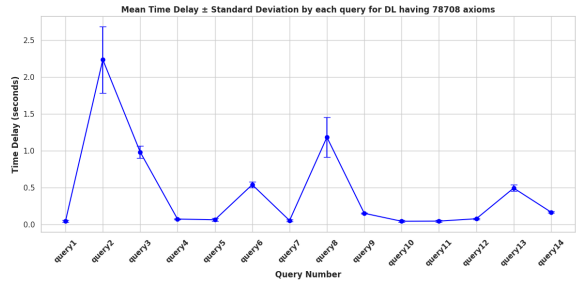


图 5：具有 78708 条公理的描述逻辑中 14 个查询的平均查询响应时间

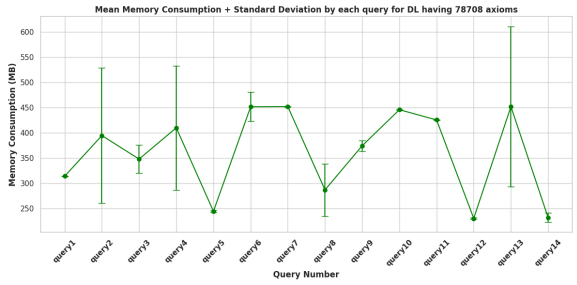


图 6：具有 78,708 条公理的描述逻辑中 14 个查询各自的平均内存消耗。

图 7：NoHr 推理器的评估结果

评估总结：在本体预处理阶段，大量时间用于本体分类和 XSB 加载，这两项工作都会受到描述逻辑公理数量的显著影响（图 2）。在规则处理阶段，XSB 加载时间与转换后的本体大小成正比（图 4）。在预处理过程中，我们还观察到内存使用量和 CPU 利用率上升，二者均与知识库的大小相关（图 3）。预处理完成后，使用 XSB-Prolog 进行查询回答的效率很高，尤其是对于根据特定条件筛选结果、仅返回少量答案的选择性查询（图 5）。然而，对于需要通过递归规则进行推理的查询，我们观察到其性能明显下降。某些描述逻辑构造子（例如等价关系和传递属性）会被转换为递归规则，从而显著影响性能。此外，在查询回答阶段，内存使用量仍然较高（图 6）。

3.2 用法

推理器过高的内存和 CPU 消耗，会给航空应用带来重大挑战，因为实时性能和资源效率至关重要。在此背景下，由于评估过程中观察到较高的内存需求，NoHr 推理器在处理较大的知识库时会面临问题。NoHr 通过一系列转换算法，将混合 MKNF 知识库转换为等价的逻辑程序。随后，XSB-Prolog 引擎执行该逻辑程序以完成查询回答。为降低内存消耗，我们建议仅在预处理阶段使用 NoHr 来生成转换后的逻辑程序。在系统运行期间，只需通过 XSB-Prolog 独立执行生成的逻辑程序。该方法在维持推理能力的同时，显著降低了运行时的内存使用量，因为实时系统不再依赖完整的 NoHr 基础设施。如图 1 所示，直到写入文件为止的步骤均在系统外部执行，以生成转换后的文件。随后，在实时应用中，可在 Prolog 环境内直接使用该文件来执行用例。

实现所提出策略的一个关键组成部分，是管理一致性解决机制。该机制采用知识库加倍和半负变换。虽然这一过程由 NoHr 推理器在内部处理，且该推理器能够返回一致或矛盾的结果，但我们的方法完全依赖于经过转换和加倍、并在 Prolog 引擎中独立执行的逻辑程序。因此，必须在 Prolog 环境中制定一套自定义策略，以复现原本由 NoHr 实现的行为。

4 其他表达能力特性

通过分析各种用例，我们发现，实现航空应用所需的一些关键表达能力特性默认情况下并未在混合 MKNF 中得到支持，因此需要补充这些特性。其中一项是对混合 MKNF 的逻辑程序组件中的否定知识进行显式表示。请看以下情形："只有在没有航行通告表明跑道已关闭，且跑道上不存在障碍物时，跑道才可运行"。这一情形最初可用以下 MKNF 规则表示：

$$\mathbf{K}\ \mathrm{OpnRwy(X)} \leftarrow \mathbf{K}\ \mathrm{Rwy(X)},\ \mathbf{not}\ \mathrm{ob(A,X)},\ \mathbf{not}\ \mathrm{CldRwy(X)}. \tag{22}$$

然而，鉴于航空领域具有安全关键特性，谓词 *ob(A,X)* 表示机场 *A* 的跑道*X* 上存在障碍物，因此格外重要。使用 **not** 运算符结合*ob(A,X)* 来推断跑道可运行，会带来安全隐患，因为这种推断依赖于证据缺失，而非明确确认。在这一领域，这种假设不可接受。因此，

我们必须明确核实跑道上没有障碍物，从而提高系统的可靠性。一种实现方式是将 **not** 运算符替换为经典否定，记作 ¬。

对于一元和二元谓词，可以通过语法转换将经典否定纳入现有的混合 MKNF 语义。对于一元谓词A，我们引入一个新概念B，并在描述逻辑中加入公理$B \equiv \neg A$，使B能够在规则中表示A的经典否定。对于二元谓词P，我们创建相应的谓词Q，并通过不相交公理$P \sqcap Q \sqsubseteq \bot$强制$Q$表示$P$的经典否定。然而，这种方法需要额外的知识工程，且仅适用于一元和二元谓词。因此，在规则部分扩展经典否定算子不仅能增强表达能力，还能简化知识工程，并更轻松地对类似上述示例的关键场景进行建模。加入经典否定扩展后，规则 (22) 可改写为：

$$\mathbf{K}\ \mathrm{OpnRwy(X)} \leftarrow \mathbf{K}\ \mathrm{Rwy(X)}, \mathbf{K}\neg\mathrm{ob(A,X)},\ \mathbf{not}\ \mathrm{CldRwy(X)}. \tag{23}$$

