TPLP: Page 1–20.

doi:10.1017/S1471068424000255

On the Foundations of Conflict-Driven Solving for Hybrid MKNF Knowledge Bases

RILEY KINAHAN, SPENCER KILLEN, KEVIN WAN and JIA-HUAI YOU
University of Alberta, Edmonton, Alberta, Canada
(*e-mails:* rdkinaha@ualberta.ca,sjkillen@ualberta.ca
kcwan1@ualberta.ca,jyou@ualberta.ca)

submitted 19 August 2024; accepted 13 September 2024

Abstract

Hybrid MKNF Knowledge Bases (HMKNF-KBs) constitute a formalism for tightly integrated reasoning over closed-world rules and open-world ontologies. This approach allows for accurate modeling of real-world systems, which often rely on both categorical and normative reasoning. Conflict-driven solving is the leading approach for computationally hard problems, such as satisfiability (SAT) and answer set programming (ASP), in which MKNF is rooted. This paper investigates the theoretical underpinnings required for a conflict-driven solver of HMKNF-KBs. The approach defines a set of completion and loop formulas, whose satisfaction characterizes MKNF models. This forms the basis for a set of nogoods, which in turn can be used as the backbone for a conflict-driven solver.

KEYWORDS: hybrid mKNF, ASP, conflict-driven solving, loop formulas, nogoods

1 Introduction

Real-world problems often require integrated reasoning of both rules-based and ontological knowledge, spanning domains such as customs, healthcare, and penal systems (Alberti *et al.* 2012; Knorr 2021). For example, in customs, ontological reasoning aids in categorizing imported goods, while rule-based reasoning determines inspection procedures (Knorr 2021). The prevalence of such applications has led to the development of frameworks for reconciling ontologies with rules, including Hybrid MKNF Knowledge Bases (Motik and Rosati 2010).

A Hybrid MKNF Knowledge Base (HMKNF-KB) consists of two components: a logic program of rules, such as in answer set programming (ASP), and an ontology representable in a decidable fragment of first-order logic, most often under a description logic. The main feature of HMKNF-KBs, compared to other approaches that combine ASP with description logics, is the *tight* integration between their two components. Here, tightness refers to the ability of an integration to allow for derivation within one component based on conclusions from the other. For certain applications, a one-way flow of information is sufficient. However, greater tightness results in a richer interplay between two knowledge sources.

Some integrations have partial tightness, such as *dl-programs* (Eiter *et al.* 2005). While these allow for back-and-forth derivation between two components, it must be localized to specific dl-atoms within rules, which may query the ontology. Conversely, HMKNF-KBs fully integrate the two components – inferences in one are immediately available to the other. The integration also has several other desirable properties, including *faithfulness* to each of the underlying components, *flexibility* in whether any predicate can be viewed under closed or open-world reasoning, and *decidability* (under the *DL-safety* assumption) (Motik and Rosati 2010).[1]

The well-founded semantics for Hybrid MKNF (Knorr *et al.* 2011) have enjoyed considerable focus due to their polynomial complexity (under some assumptions), and form the basis for the reasoner NoHR (Kasalica *et al.* 2020). However, reasoning tools for the stable model semantics are still relatively limited. Ji *et al.* (2017) define well-founded operators for non-disjunctive Hybrid MKNF. Killen and You (2021) generalize this approach to the disjunctive case for a DPLL-based solver (Nieuwenhuis *et al.* 2006). This represents significant progress towards efficient solving, but is still behind state-of-the-art conflict-driven solving, which is widely adopted by ASP and its extensions.

While there have been advancements in resolution or query-based solving for Hybrid MKNF (Alferes *et al.* 2013), we strictly focus on solvers that employ bottom-up model-search. The prominent modern ASP solvers, Clasp (Gebser *et al.* 2013) and WASP (Alviano *et al.* 2015) (within the Potassco/Clingo (Gebser *et al.* 2019) and DLV systems (Adrian *et al.* 2018) resp.), combine conflict-driven SAT solving with native ASP propagation, to achieve a high degree of performance. Clingo is designed to be extensible, and numerous applications have been built on it. The relevant case here is the system DLVHEX (Redl 2016; Eiter *et al.* 2018), which allows integrating rules with arbitrary external sources and can solve dl-programs (Eiter *et al.* 2006). Despite being only partially tight, dl-programs are closely related to HMKNF-KBs. It has been shown that under reasonable assumptions, HMKNF-KBs can be translated to dl-programs (Eiter and Šimkus 2015). Thus, one can translate HMKNF-KBs to dl-programs and use DLVHEX to compute the MKNF models; however, it is unclear whether this would be as powerful as a native conflict-driven approach.

Therefore, this paper aims to develop a general theory relating HMKNF-KB model computation to conflict-driven solving. In particular, we focus on the notion of a **K**-*interpretation*, which represents everything concludable under an MKNF model. To accomplish this task, we first characterize whether **K**-interpretation s correspond to models according to whether they satisfy a set of formulas; we call these the *completion* and *loop formulas*. They follow the naming convention of, and are directly inspired by, the formulas of Lee and Lifschitz (2003), which characterize answer sets of disjunctive logic programs. We then define *nogoods* in the sense of Gebser *et al.* (2012), which capture the constraints induced by our formulas and thereby characterize models of HMKNF-KBs. Finally, we give an overview of how our nogoods can be used within conflict-driven algorithms. We conclude with comments on related and future work, including practical considerations for implementing a solver.

[1] There are other systems of tight integration of rules and first-order formulas such as IDP-systems Wittocx et al. (2008), however they are beyond the scope of this paper since their reasoning task deals with extension of classical first-order logic which presents a very different challenge.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Complete proofs are provided in a separately available Appendix. To aid reading, we provide proof sketches for the two most crucial claims.

2 Preliminaries

2.1 Minimal knowledge and negation as failure (MKNF)

MKNF is a nonmonotonic logic formulated by Lifschitz (1991). MKNF formulas extend first-order formulas with two modal operators, **K** for minimal knowledge, and **not** for negation as failure. We define a first-order interpretation I as usual and denote the universe of I by $|I|$. An *MKNF structure* is a triple (I, M, N), where M and N are sets of first-order interpretations within the universe $|I|$. The language of MKNF formulas contains a constant for each element of $|I|$, which we call a *name*. We define the satisfaction relation between an MKNF structure (I, M, N) and an MKNF formula as follows:

$$
\begin{aligned}
&(I, M, N) \models \phi \text{ } (\phi \text{ is a first-order atom}) \text{ if } \phi \text{ is true in } I,\\
&(I, M, N) \models \neg\phi \text{ if } (I, M, N) \not\models \phi,\\
&(I, M, N) \models \phi_1 \wedge \phi_2 \text{ if } (I, M, N) \models \phi_1 \text{ and } (I, M, N) \models \phi_2,\\
&(I, M, N) \models \exists x\phi \text{ if } (I, M, N) \models \phi[a \setminus x] \text{ for some } a,\\
&(I, M, N) \models \mathbf{K}\phi \text{ if } (J, M, N) \models \phi \text{ for all } J \in M,\\
&(I, M, N) \models \mathbf{not}\ \phi \text{ if } (J, M, N) \not\models \phi \text{ for some } J \in N.
\end{aligned}
$$

The symbols $\top, \bot, \vee, \forall,$ and $\supset$ are interpreted as usual.

An *MKNF interpretation* M is a nonempty set of first-order interpretations. Throughout this work, we employ the *standard name assumption* to avoid unintended behaviors (Motik and Rosati 2010). This assumes all interpretations are Herbrand interpretations with a countably infinite number of additional constants, and that the predicate $\approx$ is a congruence relation. Thus, we do not explicitly mention the universe associated with interpretations.

An MKNF interpretation M satisfies an MKNF formula ϕ, written $M \models_{\text{MKNF}} \phi$, if $(I, M, M) \models \phi$ for each $I \in M$.

Definition 1
An MKNF interpretation M is an MKNF model of an MKNF formula ϕ, if $M \models_{\text{MKNF}} \phi$, and for all MKNF interpretations M' s.t. $M' \supset M$, we have $\forall I' \in M, (I', M', M) \not\models \phi$.

2.2 Hybrid MKNF knowledge bases (HMKNF-KBs)

Motik and Rosati (2010) identify a subset of MKNF formulas as Hybrid MKNF. In this new language, an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ consists of a finite set of *rules* termed a *rule base* $\mathcal{P}$, and an *ontology* $\mathcal{O}$ translatable to first-order logic as $\pi(\mathcal{O})$. A rule r is of the form

$$h_0, \ldots, h_m \leftarrow p_0, \ldots, p_j, \neg n_0, \ldots, \neg n_k$$

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

where h_i, p_i, and n_i are function-free first-order atoms. We denote $body^+(r) = \{p_0, \ldots, p_j\}$, $body^-(r) = \{n_0, \ldots, n_k\}$, $Body(r) = \bigwedge body^+(r) \wedge \neg \bigvee body^-(r)$, and $head(r) = \{h_0, \ldots, h_m\}$. A rule r's semantics is governed by the following MKNF formula.

$$\pi(r) = \forall \overrightarrow{x} : (\mathbf{K}h_0 \vee \cdots \vee \mathbf{K}h_z) \subset (\mathbf{K}p_0 \wedge \cdots \wedge \mathbf{K}p_j \wedge \mathbf{not}\ n_0 \wedge \cdots \wedge \mathbf{not}\ n_k)$$

where $\overrightarrow{x}$ is the vector of free variables in r. Naturally, a rule base $\mathcal{P}$ translates to an MKNF formula as $\pi(\mathcal{P}) = \bigcup_{r \in \mathcal{P}} \pi(r)$. We say that an MKNF interpretation M is an *MKNF model* of the HMKNF-KB $\mathcal{K}$ if it is an MKNF model of the MKNF formula $\pi(\mathcal{K}) = \mathbf{K}\pi(\mathcal{O}) \wedge \pi(\mathcal{P})$.

