

Query-driven Procedures for Hybrid MKNF Knowledge Bases

JOSÉ JÚLIO ALFERES

and

MATTHIAS KNORR

and

TERRANCE SWIFT

CENTRIA, FCT, Universidade Nova de Lisboa

Hybrid MKNF knowledge bases are one of the most prominent tightly integrated combinations of open-world ontology languages with closed-world (non-monotonic) rule paradigms. The definition of Hybrid MKNF is parametric on the description logic (DL) underlying the ontology language, in the sense that non-monotonic rules can extend any decidable DL language. Two related semantics have been defined for Hybrid MKNF: one that is based on the Stable Model Semantics for logic programs and one on the Well-Founded Semantics (WFS). Under WFS, the definition of Hybrid MKNF relies on a bottom-up computation that has polynomial data complexity whenever the DL language is tractable. Here we define a general query-driven procedure for Hybrid MKNF that is sound with respect to the stable model-based semantics, and sound and complete with respect to its WFS variant. This procedure is able to answer a slightly restricted form of conjunctive queries, and is based on tabled rule evaluation extended with an external oracle that captures reasoning within the ontology. Such an (abstract) oracle receives as input a query along with knowledge already derived, and replies with a (possibly empty) set of atoms, defined in the rules, whose truth would suffice to prove the initial query. With appropriate assumptions on the complexity of the abstract oracle, the general procedure maintains the data complexity of the WFS for Hybrid MKNF knowledge bases.

To illustrate this approach, we provide a concrete oracle for \mathcal{EL}^+ , a fragment of the light-weight DL \mathcal{EL}^{++} . Such an oracle has practical use, as \mathcal{EL}^{++} is the language underlying OWL 2 EL, which is part of the W3C recommendations for the Semantic Web, and is tractable for reasoning tasks such as subsumption. We show that query-driven Hybrid MKNF preserves polynomial data complexity when using the \mathcal{EL}^+ oracle and WFS.

Categories and Subject Descriptors: I.2.4 [Knowledge Representation Formalisms and Methods]: Representation languages; Modal logic; I.2.3 [Deduction and Theorem Proving]: Logic programming; Nonmonotonic reasoning and believe revision; Inference engines; F.4.1 [Mathematical Logic]: Computational logic

General Terms: Theory, Languages

Additional Key Words and Phrases: hybrid knowledge bases, ontologies, rules, tabling, well-founded semantics, description logics, tractable fragments

Author's addresses: J.J. Alferes, M. Knorr, and T. Swift: CENTRIA, Faculdade de Ciências e Tecnologia, Univ. Nova de Lisboa, 2825-516 Caparica, Portugal. Preliminary versions of some of this material appeared in [Alferes et al. 2009] and [Knorr and Alferes 2010].

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.

© 2018 ACM 1529-3785/2018/0700-0001 \$5.00

1. INTRODUCTION

It is frequently claimed that integrating open world with closed world reasoning is a key issue for practical large-scale ontology applications. As one example, [Patel et al. 2007] describes a large medical case study about matching patient records for clinical trials criteria containing up to millions of assertions. In that clinical domain, open world reasoning is needed for radiology and laboratory data, because, for example, unless a lab test asserts a negative finding, no arbitrary assumptions about the test can be made. However, in pharmacy data, the closed world assumption can be used to infer that a patient is not on a specific medication unless it is asserted.

In general, both ontologies and rules provide distinct strengths for the representation and interchange of knowledge in the Semantic Web and for applications of knowledge representation, such as the one described above. Expressive ontology languages are usually fragments of first-order logics represented in description logics (DLs) [Baader et al. 2007] and offer the deductive advantages of first-order logics with an open domain, while guaranteeing decidability. Rules on the other hand offer non-monotonic (closed-world) reasoning that can be useful for formalizing scenarios under (local) incomplete knowledge. They also enable reasoning about fixed points (e.g., reachability), which cannot be expressed within first-order logic. Interest in ontologies, rules, and their combination is demonstrated by the development of ontology languages for the Semantic Web, such as OWL [Hitzler et al. 2009], and the growing interest on rule languages for the Semantic Web, cf. the RIF [Boley and Kifer 2010] and the RuleML¹ initiatives.

The majority of proposals for combining rules and ontologies (see, e.g., related work in [Eiter et al. 2008; Knorr et al. 2011]) rely on one of the two most common semantics for rules: the Well-Founded Semantics (WFS) [van Gelder et al. 1991] or the Answer-Sets Semantics [Gelfond and Lifschitz 1991]. Both semantics are widely used and allow closed-world reasoning and the representation of fixed points. Furthermore, the relationship between the two semantics has been fully explored. Of the two, the Well-Founded Semantics is weaker (in the sense that it is more skeptical w.r.t. derivable consequences), but it has the clear advantage that its lower complexity is more suitable for applications with large amounts of data, such as the medical case study described above.

Several formalisms have concerned themselves with combining decidable DLs with WFS rules [Drabent and Małuszyński 2007; Lukasiewicz 2010; Eiter et al. 2011; Knorr et al. 2011]. Among these, the well-founded semantics for Hybrid MKNF knowledge bases ($MKNF_{WF}$), introduced first in [Knorr et al. 2008] and further refined in [Knorr et al. 2011], is based on a three-valued extension of the logics of minimal knowledge and negation as failure (MKNF) [Lifschitz 1991], and is the only one that allows knowledge about instances to be fully inter-definable between rules and an ontology that is taken as a parameter of the formalism.

