Fixpoint Characterizations of Disjunctive Hybrid MKNF Knowledge Bases

Spencer Killen[1], Jia-Huai You[1]

[1]University of Alberta, Edmonton, Alberta, Canada

Abstract

Combining answer set programming with ontologies is of great theoretical and practical interest. Hybrid MKNF is a semantics that combines ASP with ontologies without increasing the solving complexity. While there are efficient solvers for ASP and ontologies, efficient solvers for hybrid MKNF have yet to be constructed. In this work, we address some issues that must be solved before a CDNL-based (conflict-driven nogood learning) solver for hybrid MKNF knowledge bases can be developed. We formulate a framework to characterize disjunctive hybrid MKNF semantics through a family of fixpoint operators, then contextualize this framework by demonstrating that it could be possible to integrate it into a solver. Crucially, our approach can be performed without relying on a dependency graph. Finally, we recognize a property of our characterization that is analogous to head-cycle free disjunctive logic programs and demonstrate how to exploit this property to improve solver efficiency.

Keywords

Hybrid MKNF Knowledge Bases, Disjunctive ASP, Fixpoint Computation

1. Introduction

Efficient answer set programming (ASP) solvers are highly desired. Proven solver techniques, such as constraint-driven nogood learning (CDNL) [1], are essential to constructing highly efficient ASP solvers. Solvers that incorporate extensions to ASP allow for a wider range of problems to be tackled with solvers. One powerful extension is to extend ASP programs with ontologies, combining closed- and open-world reasoning. While researchers have proposed a variety of hybrid semantics and have developed efficient CDNL solvers for them [2, 3], these semantics trend towards ASP modulo theories, a class of semantics where stable models are validated by an external theory. The external theories supported by these hybrid semantics are in general nonmonotonic, and thus are ill-suited to capture the monotonic reasoning of ontologies.

Hybrid MKNF (Minimal Knowledge Negation as Failure) knowledge bases are one exception to this trend; Introduced by Motik and Rosati [4], this semantics equips an ASP program with a monotonic fragment of first-order logic obtained from an ontology. Hybrid MKNF is suitable for incorporating into a solver because it is faithful to the underlying semantics of the description

The proofs for this work are available at the following URL:
https://github.com/sjkillen/FixpointCharacterizations/raw/master/proofs.pdf

14th Workshop on Answer Set Programming and Other Computing Paradigms

sjkillen@ualberta.ca (S. Killen); jyou@ualberta.ca (J. You)
0000-0003-3930-5525 (S. Killen); 0000-0001-9372-4371 (J. You)

logic and ASP without increasing complexity [4]. The state-of-the-art of hybrid MKNF solving is only marginally more efficient than guess-and-verify, but a CDNL-based solver would be much more efficient. One major obstacle to developing an efficient CDNL-based solver for disjunctive hybrid MKNF is the unavailability of syntactic dependency; In general, a dependency graph can not be generated for a hybrid knowledge base without knowing the internal structure of the ontology. A framework that treats the ontology as a black box has a broader range of applications than a solver that must be tuned for a particular ontology. Atom dependency analysis is crucial for both model verification and conflict generation.

In this work, we present a framework for disjunctive hybrid MKNF knowledge bases that utilizes fixpoint operators. This framework does not rely on dependency graphs and the only restriction it imposes on the description logic is that the entailment relation can be checked in polynomial time.

2. Related Work

Numerous accounts on the complexity of disjunctive ASP [5, 6, 7, 8] agree that the complexity of computing answer sets of disjunctive logic programs resides in the class Σ_2^P. Leone et al. show that answer sets can be computed by generating unfounded-free interpretations [5], while Lee and Lifschitz show that loop formulas in conjunction with a program's Clark completion can be used for model-checking [7]. Both unfounded sets and loop formulas rely heavily on syntactic dependencies between atoms: One cannot generate a set of loop formulas for a program without knowing its structure and an atom is not deemed unfounded unless it is known to be underivable. The semantics of the well-known ASP solver, Clingo [9], is defined in terms of loop formulas and in terms of unfounded sets. Due to the complexity of model checking, there is an intractable number of loop formulas; Because static dependencies between atoms are easy to establish, this complexity can be handled lazily [10]. Unlike ASP, its ontology-free counterpart, hybrid MKNF knowledge bases do not lend themselves well to dependency graph generation. If a knowledge base's ontology is left unrestricted, generating a static dependency graph for a hybrid MKNF knowledge base would require testing the ontology's entailment relation for all subsets of atoms. In the remainder of this paper, we describe a framework that establishes dependencies between atoms through fixpoint operators and thus does not require static dependency analysis.

3. Preliminaries

Minimal knowledge and negation as failure (MKNF)[11] is an extension of first-order logic that adds two modal operators, $\mathbf{K}$ and **not**, for minimal knowledge and negation as failure respectively. An *MKNF structure* is a triple (I, M, N) where I is a first-order interpretation and M and N are sets of first-order interpretations. The satisfiability relation under an MKNF structure is defined as:

- $(I, M, N) \models A$ if A is true in I where A is a first-order atom
- $(I, M, N) \models \neg F$ if $(I, M, N) \not\models F$
- $(I, M, N) \models F \wedge G$ if $(I, M, N) \models F$ and $(I, M, N) \models G$

- $(I, M, N) \models \mathbf{K}\, F$ if $(J, M, N) \models F$ for each $J \in M$
- $(I, M, N) \models \mathbf{not}\, F$ if $(J, M, N) \not\models F$ for some $J \in N$

Other symbols such as $\vee$, and $\supset$ are interpreted in MKNF as they are in first-order logic.

