Closing Semantic Web Ontologies

Boris Motik[†] and Riccardo Rosati[‡]

[†] Department of Computer Science
University of Manchester
Manchester, UK
[‡] Dipartimento di Informatica e Sistemistica
Università di Roma "La Sapienza"
Rome, Italy

March 7, 2007

Abstract

In this paper, we present a novel formalism of *hybrid MKNF knowledge bases*, which allows us to seamlessly integrate an arbitrary decidable description logic with logic programming rules. We thus obtain a powerful hybrid formalism that combines the best features of both description logics, such as the ability to model taxonomic knowledge, and logic programming, such as the ability to perform nonmonotonic reasoning. Extending DLs with unrestricted rules makes reasoning undecidable. To obtain decidability, we apply the well-known *DL-safety* restriction that makes the rules applicable only to explicitly named individuals, and thus trade some expressivity for decidability. We present several reasoning algorithms for different fragments of our logic, as well as the corresponding complexity results. Our results show that, in many cases, the data complexity of reasoning with hybrid MKNF knowledge bases is not higher than the data complexity of reasoning in the corresponding fragment of logic programming.

Contents

1 Introduction **4**

2 Preliminaries **5**
2.1 Minimal Knowledge and Negation as Failure 5
2.2 Description Logics 9
2.3 Complexity Classes 10
2.4 Quantified Boolean Formulae 11

3 Equivalent Transformations of MKNF Formulae **11**
3.1 Equivalent Formulae 11
3.2 Adding Modal Operators to the Theory and the Query 13
3.3 Introducing Definitions 13
3.4 Reducing Entailment to Unsatisfiability 14

4 Extending DLs with MKNF Rules **15**
4.1 MKNF Rules 15
4.2 DL-Safety 18
4.3 Flat vs. Nonflat Rules 22
4.4 Relationship with $\mathcal{DL}$+log 23

5 Example **26**

6 Reasoning with Modally Closed MKNF Formulae **28**
6.1 Overview 29
6.2 The General Case 29
6.3 The Positive Case 33
6.4 The Positive Nondisjunctive Case 35
6.5 The Stratified Nondisjunctive Case 38
6.6 The Nonstratified Nondisjunctive Case 41

7 Reasoning with Hybrid Knowledge Bases **42**
7.1 The General Case 42
7.2 The Positive Case 45
7.3 The Positive Nondisjunctive Case 45
7.4 The Stratified and the Nonstratified Cases 46

8 Data Complexity **47**
8.1 Positive Nondisjunctive Programs 47
8.2 Stratified Nondisjunctive Programs 48
8.3 Nonstratified Nondisjunctive Programs 51
8.4 Positive Programs 53
8.5 General Programs 53

9 Reusing QBF Solvers for Reasoning in MKNF **53**
9.1 Obtaining a Propositional MKNF Formula 54
9.2 Reducing a Flat Propositional MKNF Formula into QBF . . 56

10 Conclusion **58**

1 Introduction

In the past couple of years, a significant body of Semantic Web research was devoted to defining a suitable language for ontology modeling. In 2004, this endeavor resulted in the Web Ontology Language (OWL). OWL is based on Description Logics (DLs) [2]—a family of knowledge representation formalisms based on first-order logic and exhibiting well-understood computational properties.

The experience gained by building practical applications has revealed several shortcomings of OWL. For example, OWL does not allow defining integrity constraints or closed-world reasoning. Rule-based formalisms grounded in logic programming have repeatedly been proposed as a possible solution, so adding a rule layer on top of OWL is nowadays seen as the most important task in the development of the Semantic Web language stack.

The logic of minimal knowledge and negation as failure (MKNF) [25] is an expressive formalism that allows to integrate open- and closed-world reasoning in a unifying framework. It generalizes several important nonmonotonic formalisms, such as logic programming under stable model semantics [16], and default logic with fixed universe [24] (note that this version of default logic is different from the original version by Reiter [30]).

Using MKNF as the unifying framework, in this paper we present a novel formalism of *hybrid MKNF knowledge bases* that allows to extend an arbitrary decidable first-order fragment with rules in the style of logic programming. Our formalism generalizes existing known approaches for extending DLs with first-order rules, such as the Semantic Web Rule Language (SWRL) [18]. Since combinations of DLs and rules are known to be undecidable even for very inexpressive DLs [23], to achieve decidability we apply the well-known concept of *DL-safety* [27] and thus trade some expressivity for decidability. Furthermore, we employ a special approach to dealing with the uniqueness of individuals, thus bridging the gap between logic programs, which usually assume unique name assumption (UNA), and DLs, which usually do not assume it. Thus, our approach is fully compatible with logic programming on the one hand and DLs on the other hand.

We present several algorithms for checking entailment for different classes of MKNF knowledge bases. Apart from the algorithm for the full formalism, we additionally consider the cases when the rules are positive, positive nondisjunctive, stratified nondisjunctive, and nonstratified nondisjunctive. We also analyze the data complexity of our algorithms and show that a combination of a DL with disjunctive datalog rules has the same data complexity as disjunctive datalog, even if entailment checking in DL is data complete for coNP (this is the case for most expressive DLs, such as $\mathcal{SHIQ}$ [20]).

Since most variants of our algorithm are intractable with respect to data complexity, implementing them requires advanced heuristic techniques. Similar techniques have already been developed for reasoning with Quanti-

fied Boolean Formulae (QBF) and have been implemented in several QBF solvers. To enable reusing these techniques, for the case of the DL $\mathcal{SHIQ}$, we present a way to encode our algorithms into a quantified Boolean formula. This encoding reuses the known technique for reducing a $\mathcal{SHIQ}$ knowledge base to a positive disjunctive datalog program [19].

2 Preliminaries

2.1 Minimal Knowledge and Negation as Failure

The logic of minimal knowledge and negation as failure (MKNF) [25] is an extension of first-order logic with modal operators **K** and **not**, and it is strongly related to the logic of minimal belief and negation as failure (MBNF) [26]. The syntax of MKNF formulae is defined by the following grammar, where t_i are first-order terms and P is a predicate:

$$\varphi \leftarrow P(t_1, \ldots, t_n) \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \exists x : \varphi \mid \mathbf{K}\,\varphi \mid \mathbf{not}\,\varphi$$

As usual, $P(t_1, \ldots, t_n)$ is a *first-order atom*; furthermore, $\varphi_1 \vee \varphi_2$, $\forall x : \varphi$, $\varphi_1 \supset \varphi_2$, $\varphi_1 \equiv \varphi_2$, true, and false are shortcuts for $\neg(\neg\varphi_1 \wedge \neg\varphi_2)$, $\neg(\exists x : \neg\varphi)$, $\neg\varphi_1 \vee \varphi_2$, $(\varphi_1 \supset \varphi_2) \wedge (\varphi_2 \supset \varphi_1)$, $a \vee \neg a$, and $a \wedge \neg a$, respectively. A formula of the form $\mathbf{K}\,\varphi$ is called a *modal* $\mathbf{K}$*-atom*, and a formula of the form $\mathbf{not}\,\varphi$ is called a *modal* **not***-atom*; collectively, modal **K**- and **not**-atoms are called *modal atoms*. An occurrence of a modal atom in an MKNF formula is *strict* if the atom does not occur in scope of a modal operator. An MKNF formula φ is a *sentence* if it has no free variables; φ is *ground* if it does not contain variables; φ is *modally closed* if all modal operators are applied in φ only to sentences; φ is *subjective* if all first-order atoms of φ occur within the scope of a modal operator; φ is *flat* if all occurrences of modal atoms in φ are strict and φ is subjective; φ is *positive* if it does not contain occurrences of **not**; φ is *objective* if it does not contain modal operators. With $\varphi[t_1/x_1, \ldots, t_n/x_n]$ we denote the formula obtained by simultaneously replacing in φ all free variables x_i with the terms t_i, and with $|\varphi|$ we denote the number of symbols needed to encode φ.

We call an MKNF formula of the following form a *rule*, as it generalizes the notion of a rule in disjunctive logic programs interpreted under stable model semantics [25]:

$$(1) \quad \mathbf{K}\,H_1 \vee \ldots \vee \mathbf{K}\,H_k \subset \mathbf{K}\,B_1^+ \wedge \ldots \wedge \mathbf{K}\,B_n^+ \wedge \mathbf{not}\,B_1^- \wedge \ldots \wedge \mathbf{not}\,B_m^-$$

The set of modal atoms $\mathsf{head}(r) = \{\mathbf{K}\,H_i \mid 1 \leq i \leq k\}$ is called the rule *head*, the set of modal atoms $\mathsf{body}^+(r) = \{\mathbf{K}\,B_i^+ \mid 1 \leq i \leq n\}$ is the *positive body*, and the set of modal atoms $\mathsf{body}^-(r) = \{\mathbf{not}\,B_i^- \mid 1 \leq i \leq m\}$ is called the *negative body*. If $k = 1$, we say that the rule is *nondisjunctive*; to simplify the notation, we then denote $\mathbf{K}\,H_1$ with $\mathbf{K}\,H$ and we set $\mathsf{head}(r) = \mathbf{K}\,H$

(i.e., $\mathsf{head}(r)$ returns the head atom and not a singleton set). Note that the common case of rules with no atoms in the head is captured by a nondisjunctive rule with the head atom $\mathbf{K}\,\mathsf{false}$. If $n = 0$ and $m = 0$, we call the rule a *fact*. A rule is *safe* if each variable in it occurs in a $\mathbf{K}$-atom in the body. A flat modally closed MKNF formula σ of the form $\bigwedge r_i$, where r_i are rules, is called an *MKNF program*. By an analogy to logic programming, we often identify σ with a set of rules $\{r_i\}$, and we write $r \in \sigma$ to denote that r is a conjunct or σ. An MKNF program is *nondisjunctive* (*safe*) if all its rules are nondisjunctive (*safe*).

Following [8], we employ *standard names* in the semantics of MKNF: we assume that, apart from the constants occurring in the formulae, the signature contains a countably infinite supply of constants not occurring in the formulae. With $\triangle$ we denote the Herbrand universe of such a signature and call it the *domain set*. An *MKNF structure* is a triple (I, M, N), where I is a Herbrand first-order interpretation over $\triangle$, and M and N are nonempty sets of Herbrand first-order interpretations over $\triangle$ satisfying the following property: the equality predicate $\approx$ is interpreted in I and in each interpretation from M and N as a congruence relation—that is, it is reflexive, symmetric, transitive, and it allows one to replace equals by equals (see [15, Chapter 9] for an in-depth discussion about these issues). Satisfiability of MKNF sentences in an MKNF structure (I, M, N) is defined as follows:

$$
\begin{array}{ll}
(I,M,N) \models P(t_1,\ldots,t_n) & \text{iff } P(t_1,\ldots,t_n) \text{ is true in } I \\
(I,M,N) \models \neg\varphi & \text{iff } (I,M,N) \not\models \varphi \\
(I,M,N) \models \varphi_1 \wedge \varphi_2 & \text{iff } (I,M,N) \models \varphi_1 \text{ and } (I,M,N) \models \varphi_2 \\
(I,M,N) \models \exists x : \varphi & \text{iff } (I,M,N) \models \varphi[\alpha/x] \text{ for some } \alpha \in \triangle \\
(I,M,N) \models \mathbf{K}\,\varphi & \text{iff } (J,M,N) \models \varphi \text{ for all } J \in M \\
(I,M,N) \models \mathbf{not}\,\varphi & \text{iff } (J,M,N) \not\models \varphi \text{ for some } J \in N
\end{array}
$$

An *MKNF interpretation* M is a nonempty set of Herbrand first-order interpretations over $\triangle$. M is an $\mathsf{S5}$ *model* of a closed MKNF formula φ, written $M \models \varphi$, if $(I, M, M) \models \varphi$ for each $I \in M$. As its name suggests, $\mathsf{S5}$ models of φ are obtained by interpreting φ as a first-order modal formula in the modal logic $\mathsf{S5}$ (while taking **not** to be a shortcut for $\neg\mathbf{K}$).

The $\mathsf{S5}$ semantics is monotonic, and a nonmonotonic semantics of MKNF is obtained by introducing a *preference relation* on $\mathsf{S5}$ models. In particular, an MKNF interpretation M is an MKNF model of φ if (i) M is an $\mathsf{S5}$ model of φ, and (ii) for each MKNF interpretation M' such that $M' \supset M$, we have $(I', M', M) \not\models \varphi$ for some $I' \in M'$.

An MKNF formula φ is *valid* if $(I, M, N) \models \varphi$ for any MKNF structure (I, M, N); φ is MKNF *satisfiable* if an MKNF model of φ exists; otherwise, it is MKNF *unsatisfiable*. MKNF formulae φ and ψ are MKNF equisatisfiable if φ is MKNF satisfiable if and only if ψ is MKNF satisfiable. Furthermore, φ MKNF entails ψ, written $\varphi \models_{\mathsf{MKNF}} \psi$, if $(I, M, M) \models \psi$ for each MKNF model M of φ and $I \in M$. The $\mathsf{S5}$ (un)satisfiability, equisatisfiability, and

entailment (written $\varphi \models_{\mathsf{S5}} \psi$), are defined analogously by considering only S5 instead of MKNF models. Since validity is not defined with respect to a theory, it does not make sense to distinguish the S5 and the MKNF cases.

A couple of comments regarding these definitions are in order.

Differences to the Definition of MKNF by Litschitz. In our approach, all MKNF models are defined with respect to the same countably infinite domain set $\triangle$. In contrast, in the approach by Lifschitz [25], the domain set $\triangle$ can be any (not necessarily infinite) set that corresponds one-to-one with a special set of constants called *names*. These constants are not indented for use by the modeler, but are introduced only to define the semantics of quantifiers.[1] All interpretations from an MKNF model M are still defined with respect to the same domain set $\triangle$,[2] but different MKNF models can now have different domain sets. Another difference is that $\approx$ is in our approach interpreted as a congruence relation, whereas Lifschitz provides for true equality $=$ that is interpreted as identity. We next discuss how these differences affect the consequences of the logic.

Standard Names Assumption. Let $\varphi = \mathbf{K}\, A(a)$ and $\psi = \exists x : \mathbf{K}\, A(x)$. Under the original semantics, $\varphi \not\models_{\mathsf{MKNF}} \psi$: consider an MKNF interpretation M containing a first-order interpretation I_1 such that $I_1 \models A(\alpha_1)$ and a is interpreted as some name α_1, and a first-order interpretation I_2 such that $I_2 \models A(\alpha_2)$ and a is interpreted as some name α_2 different from α_1. Since φ does not contain quantifiers, obviously $M \models \varphi$. On the contrary, to satisfy ψ, the variable x must be replaced with some name α and $A(\alpha)$ must hold in all first-order models in M. Clearly, $M \not\models \mathbf{K}\, A(\alpha_1)$ and $M \not\models \mathbf{K}\, A(\alpha_2)$, so $\varphi \not\models_{\mathsf{MKNF}} \psi$. This can be corrected if we make a a *rigid designator* by including $\exists x : \mathbf{K}(x = a)$ into φ: now a is interpreted in all first-order interpretations in M as the same name α, so $\varphi \models_{\mathsf{MKNF}} \psi$. In data management applications, all constants are typically assumed to be rigid [31], so we incorporate rigidity directly into the semantics of MKNF by identifying names with the elements of the Herbrand universe.

Finite vs. Infinite Models. We can consider only Herbrand models for first-order formulae as long as we ensure that the Herbrand universe is "large enough." For finite formulae, it suffices to make the set of constants of the signature countably infinite. This has an important side-effect: we now consider only infinite models, whereas in ordinary first-order logic and in

[1] Similar approaches were employed in [22] and [31], where the new constants are called *parameters*. Unlike names, parameters are actually intended to be used by the modeler.

[2] In both approaches, the interpretations in M are defined over the same domain set $\triangle$, because we could otherwise not satisfy the preference semantics of MKNF: we could always extend M with a new first-order interpretation having a new domain set but that is otherwise isomorphic to some existing interpretation J from M.

the version of MKNF by Lifschitz we should also consider finite models. Note, however, that this does not affect the first-order formulae without equality: an equality-free first-order formula φ is satisfiable in an arbitrary model if and only if it is satisfiable in a Herbrand model over a signature with a countably infinite supply of constants not occurring in φ [15, Theorem 5.9.4]. (Fitting calls such models *Herbrand models with parameters*. Note also that the formula φ need not be skolemized.) Hence, without equality, we cannot distinguish the models over an arbitrary interpretation domain and over $\triangle$; consequently, we cannot distinguish finite from infinite models.

Treatment of Equality. Considering only Herbrand models makes it impossible to interpret equality as identity, since syntactically different constants are automatically interpreted as different objects—a property known as *unique names assumption* (UNA). To allow for equality reasoning despite considering only Herbrand models, we provide a special predicate $\approx$ which we interpret as a congruence relation on $\triangle$. Although this interpretation of $\approx$ is weaker, it does not affect the consequences of first-order formulae: any formula of first-order equality is satisfiable in a model where $\approx$ is interpreted as identity if and only if it is satisfiable in a model where $\approx$ is interpreted as a congruence relation [15, Theorem 9.3.9]. Hence, in first-order logic we cannot distinguish these two types of models. Combining this with the discussion from the previous paragraph, we cannot arbitrary models with true equality from Herbrand models with equality interpreted as a congruence relation. We illustrate this by an example. Under standard interpretation of equality, the formula $\varphi = \forall x : (x \approx a \vee x \approx b)$ constrains the domain to have at most two elements. Now if we interpret φ in a model with the domain $\triangle$, the interpretation domain contains infinitely many elements; however, the interpretation of $\approx$ partitions $\triangle$ into at most two equivalence classes. Each object from an equivalence class can be distinguished from all other objects from the class only by its name. Even though this does not change the first-order consequences, one might argue that such a treatment of $\approx$ is too complicated and aesthetically unsatisfactory. However, as we discuss next, such a treatment of $\approx$ yields intuitive nonmonotonic consequences.

Equality and Nonmonotonic Negation. In nonmonotonic reasoning, considering non-Herbrand models often produces unintuitive consequences. For example, let $\varphi = \mathbf{K}\, A(a) \wedge \neg\, \mathbf{not}\, A(b)$. The formula $\neg\, \mathbf{not}\, A(b)$ can be paraphrased as "we should be able to derive $A(b)$." Since the formula φ provides no evidence for this, we would intuitively expect φ to be unsatisfiable. However, under the original semantics of MKNF, φ has an MKNF model: if we take the domain set $\triangle$ to contain only one name α, we must interpret both a and b as α, so φ is satisfied, and such a model clearly satisfies the preference semantics of MKNF. Thus, under the original semantics of

MKNF, φ entails $a \approx b$ and the fact that the interpretation domain contains only one element. We consider this quite unintuitive: we want to interpret $\neg\,\mathbf{not}\,A(b)$ as a constraint without any side-effects. Similar problems occur in other nonmonotonic formalisms such as logic programming, so such formalisms routinely consider only Herbrand models. We do the same in our approach, so we do not get such unintuitive consequences: there is no explicit evidence for $a \approx b$, so φ is unsatisfiable under our semantics. Now consider $\psi = \varphi \wedge \mathbf{K}\,a \approx b$. The congruence properties of $\approx$ now ensure that in each model $A(b)$ holds, so ψ is satisfiable in an MKNF model M containing all first-order interpretations I such that $I \models A(a) \wedge a \approx b$.

Summary. Our version of MKNF is related to the original MKNF in the same way as first-order logic with only Herbrand models is related to ordinary first-order logic with arbitrary models. Strictly speaking, whenever we consider a first-order formula φ in the rest of this paper, we should consider only its models with the domain set $\triangle$. However, as explained in the previous paragraphs, it does not matter if we additionally consider ordinary first-order models of φ. If we need to check satisfiability of φ or whether $\varphi \models \psi$ for a first-order formula ψ, we can equivalently consider Herbrand or arbitrary models. Therefore, in the following sections we do not stress the difference between Herbrand and arbitrary models.

2.2 Description Logics

The MKNF rules defined in Section 4 can be combined with many fragments of first-order logic $\mathcal{DL}$. Therefore, we present only a high-level overview of the syntax and the semantics of description logics without going into details; for a formal definition, please refer to [2].

The building blocks of DL knowledge bases are *concepts* (or *classes*), representing sets of objects, *roles* (or *properties*), representing relationships between objects, and *individuals*, representing specific objects. Concepts such as *Person* are called *atomic*. Using concept constructors, one can construct *complex* concepts that describe the conditions on concept membership. For example, the concept $\exists \mathit{hasFather}.\mathit{Person}$ describes those objects that are related through the *hasFather* role with an object from the concept *Person*. Expressive DLs provide a rich set of concept constructors, such as the Boolean connectives, existential and universal quantification, and number restrictions.

A DL knowledge base $\mathcal{O}$ typically consists of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$. A TBox contains axioms about the general structure of all allowed worlds, and is therefore in its purpose akin to a database schema. For example, the TBox axiom (2) states that each instance of the concept *Person* must be related by the role *hasFather* with an instance of the concept *Person*. An ABox contains axioms describing the structure of particular worlds. For

example, the ABox axiom (3) states that *Peter* is a *Person*, and (4) states that *Paul* is a brother of *Peter*.

$$\mathit{Person} \sqsubseteq \exists \mathit{hasFather}.\mathit{Person} \tag{2}$$

$$\mathit{Person}(\mathit{Peter}) \tag{3}$$

$$\mathit{hasBrother}(\mathit{Peter}, \mathit{Paul}) \tag{4}$$

A DL knowledge base can be given semantics by translating it into first-order logic with equality. Atomic concepts are translated into unary predicates, complex concepts into formulae with one free variable, and roles into binary predicates. For example, the axiom (2) can be represented as the following first-order formula:

$$\forall x : \mathit{Person}(x) \supset \exists y : [\mathit{hasFather}(x, y) \wedge \mathit{Person}(y)] \tag{5}$$

The basic reasoning problems for OWL are checking if an individual a is an instance of a concept C (written $\mathcal{O} \models C(a)$) or if the a concept C is subsumed by another concept D (written $\mathcal{O} \models C \sqsubseteq D$).

Our approach is applicable to any description logic $\mathcal{DL}$ that fulfills these requirements: (*i*) each knowledge base $\mathcal{O} \in \mathcal{DL}$ can be translated into a formula $\pi(\mathcal{O})$ of function-free first-order logic with equality, (*ii*) $\mathcal{DL}$ supports *ABoxes*—assertions of the form $P(a_1, \ldots, a_n)$ for P a predicate and a_i constants of $\mathcal{DL}$, and (*iii*) checking satisfiability and checking entailments of the form $\mathcal{O} \models \alpha$ with α of the form $A(a)$ for A an atomic concept, $R(a, b)$, $a \approx b$ and $a \not\approx b$ are decidable.

Checking entailment of role or equality atoms has traditionally not been considered in DL research; however, it can be reduced to checking entailment of atomic concepts using the well-known *pseudo-nominals* technique, where P_a and P_b are new concepts not occurring in $\mathcal{K}$:

$$\begin{array}{lll} \mathcal{K} \models R(a, b) & \text{iff} & \mathcal{K} \cup \{P_b(b)\} \models \exists R.P_b(a) \\ \mathcal{K} \models a \approx b & \text{iff} & \mathcal{K} \cup \{P_a(a), P_b(b)\} \models P_a \equiv P_b \\ \mathcal{K} \models a \not\approx b & \text{iff} & \mathcal{K} \cup \{P_a(a), P_b(b)\} \models P_a \sqcap P_b \sqsubseteq \bot \end{array}$$

Such inferences are supported by all DLs that support ABoxes, so our approach is applicable to a wide range of DLs.