$MKNF_{WF}$ is defined using a monotonic fixpoint operator that computes in each iteration step, besides the usual immediate consequences from rules, the set of all atoms derivable from the ontology that is augmented with the already proven atomic knowledge. The least fixpoint of the $MKNF_{WF}$ operator coincides with the

¹<http://ruleml.org/>

original WFS [van Gelder et al. 1991] if the ontology is empty, and coincides with the semantics of the ontology if there are no rules; in addition, if the DL underlying the ontology language is polynomial, then $MKNF_{WF}$ retains a polynomial data complexity. Furthermore, $MKNF_{WF}$ is sound with respect to the semantics of [Motik and Rosati 2010] for MKNF knowledge bases (KBs), which is based on the Answer-Set Semantics and coincides with answer-sets of logic programs [Gelfond and Lifschitz 1991] if the ontology is empty.

In one sense, the fixpoint operator of $MKNF_{WF}$ provides a way to compute, in a naive bottom-up fashion, all consequences of a knowledge base. However, such an approach is impractical for large knowledge bases. Consider the medical case study above: knowledge of whether a specific patient is using a certain medication does not require knowledge of the medications of thousands of other patients. Thus, despite its polynomial complexity, bottom-up computation of $MKNF_{WF}$ does not scale to enterprise applications, much less to those of the Semantic Web. A query-driven procedure corresponding to the semantics of $MKNF_{WF}$ that only consults information relevant for a specific patient is clearly preferable.

This paper presents such a querying mechanism, called **SLG**(\mathcal{O}), that is sound and complete for $MKNF_{WF}$ [Knorr et al. 2011], and sound for MKNF knowledge bases of [Motik and Rosati 2010]. **SLG**(\mathcal{O}) accepts DL-safe conjunctive queries, i.e., conjunctions of predicates with variables where queries have to be ground before being processed by the DL reasoner, returning all correct answer substitutions for variables in the query. To the best of our knowledge, **SLG**(\mathcal{O}) is the first query-driven, top-down like procedure for knowledge bases that tightly combines an arbitrary decidable ontology language with non-monotonic rules.

SLG(\mathcal{O}) applies to any DL and under certain conditions maintains the data complexity of $MKNF_{WF}$. To show that these conditions are realistic, we also provide a concrete oracle, with practical usage, namely for \mathcal{EL}^+ . \mathcal{EL}^+ is a fragment of the light-weight description logics \mathcal{EL}^{++} , which is the DL underlying OWL 2 EL – one of the tractable profiles [Motik et al. 2009] of OWL 2 – and thus part of the W3C recommendations for the Semantic Web. We show that the oracle thus defined is correct with respect to the general procedure and maintains the polynomial data complexity of $MKNF_{WF}$ for such a polynomial DL.

The gist of the approach

The main element of our approach addresses the interdependency of the ontology and rules. In particular, **SLG**(\mathcal{O}), presented in Section 4, extends SLG resolution [Chen and Warren 1996], which evaluates queries posed to normal logic programs, i.e., sets of non-disjunctive non-monotonic rules, under WFS. SLG is a form of resolution that handles loops within the program, and does not change the data complexity of WFS. It does that by resorting to already computed results in a forest of derivation trees, a technique also known as *tabling*.

To adjoin an ontology to rules, the first thing that needs to be done is to allow an SLG evaluation to make calls to an inference engine for the ontology. Since MKNF is parametric on any given decidable ontology formalism,² the inference engine is

²In fact, theoretically the limitation to decidable ontology formalisms is not strictly needed, but it is a pragmatic choice to achieve termination and complexity results in accordance with decidable

viewed in $\mathbf{SLG}(\mathcal{O})$ as an oracle. In fact, every time $\mathbf{SLG}(\mathcal{O})$ selects an atom, the oracle's inference engine may be called, in case the atom is not provable by the rules alone. Such a queried atom, say $P(\mathbf{a})$, might thus be provable in the ontology but only if a certain set of atoms in turn is provable via rules. Our approach captures this by allowing the oracle to return a new rule, say $P(\mathbf{a}) \leftarrow Goals$, which has the property that a (possibly empty) set *Goals*, in addition to the axioms in the ontology and the atoms already derived from the combined knowledge base, would be sufficient to prove $P(\mathbf{a})$. $\mathbf{SLG}(\mathcal{O})$ then treats these new rules just as if they were part of the knowledge base.

Note that getting these conditional answers does not endanger decidability (or tractability, if it is the case) of reasoning in the ontology alone. In fact, it is easy to conceive of a modification of a tableau-based inference engine for an ontology that is capable of returning these conditional answers and is decidable if the tableau algorithm is. Simply add all the atoms that are defined in the rules to the ontology, then proceed with the tableau as usual, but collect all those added facts that have been used in the proof. Under some assumptions on the complexity of the oracle, it is shown (in Section 5 along with some other properties) that $\mathbf{SLG}(\mathcal{O})$ also retains the data complexity of $MKNF_{WF}$.

The second element of our approach arises from the need to properly combine the classical negation usually appearing in the ontology language with the non-monotonic negation of rules. This problem, which is solved by the semantics of [Knorr et al. 2011], is similar to the issue of *coherence* that arises when adding classical (or strong) negation to logic programs [Gelfond and Lifschitz 1991; Pearce and Wagner 1990; Pereira and Alferes 1992]: the classical negation must imply negation by default. In our case, if the ontology entails that some atom A is false, then perforce the default negation **not** A must hold as well. The derivation must accordingly be modified since the proof of **not** A cannot simply rely on the failure of the proof of A as it is usual in Logic Programming. For that purpose, an alternating fixpoint approach is used in the bottom-up construction defined in [Knorr et al. 2011], where two alternating fixpoint operators are applied to two different sets of rules. In each iteration step, the fixpoint construction alternates between deriving more true atoms, and more (default) false atoms; when deriving more true atoms, the original set of rules is used; when deriving more default false atoms, a transformed set of rules is used so that rules with head A are removed if $\neg A$ holds. This ensures that **not** A is derived (see Section 2.3 for details).