A program $\mathcal{P}$ is ground if no rule $r \in \mathcal{P}$ has variables. Motik and Rosati (2010) show that under the assumption of *DL-safety*,[2] a first-order rule base is semantically equivalent to a finite ground rule base and that decidability is guaranteed for HMKNF-KBs with decidable ontologies. In this work, we assume rule bases are ground.

We define $\mathbf{KA}(\mathcal{K})$ be the set containing every atom ϕ that occurs in $\mathcal{P}$ (either as $\mathbf{K}\phi$ or $\mathbf{not}\ \phi$) and we use $\mathbf{KA}(\mathcal{O})$ to denote the maximal subset of $\mathbf{KA}(\mathcal{K})$ s.t. for each $p(t_1, \ldots, t_m) \in \mathbf{KA}(\mathcal{O})$, the predicate p also appears in $\pi(\mathcal{O})$.

We define the *objective knowledge* of an HMKNF-KB $\mathcal{K}$ w.r.t. a set $S \subseteq \mathbf{KA}(\mathcal{K})$ as the set $OB_{\mathcal{O},S} = \{\pi(\mathcal{O})\} \cup \{\phi \mid \phi \in S\}$. Intuitively, $OB_{\mathcal{O},S}$ is a first-order formula that considers $\mathcal{O}$ with the assumption that $\mathbf{K}\phi$ holds for each $\phi \in S$. In this paper, we prefer polynomial ontologies, that is for any $S \subseteq \mathbf{KA}(\mathcal{O})$ and $a \in \mathbf{KA}(\mathcal{O})$, the relation $OB_{\mathcal{O},S} \models a$ can be checked in polynomial time.

The following notion of a **K**-interpretation allows for a simplified representation of an MKNF interpretation as a single set of atoms. A central focus of this work is to show which **K**-interpretation s are *induced* by MKNF models. Such a relationship is not completely straightforward, due to the quite different nature of the two representations.

Definition 2
A **K**-interpretation is any set of atoms $\hat{I} \subseteq \mathbf{KA}(\mathcal{K})$. An MKNF interpretation M induces a **K**-interpretation $\hat{I}$, if $\hat{I} = \{a \in \mathbf{KA}(\mathcal{K}) \mid M \models_{\text{MKNF}} \mathbf{K}a\}$. Whereas, a **K**-interpretation $\hat{I}$ extends to an MKNF interpretation M if $M = \{I \mid I \models OB_{\mathcal{O},\hat{I}}\}$.

Above, we use the notation $\hat{I}$ to express that as an interpretation in an entailment relation, $\hat{I} \models \phi$, any atom $a \in \mathbf{KA}(\mathcal{K})$ but not in $\hat{I}$ is assigned to *false*.

2.3 Assignments and nogoods

Nogoods (Gebser *et al.* 2012) act as canonical representations of Boolean constraints, reflecting partial *assignments*, which cannot be extended to a *solution*. A *nogood* $\{\sigma_1, \ldots, \sigma_n\}$, is a set of literals σ_i, of the form $\mathbf{T}v_i$ or $\mathbf{F}v_i$ for $1 \leq i \leq n$, where v_i is a propositional variable. The *complement* of a literal, is referred to by $\overline{\mathbf{T}v} = \mathbf{F}v$ and $\overline{\mathbf{F}v} = \mathbf{T}v$. For any set δ of literals, $\delta^T = \{v \mid \mathbf{T}v \in \delta\}$ and $\delta^F = \{v \mid \mathbf{F}v \in \delta\}$. The set of variables occurring within a set of nogoods Δ, is denoted $var(\Delta) = \bigcup_{\delta \in \Delta}(\delta^{\mathbf{T}} \cup \delta^{\mathbf{F}})$. An *assignment* A for Δ is any subset of $\{\mathbf{T}v, \mathbf{F}v \mid v \in var(\Delta)\}$ such that $A^{\mathbf{T}} \cap A^{\mathbf{F}} = \emptyset$. A

[2] An HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ is *DL-safe*, if for all rules $r \in \mathcal{P}$, all variables present in r appear within $body^+(r)$, under a predicate that does not occur in $\pi(\mathcal{O})$.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

solution for Δ is an assignment A for Δ, such that $A^{\mathbf{T}} \cup A^{\mathbf{F}} = var(\Delta)$ and $\delta \not\subseteq A$ for all $\delta \in \Delta$. A nogood δ is *unit-resulting* for an assignment A if $|\delta \setminus A| = 1$. For a literal $\mathbf{T}\sigma$ in an assignment, σ is true within the assignment, whereas for $\mathbf{F}\sigma$, σ is false.

3 Dependency graph

A guiding principle behind the dependency graph of a logic program is to provide a syntactic overapproximation of the true semantic dependency between atoms within the program. This overapproximation can be used to bound the possible sources of circular derivation to *loops*, sets of atoms where there is a path between any two in the set.

Given an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, a dependency graph $G(\mathcal{K})$ is the graph containing all vertices and edges in either of two dependency graphs, $G(\mathcal{P})$ or $G(\mathcal{O})$. $G(\mathcal{P})$ consists of vertices for atoms in $\mathbf{KA}(\mathcal{K})$ and edges from a vertex a to a vertex b if there is a rule $r \in \mathcal{P}$ such that $a \in head(r)$ and $b \in body^{+}(r)$. Below, we offer a characterization of $G(\mathcal{O})$ in terms of the entailment relation.

Definition 3
An ontology dependency graph $G(\mathcal{O})$ for an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, contains vertices for each atom in $\mathbf{KA}(\mathcal{O})$. There is an edge from vertex a to vertex b in $G(\mathcal{O})$ if for some $S \subseteq \mathbf{KA}(\mathcal{O})$, where $a \notin S$ and $b \in S$, we have

$$OB_{\mathcal{O},S} \models a, \qquad \text{(contributes to derivation)}$$
$$OB_{\mathcal{O},S} \not\models \bot \text{ and,} \qquad \text{(is consistent)}$$
$$OB_{\mathcal{O},S\setminus\{b\}} \not\models a. \qquad \text{(is minimal)}$$

It is not difficult to see that the ontology dependency graph defined here is an overapproximation of true semantic dependencies based on entailment relation. In addition, there is a unique minimal version of $G(\mathcal{O})$ containing only those edges that are strictly required by its definition. However we do not expect to tractably generate such a dependency graph in general. This is because the ontology of an HMKNF-KB is allowed to take a variety of forms, so dependency graph generation for any particular ontology requires separate study. As such, to keep our approach general we assume some version of $G(\mathcal{O})$ is provided. To ensure correctness (of overapproximation) the assumed dependency graph must simply contain every edge within the minimal version, in accordance with Definition 3. A similar approach of leveraging externally provided information for dependency pruning is explored within (Eiter and Kaminski 2021), to enhance minimality checking.

Similar to Clark (1977), we call $\mathcal{K}$ *tight* if $G(\mathcal{K})$ is acyclic. We denote by $Loops(\mathcal{K})$ the set of loops of the dependency graph $G(\mathcal{K})$.

4 Completion and loop formulas

In this section we characterize models of HMKNF-KBs through logical formulas. Our approach follows that of Lee and Lifschitz (2003) who defined *completion* and *loop formulas* to capture the answer sets of disjunctive logic programs. While this work is self-contained, we draw frequent comparison to their seminal work to ease understanding.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

4.1 Completion

Lee and Lifschitz (2003) show that an interpretation of a tight disjunctive logic program is an answer set if and only if it satisfies a set of formulas termed the *completion*. A program's completion is composed of a *rule completion*, clauses that ensure atoms whose truth is implied must be included in an answer set, and a *support completion*, clauses that ensure true atoms within an interpretation are supported by some rule.

The rule completion of Lee and Lifschitz (2003) can be easily adopted as follows.

Definition 4
The rule completion of a rule base $\mathcal{P}$ is $\mathcal{P}_{rule} = \{Body(r) \supset \bigvee head(r) \mid r \in \mathcal{P}\}$.

The formula $\mathcal{P}_{rule}$ guarantees that for any satisfying **K**-interpretation $\hat{I}$, atoms implied by the rules of $\mathcal{P}$ under $\hat{I}$ are contained in $\hat{I}$. We can construct an analogous formula for an ontology $\mathcal{O}$ that requires all atoms entailed by $\mathcal{O}$ given $\hat{I}$ to be within $\hat{I}$. We define this notion as *saturation* and formulate what we call *saturation completion*.

Definition 5
Given an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, a **K**-interpretation $\hat{I}$ of $\mathcal{K}$ is saturated if $OB_{\mathcal{O},\hat{I}} \not\models \bot$, and $\forall a \in \mathbf{KA}(\mathcal{K})$ such that $OB_{\mathcal{O},\hat{I}} \models a$, $a \in \hat{I}$.