目前还缺少另一项关键功能：在知识库中表示完整性约束，并在查询回答过程中强制执行这些约束。在以下各节中，我们将介绍实现经典否定的启发式方法，并讨论如何确保查询结果符合混合 MKNF 上下文中指定的所有完整性约束。

4.1 经典否定

基于 [35] 和 [2], 的见解，并结合现有的矛盾消解技术（如知识库加倍和半负变换），我们提出一种将经典否定整合到混合 MKNF 知识库的逻辑程序组件中的方法。在此背景下，修改后的混合 MKNF 表示为 KB $=$ (O, P)，其中 O 表示描述逻辑本体，P 包含形式为 (1) 的规则。这里，H、A_m（$1 \leq \mathrm{i} \leq \mathrm{m}$）和 B_n（$1 \leq \mathrm{j} \leq \mathrm{n}$）是一阶原子，前面可以带有经典否定，如 (¬) 所示。在这一修订版的混合 MKNF 中，逻辑程序规则现在包含经典否定文字。正如 [35], 所强调的，这种经典否定可以通过句法转换消除，从而能够重用现有的混合 MKNF 推理算法，这正是本提案的核心思想。修改后的知识库 KB′ 通过将 P 中每个经典否定原子 ¬L 替换为一个新谓词 L′，并引入形式为 $\bot \leftarrow \mathrm{K\ L}$、$\mathrm{K\ L'}$ 的约束 C 而从 KB 得到。由此得到修改后的知识库 $\mathrm{KB'} = (\mathrm{O}, \mathrm{P'} \cup \mathrm{C})$。随后，按照 [1], 中的详细说明，应用知识库加倍和半负变换，得到加倍后的知识库 $\mathrm{KB'}^d = (\mathrm{O}, \mathrm{O}^d, \mathrm{P'}^d)$，如下所示：

1. O^d 通过将 O 中的每个谓词 L 替换为 L^d得到；2. P^d 通过将 P 中每条形式如 (1) 的规则转换为两条规则而得到：i $\mathrm{K\ H} \leftarrow \mathrm{K\ A1}, \ldots, \mathrm{K\ A}_i, \mathrm{not\ B}_1^d, \ldots, \mathrm{not\ B}_j^d$；ii $\mathrm{K\ H}^d \leftarrow \mathrm{K\ A}_1^d, \ldots, \mathrm{K\ A}_i^d, \mathrm{not\ B1}, \ldots, \mathrm{not\ B}j, \mathrm{not\ NH}$；3. C^d 由 C 中每条形式如 $\bot \leftarrow \mathrm{K\ L}, \mathrm{K\ L'}$ 的约束转换为两条规则而得到：$\mathrm{K\ NL} \leftarrow \mathrm{K\ L'}$ 和 $\mathrm{K\ NL'} \leftarrow \mathrm{K\ L}$。（$\bot \leftarrow \mathrm{K\ L}, \mathrm{K\ L'} \equiv \mathrm{K}\ \neg\mathrm{L} \leftarrow \mathrm{K\ L'}$ 和 $\mathrm{K}\ \neg\mathrm{L}' \leftarrow \mathrm{K\ L}$。将 ¬L 和 ¬L′ 替换为 NL 和 NL′。）

其中 A_i、A_i^d、Bj、B_j^d、H 和 H^d可以表示 L或 L'。

考虑混合 MKNF 知识库 $KB_2 = (O,\ P)$，其中 O 包含断言 *Rwy(rw1)*，而P 包含规则（23）和事实：$\mathbf{K}\ \neg ob(ifbo,rw1)$。根据句法转换，这些规则可以转换为以下规则，从而得到修改后的知识库 $KB_2' = (O,\ P' \cup C)$，如下所示（这里只提供逻辑程序部分）：

$$\mathbf{K}\ \mathrm{OpnRwy(X)} \leftarrow \mathbf{K}\ \mathrm{Rwy}(X), \mathbf{K}\ \mathrm{ob}'(A,X),\ \mathbf{not}\ \mathrm{CldRwy}(X). \tag{24}$$

$$\bot \leftarrow \mathbf{K}\mathrm{ob}(A, X), \mathbf{K}\mathrm{ob}'(A, X). \tag{25}$$

$$\mathbf{K}\ \mathrm{ob}'(lfbo, rw1). \tag{26}$$

之后 通过知识库加倍和半负变换，我们得到 $KB_2^{\prime d} = (O,\ O^d,\ P'^d)$ ：

$$\mathbf{K}\ \mathrm{opnRwy(X)} \leftarrow \mathbf{K}\ \mathrm{rwy}(X), \mathbf{K}\ \mathrm{ob}'(A,X),\ \mathbf{not}\ \mathrm{cldRwy}^d(X). \tag{27}$$

$$\mathbf{K}\ \mathrm{opnRwy}^d(X) \leftarrow \mathbf{K}\ \mathrm{rwy}^d(X), \mathbf{K}\ \mathrm{ob}'^d(A,X),\ \mathbf{not}\ \mathrm{cldRwy}(X), \mathbf{not}\ \mathrm{NopnRwy}(X). \tag{28}$$

$$\mathbf{K}\ \mathrm{Nob}(A,X) \leftarrow \mathbf{K}\ \mathrm{ob}'(A,X). \tag{29}$$

$$\mathbf{K}\mathrm{Nob}'(A,X) \leftarrow \mathbf{K}\mathrm{ob}(A,X)。 \tag{30}$$

$$\mathbf{K}\ \mathrm{ob}'(lfbo, rw1). \tag{31}$$

$$\mathbf{K}\ \mathrm{ob}'^d(lfbo, rw1) \leftarrow \mathbf{not}\ \mathrm{Nob}'(lfbo, rw1). \tag{32}$$