Adapting the alternating fixpoint for the top-down derivation would result in a procedure significantly different from the original SLG. So instead, we transform the original knowledge base to ensure coherence without the need for alternating between two sets of rules. This approach is simpler to understand as it is a more direct extension of SLG and separates the concerns of coherence from those of top-down derivation; in addition the transformational approach should also facilitate implementations of $\mathbf{SLG}(\mathcal{O})$. Indeed, one can rely more directly on the existing implementations that follow closely SLG.³ Accordingly, Section 3 defines the above

ontology languages, such as OWL [Hitzler et al. 2009].

³We have already made some experiments on the implementation of $\mathbf{SLG}(\mathcal{O})$ [Gomes et al. 2010], relying on the XSB-Prolog implementation.

mentioned transformation of the knowledge base. The transformation itself provides an alternative formulation of $MKNF_{WF}$ and is another result of the paper.

Finally, in Section 6, we provide a concrete oracle for \mathcal{EL}^+ . Our approach includes a preprocessing step that applies the subsumption algorithm⁴ for \mathcal{EL}^+ to compute all the subsumption relationships contained in the DL knowledge base and then remove redundant information with respect to answering queries. The resulting reduced DL knowledge base is then translated into rules and can be directly combined with the set of rules contained in the combined knowledge base, so that $\mathbf{SLG}(\mathcal{O})$ can be applied for querying. This direct integration of the oracle into the querying mechanism, as we show, immediately ensures that the data complexity of $MKNF_{WF}$ is maintained, i.e., the \mathcal{EL}^+ oracle is polynomial.

2. PRELIMINARIES

We assume a basic understanding of the Well-Founded Semantics [van Gelder et al. 1991] and first-order logics, in particular notions related to Logic Programming and resolution (see e.g. [Lloyd 1987]). In this section we recall basic concepts that we rely on in the following sections. In particular, we present description logics using the DL \mathcal{ALC} , the syntax of Hybrid MKNF knowledge bases, and their well-founded semantics.

2.1 Description Logics

We recall general notions for description logics, basing our examples on \mathcal{ALC} with role inclusions although our work is in principle applicable to any DL. Afterwards, since we present a concrete oracle for \mathcal{EL}^+ in Section 6, we also review the syntax of that DL. We refer to [Baader et al. 2007] for a general and thorough overview of description logics.

We start by recalling the syntax and semantics for a general DL \mathcal{DL} . DLs define *concept descriptions* inductively with the help of a set of constructors, starting with a set N_C of *concept names*, a set N_R of *role names*, and a set N_I of *individual names*. Concept descriptions of \mathcal{DL} are formed using a set of constructors, and the upper part of Table I shows the constructors of \mathcal{ALC} . There, and in general, we use a and b to denote individual names, R and S to denote role names, and C and D to denote concept descriptions (all possibly with indices).

The semantics of \mathcal{DL} -concept descriptions is defined in terms of an *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. The domain $\Delta^{\mathcal{I}}$ is a non-empty set of individuals and the *interpretation function* $\cdot^{\mathcal{I}}$ maps each concept name $A \in N_C$ to a subset $A^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$, each role name $R \in N_R$ to a binary relation $R^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$, and each individual name $a \in N_I$ to an individual $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. The extension of $\cdot^{\mathcal{I}}$ to arbitrary concept descriptions is inductively defined as shown in the third column of Table I for \mathcal{ALC} .

A \mathcal{DL} *TBox* \mathcal{T} is a finite set of *general concept inclusions* (GCIs) and possibly *role inclusions* (RIs), and both their syntax can be found in the middle of Table I. An interpretation is a *model* of a TBox \mathcal{T} if, for each GCI and RI in \mathcal{T} , the conditions given in the third column of Table I are satisfied. In the definition of the semantics of RIs, the symbol ' \circ ' denotes composition of binary relations.

⁴Such as the one included in Pellet (<http://clarkparsia.com/pellet/>) or CEL (<http://lat.inf.tu-dresden.de/systems/cel/>)

Table I. Syntax and semantics of \mathcal{ALC} with role inclusions.

Name	Syntax	Semantics
top	\top	$\Delta^{\mathcal{I}}$
bottom	\perp	\emptyset
negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y \in \Delta^{\mathcal{I}} : (x, y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}\}$
GCI	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
RI	$R_1 \circ \dots \circ R_k \sqsubseteq R$	$R_1^{\mathcal{I}} \circ \dots \circ R_k^{\mathcal{I}} \subseteq R^{\mathcal{I}}$
concept assertion	$C(a)$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
role assertion	$R(a, b)$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

A \mathcal{DL} ABox \mathcal{A} is a finite set of *concept assertions* for concept descriptions C and *role assertions* for role names R whose syntax can be found in the lower part of Table I. ABoxes are used to describe a snapshot or state of the world. An interpretation \mathcal{I} is a *model* of an ABox \mathcal{A} if, for each concept assertion and role assertion in \mathcal{A} , the conditions given in the third column of Table I are satisfied.

A \mathcal{DL} knowledge base \mathcal{O} consists of a \mathcal{DL} TBox \mathcal{T} and a \mathcal{DL} ABox \mathcal{A} , and \mathcal{I} is a *model* of \mathcal{O} if it is a model of both \mathcal{T} and \mathcal{A} .