Definition 6
Let $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ be an HMKNF-KB. The saturation completion of $\mathcal{K}$, with respect to a **K**-interpretation $\hat{I}$, is the set $\mathcal{O}_{satr}(\hat{I}) = \{a \in \mathbf{KA}(\mathcal{K}) \cup \{\bot\} \mid OB_{\mathcal{O},\hat{I}} \models a\}$.

In the following lemma, we provide alternative characterizations of saturation to provide further insight into its relevance and establish a relationship between **K**-interpretation s and MKNF interpretations. For instance, characterization 3 below implies that all **K**-interpretation s induced by MKNF models are saturated.

Lemma 1
The following are equivalent for a **K***-interpretation* $\hat{I}$.

1. $\hat{I}$ *is saturated.*
2. $\hat{I}$ *extends to an MKNF interpretation* M *which induces* $\hat{I}$.
3. $\hat{I}$ *is induced by some MKNF interpretation* M *such that* $M \models_{\text{MKNF}} \mathbf{K}\pi(\mathcal{O})$.
4. $\hat{I} \models \mathcal{O}_{satr}(\hat{I})$.
5. $OB_{\mathcal{O},\hat{I}} \not\models a$ *for every atom* $a \in (\mathbf{KA}(\mathcal{K}) \cup \{\bot\}) \setminus \hat{I}$.

The relationship between saturated **K**-interpretation s which satisfy the rule completion and MKNF interpretations is described by the following proposition.

Proposition 1
For any **K***-interpretation* $\hat{I}$ *of an HMKNF-KB* $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, $\hat{I} \models \mathcal{P}_{rule} \wedge \mathcal{O}_{satr}(\hat{I})$, *if and only if, there exists an MKNF interpretation* M *such that* M *induces* $\hat{I}$ *and* $M \models_{\text{MKNF}} \pi(\mathcal{K})$.

Unlike the rule completion, the support completion of Lee and Lifschitz (2003) is not immediately adaptable to HMKNF-KBs. This is because their notion of support is based on the idea that atoms must be implied by rules, but applying this idea directly to

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

HMKNF-KBS would be misguided as they combine both rules-based and ontological reasoning. We instead define two different notions of support for an atom a occurring within a **K**-interpretation $\hat{I}$ of an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$. Firstly, a is supported by a rule $r \in \mathcal{P}$, if $\hat{I} \models r$ while $\hat{I} \setminus \{a\} \not\models r$. Secondly, a is supported via the ontology $\mathcal{O}$ if $OB_{\mathcal{O},\hat{I}\setminus\{a\}} \models a$.

We can now specify a formula which ensures that all atoms within a satisfying **K**-interpretation are supported by either a rule or the ontology.

Definition 7
The support completion of an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, w.r.t. a **K**-interpretation $\hat{I}$, is the set of formulas $\mathcal{K}_{sup}(\hat{I}) = \{\phi(a) \mid a \in \mathbf{KA}(\mathcal{K}), OB_{\mathcal{O},\hat{I}\setminus\{a\}} \not\models a\}$, where

$$\phi(a) = a \supset \bigvee_{r \in \mathcal{P}\ \&\ a \in head(r)} (Body(r) \wedge \bigwedge_{p \in head(r)\setminus\{a\}} \neg p).$$

When we combine the rule completion, saturation completion, and support completion, we obtain a formula which determines whether a **K**-interpretation of a tight HMKNF-KB is induced by some MKNF model.

Definition 8
The completion of an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ with respect to a **K**-interpretation $\hat{I}$, is $\mathcal{K}_{comp}(\hat{I}) = \mathcal{P}_{rule} \wedge \mathcal{O}_{satr}(\hat{I}) \wedge \mathcal{K}_{sup}(\hat{I})$.

Theorem 1
For any **K***-interpretation* $\hat{I}$ *of a tight HMKNF-KB* $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, $\hat{I} \models \mathcal{K}_{comp}(\hat{I})$, *if and only if,* $\mathcal{K}$ *has an MKNF model* M *such that* M *induces* $\hat{I}$.

4.2 Loop formulas

The loop formulas of Lee and Lifschitz (2003) generalize the support completion and allow for the assumption of tightness to be dropped by ensuring that each set of atoms forming a loop has support. We generalize our earlier notion of support to any subset L, of a **K**-interpretation $\hat{I}$ for an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ as follows. A rule $r \in \mathcal{P}$ supports L if $\hat{I} \models r$ but $\hat{I} \setminus L \not\models r$, whereas L is supported by the ontology $\mathcal{O}$ if $OB_{\mathcal{O},\hat{I}\setminus L} \models \bigvee L$. Thus, the conversion of loop formulas to the case of HMKNF-KBs, is similar to that for the support completion, instead of requiring that each atom within a **K**-interpretation is supported, we require that each subset forming a loop within $G(\mathcal{K})$ is supported by either a rule or the ontology.

Definition 9
The loop formulas of an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ with respect to a **K**-interpretation $\hat{I}$, is the set of formulas $\mathcal{K}_{loop}(\hat{I}) = \{\psi(L) \mid L \in Loops(\mathcal{K}), OB_{\mathcal{O},\hat{I}\setminus L} \not\models \bigvee L\}$, where

$$\psi(L) = \bigvee L \supset \Big(\bigvee_{\substack{r \in \mathcal{P} \\ head(r) \cap L \neq \emptyset \\ body^+(r) \cap L = \emptyset}} (Body(r) \wedge \bigwedge_{a \in head(r)\setminus L} \neg a) \Big).$$

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

We combine the completion and loop formulas to determine whether a **K**-interpretation of an arbitrary HMKNF-KB is induced by any MKNF model.

Theorem 2
For any **K***-interpretation* $\hat{I}$*, of an HMKNF-KB* $\mathcal{K} = (\mathcal{P}, \mathcal{O})$*,* $\hat{I} \models \mathcal{K}_{comp}(\hat{I}) \wedge \mathcal{K}_{loop}(\hat{I})$*, if and only if,* $\mathcal{K}$ *has an MKNF model* M *such that* M *induces* $\hat{I}$*.*

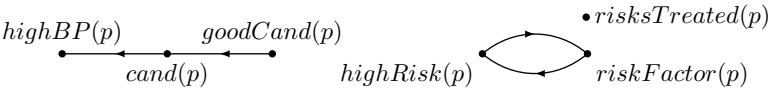
Proof Sketch.
($\Rightarrow$) It can be shown that whenever $\hat{I} \models \mathcal{P}_{rule} \wedge \mathcal{O}_{satr}(\hat{I})$, it extends to an MKNF interpretation M such that $M \models_{\text{MKNF}} \pi(\mathcal{K})$. Note that M also induces $\hat{I}$ in this case. We must still show that there is no $M' \supset M$, such that $\forall I' \in M', (I', M', M) \models \pi(\mathcal{K})$. Therefore, we investigate the **K**-interpretation $\hat{I}'$, for which such an M' induces. It can be shown that $(\hat{I} \setminus \hat{I}') \neq \emptyset$. Thus we consider whether there is an atom $p \in (\hat{I} \setminus \hat{I}')$, such that p's truth is implied via a rule or the ontology under $\hat{I}'$. We take G to be the subgraph of $G(\mathcal{K})$, containing only atoms within $(\hat{I} \setminus \hat{I}')$ which are reachable from an atom $g \in (\hat{I} \setminus \hat{I}')$. It is always possible to select an atom g, such that either G is acyclic or G contains only atoms from a single loop L. In the former case there is some atom p, which has no outgoing edges in G. The fact that $\hat{I} \models \mathcal{K}_{comp}(\hat{I})$ can then be shown to imply that p has a form of support within $\hat{I}'$. In the later case, all atoms in L only have outgoing edges in G to other atoms in L. The fact that $\hat{I} \models \mathcal{K}_{loop}(\hat{I})$ can be shown to imply that some atom $p \in L$ has a form of support outside L and within $\hat{I}'$. In both cases this is sufficient to show that $\forall I' \in M', (I', M', M) \not\models \pi(\mathcal{K})$, and thereby show that M is a model of $\mathcal{K}$.
($\Leftarrow$) This direction is relatively straightforward. It primarily relies on showing how M being a model of $\mathcal{K}$ imposes restrictions on the **K**-interpretation it induces. □

Example 1
Consider the following HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, representing whether a person p, is a good candidate for a blood-pressure medication:

$$\mathcal{P} = \left\{ \begin{array}{ll} r_1 = goodCand(p) \leftarrow (cand(p), \neg highRisk(p)). & r_2 = highBP(p). \\ r_3 = highRisk(p) \leftarrow (riskFactor(p), \neg risksTreated(p)). & \end{array} \right\}$$

and $\pi(\mathcal{O}) = \{\forall x, (highBP(x) \supset cand(x)) \wedge (highRisk(x) \supset riskFactor(x))\}$.

The ontology states that any person with high blood-pressure is a candidate, and that anyone who is high-risk for the drug has a risk factor. The rule base states that p is a good candidate for the drug if they are a non-high-risk candidate, that they are high risk if they have a risk factor they have not been treated for, and that they have high blood-pressure. Below is the dependency graph $G(\mathcal{K})$.



The following are selected **K**-interpretation s for $\mathcal{K}$, of which only $\hat{I}_4$ is induced by any MKNF model: $\hat{I}_1 = \{highBP(p)\}$, $\hat{I}_2 = \{highBP(p),\ cand(p), goodCand(p), risksTreated(p)\}$, $\hat{I}_3 = \{highBP(p),\ cand(p),\ highRisk(p),\ riskFactor(p)\}$, and $\hat{I}_4 = \{goodCand(p),\ cand(p),\ highBP(p)\}$.

