Hybridization of Description Logics and Logic Programming for Aeronautics Applications

Arun RAVEENDRAN NAIR SHEELA[1]

[1]*Thales, LIMOS Laboratory affiliated to Université Clermont-Auvergne*

Abstract

The deployment of Knowledge Representation and Reasoning technologies in aeronautics applications poses two primary challenges: achieving sufficient expressivity to capture complex domain knowledge and executing reasoning tasks efficiently while minimizing memory usage and computational overhead. An effective strategy for achieving the necessary expressivity involves integrating two fundamental KRR concepts: Description Logics(DLs) and Logic Programming(LP). The main objective of this thesis is to select a unified semantics for integrating DLs and LP, with a particular focus on its applicability and adaptability to the aeronautics domain. The chosen language will be studied in the context of representative aeronautics use cases, identifying challenges that arise during its adaptation. Based on these findings, the thesis will propose appropriate extensions to the unified semantics. Furthermore, an optimized reasoner will be developed with respect to the extended semantics. Finally, comprehensive user manuals and guidelines will be provided to assist knowledge engineers in modeling and implementing various aeronautics use cases using the proposed language.

Keywords

Description Logics, Logic Programming, Hybrid MKNF, Aeronautics

1. Introduction

The aeronautics industry is recognized as a safety-critical domain, which implies that system malfunctions can have safety consequences. Therefore, software systems used in this industry must exhibit reliable, robust, and explainable characteristics to prevent such outcomes. Additionally, many software systems in this field operate on resource-constrained hardware, such as microcontrollers. In this context, knowledge representation and reasoning (KRR) systems hold significant promise because they enable systems to produce transparent and explainable decisions. With the assistance of domain experts, knowledge specific to aeronautics applications can be captured using formal representation languages, such as ontologies. This knowledge contains critical operational information and is integrated with real-time data to make timely decisions through logical reasoning. However, adapting the KRR system to the aeronautics domain presents several significant challenges. These challenges include ensuring sufficient expressivity to encapsulate all complex domain knowledge and effectively executing necessary reasoning tasks with limited computational resources and minimal latency.

An effective strategy for tackling the expressivity challenge is the integration of rules and ontologies, which is a well-explored area of research. Ontologies, generally grounded in Description Logics (DLs), operate under the open-world assumption (OWA), which means that the absence of information does not imply its negation. They are used to encapsulate general domain knowledge through concepts and their relationships [1]. Conversely, Logic Programming (LP) rules operate under the closed-world assumption (CWA), which presumes that what is not known to be true is considered false. These rules are utilized to represent knowledge through if-then relationships [2]. Owing to their complementary characteristics, the integration of DLs and LP rules enhances the expressiveness of the resultant KRR, which is termed a hybrid knowledge base.

Motivation: To build an effective reasoning system that integrates DLs and LP, it is essential to adopt a suitable semantics that harmoniously combines the strengths of both paradigms. This hybrid semantics

RuleML+RR'25: Companion Proceedings of the 9th International Joint Conference on Rules and Reasoning, September 22–24, 2025, Istanbul, Turkiye.

Arun.RAVEENDRAN_NAIR_SHEELA@doctorant.uca.fr (A. R. N. SHEELA)

0009-0005-7260-8212 (A. R. N. SHEELA)

must integrate the strengths of both DLs and LP while ensuring semantic compatibility between them. Also, a reasoner must be implemented that is both sound and complete with respect to the chosen semantics to perform query answering answering. While query answering, the reasoner should exhibit strong performance characteristics, particularly low latency for query responses and high throughput for handling large volumes of data. Finally, to ensure broad applicability, especially in embedded or resource-constrained environments, the implementation should be optimized for minimal resource consumption, enabling lightweight and efficient operation even on platforms such as micro-controllers.

Use Case: We present a use case in the aeronautics domain that necessitates the integration of DLs with LP rules. We propose a system known as the NOTAM-Aware Reasoning System (NARS). Prior to a flight's departure, pilots must manually review all Notices to Airmen(NOTAM), which are official advisories that convey time-sensitive changes such as runway closures, airspace restrictions, or technical malfunctions, to ensure that the flight plan remains consistent with real-world. A NOTAM message typically contains crucial information, including what is affected—such as runways, airspace, or environmental changes—the location, validity time period, and the authority issuing the notice. All of these details are essential for helping pilots to plan their flights safely. For example, if a NOTAM indicates that the arrival runway, the specific runway designated for landing a flight at a given time, is closed during the scheduled landing period. In this case, the pilot must first identify the issue while reviewing hundreds of NOTAMs, and then plan the flight accordingly. Several agencies assist pilots in this review process. This manual verification and decision-making process is both time-consuming and susceptible to human error. NARS addresses this problem by reviewing NOTAMs and suggesting updates to the pilot when it detects real-world changes affecting the flight plan. The knowledge base for NARS can be constructed using a hybrid KRR system. On the DL side, static and hierarchical domain knowledge, such as airport infrastructure, different types of runways, and airspace zones can be represented. This structured knowledge provides a formal vocabulary to describe entities and their relationships-for example, stating that "a runway is part of an airport". In contrast, LP handles dynamic knowledge, which is encoded as conditional if-then rules. For instance, a rule might specify that "if a runway is closed during the scheduled landing time, then an alternate runway should be proposed"; or "if critical equipment at an airport is out of service, the flight operations team must be notified".

Section 2 presents existing approaches for combining DLs and LP, along with a brief justification for the selected formalism. Section 3 outlines the research plan with a timeline. Sections 4 and 5 introduce the chosen language, Hybrid MKNF, and its reasoner, NoHr, followed by a performance evaluation. Section 6 discusses the limitations of the current Hybrid MKNF framework based on the use case. Section 7 presents the results achieved so far, Section 8 outlines the ongoing work, and Section 9 proposes future research directions.

2. State of the Art

In general, three main approaches exist for combining DLs with LP: loose integration, tight integration, and full integration [3]. Each of these integration strategies can be based on one of two major semantics: well-founded [4] or answer set semantics [5].In loose integration, a DL ontology and LP rules are treated as distinct components, with an interface enabling controlled knowledge exchange. A prominent approach in this category is *DL-Programs*, which have been extensively studied under both answer set semantics [6] and well-founded semantics [7]. Knowledge exchange in DL-Programs is realized exclusively through external operators, i.e., additional constructors that mediate interaction between the DL ontology and the rule component. These operators necessitate supplementary rules, thereby increasing the language's complexity for knowledge engineers. For example, consider the DL axiom stating that a flight with an open arrival runway is an on-time flight:

$$\texttt{Flight} \sqcap \exists \texttt{hasArrivalRunway}.\texttt{OpnRwy} \sqsubseteq \texttt{OtmFlight}. \tag{1}$$

We may further specify that open runways are a subclass of runways and introduce a concrete runway instance as follows:

$$\texttt{OpnRwy} \sqsubseteq \texttt{Rwy}, \quad \texttt{Rwy}(\texttt{rw1}).$$

To represent the default assumption that all runways are open unless explicitly known to be closed, we define a DL-Program rule:

$$\texttt{opnRwy}(X) \leftarrow \texttt{DL[Rwy]}(X),\ \texttt{not DL[CldRwy]}(X). \tag{2}$$

Here, `DL[Rwy](X)` queries the ontology to check whether X is a runway, while `DL[CldRwy](X)` checks whether X is explicitly closed. The operator `not` denotes default negation, so `opnRwy(X)` is inferred whenever there is no evidence that X is closed. Importantly, `opnRwy(X)` is a predicate defined within the rule component and is not automatically propagated into the DL ontology. To make such inferences available to DL reasoning, DL-Programs introduce special update operators. For instance, we can define a rule that triggers an alert when a flight is not inferred to be on time:

$$\texttt{trigAlert}(X) \leftarrow \texttt{DL[Flight]}(X), \mathit{not}\ \ \texttt{DL[OpnRwy+=opnRwy; OtmFlight]}(X). \tag{3}$$

This rule (3) specifies that an alert is generated whenever a flight is not on time. The operator `+=` updates the ontology by incorporating the extension of `opnRwy` into the DL predicate `OpnRwy`. This update step is crucial: since only then the rule (2) is used in the inference process. More information about update operators is available here, [6].