One of the main inference problems in DLs, actually the one considered in [Baader et al. 2005] for \mathcal{EL}^{++} , is subsumption. Given two \mathcal{DL} -concept descriptions C, D we say that C is *subsumed by* D w.r.t. the TBox \mathcal{T} ($C \sqsubseteq_{\mathcal{T}} D$) iff $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for all models \mathcal{I} of \mathcal{T} . In addition, we recall the *instance problem* since, as we will see below, it is the reasoning task we are interested in when answering top-down queries in our system combining rules and an oracle to an ontology. An individual name a is an *instance of a concept* C in ABox \mathcal{A} w.r.t. a TBox \mathcal{T} if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} . Definition 2.1 extends the instance problem to instances of roles, a non-standard reasoning task.

Definition 2.1 A pair of individuals (a, b) is an *instance of a role* R in ABox \mathcal{A} w.r.t. a TBox \mathcal{T} if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} .

The above definition will be useful, since in the oracle we define queries for instances of roles, i.e., binary predicates, as well as concepts.

2.1.1 \mathcal{EL}^+ . The tractable DL \mathcal{EL}^+ is a large fragment⁵ of the DL \mathcal{EL}^{++} [Baader et al. 2005]. It is obtained by restricting the allowed concept constructors to $\top, \perp, \neg, \sqcap$, and \exists , i.e., negations, disjunctions, and value restrictions are not allowed. All the remaining notions, in particular the semantics, carries over from the general case.

⁵We omit nominals and concrete domains as concept constructors.

Two remarks regarding the expressivity of \mathcal{EL}^+ are in order. First, RIs allow expression of *role hierarchies* $R \sqsubseteq S$, *transitive roles* using $R \circ R \sqsubseteq R$, *right-identity rules* $R \circ S \sqsubseteq S$, and *left-identity rules* $S \circ R \sqsubseteq S$. Second, *disjointness* of complex concept descriptions C, D (and *unsatisfiability* of a concept C), can be expressed by $C \sqcap D \sqsubseteq \perp$ (resp. $C \sqsubseteq \perp$).

2.2 Syntax of Hybrid MKNF Knowledge Bases

Hybrid MKNF knowledge bases, as introduced in [Motik and Rosati 2010], are essentially formulas in the logics of minimal knowledge and negation as failure [Lifschitz 1991], i.e., first-order logics with equality and two modal operators, **K** and **not**, allowing inspection of the knowledge base. At an intuitive level, given a first-order formula φ , **K** φ asks whether φ is known, i.e., true in all models of the related Hybrid MKNF knowledge base \mathcal{K} , while **not** φ is used to check whether φ is not known. Hybrid MKNF knowledge bases consist of two components, a decidable DL knowledge base translatable into a first-order logic, and a finite set of rules.

Definition 2.2 Let \mathcal{O} be a DL knowledge base built over a language \mathcal{L} with distinguished sets of countably infinitely many variables N_V , and finitely many individuals N_I and predicates N_P , where $N_C \cup N_R \subseteq N_P$. An atom $P(t_1, \dots, t_n)$, where $P \in N_P$ and $t_i \in N_V \cup N_I$, is called a *DL-atom* if P occurs in \mathcal{O} , otherwise it is called *non-DL-atom*. An MKNF rule r has the following form where, for all i and j , H , A_i , and B_j are atoms:

$$H \leftarrow A_1, \dots, A_n, \text{not } B_1, \dots, \text{not } B_m. \quad (1)$$

H is called the (*rule*) *head*, and the sets $\{A_1, \dots, A_n\}$ and $\{\text{not } B_1, \dots, \text{not } B_m\}$ form the *body* of the rule. *Literals* are *positive literals* A or *negative literals* **not** B . We abbreviate rules by $H \leftarrow \mathcal{B}$, splitting \mathcal{B} into two sets \mathcal{B}^+ (positive literals) and \mathcal{B}^- (negative literals). A rule r is *positive* if $m = 0$ and a *fact* if $n = m = 0$. A *program* \mathcal{P} is a finite set of MKNF rules and called *positive* if all its rules are positive. A *Hybrid MKNF knowledge base* \mathcal{K} is a pair $(\mathcal{O}, \mathcal{P})$. The *ground instantiation* of \mathcal{K} is the KB $\mathcal{K}_G = (\mathcal{O}, \mathcal{P}_G)$ where \mathcal{P}_G is obtained from \mathcal{P} by replacing each rule r of \mathcal{P} with a set of rules substituting each variable in r with constants from \mathcal{K} in all possible ways.

In this definition and in the rest of the paper, we omit the modal operator **K** in rule heads and bodies.⁶

To ensure decidability, DL-safety is applied [Motik and Rosati 2010; Knorr et al. 2011]. Intuitively, DL-safety constrains the use of rules to individuals actually appearing in the knowledge base under consideration. Since, as indicated in the introduction, we are especially interested in querying the knowledge base, care also must be taken to impose DL-safety on (conjunctive) queries:

⁶The MKNF semantics in [Motik and Rosati 2010] and [Knorr et al. 2011] requires the presence of these modal operators to ensure that the interaction between the DL KB \mathcal{O} and the rules is limited to information that is known to hold. For our purposes, a simpler representation of models suffices, thus allowing us to simplify notation here.

Definition 2.3 An MKNF rule r is *DL-safe* if every variable in r occurs in at least one (positive) non-DL-atom in the body of r . A Hybrid MKNF knowledge base \mathcal{K} is *DL-safe* if all its rules are DL-safe.

A *DL-safe conjunctive query* q is a non-empty set, i.e., conjunction, of literals where each variable in q occurs in at least one (positive) non-DL-atom in q . We also write q as a rule $q(X_i) \leftarrow A_1, \dots, A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m$ where X_i is the (possibly empty) set of variables, appearing in the body.