2.3 Complexity Classes

We use standard definitions of the complexity classes [29]. For complexity classes $\mathcal{C}$ and $\mathcal{E}$, with $\mathcal{E}^{\mathcal{C}}$ we denote the class of problems that can be solved in $\mathcal{E}$ using an oracle for problems in $\mathcal{C}$. The polynomial hierarchy is then defined inductively as follows:

$$\begin{aligned} \Delta_0^p = \Sigma_0^p = \Pi_0^p &= P \\ \Delta_{k+1}^p &= P^{\Sigma_k^p} \\ \Sigma_{k+1}^p &= \mathrm{NP}^{\Sigma_k^p} \\ \Pi_{k+1}^p &= \mathrm{co}\Sigma_{k+1}^p \end{aligned}$$

Table 1: Equivalences for MKNF Formulae

1.	$\mathbf{K}(\varphi \wedge \psi) \equiv \mathbf{K}\,\varphi \wedge \mathbf{K}\,\psi$	$\mathbf{not}(\varphi \wedge \psi) \equiv \mathbf{not}\,\varphi \vee \mathbf{not}\,\psi$
2.	$\mathbf{K}(\kappa \vee \varphi) \equiv \kappa \vee \mathbf{K}\,\varphi$	$\mathbf{not}(\kappa \vee \varphi) \equiv \neg\kappa \wedge \mathbf{not}\,\varphi$
3.	$\mathbf{K}(\forall x : \varphi) \equiv \forall x : \mathbf{K}\,\varphi$	$\mathbf{not}(\forall x : \varphi) \equiv \exists x : \mathbf{not}\,\varphi$
4.	$\mathbf{K}\,\mathsf{true} \equiv \mathsf{true}$	$\mathbf{not}\,\mathsf{true} \equiv \mathsf{false}$
5.	$\mathbf{K}\,\mathsf{false} \equiv \mathsf{false}$	$\mathbf{not}\,\mathsf{false} \equiv \mathsf{true}$

Note: φ and ψ are arbitrary MKNF formulae and κ is a subjective MKNF formula.

2.4 Quantified Boolean Formulae

Quantified Boolean Formulae (QBF) extend propositional formulae by existential and universal quantification over propositional variables. As in first-order logic, a quantified Boolean formula is *closed* if each propositional variable occurs in the scope of a quantifier. The semantics of propositional connectives is defined as usual. For the quantifiers, a closed formula $\exists x : \varphi$ evaluates to true if and only if $\varphi[\mathsf{false}/x] \vee \varphi[\mathsf{true}/x]$ evaluates to true; similarly, a closed formula $\forall x : \varphi$ evaluates to true if and only if $\varphi[\mathsf{false}/x] \wedge \varphi[\mathsf{true}/x]$ evaluates to true. If a closed formula φ evaluates to true, it is also said to be *valid*. Deciding validity of an arbitrary quantified Boolean formula φ is a PSPACE-complete problem; however, if the prenex normal form of φ has i alterations of quantifier prefixes, then deciding validity of φ is in the i-th level of the polynomial hierarchy [37]. More precisely, if the first quantifier in the prefix is $\exists$, the problem is Σ_i^p-complete, and if the first quantifier in the prefix is $\forall$, the problem is Π_i^p-complete.

3 Equivalent Transformations of MKNF Formulae

In this section we present transformations that can be used to transform certain MKNF formulae into equivalent but simpler ones. We use these simplification rules extensively in the following sections. The results from this section hold for general first-order MKNF formulae.

3.1 Equivalent Formulae

It is easy to see that de Morgan laws for first-order logic also hold for MKNF formulae. The following theorem introduces the equivalences concerning the modal operators.

Theorem 3.1. *All equivalences from Table 1 are valid.*

Proof. Consider an MKNF structure (I, M, N). The formula $\mathbf{not}\,\varphi$, when evaluated in N, has the same value as $\neg\,\mathbf{K}\,\varphi$ when evaluated in M. Hence,

the equivalences from the right column are dual to the ones from the left column, and can be proved in the same way by replacing **not** with $\neg\,\mathbf{K}$. Hence, we prove just the equivalences from the left column.

(1) $(I, M, N) \models \mathbf{K}(\varphi \wedge \psi)$ implies $(J, M, N) \models \varphi \wedge \psi$ for each $J \in M$. This implies $(J, M, N) \models \varphi$ and $(J, M, N) \models \psi$ for each $J \in M$, which implies $(I, M, N) \models \mathbf{K}\,\varphi \wedge \mathbf{K}\,\psi$. Conversely, $(I, M, N) \models \mathbf{K}\,\varphi \wedge \mathbf{K}\,\psi$ implies that $(J, M, N) \models \varphi$ for each $J \in M$ and $(J', M, N) \models \varphi$ for each $J' \in M$. Hence, $(J, M, N) \models \varphi \wedge \psi$ for each $J \in M$, which implies $(I, M, N) \models \mathbf{K}(\varphi \wedge \psi)$.

(2) Since κ is subjective, its value in (I, M, N) does not depend on I. Assume $(I, M, N) \models \mathbf{K}(\kappa \vee \varphi)$. If $(I, M, N) \models \kappa$, then $(I, M, N) \models \kappa \vee \mathbf{K}\,\varphi$. Otherwise, for each $J \in M$, we have $(J, M, N) \not\models \kappa$, but then $(J, M, N) \models \varphi$, so $(I, M, N) \models \mathbf{K}\,\varphi$ and $(I, M, N) \models \kappa \vee \mathbf{K}\,\varphi$ as well. Conversely, assume that $(I, M, N) \models \kappa \vee \mathbf{K}\,\varphi$. If $(I, M, N) \models \kappa$, then $(I, M, N) \models \mathbf{K}(\kappa \vee \varphi)$. Otherwise, $(I, M, N) \models \mathbf{K}\,\varphi$, which also implies $(I, M, N) \models \mathbf{K}(\kappa \vee \varphi)$.

The equivalence (3) follows from these considerations:

$(I, M, N) \models \mathbf{K}(\forall x : \varphi)$	iff
$(J, M, N) \models \forall x : \varphi$ for each $J \in M$	iff
$(J, M, N) \models \varphi\{x \mapsto \alpha\}$ for each $J \in M$ and each $\alpha \in \triangle$	iff
$(I, M, N) \models \mathbf{K}\,\varphi\{x \mapsto \alpha\}$ for each $\alpha \in \triangle$	iff
$(I, M, N) \models \forall x : \mathbf{K}\,\varphi$	

The validity of (4) and (5) follows trivially from the definition of $\mathbf{K}$. □

Note that the equivalences $\mathbf{K}\,\kappa \equiv \kappa$ and $\mathbf{not}\,\kappa \equiv \neg\kappa$ can be derived from the equivalence (2) by taking $\varphi = \mathsf{true}$. Furthermore, dually to the equivalence (3), one can show that $\mathbf{K}(\exists x : \kappa) \equiv \exists x : \mathbf{K}\,\kappa$ is valid. We do not consider this equivalence, because the equivalence (2) is stronger: it allows us to actually delete the outer $\mathbf{K}$.

We can replace formulae with equivalent ones without changing the set of models of a formula.

Lemma 3.2. *Let $\sigma_1 \equiv \sigma_2$ be a valid equivalence, φ an MKNF formula containing σ_1 as a subformula, and $\varphi[\sigma_1 \to \sigma_2]$ the formula obtained by replacing in φ an occurrence of σ_1 with σ_2. Then, $(I, M, N) \models \varphi$ if and only if $(I, M, N) \models \varphi[\sigma_1 \to \sigma_2]$, for each MKNF structure (I, M, N).*

Proof. The proof is by a straightforward induction on the structure of φ. □

Using the equivalences from this subsection, it is possible to eliminate nesting of modalities from each propositional formula. For example, the formula $\mathbf{K}(p \vee (r \wedge \neg\,\mathbf{K}\,s))$ is equivalent to $\mathbf{K}((p \vee r) \wedge (p \vee \neg\,\mathbf{K}\,s))$; this allows $\mathbf{K}$ to be distributed over $\wedge$, resulting in $\mathbf{K}(p \vee r) \wedge \mathbf{K}(p \vee \neg\,\mathbf{K}\,s)$; by applying (1) to the second conjunct, we obtain $\mathbf{K}(p \vee r) \wedge (\mathbf{K}\,p \vee \neg\,\mathbf{K}\,s)$. Observe, however, that the above transformation requires translating the formula under the outer $\mathbf{K}$ into conjunctive normal form. Translation into

CNF can incur an exponential increase in the formula size, so this explains the difference between the complexity in checking satisfiability of flat and nonflat formulae reported in [32].

Finally, note that only propositional MKNF formulae can always be reduced to the flat ones. For example, in $\mathbf{K}\,\exists x : [A(x) \vee \neg\,\mathbf{K}\,B(x)]$, the outer $\mathbf{K}$ cannot be removed because the inner formula is not subjective; furthermore, it cannot be distributed over the existential quantifier so the strategy from the previous paragraph is not applicable.

3.2 Adding Modal Operators to the Theory and the Query

The following theorem shows that $\mathbf{K}$ can be introduced in front of a theory without changing the models in which the theory is satisfied:

Theorem 3.3. *Let σ be a closed MKNF formula and M an MKNF interpretation. Then, M is an MKNF model of σ if and only if M is an MKNF model of $\mathbf{K}\,\sigma$.*

Proof. For the ($\Leftarrow$) direction, observe that, for $I \in M$, $(I, M, M) \models \mathbf{K}\,\sigma$ implies $(I, M, M) \models \sigma$ by the definition of the semantics of $\mathbf{K}$; furthermore, for $M' \supset M$ and $I' \in M'$, $(I', M', M) \not\models \mathbf{K}\,\sigma$ implies $(I', M', M) \not\models \sigma$. Hence, M is an MKNF model of σ. The ($\Rightarrow$) direction is analogous. □

Similarly, one can introduce $\mathbf{K}$ in front of a query without affecting entailment:

Theorem 3.4. *Let σ and ψ be arbitrary closed MKNF formulae. Then, $\sigma \models_{\mathsf{MKNF}} \psi$ if and only if $\sigma \models_{\mathsf{MKNF}} \mathbf{K}\,\psi$.*

Proof. If $\sigma \models_{\mathsf{MKNF}} \psi$, then $(I, M, M) \models \psi$ for each MKNF model M of σ and $I \in M$, so clearly $(I, M, M) \models \mathbf{K}\,\psi$ as well. The converse direction is analogous. □

3.3 Introducing Definitions

In first-order logic, it is common practice to define names for complex subformulae: an axiom $Q \equiv \psi$ makes Q equivalent to ψ, thus allowing to use Q in other formulae instead of ψ. However, in [25] it was observed that introducing definitions in MKNF can actually change the semantics of an MKNF theory. Consider the formula $\varphi = \mathbf{not}\,p \supset \mathbf{K}\,r$. Clearly, the only model of φ is $M = \{I \mid I \models r\}$, so $\varphi \models_{\mathsf{MKNF}} \mathbf{K}\,r$. By introducing a new name q for the MKNF formula $\mathbf{K}\,p$, we obtain the formula $\varphi' = \varphi \wedge (q \equiv \mathbf{K}\,p)$. By an analogy to first-order logic, one would expect that $\varphi' \models_{\mathsf{MKNF}} \mathbf{K}\,r$ as well: we do not expect the theorems of φ not containing the name q to be affected. However, φ' has an MKNF model $M = \{I \mid I \models p \wedge q\}$, so $\varphi' \not\models_{\mathsf{MKNF}} \mathbf{K}\,r$.

The equivalences from the previous subsection can help us explain this phenomenon. Namely, by Theorem 3.3, we can transform $q \equiv \mathbf{K}\,p$ into

$\mathbf{K}(q \equiv \mathbf{K}\,p)$, which can be expanded to $\mathbf{K}[(\neg q \vee \mathbf{K}\,p) \wedge (\neg\,\mathbf{K}\,p \vee q)]$ and further simplified to $(\mathbf{K}\,\neg q \vee \mathbf{K}\,p) \wedge (\neg\,\mathbf{K}\,p \vee \mathbf{K}\,q)$. Now $\mathbf{K}\,\neg q \vee \mathbf{K}\,p$ clearly has two minimal models: one in which $\neg q$ holds, and one in which p holds. Succinctly put, introducing the formula $q \equiv \mathbf{K}\,p$ introduces as a side-effect a disjunction $\mathbf{K}\,\neg q \vee \mathbf{K}\,p$.

Another way to intuitively understand this issue is to observe that the implication $q \subset \mathbf{K}\,p$ is a kind of one-directional rule: it allows us to conclude q from p; however, from $\neg q$ we cannot conclude $\neg p$. Thus, $q \equiv \mathbf{K}\,p$ cannot be understood as a definition in a first-order sense.

However, definitions can be introduced for purely first-order formulae, as shown by the following lemma:

Lemma 3.5. *Let σ be an MKNF formula, φ a first-order formula with the set of free variables $\mathbf{x}$, Q a predicate not occurring in σ and φ, and $\sigma' = \sigma \wedge \forall \mathbf{x} : Q(\mathbf{x}) \equiv \varphi$. Then, each MKNF model M of σ corresponds one-to-one to an MKNF model M' of σ' obtained by interpreting in each $I \in M$ the atom $Q(\mathbf{x})$ exactly as φ.*

Proof. Follows from trivially from the fact that $\forall \mathbf{x} : Q(\mathbf{x}) \equiv \varphi$ does not contain modal operators. □

It is now clear that $Q(\mathbf{x})$ can be used as a synonym for φ—that is, it is possible to replace an occurrence of φ in σ' with $Q(\mathbf{x})$ without changing the set of MKNF models of σ'.

Unfortunately, new names cannot be introduced for subjective formulae. Consider the following propositional MKNF formula σ:

$$\sigma = \mathbf{K}\,p \wedge \mathbf{K}\,q \wedge [\mathbf{K}\,p \supset \mathbf{K}\,r \vee (\mathbf{K}\,q \wedge \neg\,\mathbf{K}\,s)]$$

It is easy to verify that $M = \{I \mid I \models p \wedge q\}$ is the only MKNF model of σ. Let us now introduce a new name for the subformula $\mathbf{K}\,r \wedge \neg\,\mathbf{K}\,s$:

$$\psi = \sigma \wedge \mathbf{K}\,t \equiv (\mathbf{K}\,q \wedge \neg\,\mathbf{K}\,s)$$

$M'_1 = \{I \mid I \models p \wedge q \wedge t\}$ and $M'_2 = \{I \mid I \models p \wedge q \wedge s\}$ are the two MKNF models of ψ. Whereas M can be obtained from M'_1 by projecting out the new symbol t, M'_2 is completely unrelated to M.

3.4 Reducing Entailment to Unsatisfiability

As it is the case for most nonmonotonic formalisms, checking entailment in MKNF cannot be reduced to satisfiability using the well-known transformation. Namely, for σ and ψ closed MKNF formulae, if $\sigma \not\models_{\mathsf{MKNF}} \psi$, then $\sigma \wedge \neg\psi$ is MKNF satisfiable; however, the converse does not hold. For example, let $\sigma = \mathsf{true}$ and $\psi = \neg\,\mathbf{K}\,p$. Clearly, σ has an MKNF model M containing all first-order interpretations over $\triangle$, so $M \not\models \mathbf{K}\,p$ and $M \models \neg\,\mathbf{K}\,p$. However,

$\sigma \wedge \neg\psi = \mathbf{K}\, p$, which has an MKNF model $M' = \{I \mid I \models p\}$. Intuitively, this problem arises because adding $\mathbf{K}\, p$ to σ makes p known, so M is not a model of $\sigma \wedge \neg\psi$.

However, in [33] it was shown how to reduce checking entailment of subjective MKNF formulae to MKNF satisfiability. For the sake of completeness, we include this result in this paper as well.

Theorem 3.6. *Let σ be an arbitrary closed MKNF formula and ψ a subjective closed MKNF formula. Then, $\sigma \models_{\mathsf{MKNF}} \psi$ if and only if $\sigma \wedge \neg\varphi$ is MKNF unsatisfiable, where φ is the MKNF formula obtained from ψ by replacing each occurrence of $\mathbf{K}$ with $\neg\,\mathbf{not}$.*

Proof. ($\Leftarrow$) Assume that $\sigma \not\models_{\mathsf{MKNF}} \psi$; then, there is an MKNF model M of σ and some $I \in M$ such that $(I, M, M) \not\models \psi$. Because $\mathbf{K}$ and $\neg\,\mathbf{not}$ have the same value when interpreted in the same set of interpretations, we have $(I, M, M) \not\models \varphi$ and $(I, M, M) \models \neg\varphi$, which implies $(I, M, M) \models \sigma \wedge \neg\varphi$. By assumption, $(I', M', M) \not\models \sigma$ for each $M' \supset M$ and $I' \in M'$, so we have $(I', M', M) \not\models \sigma \wedge \neg\varphi$ as well. Clearly, M is an MKNF model of $\sigma \wedge \neg\varphi$.

($\Rightarrow$) Assume that $\sigma \wedge \neg\varphi$ is MKNF satisfiable in an MKNF model M; that is, $(I, M, M) \models \sigma \wedge \neg\varphi$ for each $I \in M$, and $(I', M', M) \not\models \sigma \wedge \neg\varphi$ for each $M' \supset M$ and $I' \in M'$. Because φ is subjective and it does not contain occurrences of $\mathbf{K}$, its value in (I', M', M) does not depend on M' and I', so $(I', M', M) \models \neg\varphi$ and $(I', M', M) \not\models \sigma$. Hence, M is an MKNF model of σ. Furthermore, $\mathbf{K}$ and $\neg\,\mathbf{not}$ have the same truth value if interpreted in the same set of interpretations and ψ is subjective, so $(I, M, M) \models \neg\psi$ for each $I \in M$, which implies $(I, M, M) \not\models \psi$. But now, M is an MKNF model of σ which does not satisfy ψ, so $\sigma \not\models_{\mathsf{MKNF}} \psi$. □

4 Extending DLs with MKNF Rules

We now define the formalism of hybrid MKNF knowledge bases that allows to extend any description logic $\mathcal{DL}$ with nonmonotonic rules. We introduce our formalism in three steps. First, in Section 4.1, we present a definition that generalizes most currently known proposals for combining DLs with rules. Next, in Section 4.2, we present a syntactic restriction necessary to make our formalism decidable. Finally, in Section 4.3, we show that, without a loss of generality, we can consider only flat rules.

4.1 MKNF Rules

We first define the most general variant of MKNF rules.

Definition 4.1. *Let $\mathcal{DL}$ be a description logic, and $\mathcal{O} \in \mathcal{DL}$ a DL knowledge base. Let Σ be a signature containing the equality predicate $\approx$, all atomic concepts from $\mathcal{O}$ as unary predicates, all atomic roles from $\mathcal{O}$ as binary*

predicates, and all individuals from $\mathcal{O}$ as constants. A first-order function-free atom $P(t_1, \ldots, t_n)$ over Σ such that P is $\approx$ or it occurs in $\mathcal{O}$ is called a DL-atom*; all other atoms are called* non-DL-atoms*.*

Let A denote arbitrary first-order function-free atoms over the signature Σ, let H_i denote atoms of the form A or $\mathbf{K}\,A$, and let B_i denote atoms of the form A, $\mathbf{K}\,A$, or $\mathbf{not}\,A$. Then, an MKNF rule *has the following form:*

$$H_1 \vee \ldots \vee H_n \leftarrow B_1, \ldots, B_m$$

As usual, the set of atoms $\{H_i\}$ is called the rule head*, and the set of atoms $\{B_i\}$ is called the* rule body*. An MKNF rule with $n = 1$ in which all atoms are modal is a* nondisjunctive MKNF rule*; an MKNF rule with $m = 0$ is called a* fact*. If each variable in a rule occurs in a body $\mathbf{K}$-atom, the rule is* safe*. A* program *$\mathcal{P}$ is a finite set of MKNF rules.*

A hybrid MKNF knowledge base *$\mathcal{K}$ is a pair $(\mathcal{O}, \mathcal{P})$. The* size *of $\mathcal{K}$, written $|\mathcal{K}|$, is the number of symbols needed to encode $\mathcal{K}$. With $O_\mathcal{K}$ we denote the set of all constants occurring in $\mathcal{K}$ (if $\mathcal{K}$ does not contain any constant, we add an arbitrary constant to $O_\mathcal{K}$).*

As usual, we write $\approx (a, b)$ as $a \approx b$ and $\neg(a \approx b)$ as $a \not\approx b$. We now define the semantics of $\mathcal{K}$ by mapping it into first-order MKNF.

Definition 4.2. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base, and let $\pi(\mathcal{O})$ be the translation of $\mathcal{O}$ into first-order logic with equality. For an MKNF rule r, let $\mathbf{x}$ be the vector of variables occurring in the rule. We extend π to r, $\mathcal{P}$, and $\mathcal{K}$ as follows:*

$$\begin{aligned}
\pi(r) &= \forall \mathbf{x} : (H_1 \vee \ldots \vee H_n \subset B_1 \wedge \ldots \wedge B_m) \\
\pi(\mathcal{P}) &= \textstyle\bigwedge_{r \in \mathcal{P}} \pi(r) \\
\pi(\mathcal{K}) &= \mathbf{K}\,\pi(\mathcal{O}) \wedge \pi(\mathcal{P})
\end{aligned}$$

A hybrid MKNF knowledge base $\mathcal{K}$ is satisfiable *if and only if an MKNF model of $\pi(\mathcal{K})$ exists. Furthermore, $\mathcal{K}$* entails *an MKNF formula ψ, written $\mathcal{K} \models \psi$, if and only if $\pi(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$.*

We say that an MKNF knowledge base $\mathcal{K}$ is *subjective* (*flat*) if this property holds for $\pi(\mathcal{K})$; furthermore, $\mathcal{K}$ is *nondisjunctive* if all rules in $\mathcal{P}$ are *nondisjunctive*.

A couple of comments about the previous definition are in order.

Compatibility with DLs. If all rules from $\mathcal{K}$ contain only nonmodal atoms, then the semantics of $\mathcal{K}$ is essentially first-order: using the equivalences from Section 3, the formula $\pi(\mathcal{K})$ can be transformed into the formula $\mathbf{K}[\pi(\mathcal{O}) \wedge \pi(\mathcal{P})]$, which entails the same formulae as the first-order formula $\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$. Hence, our semantics of hybrid MKNF knowledge bases is a conservative extension of the standard first-order semantics.

Compatibility with Logic Programming. In [25] it was shown that each stable model of a disjunctive logic program with rules of the form

$$H_1 \vee \ldots \vee H_k \leftarrow B_1^+ \wedge \ldots \wedge B_n^+ \wedge \mathbf{not}\, B_1^- \wedge \ldots \wedge \mathbf{not}\, B_m^-$$

corresponds one-to-one to an MKNF model of an MKNF program obtained by replacing each rule with a rule of the form (1). Hence, if $\mathcal{O} = \emptyset$, the semantics of hybrid MKNF knowledge bases corresponds to the stable model semantics of disjunctive logic programs. Stable model semantics generalizes the semantics of stratified and positive programs, so our approach generalizes many well-known approaches to logic programming. Our approach currently cannot capture the well-founded semantics [39]; we shall try to address this restriction in our future work.

Relationship with other Rule Formalisms. Our approach generalizes all existing approaches to extending DLs with first-order rules, such as CARIN [23], $\mathcal{AL}$-log [7], DL-safe rules [27], or the Semantic Web Rule Language (SWRL) [18]. If $\mathcal{P}$ contains only nonmodal atoms, then it is equivalent to a set of SWRL rules (in which multiple head atoms are interpreted disjunctively). As we show in Section 4.4, hybrid MKNF knowledge bases generalize a significant portion of the $\mathcal{DL}$+log [34, 35, 36] family of proposals for integrating DLs and rules. Another related formalism was presented in [13], in which disjunctive datalog rules can contain special atoms interpreted as queries over a DL knowledge base. The semantics of such hybrid knowledge bases is defined through a generalization of the answer set semantics. This semantics is nonstandard and it does not correspond to the standard semantics of first-order extensions of DLs with rules even if the rules do not contain atoms under negation-as-failure.

Relationship with other Nonmonotonic Extensions of DLs. In [8], the authors extend the DL $\mathcal{ALC}$ with two modal operators **K** and **A** interpreted under MKNF semantics. This allows for nonmonotonic reasoning on existentially introduced individuals; however, the usage of the modal operators must be restricted significantly to obtain a decidable logic. Furthermore, this approach does not allow for general rules. Finally, the approach uses UNA and does not provide for equality reasoning. An extension of DLs with default rules was presented in [3]; to achieve decidability, the authors allow the defaults to be applied only to named individuals. Our approach also allows to model defaults on the explicitly named individuals. The approach from [3] implements the original Reiter's semantics of defaults [30], whereas our approach is compatible with the default logic with fixed universe [24]. An approach for extending DLs with circumscription was presented in [4]. Unlike other nonmonotonic extensions of DLs, this approach allows for nonmonotonic reasoning on unnamed individuals; however, to achieve decidability, nonmonotonic reasoning is allowed only on unary predicates.