An *MKNF interpretation* M is a set of first-order interpretations ("possible worlds") and we say that M satisfies a formula F, written $M \models_{MKNF} F$, if $(I, M, M) \models F$ for each $I \in M$. An *MKNF model* M of a formula F is an MKNF interpretation such that $M \models_{MKNF} F$ and there does not exist an MKNF interpretation $M' \supset M$ such that $(I, M', M) \models F$ for each $I \in M'$. Following Motik and Rosati [4], a *hybrid MKNF knowledge base* $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ consists of a decidable description logic knowledge base $\mathcal{O}$ (typically called an ontology) which is translatable to first-order logic and a set of MKNF rules $\mathcal{P}$. We denote this translation as $\pi(\mathcal{O})$ and rules in $\mathcal{P}$ are of the form:

$$\mathbf{K}\, a_1, \ldots, \mathbf{K}\, a_k \leftarrow \mathbf{K}\, a_{k+1}, \ldots, \mathbf{K}\, a_m, \mathbf{not}\, a_{m+1}, \ldots, \mathbf{not}\, a_n \tag{1}$$

In the above, $a_1, \ldots, a_n$ are function-free first-order atoms. Given a rule $r \in \mathcal{P}$, we define the following abbreviations:

$$\begin{aligned}
\mathit{head}(r) &= \{\mathbf{K}\, a_1, \ldots, \mathbf{K}\, a_k\}, \\
\mathit{body}^+(r) &= \{\mathbf{K}\, a_{k+1}, \ldots, \mathbf{K}\, a_m\}, \\
\mathit{body}^-(r) &= \{\mathbf{not}\, a_{m+1}, \ldots, \mathbf{not}\, a_n\}, \\
\mathbf{K}\,(\mathit{body}^-(r)) &= \{\mathbf{K}\, a \mid \mathbf{not}\, a \in \mathit{body}^-(r)\}, \text{ and} \\
\mathit{body}(r) &= (\mathit{body}^+(r), \mathbf{K}\,(\mathit{body}^-(r)))
\end{aligned}$$

Let $\pi(\mathcal{P})$ denote rule set $\mathcal{P}$'s corresponding MKNF formula:

$$\begin{aligned}
\pi(\mathcal{P}) &= \bigwedge_{r \in \mathcal{P}} \pi(r), \text{ where} \\
\pi(r) &= \bigvee_{i=1}^{k} \mathbf{K}\, a_i \subset \bigwedge_{i=k+1}^{m} \mathbf{K}\, a_i \wedge \bigwedge_{i=m+1}^{n} \mathbf{not}\, a_i
\end{aligned}$$

The semantics of a hybrid MKNF knowledge base $\mathcal{K}$ is obtained by applying both transformations to $\mathcal{O}$ and $\mathcal{P}$ and placing $\mathcal{O}$ within a $\mathbf{K}$ operator, i.e. $\pi(\mathcal{K}) = \pi(\mathcal{P}) \wedge \mathbf{K}\, \pi(\mathcal{O})$. We use $\mathcal{P}$, $\mathcal{O}$, and $\mathcal{K}$ in place of $\pi(\mathcal{P})$, $\pi(\mathcal{O})$, and $\pi(\mathcal{K})$ respectively when it is clear from context that the respective translated variant is intended. We refer to formulas of the form $\mathbf{K}\, a$ and $\mathbf{not}\, a$ as **K**-atoms and **Not**-atoms respectively. When it is clear from context, we may write a bare atom a in place of a **K**-atom $\mathbf{K}\, a$. Throughout this work, we assume that MKNF formulas are ground, i.e., they contain no variables.

We now outline some definitions and conventions. For a hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, we denote the set of all modal atoms found within $\mathcal{P}$ as $\mathsf{KA}(\mathcal{K})$ where $\mathsf{KA}(\mathcal{K})$ is defined as follows.

$$\mathsf{KA}(\mathcal{K}) = \{\mathbf{K}\, a \mid \text{either } \mathbf{K}\, a \text{ or } \mathbf{not}\, a \text{ occurs in the head or body of a rule in } \mathcal{P}\}$$

The *objective knowledge* of a hybrid MKNF knowledge base $\mathcal{K}$ w.r.t. a set of **K**-atoms $S \subseteq \mathsf{KA}(\mathcal{K})$ is the set of first-order formulas $\{\pi(\mathcal{O})\} \cup \{a \mid \mathbf{K}\, a \in S\}$. We denote this set as $\mathsf{OB}_{\mathcal{O},S}$.

A *(partial) partition* of $\mathsf{KA}(\mathcal{K})$ is a disjoint pair of subsets of $\mathsf{KA}(\mathcal{K})$. We usually denote a partition as (T, F). **K**-atoms in T are said to be true and **K**-atoms in F are said to be false. A partition is *total* if $T \cup F = \mathsf{KA}(\mathcal{K})$. We frequently treat (T, F) as an interpretation that contains only **K**-atoms. *A dependable partition* is a partial partition (T, F) with the additional restriction that $\mathsf{OB}_{\mathcal{O},T} \cup \{\neg b\}$ is consistent for each $\mathbf{K}\, b \in F$ or $\mathsf{OB}_{\mathcal{O},T}$ is consistent if F is empty. A partial partition that is not dependable may not be extended to an MKNF model. A rule body is applicable w.r.t. a partition (T, F) if $\mathit{body}(r) \sqsubseteq (T, F)$, i.e., if $\mathit{body}^+(r) \subseteq T$ and $\mathbf{K}\,(\mathit{body}^-(r)) \subseteq F$. We say that an MKNF interpretation M of $\mathcal{K}$ induces a partition (T, F) if

$$\bigwedge_{\mathbf{K}\, a \in T} M \models_{MKNF} \mathbf{K}\, a \wedge \bigwedge_{\mathbf{K}\, a \in F} M \models_{MKNF} \neg \mathbf{K}\, a \tag{2}$$

Note that the partition (T^*, F^*) induced by an MKNF model M is unique and dependable. For a partition (T, F) that is a subset of this (T^*, F^*), i.e., $(T, F) \sqsubseteq (T^*, F^*)$, we say that (T, F) can be extended to an MKNF model; such a partition is also dependable.