This restriction of conjunctive queries to DL-safety is not always necessary: for DLs like SHIQ, conjunctive query answering is decidable [Glimm et al. 2008] and we may make use of existing algorithms. However, for DLs where there is no known algorithm for conjunctive query answering or where the problem is not decidable, such as full \mathcal{EL}^{++} [Rosati 2007], the limitation is required to achieve decidability in Hybrid MKNF knowledge bases. For simplicity of presentation, we impose the restriction throughout the paper.

Example 2.4 We present a small technical example to illustrate the notions introduced in this section. Consider the Hybrid MKNF knowledge base \mathcal{K} consisting of an \mathcal{EL}^+ KB \mathcal{O} containing two TBox statements and one assertion and a set of MKNF rules. Here and in the following examples we follow the convention that DL-atoms are capitalized, while non-DL-atoms start with lower case letters.

$$\begin{array}{ll} \mathbf{C} \sqsubseteq \mathbf{D} & \mathbf{C}(\mathbf{b}) \\ \mathbf{C} \sqcap \mathbf{E} \sqsubseteq \perp & \\ \mathbf{p}(\mathbf{x}) \leftarrow \mathbf{not} \mathbf{D}(\mathbf{x}), \mathbf{o}(\mathbf{x}) & \mathbf{o}(\mathbf{a}) \leftarrow \\ \mathbf{E}(\mathbf{x}) \leftarrow \mathbf{not} \mathbf{E}(\mathbf{x}), \mathbf{o}(\mathbf{x}) & \mathbf{o}(\mathbf{b}) \leftarrow \end{array}$$

The ground instantiation \mathcal{K}_G is obtained by grounding both rules with \mathbf{a} and \mathbf{b} . Note that the atom $\mathbf{o}(\mathbf{x})$ ensures that both (non-ground) rules are DL-safe.

2.3 Well-founded Semantics of Hybrid MKNF Knowledge Bases

In this section, we recall the computation of the well-founded MKNF model from [Knorr et al. 2011].⁷ We adopt that terminology here and recall the notions relevant for its definition. First, we present some notions from [Knorr et al. 2011] that are useful in the definition of the operators for obtaining that well-founded MKNF model.

Definition 2.5 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base. The set of known atoms of \mathcal{K} , $\mathbf{KA}(\mathcal{K})$, is the smallest set that contains (i) all positive literals occurring in \mathcal{P} , and (ii) a positive literal ξ for each negative literal $\mathbf{not} \xi$

⁷The well-founded MKNF semantics including the well-founded MKNF model, as presented in [Knorr et al. 2011], is based on a complete three-valued extension of the original MKNF semantics of [Motik and Rosati 2010]. In it, a model consists of two sets of sets of first-order interpretations; a 3-valued truth valuation is defined that exactly determines the semantics, and in which any MNKF formula can be evaluated. However, as here we are only interested in queries that are (conjunctions of) atoms, we limit ourselves to the computation of the literals that are true and false. This is called the well-founded partition in [Knorr et al. 2011] but we term it the well-founded MKNF model here.

occurring in \mathcal{P} . For a subset S of $\text{KA}(\mathcal{K})$, the *objective knowledge* of S w.r.t. \mathcal{K} is the set of first-order formulas $\text{OB}_{\mathcal{O},S} = \{\pi(\mathcal{O})\} \cup \{\xi \mid (\mathbf{K})\xi \in S\}$ where $\pi(\mathcal{O})$ is the first-order translation of \mathcal{O} .

Basically all literals appearing in the rules are collected in the set $\text{KA}(\mathcal{K})$ as a set of positive literals while the objective knowledge $\text{OB}_{\mathcal{O},S}$ provides a first-order representation of \mathcal{O} together with a set of known/derived facts without the implicit modal operator \mathbf{K} . For the computation of the three-valued MKNF model, the set $\text{KA}(\mathcal{K})$ can be divided into true, undefined, and false literals.

Example 2.6 Recall \mathcal{K} from Example 2.4 and its ground instantiation \mathcal{K}_G . Then $\text{KA}(\mathcal{K}_G) = \{\mathbf{p}(\mathbf{a}), \mathbf{p}(\mathbf{b}), \mathbf{D}(\mathbf{a}), \mathbf{D}(\mathbf{b}), \mathbf{E}(\mathbf{a}), \mathbf{E}(\mathbf{b}), \mathbf{o}(\mathbf{a}), \mathbf{o}(\mathbf{b})\}$.

We continue by defining an operator $T_{\mathcal{K}}$ that allows us to draw conclusions from positive Hybrid MKNF knowledge bases.

Definition 2.7 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a positive, ground Hybrid MKNF knowledge base. The operators $R_{\mathcal{K}}$, $D_{\mathcal{K}}$, and $T_{\mathcal{K}}$ are defined on subsets of $\text{KA}(\mathcal{K})$:

$$\begin{aligned} R_{\mathcal{K}}(S) &= \{H \mid \mathcal{P} \text{ contains a rule of the form } H \leftarrow A_1, \dots, A_n \\ &\quad \text{such that, for all } i, 1 \leq i \leq n, A_i \in S\} \\ D_{\mathcal{K}}(S) &= \{\xi \mid \xi \in \text{KA}(\mathcal{K}) \text{ and } \text{OB}_{\mathcal{O},S} \models \xi\} \\ T_{\mathcal{K}}(S) &= R_{\mathcal{K}}(S) \cup D_{\mathcal{K}}(S) \end{aligned}$$

$R_{\mathcal{K}}$ derives consequences from the rules while $D_{\mathcal{K}}$ obtains knowledge from the ontology \mathcal{O} together with the information in S .