应用所有转换后，我们得到修改后的知识库 $KB_2^{\prime d} = (O, O^d, P'^d)$，其中规则中的所有经典否定都已通过句法方式消除。尽管引入了额外的谓词和规则，我们假设所提出的句法转换能够保持原始含经典否定的混合 MKNF 知识库与转换后不含否定的知识库之间的语义等价。现有方法支持这一假设，例如 [35] 和 [2], 均采用相同的转换来处理逻辑程序中的经典否定；此外， [1] 提出的知识库加倍和半负变换则用于处理混合 MKNF 的描述逻辑部分中的经典否定。语义等价性的形式化证明仍在进行中，并计划在后续论文中发表。因此，我们可以采用现有的查询回答算法，例如 [26] 和 [7]中提出的算法。具体做法是，先将描述逻辑部分转换为与其语义等价的逻辑程序规则 P^O，然后应用句法转换以消除经典否定，得到$KB'_2 = (\ /0, P^O, P' \cup C)$；随后应用知识库加倍和半负变换，并使用所得程序通过 XSB-Prolog 进行查询回答。

通过在KB_2, 中添加断言*rwy(rw1)* ，我们可以推断出*opnRwy(rw1)* ，因为存在一个表明没有障碍物的显式否定事实，即 $\neg ob(lfbo,rw1)$，且没有指明闭包。现在，考虑一种矛盾情形：增加了断言*ob(lfbo,rw1)* ，该断言也需要复制。在这种情况下，*ob(lfbo,rw1)* 会变得矛盾，因为原子及其经典否定同时为真。在这种情况下，规则 (29) 和 (30) 确保$Nob(X,Y)$和$Nob'(X,Y)$都为真。因此，由于半否定变换，推断$ob'^d(lfbo, rw1)$和$ob^d(lfbo, rw2)$都为假。由此可以检测出原子*ob(lfbo,rw1)*及其副本*obı(lfbo,rw2)*是矛盾的。请记住，只有L 及其副本L^d 都为真时，原子L 才被视为一致地为真。然后，可以使用前文所述的一致性查询应答机制，正确处理此类情况并仅提供一致答案。

4.2 完整性约束

完整性约束是用于验证显式知识和推断知识有效性的条件。它们的目的并非产生新的推断，而是确保数据一致性 [4]。我们的目标是显式表示这些约束，并确保给定查询结果中的所有原子都符合这些约束。任何被发现违反约束的原子都会被标记为约束违反，并从最终查询响应中排除。受 [20]和 [8], 中的研究启发，我们提出将完整性约束表示为如下形式的条件-动作规则：

$$A_1, \ldots, A_n,\ \mathrm{not}\ B_1, \ldots, \mathrm{not}\ B_m \Rightarrow a_1, \ldots, a_k \tag{33}$$

这里，$A1,\ldots,A_n$和 not $B1,\ldots,$not B_m表示条件，而$a1,\ldots,a_k$表示动作。每个动作都由原子$\mathbf{K}\,A$或其经典否定$\mathbf{K}\ \neg A$表示。考虑这样一个事实：只有满足特定条件时，它才被视为一致。当不满足该条件时，导致违反约束的原子会被认定为与已定义的约束冲突，并从后续推理过程中排除。为此，我们将被违反的条件表示为形式为 (33) 的规则，其中规则头包含待标记为冲突的原子的经典否定。当规则体成立时，即表示发生了约束违反，并导出相应的规则头。因此，原子本身及其经典否定可能同时存在于知识库中，从而导致不一致。规则 (33) 将转换为一组带有**K** 和 **not** 算子的 MKNF 规则，如下所示：

$$\begin{aligned}\mathbf{K}\,a_1 \leftarrow \mathbf{K}\,A_1,\ldots,\mathbf{K}\,A_n,\ \mathbf{not}\,B_1,\ldots,\mathbf{not}\,B_m\\ \ldots\\ \mathbf{K}\,a_k \leftarrow \mathbf{K}\,A_1,\ldots,\mathbf{K}\,A_n,\ \mathbf{not}\,B_1,\ldots,\mathbf{not}\,B_m\end{aligned}$$

转换后，该约束被视为标准的 MKNF 规则。为处理经典否定，我们采用所提出的句法转换技术，并结合基于知识库加倍和半负变换的不一致性消解策略。当转换后的 MKNF 规则体求值为真时，规则头会推导出某个原子的否定，表明存在违规。这会导致 MKNF 知识库中出现矛盾，即某个原子及其否定同时存在。因此，通过应用一致性消解策略，查询回答过程中只会产生一致答案，从而消除所有矛盾答案。这一特性在航空领域有多种应用。其中一种应用是识别不完整的航行通告，例如缺少开始时间、结束时间或发布机构等关键信息的航行通告。这些不完整的消息被视为不一致，并会从推理过程中排除，以维护系统决策的完整性。以下是一个对航行通告消息施加基本有效性条件的完整性约束示例：“航行通告必须同时包含开始时间和结束时间”。

$$\mathbf{K}\ \neg\mathsf{Notam}(X) \leftarrow \mathbf{K}\ \mathsf{Notam}(X),\ \mathbf{not}\ \mathsf{hasStartTime}(X,Y),\ \mathbf{not}\ \mathsf{hasEndTime}(X,Z). \tag{34}$$

根据规则 (34)，如果缺少开始时间或结束时间，则认为该航行通告不一致，因此将其排除在后续推理过程之外。另一种应用是检测相互矛盾的航行通告，例如，不同机构发布的两份航行通告在同一时间段内涉及同一条跑道，但一份指出该跑道已关闭，另一份则称其可运行。带有修复操作的完整性约束有助于识别此类航行通告，并将其排除在后续推理过程之外。