The only reasoning task in DL-Programs is model checking, that is, verifying whether a given interpretation satisfies the axioms and rules of the knowledge base.Then the computed model is stored in a database for query answering. However, this approach is not memory-efficient, as it requires computing and storing the full model. Nonetheless, DL-programs offer the benefit of integrating with external data sources, such as Python programs or relational databases. An extended language known as *HEX* [8], along with its corresponding reasoner, *DLV-Hex* [9], supports this integration. Moreover, the *DReW system* [10] provides an optimized DL-Program reasoner by exploiting Datalog-rewritable DLs such as $\mathcal{LDL}^{+}$ [11] and $\mathcal{SROEL}$ [12]. An alternative loose integration approach, known as *F-Logic#*, combines F-Logic with DL ontologies [13].

In tight integration, reasoning is performed with respect to an integrated model that simultaneously satisfies both the DL ontology and the LP rules. Formally, such a model is defined as $M = M_O \cup M_L$, where M_O is a model of the DL ontology and M_L is a model of the LP program. The construction of M begins with a candidate DL model M_O. Using M_O, the grounded LP program is simplified in two steps. First, any rule whose body contains DL atoms that evaluate to false in M_O is discarded, since such rules cannot be satisfied. Second, for the remaining rules, DL atoms that evaluate to true in M_O are removed from the body, as they are already satisfied. This yields a residual LP program that no longer contains DL atoms. A model M_L of this simplified program is then computed under either answer set or well-founded semantics. The procedure is repeated for each possible DL model M_O, producing a corresponding LP model in each case. Several hybrid languages have been proposed in the literature to realize this notion of tight integration. Notably, R-Hybrid KB [14] and DL+Log [15] are founded on answer set semantics, while HD-Rule is based on well-founded semantics [16]. Furthermore, an extension of DL+Log has been introduced, which incorporates closed-world predicates within a DL ontology, leading to the creation of clopen knowledge base [17]. Additionally, Resilient Logic Programs (RLP) have been proposed to merge both tight and loose integration, where loose integration shares only logical consequences between DL and LP, while tight integration enables knowledge sharing between individual models [18]. In tight integration, even when employing well-founded semantics, numerous possible models may exist, complicating the reasoning process. Consequently, constructing a practical reasoner under these conditions is challenging. Additionally, the knowledge engineer must possess an understanding of individual models of DLs to generate the corresponding rule program due to the model-based integration between DL and LP, further complicating the knowledge engineering process.

In full integration, a singular unifying non-monotonic formalism is defined to encapsulate both the semantics of DLs and LP. Several non-monotonic extensions of first-order logic have been proposed, including default logic, epistemic logic, defeasible logic, and circumscription. Ideally, selecting one of these extensions to integrate both DL and rules constitutes the fundamental concept of this approach. In Full Integration, we encountered on works grounded in Minimal Knowledge and Negation-as-a-failure(MKNF), Open-Answer Set Programming (OASP) [19], and Defeasible Logics [20]. We

intentionally excluded a significant array of research that merge DLs with first-order rules, such as SWRL [21], AL-Log [22], Carin [23], due to their lack of closed-world reasoning capabilities. Given the undecidability of OASP, several decidable variants have been proposed. These include the F-Hybrid Knowledge Base, which integrates $\mathcal{SHOQ}$ DL with a finite set of Forest Logic Programs (FLP) [24]. These programs allow only unary and binary predicates and adhere to the forest-model property. Additionally, the G-Hybrid Knowledge Base combines $\mathcal{DLRO}^{-\{\leq\}}$ with a finite set of ASP rules that comply with the guardness restriction, wherein each rule contains a guarded predicate, and all variables within the rules must appear in that predicate [25]. In this context, Hybrid MKNF offers a more general and expressive formalism that integrates DLs and LP [26], as it is based on a unified language that seamlessly combines both paradigms.

Selection: In loose integration approaches such as DL-Programs and F-Logic#, information exchange from LP rules to the DL component is managed through additional operators. This process can be very complex for knowledge engineers, especially when dealing with large knowledge bases. Furthermore, reasoning often requires computing the entire model to answer queries, leading to high memory consumption. Consequently, any update to the knowledge base necessitates recomputing the model from scratch, making this approach impractical for the aeronautics domain. Similarly, in tight integration systems, as discussed earlier, multiple models may exist—even under well-founded semantics. This multiplicity complicates the construction of efficient reasoners. Moreover, among tight integration languages, only HD-Rules supports top-down query answering; other languages rely on answer set semantics and focus solely on model computation as their reasoning task. This again results in high memory usage. A further limitation of HD-Rules is that information flow is only supported from DL to LP, not vice versa, which restricts their applicability. Therefore, languages based on these methodologies are not ideal for the aeronautics domain. In the case of full integration, as in certain fragments of open answer set programming, strong syntactic restrictions are imposed to guarantee decidability. However, these restrictions introduce significant overhead in knowledge engineering. Considering these limitations, Hybrid MKNF emerges as a more general and expressive semantics combining DL and LP in a unified framework. Due to these reasons, we chose Hybrid MKNF as the KRR approach best suited for the aeronautics domain.

3. Research Methodology

The research began with a comprehensive study of the theoretical foundations of LP, including different types of LP rules, various semantics (such as stable model and well-founded semantics), and associated reasoning tools. A similar study was conducted for DLs, covering their syntax, semantics, reasoning algorithms, and existing implementations. This stage was referred to as the Foundation Phase. Following this, we explored existing research on the integration of DLs and LP. We first identified general approaches used to unify their semantics and then analyzed the main languages proposed in each category. This extensive literature review constituted the Survey Phase, during which we studied the theoretical foundations, strengths, and limitations of each approach.

Based on insights from the survey, we proceeded to select a hybrid language that was most suitable for the aeronautics domain. We implemented simple aeronautics use cases using different hybrid knowledge representation languages to support the selection process. The selection was guided by three key factors: combining the strengths of DL and LP with bidirectional information exchange, supporting efficient knowledge engineering (i.e., imposes minimal syntactic restrictions, while remaining accessible and comprehensible to knowledge engineers), and enabling the development of an optimized reasoner for efficient query answering. After reviewing the benefits and limitations of existing languages, we chose Hybrid MKNF because it provided a more general and expressive framework that integrates DLs and LP. Moreover, it supported a well-founded semantics, which enabled the implementation of efficient top-down query answering. This process formed the Selection Phase. Once the language was chosen, we moved to the Experimentation Phase. In this phase, we evaluated the availability and

performance of existing reasoners for the selected language and implemented more complex aeronautics use cases to assess its practical suitability. Based on the results from experimentation, we propose optimization strategies to improve the efficiency of existing Hybrid MKNF reasoner implementations. We will also identify additional expressivity requirements emerging from these use cases. To address these needs, we will propose extensions to the chosen Hybrid MKNF framework. The main features of these extensions will include classical negation, allowing explicit representation of negative knowledge in the LP component. A complete theoretical study, including semantics, computation methodology, and query answering algorithms, is currently being conducted as part of ongoing work. Other proposed extensions will include integrity constraints, enabling knowledge engineers to enforce conditions on query answers, and date-and-time reasoning, which will be essential for handling NOTAMs, each of which has a start and end time, ensuring that decisions respect these constraints. Additionally, we will enable the reasoner to generate justifications for each query answer. The timeline of my research, reflecting both the current plan and the results achieved so far, is illustrated in figure 1.