The operator $T_{\mathcal{K}}$ is shown to be monotonic in [Knorr et al. 2011]. So, by the Knaster-Tarski theorem [Tarski 1955], it has a unique least fixpoint, denoted $\text{lfp}(T_{\mathcal{K}})$, which is reached after a finite number of iteration steps (since the ground knowledge base \mathcal{K} is always finite).

The computation of the well-founded MKNF model follows the alternating fixpoint construction [van Gelder 1989] of the well-founded semantics for logic programs. This construction requires a reduction that turns a Hybrid MKNF knowledge base into a positive one to make $T_{\mathcal{K}}$ applicable.

Definition 2.8 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base and $S \subseteq \text{KA}(\mathcal{K})$. The *MKNF transform* \mathcal{K}/S is defined as $\mathcal{K}/S = (\mathcal{O}, \mathcal{P}/S)$, where \mathcal{P}/S contains all rules $H \leftarrow A_1, \dots, A_n$ for which there exists a rule

$$H \leftarrow A_1, \dots, A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m$$

in \mathcal{P} with $B_j \notin S$ for all $1 \leq j \leq m$.

Example 2.9 Consider again \mathcal{K} from Example 2.4 and let S be $\text{KA}(\mathcal{K}_G)$. Then \mathcal{K}_G/S is obtained as follows:

$$\begin{array}{ll} \mathbf{C} \sqsubseteq \mathbf{D} & \mathbf{C}(\mathbf{b}) \\ \mathbf{C} \sqcap \mathbf{E} \sqsubseteq \perp & \\ \mathbf{o}(\mathbf{a}) \leftarrow & \mathbf{o}(\mathbf{b}) \leftarrow \end{array}$$

The resulting KB is positive and we may apply $T_{\mathcal{K}}$ and obtain $\{\mathbf{D}(\mathbf{b}), \mathbf{o}(\mathbf{a}), \mathbf{o}(\mathbf{b})\}$. Note that $\mathbf{C}(\mathbf{b})$ is not explicitly mentioned since it does not occur in $\mathbf{KA}(\mathcal{K}_G)$. It is nevertheless derivable from \mathcal{K}_G .

The MKNF transform resembles the well-known answer-set transformation [Gelfond and Lifschitz 1991] for logic programs. Based on it, an antitonic operator can be defined, but it is shown in [Knorr et al. 2011] that such an operator alone would not properly treat a problem called *coherence*, i.e., classical negation would not enforce default negation. Therefore, a second, slightly different transformation is introduced in [Knorr et al. 2011].

Definition 2.10 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base and $S \subseteq \mathbf{KA}(\mathcal{K})$. The *MKNF-coherent transform* $\mathcal{K} // S$ is defined as $\mathcal{K} // S = (\mathcal{O}, \mathcal{P} // S)$, where $\mathcal{P} // S$ contains all rules $H \leftarrow A_1, \dots, A_n$ for which there exists a rule

$$H \leftarrow A_1, \dots, A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m$$

in \mathcal{P} with $B_j \notin S$ for all $1 \leq j \leq m$ and $\mathbf{OB}_{\mathcal{O}, S} \not\models \neg H$.

Example 2.11 Consider again \mathcal{K} from Example 2.4 and let S be \emptyset . Then $\mathcal{K}_G // S$ is obtained:

$\mathbf{C} \sqsubseteq \mathbf{D}$	$\mathbf{C}(\mathbf{b})$
$\mathbf{C} \sqcap \mathbf{E} \sqsubseteq \perp$	$\mathbf{p}(\mathbf{a}) \leftarrow \mathbf{o}(\mathbf{a})$
$\mathbf{p}(\mathbf{b}) \leftarrow \mathbf{o}(\mathbf{b})$	$\mathbf{o}(\mathbf{a}) \leftarrow$
$\mathbf{E}(\mathbf{a}) \leftarrow \mathbf{o}(\mathbf{a})$	$\mathbf{o}(\mathbf{b}) \leftarrow$

The only rule that is removed is the one with head $\mathbf{E}(\mathbf{b})$, since $\mathbf{C}(\mathbf{b})$ holds and \mathbf{C} and \mathbf{E} are disjoint. Hence, $\mathbf{OB}_{\mathcal{O}, \emptyset} \models \neg \mathbf{E}(\mathbf{b})$. We can apply $T_{\mathcal{K}_G}$ to the resulting positive KB and obtain $\mathbf{KA}(\mathcal{K}_G) \setminus \{\mathbf{E}(\mathbf{b}), \mathbf{D}(\mathbf{a})\}$.

Note the difference between Definitions 2.8 and 2.10: we also remove a rule from the MKNF-coherent transform, in case the classical negation of the head is derivable from the ontology augmented by S .

These two transformations can now be used to define two operators for Hybrid MKNF KBs as already hinted in the examples.

Definition 2.12 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base and $S \subseteq \mathbf{KA}(\mathcal{K})$. We define:

$$\Gamma_{\mathcal{K}}(S) = \text{lfp}(T_{\mathcal{K}/S}) \quad \text{and} \quad \Gamma'_{\mathcal{K}}(S) = \text{lfp}(T_{\mathcal{K} // S}).$$

Both operators are shown to be antitonic [Knorr et al. 2011] and form the basis for defining the well-founded MKNF model. Here we present its alternating computation.