Clearly, $cand(p)$ is entailed by the ontology given $\hat{I}_1$, so $\hat{I}_1$ fails to satisfy $\mathcal{O}_{satr}(\hat{I}_1)$. For $\hat{I}_2$, it is clear that $risksTreated(p)$ has no form of support, therefore it fails to satisfy

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

$\mathcal{K}_{sup}(\hat{I}_2)$. Finally, from examination of $G(\mathcal{K})$, $\{highRisk(p), riskFactor(p)\}$ forms a loop $L \in Loops(\mathcal{K})$. As each atom is only supported by the other $\hat{I}_3$ does not satisfy $\mathcal{K}_{loop}(\hat{I}_3)$.

In contrast to the other **K**-interpretation s, $\hat{I}_4$ satisfies $\mathcal{K}_{comp}(\hat{I}_4)$ and $\mathcal{K}_{loop}(\hat{I}_4)$. It avoids the problem with $\hat{I}_1$ as $\hat{I}_4 \models cand(p)$, the one with $\hat{I}_2$ as $\hat{I}_4 \not\models risksTreated(p)$, and the one with $\hat{I}_3$ as $\hat{I}_4 \not\models \bigvee L$. This is consistent with the fact that it is the only **K**-interpretation induced by an MKNF model.

5 Nogoods

In what follows, we present sets of nogoods indirectly capturing the constraints induced by the completion and loop formulas of the previous section. Total assignments of these nogoods directly correspond with **K**-interpretation s, and their solutions to those induced by MKNF models. True atoms within an assignment reflect those evaluated as true under the corresponding **K**-interpretation, and similarly for false atoms. Through this relationship the nogoods characterize MKNF models. As such, conflict-driven approaches can be built on generating a subset of these nogoods.

In all definitions of this section, we assume a given HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$.

5.1 Completion nogoods

5.1.1 Rule nogoods

For expressing that the body of a rule r is satisfied, Gebser *et al.* (2013) use sets of literals of the form $\beta(r) = \{\mathbf{T}p \mid p \in body^+(r)\} \cup \{\mathbf{F}p \mid p \in body^-(r)\}$. These are treated as composite variables with an intrinsic meaning. The literal $\mathbf{T}\beta(r)$ represents $Body(r)$ being satisfied whereas $\mathbf{F}\beta(r)$ represents its unsatisfaction. This meaning is enforced within solutions by a set of *conjunction nogoods* to be introduced shortly.

Directly following Gebser *et al.* (2013), we define a set of *rule nogoods* corresponding to the rule completion, which in our case ensures that the rule base is satisfied.

Definition 10
The rule nogood for any $r \in \mathcal{P}$, is defined as: $\phi_{\mathcal{P}}(r) = \{\mathbf{F}p_1, \ldots, \mathbf{F}p_t, \mathbf{T}\beta(r) \mid head(r) = \{p_1, \ldots, p_t\}\}$. The rule nogoods of $\mathcal{K}$ are $\Phi_{\mathcal{P}} = \{\phi_{\mathcal{P}}(r) \mid r \in \mathcal{P}\}$.

5.1.2 Saturation nogoods

To represent whether an atom p is supported by the ontology, we use the variable $\beta_{\mathcal{O}}(p)$. Within an assignment, $\mathbf{T}\beta_{\mathcal{O}}(p)$ represents that p is supported via $\mathcal{O}$, and $\mathbf{F}\beta_{\mathcal{O}}(p)$ represents that it is not. This is enforced within solutions by a set of *entailment nogoods* to be introduced shortly.

To express that a solution must be reflective of a saturated **K**-interpretation, we introduce a novel set of *saturation nogoods* corresponding to the saturation completion.

Definition 11
The saturation nogood for any atom $p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}$, is defined as: $\phi_{\mathcal{O}}(p) = \{\mathbf{F}p, \mathbf{T}\beta_{\mathcal{O}}(p)\}$. The saturation nogoods of $\mathcal{K}$ are $\Phi_{\mathcal{O}} = \{\phi_{\mathcal{O}}(p) \mid p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}\}$.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

5.1.3 Support nogoods

To reflect whether an atom p is supported by a rule r, we use literal sets of the form $\beta_{\mathcal{P}}(r,p) = \{\mathbf{T}\beta(r)\} \cup \{\mathbf{F}q \mid q \in head(r) \setminus \{p\}\}$, as is done in Gebser *et al.* (2013).[3] The truth of literals based on these sets are also enforced by conjunction nogoods.

We introduce a novel set of *support nogoods* corresponding to the support completion, to prevent cases where there is no support from a rule or the ontology for some true atom within an assignment.

Definition 12
The support nogood for any atom $p \in \mathbf{KA}(\mathcal{K})$, is defined as

$$\psi_{\mathcal{K}}(p) = \{\mathbf{T}p\} \cup \{\mathbf{F}\beta_{\mathcal{P}}(r,p) \mid r \in \mathcal{P}, p \in head(r)\} \cup \{\mathbf{F}\beta_{\mathcal{O}}(p) \mid \text{if } p \in \mathbf{KA}(\mathcal{O})\}.$$

The support nogoods of $\mathcal{K}$, are $\Psi_{\mathcal{K}} = \{\psi_{\mathcal{K}}(p) \mid p \in \mathbf{KA}(\mathcal{K})\}$.

5.1.4 Conjunction nogoods

As aforementioned, conjunction nogoods are required to ensure that composite variables consisting of other literals are assigned correctly within solutions. Here, we directly follow Gebser *et al.* (2013).

Definition 13
The conjunction nogoods for a set of literals β are $\gamma_{\mathcal{P}}(\beta) = \{\{\mathbf{F}\beta\} \cup \beta\} \cup \{\{\mathbf{T}\beta, \overline{\sigma}\} \mid \sigma \in \beta\}$. The conjunction nogoods of $\mathcal{K}$ are $\Gamma_{\mathcal{P}} = \bigcup_{\beta \in \{\beta(r) \mid r \in \mathcal{P}\} \cup \{\beta_{\mathcal{P}}(r,p) \mid r \in \mathcal{P}, p \in head(r)\}} \gamma_{\mathcal{P}}(\beta)$.

5.1.5 Entailment nogoods

To ensure the literals of the form $\mathbf{T}\beta_{\mathcal{O}}(p)$ or $\mathbf{F}\beta_{\mathcal{O}}(p)$ appear within assignments in accordance with whether p is supported via the ontology, we introduce *entailment nogoods*.

Definition 14
Let $\beta_{\mathcal{O}}(p)$ be a variable associated with an atom $p \in \mathbf{KA}(\mathcal{O})$. A positive entailment nogood, of an atom $p \in \mathbf{KA}(\mathcal{O})$ and set $S \subseteq \mathbf{KA}(\mathcal{O})$, is defined as $\gamma^{+}_{\mathcal{O}}(p,S) = \{\mathbf{F}\beta_{\mathcal{O}}(p)\} \cup \{\mathbf{T}s \mid s \in S\}$, whereas a negative entailment nogood, is defined as $\gamma^{-}_{\mathcal{O}}(p,S) = \{\mathbf{T}\beta_{\mathcal{O}}(p)\} \cup \{\mathbf{F}s \mid s \in S\}$. The entailment nogoods of $\mathcal{K}$ are

$$\begin{aligned}\Gamma_{\mathcal{O}} = &\{\gamma^{+}_{\mathcal{O}}(p,S) \mid p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}, S \subseteq \mathbf{KA}(\mathcal{O}), OB_{\mathcal{O},S\setminus\{p\}} \models p\} \\ &\cup \{\gamma^{-}_{\mathcal{O}}(p,S) \mid p \in \mathbf{KA}(\mathcal{O}), S \subseteq \mathbf{KA}(\mathcal{O}), OB_{\mathcal{O},\mathbf{KA}(\mathcal{O})\setminus(S\cup\{p\})} \not\models p\}.\end{aligned}$$

For some atom $p \in \mathbf{KA}(\mathcal{O})$ and set of atoms $S \subseteq \mathbf{KA}(\mathcal{O})$, the purpose of a positive entailment nogood $\gamma^{+}_{\mathcal{O}}(p,S)$ is to indicate that p is supported via $\mathcal{O}$ by true atoms within an assignment, given it has all atoms in S as true. Conversely, the purpose of a negative entailment nogood $\gamma^{-}_{\mathcal{O}}(p,S)$ is to indicate that p has no way of being supported via $\mathcal{O}$ by true atoms within an assignment, given it has all atoms in S as false.