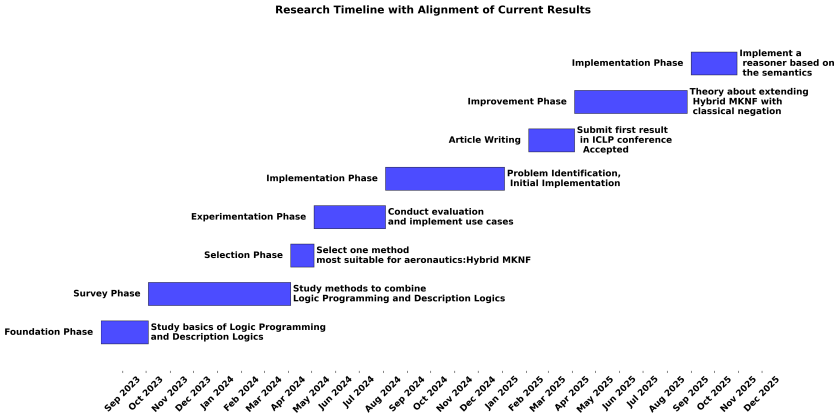

Figure 1: Approximate Timeline

4. Hybrid MKNF

We chose Hybrid MKNF as the knowledge representation language to DLs and LP for the aeronautics domain. We now provide a detailed overview of Hybrid MKNF. The Hybrid MKNF, as introduced in [26], is based on the logic of Minimal Knowledge and Negation as a Failure (MKNF), an extension of first-order logic that incorporates two modal operators, namely K and *not* originally described in [27]. The K operator denotes knowledge known by the system, whereas the *not* operator signifies knowledge not known by the system, functioning analogously to default negation in logic programming and facilitating closed-world reasoning. A Hybrid MKNF under well-founded semantics, $KB = (O, P)$ where O is a DL ontology and P is the set of MKNF rules of the form [28]:

$$KH \leftarrow KA_1, \ldots, KA_m, not\ B_1, \ldots, not\ B_n. \tag{4}$$

where H, A_i for $1 \leq i \leq m$, and B_j for $1 \leq j \leq n$ are first-order atoms. The rule (4) can be read as *"if all A_i are known to hold and all B_j are not known to hold then H is known to hold"*. We consider a simplified scenario from the use case, in which airport and runway availability is inferred from NOTAM data, and operational flight recommendations are derived using a Hybrid MKNF knowledge base, $KB = (O, P)$:

$$\texttt{Airport} \sqcap \forall \texttt{hasRwy}.\texttt{CldRwy} \sqsubseteq \texttt{CldAirport} \tag{5}$$

$$\texttt{Airport} \sqcap \exists \texttt{hasRwy}.\texttt{OpnRwy} \sqsubseteq \texttt{OpnAirport} \tag{6}$$

$$\texttt{OpnRwy} \sqsubseteq \texttt{Rwy} \tag{7}$$

$$\texttt{CldRwy} \sqcap \texttt{OpnRwy} \sqsubseteq \bot \tag{8}$$

$$K\ \texttt{CldRwy}(X) \leftarrow K\ \texttt{NOTAM}(X), K\ \texttt{affdBy}(X, Y), K\ \texttt{Rwy}(Y),\ K\ \texttt{opStat}(X, \text{closed}). \tag{9}$$

$$K\ \texttt{OpnRwy}(X) \leftarrow K\ \texttt{Rwy}(X),\ not\ \texttt{CldRwy}(X). \tag{10}$$

$$K\ \texttt{recommendDelay}(X) \leftarrow K\ \texttt{Flight}(X), K\ \texttt{hasDepartureAirport}(X, Y), not\ \texttt{OpnAirport}(Y). \tag{11}$$

Rule (10) infers that `rwy(rw1)` is considered an `OpnRwy(rw1)` by default, unless there is explicit evidence indicating that `rwy(rw1)` is a `CldRwy(rw1)`. The predicate `CldRwy(rw1)` can be derived when there exists a NOTAM that affects the specific runway, as described in Rule (10). This can be achieved using `not` operator with `CldRwy(X)` signifies that `rw1` is not known to be a closed runway. This allows the inference that, in the absence of a NOTAM message indicating a closure, the runway remains operational. Furthermore, by asserting the following additional facts:`hasRwy(lfbo, rw1)` and `Airport(lfbo)` and applying Rule (10) in conjunction with DL axiom (6), it can be concluded that `Airport(lfbo)` qualifies as an `OpnAirport`.

4.1. Semantics

The semantics of Hybrid MKNF can be formalized by transforming the knowledge base into a first-order formula with modal operators [28]. Consider $KB_1 = (O_1, P_1)$ where O_1 contain DL axiom (7) and an an assertion `Rwy(rw1)` , and P_1 contain a rule (10). The first-order translation of KB_1, denoted as $\pi(KB_1)$ is given by $K\pi(O_1) \wedge \pi(P_1)$:

$$\forall X(K\ \texttt{Rwy}(X) \leftarrow K\ \texttt{OpnRwy}(X)). \tag{12}$$

$$\forall X(K\ \texttt{OpnRwy}(X) \leftarrow K\ \texttt{Rwy}(X),\ \texttt{not}\ \texttt{CldRwy}(X)). \tag{13}$$

A model of an MKNF formula is defined as a maximal set of interpretations that satisfies the formula, such that no proper superset of this set also satisfies the formula [27]. KB_1 is satisfiable, if there exists a MKNF model for $\pi(KB_1)$ and KB_1 entails a first-order formula ψ, denoted as $KB_1 \models \psi$ if and only if $\pi(KB_1) \models_{MKNF} \psi$. The semantics of $\pi(KB_1)$ can be explained using either answer set [29] or well-founded semantics [28]. In the field of aeronautics, well-founded semantics are essential because they establish a single three-valued model comprising true, false, and undefined atoms for every logic program, thereby facilitating an efficient top-down query answering. To explain well-founded semantics for Hybrid MKNF, a three-valued MKNF interpretation is defined as a pair (M, N), where both M and N are sets of first-order interpretations with $N \subseteq M$. To avoid unintended consequences when integrating DLs with rules, the standard name assumption (SNA) is enforced, ensuring each constant uniquely identifies an individual and equality is explicitly handled using a equality predicate [29].

Truth evaluation of first-order atoms with modal operators, in relation to the three-valued MKNF model, proceeds as follows. An atomic formula $P(t_1, \ldots, t_n)$ is evaluated as *true* under a first-order interpretation I if the tuple $(t_1^I, \ldots, t_n^I)$ is an element of the interpretation of the predicate P in I, denoted P^I; formally, if $(t_1^I, \ldots, t_n^I) \in P^I$. Otherwise, the formula is evaluated as *false* when $(t_1^I, \ldots, t_n^I) \notin P^I$. The implication $\varphi_1 \supset \varphi_2$ is evaluated as *true* if the truth value of φ_2 is greater than or equal to that of φ_1 with respect to the order $f < u < t$, and *false* otherwise. The truth value of $K\varphi$ is *true* if φ evaluates to true in all interpretations in M, *false* if φ evaluates to false in some interpretation in $N \subseteq M$, and *undefined* otherwise. Similarly, not φ is *true* if φ is false in some interpretation in $N \subseteq M$, *false* if φ is true in all interpretations in M, and *undefined* otherwise [28].

An MKNF interpretation pair (M, N) satisfies a closed MKNF formula φ, denoted $(M, N) \models \varphi$, if and only if $(M, N)(\varphi) = t$, meaning that all rules within φ evaluate to true under this interpretation. An MKNF interpretation pair (M, N) is considered a three-valued MKNF model for a formula φ if it satisfies two key conditions. First, it must satisfy the formula φ itself. Second, it must be minimal in the sense that one cannot extend M or N to larger interpretation pairs that still satisfy φ. The only three-valued model of KB_1:`M = {{Rwy(rw1),OpnRwy(rw1)},{Rwy(rw1),OpnRwy(rw1),CldRwy(rw1)}},N = {{Rwy(rw1),OpnRwy(rw1)}}`.