4.2 DL-Safety

We now turn our attention to decidability of reasoning in hybrid MKNF knowledge bases. Clearly, for reasoning to be decidable, the description logic $\mathcal{DL}$ should be decidable. Also, it is well known that combining arbitrary first-order rules with decidable description logics containing just the very basic DL constructs leads to undecidability of the satisfiability problem [23]. Since nonmodal MKNF rules correspond to arbitrary first-order rules, the result from [23] implies undecidability of checking satisfiability of an MKNF knowledge base $\mathcal{K}$ for a wide range of languages $\mathcal{DL}$.

Consider now the case when MKNF rules are flat. Such rules are not semantically equivalent to first-order rules considered in [23], so the undecidability result presented there does not apply directly. However, even in this case reasoning is undecidable:

Theorem 4.3. *For $\mathcal{K}$ a safe hybrid MKNF knowledge base and A a ground atom, checking whether $\mathcal{K} \models A$ is undecidable if $\mathcal{DL}$ allows us to express an axiom $\top \sqsubseteq C$.*

Proof. We adapt the reduction of the haling problem of a Turing machine T to the entailment problem from [23]. Without loss of generality, we can assume that T starts execution with an empty tape. The haling problem is undecidable [29], which implies our theorem.

For a Turing machine T, we construct a hybrid MKNF knowledge base $\mathcal{K}_T$ as follows. The DL knowledge base $\mathcal{O}_T$ of $\mathcal{K}_T$ contains only the following DL axiom:

$$\top \sqsubseteq \mathit{integer} \tag{6}$$

Furthermore, we add to the MKNF program $\mathcal{P}_T$ of $\mathcal{K}_T$ the following rules:

$$\mathbf{K}\,\mathsf{false} \leftarrow \mathbf{K}\,\mathit{integer}(x), \mathbf{not}\,\mathit{hasSucc}(x) \tag{7}$$

$$\mathbf{K}\,\mathit{hasSucc}(x) \leftarrow \mathbf{K}\,\mathit{succ}(x, y) \tag{8}$$

The remaining rules are constructed in the same way as in [23]. For the sake of completeness, we repeat this construction here. In short, we use the predicate $\mathit{lt}(x, y)$ to order the elements of the *integer* concept, the predicate $\mathit{state}(x, y, z)$ to denote that, at time x, T is in state y with the head at position z,[3] and the predicate $\mathit{tape}(x, y, z)$ to denote that, at time x, the tape of T at position y contains the symbol z. Finally, we represent all states q_i including the initial state q_0 and the halting state q_h, all symbols σ_i, the empty symbol $\sqcup$, the tape boundary symbol $\triangleright$, and the integer 1 as constants. We now show the remaining rules of $\mathcal{P}_T$.

[3]The reduction from [23] uses two binary predicates to encode the state and the head position of T. We use a single ternary predicate to make the presentation compact.

The following rules axiomatize lt as the transitive closure of $succ$:

$$\mathbf{K}\, lt(x, y) \leftarrow \mathbf{K}\, succ(x, y) \tag{9}$$

$$\mathbf{K}\, lt(x, y) \leftarrow \mathbf{K}\, succ(x, z), \mathbf{K}\, succ(z, y) \tag{10}$$

The following rules specify the initial configuration of T:

$$\mathbf{K}\, integer(1) \tag{11}$$

$$\mathbf{K}\, state(1, q_0, 1) \tag{12}$$

$$\mathbf{K}\, tape(1, 1, \triangleright) \tag{13}$$

$$\mathbf{K}\, tape(1, x, \sqcup) \leftarrow \mathbf{K}\, lt(1, x) \tag{14}$$

We now encode each transition $\delta(q, \sigma) = (q', \sigma', D)$ with $D \in \{\leftarrow, \rightarrow, -\}$. The first rule updates the symbol at the position of the head, the second rule is added for $D = \leftarrow$ and moves the head to the left, the third rule is added for $D = \rightarrow$ and moves the head to the right, and the fourth rule is added for $D = -$ and does not change the head position:

$$\mathbf{K}\, tape(x', y, \sigma') \leftarrow \mathbf{K}\, state(x, q, y), \mathbf{K}\, tape(x, y, \sigma), \mathbf{K}\, succ(x, x') \tag{15}$$

$$\mathbf{K}\, state(x', q', y') \leftarrow \mathbf{K}\, state(x, q, y), \mathbf{K}\, tape(x, y, \sigma), \mathbf{K}\, succ(x, x'), \mathbf{K}\, succ(y', y) \tag{16}$$

$$\mathbf{K}\, state(x', q', y') \leftarrow \mathbf{K}\, state(x, q, y), \mathbf{K}\, tape(x, y, \sigma), \mathbf{K}\, succ(x, x'), \mathbf{K}\, succ(y, y') \tag{17}$$

$$\mathbf{K}\, state(x', q', y) \leftarrow \mathbf{K}\, state(x, q, y), \mathbf{K}\, tape(x, y, \sigma), \mathbf{K}\, succ(x, x') \tag{18}$$

The following two rules copy the symbols on the tape from one time instant to the other to the left and to the right of the head:

$$\mathbf{K}\, tape(x', y', w) \leftarrow \mathbf{K}\, state(x, v, y), \mathbf{K}\, succ(x, x'), \mathbf{K}\, lt(y', y), \mathbf{K}\, tape(x, y', w) \tag{19}$$

$$\mathbf{K}\, tape(x', y', w) \leftarrow \mathbf{K}\, state(x, v, y), \mathbf{K}\, succ(x, x'), \mathbf{K}\, lt(y, y'), \mathbf{K}\, tape(x, y', w) \tag{20}$$

Finally, the following rule detects the halting condition:

$$\mathbf{K}\, halt \leftarrow \mathbf{K}\, state(x, q_h, y) \tag{21}$$

We now show that $\mathcal{K}_T$ encodes the execution of T. More formally, T does not halt on the empty string if and only if $\mathcal{K}_T \not\models \mathbf{K}\, halt$.

($\Rightarrow$) Assume that the execution of T does not halt. Then, we construct a model M such that, starting from 1, all elements of $\triangle$ are connected through the $succ$ role. We use these elements to represent the time instants and the positions on the tape. Furthermore, let $M \models \mathbf{K}\, state(t, s, p)$ if and only if in the execution of T the state and the head position at time t are s and p, respectively. Also, let $M \models \mathbf{K}\, tape(t, p, m)$ if and only if the tape contains at time t the symbol m at position p. It is easy to see that M is an MKNF model of $\mathcal{K}_T$ and that $M \not\models \mathbf{K}\, halt$.

($\Leftarrow$) Assume that $\mathcal{K}_T$ has an MKNF model M such that $M \not\models \mathbf{K}\, \mathit{halt}$. By (6), the extension of *integer* is equal to $\triangle$ in each $I \in M$. The rule (8) identifies those objects α for which $\beta \in \triangle$ exists such that $M \models \mathbf{K}\, \mathit{succ}(\alpha, \beta)$, and the rule (7) ensures that a successor exists in M for each $\alpha \in \triangle$. Hence, starting from 1, the model M contains an infinite sequence of objects connected through *succ*. Furthermore, the rules (9)–(20) encode the movements of T, so a run of T on the empty string can be extracted from M. Since $M \not\models \mathbf{K}\, \mathit{halt}$, this run does not halt. □

Our proof essentially differs from [23] only in one aspect. There, an infinite sequence of integers is obtained by $\mathit{integer} \sqsubseteq \exists \mathit{succ}.\mathit{integer}$. In our case, this would not produce the desired effect: the existentially introduced individuals can differ in each interpretation I from an MKNF model M, so $M \not\models \mathbf{K}\, \mathit{succ}(\alpha, \beta)$ for any such individuals α and β. To obtain the required sequence, we employ the following trick: we make the concept *integer* equivalent to $\triangle$, which ensures $M \models \mathbf{K}\, \mathit{integer}(\alpha)$ for each $\alpha \in \triangle$. We thus make the entire domain $\triangle$ "visible" to the rules in $\mathcal{P}_T$. We then employ **not** to connect all individuals appropriately through *succ* in rules (7)–(8). The rule (7) is safe syntactically; however, $\mathbf{K}\, \mathit{integer}(x)$ actually makes it applicable to the whole domain, so it is not safe semantically.

To obtain decidability, we apply the well-known concept of DL-safety [34, 27]. Intuitively, this restriction makes the rules applicable only to individuals known in the ABox. We discuss the practical consequences of DL-safety in Section 5 by means of an example.

Definition 4.4. *An MKNF rule r is* DL-safe *if every variable in r occurs in at least one non-DL-atom $\mathbf{K}\, B$ occurring in the body of r. A hybrid MKNF knowledge base $\mathcal{K}$ is* DL-safe *if all its rules are DL-safe.*

Due to DL-safety, a hybrid MKNF knowledge base $\mathcal{K}$ is equisatisfiable with a hybrid MNKF knowledge base $\mathcal{K}_G$ obtained from $\mathcal{K}$ by replacing all MKNF rules with the set of their ground instances.

Definition 4.5. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a hybrid MKNF knowledge base. Then, the knowledge base $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$, where $\mathcal{P}_G$ is obtained by replacing in each rule of $\mathcal{P}$ all variables with constants from $O_\mathcal{K}$ in all possible ways, is called the* ground instantiation *of $\mathcal{K}$.*

Lemma 4.6. *Let $\mathcal{K}$ be a DL-safe hybrid MKNF knowledge base, $\mathcal{K}_G$ the ground instantiation of $\mathcal{K}$, and ψ a ground MKNF formula. Then, $\mathcal{K} \models \psi$ if and only if $\mathcal{K}_G \models \psi$.*

Proof. We prove the lemma by showing the contrapositive statement—that is, $\pi(\mathcal{K}) \not\models_{\mathsf{MKNF}} \psi$ if and only if $\pi(\mathcal{K}_G) \not\models_{\mathsf{MKNF}} \psi$. For the ($\Rightarrow$) direction, let M be an MKNF model of $\pi(\mathcal{K})$ such that $M \not\models \psi$. For each ground

non-DL-atom B containing a constant from $\triangle \setminus O_{\mathcal{K}}$, we have $M \not\models \mathbf{K}\, B$. Namely, assume the contrary and consider an interpretation M' obtained by adding, for each ground non-DL-atom B containing a constant from $\triangle \setminus O_{\mathcal{K}}$, an interpretation that coincides with any other interpretation from M on all atoms but on B. Clearly, $M' \subset M$. Consider now each rule $r \in P$ and its ground instance r_G. If r_G contains only constants from $O_{\mathcal{K}}$, we have $M' \models r_G$ because the values of ground non-DL-atoms containing only constants from $O_{\mathcal{K}}$ coincide in M and M'. Otherwise, r_G is DL-safe, so it contains an atom $\mathbf{K}\, B$ containing a constant from $\triangle \setminus O_{\mathcal{K}}$; but then, $M' \not\models \mathbf{K}\, B$, so $M' \models r_G$. Hence, $M' \models \pi(KB)$, which contradicts the assumption that M is an MKNF model of $\pi(\mathcal{K})$. Hence, $M \not\models \mathbf{K}\, B$ if B contains a constant from $\triangle \setminus O_{\mathcal{K}}$, which immediately implies that M is an MKNF model of $\pi(\mathcal{K}_G)$, so $\pi(\mathcal{K}_G) \not\models_{\mathsf{MKNF}} \psi$.

For the ($\Leftarrow$) direction, let M be an MKNF model of $\pi(\mathcal{K}_G)$ such that $M \not\models \psi$. Clearly, $M \models \mathbf{K}\, \pi(\mathcal{O})$ so, to prove $M \models \pi(\mathcal{K})$, we just need to show that $M \models \pi(r)$ for each $r \in \mathcal{P}$. Consider a ground instance r_G of r. If r_G contains only constants from $O_{\mathcal{K}}$, then $M \models r_G$. Otherwise, since r is DL-safe, each constant from $\triangle \setminus O_{\mathcal{K}}$ occurs in some ground non-DL-atom $\mathbf{K}\, B$ occurring in the body of r_G. Assume that $M \models \mathbf{K}\, B$ and consider $M' = M \cup \{I'\}$ where I' is obtained from $I \in M$ by just changing the truth value of B in I'. Since $\pi(\mathcal{K}_G)$ does not contain a constant from $\triangle \setminus O_{\mathcal{K}}$, we have $M' \models \pi(\mathcal{K}_G)$, which contradicts the assumption that M is an MKNF model of $\pi(\mathcal{K}_G)$. Hence, $M \models \neg\, \mathbf{K}\, B$, so $M \models r_G$ and, consequently, $M \models \pi(\mathcal{K})$. Furthermore, assume that an MKNF interpretation $M'' \supset M$ exists such that $M'' \models \pi(\mathcal{K})$. Clearly, $M'' \models \pi(\mathcal{K}_G)$, which contradicts the assumption that M is an MKNF model of $\pi(\mathcal{K}_G)$. Hence, M is an MKNF model of $\pi(\mathcal{K})$. Because $M \not\models \psi$, we have $\pi(\mathcal{K}) \not\models_{\mathsf{MKNF}} \psi$. □

Lemma 4.6 holds because we ground the rules with respect to the Herbrand universe of $\mathcal{K}$. In 2.2, we assume that the description logic $\mathcal{DL}$ corresponds to a function-free fragment of first-order logic. Namely, the Herbrand universe would become infinite if we introduced even just one function symbol. This would make $\mathcal{K}_G$ infinite, so we could not use grounding as a preprocessing step in the reasoning algorithms from Section 7.

Strictly speaking, Definition 4.5 does not generalize the notion of first-order DL-safety from [27], because it requires each variable in the rule to occur in a **K**-atom in the rule body. We use such a definition to simplify the presentation. Namely, first-order DL-safe rules can always be considered to be a part of the DL knowledge base $\mathcal{O}$. In this paper we are interested in nonmonotonic reasoning, for which we need rules with modal operators in the head and body.

4.3 Flat vs. Nonflat Rules

Definition 4.2 allows an MKNF rule to contain both modal and nonmodal atoms. In this way, we obtain a formalism that generalizes most known approaches for combining DLs with rules. However, we now show that we can consider only flat rules without loss of generality. Namely, each rule can be written in the following form:

$$\forall x : [\bigvee H_i \vee \bigvee \mathbf{K}\, H_j \vee \neg \bigwedge B_k \vee \neg \bigwedge \mathbf{K}\, B_m \vee \neg \bigwedge \mathbf{not}\, B_n]$$

Now $\pi(\mathcal{K}_G)$ is MKNF equivalent with $\mathbf{K}\,\pi(\mathcal{K}_G)$ and, since all rules are interpreted conjunctively, the outer occurrence of $\mathbf{K}$ can be distributed to each rule. Hence, the above rule is equivalent to the following formula:

$$\mathbf{K}\,\forall x : [\bigvee H_i \vee \bigvee \mathbf{K}\, H_j \vee \neg \bigwedge B_k \vee \neg \bigwedge \mathbf{K}\, B_m \vee \neg \bigwedge \mathbf{not}\, B_n]$$

We can switch the order of $\mathbf{K}$ and $\forall$, which produces the following formula:

$$\forall x : \mathbf{K}[\bigvee H_i \vee \bigvee \mathbf{K}\, H_j \vee \neg \bigwedge B_k \vee \neg \bigwedge \mathbf{K}\, B_m \vee \neg \bigwedge \mathbf{not}\, B_n]$$

We can extract the objective part of the formula outside the outer occurrence of $\mathbf{K}$ to obtain the following formula:

$$\forall x : [\mathbf{K}(\bigvee H_i \vee \neg \bigwedge B_k) \vee \bigvee \mathbf{K}\, H_j \vee \neg \bigwedge \mathbf{K}\, B_m \vee \neg \bigwedge \mathbf{not}\, B_n]$$

Finally, we can introduce a new name Q for $\bigvee H_i \vee \neg \bigwedge B_k$ and obtain the following flat MKNF rule and a first-order definition:

$$\forall x : [\mathbf{K}\, Q \vee \bigvee \mathbf{K}\, H_j \vee \neg \bigwedge \mathbf{K}\, B_m \vee \neg \bigwedge \mathbf{not}\, B_n]$$
$$\forall \mathbf{x} : [Q \equiv \bigvee H_i \vee \neg \bigwedge B_k]$$

Now the definition for Q is a first-order formula so, assuming that it is allowed in the fragment $\mathcal{DL}$, it can be added to the DL knowledge base $\mathcal{O}$. In practical cases, our rule will be DL-safe, so we can ground it; then, the formula $\bigvee H_i \vee \neg \bigwedge B_k$ is ground and can be interpreted as a first-order DL-safe rule [27]. It is well-known that first-order DL-safe rules can combined with any description logic without losing decidability in a straightforward way. Hence, the above transformation allows us to reduce reasoning with nonflat DL-safe rules in description logic $\mathcal{DL}$ to reasoning with flat DL-safe rules in description logic $\mathcal{DL}'$ obtained by extending $\mathcal{DL}$ with first-order DL-safe rules. Since a decision procedure for $\mathcal{DL}'$ can be obtained in a straightforward way from a decision procedure for $\mathcal{DL}$ [34], this transformation allows us to consider only flat rules in the following sections.

The presented transformation also shows why DL-safety requires each variable to occur in a body $\mathbf{K}$-atom. Namely, if some variable occurs only in a nonmodal atom, this atom will occur in the disjunction $\bigvee H_i \vee \neg \bigwedge B_k$ which, as shown previously, can be removed from the rule.

4.4 Relationship with $\mathcal{DL}$+log

Several approaches for extending DLs with nonmonotonic rules were presented recently in [34, 35] and were generalized in [36] to a formalism called $\mathcal{DL}$+log. We now show that hybrid MKNF knowledge bases are able to capture the semantics of $\mathcal{DL}$+log knowledge bases. To make this paper self contained, we recall first the definition of $\mathcal{DL}$+log.

The signature Σ of a $\mathcal{DL}$+log knowledge base is divided into a set of concept names Σ_C, role names Σ_R, and datalog predicates Σ_D. The predicates from $\Sigma_C \cup \Sigma_R$ are called *DL-predicates* and the predicates from Σ_D are called *non-DL-predicates*. The atoms of $\mathcal{DL}$+log are function-free first-order atoms defined as usual; they are called *DL-atoms* or *non-DL-atoms* depending on the type of the predicate. A $\mathcal{DL}$+log knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ consists of a description logic knowledge base $\mathcal{O}$ expressed in some first-order fragment $\mathcal{DL}$ and a set of rules $\mathcal{P}$ of the following form:

$$P_1 \vee \ldots \vee P_n \leftarrow R_1, \ldots, R_m, S_1, \ldots, S_k, \mathbf{not}\, U_1, \ldots, \mathbf{not}\, U_h \tag{22}$$

The atoms P_i are allowed to be either DL- or non-DL-atoms, the atoms R_i and U_i are required to be non-DL-atoms, and the atoms S_i are required to be DL-atoms. $\mathcal{DL}$+log generalizes known first-order extensions of DLs with rules such as CARIN [23], so reasoning with it is trivially undecidable. To obtain decidability, in [36] the author introduces the notion of *weak safety*: a rule r of form (22) is weakly safe if every variable from some atom P_i occurs in at least one of the non-DL-atoms R_i. To allow comparing $\mathcal{DL}$+log to hybrid MKNF knowledge bases, we also extend the definition of DL-safety to $\mathcal{DL}$+log rules: a rule r of form (22) is DL-safe if every variable from r occurs in at least one of the non-DL-atoms R_i. Clearly, weak safety generalizes the notion of DL-safety.

Like MKNF knowledge bases, $\mathcal{DL}$+log employs the standard names assumption in the definition of the semantics: the interpretation $\triangle$ corresponds one-to-one with a countably infinite set of constants $\mathcal{C}$ of the signature. The presentation in [36] is not explicit about the treatment of equality; however, to achieve compatibility with standard semantics of DLs, it is necessary to adopt an approach similar to ours and treat $\approx$ as a DL-predicate that is interpreted as a congruence in each model over $\triangle$.

$\mathcal{DL}$+log comes with two types of semantics. The first-order (FOL) semantics is obtained by interpreting each rule of the form (22) as the following first-order implication, where $\mathbf{x}$ is the set of free variables of the rule:

$$\forall \mathbf{x} : P_1 \vee \ldots \vee P_n \subset R_1 \wedge \ldots \wedge R_m \wedge S_1 \wedge \ldots \wedge S_k \wedge \neg U_1 \wedge \ldots \wedge \neg U_h \tag{23}$$

To define the nonmonotonic (NM) semantics, the following standard definitions for datalog programs are needed. For $\mathcal{P}_G$ a ground datalog program and I an interpretation, the GL-reduct of $\mathcal{P}_G$ with respect to I, written $\mathsf{GL}(\mathcal{P}_G, I)$ is obtained by transforming each rule $r \in \mathcal{P}_G$ as follows:

- Delete r if it contains a negated atom $\mathbf{not}\, B_i$ such that $I \models B_i$;
- Delete all negated body atoms $\mathbf{not}\, B_i$ such that $I \not\models B_i$.

An interpretation I is a *model* of a ground datalog program $\mathcal{P}_G$ without **not**-atoms if it satisfies the rules of $\mathcal{P}_G$ when these are interpreted as implications (23); I is a *minimal model* if no interpretation $I' \subset I$ is a model of $\mathcal{P}_G$. For a ground datalog program $\mathcal{P}_G$ possibly containing **not**-atoms, an interpretation I is a *stable model* if it is the minimal model of $\mathsf{GL}(\mathcal{P}_G, I)$.

We now define the NM semantics of $\mathcal{DL}$+log. Let $\mathsf{gr}(\mathcal{P})$ be the ground program obtained by replacing in each rule from $\mathcal{P}$ all variables with constants from $\triangle$ in all possible ways. For I an interpretation and Σ a set of predicates, I_Σ is the interpretation obtained by restricting I to the predicates in Σ; furthermore, for $\mathcal{P}_G$ a ground program, $\Pi(\mathcal{P}_G, I)$ is the *projection of $\mathcal{P}_G$ with respect to I* and Σ is equal to the set of rules obtained by transforming each rule $r \in \mathcal{P}_G$ as follows:

- Delete r if a head atom H_i with a predicate from Σ exists in r such that $I \models H_i$;
- Delete each head atom H_i with a predicate from Σ if $I \not\models H_i$;
- Delete r if a body atom B_i with a predicate from Σ exists in r such that $I \not\models B_i$;
- Delete each body atom B_i with a predicate from Σ if $I \models H_i$.

Now I is a NM model of a $\mathcal{DL}$+log knowledge base $\mathcal{K}$ if it is a FOL model of $\mathcal{K}$ and I_{Σ_D} is a stable model of $\Pi(\mathsf{gr}(\mathcal{P}), I_{\Sigma_C \cup \Sigma_R})$.

We now show that each $\mathcal{DL}$+log knowledge base can be embedded into an equisatisfiable hybrid MKNF knowledge base.

Definition 4.7. *Let μ be a mapping of possibly negated $\mathcal{DL}$+log atoms to MKNF atoms as follows: (*i*) for A a DL-atom, $\mu(A) = A$; (*ii*) for A a non-DL-atom, $\mu(A) = \mathbf{K}\, A$; and (*iii*) for $\mathbf{not}\, A$ a negated non-DL-atom, $\mu(\mathbf{not}\, A) = \mathbf{not}\, A$. For r a $\mathcal{DL}$+log rule, $\mu(r)$ is the MKNF rule obtained by applying μ to each atom of r. For $\mathcal{P}$ a set of $\mathcal{DL}$+log rules, $\mu(\mathcal{P})$ is the set of MKNF rules obtained by applying μ to each rule $r \in \mathcal{P}$. Finally, for $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ a $\mathcal{DL}$+log knowledge base, $\mu(\mathcal{K}) = (\mathcal{O}, \mu(\mathcal{P}))$.*

Theorem 4.8. *A $\mathcal{DL}$+log knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ is NM satisfiable if and only if the MKNF knowledge base $\mu(\mathcal{K})$ is satisfiable.*

Proof. By the definition of universal quantification in $\mathcal{DL}$+log and MKNF, our claim trivially follows from the following property (*): $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ is NM satisfiable if and only if the MKNF knowledge base $\mu(\mathcal{K}_G)$ is satisfiable,

where $\mathcal{P}_G = \mathsf{gr}(\mathcal{P})$. We next prove (*). For a ground rule $r \in \mathcal{P}_G$, with r_{DL} we denote the rule obtained from r by deleting all non-DL-atoms.