Definition 2.13 Let \mathcal{K} be a ground Hybrid MKNF knowledge base. We define:

$$\begin{aligned} \mathbf{P}_0 &= \emptyset & \mathbf{N}_0 &= \text{KA}(\mathcal{K}) \\ \mathbf{P}_{n+1} &= \Gamma_{\mathcal{K}}(\mathbf{N}_n) & \mathbf{N}_{n+1} &= \Gamma'_{\mathcal{K}}(\mathbf{P}_n) \\ \mathbf{P}_{\omega} &= \bigcup \mathbf{P}_i & \mathbf{N}_{\omega} &= \bigcap \mathbf{N}_i \end{aligned}$$

\mathbf{P}_{ω} contains everything that is necessarily true, while \mathbf{N}_{ω} contains everything that is not false. Note that, by finiteness of the ground knowledge base, the iteration stops before reaching ω . It was shown in [Knorr et al. 2011] that the sequences are monotonically increasing, decreasing respectively. The two fixpoints can also be used to detect whether a knowledge base is MKNF-consistent or not [Knorr et al. 2011].

Theorem 2.14 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base. \mathcal{K} is MKNF-inconsistent iff $\Gamma'_{\mathcal{K}}(\mathbf{P}_{\omega}) \subset \Gamma_{\mathcal{K}}(\mathbf{P}_{\omega})$ or $\Gamma'_{\mathcal{K}}(\mathbf{N}_{\omega}) \subset \Gamma_{\mathcal{K}}(\mathbf{N}_{\omega})$ or \mathcal{O} is inconsistent.

Intuitively, MKNF-consistency requires that \mathcal{O} is (first-order) consistent and that neither of the two additional conditions of Theorem 2.14 succeed.⁸ These two comparisons ensure that the two fixpoints never contain contradictions and that there is no rule such that the truth value of the (conjunction in the) body is greater than the truth value of the head.

Example 2.15 Consider the following KB \mathcal{K}_1 :

$$\mathbf{P}(\mathbf{a}) \leftarrow \text{not } \mathbf{P}(\mathbf{a}) \qquad \mathbf{Q}(\mathbf{a}) \leftarrow \qquad \mathbf{Q} \sqsubseteq \neg \mathbf{P}$$

\mathcal{K}_1 is MKNF-inconsistent, since $\mathbf{P}(\mathbf{a})$ is necessarily false, so the first rule violates the intuitive condition that the body should not have a higher truth value than the head. In fact, the computation yields that $\mathbf{P}(\mathbf{a})$ is true and false at the same time, and the test reveals that. Consider the KB \mathcal{K}_2 :

$$\begin{aligned} \mathbf{R} &\sqsubseteq \neg \mathbf{P} & \mathbf{R}(\mathbf{a}) \\ \mathbf{P}(\mathbf{a}) &\leftarrow \text{not } \mathbf{u} & \mathbf{u} \leftarrow \text{not } \mathbf{u} \end{aligned}$$

$\mathbf{P}(\mathbf{a})$ is false, but \mathbf{u} is undefined, so that we obtain a rule with false head and undefined body, and this is detected with the test.

If \mathcal{K} is MKNF-consistent, then the well-founded MKNF model exists.

Definition 2.16 The *well-founded MKNF model* M_{WF} of an MKNF-consistent, ground Hybrid MKNF knowledge base $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ is defined as

$$M_{WF} = \{A \mid A \in \mathbf{P}_{\omega}\} \cup \{\pi(\mathcal{O})\} \cup \{\text{not } B \mid B \in \text{KA}(\mathcal{K}) \setminus \mathbf{N}_{\omega}\}.$$

⁸The formal definition of MKNF-consistency from [Knorr et al. 2011] would require the complete material on three-valued MKNF semantics, which we want to avoid.

Example 2.17 Consider again the running Example 2.4. We provide the results of the computation.

$$\begin{array}{ll}
\mathbf{P}_0 = \emptyset & \mathbf{N}_0 = \text{KA}(\mathcal{K}_G) \\
\mathbf{P}_1 = \{\mathbf{D}(\mathbf{b}), \mathbf{o}(\mathbf{a}), \mathbf{o}(\mathbf{b})\} & \mathbf{N}_1 = \text{KA}(\mathcal{K}_G) \setminus \{\mathbf{E}(\mathbf{b}), \mathbf{D}(\mathbf{a})\} \\
\mathbf{P}_2 = \{\mathbf{D}(\mathbf{b}), \mathbf{o}(\mathbf{a}), \mathbf{o}(\mathbf{b}), \mathbf{p}(\mathbf{a})\} & \mathbf{N}_2 = \text{KA}(\mathcal{K}_G) \setminus \{\mathbf{E}(\mathbf{b}), \mathbf{D}(\mathbf{a}), \mathbf{p}(\mathbf{b})\} \\
\mathbf{P}_3 = \mathbf{P}_2 & \mathbf{N}_3 = \mathbf{N}_2
\end{array}$$

Thus we obtain the well-founded MKNF model as

$$M_{WF} = \{\pi(\mathcal{O})\} \cup \{\mathbf{D}(\mathbf{b}), \mathbf{o}(\mathbf{a}), \mathbf{o}(\mathbf{b}), \mathbf{p}(\mathbf{a})\} \cup \{\mathbf{not} \mathbf{E}(\mathbf{b}), \mathbf{not} \mathbf{D}(\mathbf{a}), \mathbf{not} \mathbf{p}(\mathbf{b})\}.$$

Consequently, $\mathbf{E}(\mathbf{a})$ is undefined.

To ease some proofs in the following section, we also adapt the notion of unfounded sets [van Gelder et al. 1991] for Hybrid MKNF which relates to the sequence \mathbf{N}_i (see [Knorr et al. 2011]). The essential advantage is that the reasons why a certain atom is considered false are better characterized. For that purpose, we first need to define a notion of dependency that captures more precisely the derivations from $\text{OB}_{\mathcal{O},S}$, for some S , by the operator $D_{\mathcal{K}}$.