4. Headcut Semantics

In this section, we formulate a framework that relies on fixpoint operators to represent MKNF models. First consider the MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ where $\pi(\mathcal{O}) = \{(a \vee b) \supset c\}$ and $\mathcal{P}$ only contains the rule $\mathbf{K}\, a, \mathbf{K}\, b \leftarrow \mathbf{not}\, c$. This knowledge base has no MKNF models. Intuitively, we can verify that $\mathcal{K}$ does not have an MKNF model by recognizing that there is no rule to derive $\mathbf{K}\, c$ and thus some **K**-atom in the head of this rule must be true in an MKNF model of $\mathcal{K}$. With either $\mathbf{K}\, a$ or $\mathbf{K}\, b$ true, an inconsistency is created if $\mathbf{K}\, c$ is false.

We generalize and formally express these intuitive semantics by considering head-cuts of $\mathcal{P}$. We define a head-cut R of $\mathcal{K}$ to be a set $R \subseteq \mathcal{P} \times \mathsf{KA}(\mathcal{K})$ where each rule $r \in \mathcal{P}$ occurs in no more than one pair $(r, h) \in R$ and $\mathbf{K}\, h \in \mathit{head}(r)$. For example, the program $\mathcal{P} = \{\mathbf{K}\, a, \mathbf{K}\, b \leftarrow\}$, $\mathcal{P}$ has exactly two head-cuts, $\{(r, a)\}$ and $\{(r, b)\}$ where r refers to the only rule in $\mathcal{P}$ (Note that we omit "**K**" when describing head-cuts). Given a head-cut R, we use $\mathit{head}(R)$ and $\mathit{rule}(R)$ to denote the sets $\{h \mid (r, h) \in R\}$ and $\{r \mid (r, h) \in R\}$ respectively.

Definition 4.1. *For a total partition (T, F), we define $H_{\mathcal{K}}^{(T,F)}$ to be the set containing every head-cut R of $\mathcal{K}$ such that $\mathit{head}(R) \subseteq T$ and for each rule $r \in \mathcal{P}$, $r \in \mathit{rule}(R)$ if and only if $\mathit{body}(r) \sqsubseteq (T, F)$.*

Intuitively, there is a head-cut in $H_{\mathcal{K}}^{(T,F)}$ for each way of selecting a single head-atom for every satisfied rule in $\mathcal{P}$. In essence, $H_{\mathcal{K}}^{(T,F)}$, gives us a way to avoid dealing with negation or disjunction. We use this set to show atoms are justified by defining a family of operators induced by a head-cut R:

$$\begin{aligned} Q^R(X) = &\{\mathbf{K}\, h \mid \text{ where } (r, h) \in R \text{ and } \mathit{body}^+(r) \subseteq X \text{ for each } r \in \mathcal{P}\ \} \\ &\cup \{\mathbf{K}\, a \in \mathsf{KA}(\mathcal{K}) \mid \mathsf{OB}_{\mathcal{O},X} \models a\} \end{aligned}$$

We denote the least fixpoint of an operator Q^R as $\mathbf{lfp}\, Q^R$ and use Q^R_i to denote applying the Q^R operator i times on the empty set (e.g., $Q^R_2 = Q^R(Q^R(\emptyset))$). This operator simply takes a set of $\mathbf{K}$-atoms X and extends it with immediate consequences under R.

Revisiting the initial example where $\mathcal{K} = (\mathcal{O}, \mathcal{P})$, $\pi(\mathcal{O}) = \{(a \vee b) \supset c\}$, and $\mathcal{P} = \{\mathbf{K}\, a, \mathbf{K}\, b \leftarrow \mathbf{not}\, c\}$, consider any total dependable partition (T, F) where $\mathbf{K}\, c \in F$. Observe that for each head-cut $R \in H^{(T,F)}_{\mathcal{K}}$, we have $c \in \mathbf{lfp}\, Q^R$. For example, let $(T, F) = (\{\mathbf{K}\, a\}, \{\mathbf{K}\, c, \mathbf{K}\, b\})$. Every head-cut R in $H^{(T,F)}_{\mathcal{K}}$ contains the rule from $\mathcal{P}$, thus the Q^R operator will compute either a or b on the first iteration and then c on the second iteration. Conversely, if we consider a dependable partition (T, F) such that $\mathbf{K}\, c \in T$, then for each head-cut $R \in H^{(T,F)}_{\mathcal{K}}$, we have $c \notin \mathbf{lfp}\, Q^R$. For example, let $(T, F) = (\{\mathbf{K}\, a, \mathbf{K}\, c\}, \{\mathbf{K}\, b\})$. No rule in $\mathcal{P}$ is applicable w.r.t. (T, F), thus $H^{(T,F)}_{\mathcal{K}}$ contains a single head-cut, $\emptyset$. We have $Q^\emptyset_0 = Q^\emptyset_1 = \emptyset$, thus $c \notin \mathbf{lfp}\, Q^R$.

We now formally show that sets of the form $H^{(T,F)}_{\mathcal{K}}$ coincide with an MKNF models of hybrid knowledge bases. First, we connect family of sets $H^{(T,F)}_{\mathcal{K}}$ to total partitions that satisfy all rules in $\mathcal{P}$ but are not necessarily induced by an MKNF model.