($\Rightarrow$) Assume that $\mathcal{K}$ is satisfiable in a model I. Let M be the maximal set of first-order interpretations over $\triangle$ satisfying the following conditions:

- If $I \models r_{DL}$ for some $r \in \mathcal{P}_G$, then $J \models r_{DL}$ for each $J \in M$;
- If $I \not\models r_{DL}$ for some $r \in \mathcal{P}_G$, then $J \not\models r_{DL}$ for at least one $J \in M$;
- If $I \models A$ for some non-DL-atom A, then $J \models A$ for each $J \in M$;
- If $I \not\models A$ for some non-DL-atom A, then $J \not\models A$ for at least one $J \in M$;
- $J \models \mathcal{O}$ for each $J \in M$.

The set M is not empty since it contains the interpretation I. Furthermore, for each ground non-DL-atom A, we have $I \models A$ if and only if $M \models \mathbf{K}\, A$ if and only if $M \not\models \mathbf{not}\, A$; similarly, for each $r \in \mathcal{P}_G$, we have $M \models \mu(r_{DL})$ if and only if $I \models r_{DL}$. Clearly, $M \models \mu(r)$ for each $r \in \mathcal{P}_G$. By the definition of M, we have $M \models \mathcal{O}$, so $M \models \mu(\mathcal{K})$. To show that M satisfies the preference semantics of MKNF, assume that an MKNF interpretation $M' \supset M$ exists such that $(J', M', M) \models \pi(\mathcal{O}) \wedge \pi(\mu(\mathcal{P}_G))$ for some $J' \in M$. Let J be an interpretation that coincides with J' on the DL-atoms but, for each non-DL-atom A, we have $I \models A$ if and only if $M' \models \mathbf{K}\, A$. Clearly, $J \models \mathcal{O}$. Since $M' \supset M$, we have $J \subset I$. Furthermore, it is easy to see that $J \models \mathsf{GL}(\mathcal{P}_G, I)$, which contradicts the assumption that I is an NM model of $\mathcal{P}_G$.

($\Rightarrow$) Assume that $\mu(\mathcal{K}_G)$ is satisfiable in an MKNF model M. Let I' be any interpretation from M (note that M is not empty), and let I be an interpretation that coincides with I' on the DL-atoms but, for each non-DL-atom A, we have $I \models A$ if and only if $M \models \mathbf{K}\, A$. Similarly as in the previous paragraph, it is easy to see that $I \models r$ for each $r \in \mathcal{P}_G$ and, by the definition of M, we have $I \models \mathcal{O}$. Assume now that an interpretation $J \subset I$ exists such that $J \models \mathsf{GL}(\mathcal{P}_G, I)$. In exactly the same way as this was done for M and I in the ($\Rightarrow$) direction, we can construct an MKNF interpretation M' from J and show that $(J, M', M) \models \mu(\mathcal{K})$. Since $J \subset I$, we have $M' \supset M$, but this contradicts the assumption that M is an MKNF model of $\mu(\mathcal{K}_G)$. □

Hence, hybrid MKNF knowledge bases with mixed atoms in the rules semantically generalize $\mathcal{DL}$+log rules, regardless of any safety condition. If $\mathcal{DL}$+log rules are DL-safe, the corresponding MKNF rules are also DL-safe; furthermore, as shown in Section 4.3, rules with mixed atoms can be converted to rules with only modal atoms. Hence, the results from this paper provide an alternative reasoning algorithms for DL-safe $\mathcal{DL}$+log. Furthermore, we believe it is possible to extend our approach to handle weakly safe rules are well; to work out the details will be the part of our future work.

Note that our approach has a significant advantage over $\mathcal{DL}$+log: it allows DL-atoms to occur under modalities, so nonmonotonic reasoning can

Table 2: A Hybrid MKNF Knowledge Base about Cities

(24) $\mathit{historicCity} \sqsubseteq \exists \mathit{hasChurch}.\mathit{church}$	Historic cities have churches.
(25) $\mathit{church} \sqsubseteq \exists \mathit{designedBy}.\mathit{architect}$	Churches are designed by architects.
(26) $\mathbf{K}\,\mathit{famousCitizen}(x,z) \leftarrow \mathbf{K}\,\mathit{hasChurch}(x,y), \mathbf{K}\,\mathit{designedBy}(y,z), \mathbf{K}\,O(x), \mathbf{K}\,O(y), \mathbf{K}\,O(z)$	Architects are famous citizens in cities where they build their churches.
(27) $\exists \mathit{famousCitizen}.\top \sqsubseteq \mathit{interestingCity}$	Cities with famous people are interesting.
(28) $\mathit{historicCity}(\mathit{Barcelona})$	Barcelona is a historic city.
(29) $\mathit{hasChurch}(\mathit{Barcelona}, \mathit{SagradaFamilia})$	The famous church in Barcelona...
(30) $\mathit{designedBy}(\mathit{SagradaFamilia}, \mathit{Gaudi})$	...was designed by Antonio Gaudi.
(31) $\mathit{seasideCity} \sqsubseteq \exists \mathit{hasRegion}.\mathit{beach}$	Seaside cities have a beach.
(32) $\mathit{beach} \sqsubseteq \mathit{recreational}$	Beaches are for recreation.
(33) $\exists \mathit{hasRegion}.\mathit{recreational} \equiv \mathit{livableCity}$	Livable cities provide for recreation.
(34) $\mathit{portCity}(\mathit{Barcelona})$	Barcelona is a city with a port.
(35) $\mathit{portCity}(\mathit{Hamburg})$	Hamburg is a city with a port.
(36) $\neg \mathit{seasideCity}(\mathit{Hamburg})$	Hamburg is not a seaside city.
(37) $\mathbf{K}\,\mathit{DesignOK}(x) \leftarrow \mathbf{K}\,\mathit{designedBy}(x,y), \mathbf{K}\,O(x), \mathbf{K}\,O(y)$	Auxiliary for the following rule.
(38) $\leftarrow \mathbf{K}\,\mathit{church}(x), \mathbf{not}\,\mathit{DesignOK}(x), \mathbf{K}\,O(x)$	Each church must have an architect.
(39) $\mathit{church}(\mathit{HolyFamily})$	Holy Family is a church.
(40) $\mathit{HolyFamily} \approx \mathit{SagradaFamilia}$	Definition of synonyms.
(41) $\neg \mathit{seasideCity} \equiv \mathit{notSC}$	An atomic name for $\neg \mathit{seasideCity}$.
(42) $\mathbf{K}\,\mathit{seasideCity}(x) \leftarrow \mathbf{K}\,\mathit{portCity}(x), \mathbf{not}\,\mathit{notSC}(x), \mathbf{K}\,O(x)$	Port cities are usually at the seaside.
(43) $\mathbf{K}\,\mathit{Suggest}(x) \leftarrow \mathbf{K}\,\mathit{livableCity}(x), \mathbf{K}\,\mathit{historicCity}(x)$	Suggest to visit livable and historic cities.
(44) $\neg \mathit{livableCity} \equiv \mathit{notLivableCity}$	An atomic name for $\neg \mathit{livable}$.
(45) $\mathbf{K}\,\mathit{Consider}(x) \leftarrow \mathbf{not}\,\mathit{notLivableCity}(x), \mathbf{K}\,O(x)$	Take cities that are not known to be unlivable into consideration as well.

Note: DL-predicates start with a lowercase, and non-DL-predicates with an uppercase letter. There is an assertion $O(\alpha)$ for each object α.

be applied equally to DL- and non-DL-atoms. Thus, in $\mathcal{DL}$+log one cannot state that "all birds fly but penguins are an exception," which is easily possible with hybrid MKNF knowledge bases.

5 Example

At first glance, our proposal may seem to be difficult to use and understand. However, we believe MKNF rules to be quite intuitive: just read $\mathbf{K}\,A$ as "A is known to hold" and $\mathbf{not}\,A$ as "is it possible for A not to hold." We demonstrate this on the following example. Imagine a system helping us to decide where to go on holiday using the tourism ontology $\mathcal{K}$ from Table 2.

The impact of DL-safety is demonstrated by axioms (24)–(27). By (24) and (25), each historic city α has at least one church β, which has at least one architect γ. By (26), γ is a famous citizen of α so, by (27), α is an interesting city. Now if (26) were a normal (non-DL-safe, first-order) rule, one might perform this inference for *any* individuals α, β, and γ, which would thus imply $\mathcal{K} \models \mathit{historicCity} \sqsubseteq \mathit{interestingCity}$. However, (26) is DL-safe—all variables occur in an atom with the predicate O. Hence, it is applicable only to the individuals *known in the ABox by name*, and not to those introduced by the existential quantifier, so we cannot conclude that

interestingCity subsumes *historicCity*. Note that (26) could be stated in $\mathcal{SROIQ}$ [21] (a DL that allows for certain types of role chaining and inclusion axioms), and this would correctly imply the subsumption relationship.

Whereas DL-safety usually restricts the subsumption inferences, it typically has less impact on ABox query answering. Namely, (28)–(30) specify the names of a church in Barcelona and its architect. All variables in (26) can be bound to known individuals, so $\mathcal{K} \models \mathit{famousCitizen}(\mathit{Barcelona}, \mathit{Gaudi})$; by (27), we derive $\mathcal{K} \models \mathit{interestingCity}(\mathit{Barcelona})$. Hence, DL-safety is a compromise that provides for ABox query answering at the expense of some subsumption inferences if expressivity beyond $\mathcal{SROIQ}$ is needed, but without losing decidability. DL-safety is crucial for nonmonotonic reasoning: without it, most nonmonotonic logics with existential quantification are not even semidecidable.

Consider an integrity constraint requiring that an architect should be explicitly specified for each explicitly mentioned church. One might intuitively write the rule $\leftarrow \mathbf{K}\, \mathit{church}(x), \mathbf{not}\, \mathit{designedBy}(x, y), \mathbf{K}\, O(x), \mathbf{K}\, O(y)$ (paraphrased as "it is an error to have a known church without a known designer"). However, this rule is incorrect: all variables in rules are universally quantified, so this rule requires each church to be connected through *designedBy* to each other object. To formulate the integrity constraint correctly, we introduce the auxiliary rule (37) which projects the variable y from $\mathit{designedBy}(x, y)$, and then use the result in (38) to identify the churches without a designer.

Nonmonotonic formalisms usually assume UNA. Let us for the moment assume that $\mathcal{K}$ does not contain (40). We would then intuitively expect (38) to be violated, since the designer of *HolyFamily* has not been specified. However, without UNA, $\mathcal{K}$ would be satisfiable in a model where *HolyFamily* and *SagradaFamilia* are interpreted as the same object. In fact, *HolyFamily* might become equal to any other object, so **not** could not any more be intuitively read as "assumed not to hold." The semantics of **not** without UNA is counterintuitive, so nonmonotonic formalisms usually assume it.

In contrast, many DLs do not require UNA, but allow explicit equality statements to define synonyms. To overcome this difference, we adopt a special approach in defining the semantics of hybrid MKNF knowledge bases. Roughly speaking, we assume UNA at the level of MKNF to ensure that **not** has an intuitive semantics; however, at the level of $\mathcal{DL}$, we consider $\approx$ to be interpreted as a congruence relation. This subjects $\approx$ to nonmonotonic reasoning just like any other predicate, so two individuals are assumed to be equal only if this there is evidence for doing so. Returning to our example, we make *HolyFamily* and *SagradaFamilia* synonyms by (40), which then makes (38) satisfied for (30) and (39).

Rule (42) asserts the common-sense knowledge that port cities are usually at the seaside, allowing us to conclude $\mathcal{K} \models \mathit{seasideCity}(\mathit{Barcelona})$. However, (42) allows for exceptions: the atom $\mathbf{not}\, \mathit{notSC}(x)$ basically says

"if not proven not to be at the seaside." (Axiom (41) is needed because only atomic concepts can occur in the rules.) By (36), Hamburg is an exception (it is located on the river Elbe), so the conclusion $\mathcal{K} \models \mathit{seasideCity}(\mathit{Hamburg})$ of (42) is suppressed, as it would lead to contradiction.

The rule (43) is intended as a query that suggests which cities to visit. Even though the conclusion $\mathit{seasideCity}(\mathit{Barcelona})$ was derived by nonmonotonic reasoning, it implies further conclusions through monotonic reasoning. Namely, (31)–(33) imply $\mathcal{K} \models \mathit{livableCity}(\mathit{Barcelona})$, which is derived by standard DL reasoning involving unnamed individuals (introduced by $\exists \mathit{hasRegion}.\mathit{Beach}$). Hence, $\mathcal{K} \models \mathit{Suggest}(\mathit{Barcelona})$.

Finally, (45) shows how negation-as-failure of logic programming is layered over open-world semantics of description logics. Intuitively, MKNF performs open- and closed-world inferences "in parallel." For example, the conclusions $\mathcal{K} \not\models \mathit{livableCity}(\mathit{Hamburg})$ and $\mathcal{K} \not\models \neg\mathit{livableCity}(\mathit{Hamburg})$ hold according to the usual DL semantics. By reformulating these questions with closed-world interpretation in mind, we get $\mathcal{K} \models \mathbf{not}\, \mathit{livableCity}(\mathit{Hamburg})$ (Hamburg is not known to be livable) and $\mathcal{K} \models \mathbf{not}\, \mathit{notLivable}(\mathit{Hamburg})$ (Hamburg is not known not to be livable either). Hence, (45) allows to conclude $\mathit{Consider}(\mathit{Hamburg})$—even though we do not know for sure that Hamburg is a livable city, we do not know the opposite either, so it might still be worth a visit. Intuitively speaking, the DL part of $\mathcal{K}$ is interpreted under open-world semantics; however, **K** and **not** allow the user to put on "closed-world glasses" and examine the nonmonotonic consequences of the DL part. By using these consequences in rules, one can enforce new nonmonotonic conclusions.

6 Reasoning with Modally Closed MKNF Formulae

We develop algorithms for reasoning with hybrid MKNF knowledge bases in two stages. In this section, we define general algorithms capable of handling various types of modally closed MKNF formulae. In Section 7 we specialize these general algorithms to the types of formulae obtained by translating hybrid MKNF knowledge bases.

We present five different algorithms for reasoning with different types of modally closed MKNF formulae because all these cases differ in the complexity of reasoning. The first algorithm handles flat MKNF formulae, the second one handles positive MKNF formulae, the third one handles positive nondisjunctive MKNF formulae, the fourth one handles stratified nondisjunctive MKNF formulae, and the fifth one handles nonstratified nondisjunctive MKNF formulae.

Our algorithms are closely related to the algorithms for propositional MBNF [32] and for MBNF($\mathcal{K}$) [33]—an MBNF-based extension of multi-modal logics for knowledge and belief.

6.1 Overview

Before presenting the algorithms, we briefly discuss the basic principles behind them. Satisfiability of some formula is usually demonstrated by constructing a model of the formula. However, an MKNF model M of an MKNF formula σ is a set of first-order interpretations, and is as such infinite. For a practical algorithm, we need an appropriate finite representation of M.

A possible solution is not to represent M directly, but to compute a first-order formula φ such that M is exactly the set of first-order models of φ; this is usually written as $M = \{I \mid I \models \varphi\}$. As shown in [8], this is not possible in general; however, this is possible for the modally closed subset of MKNF, which we consider in this paper. Similarly as this was done in [32, 33], we show that φ is uniquely defined through a partition (P, N) of modal atoms of σ into positive and negative ones. The formula φ then corresponds to the *objective knowledge*, written ob_P, and it can be computed from the atoms chosen to be positive in a straightforward way.

For different fragments of MKNF, we can adopt different strategies for computing (P, N). For flat formulae, we must guess such a partition. Since modal atoms of σ are general first-order formulae, not each partition will make sense. For example, let $\sigma = \mathbf{K}[p \wedge (p \supset r)] \wedge [\mathbf{K}\, r \supset \mathbf{K}\, s]$. Observe that $p \wedge (p \supset r) \models r$, so assuming that $\mathbf{K}[p \wedge (p \supset r)]$ is positive and $\mathbf{K}\, r$ is negative is inconsistent. Therefore, we eliminate guesses which lead to such inconsistencies. Furthermore, we check whether σ is true when its modal atoms are replaced with their values in (P, N). Finally, we need to ensure that the model defined by the objective knowledge contained in the partition satisfies the MKNF preference semantics. For this, we try to guess another partition which also defines a model.

For the (stratified and nonstratified) nondisjunctive fragment of MKNF, we show that we can construct the objective knowledge in a bottom-up fashion, much like this is done in ordinary datalog. In this way we can eliminate unnecessary guessing and obtain better complexity results.

6.2 The General Case

Definition 6.1. *Let σ be a flat modally closed MKNF formula. The* set of **K**-atoms *of σ, written $\mathsf{KA}(\sigma)$, is the smallest set containing (*i*) all modal atoms $\mathbf{K}\,\xi$ occurring in σ, and (*ii*) an atom $\mathbf{K}\,\xi$ for each modal atom $\mathbf{not}\,\xi$ occurring in σ.*

Let P and N be disjoint sets of modal atoms. With $\sigma[\mathbf{K}, P, N]$ we denote the formula obtained by replacing each strict occurrence of a modal

atom $\mathbf{K}\,\xi$ *in* σ *with* true *if* $\mathbf{K}\,\xi \in P$*, and with* false *if* $\mathbf{K}\,\xi \in N$*. Similarly, with* $\sigma[\mathbf{not}, P, N]$ *we denote the formula obtained by replacing each strict occurrence of a modal atom* $\mathbf{not}\,\xi$ *in* σ *with* true *if* $\mathbf{K}\,\xi \in N$*, and with* false *if* $\mathbf{K}\,\xi \in P$*. Finally, with* $\sigma[P, N]$ *we denote* $\sigma[\mathbf{K}, P, N][\mathbf{not}, P, N]$*.*

We comment on a technical difference between Definition 6.1 and similar definitions in [32, 33]. Here, we represent the values of all modal atoms by considering only **K**-atoms; that is, the value of a negative modal atom $\mathbf{not}\,\xi$ is represented by the value of the dual positive modal atom $\mathbf{K}\,\xi$. This makes the presentation somewhat simpler and it eliminates obviously inconsistent partitions (such as choosing both $\mathbf{K}\,a$ and $\mathbf{not}\,a$ to be true).

We now define the objective knowledge implicit in a set of **K**-atoms P:

Definition 6.2. *The* objective knowledge *of a set* P *of flat modally closed* **K***-atoms is the following first-order formula:*

$$\mathsf{ob}_P = \bigwedge_{K\xi \in P} \xi$$

We now establish a link between sets of first-order interpretations and a partition of a set of **K**-atoms.

Definition 6.3. *A set of first-order interpretations* M induces *a partition* (P, N) *of a set of flat modally closed* **K***-atoms* S *if* $\mathbf{K}\,\xi \in P$ *implies* $M \models \mathbf{K}\,\xi$ *and* $\mathbf{K}\,\xi \in N$ *implies* $M \not\models \mathbf{K}\,\xi$*.*

The following corollary follows immediately from Definition 6.3 and the definition of satisfiability of an MKNF formula in an MKNF structure:

Corollary 6.4. *Let* σ *be a flat modally closed MNKF formula,* M *an MKNF interpretation, and* (P, N) *a partition of* $\mathsf{KA}(\sigma)$ *induced by* M*. Then, for each* $I \in M$*, we have* $(I, M, M) \models \sigma$ *if and only if* $\sigma[P, N] = \mathsf{true}$*.*

We now show that ob_P characterizes the MKNF models of σ:

Lemma 6.5. *Let* σ *be a flat modally closed MNKF formula,* M *an MKNF model of* σ*, and* (P, N) *a partition of* $\mathsf{KA}(\sigma)$ *induced by* M*. Then,* M *is equal to the set of interpretations* $M' = \{I \mid I \models \mathsf{ob}_P\}$*.*

Proof. Let I be any interpretation from M. The set M induces the partition (P, N) so, for each $\mathbf{K}\,\xi \in P$, we have $M \models \mathbf{K}\,\xi$, which implies $I \models \xi$. Clearly, $I \models \mathsf{ob}_P$, which proves $M \subseteq M'$.

To show that $M' = M$, assume that $M' \setminus M$ contains an interpretation I'. Then, for each $\mathbf{K}\,\xi \in P$, by Definition 6.2, we have $(I', M', M) \models \mathbf{K}\,\xi$. Furthermore, for each $\mathbf{K}\,\xi \in N$, because M induces the partition (P, N), we have $M \not\models \mathbf{K}\,\xi$, so, because $M \subset M'$, we have $(I', M', M) \not\models \mathbf{K}\,\xi$ as well. For each **not**-atom, $(I', M', M) \models \mathbf{not}\,\xi$ if and only if $(I', M, M) \models \mathbf{not}\,\xi$. Since σ is flat, its value in (I', M', M) is completely defined by the values of the modal atoms. Hence, $(I', M', M) \models \sigma$, which contradicts the assumption that M is an MKNF model of σ. □

We now identify the partitions of $\mathsf{KA}(\sigma)$ which are not contradictory:

Definition 6.6. *A partition (P, N) of a set of flat modally closed* $\mathbf{K}$*-atoms S is* consistent *if $\mathbf{K}\,\xi \in N$ implies $\mathsf{ob}_P \not\models \xi$.*

The following properties follow immediately from the definition of consistency:

Corollary 6.7. *Let (P, N) be a consistent partition of a set of flat modally closed* $\mathbf{K}$*-atoms S, and let $M = \{I \mid I \models \mathsf{ob}_P\}$. Then, $\mathbf{K}\,\xi \in P$ if and only if $M \models \mathbf{K}\,\xi$ if and only if $M \not\models \mathbf{not}\,\xi$—that is, (P, N) is the partition of S induced by M.*

Corollary 6.8. *Each partition (P, N) of a set of flat modally closed* $\mathbf{K}$*-atoms S induced by a set of first-order interpretations M is consistent.*

We now define what it means to evaluate some modally closed MKNF formula ψ in a partition (P, N) of $\mathbf{K}$-atoms. Note that the following definition allows ψ to contain nested occurrences of modal operators, as well as modal atoms not occurring in P or N.

Definition 6.9. *Let P be a set of flat modally closed* $\mathbf{K}$*-atoms. A flat modally closed atom $\mathbf{K}\,\xi$* evaluates to true *in P if and only if $\mathsf{ob}_P \models \xi$; similarly, a flat modally closed atom $\mathbf{not}\,\xi$* evaluates to true *in P if and only if $\mathsf{ob}_P \not\models \xi$. The value of nested modally closed atoms is defined by exhaustively replacing each flat modal subatom with its value in P. The* value *of a modally closed MKNF formula ψ in P, written $\psi[P]$, is the formula obtained by replacing each strict modal atom of ψ with its value in P.*

Lemma 6.10. *Let P be a set of flat modally closed* $\mathbf{K}$*-atoms, ψ a modally closed MKNF formula, and $M = \{I \mid I \models \mathsf{ob}_P\}$. Then, $(I, M, M) \models \psi$ for each $I \in M$ if and only if $\mathsf{ob}_P \models \psi[P, N]$.*

Proof. By a straightforward induction on the depth of the modal atoms in ψ, one can show that each modal atom $\mathbf{K}\,\xi$ ($\mathbf{not}\,\xi$) from ψ evaluates to true if and only if $M \models \mathbf{K}\,\xi$ ($M \models \mathbf{not}\,\xi$). Hence, for each $I \in M$, we have $(I, M, M) \models \psi$ if and only if $(I, M, M) \models \psi[P, N]$, which now immediately implies the claim of the lemma. □

The procedure $\mathsf{not\text{-}entails}(\sigma, \psi)$ for checking whether $\sigma \not\models_{\mathsf{MKNF}} \psi$ is given in Algorithm 1.