5 系统描述

图 8 展示了该系统的简化架构。我们开发了一种预处理工具，它利用 NoHr 推理器中嵌入的转换器，将描述逻辑本体转换为等价的规则集。此过程涉及知识库加倍和半负变换。在转换过程中，NoHr 转换器将描述逻辑本体中的每个概念和角色系统地映射为两个不同的谓词，分别表示原始版本和加倍版本。例如，*Rwy* 被映射为 *aRwy*，表示原始谓词；并被映射为 *dRwy*，表示加倍后的谓词。此外，转换后程序中的所有谓词都声明为表格化谓词，并且标准的 **not** 运算符被替换

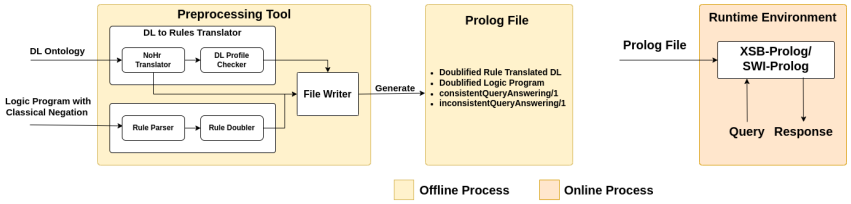


图 8：所提工作流程的架构

为表格化否定，记作 **tnot**。例如，描述逻辑公理 (4) 被转换为规则 (11) 和 (12)，这些规则出现在下面由 NoHr 转换器生成的 Prolog 程序中。

```
1 arwy(X):- aopnRwy(X).2 drwy(X):-
dopnRwy(X),tnot(nrwy(X)).
```

规则中的经典否定可以使用符号“-”表示。需要额外的解析器来处理混合 MKNF 逻辑程序部分中的这种经典否定。该解析器根据所提出的句法转换技术消除经典否定，随后应用知识库加倍和半负变换。此外，它将所有原子指定为表格化原子，并将 **not** 运算符改为 **tnot**。规则解析器将规则 (27) 和 (28) 转换为如下 Prolog 程序（见第 6.1 节）： $\neg ob(X)$ 通过添加前缀“*non*”替换为新引入的谓词$nonob(X)$，而不是 ¬。

```
1 aopnRwy(X):- arwy(X),anonob(A,X),tnot(dcldRwy(X)).2 dopnRwy(X):-
drwy(X),dnonob(A,X),tnot(acldRwy(X)),tnot(nopnRwy(X)).
```

为实现查询回答，我们引入了两个元谓词：*consistentQuery/1*，用于返回一致答案；以及 *inconsistentQuery/1*，用于返回矛盾答案。如（20）和（21）所述，这些元谓词通过与转换后的逻辑程序进行匹配来实现查询加倍。加倍后，这些元谓词分别实现条件$q(X) \wedge q^d(X)$和$q(X) \wedge \mathbf{not}\, q^d(X)$，以查询一致答案和矛盾答案。内置的 Prolog 谓词*atom_concat/3* 用于添加前缀字符串，这对查询加倍很有帮助。随后，使用*call/1* 谓词执行转换后的查询。*file writermodule*文件写入模块将翻译后的描述逻辑与预定义的元谓词及解析器生成的加倍逻辑程序整合，并导出至可由 Prolog 引擎执行的文件。它还会生成一份关于描述逻辑本体中不支持的公理的报告，示例见 https://drive.uca.fr/d/040c6ecbe940449e9c0d/。翻译完成后，加倍的逻辑程序部分以及翻译后的描述逻辑公理可由支持良基语义的 Prolog 引擎执行。对于查询回答，我们使用 SWI-Prolog 或 XSB-Prolog 进行评估。图 9 和图 10 分别展示了使用 XSB-Prolog 和 SWI-Prolog 评估每个查询时的查询响应时间和内存消耗。总体而言，除首次执行较复杂的递归查询外，我们观察到所有查询的内存使用量都显著降低。这是因为表格化功能会在初次计算后存储先前的结果，使后续执行相同查询时可以复用这些结果，而无需重复完整计算。本次评估所用的预处理文件和查询见：https://drive.uca.fr/d/e5482dd1af5f4fdd8bfb/。