Lemma 4.1. *For a total partition (T, F), the set $H^{(T,F)}_{\mathcal{K}}$ is empty if and only if there exists a rule $r \in \mathcal{P}$ where body$(r) \sqsubseteq (T, F)$ and head$(r) \cap T = \emptyset$.*

Definition 4.2. *A set of head-cuts H is a supporting set for a total dependable partition (T, F) if: An MKNF model M of $\mathcal{K}$ that induces (T, F) exists if and only if H is nonempty and for each $R \in H$ the set computed by $\mathbf{lfp}\, Q^R$ is precisely T.*

Proposition 4.1. *The set $H^{(T,F)}_{\mathcal{K}}$ is a supporting set of the dependable partition (T, F).*

In the following example, we demonstrate how the set $H^{(T,F)}_{\mathcal{K}}$ can be used to verify that a partition can be extended to a model.

Example 1. *Let $\mathcal{K} = (\emptyset, \mathcal{P})$ where $\mathcal{P}$ is defined as*

$$1 : \mathbf{K}\, a, \mathbf{K}\, b \leftarrow \mathbf{K}\, c \qquad\qquad 2 : \mathbf{K}\, b, \mathbf{K}\, c \leftarrow$$

Let $(T, F) = (\{\mathbf{K}\, b, \mathbf{K}\, c\}, \{\mathbf{K}\, a\})$. By definition, the set $H^{(T,F)}_{\mathcal{K}}$ contains two head-cuts: $R_0 = \{(1, b), (2, b)\}$ and $R_1 = \{(1, b), (2, c)\}$. When we repeatedly apply the Q operator on each of these head-cuts we obtain the following sets.

$$\begin{array}{llll} Q^{R_0}_0 = \emptyset & Q^{R_0}_1 = \{b\} & Q^{R_0}_2 = Q^{R_0}_1 & Q^{R_0}_3 = Q^{R_0}_1 \\ Q^{R_1}_0 = \emptyset & Q^{R_1}_1 = \{c\} & Q^{R_1}_2 = \{b, c\} & Q^{R_1}_3 = Q^{R_1}_2 \end{array}$$

Using Proposition 4.1, it is easy to confirm that $(T, F) = (\{\mathbf{K}\, a, \mathbf{K}\, b, \mathbf{K}\, c\}, \emptyset)$ can not be extended to a model of $\mathcal{K}$ by observing that neither $\mathbf{lfp}\, Q^{R_0}$ nor Q^{R_1} computes T. If we instead use $(T, F) = (\{\mathbf{K}\, a, \mathbf{K}\, c\}, \{\mathbf{K}\, b\})$, then the set $H^{(T,F)}_{\mathcal{K}}$ contains the just a single head-cut, $R_0 = \{(1, a), (2, c)\}$. We have $\mathbf{lfp}\, Q^{R_0} = T$, thus (T, F) can be extended to an MKNF model.

Now that we have established a semantics in terms of a family of fixpoint operators, we discuss some optimizations that, when applied to $H_{\mathcal{K}}^{(T,F)}$, greatly reduce the number of head-cuts in the set. This is a crucial step that enables the set to be used in a solver. For a dependable partition (T, F), that cannot be extended to a model, there are many head-cuts in $H_{\mathcal{K}}^{(T,F)}$ that contain rules whose bodies are never satisfied by iterative construction or do not derive anything new. Our first optimization step is to remove such rules from head-cuts by defining a new set $HM_{\mathcal{K}}^{(T,F)}$ based on $H_{\mathcal{K}}^{(T,F)}$. A head-cut R is in $HM_{\mathcal{K}}^{(T,F)}$ if and only if there is a head-cut $R' \in H_{\mathcal{K}}^{(T,F)}$ such that $R = R' \setminus RM$, where

$$RM = \{(r, h) \in R \mid \forall i, \mathit{body}(r) \subseteq Q_i^R \implies \mathit{head}(r) \subseteq Q_i^R\}$$

This set removes some pairs in each head-cut from $H_{\mathcal{K}}^{(T,F)}$. For each head-cut $R \in HM_{\mathcal{K}}^{(T,F)}$ there is some head-cut $R' \in H_{\mathcal{K}}^{(T,F)}$ for which $R \subseteq R'$. However, R' is not unique in general. The set $HM_{\mathcal{K}}^{(T,F)}$ also has the convenient property of only including pairs that contribute to the computation of Q^R, thus $\mathit{head}(R) \subseteq \mathbf{lfp}\, Q^R$.

In the example following example, we demonstrate that $HM_{\mathcal{K}}^{(T,F)}$ can be used as a supporting set.

Example 2. *Let $\mathcal{K} = (\emptyset, \mathcal{P})$ where $\mathcal{P}$ contains the following rules*

$$\begin{aligned} 0: &\quad \mathbf{K}\, a, \mathbf{K}\, b \leftarrow \\ 1: &\quad \mathbf{K}\, c \leftarrow \mathbf{K}\, a \\ 2: &\quad \mathbf{K}\, c \leftarrow \mathbf{K}\, b \\ 3: &\quad \mathbf{K}\, a, \mathbf{K}\, d \leftarrow \mathbf{K}\, c \end{aligned}$$

Let $(T, F) = (\mathsf{KA}(\mathcal{K}), \emptyset)$ and $R = \{(0, a), (1, c), (3, d)\}$. Q_1^R computes "a" via rule 0, then Q_2^R computes "c" via rule 2. On the third iteration, Q_3^R computes "d" via rule 3, however, we already have $\mathit{head}(3) \subseteq Q_2^R$, thus R is not in $HM_{\mathcal{K}}^{(T,F)}$. Instead, the head-cut $R = \{(0, a), (1, c)\}$ is in $HM_{\mathcal{K}}^{(T,F)}$.

Like $H_{\mathcal{K}}^{(T,F)}$, the set $HM_{\mathcal{K}}^{(T,F)}$ is a supporting set of (T, F).

Proposition 4.2. *$HM_{\mathcal{K}}^{(T,F)}$ is a supporting set of (T, F).*

We define more optimizations that further reduce the number of head-cuts that need to be tested to verify a model. A head-cut R is *branch-minimal* w.r.t. a set of head-cuts S if for each $R' \in S$ such that $\mathit{head}(R) \subseteq \mathit{head}(R')$ or $\mathit{head}(R') \subseteq \mathit{head}(R)$, we have $\mathit{head}(R) \subseteq \mathit{head}(R')$. It can be easily shown that this relation is a partial order between head-cuts.