Consider an MKNF formula φ. A partial three-valued MKNF model (M, N) is called the *well-founded model* of φ if and only if, for every other three-valued MKNF model (M_1, N_1) of φ, the relation $(M_1, N_1) \succeq (M, N)$ holds, meaning $M_1 \subseteq M$ and $N_1 \supseteq N$. For every MKNF formula there exist only one well-founded three-valued model.

4.2. Query Answering

The alternating fixed-point method provides a bottom-up strategy for computing the three-valued, well-founded model of a Hybrid MKNF knowledge base [28]. Initially, this model must be computed using

the proposed bottom-up approach. Once computed, it can be stored in a database for querying purposes. However, this method is impractical for large knowledge bases, particularly when only specific parts of information are relevant. To address this issue, a query answering algorithm called SLG(O) resolution has been proposed, which is both sound and complete with respect to the well-founded Hybrid MKNF model. SLG(O) resolution is a goal-directed query answering that employs SLG resolution to manage LP rules, where O acts as the oracle responsible for handling queries related to the DL component [30]. The resolution process begins with a query that is matched against the rule head, subsequently generating new subgoals derived from the body of that rule. When a subgoal corresponds to a DL atom, the system invokes the DL oracle to determine whether the atom is entailed by DL. Throughout the process, tabling is used to store the intermediate results, thereby preventing infinite loops and ensuring termination. Before implementing the SLG(O) resolution, a knowledge-based doubling step must be conducted to align with the bottom-up computation of the well-founded MKNF model [30]. Consider $KB = (O, P)$ be a Hybrid MKNF knowledge base and in [30] introduce new predicates A^d and NA for each predicate A appearing in KB, and then constructively define:

1. O^d by substituting each concepts or role A in O by A^d; and
2. P^d by transforming each rule of form (1) occurring in P into two rules:

   i $KH \leftarrow KA_1, \ldots, KA_n, not\ B_1^d, \ldots, not\ B_m^d$ and either,
   ii(a) $KH^d \leftarrow KA_1^d, \ldots, KA_n^d, not\ B_1, \ldots, not\ B_m, not\ NH$ if H is a DL concept or role; or
   ii(b) $KH^d \leftarrow KA_1^d, \ldots, KA_n^d, not\ B_1, \ldots, not\ B_m$ otherwise.

The doubled Hybrid MKNF knowledge base is defined as $KB^d = (O, O^d, P^d)$. In this transformation, each original predicate A in the knowledge base KB is paired with two predicates, A and A^d, where A^d indicates the non-falsity of A. In addition, a new predicate NA, semantically equivalent to the classical negation $\neg H$, is introduced. This predicate prevents the derivation of A^d when $\neg A$ is entailed in the ontology. To enforce this, $not\ NA$ is added to the body of every rule, with A^d as its head, ensuring that A^d can only be inferred when $\neg A$ is not supported by the ontology. This transformation process is called a *semi-negative transformation.* It aids in verifying MKNF consistency by identifying cases where H is true but A^d is not. In such instances, because A is true and A^d is false, it follows that $\neg A$ is entailed by DL ontology O, leading to inconsistency.

5. NoHr Reasoner

An existing implementation of Hybrid MKNF is the `NoHr` reasoner. `NoHr` employs SLG(O) resolution, which fundamentally translates the doubled Hybrid MKNF knowledge base into logic programming rules [31]. This method facilitates efficient reasoning through query answering using `XSB Prolog`. Specifically, the doubled Hybrid MKNF knowledge base, represented as $KB = (O, O^d, P^d)$, is converted into a semantically equivalent form: $KB = (\emptyset, P^d \cup P^O)$, where P^d is the doubled logic program, and P^O is the DL ontology translated into equivalent set of rules, followed by knowledge base doubling. The NoHr reasoner employs two distinct translation methodologies to convert DL into rules. First, a classification algorithm is used to deduce subsumption relationships, which are then translated into LP rules[32]. For classification, NoHr integrates three existing DL reasoners: ELK, Hermit, and Konclude, with Hermit and Konclude supporting more expressive DL constructors. Second, for the QL profile, a direct translation method was applied to convert DL axioms into Logic Programs without the need for prior classification[33]. We chose to focus on the classification-based translation with HermiT reasoner, as it supports more expressive DL constructors than the QL. The various steps involved in the NoHr reasoner with classification-based translator are illustrated in figure 2.

Consider DL axioms (7) and (8), which are translated into rules according to the NoHR proposition, followed by knowledge base doubling and the semi-negative transformation (see [32] for more details):

$$\mathrm{Rwy}(X) \leftarrow \mathtt{OpnRwy}(X). \quad (14)$$

$$\mathrm{Rwy}^d(X) \leftarrow \mathtt{OpnRwy}^d(X), not\ \mathtt{NRwy}(X). \quad (15)$$

$$\mathtt{NCldRwy}(X) \leftarrow \mathtt{OpnRwy}(X). \quad (16)$$

$$\mathtt{NOpnRwy}(X) \leftarrow \mathtt{CldRwy}(X). \quad (17)$$

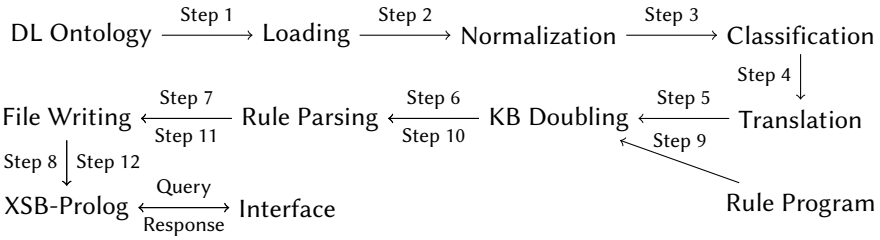

Figure 2: Different steps in NoHr Reasoner for EL Profile

The combination of knowledge base doubling and semi-negative transformation enables a paraconsistent-like behavior in the NoHR reasoner, allowing reasoning to continue even when parts of the knowledge base are inconsistent by isolating and reasoning over consistent fragments. In a Hybrid MKNF knowledge base, inconsistencies occur when both an atom H and its classical negation $\neg H$ are derivable. For example, consider adding the two assertions `OpnRwy(rw1)` and `cldRwy(rw1)`. According to DL axiom (8), these assertions are contradictory, as a runway cannot be both operational and closed simultaneously. After applying knowledge base doubling and the semi-negative transformation, these assertions are translated as follows:

$$\texttt{OpnRwy}^d(rw1) \leftarrow not\ \texttt{NOpnRwy}(rw1). \quad (18) \qquad \texttt{OpnRwy}(rw1).$$

$$\texttt{CldRwy}^d(rw1) \leftarrow not\ \texttt{NCldRwy}(rw1). \quad (19) \qquad \texttt{CldRwy}(rw1).$$

Using rule (14), we can infer `Rwy(rw1)`. However, according to rule (15), we cannot infer $\text{Rwy}^d(rw1)$ because `NOpnRwy(rw1)` holds, as established by rule (17). A literal H is considered *consistently true* only if both H and H^d are true. In this case, although $\text{Rwy}(rw1)$ is true, $\text{Rwy}^d(rw1)$ is false. Thus, the situation is treated as an inconsistency. Consequently, $\text{Rwy}(rw1)$ is excluded from the set of consistent query answers. By employing this syntactic approach, the NoHR reasoner effectively detect inconsistencies, ensuring that only consistent information is conveyed in response to queries, wherein an atom and its duplicate are considered true.