6 相关工作

一般而言，描述逻辑与逻辑程序的结合主要有三种方式：松散集成、紧密集成和完全集成 [14]。这些集成策略各自都可以基于两种主要语

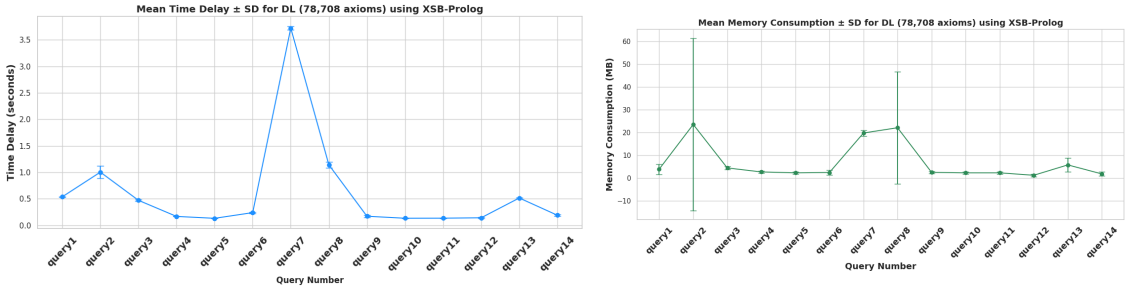


图 9：使用 XSB-Prolo 时各查询的响应时间和内存消耗 g

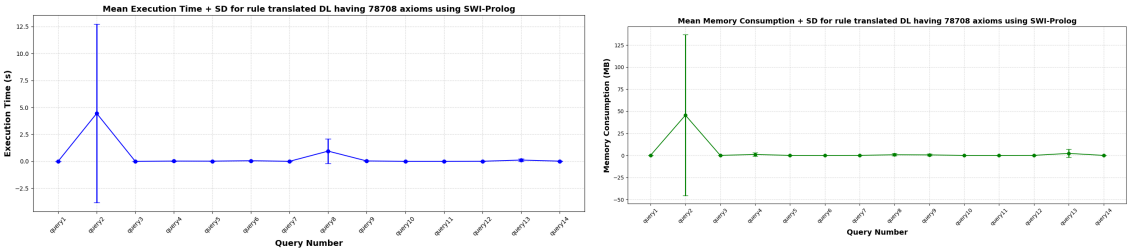


图 10：使用 SWI-Prolog 时各查询的响应时间和内存消耗

义之一：良基语义 [21] 或答案集语义 [12]。在松散集成中，描述逻辑本体和逻辑程序规则被视为两个独立实体，并通过指定的接口交换知识。DL 程序是一种松散集成方法，提出时同时采用了答案集语义 [16] 和良基语义 [15]。在 DL 程序中，描述逻辑与逻辑程序规则之间的知识交换仅通过外部运算符实现，因此需要额外的规则，这增加了知识工程师使用该语言的复杂性。DL 程序中唯一的推理任务是模型计算，计算所得的模型会存储在数据库中，以便进行查询回答。然而，这种方法对内存的利用效率不高，因为它需要计算并存储完整模型。尽管如此，DL 程序的优点在于能够与外部数据源集成，例如 Python 程序或关系数据库。一种名为 *HEX* [18]，的扩展语言及其对应的推理器 *DLV-Hex* [17]，支持这种集成。此外，还有一种名为 Drew System [44] 的优化推理器，它利用两种可通过 Datalog 重写的描述逻辑，$\mathscr{LDL}+$ [22] 和 $\mathscr{SROEL}$ [31]，用于 DL 程序。此外，还有一种名为 F-Logic# 的松散集成替代方案，它将 F-逻辑与 DL 本体结合起来[23]。

在紧密集成中，定义一个同时满足描述逻辑部分和逻辑程序部分的集成模型。该集成模型为$M = M_O \cup M_L$，其中M_O满足描述逻辑本体，而M_L满足相应的逻辑程序规则。我们从描述逻辑部分的一个模型开始，记为M_O。利用该模型，我们分两步简化基化后的逻辑程序部分。首先，移除所有主体中含有在M_O中为假的描述逻辑原子的逻辑程序规则，因为这些规则无法满足。然后，在剩余规则中，移除主体里在M_O中为真的描述逻辑原子，因为它们已得到满足。由此得到的逻辑程序版本不再包含任何描述逻辑原子。接着，我们采用答案集语义或良基语义，为这个简化后的逻辑程序计算一个模型。对每个可能的描述逻辑模型M_O重复这一过程，并为每个模型生成相应的逻辑程序模型。目前已有多种混合语言以紧密集成为重点开发。其中，R-Hybrid KB [40] 和 DL+Log [41] 以答案集语义为基础，而 HD-Rule 则以良基语义为基础 [11]。此外，

人们还提出了 DL+Log 的一种扩展，将封闭世界谓词纳入描述逻辑本体，从而创建了 Clopen 知识库 [6]。此外，人们还提出了弹性逻辑程序（RLP），以融合紧密集成与松散集成：松散集成仅在描述逻辑与逻辑程序之间共享逻辑后果，而紧密集成则支持不同个体模型之间的知识共享 [36]。在紧密集成中，即使采用良基语义，也可能存在许多模型，这使推理过程变得复杂。因此，在这种情况下构建实用的推理器颇具挑战性。此外，由于描述逻辑与逻辑程序之间采用基于模型的集成，知识工程师必须了解描述逻辑的各个个体模型，才能生成相应的规则程序，这进一步增加了知识工程的难度。

在完全集成中，定义了一种统一的非单调形式体系，用于封装描述逻辑和逻辑程序的语义。人们提出了多种一阶逻辑的非单调扩展，包括默认逻辑、认知逻辑、可废止逻辑和 circumscription（ circumscription）。理想情况下，选择其中一种扩展来集成描述逻辑和规则，构成了该方法的基本概念。在完全集成中，我们考察了以最小知识与否定即失败（ MKNF）、开放答案集编程（OASP） [24], 以及可废止逻辑 [3]为基础的研究。由于缺乏封闭世界推理能力，我们有意排除了大量将描述逻辑与一阶规则相结合的研究，例如 SWRL [32], AL-Log [9], CARIN [33], 。鉴于 OASP 不可判定，研究者提出了若干可判定变体。其中包括 F-Hybrid 知识库，它将 $\mathscr{SHOQ}$ 描述逻辑与有限集合的森林逻辑程序（FLP） [19]相集成。这些程序仅允许一元谓词和二元谓词，并遵循森林模型性质。此外，G-Hybrid 知识库将 $\mathscr{DLROO}^{-\{\leq\}}$ 与满足守卫限制的有限集合 ASP 规则相结合；在该限制下，每条规则都包含一个守卫谓词，且规则中的所有变量都必须出现在该谓词中 [25]。在此背景下，混合 MKNF 提供了一种更通用、更具表达力的语言，将描述逻辑与逻辑程序相集成 [38]。

7 结论

我们将混合 MKNF 引入为一种面向航空应用（例如 NARS）的知识表示语言。为解决航空系统中的内存限制问题，我们提出了一种替代工作流程，通过离线预处理使用 NoHr 推理器。我们使用 Prolog 元谓词处理一致和矛盾的查询响应。我们开发了启发式方法，通过经典否定扩展混合 MKNF，并通过修复操作支持完整性约束。我们创建了一种工具，将这些启发式方法与 NoHr 相结合，把混合 MKNF 知识库转换为等价的逻辑程序规则集合。在未来的研究中，我们计划在良基语义下，正式定义如何将逻辑程序中的经典否定整合到混合 MKNF 语义中。本研究旨在识别混合 MKNF 的句法片段，例如确定程序和分层程序，并确定它们是否兼容 I-DLV、Trealla Prolog 和 Tau-Prolog 等轻量级规则引擎。我们的目标是在资源受限平台上实现高效推理，并根据表达能力要求指导合适规则引擎的选择。我们将通过调整溯源技术，使其适用于良基语义下的混合 MKNF，从而提供推理依据，增强可解释性。最后，我们将把时间推理纳入混合 MKNF，以支持航行通告等场景：此类场景中的事实在特定时间区间内有效。为此，我们会在框架中扩展时态算子。