Theorem 6.11. *For a flat modally closed MKNF formula σ and a modally closed MKNF formula ψ, the algorithm $\mathsf{not\text{-}entails}(\sigma, \psi)$ returns true if and only if $\sigma \not\models_{\mathsf{MKNF}} \psi$.*

Algorithm 1 Checking Entailment in Flat MKNF

Algorithm: not-entails(σ, ψ)
Input:
σ: a flat modally closed MKNF formula
ψ: a modally closed MKNF formula (not necessarily flat)
Output:
true if $\sigma \not\models_{\mathsf{MKNF}} \psi$; false otherwise

if a partition (P, N) of $\mathsf{KA}(\sigma)$ exists such that
1. $\sigma[P, N]$ evaluates to true, and
2. ob_P is satisfiable, and
3. $\neg\xi \wedge \mathsf{ob}_P$ is satisfiable for each $\mathbf{K}\,\xi \in N$, and
4. **for** $\sigma' = \sigma[\mathbf{not}, P, N]$ and each partition (P', N') of P such that $N' \neq \emptyset$
   (a) $\sigma'[P', N \cup N']$ evaluates to false, or
   (b) $\mathsf{ob}_{P'}$ is unsatisfiable, or
   (c) $\neg\xi \wedge \mathsf{ob}_{P'}$ is unsatisfiable for some $\mathbf{K}\,\xi \in N'$

   and
5. $\mathsf{ob}_P \wedge \neg\psi[P]$ is satisfiable

then return true; otherwise **return** false

Proof. ($\Rightarrow$) If not-entails(σ, ψ) returns true, a partition (P, N) of $\mathsf{KA}(\sigma)$ satisfying conditions (1)–(5) exists. We show that $M = \{I \mid I \models \mathsf{ob}_P\}$ is an MKNF model of σ. By Condition (2), M is not empty. By Condition (3), (P, N) is consistent, so it defines the truth of the modal atoms in σ by Corollary 6.7. By Condition (1) and Corollary 6.4, $(I, M, M) \models \sigma$ for each $I \in M$. To verify that M satisfies the preference semantics of MKNF, assume that $M' \supset M$ exists such that $(I', M', M) \models \sigma$ for each $I' \in M'$. Then, M' induces a partition (P'', N'') of $\mathsf{KA}(\sigma)$, which is consistent by Corollary 6.8. Since $M' \neq M$, clearly $(P'', N'') \neq (P, N)$. Because $M' \supset M$, we have that $M' \models \mathbf{K}\,\xi$ implies $M \models \mathbf{K}\,\xi$ for each modal atom $\mathbf{K}\,\xi$, which implies $P'' \subset P$. Hence, (P'', N'') can be equivalently represented by a partition (P', N') of P with $N' \neq \emptyset$. By Lemma 6.5, $M' = \{I \mid I \models \mathsf{ob}_{P'}\}$, which, together with Corollary 6.7, implies that $(P', N \cup N')$ defines the truth of the modal atoms occurring in σ'. By Corollary 6.8, $(P', N \cup N')$ is consistent, which falsifies Condition (c). Because $M' \neq \emptyset$, $\mathsf{ob}_{P'}$ is satisfiable, which falsifies Condition (b). Furthermore, $(I', M', M) \models \sigma$ for each $I' \in M'$, so $(I', M', M) \models \sigma'$ for each $I' \in M'$ as well. By Corollary 6.4, $\sigma'[P', N \cup N'] = \mathsf{true}$, which falsifies Condition (a). By assumption, Condition (4) is satisfied for all partitions (P', N') of P, so such M' cannot exist—that is, M is an MKNF model of σ. By Condition (5), $\mathsf{ob}_P \not\models \psi[P]$, so, by Lemma 6.10, $(I, M, M) \not\models \psi$ for some $I \in M$. We therefore conclude $\sigma \not\models_{\mathsf{MKNF}} \psi$.

($\Leftarrow$) If $\sigma \not\models_{\mathsf{MKNF}} \psi$, then an MKNF model M of σ exists such that $M \not\models \psi$, and it induces a partition (P, N) of $\mathsf{KA}(\sigma)$. By Corollary 6.8,

(P, N) is consistent, which validates Condition (3). By Lemma 6.5, we have $M = \{I \mid I \models \mathsf{ob}_P\}$, which, together with Corollary 6.7, implies that (P, N) defines the truth of the modal atoms occurring in σ. Furthermore, $(I, M, M) \models \sigma$ for each $I \in M$ implies $\sigma[P, N] = \mathsf{true}$ by Corollary 6.4, which validates Condition (1). Since $M \neq \emptyset$, ob_P is satisfiable, which validates Condition (2). Since $M \not\models \psi$, by Lemma 6.10, $\mathsf{ob}_P \not\models \psi[P]$, which validates Condition (5). It remains to show that Condition (4) is also validated. Assume that Condition (4) is falsified. Then, a partition (P', N') of P with $N' \neq \emptyset$ exists for which conditions (a)–(c) are falsified. Let $M' = \{I \mid I \models \mathsf{ob}_{P'}\}$. Since $(P, N) \neq (P', N \cup N')$, clearly $M' \neq M$. By Condition (b), $M' \neq \emptyset$. By Condition (c), $(P', N \cup N')$ is consistent, so it defines the truth values of modal atoms in σ' by Corollary 6.7. By Condition (a), $\sigma'[P', N \cup N'] = \mathsf{true}$, so, by Corollary 6.4, $(I', M', M) \models \sigma'$ and $(I', M', M) \models \sigma$ for each $I' \in M'$. Since $P' \subset P$, clearly $\mathsf{ob}_{P'} \models \mathsf{ob}_P$, so $M' \supset M$. Hence, M is not an MKNF model of σ, which is a contradiction. Thus, Condition (4) is satisfied and $\mathsf{not\text{-}entails}(\sigma, \psi)$ returns true. □

From the proof of Theorem 6.11 we can see that, if $\mathsf{not\text{-}entails}(\sigma, \psi)$ returns true, then it yields a partition (P, N) such that $M = \{I \mid I \models \mathsf{ob}_P\}$ is the MKNF model of σ invalidating ψ. We next estimate the complexity of the algorithm.

Theorem 6.12. *Assuming that the satisfiability of first-order formulae in Algorithm 1 is decidable with complexity $\mathcal{C}$, the complexity of the algorithm $\mathsf{not\text{-}entails}(\sigma, \psi)$ is in $\mathcal{E}^{\mathcal{E}}$, where $\mathcal{E} = \mathrm{NP}$ if $\mathcal{E} \subseteq \mathrm{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise.*

Proof. A partition (P, N) can be guessed in time polynomial in $|\sigma|$, Condition (1) can be checked in polynomial time, and Conditions (2) and (3) can be verified by a polynomial number of calls to an oracle running in $\mathcal{C}$; hence, these actions can be performed in $\mathcal{E}$. Consider now Condition (4). A partition (P', N') can be guessed in polynomial time, Condition (a) can be checked in polynomial type, and Conditions (b)–(c) can be falsified by a polynomial number of calls to an oracle running in $\mathcal{C}$; hence, these actions can be performed in $\mathcal{E}$. Thus, Condition (4) can be verified by an oracle running in co$\mathcal{E}$. Finally, computing $\psi[P]$ requires $|\psi|$ calls to an oracle running in $\mathcal{C}$, and checking Condition (5) requires another call to an oracle running in $\mathcal{C}$. Hence, the entire algorithm runs in $\mathcal{E}^{\mathcal{E}}$. □

6.3 The Positive Case

As we show next, the algorithm from the previous section can be simplified for checking entailment of a positive modal atom from a positive MKNF formula. Note that the following lemma holds for an arbitrary positive MKNF formula (not necessarily flat or modally closed).

Algorithm 2 Checking Entailment in Positive Flat MKNF

Algorithm: $\mathsf{not\text{-}entails}^{+}(\sigma, \mathbf{K}\,\psi)$
Input:
σ: a flat modally closed MKNF formula
ψ: a closed first-order formula
Output:
true if $\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$; false otherwise

if a partition (P, N) of $\mathsf{KA}(\sigma)$ exists such that
1. $\sigma[P, N]$ evaluates to true, and
2. ob_P is satisfiable, and
3. $\neg\xi \wedge \mathsf{ob}_P$ is satisfiable for each $\mathbf{K}\,\xi \in N$, and
4. $\mathsf{ob}_P \wedge \neg\psi$ is satisfiable

then return true; otherwise **return** false

Lemma 6.13. *A positive MKNF formula σ is MKNF satisfiable if and only if it is* S5 *satisfiable. Furthermore, $\sigma \models_{\mathsf{MKNF}} \mathbf{K}\,\psi$ if and only if $\sigma \models_{\mathsf{S5}} \mathbf{K}\,\psi$, for a first-order formula ψ.*

Proof. (Claim 1.) The ($\Rightarrow$) direction follows immediately from the definition of MKNF models. For the ($\Leftarrow$) direction, simply observe that, for each S5 model M, a maximal interpretation $M' \supseteq M$ exists such that $M' \models \sigma$. Since σ is positive, M' is an MKNF model of σ. Hence, to check MKNF satisfiability of σ, it suffices to find any S5 model of σ.

(Claim 2.) The ($\Leftarrow$) direction follows immediate from the definition of MKNF models. For the ($\Rightarrow$) direction, assume that $\sigma \not\models_{\mathsf{S5}} \psi$; hence, an S5 model M of σ exists such that $I \not\models \psi$ for some $I \in M$. Since σ is positive, it has an MKNF model M' such that $M' \supseteq M$. Clearly, $I \in M'$, so $M' \not\models \mathbf{K}\,\psi$ and $\sigma \not\models_{\mathsf{MKNF}} \psi$. □

Based on this lemma, for a flat positive modally closed MKNF formula σ and a first-order formula ψ, we define the $\mathsf{not\text{-}entails}^{+}(\sigma, \mathbf{K}\,\psi)$ as shown in Algorithm 2. The following claim follows immediately from Lemma 6.13:

Theorem 6.14. *For a flat modally closed MKNF formula σ and a closed first-order formula ψ, the algorithm* $\mathsf{not\text{-}entails}^{+}(\sigma, \mathbf{K}\,\psi)$ *returns* true *if and only if $\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$. Assuming that the satisfiability of first-order formulae in Algorithm 2 is decidable in $\mathcal{C}$, the complexity of the algorithm* $\mathsf{not\text{-}entails}^{+}(\sigma, \mathbf{K}\,\psi)$ *is in $\mathcal{E}$, where $\mathcal{E} = \mathrm{NP}$ if $\mathcal{E} \subseteq \mathrm{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise.*

Proof. (Claim 1.) From the proof of Theorem 6.11, one can see that Condition (4) of $\mathsf{not\text{-}entails}(\sigma, \mathbf{K}\,\psi)$ ensures that a model induced by a partition (P, N) of $\mathsf{KA}(\sigma)$ satisfies the preference semantics of MKNF. By eliminating this condition, we make the algorithm $\mathsf{not\text{-}entails}^{+}(\sigma, \mathbf{K}\,\psi)$ check for S5 satisfiability, which, by Lemma 6.13, can be used to check entailment

$\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$. Note that ψ is a first-order formula, so Condition (4) is a simplification of Condition (5) of $\mathsf{not\text{-}entails}(\sigma, \mathbf{K}\,\psi)$.

(Claim 2.) A partition (P, N) can be guessed in time polynomial in $|\sigma|$, Condition (1) can be checked in polynomial time, Conditions (2) and (3) can be verified by a polynomial number of calls to an oracle running in $\mathcal{C}$, and Condition (4) can be verified by one call to an oracle running in $\mathcal{C}$. Hence, all conditions can be checked in $\mathcal{E}$. □

6.4 The Positive Nondisjunctive Case

We now turn our attention to positive nondisjunctive MKNF programs. Namely, such programs are either MKNF unsatisfiable, or they are MKNF satisfiable in a single MKNF model that corresponds to the least fixpoint of a certain operator. We show that positive nondisjunctive MKNF programs can have at most one MKNF model.

Lemma 6.15. *Let σ be a positive nondisjunctive MKNF program. If M_1 and M_2 are sets of interpretations such that $M_1 \models \sigma$ and $M_2 \models \sigma$, then $M_1 \cup M_2 \models \sigma$ as well.*

Proof. Consider a rule $r \in \sigma$ of the form (1). If $M_1 \cup M_2 \not\models \mathbf{K}\,B_i^+$ for some $1 \leq i \leq n$, then clearly $M_1 \cup M_2 \models r$. Assume now that r is such that $M_1 \cup M_2 \models \mathbf{K}\,B_i^+$ for each $1 \leq i \leq n$. Because $M_1 \subseteq M_1 \cup M_2$ and $M_2 \subseteq M_1 \cup M_2$, clearly $M_1 \models \mathbf{K}\,B_i^+$ and $M_2 \models \mathbf{K}\,B_i^+$, for each $1 \leq i \leq n$. By assumption that $M_1 \models r$ and $M_2 \models r$, we get $M_1 \models \mathbf{K}\,H$ and $M_2 \models \mathbf{K}\,H$. But then, $M_1 \cup M_2 \models \mathbf{K}\,H$, so $M_1 \cup M_2 \models r$ as well. □

The previous lemma immediately implies the following theorem:

Theorem 6.16. *Each MKNF satisfiable positive nondisjunctive MKNF program σ has exactly one MKNF model.*

Proof. Assume that M_1 and M_2 are MKNF models of σ and that $M_1 \neq M_2$. Let $M = M_1 \cup M_2$; obviously, $M_1 \subset M$ and $M_2 \subset M$. By Lemma 6.15, $M \models \sigma$, which contradicts the assumption that M_1 and M_2 are MKNF models of σ. □

Hence, an MKNF satisfiable positive MKNF program σ has a unique model, which, by Lemma 6.5, can be represented by a partition (P, N) of $\mathsf{KA}(\sigma)$. We now show how to compute this partition in a deterministic way.

Definition 6.17. *For σ a positive nondisjunctive MKNF program, let R_σ, D_σ, and T_σ be the operators defined on the subsets of $\mathsf{KA}(\sigma)$ as follows:*

$$R_\sigma(S) = S \cup \{\mathbf{K}\,H \mid \sigma \text{ contains a rule of the form (1) such that } \mathbf{K}\,B_i^+ \in S \text{ for each } 1 \leq i \leq n\}$$

$$D_\sigma(S) = \{\mathbf{K}\,\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\sigma) \text{ and } \mathsf{ob}_S \models \xi\}$$

$$T_\sigma(S) = R_\sigma(S) \cup D_\sigma(S)$$

Intuitively, $R_\sigma(S)$ computes the immediate consequences of S with respect to the rules of σ, and $D_\sigma(S)$ computes the immediate consequences of S with respect to the objective knowledge accumulated in S. We point out a small technical difference in the definition of R_σ to the usual case. For ordinary datalog, it suffices to define $R_\sigma(S)$ as the set of head atoms of the rules whose body atoms are satisfied in S—that is, it is not necessary to explicitly append S to the result set. In our case, however, the operators R_σ and D_σ interact. The operator $D_\sigma(S)$ may derive facts that do not occur in the rules, so the definition of R_σ merely ensures that these facts are not lost by computing the consequences of the rules. This ensures $S \subseteq R_\sigma(S)$, which we use extensively in our proofs. Furthermore, $S \subseteq D_\sigma(S)$ by the definition of D_σ.

Observe that, for S a set of modal atoms such that ob_S is unsatisfiable, $D_\sigma(S) = \mathsf{KA}(\sigma)$. The following property of T_σ is easy to prove:

Lemma 6.18. *The operator $T_\sigma(S)$ is monotone on the lattice of subsets of $\mathsf{KA}(\sigma)$—that is, $S \subseteq S'$ implies $T_\sigma(S) \subseteq T_\sigma(S')$, for each $S, S' \subseteq \mathsf{KA}(\sigma)$.*

Proof. Observe that, if $\mathbf{K}\,H \in R_\sigma(S)$, then either (i) $\mathbf{K}\,H \in S$, but then $\mathbf{K}\,H \in R_\sigma(S')$ holds trivially, or (ii) σ contains a rule of the form (1) such that $\mathbf{K}\,B_i^+ \in S$ for all $1 \leq i \leq n$, but then $\mathbf{K}\,B_i^+ \in S'$ for all $1 \leq i \leq n$, so $\mathbf{K}\,H \in R_\sigma(S')$ as well. Hence, R_σ is monotone. Monotonicity of D_σ holds trivially because $\mathsf{ob}_S = \bigwedge_{\mathbf{K}\,\xi \in S} \xi$ implies each of the conjuncts ξ. Now monotonicity of R_σ and D_σ implies monotonicity of T_σ as well. □

Hence, T_σ is a monotone operator on a complete lattice of the subsets of $\mathsf{KA}(\sigma)$, so, by the well-known Knaster-Tarski's theorem, it has a unique least fixpoint, which we denote with T_σ^ω. As usual, T_σ^ω can be computed by setting $S_0 = \emptyset$ and computing $S_i = T_\sigma(S_{i-1})$ for $i > 0$; since $\mathsf{KA}(\sigma)$ is finite, after some finite number n we shall have $S_n = S_{n+1} = \ldots = T_\sigma^\omega$. We next show the following two properties:

Lemma 6.19. *Let σ be a positive nondisjunctive MKNF program with an MKNF model M, and let (P, N) be the partition of $\mathsf{KA}(\sigma)$ induced by M. Then, $T_\sigma(P) = P$.*

Proof. By Corollary 6.8, the partition (P, N) is consistent, meaning that there is no $\mathbf{K}\,\xi \in N$ such that $\mathsf{ob}_P \models \xi$; in other words, $D_\sigma(P) = P$. Furthermore, P is exactly the subset of $\mathsf{KA}(\sigma)$ that is true in M so, since M satisfies all rules from σ, we have $R_\sigma(P) \subseteq P$. Furthermore, $P \subseteq R_\sigma(P)$ holds by the definition of R_σ, so we conclude that $R_\sigma(P) = P$. Clearly, this implies $T_\sigma(P) = P$. □

Lemma 6.20. *Let σ be a positive nondisjunctive MKNF program, P a subset of $\mathsf{KA}(\sigma)$ such that $T_\sigma(P) = P$, and $M = \{I \mid I \models \mathsf{ob}_P\}$. Then, $M \models \sigma$.*

Proof. $T_\sigma(P) = P$ implies $R_\sigma(P) \subseteq P$ and $D_\sigma(P) \subseteq P$. Furthermore, $P \subseteq R_\sigma(P)$ by the definition of R_σ, so $R_\sigma(P) = P$; similarly, $P \subseteq D_\sigma(P)$ by the definition of D_σ, so $D_\sigma(P) = P$. Now, $D_\sigma(P) = P$ implies that the partition $(P, \mathsf{KA}(\sigma) \setminus P)$ is consistent so, by Corollary 6.7, P is exactly the set of modal atoms from $\mathsf{KA}(\sigma)$ that are true in M. But then, $R_\sigma(P) = P$ implies that all rules from σ are satisfied in M, so $M \models \sigma$. □

We are now ready to present the main result of this subsection:

Theorem 6.21. *Let σ be a positive nondisjunctive MKNF program. Then, the following claims hold for $M = \{I \mid I \models \mathsf{ob}_{T_\sigma^\omega}\}$:*

- *If $M \neq \emptyset$, then M is the single MKNF model of σ.*
- *If σ has an MKNF model, then this model is equal to M.*

Proof. (Claim 1.) Assume that $M \neq \emptyset$. T_σ^ω is a fixpoint of T_σ, so $M \models \sigma$ by Lemma 6.20. To show that M is an MKNF model of σ, we must show that it satisfies the preference semantics of MKNF. Assume that a set of interpretations M' exists such that $M' \supset M$ and $M' \models \sigma$. Then, M' induces a partition (P', N') of $\mathsf{KA}(\sigma)$. By Lemma 6.19, $T_\sigma(P') = P'$. Furthermore, $M' \supset M$ implies that, for each $\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)$, if $M' \models \mathbf{K}\,\xi$, then $M \models \mathbf{K}\,\xi$ as well, so $P' \subset P$, which now contradicts the assumption that $P = T_\sigma^\omega$ is the least fixpoint of T_σ. Hence, M is an MKNF model of σ, and it is unique by Theorem 6.16.

(Claim 2.) Assume that σ has an MKNF model M'. Then, M' induces a partition (P', N') of $\mathsf{KA}(\sigma)$, for which $T_\sigma(P') = P'$ by Lemma 6.19. We now show that P' is the least fixpoint of T_σ. Assume that there is some $P'' \subset P'$ such that $T_\sigma(P'') = P''$, and let $M'' = \{I \mid I \models \mathsf{ob}_{P''}\}$. By Lemma 6.20, $M'' \models \sigma$; moreover, $M'' \supset M'$, which now contradicts the assumption that M' is an MKNF model of σ. Hence, $P' = T_\sigma^\omega$. But then, $M' = M$ by Lemma 6.5. □

By Lemma 6.10, it is clear that $\sigma \models_{\mathsf{MKNF}} \psi$ for some modally closed MKNF formula ψ if and only if $\mathsf{ob}_{T_\sigma^\omega} \models \psi[T_\sigma^\omega]$. We now estimate the complexity of our algorithm.

Theorem 6.22. *Let σ be a positive nondisjunctive MKNF program. Assuming that the entailment of first-order formulae encountered while computing T_σ^ω is decidable in $\mathcal{C}$, the complexity of computing T_σ^ω is in $\mathrm{P}^{\mathcal{C}}$.*

Proof. The fixpoint of T_σ is reached after at most $|\mathsf{KA}(\sigma)|$ iterations: in the worst case, exactly one new atom is appended to the set of consequences in each iteration. Consider now the complexity of each iteration. $R_\sigma(S_{i-1})$ can be computed by checking, for each rule $r \in \sigma$, whether the body atoms of r are contained in S_{i-1}; obviously, this requires time polynomial in $|\mathcal{K}|$. Furthermore, computing $D_\sigma(S_{i-1})$ requires checking whether $\mathsf{ob}_{S_{i-1}} \models \xi$ for each $\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)$, and it can be performed by a linear number of calls to an oracle running in $\mathcal{C}$. Since the number of iterations is linear in $|\sigma|$, the algorithm runs in $\mathrm{P}^{\mathcal{C}}$. □

6.5 The Stratified Nondisjunctive Case

We now extend the results from the previous section to include a class of *stratified* nondisjunctive programs. Such programs are allowed to contain **not**-atoms; however, the rules of the program can be separated in strata, each of which can be evaluated separately. The following definition generalizes the notion of stratification of datalog programs to hybrid MKNF knowledge bases.

Definition 6.23. *Let σ be a nondisjunctive MKNF program and $\lambda : \sigma \to \mathbb{N}^+$ a function assigning to each rule $r \in \sigma$ a positive integer $\lambda(r)$. For an integer k and $\bowtie \in \{<, \leq, >, \geq\}$, let $\mathsf{head}(\sigma)^{\bowtie k} = \{\mathsf{head}(r) \mid r \in \sigma \text{ and } \lambda(r) \bowtie k\}$. We say that λ is a* stratification *of σ if the following conditions hold for each rule $r \in \sigma$:*

- *for each $\mathbf{K}\,\varphi \in \mathsf{body}^+(r)$, each $P \subseteq \mathsf{head}(\sigma)^{\leq \lambda(r)}$ such that $\mathsf{ob}_P \not\models \varphi$, and each $P' \subseteq \mathsf{head}(\sigma)^{> \lambda(r)}$, either $\mathsf{ob}_{P \cup P'} \not\models \varphi$ or $\mathsf{ob}_{P \cup P'}$ is unsatisfiable;*
- *for each $\mathbf{not}\,\varphi \in \mathsf{body}^-(r)$, each $P \subseteq \mathsf{head}(\sigma)^{< \lambda(r)}$ such that $\mathsf{ob}_P \not\models \varphi$, and each $P' \subseteq \mathsf{head}(\sigma)^{\geq \lambda(r)}$, either $\mathsf{ob}_{P \cup P'} \not\models \varphi$ or $\mathsf{ob}_{P \cup P'}$ is unsatisfiable.*

The program σ is stratified *if a stratification λ of σ exists. A stratification λ partitions σ into* strata *$\sigma_i = \{r \mid \lambda(r) = i\}$; the sequence of strata $\sigma_1, \ldots, \sigma_n$ is often identified with λ and is also called a stratification.*

The conditions of Definition 6.23 ensure that, when evaluating a stratum σ_i, the values of all **not**-atoms in σ_i have already been computed, and that evaluating any subsequent stratum will not change the values of any **K**- and **not**-atoms from any stratum σ_j with $j \leq i$. For ordinary datalog programs, a stratification is defined by the strongly connected components

of the dependency graph associated with the program. However, MKNF programs can contain arbitrary first-order formulae as atoms, which makes checking stratification more difficult. Consider the following program σ:

$$\mathbf{K}(p \vee q) \subset \mathbf{not}\, p \tag{46}$$

$$\mathbf{K}(\neg q) \subset \mathbf{K}(p \vee q) \tag{47}$$

The dependency graph of σ, build as usual by treating modal atoms as being "opaque," would suggest a stratification in which (46) comes before (47). By evaluating the rules in this order, we get the following result: the body of (46) is satisfied, so we derive $\mathbf{K}(p \vee q)$; this satisfies the body of (47), so we derive $\mathbf{K}(\neg q)$ as well. Thus, the objective knowledge is now $(p \vee q) \wedge \neg q$, which is equivalent to p. However, the body of (46) is not satisfied any more, so the model $M = \{I \mid I \models p\}$ is not minimal. In fact, Algorithm 1 shows that σ has no MKNF models.