We formulate a further optimization of $HM_{\mathcal{K}}^{(T,F)}$ based on this notion of minimality.

$$HP_{\mathcal{K}}^{(T,F)} = \{R \in HM_{\mathcal{K}}^{(T,F)} \mid R \text{ is branch-minimal w.r.t. } HM_{\mathcal{K}}^{(T,F)}\}$$

Proposition 4.3. *$HP_{\mathcal{K}}^{(T,F)}$ is a supporting set of (T, F).*

The set $HP_{\mathcal{K}}^{(T,F)}$ is not practical for use in solver: Because of the complexity of determining whether a head-cut is branch-minimal, the set cannot be efficiently enumerated. We develop a supporting set that is a compromise of $HP_{\mathcal{K}}^{(T,F)}$.

Given a head-cut R, we define $R[i]$ to be the subset of R that contains only the rules in *rule*(R) whose bodies are satisfied after i iterations of the Q^R operator, that is,

$$R[i] = \{(r, h) \in R \mid h \notin Q_i^R, body^+(r) \subseteq Q_i^R\}$$

Intuitively, $R[i]$ contains the atoms newly derived on iteration i. Likewise, we define a range $R[0..j]$ where $0 \leq j$

$$R[0..j] = \bigcup_{k=0}^{j} R[k]$$

A head-cut R is *semi-branch-minimal* w.r.t. a head-cut R' if for the largest i such that

$$R[0..(i-1)] = R'[0..(i-1)]$$

we have *head*$(R[i]) \subseteq$ *head*$(R'[i])$.

We define $HG_{\mathcal{K}}^{(T,F)}$ to be the set

$$HG_{\mathcal{K}}^{(T,F)} = \{R \in HM_{\mathcal{K}}^{(T,F)} \mid R \text{ is semi-branch-minimal w.r.t. every other head-cut } R' \in HM_{\mathcal{K}}^{(T,F)}\}$$

We give an example of a head-cut from $HM_{\mathcal{K}}^{(T,F)}$ that is also in $HG_{\mathcal{K}}^{(T,F)}$

Example 3. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ where $\mathcal{O} = \emptyset$ and*

$$\begin{aligned} 1 :& \mathbf{K}\, a, \mathbf{K}\, b \leftarrow \mathbf{K}\, c \\ 2 :& \mathbf{K}\, a, \mathbf{K}\, b \leftarrow \mathbf{K}\, c \\ 3 :& \mathbf{K}\, c \leftarrow \end{aligned}$$

Let $(T, F) = (\mathsf{KA}(\mathcal{K}), \emptyset)$. Define the following head-cuts:

$$\begin{aligned} R_0 &= \{(1, a), (2, a), (3, c)\}, \\ R_1 &= \{(1, a), (2, b), (3, c)\}, \\ R_2 &= \{(1, b), (2, a), (3, c)\}, \\ R_3 &= \{(1, b), (2, b), (3, c)\} \end{aligned}$$

We have $HM_{\mathcal{K}}^{(T,F)} = \{R_0, R_1, R_2, R_3\}$. For each pair of head-cuts R_i and R_j, we have $R_i[0..1] = R_j[0..1]$. However, we have head$(R_0[2]) \subset$ head$(R_1[2])$ and head$(R_0[2]) \subset$ head$(R_2[2])$, thus neither R_1 nor R_2 is semi-branch-minimal. This gives us $HG_{\mathcal{K}}^{(T,F)} = \{R_0, R_2\}$.

Both the sets $HP_{\mathcal{K}}^{(T,F)}$ and $HG_{\mathcal{K}}^{(T,F)}$ are subsets of $HM_{\mathcal{K}}^{(T,F)}$, however, in general, neither set is a subset of the other. The set $HG_{\mathcal{K}}^{(T,F)}$ is easier to enumerate than $HP_{\mathcal{K}}^{(T,F)}$ because head-cuts can be constructed iteratively whereas to enumerate head-cuts in $HP_{\mathcal{K}}^{(T,F)}$, one must test whether each head-cut in $HM_{\mathcal{K}}^{(T,F)}$ is branch-minimal. We demonstrate how $HG_{\mathcal{K}}^{(T,F)}$ may be iterated later on when we construct an abstract solver.

Example 4. *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ such that $\mathcal{O} = \emptyset$ and*

$$\begin{aligned}\mathcal{P} = \{1 : \mathbf{K}\,a, \mathbf{K}\,b, \mathbf{K}\,c, \mathbf{K}\,d \leftarrow; 2 : \mathbf{K}\,a, \mathbf{K}\,b, \mathbf{K}\,d \leftarrow;\\ \mathbf{K}\,c \leftarrow a; \mathbf{K}\,c \leftarrow b; \mathbf{K}\,a \leftarrow b; \mathbf{K}\,b \leftarrow a; \mathbf{K}\,a \leftarrow d; \mathbf{K}\,b \leftarrow d;\}\end{aligned}$$

We use the total partition $(T, F) = (\mathsf{KA}(\mathcal{K}), \emptyset)$ to consider various head-cuts in the sets $HG_{\mathcal{K}}^{(T,F)}$ and $HP_{\mathcal{K}}^{(T,F)}$. For brevity, we omit pairs in head-cuts that contain normal rules. First, we give a head-cut R found in $HG_{\mathcal{K}}^{(T,F)}$ but not $HP_{\mathcal{K}}^{(T,F)}$.

$$\begin{aligned} R &= \{(1, d), (2, d)\} \\ Q_1^R &= \{d\} \\ \textbf{\textit{lfp}}\, Q^R &= \{a, b, c, d\}\end{aligned}$$