[3] In Gebser *et al.* (2013) literal sets denoted as $\beta_{\mathcal{P}}(r,p)$, are defined such that they are substituted for by other literal sets of the form $\beta(r)$, whenever they effectively coincide. This is a relevant consideration for implementation efficiency; however it is omitted here for simplicity.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

5.1.6 Relation to MKNF models

So far we have discussed the relationship between assignments and **K***-interpretation s using general language. The exact correspondence is given by the following definition.*

Definition 15
Let $\hat{I}$ a **K**-interpretation for $\mathcal{K}$. The induced assignment of $\hat{I}$ for $\mathcal{K}$ is

$$\begin{aligned}
A^{\hat{I}}_{\mathcal{K}} &= \{\mathbf{T}p \mid p \in \hat{I}\} \cup \{\mathbf{F}p \mid p \in \mathbf{KA}(\mathcal{K}) \setminus \hat{I}\} \cup \{\mathbf{F}\bot\} \\
&\cup \{\mathbf{T}\beta(r) \mid r \in \mathcal{P}, \hat{I} \models Body(r)\} \cup \{\mathbf{F}\beta(r) \mid r \in \mathcal{P}, \hat{I} \not\models Body(r)\} \\
&\cup \{\mathbf{T}\beta_{\mathcal{P}}(r,p) \mid r \in \mathcal{P}, \hat{I} \models Body(r), head(r) \cap (\hat{I} \cup \{p\}) = \{p\}\} \\
&\cup \{\mathbf{F}\beta_{\mathcal{P}}(r,p) \mid r \in \mathcal{P}, \hat{I} \not\models Body(r), p \in head(r)\} \\
&\cup \{\mathbf{F}\beta_{\mathcal{P}}(r,p) \mid r \in \mathcal{P}, p \in head(r), head(r) \cap \hat{I} \not\subseteq \{p\}\} \\
&\cup \{\mathbf{T}\beta_{\mathcal{O}}(p) \mid p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}, OB_{\mathcal{O},\hat{I}\setminus\{p\}} \models p\} \\
&\cup \{\mathbf{F}\beta_{\mathcal{O}}(p) \mid p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}, OB_{\mathcal{O},\hat{I}\setminus\{p\}} \not\models p\}.
\end{aligned}$$

The nogoods we have defined thus far make up our completion nogoods.

Definition 16
The completion nogoods of $\pi(\mathcal{K})$ are

$$\Delta_{\mathcal{K}} = \Phi_{\mathcal{P}} \cup \Phi_{\mathcal{O}} \cup \Psi_{\mathcal{K}} \cup \Gamma_{\mathcal{P}} \cup \Gamma_{\mathcal{O}}.$$

We have designed the induced assignment of any **K**-interpretation to be a total assignment for the completion nogoods, and the nogoods themselves to occur in and only in assignments induced by **K**-interpretation s which do not satisfy their completion formulas. In showing this to be the case, we obtain the following result.

Theorem 3
Let $\hat{I}$ *be a* **K***-interpretation for an HMKNF-KB* $\mathcal{K} = (\mathcal{P}, \mathcal{O})$*, then we have that* $\hat{I} \models \mathcal{K}_{comp}(\hat{I})$ *if and only if* $A^{\hat{I}}_{\mathcal{K}}$ *is a solution to* $\Delta_{\mathcal{K}} \cup \{\mathbf{T}\bot\}$.

The above theorem allows us to conclude that the **K***-interpretation s that induce solutions to the completion nogoods are the ones which satisfy their completion formulas. It naturally follows from Theorem 1, that for tight HMKNF-KBs these are also the* **K***-interpretation s induced by MKNF models.*

5.2 Loop nogoods

Loop nogoods are intended to parallel the loop formulas of Section 4.2 in ensuring non-circular support. Directly following Gebser *et al.* (2013), we denote the external program supports of a set of atoms L, by $\mathcal{E}_{\mathcal{P}}(L) = \{r \in \mathcal{P} \mid head(r) \cap L = \emptyset, body^{+}(r) \cap L \neq \emptyset\}$. Moreover, $\rho(r, L) = \{\mathbf{F}\beta(r)\} \cup \{\mathbf{T}p \mid p \in head(r) \setminus L\}$, collects all literals satisfying a rule r, regardless of whether any atom from L is true.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Definition 17
The loop nogoods for any $L \subseteq \mathbf{KA}(\mathcal{K})$ and $S \subseteq \mathbf{KA}(\mathcal{O})$ are defined as

$$\lambda_{\mathcal{K}}(L,S) = \{\{\mathbf{T}p, \sigma_1, \ldots, \sigma_k, \mathbf{F}s_1, \ldots, \mathbf{F}s_n\} \mid p \in L, \mathcal{E}_{\mathcal{P}}(L) = \{r_1, \ldots, r_k\},$$
$$\sigma_1 \in \rho(r_1, L), \ldots, \sigma_k \in \rho(r_k, L), S = \{s_1, \ldots, s_n\}\}.$$

The loop nogoods of $\mathcal{K}$ are

$$\Lambda_{\mathcal{K}} = \bigcup_{\substack{L \in Loops(\mathcal{K}), S \subseteq \mathbf{KA}(\mathcal{O}), \\ L \cap S = \emptyset, OB_{\mathcal{O}, \mathbf{KA}(\mathcal{O}) \setminus (S \cup L)} \not\models_{\vee} L}} \lambda_{\mathcal{K}}(L,S).$$

Above, $\Lambda_{\mathcal{K}}$ includes a nogood for each loop, L, and subset of $\mathbf{KA}(\mathcal{O})$, S, for which L cannot be supported via $\mathcal{O}$ whenever all atoms in S are false. Each nogood $\lambda(L,S)$ requires that the loop L must be supported by a rule if all atoms in S are false. We can show that loop nogoods occur only within assignments induced by **K**-interpretation s which do not satisfy their loop formulas, to obtain the following result.

Theorem 4
Let $\hat{I}$ be a **K***-interpretation for an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$, then we have that $\hat{I} \models \mathcal{K}_{comp}(\hat{I}) \wedge \mathcal{K}_{loop}(\hat{I})$ if and only if $A^{\hat{I}}_{\mathcal{K}}$ is a solution to $\Delta_{\mathcal{K}} \cup \Lambda_{\mathcal{K}} \cup \{\mathbf{T}\bot\}$.*

Proof Sketch.
($\Rightarrow$) It can be shown relatively easily that $A^{\hat{I}}_{\mathcal{K}}$, the assignment $\hat{I}$ induces, is a total assignment for the nogoods $\Delta_{\mathcal{K}} \cup \Lambda_{\mathcal{K}} \cup \{\bot\}$.

The other condition which must be shown is that no nogood from $\Delta_{\mathcal{K}} \cup \Lambda_{\mathcal{K}} \cup \{\bot\}$ occurs within $A^{\hat{I}}_{\mathcal{K}}$. This is done by considering each of $\Phi_{\mathcal{P}}$, $\Phi_{\mathcal{O}}$, $\Psi_{\mathcal{K}}$, $\Gamma_{\mathcal{O}}$, $\Gamma_{\mathcal{P}}$, and $\Lambda_{\mathcal{K}}$ individually. For each of nogood set ∇ we make the assumption that $\delta \in \nabla$ is a subset of $A^{\hat{I}}_{\mathcal{K}}$, then show that the restrictions this imposes on $\hat{I}$ also imply that for some literal $\sigma \in \delta$, $\overline{\sigma} \in A^{\hat{I}}_{\mathcal{K}}$. This contradicts that $A^{\hat{I}}_{\mathcal{K}}$ is a total assignment and in doing so proves no such nogood can exist.

For example: Assume there is a nogood $\gamma^{+}_{\mathcal{O}}(p,S) \in \Gamma_{\mathcal{O}}$ for $p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}$, $S \subseteq \mathbf{KA}(\mathcal{O})$ such that $\gamma^{+}_{\mathcal{O}}(p,S) \subseteq A^{\hat{I}}_{\mathcal{K}}$. Then by the fact that $\gamma^{+}_{\mathcal{O}}(p,S) \in \Gamma_{\mathcal{O}}$, $OB_{\mathcal{O},S\setminus\{p\}} \models p$, moreover since $\gamma^{+}_{\mathcal{O}}(p,S) = \{\mathbf{F}\beta_{\mathcal{O}}(p)\} \cup \{\mathbf{T}s \mid s \in S\}$ and $\gamma^{+}_{\mathcal{O}}(p,S) \subseteq A^{\hat{I}}_{\mathcal{K}}$, $S \subseteq A^{\hat{I}}_{\mathcal{K}}$. Therefore $OB_{\mathcal{O},\hat{I}\setminus\{p\}} \models p$ as well, so $\mathbf{T}\beta_{\mathcal{O}}(p) \in A^{\hat{I}}_{\mathcal{K}}$. This contradicts the fact that $A^{\hat{I}}_{\mathcal{K}}$ is a total assignment proving that no such $\gamma^{+}_{\mathcal{O}}(p,S)$ exists. The inverse logic can be applied to show that no nogood $\gamma^{-}_{\mathcal{O}}(p,S) \in \Gamma_{\mathcal{O}}$ for $p \in \mathbf{KA}(\mathcal{O})$, $S \subseteq \mathbf{KA}(\mathcal{O})$ can occur in $A^{\hat{I}}_{\mathcal{K}}$.

($\Leftarrow$) We wish to show that $\hat{I}$ must satisfy all of $\mathcal{P}_{rule}$, $\mathcal{O}_{satr}(\hat{I})$, $\mathcal{K}_{sup}(\hat{I})$, and $\mathcal{K}_{loop}(\hat{I})$. To do so we consider the case where $\hat{I}$ fails to satisfy each of them individually, and show that it would require a nogood to exist within $A^{\hat{I}}_{\mathcal{K}}$, violating the initial assumptions.