Checking stratification can be difficult in general; however, it can be done using syntactic means for certain natural classes of nondisjunctive MKNF programs. For example, in case $\mathcal{DL}$ is propositional logic, σ is stratified if it is stratified in the usual sense (by treating modal atoms to be "opaque") and, furthermore, each propositional letter from the head of some rule from stratum i occurs in neither in the heads of rules from strata $j > i$ nor in **not**-atoms in the bodies of the rules from strata $j \geq i$. Also, stratification can be checked as usual when the MKNF program contains only non-DL-atoms in the heads (if $\mathcal{DL}$ allows for equality, then unique name assumption is additionally required). This is an important case because it allows defining constraints over DL knowledge bases.

We now show that a model for a stratified MKNF program σ can be computed by processing strata sequentially.

Definition 6.24. *For a stratification $\sigma_1, \ldots, \sigma_k$ of an MKNF program σ, the sequence of subsets $U_0, \ldots, U_k$ of $\mathsf{KA}(\sigma)$ is inductively defined as $U_0 = \emptyset$ and, for $0 < i \leq k$, $U_i = T^{\omega}_{\chi_i}$ where $\chi_i = U_{i-1} \cup \sigma'_i$ and σ'_i is obtained from σ_i by replacing each atom $\mathbf{not}\, \xi$ with its value in U_{i-1}. Finally, $U^{\omega}_{\sigma} = U_k$.*

We now show that the partition $(U^{\omega}_{\sigma}, \mathsf{KA}(\sigma) \setminus U^{\omega}_{\sigma})$ induces the MKNF model of a stratified program σ. Before doing so, we prove the following auxiliary lemma, which shows that a model of a positive nondisjunctive program can be represented only using the head atoms of the rules:

Lemma 6.25. *Let σ be a satisfiable positive nondisjunctive MKNF program. Furthermore, let M be an MKNF model of σ inducing a partition (P, N) of $\mathsf{KA}(\sigma)$. Finally, let P' be the subset of the modal atoms of P that occur in the head of a rule from σ. Then, M is equal to $M' = \{I \mid I \models \mathsf{ob}_{P'}\}$.*

Proof. Because $P' \subseteq P$, clearly $M \subseteq M'$. Assume now that $M \subset M'$ and consider a rule $r \in \sigma$; clearly, $M \models r$. If $M \models \mathbf{K}\, \xi$ for $\mathbf{K}\, \xi$ the head atom of

r, since $P' \subseteq P$, we have $M' \models \mathbf{K}\,\xi$ as well, so $M' \models r$. If the head atom of r is false in M, then there is a body atom $\mathbf{K}\,\xi$ of r such that $M \not\models \mathbf{K}\,\xi$, but then, $M' \not\models \mathbf{K}\,\xi$ as well, so $M' \models r$. Thus, we get $M' \models \sigma$, which contradicts the assumption that M is an MKNF model of σ. □

Theorem 6.26. *Let σ be a stratified nondisjunctive MKNF program and U^ω_σ be computed as specified in Definition 6.24 using any stratification. Then, the following claims hold for $M = \{I \mid I \models \mathsf{ob}_{U^\omega_\sigma}\}$:*

- *If $M \neq \emptyset$, then M is an MKNF model of σ.*
- *If σ has an MKNF model, then this model is equal to M.*

Proof. Let $\sigma_1, \ldots, \sigma_k$ be the stratification of σ used to compute U^ω_σ, χ_i as in Definition 6.24, $\zeta_i = \bigcup_{j \leq i} \sigma_j$, and $M_i = \{I \mid I \models \mathsf{ob}_{U_i}\}$ for $0 \leq i \leq k$. Each χ_i is a positive nondisjunctive program so, by Theorem 6.16, it has at most one MKNF model that corresponds to M_i by Theorem 6.21.

Assume that $M_i \neq \emptyset$ and consider an atom $\mathbf{K}\,\varphi$ occurring in the body of a rule from ζ_{i-1}. Clearly, $M_{i-1} \models \mathbf{K}\,\varphi$ implies $M_i \models \mathbf{K}\,\varphi$. Assume now that $M_{i-1} \not\models \mathbf{K}\,\varphi$ and $M_i \models \mathbf{K}\,\varphi$. Let U'_{i-1} be the subset of U_{i-1} containing only the atoms occurring in the head of some rule in χ_i; by Lemma 6.25, $M_{i-1} = \{I \mid I \models \mathsf{ob}_{U'_{i-1}}\}$, so $\mathsf{ob}_{U'_{i-1}} \not\models \varphi$. Let (P, N) be the partition of $\mathsf{KA}(\sigma)$ induced by M_i, and let P' be the subset of P containing those atoms that occur in a head of a rule from σ; by Lemma 6.25, $M_i = \{I \mid I \models \mathsf{ob}_{P'}\}$, so $\mathsf{ob}_{P'} \models \varphi$. Since $M_{i-1} \subseteq M_i$, we have $U'_{i-1} \subseteq P'$, so σ is not stratified, which is a contradiction. Hence, the following property (*) holds: the value of all **K**-atoms and **not**-atoms occurring in a rule from χ_{i-1} is the same in M_{i-1} and M_i if $M_i \neq \emptyset$. In a similar way, one can show that the following property (**) holds as well: the value of all **not**-atoms occurring in a rule in σ_i is the same in M_{i-1} and M_i if $M_i \neq \emptyset$. Finally, the following property (***) is trivial: for $\mathbf{K}\,\varphi$ an atom occurring in the head of a rule from χ_{i-1}, $M_i \not\models \mathbf{K}\,\varphi$ implies $M_{i-1} \not\models \mathbf{K}\,\varphi$.

We now prove the two claims of this theorem inductively for each ζ_i and its corresponding M_i. The induction basis for $i = 0$ is trivial, so we consider the inductive step.

(Claim 1.) Assume that $M_i \neq \emptyset$. Clearly, $M_{i-1} \neq \emptyset$ as well so, by induction assumption, M_{i-1} is an MKNF model of ζ_{i-1}. By the properties (*) and (***), $M_i \models \zeta_{i-1}$. Furthermore, by the property (**), $M_i \models \sigma_i$ if and only if $M_i \models \sigma'_i$. Hence, $M_i \models \zeta_i$. Assume now that an MKNF interpretation M''_i exists such that $M''_i \models \zeta_i$ and $M''_i \supset M_i$. But then, $M''_i \models \chi_i$, which contradicts the fact that M_i is an MKNF model of χ_i. Hence, M_i is an MKNF model of ζ_i.

(Claim 2.) Assume that ζ_i has an MKNF model M''_i. By the property (**), M''_i is an MKNF model of χ_i, but then the partition of $\mathsf{KA}(\sigma)$ induced by M''_i is $(U_i, \mathsf{KA}(\sigma) \setminus U_i)$. Hence, $M''_i = M_i$. □

By Lemma 6.10, it is clear that $\sigma \models_{\mathsf{MKNF}} \psi$ for some modally closed MKNF formula ψ if and only if $\mathsf{ob}_{U^\omega_\sigma} \models \psi[U^\omega_\sigma]$. We now estimate the complexity of our algorithm.

Theorem 6.27. *Let σ be a stratified nondisjunctive MKNF program. Assuming that the entailment of first-order formulae encountered while computing $T^\omega_{\sigma_i}$ for each stratum σ_i is decidable in $\mathcal{C}$, the complexity of computing U^ω_σ is in $\mathrm{P}^{\mathcal{C}}$.*

Proof. For each stratum σ_i, the set $T^\omega_{\sigma_i}$ can be computed in $\mathrm{P}^{\mathcal{C}}$ by Theorem 6.22. The formula σ'_i can be computed by a linear number of calls to the oracle running in $\mathcal{C}$. Finally, the number of strata of σ is linear in $|\sigma|$, so U^ω_σ can be computed with complexity $\mathrm{P}^{\mathcal{C}}$. □

The following lemma is useful for estimating complexity of entailment of negative facts from stratified MKNF programs:

Lemma 6.28. *Let σ be a stratified MKNF program and ψ a closed first-order formula. Then, $\sigma \models_{\mathsf{MKNF}} \neg \mathbf{K}\,\psi$ if and only if σ is MKNF unsatisfiable or $\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$.*

Proof. Both directions are trivial if σ is MKNF unsatisfiable, so we assume that σ has a (unique) MKNF model M. For the ($\Rightarrow$) direction, if $\sigma \models_{\mathsf{MKNF}} \neg K\psi$, then $M \models \neg\,\mathbf{K}\,\psi$, so $M \not\models \mathbf{K}\,\psi$ and $\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$. For the ($\Leftarrow$) direction, $\sigma \not\models_{\mathsf{MKNF}} \mathbf{K}\,\psi$ implies $M \not\models \mathbf{K}\,\psi$, which implies $M \models \neg\,\mathbf{K}\,\psi$; since M is the only model of σ, we have $\sigma \models_{\mathsf{MKNF}} \neg\,\mathbf{K}\,\psi$. □

6.6 The Nonstratified Nondisjunctive Case

We now consider the case when σ is a nondisjunctive program for which a stratification cannot be found. For such programs, one still needs to guess the partition (P, N) to determine the objective knowledge; however, $\sigma[\mathbf{not}, P, N]$ is a positive nondisjunctive MKNF program, which allows to check the minimality condition as explained in Section 6.4, possibly leading to lower complexity. This idea is reflected in Algorithm 3.

Theorem 6.29. *For σ a nonstratified nondisjunctive MKNF program and ψ a modally closed MKNF formula,* $\mathsf{nondisjunctive\text{-}not\text{-}entails}(\sigma, \psi)$ *returns* true *if and only if $\sigma \models_{\mathsf{MKNF}} \psi$. Assuming that the satisfiability of first-order formulae in Algorithm 3 is decidable in $\mathcal{C}$, the complexity of the algorithm* $\mathsf{nondisjunctive\text{-}not\text{-}entails}(\sigma, \psi)$ *is in $\mathcal{E}^{P^{\mathcal{C}}}$, where $\mathcal{E} = \mathrm{NP}$ if $\mathcal{E} \subseteq \mathrm{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise.*

Proof. (Claim 1.) From the proof of Theorem 6.11, one can see that Condition (4) of $\mathsf{not\text{-}entails}(\sigma, \psi)$ ensures that a model induced by a partition (P, N) of $\mathsf{KA}(\sigma)$ satisfies the preference semantics of MKNF. Since σ is a flat formula, this is equivalent to checking whether $M = \{I \mid I \models \mathsf{ob}_P\}$ is

Algorithm 3 Checking Entailment in Nondisjunctive MKNF Programs

Algorithm: nondisjunctive-not-entails(σ, ψ)
Input:
σ: a nonstratified nondisjunctive modally closed MKNF formula
ψ: a modally closed MKNF formula (not necessarily flat)
Output:
true if $\sigma \not\models_{\mathsf{MKNF}} \psi$; false otherwise

if a partition (P, N) of $\mathsf{KA}(\sigma)$ exists such that
1. $\sigma[P, N]$ evaluates to true, and
2. ob_P is satisfiable, and
3. $\neg\xi \wedge \mathsf{ob}_P$ is satisfiable for each $\mathbf{K}\,\xi \in N$, and
4. $T^{\omega}_{\sigma'} = P$ for $\sigma' = \sigma[\mathbf{not}, P, N]$ and
5. $\mathsf{ob}_P \wedge \neg\psi[P]$ is satisfiable

then return true; otherwise **return** false

an MKNF model of $\sigma' = \sigma[\mathbf{not}, P, N]$. Since σ' is a positive nondisjunctive program, its only model corresponds to $T^{\omega}_{\sigma'}$ by Theorem 6.21. Thus, (P, N) satisfies the preference semantics of MKNF if and only if $T^{\omega}_{\sigma'} = P$.

(Claim 2.) A partition (P, N) can be guessed in time polynomial in $|\sigma|$, Condition (1) can be checked in polynomial time, Conditions (2) and (3) can be verified by a polynomial number of calls to an oracle running in $\mathcal{C}$, $\psi[P]$ can be computed by a polynomial number of calls to an oracle running in $\mathcal{C}$, and satisfiability of $\mathsf{ob}_P \wedge \neg\psi[P]$ can be computed by an additional call to the oracle. By Theorem 6.22, Condition (4) can be checked in $P^{\mathcal{C}}$, so the whole algorithm runs in $\mathcal{E}^{P^{\mathcal{C}}}$. □

7 Reasoning with Hybrid Knowledge Bases

In this section, we apply the techniques from Section 4 to obtain algorithms for reasoning with a hybrid MKNF knowledge base $\mathcal{K}$. In particular, we show how to check various conditions of the algorithms from Section 4 using standard DL inferences of knowledge base satisfiability and knowledge base entailment. Thus, the results from this section show how to build a reasoner for hybrid MKNF knowledge bases on top of any DL reasoner.

7.1 The General Case

Let $\mathcal{K}$ be a DL-safe flat hybrid MKNF knowledge base and ψ a formula of the form $(\neg)\,\mathbf{K}\,A$ with A a ground atom, for which we want to check whether $\mathcal{K} \models \psi$. The formula $\pi(\mathcal{K})$ need not be modally closed, so we cannot apply the algorithm not-entails$(\pi(\mathcal{K}), \psi)$ directly. Therefore, we first compute $\mathcal{K}_G$—the ground instantiation of $\mathcal{K}$—which, since $\mathcal{K}$ is DL-safe, entails the

same set of ground MKNF formulae as $\mathcal{K}$ by Lemma 4.6. The size of $\mathcal{K}_G$ can be exponentially larger than the size of $\mathcal{K}$. However, this is already the case for datalog [6] and disjunctive datalog under stable model semantics [11], both of which are included in our logic: checking satisfiability of non-ground (disjunctive) programs more complex than for ground (disjunctive) programs by an exponential factor. Therefore, in the following sections, we do not consider combined complexity, but focus on *data complexity*: for a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, data complexity is measured in the size of the ABox of $\mathcal{O}$ and the number of facts in $\mathcal{P}$.

For $\sigma = \pi(\mathcal{K}_G)$, we check whether $\sigma \models_{\mathsf{MKNF}} \psi$ using $\mathsf{not\text{-}entails}(\sigma, \psi)$. To satisfy Condition (1) of the algorithm, any partition (P, N) of $\mathsf{KA}(\sigma)$ must be such that $\mathbf{K}\,\pi(\mathcal{O}) \in P$; similarly, to falsify Condition (a), any partition (P', N') must be such that $\mathbf{K}\,\pi(\mathcal{O}) \in P'$. Hence, we can set $\mathbf{K}\,\pi(\mathcal{O})$ to be true in advance and consider only partitions (P, N) of the remaining modal atoms; we denote this set with $\mathsf{KA}(\mathcal{K}) = \mathsf{KA}(\sigma) \setminus \{\mathbf{K}\,\pi(\mathcal{O})\}$. Clearly, $\mathsf{KA}(\mathcal{K})$ contains only ground $\mathbf{K}$-atoms. The objective knowledge of a partition (P, N) of $\mathsf{KA}(\mathcal{K})$ is then given by the following formula, where ξ are positive ground atoms:

$$\mathsf{ob}_P = \pi(\mathcal{O}) \wedge \bigwedge_{\mathbf{K}\,\xi \in P} \xi$$

To simplify the notation, in the following presentation we identify $\mathcal{P}_G$ with $\pi(\mathcal{P}_G)$; hence, by saying "$\mathcal{P}_G[P, N]$ evaluates to true" we actually mean "$\pi(\mathcal{P}_G)[P, N]$ evaluates to true."

For a set of $\mathbf{K}$-atoms S, let $\widehat{S} = \{\xi \mid \mathbf{K}\,\xi \in S\}$, and let S_{DL} and $\widehat{S}_{DL}$ denote the subset of the DL-atoms from S and $\widehat{S}$, respectively. Assuming that $\mathcal{DL}$ supports ABoxes (which is the case for most practically relevant DLs), $\widehat{P}_{DL}$ can be considered a DL ABox. All atoms from $\widehat{P}$ are positive, so $\mathcal{DL}$ is not required to support negative ABox assertions. Hence, ob_P can be rewritten as follows:

$$\mathsf{ob}_P = \pi(\mathcal{O} \cup \widehat{P}_{DL}) \wedge \bigwedge_{\mathbf{K}\,\xi \in P \setminus P_{DL}} \xi$$

Since $\bigwedge_{\mathbf{K}\,\xi \in P \setminus P_{DL}} \xi$ is a conjunction of positive non-DL-atoms, it cannot affect the satisfiability of ob_P or the entailment of DL-atoms from it. However, $\pi(\mathcal{O} \cup \widehat{P}_{DL})$ can affect the entailment of non-DL-atoms from ob_P: for example, if $\mathcal{O} \models a \approx b$, then a non-DL-atom $Q(b)$ is actually a synonym for the non-DL-atom $Q(a)$, so a partition (P, N) such that $\mathbf{K}\,Q(a) \in P$ and $\mathbf{K}\,Q(b) \in N$ is inconsistent. Therefore, to ensure consistency of (P, N), we check whether, for each $\mathbf{K}\,Q(a_1, \ldots, a_n) \in N \setminus N_{DL}$ and $\mathbf{K}\,Q(b_1, \ldots, b_n) \in P \setminus P_{DL}$, for some $1 \leq i \leq n$ we have $\mathcal{O} \cup \widehat{P}_{DL} \not\models a_i \approx b_i$; if this is so, then $Q(a_1, \ldots, a_n)$ and $Q(b_1, \ldots, b_n)$ are not synonyms, so the partition (P, N) is consistent.

The procedure $\mathsf{not\text{-}entails\text{-}DL}(\mathcal{K}, \psi)$ is presented in Algorithm 4.

Algorithm 4 Reasoning with General Hybrid Knowledge Bases

Algorithm: not-entails-DL$(\mathcal{K}, \psi)$
Input:
$\mathcal{K} = (\mathcal{O}, \mathcal{P})$: a DL-safe flat hybrid MKNF knowledge base
ψ : a formula of the form $(\neg)\,\mathbf{K}\,A$ where A is a ground atom
Output:
true if $\mathcal{K} \not\models \psi$; false otherwise

let $\mathcal{K}_G$ be the ground instantiation of $\mathcal{K}$ w.r.t. $O_{\mathcal{K}}$
if a partition (P, N) of $\mathsf{KA}(\mathcal{K}_G) \cup \{\mathbf{K}\,A\}$ exists such that

1. $\mathcal{P}_G[P, N]$ evaluates to true, and
2. $\mathcal{O} \cup \widehat{P}_{DL}$ is satisfiable, and
3. $\mathcal{O} \cup \widehat{P}_{DL} \not\models \xi$ for each $\mathbf{K}\,\xi \in N_{DL}$, and
4. **for each** $\mathbf{K}\,Q(a_1, \ldots, a_n) \in N \setminus N_{DL}$ and $\mathbf{K}\,Q(b_1, \ldots, b_n) \in P \setminus P_{DL}$, we have $\mathcal{O} \cup \widehat{P}_{DL} \not\models a_i \approx b_i$ for some $1 \leq i \leq n$
5. **for** $\gamma = \mathcal{P}_G[\mathbf{not}, P, N]$ and each partition (P', N') of P such that $N' \neq \emptyset$
   (a) $\gamma[P', N \cup N']$ evaluates to false, or
   (b) $\mathcal{O} \cup \widehat{P'}_{DL}$ is unsatisfiable, or
   (c) $\mathcal{O} \cup \widehat{P'}_{DL} \models \xi$ for some $\mathbf{K}\,\xi \in N'_{DL}$, or
   (d) **for some** $\mathbf{K}\,Q(a_1, \ldots, a_n) \in N' \setminus N'_{DL}$ and $\mathbf{K}\,Q(b_1, \ldots, b_n) \in P' \setminus P'_{DL}$, we have $\mathcal{O} \cup \widehat{P'}_{DL} \models a_i \approx b_i$ for all $1 \leq i \leq n$

   and
6. one of the following conditions holds:
   (i) $\psi = \mathbf{K}\,A$ and $\mathbf{K}\,A \notin P$, or
   (ii) $\psi = \neg\,\mathbf{K}\,A$ and $\mathbf{K}\,A \in P$

then return true; otherwise **return** false

Theorem 7.1. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe flat hybrid MKNF knowledge base and $\psi = (\neg)\,\mathbf{K}\,A$ for A a ground atom. Then, the algorithm* not-entails-DL$(\mathcal{K}, \psi)$ *returns* true *if and only if $\mathcal{K} \not\models \psi$. Furthermore, assuming that the entailment of ground DL-atoms in $\mathcal{DL}$ is decidable with data complexity $\mathcal{C}$, the data complexity of the algorithm is in $\mathcal{E}^{\mathcal{E}}$, where $\mathcal{E} = \text{NP}$ if $\mathcal{E} \subseteq \text{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise.*

Proof. (Claim 1.) By Lemma 4.5, $\mathcal{K} \not\models \psi$ if and only if $\mathcal{K}_G \not\models \psi$. From the discussion from this section, it is clear that Conditions (1), (2), (3)–(4), (5), and (6) of not-entails-DL$(\mathcal{K}, \psi)$ correspond to Conditions (1), (2), (3), (4) and (5) of not-entails$(\pi(\mathcal{K}_G), \psi)$, respectively. Note that the partition (P, N) determines the value of $\mathbf{K}\,A$ because it is a partition of $\mathsf{KA}(\mathcal{K}) \cup \{\mathbf{K}\,A\}$; hence, $\mathsf{ob}_P \models A$ if and only if $\mathbf{K}\,A \in P$. The claim of this theorem now follows from Theorem 6.11.

(Claim 2.) If the size of nonground rules in $\mathcal{P}_G$ is bounded, $\mathcal{K}_G$ can be computed in polynomial time. Now the proof of the second claim of this theorem is completely analogous to the proof of Theorem 6.12. □

7.2 The Positive Case

As discussed in Section 6.3, entailment in MKNF coincides with entailment in S5 for positive formulae, which allows the reasoning algorithm to find any S5 model, and not necessarily a minimal one. Hence, for a positive hybrid MKNF knowledge base $\mathcal{K}$, checking whether $\mathcal{K} \not\models \mathbf{K}\,A$ can be performed by an algorithm not-entails-DL$^+(\mathcal{K}, \mathbf{K}\,A)$, which is the same as the algorithm not-entails-DL(σ, ψ) without Condition (5).

Theorem 7.2. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a positive DL-safe flat hybrid MKNF knowledge base and A a ground atom. Then,* not-entails-DL$^+(\mathcal{K}, \mathbf{K}\,A)$ *returns* true *if and only if $\mathcal{K} \not\models \mathbf{K}\,A$. Furthermore, assuming that the entailment of ground DL-atoms in $\mathcal{DL}$ is decidable with data complexity $\mathcal{C}$, the data complexity of the algorithm is in $\mathcal{E}$, where $\mathcal{E} = \text{NP}$ if $\mathcal{E} \subseteq \text{NP}$, and $\mathcal{E} = \mathcal{C}$ otherwise.*

Proof. Both claims follow from Theorems 6.13 and 7.1. □

7.3 The Positive Nondisjunctive Case

For the case when $\mathcal{K}$ is a positive nondisjunctive DL-safe hybrid MKNF knowledge base, we can apply the ideas from the previous section to the algorithm for positive nondisjunctive MKNF programs from Section 6.4.