参考文献

[1] José Júlio Alferes、Matthias Knorr 和 Terrance Swift（2013）：混合 MKNF 知识库的查询驱动过程。ACM计算逻辑汇刊 14(2)，第 16:1–16:43 页，doi:10.1145/2480759.2480768。[2]José Júlio Alferes、Luís Moniz Pereira 和 Teodor C. Przymusinski（1998）：非单调推理与逻辑编程中的“经典”否定。J. Autom. Reason. 20(1)，第 107–142 页，doi:10.1023/A:1005900924623。[3] Grigoris Antoniou 和 Antonis Bikakis（2007）：DR-Prolog：一种在语义网上使用规则与本体进行可撤销推理的系统。IEEE Trans. Knowl. Data Eng. 19(2)，第 233–245 页，doi:10.1109/TKDE.2007.29。[4] Patrizia Asirelli、Michelle De Santis 和 Maurizio Martelli（1985）：逻辑数据库中的完整性约束。《逻辑编程杂志》2(3)，第 221–232 页，doi:10.1016/0743-1066(85)90020-2。网址：https://www.sciencedirect.com/science/article/pii/0743106685900202。[5] Franz Baader、Diego Calvanese、Deborah L. McGuinness、Daniele Nardi 和 Peter F. Patel-Schneider 编（2003）：描述逻辑手册：理论、实现与应用。剑桥大学出版社。[6] Labinot Bajraktari、Magdalena Ortiz 和 Mantas Simkus（2018）：将规则与本体结合为 Clopen 知识库。载于 Sheila A. McIlraith 和 Kilian Q. Weinberger 编：《第三十二届美国人工智能协会人工智能会议（AAAI-18）、第三十届人工智能创新应用会议（IAAI-18）和第八届美国人工智能协会人工智能教育进展研讨会（EAAI-18）论文集》，美国路易斯安那州新奥尔良，2018年2月2日至7日，AAAI出版社，第 1728–1735 页，doi:10.1609/aaai.v32i1.11565。[7] Nuno Costa、Matthias Knorr 和 João Leite（2015）：NoHR 的下一步：OWL 2 QL。载于 Marcelo Arenas、Óscar Corcho、Elena Simperl、Markus Strohmaier、Mathieu d’Aquin、Kavitha Srinivas、Paul Groth、Michel Dumon- tier、Jeff Heflin、Krishnaprasad Thirunarayan 和 Steffen Staab 编：《语义网——ISWC 2015——第十四届国际语义网会议论文集，第一部分》，美国宾夕法尼亚州伯利恒，2015年10月11日至15日，计算机科学讲义 9366，施普林格，第 569–586 页，doi:10.1007/978-3-319-25007-6_33.[8] Luís Cruz-Filipe、Michael Franz、Artavazd Hakhverdyan、Marta Ludovico、Isabel Nunes 和 Peter Schneider-Kamp（2015）：主动完整性约束：从理论到实现。载于 Ana L. N. Fred、Jan L. G. Dietz、David Aveiro、Kecheng Liu 和 Joaquim Filipe 编：《知识发现、知识工程与知识管理——第七届国际联合会议，IC3K 2015，葡萄牙里斯本，2015年11月12日至14日，修订版精选论文集》，计算机与信息科学通信 631，施普林格，第 399–420 页，doi:10.1007/978-3-319-52758-1_22.[9] Francesco M. Donini、Maurizio Lenzerini、Daniele Nardi 和 Andrea Schaerf（1998）：AL-log：整合 Datalog 与描述逻辑。智能信息系统杂志 10(3)，第 227–252 页，doi:10.1023/A:1008687430626。[10] Wlodzimierz Drabent（2010）：使用非单调规则进行混合推理。载于 Uwe Aßmann、Andreas Bartho 和 Christian Wende 编：《推理网：面向软件工程的语义技术，第六届国际暑期学校 2010》，德国德累斯顿，2010年8月30日至9月3日。教程讲座，计算机科学讲义 6325，施普林格，第 28–61 页，doi:10.1007/978-3-642-15543-7_2.[11] Wlodzimierz Drabent、Jakob Henriksson 和 Jan Maluszynski（2007）：HD-rules：一种将 Prolog 与描述逻辑推理器连接起来的混合系统。载于 Axel Polleres、David Pearce、Stijn Heymans 和 Edna Ruckhaus 编：《ICLP’07研讨会论文集：逻辑编程在万维网、语义网和语义网服务中的应用》，ALPSWS 2007，葡萄牙波尔图，2007年9月13日，CEUR Workshop Proceedings 287，CEUR-WS.org。网址：https://ceur-ws.org/Vol-287/paper_9.pdf。[12] Thomas Eiter、Giovambattista Ianni 和 Thomas Krennwallner（2009）：答案集编程：入门。载于 Sergio Tessaris 、 Enrico Franconi 、 Thomas Eiter 、 Claudio Gutierrez 、 Siegfried Handschuh 、 Marie-Christine Rousset 和 Renate A. Schmidt 编：《推理网：信息系统的语义技术，第五届国际暑期学校 2009 》，意大利布雷萨诺内，2009 年 8 月 30 日至 9 月 4 日。教程