For example: Assume that $\hat{I} \not\models \mathcal{K}_{sup}(\hat{I})$. Clearly there is some atom $p \in \hat{I}$ such that $OB_{\mathcal{O},\hat{I}\setminus\{p\}} \not\models p$ for which for all $r \in \mathcal{P}$ where $p \in head(r)$ $\hat{I} \not\models Body(r)$ or $\exists a \in head(r) \setminus \{p\}$ such that $a \in \hat{I}$. From $p \in \hat{I}$, $\mathbf{T}p \in A^{\hat{I}}_{\mathcal{K}}$. From $OB_{\mathcal{O},\hat{I}\setminus\{p\}} \not\models p$, either $p \in (\mathbf{KA}(\mathcal{K}) \setminus \mathbf{KA}(\mathcal{O}))$ or $\mathbf{F}\beta_{\mathcal{O}} \in A^{\hat{I}}_{\mathcal{K}}$. Finally, since for all $r \in \mathcal{P}$ where $p \in head(r)$, $\hat{I} \not\models Body(r)$ or $\exists a \in head(r) \setminus \{p\}$ such that $a \in \hat{I}$, $\mathbf{F}\beta_{\mathcal{P}}(r,p) \in A^{\hat{I}}_{\mathcal{K}}$. It follows that $\psi_{\mathcal{K}}(p) \subseteq A^{\hat{I}}_{\mathcal{K}}$, since

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

$\psi_{\mathcal{K}}(p) \in \Psi_{\mathcal{K}}$ this violates the assumption that $A^{\hat{I}}_{\mathcal{K}}$ is a solution for $\Delta_{\mathcal{K}}$. Therefore $\hat{I} \models \mathcal{K}_{sup}(\hat{I})$. □

This theorem, together with Theorem 2, implies that, for any HMKNF-KB, the K-interpretations that induce solutions to the completion and loop nogoods are precisely those that are induced by MKNF models.

Example 2
Consider the following HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ where only the **K**-interpretation $\hat{I} = \{a, b\}$ is induced by any MKNF model.

$$\mathcal{P} = \left\{ \begin{array}{ll} r_1 = a. & r_2 = a \vee d. \\ r_3 = f \leftarrow d. & r_4 = e \leftarrow f. \end{array} \right\}$$

and $\pi(\mathcal{O}) = \{(a \supset b) \wedge (c \supset d) \wedge (c \supset e) \wedge (e \supset f)\}$. $G(\mathcal{K})$ is shown below.

We will show that $A^{\hat{I}}_{\mathcal{K}}$, the assignment $\hat{I}$ induces, is the only solution to our nogoods. To do so we take A to be an arbitrary solution and prove that it contains every literal within $A^{\hat{I}}_{\mathcal{K}}$. Some steps are omitted for conciseness, for instance we leave it as an exercise to the reader to show that:

$$\begin{aligned} A^{\hat{I}}_{\mathcal{K}} = \{ & \mathbf{T}a, \mathbf{T}b, \mathbf{F}c, \mathbf{F}d, \mathbf{F}e, \mathbf{F}f, \\ & \mathbf{T}\beta(r_1) = \mathbf{T}\beta(r_2) = \mathbf{T}\emptyset, \mathbf{F}\beta(r_3) = \mathbf{F}\{d\}, \mathbf{F}\beta(r_4) = \mathbf{F}\{f\}, \\ & \mathbf{T}\beta(r_1, a) = \mathbf{T}\{\mathbf{T}\beta(r_1)\}, \mathbf{T}\beta(r_2, a) = \mathbf{F}\{\mathbf{T}\beta(r_2), \mathbf{F}d\}, \\ & \mathbf{F}\beta(r_2, d) = \mathbf{F}\{\mathbf{T}\beta(r_2), \mathbf{F}a\}, \mathbf{F}\beta(r_3, f) = \mathbf{F}\{\mathbf{T}\beta(r_3)\}, \mathbf{F}\beta(r_4, f), = \mathbf{F}\{\mathbf{T}\beta(r_4)\}, \\ & \mathbf{F}\beta_{\mathcal{O}}(a), \mathbf{T}\beta_{\mathcal{O}}(b), \mathbf{F}\beta_{\mathcal{O}}(c), \mathbf{F}\beta_{\mathcal{O}}(d), \mathbf{F}\beta_{\mathcal{O}}(e), \mathbf{F}\beta_{\mathcal{O}}(f)\} \end{aligned}$$

Firstly, we show that $\mathbf{T}a$ and $\mathbf{T}b$ must be in A. Consider the conjunction nogood $\gamma(r_1) = \{\{\mathbf{F}\beta(r_1)\} \cup \beta(r_1)\} \cup \{\{\mathbf{T}\beta(r_1), \overline{\sigma}\} \mid \sigma \in \beta(r_1)\} \in \Gamma_{\mathcal{P}}$. As the body of r_1 is empty $\beta(r_1) = \emptyset$, thus $\gamma(r_1) = \{\mathbf{F}\emptyset\}$, and so $\mathbf{T}\emptyset \in A$. Combined with the rule nogood $\phi_{\mathcal{P}}(r_1) = \{\mathbf{F}a, \mathbf{T}\beta(r_1)\} \in \Phi_{\mathcal{P}}$, this also means that $\mathbf{T}a \in A$. Due to the fact that $OB_{\mathcal{O},a} \models b$, the positive entailment nogood $\gamma^{+}_{\mathcal{O}}(b, \{a\})$ occurs within $\Gamma_{\mathcal{O}}$. As $\gamma^{+}_{\mathcal{O}}(b, \{a\}) = \{\mathbf{F}\beta_{\mathcal{O}}(b), \mathbf{T}a\}$ it follows that $\mathbf{T}\beta_{\mathcal{O}}(b) \in A$. Consequently, due to the saturation nogood $\phi_{\mathcal{O}}(b) = \{\mathbf{F}b, \mathbf{T}\beta_{\mathcal{O}}(b)\} \in \Phi_{\mathcal{O}}$, $\mathbf{T}b \in A$ as well.

It is simple to show that $\mathbf{F}c \in A$, as such we focus on the more interesting case of $\mathbf{F}d$. Turn your attention to the conjunction nogoods $\gamma_{\mathcal{P}}(\beta_{\mathcal{P}}(d, r_2)) \subseteq \Gamma_{\mathcal{P}}$, which are $\{\mathbf{F}\beta_{\mathcal{P}}(d, r_2), \mathbf{T}\beta(r_2), \mathbf{F}a\}$, $\{\mathbf{T}\beta_{\mathcal{P}}(d, r_2), \mathbf{F}\beta(r_2)\}$, and $\{\mathbf{T}\beta_{\mathcal{P}}(d, r_2), \mathbf{T}a\}$. The last one shows that $\mathbf{F}\beta_{\mathcal{P}}(d, r_2) \in A$. Also note that unless c is true d cannot be supported by the ontology – $OB_{\mathcal{O},\mathbf{KA}(\mathcal{O})\setminus\{c,d\}} \not\models d$ – therefore the negative entailment nogood $\gamma^{-}_{\mathcal{O}}(d, \{c\}) = \{\mathbf{T}\beta_{\mathcal{O}}(d), \mathbf{F}c\}$ is in $\Gamma_{\mathcal{K}}$ and thereby $\mathbf{F}\beta_{\mathcal{O}}(d) \in A$. Now consider the support nogood $\psi_{\mathcal{K}}(d) = \{\mathbf{T}d, \mathbf{F}\beta_{\mathcal{P}}(d, r_2), \mathbf{F}\beta_{\mathcal{O}}(d)\} \in \Psi_{\mathcal{K}}$, to see that $\mathbf{F}d \in A$.

Finally we show that $\mathbf{F}e$ and $\mathbf{F}f$ are in A. Clearly the set of both atoms is a loop $\{e, f\} \in \mathit{Loops}(\mathcal{K})$, and since neither atom can be supported by the ontology unless c, e, or

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

f is true – $OB_{\mathcal{O},\mathbf{KA}(\mathcal{O})\setminus\{c,e,f\}} \not\models \bigvee\{e,f\}$ – the loop nogoods $\lambda(\{e,f\},\{c\})$ are in $\Lambda_{\mathcal{K}}$. The external supports for $\{e,f\}$ are simply $\mathcal{E}(\{e,f\}) = \{r_3\}$, and $\rho(r_3,\{e,f\}) = \{\mathbf{F}\beta(r_3)\}$, therefore $\lambda(\{e,f\},\{c\}) = \{\{\mathbf{T}e,\mathbf{F}\beta(r_3)\},\{\mathbf{T}f,\mathbf{F}\beta(r_3)\}\}$. From the fact that $\mathbf{F}d \in A$, it is easy to show that $\mathbf{F}\{d\} = \mathbf{F}\beta(r_3) \in A$. Consequently $\mathbf{F}e$ and $\mathbf{F}f$ both must be in A.

We have that $\{\mathbf{T}a,\mathbf{T}b,\mathbf{F}c,\mathbf{F}d,\mathbf{F}e,\mathbf{F}f\} \subseteq A$ which is the most essential aspect of $A^{\hat{I}}_{\mathcal{K}}$'s relationship to $\hat{I}$, the rest is left as an exercise.

6 Conflict-Driven solving

A conflict-driven solver which determines **K**-interpretation s induced by MKNF models of an HMKNF-KB, can be built based on the completion and loop nogoods of Section 5, following the same general approach of Gebser *et al.* (2012). The following is a sketch of such a solver. Our goal is to provide an overview which is open to further specification.

6.1 Main procedures

Due to the similarity of their formulation to that of Gebser *et al.* (2012) the details of the main conflict-driven procedures will be explained only at a high level, with some comments on specific differences.