Below, we outline the consistency resolving techniques employed by the NoHr reasoner. Consider a conjunctive query of the form: $q(X) \leftarrow A_1, A_2, \ldots, A_n, not\ B_1, not\ B_2, \ldots, not\ B_m$ where $A_1, A_2, \ldots, A_n, not\ B_1, not\ B_2, \ldots, not\ B_m$ represent the conjunction of predicates being queried within the context of a Hybrid MKNF framework. To address the paraconsistent feature, the query is transformed into two distinct queries: $q(X) \leftarrow A_1, A_2, \ldots, A_n, not\ B_1^d, not\ B_2^d, \ldots, not\ B_m^d$ and $q^d(X) \leftarrow A_1^d, A_2^d, \ldots, A_n^d, not\ B_1, not\ B_2, \ldots, not\ B_m$. Consistent answers were identified by querying the prolog engine with: $q(X) \wedge q^d(X)$ While Inconsistent answers were determined using: $q(X) \wedge not\ q^d(X)$. This approach enables the system to manage contradictions by distinguishing between consistent and inconsistent results, to ensure that contradictions do not invalidate the reasoning process.

Evaluation: A thorough evaluation of the NoHr reasoner is crucial for its deployment in aeronautics, where real-time performance and reliability are essential. Key evaluation criteria include: (i) identifying the supported DL constructors to aid ontology modeling; (ii) measuring preprocessing time for ontologies and rules, to assess readiness after updates or reboots; (iii) analyzing query response latency, which must remain under one second due to safety-critical requirements; and (iv) assessing resource usage, as the reasoner may run on devices with limited memory and CPU capacity.

To analyze the preprocessing delay of the DL component, we utilized the OWL2Bench benchmark, which is an extension of LUBM dataset designed to generate DL ontologies of varying sizes across four OWL profiles: EL, QL, RL, and DL[34]. We use this benchmark to evaluate the DL part with the NoHr reasoner. For the logic program component, we employed the benchmark provided by the NoHr reasoner, which generates "n" number of rules based on an associated DL ontology. For each DL ontology generated from the OWL2Benchmark, utilizing the rule benchmark provided by the NoHR reasoner, rules are generated in quantities of 10, 100, 1,000, 10,000, and 25,000. The rule benchmark is available here:https://github.com/NoHRReasoner/NoHR/tree/master/nohr-benchmark

and OWL2Bench is available here:https://github.com/kracr/owl2bench. For query evaluation, we used the set of queries provided with the LUBM benchmark, making minor modifications to match the structure and vocabulary of the DL ontologies generated by the OWL2Bench benchmark. These were then converted into equivalent queries compatible with a prolog engine. Queries are available here for reference:https://swat.cse.lehigh.edu/projects/lubm/queries-sparql.txt.To evaluate memory consumption and CPU utilization, we use ontology files containing varying numbers of axioms. The measurements focus exclusively on the query answering phase of the NoHR reasoner.

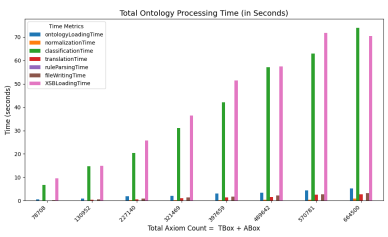

Figure 3: Ontology Processing

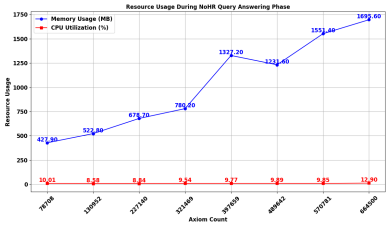

Figure 4: Resource consumption

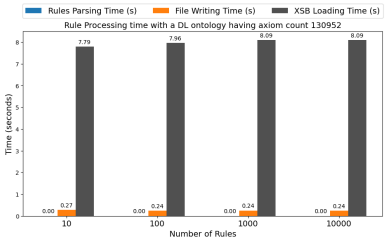

Figure 5: Rule Processing (78708 DL axioms)

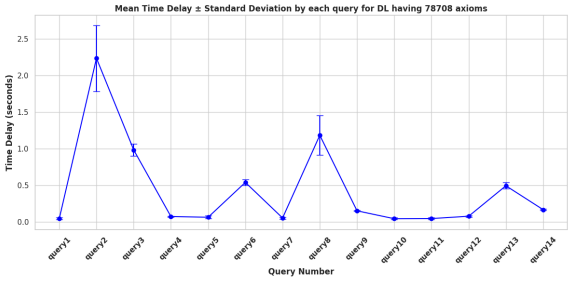

Figure 6: Average Query response time for 14 queries with a DL having 78708 axioms

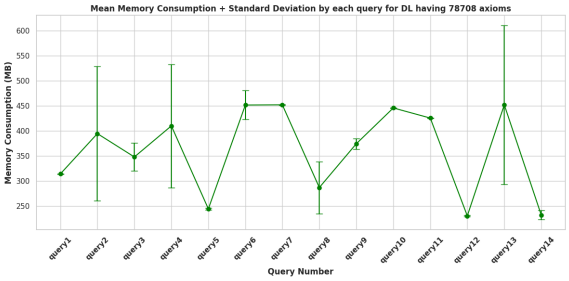

Figure 7: Average Memory Consumption for 14 queries with a DL having 78708 axioms

Evaluation Result of NoHr Reasoner

Evaluation Summary: During the ontology preprocessing phase, a considerable amount of time is devoted to ontology classification and XSB loading, both of which are significantly affected by the number of DL axioms (figure 2). In the rule processing stage, the XSB loading time is directly proportional to the size of the translated ontology (figure 4). During preprocessing, we also noted increased memory usage and CPU utilization, both of which were correlated with the size of the knowledge base (figure 3). Once preprocessing is complete, query answering with XSB-Prolog becomes highly efficient, particularly for selective queries that filter results based on specific conditions and return only a small subset of answers (figure 5). However, we observed a noticeable decline in the performance of queries that required reasoning over recursive rules. Certain DL constructors, such as *Equivalence Relationships* and *Transitive Properties*, are translated into recursive rules, which significantly affect performance. Furthermore, elevated memory usage persisted in the query-answering phase (figure 5 and figure 7).

6. Problem Statement

The primary objective of this thesis is to develop a KRR system tailored for the aeronautics domain. We have selected Hybrid MKNF under well-founded semantics from existing state-of-the-art approaches due to its strong reasoning capabilities and a prototype reasoner is already available. The next step is to adapt this framework for practical use in aeronautics context and to propose necessary extensions to enhance the feature of Hybrid MKNF. The following paragraph analyzes the main challenges involved in adapting Hybrid MKNF to meet these real-world requirements.

Problem 1: The NoHr reasoner is an existing system based on Hybrid MKNF under well-founded semantics. However, our evaluation shows that it consumes significant computational resources, which can limit its applicability in resource-constrained environments,which may restrict its practical use in aeronautics applications. To make it suitable for this domain, further optimization techniques are essential to enhance its performance and ensure efficient, reliable reasoning in real-world aeronautical environments.

Problem 2: One missing feature is the explicit representation of negative knowledge within the logic programming component of Hybrid MKNF. Consider the following use case: *A runway is operational only if there is no NOTAM indicating that it is closed, and no obstacle is present on the runway.* This can be initially represented by the following MKNF rule:

$$K\,\texttt{OpnRwy(X)} \leftarrow K\,\texttt{Rwy(X)},\ not\ \texttt{ob(A,X)},\ not\ \texttt{CldRwy(X)}. \tag{20}$$

However, due to the safety-critical nature of aeronautics, the predicate `ob(A,X)` which indicate that there exist an obstacle for a runway X in an airport A, carries particular importance. Using `not` operator with `ob(A,X)` to infer that a runway is operational introduces safety concerns because it relies on the absence of evidence rather than explicit confirmation. Such assumptions are unacceptable in this domain. Instead, we must explicitly verify that the runway is free of obstacles, thereby enhancing the reliability of the system. One way to achieve this is by replacing the `not` operator with classical negation denoted as $\neg$.