讲座，《计算机科学讲义》5689，施普林格，第 40–110 页，doi:10.1007/978-3-642-03754-2_-2.[13] Thomas Eiter、Giovambattista Ianni、Thomas Krennwallner 和 Axel Polleres（2008）：语义网的规则与本体。载于 Cristina Baroglio、Piero A. Bonatti、Jan Maluszynski、Massimo Marchiori、Axel Polleres 和 Sebastian Schaffert 编：《推理网》，第四届2008年国际暑期学校，威尼斯，意大利，2008年9月7日至11日，教程讲座，《计算机科学讲义》5224，施普林格，第 1–53 页，doi:10.1007/978-3-540-85658-0 1._[14] Thomas Eiter、Giovambattista Ianni、Thomas Krennwallner 和 Axel Polleres（2008）：语义网的规则与本体。载于 Cristina Baroglio、Piero A. Bonatti、Jan Maluszynski、Massimo Marchiori、Axel Polleres 和 Sebastian Schaffert 编：《推理网》，第四届2008年国际暑期学校，威尼斯，意大利，2008年9月7日至11日，教程讲座，《计算机科学讲义》5224，施普林格，第 1–53 页，doi:10.1007/978-3-540-85658-0 1._[15] Thomas Eiter、Giovambattista Ianni、Thomas Lukasiewicz 和 Roman Schindlauer（2011）：语义网描述逻辑程序的良基语义。《ACM计算逻辑汇刊》12(2)，第 11:1–11:41 页，doi:10.1145/1877714.1877717.[16] Thomas Eiter、Giovambattista Ianni、Thomas Lukasiewicz、Roman Schindlauer 和 Hans Tompits（2008）：将答案集编程与描述逻辑结合用于语义网。《人工智能》172(12)，第 1495–1539 页，doi:10.1016/j.artint.2008.04.002.[17] Thomas Eiter、Giovambattista Ianni、Roman Schindlauer 和 Hans Tompits（2005）：DLV-HEX：通过答案集编程处理语义网。可见 https://api.semanticscholar.org/ CorpusID:9592774.[18] Thomas Eiter、Giovambattista Ianni、Roman Schindlauer 和 Hans Tompits（2005）：在答案集编程中统一整合高阶推理与外部求值。载于 Leslie Pack Kaelbling 和 Alessandro Saffiotti 编：《IJCAI-05》，第十九届国际人工智能联合会议论文集，爱丁堡，苏格兰，英国，2005年7月30日至8月5日，Professional Book Center，第 90–96 页。可见 http://ijcai.org/Proceedings/05/Papers/1353.pdf.[19] Cristina Feier 和 Stijn Heymans（2013）：森林逻辑程序与 f-混合知识库的推理。《逻辑编程理论与实践》13(3)，第 395–463 页，doi:10.1017/S1471068411000597.[20] Sergio Flesca、Sergio Greco 和 Ester Zumpano（2004）：主动完整性约束。载于：第六届 ACM SIGPLAN 声明式编程原理与实践国际会议论文集，PPDP ’04，计算机协会，纽约，美国，纽约州，第 98–107 页，doi:10.1145/1013963.1013977.[21] Allen Van Gelder、Kenneth A. Ross 和 John S. Schlipf（1991）：一般逻辑程序的良基语义。《ACM杂志》38(3)，第 620–650 页，doi:10.1145/116825.116838.[22] Stijn Heymans、Thomas Eiter 和 Guohui Xiao（2010）：在可重写为 Datalog 的描述逻辑上对 DL 程序进行可处理推理。载于 Helder Coelho、Rudi Studer 和 Michael J. Wooldridge 编：《ECAI 2010——第十九届欧洲人工智能会议》，葡萄牙里斯本，2010年8月16日至20日，论文集，《人工智能与应用前沿》215，IOS Press，第 35–40 页。可见 http://www. booksonline.iospress.nl/Content/View.aspx?piid=17710.[23] Stijn Heymans、Roman Korf、Michael Erdmann、Jörg Pührer 和 Thomas Eiter（2010）：F-Logic#：松散耦合 F-Logic 规则与本体。载于 Jimmy X. Huang、Irwin King、Vijay V. Raghavan 和 Stefan M. Rüger 编：《WI 2010——2010年 IEEE/WIC/ACM 国际网络智能会议》，加拿大多伦多，2010年8月31日至9月3日，主会议论文集，IEEE 计算机学会，第 248–255 页，doi:10.1109/WI-IAT.2010.44.[24] Stijn Heymans、Davy Van Nieuwenborgh 和 Dirk Vermeir（2007）：面向语义网的开放答案集编程。《应用逻辑杂志》5(1)，第 144–169 页，doi:10.1016/J.JAL.2006.02.001.[25] Stijn Heymans、Livia Predoiu、Cristina Feier、Jos de Bruijn 和 Davy Van Nieuwenborgh（2003）：G-混合知识库。可见 https://api.semanticscholar.org/CorpusID:9440798。