Definition 7.3. *For $\mathcal{K}$ a positive nondisjunctive DL-safe hybrid MKNF knowledge base, let $R_{\mathcal{K}}$, $D_{\mathcal{K}}$, and $T_{\mathcal{K}}$ be the operators defined on the subsets of $\mathsf{KA}(\mathcal{K})$ as follows:*

$$R_{\mathcal{K}}(S) = S \cup \{\mathbf{K}\, H \mid \mathcal{K}_G \text{ contains a rule of the form (1) such that } \mathbf{K}\, B_i \in S \text{ for each } 1 \leq i \leq n\}$$

$$D_{\mathcal{K}}(S) = \{\mathbf{K}\, \xi \mid \mathbf{K}\, \xi \in \mathsf{KA}(\mathcal{K}) \text{ and } \mathcal{O} \cup \widehat{S}_{DL} \models \xi\} \cup \{\mathbf{K}\, Q(b_1, \ldots, b_n) \mid \mathbf{K}\, Q(a_1, \ldots, a_n) \in S \setminus S_{DL} \text{ and } \mathcal{O} \cup \widehat{S}_{DL} \models s_i \approx b_i \text{ for } 1 \leq i \leq n\}$$

$$T_{\mathcal{K}}(S) = R_{\mathcal{K}}(S) \cup D_{\mathcal{K}}(S)$$

With $T_{\mathcal{K}}^{\omega}$ we denote the least fixpoint of $T_{\mathcal{K}}$ on $\mathsf{KA}(\mathcal{K})$.

An analogous algorithm for ordinary datalog does not require grounding the rules before applying the algorithm. The same holds for the rules component of a nondisjunctive hybrid MKNF knowledge base $\mathcal{K}$—the value of $R_{\mathcal{K}}(S)$ could be computed using nonground rules exactly as this done for ordinary datalog. Unfortunately, computing $D_{\mathcal{K}}(S)$ involves examining all atoms from $\mathsf{KA}(\mathcal{K})$, which requires at least computing all possible instantiations of atoms from $\mathcal{P}$. Because grounding seems inevitable, we do not formulate $R_{\mathcal{K}}(S)$ in a more general way to keep the presentation simple.

It is easy to see that $T_{\mathcal{K}}(S)$ corresponds to $T_{\sigma}(S)$ for $\sigma = \pi(\mathcal{K}_G)$, which leads us to the following theorem:

Theorem 7.4. *Let $\mathcal{K}$ be a positive nondisjunctive DL-safe hybrid MKNF knowledge base. Then, the following claims hold for $M = \{I \mid I \models \mathcal{O} \cup \widehat{T_{\mathcal{K}}^{\omega}}\}$:*

- *If $M \neq \emptyset$, then M is the single MKNF model of $\mathcal{K}$.*
- *If $\mathcal{K}$ has an MKNF model, then this model is equal to M.*

Assuming that the entailment of ground DL-atoms in $\mathcal{DL}$ is decidable with data complexity $\mathcal{C}$, the data complexity of computing $T_{\mathcal{K}}^{\omega}$ is in $\mathrm{P}^{\mathcal{C}}$.

Proof. Follows immediately from Theorems 6.21 and 6.22. □

Clearly, for A a ground atom, $\mathcal{K} \models \mathbf{K}\, A$ if and only if $\mathcal{O} \cup \widehat{T_{\mathcal{K}}^{\omega}} \models A$, and $\mathcal{K} \models \neg\, \mathbf{K}\, A$ if and only if $\mathcal{O} \cup \widehat{T_{\mathcal{K}}^{\omega}} \not\models A$.

7.4 The Stratified and the Nonstratified Cases

Let $\mathcal{K}$ be a not necessarily positive nondisjunctive DL-safe hybrid MKNF knowledge base. In general, checking whether $\mathcal{K}$ is stratified may be quite complicated. As explained in Section 6.5, in stratified programs, deriving a $\mathbf{K}$-atom by a rule in a stratum i should not affect the values of modal atoms in strata below i. Given a general DL knowledge base $\mathcal{O}$, this might be quite difficult. Furthermore, if $\mathcal{DL}$ allows for equality (e.g., by allowing

number restrictions), such an analysis cannot be performed just by means of concept and role names from $\mathcal{O}$: it is always possible that an atom derived by a rule in some stratum makes $\mathcal{O}$ derive new equalities that change the value of the atoms from lower strata.

If $\mathcal{K}$ employs the unique name assumption or $\mathcal{DL}$ does not employ equality, and the rules from $\mathcal{P}$ do not contain DL-predicates in the head, then the rules of $\mathcal{K}$ are "layered on top" of $\mathcal{O}$, so their stratification can be checked as usual [1]. This case is particularly important because it allows to define constraints over DL knowledge bases.

An algorithm for computing the set of **K**-atoms following from a stratified nondisjunctive DL-safe hybrid MKNF knowledge base $\mathcal{K}$ can be defined in a straightforward way analogously to Definition 6.24, whose correctness and complexity can be proved as in Theorem 6.26 and Lemma 6.27.

Similarly, an algorithm for computing the set of **K**-atoms following from a nonstratified nondisjunctive DL-safe hybrid MKNF knowledge base $\mathcal{K}$ can be defined in a straightforward way analogously to Algorithm 3, whose correctness and complexity can be proved as in Theorem 6.29.

8 Data Complexity

We now investigate the data complexity of checking entailment of ground literals for hybrid MKNF knowledge bases—that is, the complexity measured in the size of the ABox of the DL knowledge base and the number of facts in the MKNF program. To present a precise characterization, we must make assumptions about the data complexity of checking entailment of ground literals in the underlying fragment $\mathcal{DL}$. In [20], it was shown that checking entailment of ground atoms in many very expressive DLs, such as $\mathcal{SHIQ}$, is data complete for coNP; furthermore, there are expressive fragments, such as Horn-$\mathcal{SHIQ}$ [20] or DL-lite [5], which are data complete for P. Therefore, we analyze the complexity of MKNF knowledge bases for these two cases. To estimate the impact of adding a DL knowledge base to logic programs, we also contrast these results with the well-known results for logic programs without a DL knowledge base.

Table 3 summarizes the results for complexity of checking $\mathcal{K} \models \psi$, for $\psi = (\neg)\,\mathbf{K}\,A$ with A a ground atom. In some cases, the complexity differs depending on whether $\psi = \mathbf{K}\,A$ or $\psi = \neg\,\mathbf{K}\,A$, so we then present both results. All results are completeness results. In the following sections we prove the results from the table.

8.1 Positive Nondisjunctive Programs

For $\mathcal{DL} = \emptyset$ and $\mathcal{DL} \in$ P, the polynomial lower bound follows from P-hardness of ordinary datalog [6], and the upper bound follows from Theorem 7.4 and the fact that $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$.

Table 3: Data Complexity of Entailment Checking for Hybrid MKNF KBs

	$\vee$	**not**	$\mathcal{DL} = \emptyset$	$\mathcal{DL} \in$ P	$\mathcal{DL} \in$ coNP
1	no	no	P	P	coNP
2	no	stratified	P	P	Δ_2^p
3	no	yes	coNP	coNP	Π_2^p
4	yes	no	coNP/Π_2^p	coNP/Π_2^p	coNP/Π_2^p
5	yes	yes	Π_2^p	Π_2^p	Π_2^p

Consider now $\mathcal{DL} \in$ coNP. The coNP lower bound of entailment is inherited from entailment in $\mathcal{DL}$. For a positive ground atom $\mathbf{K}\,A$, the coNP upper bound follows from Theorem 7.2. Finally, by Lemma 6.28, $\mathcal{K} \models \neg\,\mathbf{K}\,A$ if either $\mathcal{K}$ is unsatisfiable or $\mathcal{K} \not\models \mathbf{K}\,A$; by Theorem 7.2, both checks can be performed in coNP.

8.2 Stratified Nondisjunctive Programs

For $\mathcal{DL} = \emptyset$ and $\mathcal{DL} \in$ P, the polynomial lower bound is inherited from the case of stratified datalog, and the upper bound follows from Theorem 6.27 and the fact that $\mathrm{P}^{\mathrm{P}} = \mathrm{P}$.

Consider now $\mathcal{DL} \in$ coNP. The coNP upper bound is an immediate consequence of Theorem 6.27, and for the lower bound, we give a hardness proof next. We consider only the case of positive atoms because, by Lemma 6.28, entailment of negative atoms can be reduced to checking entailment of positive atoms. Furthermore, it is sufficient to consider $\mathcal{DL}$ to be the logic of disjunctive datalog rules interpreted under first-order semantics: entailment in such a logic is data complete for coNP, so any other logic $\mathcal{DL}'$ in which entailment is data complete for coNP can be polynomially reduced to $\mathcal{DL}$.

Theorem 8.1. *Let $\mathcal{DL}$ be the logic of disjunctive datalog interpreted under first-order semantics, $\mathcal{K}$ a stratified nondisjunctive hybrid MKNF knowledge base, and A a ground atom. Then, the problem of checking whether $\mathcal{K} \models \mathbf{K}\,A$ is Δ_2^p-hard w.r.t. data complexity.*

Proof. The proof is by a reduction from DAGS(SAT) [17]. An instance of DAGS(SAT) is a triple $D = \langle \mathit{Var}, G, \varphi_R \rangle$ with the following properties:

- $\mathit{LVar} = \{v_1, \ldots, v_n\}$ is the set of *linking variables*.
- $G = \langle V, E \rangle$ is a direct acyclic graph in which the vertices are propositional formulae—that is, $V = \{\varphi_1, \ldots, \varphi_n\}$. Each φ_i is associated with a unique linking variable v_i. Furthermore, φ_i should contain the linking variable v_j corresponding to each φ_j such that $(\varphi_j, \varphi_i) \in E$; also, φ_i can contain *private* variables not appearing in any other formula.

- $\varphi_R \in V$ is a distinguished *result node*.

Given a DAGS(SAT) instance D, a valuation $\nu : LVar \to \{\mathsf{true}, \mathsf{false}\}$ is defined inductively as follows: $\nu(v_i) = \mathsf{true}$ if and only if the propositional formula φ_i', obtained by replacing in φ the linking variables with their values under ν, is satisfiable (since G is a direct acyclic graph, this induction is correctly defined). Now DAGS(SAT) is the problem of deciding whether $\nu(\varphi_R) = \mathsf{true}$, and it is known to be Δ_2^p-complete [17].

Without loss of generality, we can assume that each formula φ_i is of the form

$$\varphi_i = \bigwedge_j \omega_{i,j} \tag{48}$$

where $\omega_{i,j}$ has one of the following forms:

$$u \equiv u_1 \wedge u_2 \tag{49}$$

$$u \equiv u_1 \vee u_2 \tag{50}$$

$$u \equiv \neg u_1 \tag{51}$$

Namely, a general propositional formula φ_i can be brought into the form (48) by iteratively replacing each nonatomic subformula of φ_i with a fresh propositional variable, and by introducing an explicit definition of the form (49), (50), or (51) for that variable. For each formula φ_i, with u_{φ_i} we denote a private variable corresponding to the translation of φ_i into the form (48) and with v_i we denote the linking variable corresponding to φ_i. Thus, v_R is the linking variable that corresponds to φ_R. Furthermore, with $PVar$ we denote the set of all private variables used in D.

For an instance D of DAGS(SAT), we construct a hybrid MKNF knowledge base $\mathcal{K}_D = (\mathcal{O}_D, \mathcal{P}_D)$ as follows. We add to $\mathcal{O}_D$ the following axioms:

$$T(x) \vee F(x) \leftarrow PVar(x) \tag{52}$$

$$\leftarrow T(x) \wedge F(x) \tag{53}$$

$$T(x) \leftarrow and(x, y, z), T(y), T(z) \tag{54}$$

$$T(y) \leftarrow and(x, y, z), T(x) \tag{55}$$

$$T(z) \leftarrow and(x, y, z), T(x) \tag{56}$$

$$T(x) \leftarrow or(x, y, z), T(y) \tag{57}$$

$$T(x) \leftarrow or(x, y, z), T(z) \tag{58}$$

$$T(y) \vee T(z) \leftarrow or(x, y, z), T(x) \tag{59}$$

$$F(y) \leftarrow not(x, y), T(x) \tag{60}$$

$$T(x) \leftarrow not(x, y), F(y) \tag{61}$$

For each $u \in PVar$, we add to $\mathcal{O}_D$ the following fact:

$$PVar(u) \tag{62}$$

For each $\omega_{i,j}$ occurring in a formula φ_i, we include into $\mathcal{O}_D$ a fact (63) if $\omega_{i,j}$ is of the form (49), a fact (64) if $\omega_{i,j}$ is of the form (50), and a fact (65) if $\omega_{i,j}$ is of the form (51):

$$and(u, u_1, u_2) \tag{63}$$

$$or(u, u_1, u_2) \tag{64}$$

$$not(u, u_1) \tag{65}$$

For each $\varphi_i \in V$, we add to $\mathcal{P}_D$ a fact of the following form:

$$\mathbf{K}\, formula(u_{\varphi_i}, v_i) \tag{66}$$

Finally, we add to $\mathcal{P}_D$ the following rules:

$$\mathbf{K}\, T(y) \leftarrow \mathbf{K}\, formula(x, y), \mathbf{not}\, F(x) \tag{67}$$

$$\mathbf{K}\, F(y) \leftarrow \mathbf{K}\, formula(x, y), \mathbf{K}\, F(x) \tag{68}$$

Clearly, $|\mathcal{K}_D|$ is linear in $|D|$. Furthermore, the nonground rules in $\mathcal{K}_D$ are fixed for any D, so the size of the ground instantiation $\mathcal{K}_{DG}$ of $\mathcal{K}_D$ is polynomial in $|D|$; since $\mathcal{K}_D$ is DL-safe, $\mathcal{K}_{DG} \models \mathbf{K}\, T(v_R)$ if and only if $\mathcal{K}_D \models \mathbf{K}\, T(v_R)$ by Lemma 4.6. In computing the ground instantiation of $\mathcal{K}_D$, it suffices to instantiate the rules (67) and (68) only for those values $x = u_{\varphi_i}$ and $y = v_i$ for which a fact (66) occurs in $\mathcal{P}_D$: all other instantiations are clearly true in any model of $\mathcal{K}_D$. We denote such an instantiation of $\mathcal{K}_D$ with $\mathcal{K}'_D$; clearly, $\mathcal{K}'_D \models \mathbf{K}\, T(v_R)$ if and only if $\mathcal{K}_D \models \mathbf{K}\, T(v_R)$.

Consider now any first-order interpretation I. The axioms (52)–(53) and (62) ensure that, for each $u \in PVar$, either $I \models T(u)$ or $I \models F(u)$, but not both. Intuitively, $I \models T(u)$ means that u is assigned the value true, and $I \models T(u)$ means that u is assigned the value false. The axioms (54)–(61) ensure that the truth values for formulae of the form (49)–(51) are propagated in I according to the standard semantics of the propositional connectives.

Let M_D be an MKNF model containing all first-order interpretations I satisfying (52)–(62) and (66) such that $I \models T(v_i)$ if $\nu(v_i) = \mathsf{true}$ and $I \models F(v_i)$ if $\nu(v_i) = \mathsf{false}$. We now show that M_D is an MKNF model of $\mathcal{K}'_D$. Since D is acyclic, there is a sequence of formulae $\varphi_0, \varphi_1, \ldots, \varphi_n$ such that the linking variables in each formula φ_i correspond to formulae φ_j that precede φ_i in the sequence. Consider now the formula φ_0 without any linking variables.

- If $\nu(v_0) = \mathsf{true}$, then φ_0 is satisfiable. M_D contains a first-order interpretation I for each truth assignment of private variables of φ_0, so $I \not\models F(u_{\varphi_0})$ for some $I \in M_D$, and $M_D \models \mathbf{not}\, F(u_{\varphi_0})$. But then, the rule (67) instantiated for $x = u_{\varphi_0}$ and $y = v_0$ is clearly satisfied, since $M_D \models \mathbf{K}\, T(v_0)$.

- If $\nu(v_0) = \mathsf{false}$, then φ_0 is unsatisfiable. M_D contains a first-order interpretation I for each truth assignment of private variables of φ_0, so $I \models F(u_{\varphi_0}$ for each $I \in M_D$, and $M_D \models \mathbf{K}\, F(u_{\varphi_0})$. But then, the rule (68) instantiated for $x = u_{\varphi_0}$ and $y = v_0$ is clearly satisfied, since $M_D \models \mathbf{K}\, F(v_0)$.

By inductively considering all remaining formulae φ_i in the sequence, we can see that M_D satisfies all the rules from $\mathcal{P}'_D$ (and it satisfies $\mathcal{O}'_D$ by the assumption). To show that M_D is an MKNF model of $\mathcal{K}'_D$, consider any $M'_D \supset M_D$; now such a model would necessarily invalidate either $\mathcal{O}'_D$, some fact (66), the head of a ground instance of (67), or the head of a ground instance of (68). Hence, M_D is indeed an MKNF model of $\mathcal{K}'_D$.

In a completely analogous way one can show that M_D is the only MKNF model of $\mathcal{K}'$; furthermore, the previous proof shows that $\mathcal{K}'_D$ is stratified. Since $|\mathcal{K}'_D|$ is polynomial in $|D|$, the claim of this theorem follows. □

8.3 Nonstratified Nondisjunctive Programs

For $\mathcal{DL} = \emptyset$ and $\mathcal{DL} \in \mathrm{P}$, the coNP lower bound follows from coNP-hardness of answering queries in nondisjunctive programs with nonstratified negation under stable model semantics [6]. The upper bound follows from Theorem 6.29 and the fact that $\mathrm{NP}^{P^P} = \mathrm{NP}^P = \mathrm{NP}$.

For $\mathcal{DL} \in \mathrm{coNP}$, the Π^p_2 upper bound follows immediately from Theorem 7.1. For the lower bound, we consider the case when $\mathcal{DL}$ is the logic of disjunctive datalog rules interpreted under first-order semantics, which is data complete for coNP.

Theorem 8.2. *Let $\mathcal{DL}$ be the logic of disjunctive datalog interpreted under first-order semantics, $\mathcal{K}$ a stratified nondisjunctive hybrid MKNF knowledge base, and A a ground atom. Then, checking whether $\mathcal{K} \not\models \neg\, \mathbf{K}\, A$ is Σ^p_2-hard w.r.t. data complexity.*

Proof. The proof is by a reduction from 2-QBF—the problem of checking validity of a QBF $\varphi = \exists x_1, \ldots, x_n \forall y_1, \ldots, y_m : \psi$ for ψ a propositional formula—which is known to be Σ^p_2-hard [37]. As in the proof of Theorem 8.1, without loss of generality we can assume that ψ is of the form $\bigwedge_i \omega_i$, where each ω_i is of the form (49)–(51); with u_ψ we denote the propositional variable corresponding to ψ under this encoding. With $PVar$ we denote the set of all propositional variables of ψ, and let $XVar = \{x_1, \ldots, x_n\}$.

For an arbitrary 2-QBF φ, we construct a hybrid MKNF knowledge base $\mathcal{K}_\varphi = (\mathcal{O}_\varphi, \mathcal{P}_\varphi)$ as follows. We add to $\mathcal{O}_\varphi$ the axioms (52)–(61). For each variable $v \in PVar$, we add to $\mathcal{O}_\varphi$ the following axiom:

$$\mathbf{K}\,\mathrm{PVar}(u_c) \tag{69}$$

Furthermore, we encode the formula ψ into $\mathcal{O}_\varphi$ using the axioms (63)–(65) in the same way as in the proof of Theorem 8.1. For each variable $x \in XVar$, we add to $\mathcal{P}_\varphi$ the following axiom:

$$\mathbf{K}\,\mathrm{XVar}(u_x) \tag{70}$$

Finally, we add to $\mathcal{P}_\varphi$ the following axioms:

$$\mathbf{K}\,T(x) \leftarrow \mathbf{K}\,XVar(x), \mathbf{not}\,F(x) \tag{71}$$

$$\mathbf{K}\,F(x) \leftarrow \mathbf{K}\,XVar(x), \mathbf{not}\,T(x) \tag{72}$$

In computing the ground instantiation of $\mathcal{K}_\varphi$, it suffices to instantiate the rules (67) and (68) only for those $x = u_x$ for which a fact (70) occurs in $\mathcal{P}_\varphi$: all other instantiations are clearly true in any model of $\mathcal{K}_\varphi$. Let $\mathcal{K}'_\varphi$ be the ground instantiation of $\mathcal{K}_\varphi$ computed in this way; clearly, $\mathcal{K}_\varphi$ and $\mathcal{K}'_\varphi$ entail the same ground formulae. Furthermore, since the number of nonground rules in $\mathcal{K}_\varphi$ is bounded, $|\mathcal{K}'_\varphi|$ is polynomial in $|\mathcal{K}_\varphi|$. Finally, observe that $\mathcal{K}'_\varphi$ is a nondisjunctive nonstratified MKNF knowledge base.

We now prove that φ is valid if and only if $\mathcal{K}'_\varphi \not\models \neg\,\mathbf{K}\,T(u_\psi)$; this implies Σ_2^p-hardness of nonentailment of nonstratified MKNF knowledge bases and, consequently, the claim of this theorem.

If φ is valid, then there is a valuation ν for the variables x_i such that ψ is satisfied for each valuation of the variables y_i. Let M_φ be a set of first-order interpretations I satisfying the axioms from $\mathcal{O}'_\varphi$ such that $I \models T(u_{x_i})$ if $\nu(x_i) = \mathsf{true}$ and $I \models F(u_{x_i})$ if $\nu(x_i) = \mathsf{false}$. Clearly, M_φ satisfies all the rules from $\mathcal{K}'_\varphi$; furthermore, for each $M'_\varphi \supset M_\varphi$, the head of either (71) or (72) is invalidated. Hence, M_φ is an MKNF model of $\mathcal{K}'_\varphi$. Since ψ is true for each value of the variables y_i and the axioms (52)–(62) encode the semantics of Boolean connectives, $M_\varphi \models \mathbf{K}\,T(u_\psi)$; hence, $M_\varphi \not\models \neg\,\mathbf{K}\,T(u_\psi)$, so $\mathcal{K}'_\varphi \not\models \neg\,\mathbf{K}\,T(u_\psi)$ as well.

Conversely, if $\mathcal{K}'_\varphi \not\models \neg\,\mathbf{K}\,T(u_\psi)$, then there is an MKNF model M of $\mathcal{K}'_\varphi$ such that $M \not\models \neg\,\mathbf{K}\,T(u_\psi)$—that is, $M \models \mathbf{K}\,T(u_\psi)$. Clearly, for each variable x_i, either $M \models \mathbf{K}\,T(u_{x_i})$ or $M \models \mathbf{K}\,F(u_{x_i})$. Furthermore, by the preference semantics of MKNF, M contains an interpretation I for each valuation of the variables y_i. Now $M \models \mathbf{K}\,T(u_\psi)$ implies that ψ is true for each valuation of the variables y_i, so φ is valid. □

For a positive query, observe that a 2-QBF φ is valid if and only if the MKNF knowledge base $\mathcal{K}^+_\varphi = \mathcal{K}_\varphi \cup \{\mathbf{K}\,T(u_\psi)\}$ is satisfiable, which is the case if and only if $\mathcal{K}^+_\varphi \not\models \neg\,\mathbf{K}\,\mathsf{true}$.

8.4 Positive Programs

For any $\mathcal{DL}$ and a query of the form $\mathbf{K}\,A$, the coNP lower bound follows from coNP-hardness of answering positive queries in positive disjunctive datalog programs, and the coNP upper bound follows from Theorem 7.2. Namely, for positive queries and positive knowledge bases, it is sufficient to find any model, and not necessarily the minimal one, as reflected by Theorem 6.13.

For any $\mathcal{DL}$ and a query of the form $\neg\,\mathbf{K}\,A$, the Σ_2^p lower bound follows from Σ_2^p-hardness of answering negative queries in positive disjunctive datalog programs, and the Σ_2^p upper bound follows from Theorem 7.1. Namely, the minimality test is required for negative queries.

8.5 General Programs

The Σ_2^p lower bound follows from the Σ_2^p-hardness of answering positive and negative queries in disjunctive datalog under stable model semantics [6], and the Σ_2^p upper bound follows from Theorem 7.1.