Classical negation can be incorporated into existing Hybrid MKNF semantics via syntactic transformation for unary and binary predicates. For a unary predicate A, we introduce a new concept B with the axiom $B \equiv \neg A$ in DL, allowing B to stand for the classical negation of A within rules. For a binary predicate P, a corresponding predicate Q is created, and the disjointness axiom $P \sqcap Q \sqsubseteq \bot$ enforces that Q represents the classical negation of P. However, this approach requires additional knowledge engineering and is limited to unary and binary predicates. Extending the rule component with classical negation enhances expressiveness, simplifies knowledge engineering, and facilitates the modeling of critical scenarios, such as the one illustrated above. After the extension of classical negation , the rule (20) can be rewritten as:

$$K\,\texttt{OpnRwy(X)} \leftarrow K\,\texttt{Rwy(X)}, K\neg\texttt{ob(A,X)},\ not\ \texttt{CldRwy(X)}. \tag{21}$$

Problem 3: Another crucial feature currently lacking is the capability to represent integrity constraints within the knowledge base and enforce them during query answering. Consider NARS use case, many NOTAMs are incomplete or invalid. A NOTAM may lack crucial information such as location, effective time, or activity description. These incomplete NOTAMs must be excluded from the reasoning process to avoid producing misleading or inconsistent conclusions. To handle this, we formally define the conditions that constitute a valid NOTAM and represent them as integrity constraints in the knowledge base. This allows the reasoning engine to automatically detect and disregard invalid NOTAMs before any inference or query answering is performed.

Problem 4: Providing justifications for all decisions derived from Hybrid MKNF reasoning is vital, especially given the safety-critical nature of aeronautics. It is essential to develop formal methods that can provide clear and reliable explanations for each query answering result, capabilities that are currently unavailable.

7. Results

As a first step, we began addressing the problems explained in Section 4 by leveraging existing state-of-the-art methods. Optimizing a reasoner for hybrid MKNF, while proposing extensions such as classical negation and integrity constraints, involves substantial theoretical work. Therefore, as an initial approach, we proposed extensions based on syntactic transformations, allowing us to solve these problems within the existing semantics and provide an initial solution.

Handling Classical Negation: A syntactic transformation inspired from [35, 36], We extend Hybrid MKNF knowledge bases $KB = (O, P)$ to support classical negation in P by replacing each negated atom $\neg L$ with a fresh predicate L' and adding constraints $\bot \leftarrow L, L'$ to ensure consistency. This leads to a modified knowledge base, $K' = (O, P' \cup C)$ where P' is updated program by replacing the classically negated atoms and constraints C in the form $\bot \leftarrow L, L'$.Subsequently, knowledge base doubling and semi-negative transformation are applied as described in [30], resulting in the doubled knowledge base $K'^d = (O, O^d, P'^d \cup C^d)$, as outlined below:

1. O^d is obtained by substituting each predicate A in O with A^d;
2. P^d is derived by transforming each rule of the form (1) occurring in P into two rules:
   i $KH \leftarrow KA_1, \ldots, KA_i, \texttt{not}\ B_1^d, \ldots, \texttt{not}\ B_j^d$;
   ii $KH^d \leftarrow KA_1^d, \ldots, KA_i^d, \texttt{not}\ B_1, \ldots, \texttt{not}\ B_j, \texttt{not}\ NH$;
3. C^d is derived from each constraint of the form $\bot \leftarrow L, L'$ in C is transformed into two rules: $NL \leftarrow K\ L'$ and $NL' \leftarrow K\ L$.

where A_i, A_i^d, B_j, B_j^d, H, and H^d may represent either L or L'.

We hypothesize that this syntactic transformation maintains semantic equivalence, consistent with the existing state of the art. A formal explanation and theoretical study, including proof, are ongoing and planned as part of future work. This transformation enables the use of existing Hybrid MKNF query answering methods. After the knowledge base doubling, we can use consistent and inconsistent query answering with the steps defined in Section 5.

Consider the Hybrid MKNF knowledge base $KB = (O, P)$, where O contains an assertion $\texttt{Rwy(rw1)}$ and P includes rules (21) and a fact: $K\neg\texttt{ob(ifbo, rw1)}$. These rules can be transformed into the following rules as per the syntactic transformation, resulting in the modified knowledge base $KB' = (O, P' \cup C)$ as shown below(only logic programming part is provided):

$$K\texttt{OpnRwy(X)} \leftarrow K\texttt{Rwy}(X), K\texttt{ob}'(A, X),\ not\ \text{CldRwy}(X). \tag{22}$$

$$\bot \leftarrow K\texttt{ob}(A, X), K\texttt{ob}'(A, X). \tag{23}$$

$$K\texttt{ob}'(lfbo, rw1). \tag{24}$$

After knowledge base doubling and semi-negative transformation we get $KB'^d = (O, O^d, P'^d \cup C^d)$:

$$K\texttt{opnRwy(X)} \leftarrow K\texttt{rwy}(X), K\texttt{ob}'(A, X),\ not\ \texttt{cldRwy}^d(X). \tag{25}$$

$$K\text{opnRwy}^d(X) \leftarrow K\texttt{rwy}^d(X), K\texttt{ob}'^d(A, X),\ not\ \texttt{cldRwy}(X), not\ \texttt{NopnRwy}(X). \tag{26}$$

$$K\texttt{Nob}(A, X) \leftarrow K\texttt{ob}'(A, X). \tag{27}$$

$$K\texttt{Nob}'(A, X) \leftarrow K\texttt{ob}(A, X). \tag{28}$$

$$K\texttt{ob}'(lfbo, rw1). \tag{29}$$

$$K\texttt{ob}'^d(lfbo, rw1) \leftarrow not\ \texttt{Nob}'(lfbo, rw1). \tag{30}$$

After applying all transformations, we obtain the modified knowledge base $KB'^d = (O, O^d, P'^d \cup C^d)$,where all occurrences of classical negation within the rules are syntactically eliminated. By adding the assertion $\texttt{rwy(rw1)}$, we can infer $\texttt{opnRwy(rw1)}$ because there is an explicit negative fact indicating the absence of an obstacle, specifically $\neg\texttt{ob(lfbo, rw1)}$, and no closure is indicated. Now, consider a scenario of inconsistency where an additional assertion $\texttt{ob(lfbo, rw1)}$ is introduced, which also needs to be doubled. In this case, $\texttt{ob(lfbo, rw1)}$ becomes inconsistent because both the atom and its classical negation are simultaneously true. In such a situation, the rules defined in (27) and (28) ensure that both $\texttt{Nob}(X, Y)$ and $N\texttt{ob}'(X, Y)$ are inferred to be true. Consequently, due to the semi-negative transformation, both $\texttt{ob}'^d(lfbo, rw1)$ and $\texttt{ob}^d(lfbo, rw2)$ are inferred as false. This allows to detect that atom $\texttt{ob(lfbo, rw1)}$, along with its duplicate $\texttt{ob'(lfbo, rw2)}$, is inconsistent. Recall that an atom A is considered consistently true only if both A and its duplicate A^d are true. Then consistent and inconsistent query answering mechanism described earlier can be used to correctly handle such situations and provide only consistent answers.

Integrity Constraints: A syntactic approach to enforcing integrity constraints is also proposed, inspired by [37, 38] on using classical negation [39]. The idea is to treat any data that violates a specified condition as inconsistent. For example, if a NOTAM message lacks essential information such as a start or end time, it should be considered invalid. This can be captured with a rule that states:

$$K\neg\texttt{Notam}(X) \leftarrow K\texttt{Notam}(X),\ not\ \texttt{hasStartTime}(X,Y). \tag{31}$$

This rule means that if a NOTAM exists, but we cannot find any associated start time or end time, then its negation is derived, creating an inconsistency. When this happens, both the original NOTAM and its negation coexist in the knowledge base. During consistent query answering, such inconsistent atoms are filtered out(see Section 5).