R is semi-branch-minimal w.r.t. every head-cut from $HM_{\mathcal{K}}^{(T,F)}$ because head(R) contains a single atom and there is no head-cut R' in $HM_{\mathcal{K}}^{(T,F)}$ such that head$(R') = \emptyset$. However, the selection of d in R results in the atoms a, b, and c, also being derived; For the head $R' = \{(1, a), (2, a)\}$, $\textbf{\textit{lfp}}\, Q^{R'} = \{a, b, c\}$, thus R is not branch-minimal. We give a head-cut R that is in $HP_{\mathcal{K}}^{(T,F)}$ and not $HG_{\mathcal{K}}^{(T,F)}$:

$$\begin{aligned} R &= \{(1, c), (2, b)\} \\ Q_1^R &= \{b, c\} \\ \textbf{\textit{lfp}}\, Q^R &= \{a, b, c\}\end{aligned}$$

R is branch-minimal because every head-cut in $HM_{\mathcal{K}}^{(T,F)}$ computes at least a, b, and c, however, R is not semi-branch-minimal because $c \in$ head(R); A head-cut R in $HG_{\mathcal{K}}^{(T,F)}$ cannot contain a pair with c in it because there is always a head-cut R' that is semi-branch-minimal w.r.t. R.

We give an exhaustive account of the head-cuts that are in both $HG_{\mathcal{K}}^{(T,F)}$ and $HP_{\mathcal{K}}^{(T,F)}$:

$$\begin{aligned} R_0 &= \{(1, a), (2, a)\} \\ R_1 &= \{(1, b), (2, b)\} \\ \textbf{\textit{lfp}}\, Q^{R_0} &= \textbf{\textit{lfp}}\, Q^{R_1} = \{a, b\}\end{aligned}$$

As mentioned above, no head-cut R in $HP_{\mathcal{K}}^{(T,F)}$ can have $d \in$ head(R) and no head-cut R in $HG_{\mathcal{K}}^{(T,F)}$ can have $c \in$ head(R), thus no head-cut in $HG_{\mathcal{K}}^{(T,F)} \cap HP_{\mathcal{K}}^{(T,F)}$ can contain either

atom. Finally, we give an example of a head-cut in $HM_{\mathcal{K}}^{(T,F)}$ *that is neither in* $HM_{\mathcal{K}}^{(T,F)}$ *nor* $HG_{\mathcal{K}}^{(T,F)}$*:*

$$R = \{(1, a), (2, d)\}$$
$$Q_1^R = \{a, d\}$$
$$\textbf{\textit{lfp}}\, Q^R = \{a, b, c, d\}$$

The head-cut demonstrated above is neither branch-minimal nor semi-branch-minimal.

Even though $HP_{\mathcal{K}}^{(T,F)}$ and $HG_{\mathcal{K}}^{(T,F)}$ are disjoint, they are related in a crucial way that allows us to show that $HG_{\mathcal{K}}^{(T,F)}$ is a supporting set. Intuitively, a head-cut R from $HP_{\mathcal{K}}^{(T,F)}$ can be thought of as having a minimal set *head*(R) that is globally minimal w.r.t. iterations of Q_i^R whereas a head-cut R from $HM_{\mathcal{K}}^{(T,F)}$'s set *head*(R) is only locally minimal w.r.t. an iteration of Q_i^R. As it turns out, $HP_{\mathcal{K}}^{(T,F)}$ is more precise and if there is a head-cut in $HP_{\mathcal{K}}^{(T,F)}$ that fails to compute T, we can guarantee that such a head-cut exists in $HG_{\mathcal{K}}^{(T,F)}$ as well. We demonstrate this property formally.

Lemma 4.2. *If there exists a head-cut in* $R \in HP_{\mathcal{K}}^{(T,F)}$ *such that* $\textbf{\textit{lfp}}\, Q^R \subset T$*, then there is a head-cut* $R' \in HP_{\mathcal{K}}^{(T,F)} \cap HG_{\mathcal{K}}^{(T,F)}$ *such that* $\textbf{\textit{lfp}}\, Q^{R'} \subset T$

Finally, we can use the property demonstrated above to show that $HG_{\mathcal{K}}^{(T,F)}$ is a supporting set of (T, F).

Proposition 4.4. $HG_{\mathcal{K}}^{(T,F)}$ *is a supporting set of* (T, F)*.*

5. An Abstract Solver

Up until now, we have dealt exclusively with total partitions. We now discuss how the techniques described in the previous section can be applied to partial partitions and in turn be used to develop an abstract solver.

We define the set $D_{\mathcal{K}}^R$ under a head-cut R.

$$D_{\mathcal{K}}^R = \{HG_{\mathcal{K}}^{(T^*,F^*)} \mid \text{ where } R = R^*[0..i] \text{ for some } i \text{ and } R^* \in HG_{\mathcal{K}}^{(T^*,F^*)} \text{ and total partition } (T^*, F^*)\}$$

Given a head-cut R, we can extract an appropriate partition from R:

$$\mathcal{S}(R) = (\textbf{lfp}\, Q^R, \bigcup\{body^-(r) \mid (r, h) \in R\})$$

Note that for each $HG_{\mathcal{K}}^{(T^*,F^*)} \in D_{\mathcal{K}}^R$, we have $\mathcal{S}(R) \sqsubseteq (T^*, F^*)$. Intuitively, the set $D_{\mathcal{K}}^R$ holds every possible set $HG_{\mathcal{K}}^{(T^*,F^*)}$ if $\mathcal{S}(R)$ were extended to a total partition (T^*, F^*). Note that (T^*, F^*) may not be dependable. We also define a total variant of $\mathcal{S}(R)$:

$$\mathcal{S}^*(R) = (\textbf{lfp}\, Q^R, \mathsf{KA}(\mathcal{K}) \setminus \textbf{lfp}\, Q^R)$$

We recursively define a subclass of head-cuts to limit the use of $D_{\mathcal{K}}^R$.