9 Reusing QBF Solvers for Reasoning in MKNF

The reasoning algorithms for the flat and the positive fragments from Sections 6 and 7, if implemented in a naïve way, are unlikely to provide good performance in practice because they are based on a blind guess-and-check strategy. To obtain a practical algorithm, heuristics are required in order to structure the search space.

A lot of research has been invested into developing techniques for evaluating QBF efficiently, and several efficient QBF solvers are currently available.[4] To enable applying existing tools and optimizations techniques to reasoning in MKNF, in this section we present an algorithm that reduces checking entailment of a ground atom in a flat hybrid MKNF knowledge base to the validity problem of quantified Boolean formulae. The reduction can trivially be modified to handle the case of a positive knowledge base and a positive query. For nondisjunctive knowledge bases, reduction to QBF would produce an algorithm that is not optimal, so we do not consider such knowledge bases in this section.

Our approach was inspired by a similar idea presented in [9], where an embedding of propositional abduction, autoepistemic logic, default logic, disjunctive logic programming under stable models, and circumscription into QBF was presented. These translations were implemented in the QUIP[5] prototype and were shown to be useful in practice [9].

[4] See `http://www.qbflib.org/` for an overview of available tools.

[5] `http://www.kr.tuwien.ac.at/research/quip.html`

9.1 Obtaining a Propositional MKNF Formula

For a DL-safe hybrid MKNF knowledge base $\mathcal{K}$ and a formula $\psi = (\neg)\,\mathbf{K}\,A$ with A a ground atom, our goal is to compute a quantified Boolean formula φ which is valid if and only if $\pi(\mathcal{K}) \not\models_{\mathsf{MKNF}} \psi$. However, a fundamental mismatch between φ and $\pi(\mathcal{K})$ seems to exists: φ is a propositional formula, whereas $\pi(\mathcal{K})$ is a first-order formula. Hence, the first step in our algorithm is to convert $\pi(\mathcal{K})$ into a propositional MKNF formula that is MKNF equisatisfiable with $\pi(\mathcal{K})$.

No algorithm for embedding an arbitrary decidable first-order fragment $\mathcal{DL}$ into propositional logic is currently known; however, this is possible for a set of function-free clauses. Namely, the set of function-free clauses Γ is equisatisfiable with the set Γ_G of its ground instances; furthermore, since Γ does not contain function symbols, Γ_G is finite, so the reduction can actually be implemented in practice. Thus, our problem becomes simpler if we can reduce $\mathcal{DL}$ to the logic of function-free clauses.

In [19, 28], the authors present such a reduction for the DL $\mathcal{SHIQ}$: given a $\mathcal{SHIQ}$ knowledge base $\mathcal{O}$, they show how to compute a disjunctive datalog program $\mathsf{DD}(\mathcal{O})$ that is equisatisfiable with $\mathcal{O}$ and that entails the set of ground facts. The properties of the transformation are summarized in the following theorem:

Theorem 9.1 ([19, 28])**.** *There is an algorithm that, for a $\mathcal{SHIQ}$ knowledge base $\mathcal{O}$, computes a positive disjunctive datalog program $\mathsf{DD}(\mathcal{O})$ with the following properties:*

- *$\mathcal{O}$ is satisfiable if and only if $\mathsf{DD}(\mathcal{O})$ is satisfiable;*
- *$\mathcal{O} \models \alpha$ if and only if $\mathsf{DD}(\mathcal{O}) \models \alpha$ for α a ground atom of the form $(\neg)A(a)$, $(\neg)S(a,b)$, or $(\neg)a \approx b$, with A an atomic concept and S a simple role (both $\models$ denote standard entailment of first-order logic with equality);*
- *The size of the rules in $\mathsf{DD}(\mathcal{O})$ is at most exponential in $|\mathcal{O}|$;*
- *The number of the facts in $\mathsf{DD}(\mathcal{O})$ is at most polynomial in $|\mathcal{O}|$;*
- *$\mathsf{DD}(\mathcal{O})$ can be computed from $\mathcal{O}$ in time exponential in the size of the TBox of $\mathcal{O}$, and polynomial in the size of the ABox of $\mathcal{O}$.*

We call the atoms α for which $\mathcal{O} \models \alpha$ if and only if $\pi(\mathcal{O}) \models \alpha$ the *preserved atoms*; we call the predicates occurring in such atoms the *preserved predicates*. A positive disjunctive datalog program P does not contain negation-as-failure, so it can be interpreted as a first-order formula. Let $\pi(P) = \bigwedge_{r \in P} \forall \mathbf{x} : r$, where $\mathbf{x}$ is the set of the free variables of the rule r.

We now show that we can replace $\pi(\mathcal{O})$ with $\pi(\mathsf{DD}(\mathcal{O}))$ in translating a hybrid MKNF knowledge base $\mathcal{K}$ into first-order MKNF without affecting entailments of ground MKNF formulae.

Lemma 9.2. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe hybrid MKNF knowledge base containing only preserved predicates in rules, ψ a ground MKNF formula containing only preserved atoms, and $\mathsf{DD}(\mathcal{K})$ the following MKNF formula:*

$$\mathsf{DD}(\mathcal{K}) = \mathbf{K}\,\pi(\mathsf{DD}(\mathcal{O})) \wedge \pi(\mathcal{P})$$

Then, $\pi(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$ if and only if $\mathsf{DD}(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$.

Proof. By Lemma 4.6, $\mathcal{K} \models \psi$ if and only if $\mathcal{K}_G \models \psi$, where $\mathcal{K}_G$ is the grounding of $\mathcal{K}$ w.r.t. $O_\mathcal{K}$. Since $\mathcal{K}$ contains only preserved predicates in rules, the ground rules in $\mathcal{K}_G$ contain only preserved atoms. Now let $\sigma = \pi(\mathcal{K}_G)$ and $\sigma' = \mathsf{DD}(\mathcal{K}_G)$. In a run of the algorithm $\mathsf{not\text{-}entails}(\sigma, \psi)$, all first-order reasoning problems involved in checking Conditions (1)–(4) involve either checking satisfiability of ob_P for $\mathsf{ob}_P \wedge \neg\xi$ for ξ a ground atom. For Condition (5), computing $\psi[P]$ involves checking entailments of the form $\mathsf{ob}_P \models \varphi$ where φ is a ground formula, and checking satisfiability of $\mathsf{ob}_P \wedge \neg\psi[P]$ where $\psi[P]$ is a ground formula. The only difference in a run of $\mathsf{not\text{-}entails}(\sigma', \psi)$ is that the formula ob_P contains $\pi(\mathsf{DD}(\mathcal{O}))$ instead of $\pi(\mathcal{O})$. By assumption, the rules in $\mathcal{P}_G$ and ψ contain only preserved atoms, so ξ and φ contain only such atoms as well. Hence, each condition in $\mathsf{not\text{-}entails}(\sigma', \psi)$ holds if and only if the corresponding condition holds in $\mathsf{not\text{-}entails}(\sigma, \psi)$ by Theorem 9.1, so $\mathsf{not\text{-}entails}(\sigma, \psi)$ and $\mathsf{not\text{-}entails}(\sigma', \psi)$ return the same values. □

Now $\pi(\mathsf{DD}(\mathcal{K}_G))$ can be grounded w.r.t. the constants from $O_\mathcal{K}$:

Lemma 9.3. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a DL-safe hybrid MKNF knowledge base containing only preserved predicates in rules and let ψ be a ground MKNF formula containing only preserved atoms, Furthermore, let $\mathsf{gr}'(\mathcal{K})$ be the MKNF formula obtained from $\mathsf{DD}(\mathcal{K})$ by grounding $\pi(\mathsf{DD}(\mathcal{O}))$ and $\pi(\mathcal{P})$ w.r.t. $O_\mathcal{K}$. Finally, let $\mathsf{gr}(\mathcal{K}) = \mathsf{gr}'(\mathcal{K}) \wedge \mathbf{K}\,\Theta$ where Θ is a conjunction containing the following conjuncts:*

- *$a \approx a$ for each $a \in O_\mathcal{K}$;*
- *$a \approx b \supset b \approx a$ for each $a, b \in O_\mathcal{K}$;*
- *$a \approx b \wedge b \approx c \supset a \approx c$ for each $a, b, c \in O_\mathcal{K}$; and*
- *$P(\mathbf{t_1}, a, \mathbf{t_2}) \wedge a \approx b \supset P(\mathbf{t_1}, b, \mathbf{t_2})$ for each $a, b \in O_\mathcal{K}$, each vectors of constants $\mathbf{t_1}$ and $\mathbf{t_2}$ from $O_\mathcal{K}$, and each predicate P from $\mathcal{K}$.*

The formula $\mathsf{gr}(\mathcal{K})$ is ground, so we interpret it as a propositional MKNF formula where each ground atom corresponds to one proposition. Then, $\pi(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$ if and only if $\mathsf{gr}(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$.

Proof. By Lemma 9.2, $\pi(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$ if and only if $\mathsf{DD}(\mathcal{K}) \models_{\mathsf{MKNF}} \psi$. It is now clear that the algorithm $\mathsf{not\text{-}entails}(\sigma, \psi)$ returns the same values for $\sigma = \mathsf{DD}(\mathcal{K}_G)$ and for $\sigma = \mathsf{gr}(\mathcal{K})$: the formula $\pi(\mathsf{DD}(\mathcal{K}))$ does not contain existential quantifiers and is safe (each variable of an implication occurs in the antecedent), so grounding it does not affect satisfiability or entailment of ground facts; furthermore, Θ ensures that the congruence properties of equality are satisfied as usual. □

Grounding large knowledge bases is unlikely to yield good practical results. Current state-of-the-art answer set solvers, such as DLV [10] or Smodels [38], address this problem by *intelligent grounding* [12]—a technique that identifies the part of the program that needs to be grounded. This enables answer set solvers to ground only the relevant part of the program, and thus to significantly reduce the size of the obtained ground program. Extending these techniques to our case will be in the focus of our future research.

9.2 Reducing a Flat Propositional MKNF Formula into QBF

Let σ be a flat propositional MKNF formula and ψ a propositional formula of the form $(\neg)\,\mathbf{K}\,a$. We now show how to compute the quantified Boolean formula $\mathsf{flat}(\sigma, \psi)$ which is valid if and only if $\sigma \not\models_{\mathsf{MKNF}} \psi$. To each $\mathbf{K}$-atom $\mathbf{K}\,\xi$, we assign two unique propositional variables a_ξ and b_ξ. We use the following abbreviations: T is the set of all propositional variables used in all modal atoms $\mathbf{K}\,\xi$ from $\mathsf{KA}(\sigma)$, $T_a = \{a_\xi \mid \mathbf{K}\,\xi \in \mathsf{KA}(\sigma)\}$, and $T_b = \{b_\xi \mid \xi \in \mathsf{KA}(\sigma)\}$. With $\sigma[\mathbf{K}, T_a]$ we denote the MKNF formula obtained by replacing each modal atom $\mathbf{K}\,\xi$ in σ with a_ξ; with $\sigma[\mathbf{not}, T_a]$ we denote the MKNF formula obtained by replacing each modal atom $\mathbf{not}\,\xi$ in σ with $\neg a_\xi$; finally, $\sigma[T_a] = \sigma[\mathbf{K}, T_a][\mathbf{not}, T_a]$. We use analogous definitions for $\sigma[\mathbf{K}, T_b]$, $\sigma[\mathbf{not}, T_b]$, and $\sigma[T_b]$.

Definition 9.4. *For a flat propositional MKNF formula σ and ψ a propositional formula of the form $(\neg)\,\mathbf{K}\,a$, the quantified Boolean formula $\mathsf{flat}(\sigma, \psi)$ is defined as follows:*

$$\mathsf{flat}(\sigma,\psi) = \exists T_a : [\varphi_1 \wedge \varphi_2 \wedge \varphi_3 \wedge \varphi_4 \wedge \varphi_5] \quad \mathsf{sat}(\varphi) = \exists T : \varphi$$

$$\varphi_{\mathsf{ob}} = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} a_\xi \supset \xi \qquad \varphi_1 = \sigma[T_a]$$

$$\varphi_2 = \mathsf{sat}(\varphi_{\mathsf{ob}}) \qquad \varphi_3 = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} \neg a_\xi \supset \mathsf{sat}(\varphi_{\mathsf{ob}} \wedge \neg\xi)$$

$$\varphi_4 = \forall T_b : [(T_b \leq T_a) \wedge \neg\,(T_a \leq T_b)] \supset (\neg\varphi_a \vee \neg\varphi_b \vee \neg\varphi_c)$$

$$T_a \leq T_b = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} a_\xi \supset b_\xi \qquad T_b \leq T_a = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} b_\xi \supset a_\xi$$

$$\varphi_{\mathsf{ob}'} = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} b_\xi \supset \xi \qquad \varphi_a = \sigma[\mathbf{not}, T_a][T_b]$$

$$\varphi_b = \mathsf{sat}(\varphi_{\mathsf{ob}'}) \qquad \varphi_c = \bigwedge_{\mathbf{K}\,\xi \in \mathsf{KA}(\sigma)} \neg b_\xi \supset \mathsf{sat}(\varphi_{\mathsf{ob}'} \wedge \neg\xi)$$

$$\varphi_5 = \begin{cases} \mathsf{sat}(\varphi_{\mathsf{ob}} \wedge \neg a) & \text{if } \psi = \mathbf{K}\,a \\ \neg\mathsf{sat}(\varphi_{\mathsf{ob}} \wedge \neg a) & \text{if } \psi = \neg\,\mathbf{K}\,a \end{cases}$$

Theorem 9.5. *Let σ be a flat propositional MKNF formula and ψ a propositional formula of the form $(\neg)\,\mathbf{K}\,a$. Then, $\sigma \not\models_{\mathsf{MKNF}} \psi$ if and only if the quantified Boolean formula $\mathsf{flat}(\sigma,\psi)$ is valid.*

Proof. By Theorem 6.11, $\sigma \not\models_{\mathsf{MKNF}} \psi$ if and only if $\mathsf{not\text{-}entails}(\sigma,\psi)$ returns true. Note that a partition (P,N) of $\mathsf{KA}(\sigma)$ from the algorithm can be encoded as a valuation of propositional atoms from T_a by assigning true to those a_ξ for which $\mathbf{K}\,\xi \in P$. A partition $(P', N \cup N')$ can be encoded as a valuation of propositional atoms T_b in an analogous way.

For a valuation for T_a (or, equivalently, a partition (P,N)), the formula φ_{ob} is equivalent to the formula ob_P; similarly, for a valuation for T_b (or, equivalently, a partition $(P', N \cup N')$), the formula $\varphi_{\mathsf{ob}'}$ is equivalent to the formula $\mathsf{ob}_{P'}$. Furthermore, $\mathsf{sat}(\varphi)$ clearly evaluates to true if and only if the formula φ is satisfiable. Hence, φ_1 encodes exactly Condition (1), φ_2 encodes exactly Condition (2), φ_3 encodes exactly Condition (3), φ_a encodes exactly Condition (a), φ_b encodes exactly Condition (b), φ_c encodes exactly Condition (c), and φ_5 encodes exactly Condition (5) of $\mathsf{not\text{-}entails}(\sigma,\psi)$. Finally, $[(T_b \leq T_a) \wedge \neg\,(T_a \leq T_b)]$ evaluates to true if and only if $P' \subset P$. It is now clear that $\mathsf{flat}(\sigma,\psi)$ is valid exactly if all conditions of the algorithm $\mathsf{not\text{-}entails}(\sigma,\psi)$ are satisfied. □

Observe that, in prenex normal form, $\mathsf{flat}(\sigma,\psi)$ has the quantifier prefix $\exists\forall$, so validity of the formula can be decided in Σ_2^p. Hence, for $\mathcal{DL}$ a description logic which is data complete for NP, reasoning with hybrid MKNF knowledge bases by reduction to QBF gives an algorithm with optimal worst-case complexity. Also, for $\sigma = \mathsf{gr}(\mathcal{K})$, we can somewhat simplify $\mathsf{flat}(\sigma,\psi)$ since the propositional atoms corresponding to Θ and the grounding of $\mathbf{K}\,\pi(\mathsf{DD}(\mathcal{O}))$ must evaluate to true in each valuation T_a.

10 Conclusion

Based on the logic of Minimal Knowledge and Negation as Failure by Lifschitz [25], in this paper we present the formalism of hybrid MKNF knowledge bases that seamlessly integrates DL with logic programming. In this way, we obtain a powerful hybrid formalism that combines the best features of both worlds: on the one hand, it provides DL-style modeling of taxonomic knowledge, and on the other hand, it provides LP-style constructs, such as negation-as-failure. To make our formalism decidable, we apply the well-known DL-safety restriction that makes the rules applicable only to explicitly known individuals, thus trading some expressivity for decidability.

We present several reasoning algorithms for different fragments of our logic. Furthermore, we analyze the data complexity of each fragment, and show that, in many cases, reasoning with hybrid MKNF knowledge bases is not harder than in the corresponding fragment of logic programming.

To enable reusing existing heuristics developed for reasoning with quantified Boolean formulae, we encode the entailment problem for hybrid MKNF knowledge bases into QBF. This encoding can be applied to hybrid MKNF knowledge bases based on the $\mathcal{SHIQ}$ description logic, and it is based on the reduction of $\mathcal{SHIQ}$ knowledge bases into disjunctive datalog from [19].

A challenging problem for our future work is to define a well-founded semantics [39] for our formalism. The well-founded semantics is often used in practice because of its polynomial data complexity. Such an extension is nontrivial, because it actually requires redefining the semantics of MKNF. This might be feasible by building on the ideas from [14].

References

[1] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.

[2] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, January 2003.

[3] F. Baader and B. Hollunder. Embedding Defaults into Terminological Knowledge Representation Formalisms. *Journal of Automated Reasoning*, 14(1):149–180, 1995.

[4] P. Bonatti, C. Lutz, and F. Wolter. Description Logics with Circumscription. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 400–410, Lake District, UK, June 2–5 2006. AAAI Press.

[5] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. DL-Lite: Tractable Description Logics for Ontologies. In M. M. Veloso and S. Kambhampati, editors, *Proc. of the 20th National Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, Pittsburgh, PA, USA, July 9–13 2005. AAAI Press.

[6] E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. *ACM Computing Surveys*, 33(3):374–425, 2001.

[7] F. M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and Description Logics. *Journal of Intelligent Information Systems*, 10(3):227–252, 1998.

[8] F. M. Donini, D. Nardi, and R. Rosati. Description Logics of Minimal Knowledge and Negation as Failure. *ACM Transactions on Computational Logic*, 3(2):177–225, 2002.

[9] U. Egly, T. Eiter, H. Tompits, and S. Woltran. Solving Advanced Reasoning Tasks Using Quantified Boolean Formulas. In *Proc. of the 17th National Conf. on Artificial Intelligence (AAAI 2000)*, pages 417–422, Austin, TX, USA, July 30–August 3 2000. AAAI Press.

[10] T. Eiter, W. Faber, N. Leone, and G. Pfeifer. Declarative problem-solving using the DLV system. *Logic-Based Artificial Intelligence*, pages 79–103, 2000.

[11] T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.

[12] T. Eiter, N. Leone, C. Mateis, G. Pfeifer, and F. Scarcello. A Deductive System for Non-Monotonic Reasoning. In J. Dix, U. Furbach, and A. Nerode, editors, *Proc. of the 4th Int. Conf. on Logic Programming and Non-monotonic Reasoning (LPNMR '97)*, volume 1265 of *LNAI*, pages 364–375, Dagstuhl, Germany, July 28–31 1997. Springer.

[13] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Combining Answer Set Programming with Description Logics for the Semantic Web. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 141–151, Whistler, Canada, June 2–5 2004. AAAI Press.

[14] T. Eiter, T. Lukasiewicz, R. Schindlauer, and H. Tompits. Well-Founded Semantics for Description Logic Programs in the Semantic Web. In G. Antoniou and H. Boley, editors, *Proc. of the 3rd Int. Workshop on Rules and Rule Markup Languages for the Semantic Web*

(RuleML 2004), volume 3323 of *LNCS*, pages 81–97, Hiroshima, Japan, November 8 2004. Springer.

[15] M. Fitting. *First-Order Logic and Automated Theorem Proving, 2nd Edition*. Texts in Computer Science. Springer, 1996.

[16] M. Gelfond and V. Lifschitz. The Stable Model Semantics for Logic Programming. In *Proc. of the 5th Int. Conf. on Logic Programming (ICLP '88)*, pages 1070–1080, Seattler, WA, USA, August 15–19 1988. MIT Press.

[17] G. Gottlob. NP Trees and Carnap's Modal Logic. *Journal of the ACM*, 42(2):421–457, 1995.

[18] I. Horrocks and P. F. Patel-Schneider. A Proposal for an OWL Rules Language. In *Proc. of the 13th Int. World Wide Web Conference (WWW 2004)*, pages 723–731, New York, NY, USA, May 17–22 2004. ACM Press.

[19] U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ Description Logic to Disjunctive Datalog Programs. In D. Dubois, C. A. Welty, and M.-A. Williams, editors, *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2004)*, pages 152–162, Whistler, Canada, June 2–5 2004. AAAI Press.

[20] U. Hustadt, B. Motik, and U. Sattler. Data Complexity of Reasoning in Very Expressive Description Logics. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, pages 466–471, Edinburgh, UK, July 30–August 5 2005. Morgan Kaufmann Publishers.

[21] O. Kutz, I. Horrocks, and U. Sattler. The Even More Irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[22] H. J. Levesque. Foundations of a Functional Approach to Knowledge Representation. *Artificial Intelligence*, 23(2):155–212, 1984.

[23] A. Y. Levy and M.-C. Rousset. Combining Horn Rules and Description Logics in CARIN. *Artificial Intelligence*, 104(1–2):165–209, 1998.

[24] V. Lifschitz. On Open Defaults. In *Proc. of the Symposium on Computational Logic*, ESPRIT Basic Research Series, pages 80–95, Bruxelles, Belgium, November 13–14 1990. Springer.

[25] V. Lifschitz. Nonmonotonic Databases and Epistemic Queries. In J. Mylopoulos and R. Reiter, editors, *Proc. of the 12th Int. Joint Conf.*

on Artificial Intelligence (IJCAI '91), pages 381–386, Sydney, Australia, August 24–30 1991. Morgan Kaufmann Publishers.

[26] V. Lifschitz. Minimal Belief and Negation as Failure. *Artificial Intelligence*, 70(1–2):53–72, 1994.

[27] B. Motik, U. Sattler, and R. Studer. Query Answering for OWL-DL with rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):41–60, 2005.

[28] Boris Motik. *Reasoning in Description Logics using Resolution and Deductive Databases*. PhD thesis, Univesität Karlsruhe (TH), Karlsruhe, Germany, January 2006.

[29] C. H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1993.

[30] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1–2):81–132, 1980.

[31] R. Reiter. What Should a Database Know? *Journal of Logic Programming*, 14(1–2):127–153, 1992.

[32] R. Rosati. Reasoning about Minimal Belief and Negation as Failure. *Journal of Artificial Intelligence Research*, 11:277–300, 1999.

[33] R. Rosati. Minimal Belief and Negation as Failure in Multi-Agent Systems. *Annals of Mathematics and Artificial Intelligence*, 37(1–2):5–32, 2003.

[34] R. Rosati. On the decidability and complexity of integrating ontologies and rules. *Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):61–73, 2005.

[35] R. Rosati. Semantic and Computational Advantages of the Safe Integration of Ontologies and Rules. In F. Fages and S. Soliman, editors, *Proc. of the 3rd Int. Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR 2005)*, volume 3703 of *LNCS*, pages 50–64, Dagstuhl Castle, Germany, September 11–16 2005. Springer.

[36] R. Rosati. $\mathcal{DL} + log$: A Tight Integration of Description Logics and Disjunctive Datalog. In P. Doherty, J. Mylopoulos, and C. A. Welty, editors, *Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006)*, pages 68–78, Lake District, UK, June 2–5 2006. AAAI Press.

[37] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1976.

[38] T. Syrjänen and I. Niemelä. The Smodels System. In T. Eiter, W. Faber, and M. Truszczynski, editors, *Proc. 6th Int. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR 2001)*, volume 2173 of *LNAI*, pages 434–438, Vienna, Austria, September 17–19 2001. Springer.

[39] A. van Gelder, K. Ross, and J. S. Schlipf. The Well-Founded Semantics for General Logic Programs. *Journal of the ACM*, 38(3):620–650, 1991.