Optimize Resource Consumption: NoHr operates by translating a Hybrid MKNF knowledge base into an equivalent logic program, using a sequence of transformation algorithms(figure 1). This logic program is then executed by the XSB-Prolog engine to perform query answering. To mitigate memory consumption, we propose using NoHr exclusively during a preprocessing phase to generate the transformed logic program. During system operation, only the resulting logic program is executed independently with a prolog engine such XSB-Prolog or SWI-Prolog. This approach significantly reduces runtime memory usage while maintaining the reasoning capabilities, as the live system no longer depends on the full NoHr infrastructure.We can also integrate the syntactic approach proposed for combining classical negation with integrity constraints into this. The architecture of the proposed work flow is available in figure 8:

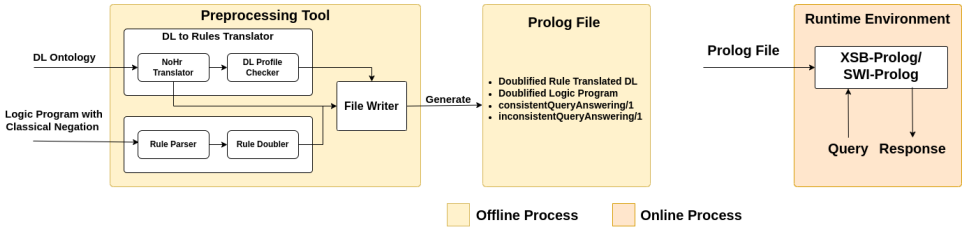

Figure 8: Architecture of Proposed Workflow

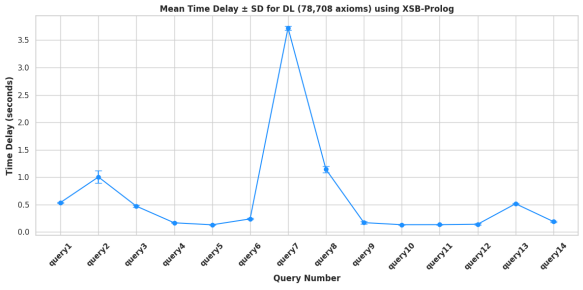

Figure 9: Response Time for Each Query

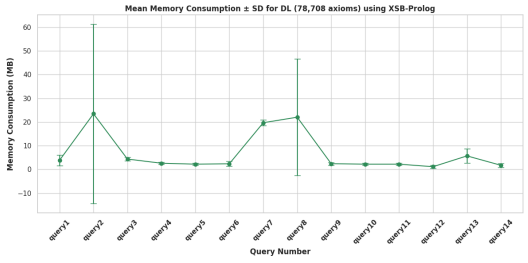

Figure 10: Memory Consumption for Each Query

The results were submitted to the International Conference on Logic Programming 2025 and the paper was accepted.

8. Current Work

The syntactic approach outlined above provides a straightforward means of incorporating classical negation into an LP while enabling reuse of the NoHR reasoner. However, this approach remains heuristic in nature, and a more rigorous account of the semantics, supported by formal proofs, is required. Our next objective is therefore to define a formal semantics for the extended Hybrid MKNF framework, with particular emphasis on the truth evaluation of modal operators involving classically negated atoms, such as $K\neg A$ and not $\neg A$, within the context of a three-valued model.

Building on this foundation, we will introduce a bottom-up computational methodology that generalizes the existing alternating fixpoint approach to compute the extended (T, F) partition, which incorporates both positive and classically negated literals. We will formally prove that the resulting partition is sound and complete with respect to the extended well-founded semantics. Subsequently, we will develop a query answering algorithm extending the SLG(O) resolution method discussed in Section 3.2 of [30]. This algorithm will employ knowledge base doubling and semi-negative transformation techniques to identify inconsistencies and filter out contradictory atoms during query evaluation, thereby ensuring paraconsistent reasoning capabilities. We will also provide a formal proof that the proposed query answering procedure is sound and complete with respect to the extended well-founded (T, F) partition. Finally, we will implement the extended query answering algorithm for specific DLs.

The theoretical contributions, including the formal semantics, computational methodology, and query answering algorithm, will be submitted as a full paper. In parallel, we will submit the implementation results, focusing on query answering over specific DLs, as a short paper to the Automated Reasoning Conference 2026, whose submission deadline is expected around January or February.

9. Future Research Directions

This section outlines the upcoming steps following the completion of the current task—extending the hybrid MKNF framework with classical negation. Once this work is finalized and the results are submitted to a conference, We will select the next objective from the list below to pursue. The objectives are presented in order of priority, with the first being the most immediate focus after the current phase. The scope of these objective will be carried within the phd objective or beyond depends on the time.

Explore rule engines: Our goal is to construct a practical reasoner capable of running efficiently on resource-constrained devices. Currently, reasoning is performed by translating a hybrid MKNF knowledge base into a Prolog program with tabling, using XSB-Prolog for query answering. However, by restricting the syntax, we can identify several useful subsets of hybrid MKNF, enabling the use of a wider range of solvers and also facilitating the development of lightweight, practical reasoners. For instance, DLs can be integrated with definite logic programs—programs consisting only of positive rules—and with stratified programs, where predicates are organized into layers (strata) such that no predicate depends negatively on itself, either directly or indirectly. Additionally, reasoning can be extended to unrestricted, unstratified programs without such constraints. In the cases of definite logic programs and stratified programs, there are existing solvers such as I-DLV[1], Trealla Prolog[2], Tau-Prolog [3] which are more lightweight. We will formally define these syntactic subsets of Hybrid MKNF and conduct a detailed evaluation of various rule engines applicable to each subset. In addition to this, we aim to identify lightweight solvers that offer practical performance benefits for constrained environments. This objective enables knowledge engineers to select a reasoner that best aligns with the specific requirements of the application.

Explanation: Another important objective is to develop a formal theory for explaining the reasoning process in Hybrid MKNF. Provenance, a concept in logic programming, has been widely used to generate justifications for query answering results[40]. Inspired by this approach, we will propose an extension of provenance techniques to Hybrid MKNF knowledge bases under the well-founded semantics.

Paraconsistent Semantics: Extending Hybrid MKNF with classical negation introduces potential inconsistencies. A paraconsistent semantics for Hybrid MKNF under the well-founded semantics has already been proposed [41], but no implementation currently supports it. Extending this semantics to accommodate classical negation within the LP component, and devising a sound and complete query answering algorithm for the extended framework, constitute important directions for future research, particularly given their practical relevance in safety-critical domains such as aeronautics.

[1] https://dlv.demacs.unical.it/i-dlv
[2] https://github.com/trealla-prolog/trealla
[3] https://tau-prolog.org/

Date-and-Time Reasoning: Another important research direction is the integration of date-and-time reasoning within the reasoner. For instance, in the NARS use case, NOTAM messages are associated with a start and end time. Therefore, the reasoner must take these date-and-time constraints into account when performing inference.

Acknowledgments

I express my sincere thanks to my supervisors: Christophe REY(Université Clermont-Auvergne), Florence DE GRANCEY(Thales), Victor CHARPENAY(Ecole des Mines de Saint-Étienne), and Farouk TOUMANI(Université Clermont-Auvergne) for their guidance and assistance thus far. This research is jointly funded by Thales and National Association of Research and Technology(ANRT),France.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT (OpenAI, GPT-4) to assist with grammar and spelling correction, as well as to improve clarity and readability. After using this tool, the author(s) reviewed and edited the content as needed and take full responsibility for the final version of the manuscript.