Definition 5.1. *Given, a head-cut R where $\mathcal{S}(R)$ is a dependable partition, we call R a head-cut state if either $R = \emptyset$ or there is another head-cut state R' such that either $R = R'[0..i]$ or $R \in \bigcup D^{R'}_{\mathcal{K}}$.*

Intuitively, a head-cut state is a head-cut that can be extended to some a head-cut in $R \in H$ for every $H \in D^R_{\mathcal{K}}$. Note that $\mathcal{S}(R) \sqsubseteq \mathcal{S}^*(R)$ for a head-cut state R.

We now define an abstract solver that operates on head-cut states. Supporting can assist a solver in several key ways: Conflict propagation and immediate propagation that adds positive direct consequences to a head-cut state R (See $T(R)$ in Algorithm 1), Guiding solver decisions at points where the current head-cut state cannot be extended through means of well-founded propagation (See $decisions(R)$ in Algorithm 2) and as already shown, supporting sets can be used for model verification. We join these roles together in an abstract solver outlined in Algorithm 3. Finally, we show how supporting sets can be used to reason about head-cut states that can be verified in polynomial time (similar to head-cycle free). We leave certain details unspecified such as how to enumerate the set $D^R_{\mathcal{K}}$ in an efficient way in the context of each of the algorithms and the overall complexity of the algorithms we provide.

Algorithm 1: $T(R)$

$(T, F) \leftarrow \mathcal{S}(R)$;
$B \leftarrow \{R'[i+1] \mid R' \in \bigcup D^R_{\mathcal{K}}, R'[0..i] = R\}$;
if $\exists (r, h) \in \bigcap B,\ |head(r) \setminus F| \neq 1$ *or* $body(r) \not\sqsubseteq (T, F)$ **then**
    return R;
assert $|B| \leq 1$;
return $R \cup \bigcap B$;

Intuitively, Algorithm 1 adds information to R only if there are only rules whose bodies are satisfied w.r.t. $\mathcal{S}(R)$ and the head of the rule contains no true atoms and only a single atom that is not false. We demonstrate that a solver cannot miss any models by applying $T(R)$ to a head-cut state.

Lemma 5.1. *For a head-cut state R and any head-cut $R^* \in \bigcup D^R_{\mathcal{K}}$ where $R = R^*[0..i]$, we either have $R^*[0..(i+1)] = T(R)$ or $R = T(R)$.*

With $T(R)$, we only propagate information if it holds in all models that $\mathcal{S}(R)$ can be extended to. At some point in the solving process, we must add information for which this does not hold.

We describe a process for extending head-cut states with decision atoms.

Algorithm 2: $decisions(R)$

```
Decisions ← ∅;
if R ∈ ⋃ D_K^R then
    return ∅;
for R′ ∈ ⋃ D_K^R where ∃i, R′[0..i] = R do
    if S(R′[0..(i + 1)]) is dependable then
        Decisions ← Decisions ∪ {R′[0..(i + 1)]};
return Decisions;
```

We guarantee very little about the atoms added by $decisions(R)$ and in general, the solver will be forced to backtrack because of the atoms that were added by this procedure. However, extensions made by $decision(R)$ will maintain the property that R is a head-cut state. Furthermore, if $\mathcal{S}(R)$ can be extended to an MKNF model of $\mathcal{K}$, then a head-cut in $decisions(R)$ also has this property. This ensures that a solver that uses $decisions(R)$ will not miss any models.

Lemma 5.2. *Given a head-cut state R and an MKNF model M that induces $\mathcal{S}(R)$, there is either a head-cut state $R' \in decisions(R)$ such that M induces $\mathcal{S}(R')$ or $decisions(R)$ is empty.*

We define *check-model*(R) to be a procedure that simply enumerates all head-cuts in $HG_{\mathcal{K}}^{\mathcal{S}^*(R)}$ to verify that $\mathcal{S}^*(R)$ can be extended to a model by applying the Q^R until a fixed point is reached. The correctness of such an algorithm follows directly from Proposition 4.4.

We now integrate all preceding algorithms into an abstract solver.

Algorithm 3: $solver(R)$

```
R ← lfp T(R);
for R′ ∈ decisions(R) do
    if solver(R′) then
        return solver(R′);
if decisions(R) = ∅ and check-model(R) then
    return S*(R)
return false;
```

While we do not specify which head-cut from $decisions(R)$ should be selected to minimize backtracking, our algorithm can locate a model if one exists.

Lemma 5.3. *Given a head-cut state R, if there exists a model that induces $\mathcal{S}(R)$, then $solver(R)$ will return a total partition induced by a model.*

While more efficient than a guess-and-verify solver, Algorithm 3 does not efficiently verify total partitions. It is well known that head-cycle free disjunctive logic programs can be solved in NP [12]. We demonstrate an analogous subclass of disjunctive MKNF knowledge bases that can be verified in polynomial time.

First, we identify a property of head-cut states that enables polynomial verification. We show that this property holds if and only if a head-cut state coincides with an MKNF model.

Definition 5.2. *A head-cut state R is P-verifiable if the following holds. For every head-cut $R^* \in \bigcup D^R_{\mathcal{K}}$, where $R^*[0..i] = R[0..i]$ and $R^*[i+1] \neq R[i+1]$, we have **lfp** $Q^{R^*} \supseteq$ head$(R[i+1])$.*

Proposition 5.1. *If a head-cut state R is P-verifiable and $\mathcal{S}^*(R)$ is dependable, then **lfp** $Q^R = T$ if and only if **lfp** $Q^{R'} = T$ for each $R' \in HG^{\mathcal{S}^*(R)}_{\mathcal{K}}$.*

Corollary 5.1. *A head-cut state R is P-verifiable if and only if $\mathcal{S}^*(R)$ can be extended to an MKNF model of $\mathcal{K}$.*

Using the above property, we construct a verification algorithm that is more efficient than *check-model*(R).