Algorithm 1 CDNL is primarily responsible for keeping track of an assignment representing a partial candidate solution, through a tree-like search procedure. Additionally, it tracks a set of nogoods which are a subset of those introduced in Section 5, and a decision level indicating the current depth of the search tree. It operates by calling the reasoning procedure *NogoodProp* to deterministically extend the assignment, and track additional nogoods which prove helpful in doing so. Following this it takes one of three actions. If the current assignment is incomplete but compatible with the current nogood, an arbitrary decision literal is added to the assignment increasing the decision level. If the current assignment conflicts with the current nogoods, the search backtracks to a lower decision level or returns "no model" if already at the lowest decision level. Finally, if the assignment is total and compatible with all tracked nogoods, a model-checking oracle is consulted. If the assignment is verified as a solution the **K**-interpretation that induces it is returned. Otherwise, the trivial nogood – the full invalid assignment – is added to the tracked set enabling backtracking.

The significant changes from the main algorithm of Gebser *et al.* (2012) are: the different initial set of nogoods, and the final model check ultimately required for the soundness of the algorithm. This check can only be avoided if we assume we can always determine a nogood whenever one is a subset of a total assignment, however this requires substantial procedures for determining loop nogoods.

Algorithm 2 NogoodProp is the core reasoning procedure. A unit-resulting nogood for an assignment contains one literal which is not within the assignment. We use the term unit-propagation to refer to the process of adding the complement of this literal to the assignment which the nogood is unit-resulting for. *NogoodProp* unit-propagates the

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Algorithm 1. *CDNL*

let Δ be $\Phi_{\mathcal{P}} \cup \Phi_{\mathcal{O}} \cup \Psi_{\mathcal{K}} \cup \{\bot\}$
$(\nabla, A, dl) \leftarrow (\emptyset, \{\mathbf{F}\bot\}, 0)$
while *True* **do**
  $(A, \nabla) \leftarrow NogoodProp(dl, \mathcal{K}, \nabla, A)$
  if $\mathcal{E} \subseteq A$ *for some* $\mathcal{E} \in \Delta \cup \nabla$ **then**
    if $dl = 0$ **then**
      return *no model*
    $(\delta, dl) \leftarrow ConfAnal(\mathcal{E}, \mathcal{K}, \nabla, A)$
    $\nabla \leftarrow \nabla \cup \{\delta\}$
    $A \leftarrow A \setminus \{\sigma \in A \mid dl < dl(\sigma)\}$
  else if $A^T \cup A^F$ *is total* **then**
    if *ModelCheck(A)* **then**
      return $A^T \cap \mathbf{KA}(\mathcal{K})$
    else
      $\nabla \leftarrow \nabla \cup A$
  else
    $\sigma_d \leftarrow Select(\mathcal{K}, \nabla, A)$
    $dl \leftarrow dl + 1$
    $dl(\sigma_d) \leftarrow dl$
    $A \leftarrow A \circ \sigma_d$

Algorithm 2. *NogoodProp*

$U \leftarrow \emptyset$
while *True* **do**
  if $\delta \subseteq A$ *for some* $\delta \in \Delta \cup \nabla$ **then**
    return (A, ∇)
  $\Sigma \leftarrow \{\delta \in \Delta \cup \nabla \mid \delta \setminus A = \{\overline{\sigma}\}, \sigma \notin A\}$
  if $\Sigma \neq \emptyset$ **then**
    let $\overline{\sigma} \in \delta \setminus A$ for some $\delta \in \Sigma$
    $dl(\sigma) \leftarrow dl$
    $A \leftarrow A \circ \sigma$
  $(\nabla, ent) \leftarrow EntNogoods(A, \mathcal{K}, \nabla)$
  if not *ent* **then**
    $U \leftarrow U \setminus A^F$
    if $U = \emptyset$ **then**
      $(U, S) \leftarrow UnfoundedSet(\mathcal{K}, A)$
    if $U = \emptyset$ **then**
      return (A, ∇)
    else
      let $p \in U, \delta \in \lambda_{\mathcal{K}}(p, U, S)$
      $\nabla \leftarrow \nabla \cup \delta$

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

current assignment based on the set of tracked nogoods, and adds additional nogoods whenever propagation reaches a fixpoint. The novel feature of this procedure is that its first resort after reaching a fixpoint is to call *EntNogoods*, which will attempt to add unit-resulting entailment nogoods to ∇. If no such nogoods can be found based on the current assignment, the procedure will instead attempt to generate a loop nogood, by calling the procedure *UnfoundedSet*. This procedure returns a set $U \in Loops(\mathcal{K})$[4] for which $U \subseteq A^T$, and $S \subseteq A^F$ such that $OB_{\mathcal{O},\mathbf{KA}(\mathcal{K})\setminus(U\cup S)} \not\models \bigvee U$. In principle, this can be a modification of existing methods for detecting unfounded sets. Eventually, it reaches a point where either a tracked nogood conflicts with the current assignment or no more unit-resulting nogoods can be discovered. At this point, the procedure returns.

6.2 Determining entailment nogoods

Algorithm 3 *EntNogoods* is the novel procedure responsible for determining unit-resulting entailment nogoods. It relies on the function *Entailment*, which takes a set $S \subseteq \mathbf{KA}(\mathcal{O})$ as an argument. It returns the set $\{\bot\}$ if $OB_{\mathcal{O},S}$ is inconsistent, and otherwise returns a set $\Omega = \{p \in \mathbf{KA}(\mathcal{O}) \mid OB_{\mathcal{O},S} \models p\} \cup \{\neg p \mid p \in \mathbf{KA}(\mathcal{O}), OB_{\mathcal{O},S} \models \neg p\}$. We let Ω^+ refer to $\{p \in \Omega \mid p \in \mathbf{KA}(\mathcal{O}) \cup \{\bot\}\}$ and Ω^- refer to $\{\neg p \in \Omega \mid p \in \mathbf{KA}(\mathcal{O})\}$. We denote the set of atoms which some $p \in \mathbf{KA}(\mathcal{O})$ has an edge to in $G(\mathcal{O})$ as $ext(p)$.

The procedure determines the minimum set of entailable information Ω in line 2, by calling *Entailment* with all atoms from $\mathbf{KA}(\mathcal{O})$ which are true in the current assignment. It then checks if this set contains the contradiction atom $\bot$ in line 3. If so it adds the appropriate nogood, and returns indicating success. Otherwise, it checks whether any atom not currently known to be true can be entailed as such in line 6. If so it adds a nogood for each such atom, allowing the atom to be unit-propagated. Then in line 9, the generated nogood is associated with the entailed atom p and the true atoms with an edge from p within $G(\mathcal{O})$, $(A^T \cap ext(p))$. Clearly, this set is sufficient to entail and therefore support p. Conversely, the nogood added in line 13 represents that the atom p cannot be true along with the true atoms of the current assignment, or else there would be a contradiction.

The final section of the algorithm aims to generate negative entailment nogoods. For each entailment atom $\beta_{\mathcal{O}}(p)$ which is not yet assigned false, we check whether p is within the set of entailed atoms O under the assumption that all atoms in $\mathbf{KA}(\mathcal{O}) \setminus (A^F \cup \{p\})$ are true, in line 15. If O contains neither p nor the contradiction atom $\bot$, then we can be certain that p cannot be supported via $\mathcal{O}$ whenever all atoms in $A^F \cap ext(\{p\})$ are false. Therefore, in line 18 we add the corresponding nogood. Finally, procedure then returns. If any nogood was added during the process, *entailed* will be true, and otherwise false.

Example 3

Below are the results of calling EntNogoods with different assignments A as input for an HMKNF-KB $\mathcal{K} = (\mathcal{P}, \mathcal{O})$ where $\mathcal{O} = ((\neg a \vee \neg b) \wedge (\neg a \vee c))$ and $\mathbf{KA}(\mathcal{O}) = \{a, b, c\}$.

[4] The condition that U be in $Loops(\mathcal{K})$ is relaxable. As the dependency graph $G(\mathcal{K})$ can contain extra edges without affecting the completeness of Theorem 2, any $S \subseteq \mathbf{KA}(\mathcal{K})$ may be added to $Loops(\mathcal{K})$.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Algorithm 3. *EntNogoods*

```
1  entailed ← False
2  Ω ← Entailment(A^T ∩ KA(O))
3  if ⊥ ∈ Ω^+ then
4  |  Δ ← Δ ∪ γ^+_O(⊥, A^T)
5  |  return (∇, True)
6  if Ω^+ \ A^T ≠ ∅ then
7  |  entailed ← True
8  |  for p ∈ Ω^+ \ A^T do
9  |  |  ∇ ← ∇ ∪ γ^+_O(p, A^T ∩ ext({p}))
10 if Ω^− \ A^F ≠ ∅ then
11 |  entailed ← True
12 |  for p ∈ Ω^− \ A^F do
13 |  |  ∇ ← ∇ ∪ γ^+_O(⊥, A^T ∪ {p})
14 for p ∈ KA(O) such that β_O(p) ∉ A^F do
15 |  O ← Entailment(KA(O) \ (A^F ∪ {p}))
16 |  if ⊥ ∉ O^+ and p ∉ O^+ then
17 |  |  entailed ← True
18 |  |  ∇ ← ∇ ∪ γ^−_O(p, A^F ∩ ext({p}))
19 return (∇, entailed)
```