References

[1] F. Baader, D. L. McGuinness, D. Nardi, P. F. Patel-Schneider, The description logic handbook: Theory, implementation, and applications, 2003. URL: https://api.semanticscholar.org/CorpusID:35467721.

[2] J. Schulze, Handbook of logic in artificial intelligence and logic programming, 2016. URL: https://api.semanticscholar.org/CorpusID:63249517.

[3] T. Eiter, G. Ianni, T. Krennwallner, A. Polleres, Rules and ontologies for the semantic web, volume 5224, 2008, pp. 1–53. doi:10.1007/978-3-540-85658-0_1.

[4] A. V. Gelder, K. A. Ross, J. S. Schlipf, The well-founded semantics for general logic programs, J. ACM 38 (1991) 620–650. URL: https://api.semanticscholar.org/CorpusID:12753612.

[5] T. Eiter, G. Ianni, T. Krennwallner, Asp: A primer, in: Reasoning Web, 2009. URL: https://api.semanticscholar.org/CorpusID:1078190.

[6] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, H. Tompits, Combining answer set programming with description logics for the semantic web, Artificial Intelligence 172 (2008).

[7] T. Eiter, G. Ianni, T. Lukasiewicz, R. Schindlauer, Well-founded semantics for description logic programs in the semantic web, 2004. URL: https://api.semanticscholar.org/CorpusID:2347075.

[8] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming., 2005.

[9] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, Dlv-hex : Dealing with semantic web under answer-set programming , 2005. URL: https://api.semanticscholar.org/CorpusID:9592774.

[10] G. Xiao, T. Eiter, S. Heymans, The drew system for nonmonotonic dl-programs, in: China Semantic Web Symposium, 2012. URL: https://api.semanticscholar.org/CorpusID:8877458.

[11] S. Heymans, T. Eiter, G. Xiao, Tractable reasoning with dl-programs over datalog-rewritable description logics, in: European Conference on Artificial Intelligence, 2010.

[12] M. Krotzsch, Efficient rule-based inferencing for owl el, in: International Joint Conference on Artificial Intelligence, 2011. URL: https://api.semanticscholar.org/CorpusID:14915021.

[13] S. Heymans, R. Korf, M. Erdmann, J. Pührer, T. Eiter, F-logic: Loosely coupling f-logic rules and ontologies, 2010, pp. 248–255. doi:10.1109/WI-IAT.2010.44.

[14] R. Rosati, On the decidability and complexity of integrating ontologies and rules, J. Web Semant. 3 (2005) 61–73. URL: https://api.semanticscholar.org/CorpusID:7666586.

[15] R. Rosati, Dl+log: Tight integration of description logics and disjunctive datalog, in: International Conference on Principles of Knowledge Representation and Reasoning, 2006.

[16] W. Drabent, J. Henriksson, J. Maluszynski, Hd-rules: A hybrid system interfacing prolog with dl-reasoners, in: Applications of Logic Programming to the Web, 2007.
[17] L. Bajraktari, M. Ortiz, M. imkus, Combining rules and ontologies into clopen knowledge bases, in: AAAI Conference, 2018. URL: https://api.semanticscholar.org/CorpusID:19234928.
[18] S. Lukumbuzya, M. Ortiz, M. imkus, Resilient logic programs: Answer set programs challenged by ontologies, in: AAAI Conference on Artificial Intelligence, 2020.
[19] S. Heymans, D. V. Nieuwenborgh, D. Vermeir, Open answer set programming for the semantic web, J. Appl. Log. 5 (2007) 144–169. URL: https://api.semanticscholar.org/CorpusID:7957259.
[20] G. Antoniou, A. Bikakis, Dr-prolog: A system for defeasible reasoning with rules and ontologies on the semantic web, IEEE Transactions on Knowledge and Data Engineering 19 (2007). URL: https://api.semanticscholar.org/CorpusID:14844582.
[21] H. Lan, Swrl : A semantic web rule language combining owl and ruleml, , 2004. URL: https://api.semanticscholar.org/CorpusID:207974723.
[22] F. Donini, M. Lenzerini, D. Nardi, Al-log: Integrating datalog and description logics, Journal of Intelligent Information Systems 10 (1999). doi:10.1023/A:1008687430626.
[23] Combining horn rules and description logics in carin, Artificial Intelligence 104 (1998) 165–209.
[24] C. Feier, S. Heymans, Hybrid reasoning with forest logic programs, in: Extended Semantic Web Conference, 2009. URL: https://api.semanticscholar.org/CorpusID:11089946.
[25] S. Heymans, L. Predoiu, C. Feier, J. de Bruijn, D. V. Nieuwenborgh, G-hybrid knowledge bases, 2003. URL: https://api.semanticscholar.org/CorpusID:9440798.
[26] B. Motik, R. Rosati, A faithful integration of description logics with logic programming, in: Proceedings of the 20th IJCAI, IJCAI'07, 2007.
[27] V. Lifschitz, Nonmonotonic databases and epistemic queries, in: Proceedings of the 12th International Joint Conference on Artificial Intelligence - Volume 1, IJCAI'91, 1991.
[28] M. Knorr, J. J. Alferes, P. Hitzler, Local closed world reasoning with description logics under the well-founded semantics, Artif. Intell. 175 (2011) 1528–1554.
[29] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (2008). URL: https://doi.org/10.1145/1754399.1754403. doi:10.1145/1754399.1754403.
[30] J. J. Alferes, M. Knorr, T. Swift, Query-driven procedures for hybrid mknf knowledge bases, ArXiv abs/1007.3515 (2010). URL: https://api.semanticscholar.org/CorpusID:3126629.
[31] V. Kasalica, Nohr: An overview, KI - Knstliche Intelligenz (2020) 1–7. URL: https://api.semanticscholar.org/CorpusID:212408100.
[32] V. Ivanov, M. Knorr, J. Leite, A query tool for el with non-monotonic rules, in: International Workshop on the Semantic Web, 2013. URL: https://api.semanticscholar.org/CorpusID:16285536.
[33] N. Costa, M. Knorr, J. Leite, Next step for nohr: Owl 2 ql, in: International Workshop on the Semantic Web, 2015. URL: https://api.semanticscholar.org/CorpusID:41958726.
[34] G. Singh, S. K. Bhatia, R. Mutharaju, Owl2bench: A benchmark for owl 2 reasoners, in: International Workshop on the Semantic Web, 2020. URL: https://api.semanticscholar.org/CorpusID:226229397.
[35] V. Lifschitz, D. Pearce, A. Valverde, Strongly equivalent logic programs, ACM Trans. Comput. Logic 2 (2001) 526–541. URL: https://doi.org/10.1145/383779.383783. doi:10.1145/383779.383783.
[36] J. Alferes, L. Pereira, T. Przymusinski, 'classical' negation in nonmonotonic reasoning and logic programming, J. Autom. Reasoning 20 (1998) 107–142. doi:10.1023/A:1005900924623.
[37] S. Flesca, S. Greco, E. Zumpano, Active integrity constraints, PPDP '04, Association for Computing Machinery, New York, NY, USA, 2004, p. 98–107.
[38] L. e. a. Cruz-Filipe, Active integrity constraints: From theory to implementation, in: Knowledge Discovery, Knowledge Engineering and Knowledge Management, Cham, 2016, pp. 399–420.
[39] J. Chomicki, G. Saake, P. Godfrey, J. Grant, J. Gryz, J. Minker, Integrity constraints: Semantics and applications (1998).
[40] C. V. Damásio, A. Analyti, G. Antoniou, Justifications for logic programming, in: International Conference on Logic Programming and Non-Monotonic Reasoning, 2013.
[41] T. Kaminski, Efficient paraconsistent reasoning with rules and ontologies for the semantic web, 2014. URL: https://api.semanticscholar.org/CorpusID:9076559.