[26] Vadim Ivanov、Matthias Knorr 和 João Leite（2013）：带非单调规则的 EL 查询工具。载于 Harith Alani、Lalana Kagal、Achille Fokoue、Paul Groth、Chris Biemann、Josiane Xavier Parreira、Lora Aroyo、Natasha F. Noy、Chris Welty 和 Krzysztof Janowicz 编：《语义网——ISWC 2013——第十二届国际语义网会议》，悉尼，新南威尔士州，澳大利亚，2013年10月21日至25日，论文集，第一部分，《计算机科学讲义》8218，施普林格，第 216–231 页，doi:10.1007/978-3-642-41335-3_14.[27] Vedran Kasalica、Matthias Knorr、João Leite 和 Carlos Lopes（2020）：NoHR：概述。《人工智能》34(4)，第 509–515 页，doi:10.1007/S13218-020-00650-1.[28] Matthias Knorr（2021）：论本体与规则的结合。载于 Mantas Simkus 和 Ivan Varzinczak 编：《推理网：声明式人工智能——第十七届国际暑期学校》，鲁汶，比利时，2021年9月8日至15日，教程讲座，《计算机科学讲义》13100，施普林格，第 22–58 页，doi:10.1007/978-3-030-95481-9_2.[29] Matthias Knorr、José Júlio Alferes 和 Pascal Hitzler（2008）：混合 MKNF 知识库的一致良基模型。载于 Malik Ghallab、Constantine D. Spyropoulos、Nikos Fakotakis 和 Nikolaos M. Avouris 编：《ECAI 2008——第十八届欧洲人工智能会议》，希腊帕特雷，2008年7月21日至25日，论文集，《人工智能与应用前沿》178，IOS Press，第 99–103 页，doi:10.3233/978-1-58603-891-5-99.[30]Matthias Knorr、José Júlio Alferes 和 Pascal Hitzler（2011）：良基语义下描述逻辑的局部闭世界推理。《人工智能》175(9-10)，第 1528–1554 页，doi:10.1016/j.artint.2011.01.007.[31] Markus Krötzsch（2011）：面向 OWL EL 的高效基于规则的推理。载于 Toby Walsh 编：《IJCAI 2011》，第二十二届国际人工智能联合会议论文集，巴塞罗那，加泰罗尼亚，西班牙，2011年7月16日至22日，IJCAI/AAAI，第 2668–2673 页，doi:10.5591/978-1-57735-516-8/IJCAI11-444.[32] Hong Lan（，2004）：SWRL：一种结合 OWL 与 RuleML 的语义网规则语言。可见 https://api.semanticscholar.org/CorpusID:207974723.[33] Alon Y. Levy 和 Marie-Christine Rousset（1998）：在 CARIN 中结合 Horn 规则与描述逻辑。《人工智能》104(1-2)，第 165–209 页，doi:10.1016/S0004-3702(98)00048-4.[34] Vladimir Lifschitz（1991）：非单调数据库与认知查询。载于 John Mylopoulos 和 Raymond Reiter 编：《第十二届国际人工智能联合会议论文集》，悉尼，澳大利亚，1991年8月24日至30日，Morgan Kaufmann，第 381–386 页。可见 http://ijcai.org/Proceedings/91-1/Papers/059.pdf.[35] Vladimir Lifschitz、David Pearce 和 Agustín Valverde（2001）：强等价逻辑程序。《ACM计算逻辑汇刊》2(4)，第 526–541 页，doi:10.1145/383779.383783.[36] Sanja Lukumbuzya、Magdalena Ortiz 和 Mantas Simkus（2020）：弹性逻辑程序：受本体挑战的答案集程序。载于：第三十四届 AAAI 人工智能会议（AAAI 2020）、第三十二届人工智能创新应用会议（IAAI 2020）、第十届 AAAI人工智能教育进展研讨会（EAAI 2020）论文集，纽约，美国，纽约州，2020年2月7日至12日，AAAI出版社，第 2917–2924 页，doi:10.1609/aaai.v34i03.5683.[37] Boris Motik、Ian Horrocks、Riccardo Rosati 和 Ulrike Sattler（2006）：OWL 与逻辑编程能否始终和谐共存？载于 Isabel F. Cruz、Stefan Decker、Dean Allemang、Chris Preist、Daniel Schwabe、Peter Mika、Michael Uschold 和 Lora Aroyo 编：《语义网——ISWC 2006》，第五届国际语义网会议，雅典，佐治亚州，美国，2006年11月5日至9日，论文集，《计算机科学讲义》4273，施普林格，第 501–514 页，doi:10.1007/11926078_36.[38] Boris Motik 和 Riccardo Rosati（2007）：描述逻辑与逻辑编程的忠实整合。载于 Manuela M. Veloso 编：《IJCAI 2007》，第二十届国际人工智能联合会议论文集，海得拉巴，印度，2007年1月6日至12日，第 477–482 页。可见 http://ijcai.org/Proceedings/07/Papers/075.pdf.[39] Boris Motik 和 Riccardo Rosati（2010）：调和描述逻辑与规则。《ACM杂志》57(5)，第 30:1–30:62 页，doi:10.1145/1754399.1754403。

[40] Riccardo Rosati（2005）：本体与规则集成的可判定性和复杂性。《J. Web Semant.》3(1)，第 61–73 页，doi:10.1016/j.websem.2005.05.002。[41] Riccardo Rosati（2006）：描述逻辑+*log*：描述逻辑与析取 *Datalog* 的紧密集成。载于 Patrick Doherty、John Mylopoulos 和 Christopher A. Welty 编：《论文集：第十届知识表示与推理原理国际会议，英国湖区，*2006* 年 *6* 月 *2—5* 日》，AAAI出版社，第 68–78 页。可访问：http://www.aaai.org/Library/KR/2006/kr06-010.php。[42] Jessika Schulze（2016）：《人工智能与逻辑编程中的逻辑手册》。可访问：https://api.semanticscholar.org/CorpusID:63249517。[43] Gunjan Singh、Sumit Bhatia 和 Raghava Mutharaju（2020）：*OWL2Bench*：面向 *OWL 2* 推理器的基准测试。载于 Jeff Z. Pan、Valentina Tamma、Claudia d’Amato、Krzysztof Janowicz、Bo Fu、Axel Polleres、Oshani Seneviratne 和 Lalana Kagal 编：《语义网——*ISWC 2020*——第 *19* 届国际语义网会议，希腊雅典，*2020* 年 *11* 月 *2—6* 日，论文集，第二部分》，计算机科学讲义12507，施普林格，第 81–96 页，doi:10.1007/978-3-030-62466-8_6。[44] Guohui Xiao、Thomas Eiter 和 Stijn Heymans（2012）：非单调 *DL* 程序的 *DReW* 系统。载于 Juanzi Li、Guilin Qi、Dongyan Zhao、Wolfgang Nejdl 和 Hai-Tao Zheng 编：《语义网与网络科学——第六届中国语义网研讨会暨第一届中国网络科学会议，*CSWS 2012*，中国深圳，*2012* 年 *11* 月 *28—30* 日》，施普林格，第 383–390 页，doi:10.1007/978-1-4614-6880-6_33。