Algorithm 4: *check-model*$_2(R)$

```
1 if HG_K^{S*(R)} = ∅ then
2     return false;
3 for i ← 0; Q_i^R ≠ Q_{i-1}^R; i ← i + 1 do
4     for R' ∈ HG_K^{S*(R)} where R[0..i] = R'[0..i] and R[i+1] ≠ R'[i+1] do
5         if head(R[i+1]) ⊈ lfp Q^{R'} then
6             return false;
7 return true;
```

When *check-model*(R) is replaced with Algorithm 4 in the solver (Algorithm 3), P-verifiable head-cut states can be quickly verified. We feel strongly that with further complexity analysis we will be able conclude that our abstract solver algorithm, when used with an empty ontology and head-cycle free disjunctive logic program, can verify any enumerated partition in polynomial time.

6. Conclusion

We have provided a new way of characterizing disjunctive MKNF models through supporting sets. The largest set we defined, $H^{(T,F)}_{\mathcal{K}}$, contains many redundant head-cuts and is not practical for use in a solver. We defined a smaller set, $HM^{(T,F)}_{\mathcal{K}}$, where each head-cut in $HM^{(T,F)}_{\mathcal{K}}$ is limited to rules that contribute to the fixpoint computation. Next we refined $HM^{(T,F)}_{\mathcal{K}}$ further to obtain $HP^{(T,F)}_{\mathcal{K}}$ and the less precise but more tractable set $HG^{(T,F)}_{\mathcal{K}}$. We provided an abstract solver that utilizes supporting sets to enumerate partitions and to verify models. Finally, we characterized P-verifiable head-cut states, a property of head-cut states that is comparable to head-cycle-free disjunctive logic programs, and we give a more efficient model verification procedure that leverages this property.

We speculate that the complexity of our abstract solver algorithm is no worse than a guess-and-verify solver if the entailment relation of the accompanying ontology can be computed

in polynomial time. We also speculate that if the ontology is empty and $\mathcal{P}$ is a head-cycle free disjunctive logic program that the complexity of finding a model lies in NP. However, we leave a full analysis of the complexity of our algorithm and the complexity of recognizing P-verifiable head-cut states to future work. In this work, we introduce many new structures to characterize our semantics; It would be interesting to recast this work using more familiar fixpoint structures such as approximators in approximation fixpoint theory [13]. In the future, we would also like to leverage this framework to generate conflicts so that a CDNL-based solver may be constructed.

References

[1] M. Gebser, B. Kaufmann, T. Schaub, Advanced conflict-driven disjunctive answer set solving, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, AAAI Press, 2013, p. 912–918.

[2] T. Eiter, G. Ianni, R. Schindlauer, H. Tompits, A uniform integration of higher-order reasoning and external evaluations in answer-set programming., 2005, pp. 90–96.

[3] M. Gebser, R. Kaminski, B. Kaufmann, M. Ostrowski, T. Schaub, P. Wanko, Theory solving made easy with clingo 5, in: M. Carro, A. King, N. Saeedloei, M. D. Vos (Eds.), Technical Communications of the 32nd International Conference on Logic Programming, ICLP 2016 TCs, October 16-21, 2016, New York City, USA, volume 52 of *OASICS*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016, pp. 2:1–2:15.

[4] B. Motik, R. Rosati, Reconciling description logics and rules, J. ACM 57 (2010) 30:1–30:62.

[5] N. Leone, P. Rullo, F. Scarcello, Disjunctive stable models: Unfounded sets, fixpoint semantics, and computation, Inf. Comput. 135 (1997) 69–112.

[6] V. Lifschitz, A. A. Razborov, Why are there so many loop formulas?, ACM Trans. Comput. Log. 7 (2006) 261–268.

[7] J. Lee, V. Lifschitz, Loop formulas for disjunctive logic programs, in: C. Palamidessi (Ed.), Logic Programming, 19th International Conference, ICLP 2003, Mumbai, India, December 9-13, 2003, Proceedings, volume 2916 of *Lecture Notes in Computer Science*, Springer, 2003, pp. 451–465.

[8] T. Eiter, G. Gottlob, On the computational cost of disjunctive logic programming: Propositional case, Ann. Math. Artif. Intell. 15 (1995) 289–323.

[9] M. Gebser, B. Kaufmann, T. Schaub, Conflict-driven answer set solving: From theory to practice, Artif. Intell. 187 (2012) 52–89.

[10] C. Drescher, T. Walsh, Answer set solving with lazy nogood generation, in: A. Dovier, V. S. Costa (Eds.), Technical Communications of the 28th International Conference on Logic Programming, ICLP 2012, September 4-8, 2012, Budapest, Hungary, volume 17 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2012, pp. 188–200.

[11] V. Lifschitz, Nonmonotonic databases and epistemic queries, in: J. Mylopoulos, R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence. Sydney, Australia, August 24-30, 1991, Morgan Kaufmann, 1991, pp. 381–386. URL: http://ijcai.org/Proceedings/91-1/Papers/059.pdf.

[12] R. Ben-Eliyahu, R. Dechter, Propositional semantics for disjunctive logic programs, Ann. Math. Artif. Intell. 12 (1994) 53–87.

[13] F. Liu, J. You, Alternating fixpoint operator for hybrid MKNF knowledge bases as an approximator of AFT, in: P. Fodor, M. Montali, D. Calvanese, D. Roman (Eds.), Rules and Reasoning - Third International Joint Conference, RuleML+RR 2019, Bolzano, Italy, September 16-19, 2019, Proceedings, volume 11784 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 113–127.