A^T	A^F	Ω^+	Ω^-	O^+	p	*Added Nogood*	*entailed*	*Line*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$						*False*	*1*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$				*False*	*2*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$				*True*	*7*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$		c	$\gamma^+_\mathcal{O}(c,\{a\})$	*True*	*9*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$		b	$\gamma^+_\mathcal{O}(\bot,\{a,b\})$	*True*	*13*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$	$\{a,c\}$	b		*True*	*15*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$	$\{\bot\}$	c		*True*	*15*
$\{a\}$	$\{\beta_\mathcal{O}(a)\}$	$\{a,c\}$	$\{b\}$	$\{\bot\}$			*True*	*19*
$\{a,b\}$	$\emptyset$						*False*	*1*
$\{a,b\}$	$\emptyset$	$\{\bot\}$	$\emptyset$				*False*	*2*
$\{a,b\}$	$\emptyset$	$\{\bot\}$	$\emptyset$			$\gamma^+_\mathcal{O}(\bot,\{a,b\})$	*False*	*4*
$\{a,b\}$	$\emptyset$	$\{\bot\}$	$\emptyset$				*True*	*5*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$						*False*	*1*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$				*False*	*2*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{c\}$	b		*False*	*15*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{c\}$	b		*True*	*17*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{c\}$	b	$\gamma^-_\mathcal{O}(b,\{a\})$	*True*	*18*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{b\}$	c		*True*	*15*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{b\}$	c	$\gamma^-_\mathcal{O}(c,\{a\})$	*True*	*18*
$\emptyset$	$\{a,\beta_\mathcal{O}(a)\}$	$\emptyset$	$\emptyset$	$\{b\}$	c		*True*	*19*

Above, the leftmost two columns A^T and A^F display the true and false atoms of the input assignment respectively. The next two display the atoms entailed under minimal assumptions as true Ω^+ and false Ω^-. Following that is the set of atoms entailed as true under maximal assumptions O^+. After that is the variable p, which reflects the current value of the iterator within the active for-loop. Then, there is a column indicating which, if any, nogood was added to the tracked set in the current step. Following which is the

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Boolean variable *entailed* indicating whether any nogood has been added. Finally, the active line number is displayed on the right.

7 Related and future work

The first challenge in applying the theories of this work to the implementation of a conflict-driven solver, involves procedures for the generation and management of unit-resulting entailment and loop nogoods. Generation of entailment nogoods can follow the approach of *EntNogoods*, but requires the integration of ontology reasoners. Generating loop nogoods requires additional theoretical work, but in principle can be based on detection of unfounded sets. There must also be an investigation into the ideal policies for when to attempt generation of these nogoods, and when they should be forgotten. It is essential to keep the set of tracked nogoods small in order to reduce time spent performing unit-propagation. This is likely to be uniquely challenging for HMKNF-KBs.

The second is the practical challenge of integrating these theories with existing solvers. The Clingo system includes a Theory-enhanced ASP solving API (Gebser *et al.* 2016), allowing the user to introduce additional *theory atoms*, whose truth is regulated by user-supplied *theory nogoods*. Using this approach for a tightly integrated framework such as HMKNF-KBs is complicated by the support and loop nogoods already included by Clingo. In principle these can be reduced to a subset of our proposed nogoods by adding supporting rules, the bodies of whom are theory atoms semantically governed entirely by theory nogoods. However, this approach is non-trivial. It requires determining both which support rules to add and the additional theory nogoods required to construct a set equivalent to our proposed one.

A direct implementation using Clingo's source code is also possible but comes with its own complexity. Another option is to build on the DLVHEX system, which despite its existing integration with ontology reasoners is less straightforward to layer our framework upon. Conversely, a translation plugin that allows DLVHEX to solve HMKNF-KBs as dl-programs could provide a useful benchmark.

8 Conclusion

Our work establishes the critical foundation for the development of a native conflict-driven solver of HMKNF-KBs. We have made significant theoretical contributions including the first formulation of a dependency graph, and first adaption of a completion and loop formulas for the formalism. These advancements have enabled us to derive a set of nogoods, essential for implementing a conflict-driven solver. We have outlined the architecture of such a solver and critically examined both the potential and challenges in leveraging existing systems. Our findings have significant implications for enhancing the efficiency and practicality of reasoning with HMKNF-KBs under the stable model semantics. The immediate next steps include the implementation of a conflict-driven solver based on our theoretical framework and further refinement of the characterizations we have proposed.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press

Acknowledgements

We thank the reviewers for their constructive comments that helped improve the presentation of the paper. This research was supported in part by Alberta Innovates (under grant Alberta Innovates Advance Program 222301990) and by the Natural Sciences and Engineering Research Council of Canada (under grant NSERC RGPIN-2020-05211). For his research, the first author also received an NSERC Undergraduate Student Research Award and the second author was supported by an Alberta Innovates Graduate Student Scholarship.

Supplementary material

To view supplementary material for this article, please visit http://doi.org/10.1017/S1471068424000255.

References

Adrian, W. T., Alviano, M., Calimeri, F., Cuteri, B., Dodaro, C., Faber, W., Fuscà, D., Leone, N., Manna, M., Perri, S., Ricca, F., Veltri, P. and Zangari, J. 2018. The ASP system DLV: Advancements and applications. *KI - Künstliche Intelligenz* 32, 2-3, 177–179.

Alberti, M., Knorr, M., Gomes, A. S., Leite, J., Gonçalves, R. and Slota, M. 2012. Normative systems require hybrid knowledge bases. In *Proc. of International Conference on Autonomous Agents and Multiagent Systems*, IFAAMS, 1425–1426.

Alferes, JJ., Knorr, M. and Swift, T. 2013. Query-driven procedures for hybrid MKNF knowledge bases. *ACM Transactions on Computational Logic* 16, 2, 1–43.

Alviano, M., Dodaro, C., Leone, N. and Ricca, F. 2015. Advances in WASP. In *Proc. of LPNMR*, Springer, 40–54.

Clark, K. L. 1977.Negation as failure. In *Logic and Data Bases*. Springer, 293–322.

Eiter, T., Germano, S., Ianni, G., Kaminski, T., Redl, C., Schüller, P. and Weinzierl, A. 2018. The DLVHEX system. *KI - Künstliche Intelligenz* 32, 2-3, 187–189.

Eiter, T., Ianni, G., Schindlauer, R. and Tompits, H. 2005. Nonmonotonic description logic programs: Implementation and experiments. In *Proc. of LPAR*, Springer, 511–527.

Eiter, T., Ianni, G., Schindlauer, R. and Tompits, H. 2006. Towards efficient evaluation of hex programs. In *Proc. of the International Workshop on NMR*, 40–46.

Eiter, T. and Kaminski, T. 2021. Pruning external minimality checking for answer set programs using semantic dependencies. *Artificial Intelligence* 290, 103402.

Eiter, T. and Šimkus, M. 2015. Linking open-world knowledge bases using nonmonotonic rules. In *Proc. of LPNMR*, Springer, 294–308.

Gebser, M., Kaminski, R., Kaufmann, B., Ostrowski, M., Schaub, T. and Wanko, P. 2016. Theory solving made easy with clingo. In *Technical Communications of the 32nd International Conference on Logic Programming (ICLP 2016)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2:1–2:15.

Gebser, M., Kaminski, R., Kaufmann, B. and Schaub, T. 2019. Multi-shot ASP 587 solving with Clingo. *Theory and Practice of Logic Programming* 19, 1, 27–82.

Gebser, M., Kaufmann, B. and Schaub, T. 2012. Conflict-driven answer set solving: From theory to practice. *Artificial Intelligence* 187-188, 52–89.

Gebser, M., Kaufmann, B. and Schaub, T. 2013. Advanced conflict-driven disjunctive answer set solving. In *Proc. of IJCAI*, 13, 912–918.

Ji, J., Liu, F. and You, J.-H. 2017. Well-founded operators for normal hybrid MKNF knowledge bases. *Theory and Practice of Logic Programming* 17, 5-6, 889–905.

Kasalica, V., Knorr, M., Leite, J. and Lopes, C. 2020. NoHR: An overview: Reasoning with ontologies and nonmonotonic rules. *KI - Künstliche Intelligenz* 34, 4, 509–515.

Killen, S. and You, J.-H. 2021. Unfounded sets for disjunctive hybrid MKNF knowledge bases. In *Proc. of KR*, 432–441.

Knorr, M. (2021) On combining ontologies and rules. In *Reasoning Web*. Springer,22–58.

Knorr, M., Alferes, J. J. and Hitzler, P. 2011. Local closed world reasoning with description logics under the well-founded semantics. *Artificial Intelligence* 175, 9-10, 1528–1554.

Lee, J. and Lifschitz, V. 2003. Loop formulas for disjunctive logic programs. In *Proc. of ICLP*, Springer, 2916, 451–465,

Lifschitz, V. 1991. Nonmonotonic databases and epistemic queries. In *Proc. of IJCAI*, 381–386.

Motik, B. and Rosati, R. 2010. Reconciling description logics and rules. *Journal of the ACM* 57, 1-30, 62–62.

Nieuwenhuis, R., Oliveras, A. and Tinelli, C. 2006. Solving SAT and SAT modulo theories: From an abstract Davis–Putnam–Logemann–Loveland procedure to DPLL (T). *Journal of the ACM* 53, 6, 937–977.

Redl, C. 2016. The DLVHEX system for knowledge representation: Recent advances (system description). *Theory and Practice of Logic Programming* 16, 5-6, 866–883.

Wittocx, J., Mariën, M. and Denecker, M. 2008. The IDP system: A model expansion system for an extension of classical logic. In *Proc. of Workshop on Logic and Search*, 153–165.

https://doi.org/10.1017/S1471068424000255 Published online by Cambridge University Press