Definition 2.18 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base, H an atom with $H \in \text{KA}(\mathcal{K})$, and S a (possibly empty) set of atoms with $S \subseteq \text{KA}(\mathcal{K})$. We say that H *depends* on S if and only if

- (i) $\text{OB}_{\mathcal{O},S} \models H$ and
- (ii) there is no S' with $S' \subset S$ such that $\text{OB}_{\mathcal{O},S'} \models H$.

Intuitively, S is a minimal set that, in combination with \mathcal{O} , allows us to derive H . Note that there may exist several such minimal sets. Based on this notion of dependency, the notion of an unfounded set can be extended to Hybrid MKNF KBs.

Definition 2.19 Let \mathcal{K} be a ground Hybrid MKNF knowledge base and (T, F) a pair of sets of atoms with $T, F \subseteq \text{KA}(\mathcal{K})$. We say that $U \subseteq \text{KA}(\mathcal{K})$ is an *unfounded set* (of \mathcal{K}) *with respect to* (T, F) if, for each atom $H \in U$, the following conditions are satisfied:

- (Ui) for each rule $H \leftarrow \mathcal{B}$ in \mathcal{P} at least one of the following holds.
 - (Uia) Some atom A appears in \mathcal{B} and in $U \cup F$.
 - (Uib) Some negative literal $\mathbf{not} B$ appears in \mathcal{B} and in T .
 - (Uic) $\text{OB}_{\mathcal{O},T} \models \neg H$
- (Uii) for each (possibly empty) S on which H depends, with $S \subseteq \text{KA}(\mathcal{K})$ and $\text{OB}_{\mathcal{O},S}$ consistent, there is at least one atom A such that $\text{OB}_{\mathcal{O},S \setminus \{A\}} \not\models H$ and A in $U \cup F$.

The union of all unfounded sets of \mathcal{K} w.r.t. (T, F) is called the *greatest unfounded set* of \mathcal{K} w.r.t. (T, F) and denoted $U_{\mathcal{K}}(T, F)$.

It can be shown that the computation of \mathbf{N}_i based on $\Gamma'_{\mathcal{K}}$ directly corresponds to the computation of the greatest unfounded set w.r.t. $(\mathbf{P}_{i-1}, \mathbf{N}_{i-1})$.

Example 2.20 Consider the computation in Example 2.17. All three atoms that were removed in the sequence of $\mathbf{N}_i \leftarrow \mathbf{E}(\mathbf{b}), \mathbf{D}(\mathbf{a}), \mathbf{p}(\mathbf{b})$ — obviously satisfy (Uii). However these removed atoms satisfy different conditions of (Ui). For $\mathbf{E}(\mathbf{b})$, (Uic) applies. In the case of $\mathbf{D}(\mathbf{a})$, there is no rule with this head, so (Ui) is vacuously true. Finally, for $\mathbf{p}(\mathbf{b})$, (Uib) applies because **not** $\mathbf{D}(\mathbf{b})$ occurs in the single rule for this atom, while $\mathbf{D}(\mathbf{b})$ is true.

3. ALTERNATIVE COMPUTATION OF $MKNF_{WF}$

As presented in Section 2, the bottom-up computation of the well-founded MKNF model requires essentially two operators each with its own transformation of the knowledge base. Using the operators directly would make the top-down procedure quite different from the original SLG procedure, which operates on a single logic program, and does not differentiate between the two phases of the alternating fix-point. To approximate the bottom-up computation to the SLG procedure, in this section we define that computation in a different way. Namely, we transform the original knowledge base, by doubling the rules and the ontology in \mathcal{K} using new predicates, and transform both so that a single operator and copy of the KB can be used. As we shall see, a simpler bottom-up computation, with a single operator, performed over this single transformed knowledge base yields the same results as the one defined in Section 2, and in particular still guarantees that classical negation enforces default negation.

The first definition introduces two new special predicates for each predicate appearing in \mathcal{K} based on which the transformation that doubles a knowledge base \mathcal{K} is defined.

Definition 3.1 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a Hybrid MKNF knowledge base. We introduce new predicates A^d and NA for each predicate A appearing in \mathcal{K} , and then constructively define

- (1) \mathcal{O}^d by substituting each predicate A in \mathcal{O} by A^d ; and
- (2) \mathcal{P}^d by transforming each rule

$$H(\vec{t}_H) \leftarrow A_1(\vec{t}_{A1}), \dots, A_n(\vec{t}_{An}), \mathbf{not} B_1(\vec{t}_{B1}), \dots, \mathbf{not} B_m(\vec{t}_{Bm})$$

occurring in \mathcal{P} into two rules:

- (2a) $H(\vec{t}_H) \leftarrow A_1(\vec{t}_{A1}), \mathbf{not} B_1^d(\vec{t}_{B1}), \dots, \mathbf{not} B_m^d(\vec{t}_{Bm})$ and either
- (2b.i) $H^d(\vec{t}_H) \leftarrow A_1^d(\vec{t}_{A1}), \dots, A_n^d(\vec{t}_{An}), \mathbf{not} B_1(\vec{t}_{B1}), \dots, \mathbf{not} B_m(\vec{t}_{Bm}),$
 $\mathbf{not} NH(\vec{t}_H)$ if $H(\vec{t}_H)$ is a DL-atom; or
- (2b.ii) $H^d(\vec{t}_H) \leftarrow A_1^d(\vec{t}_{A1}), \dots, A_n^d(\vec{t}_{An}), \mathbf{not} B_1(\vec{t}_{B1}), \dots, \mathbf{not} B_m(\vec{t}_{Bm})$
if $H(\vec{t}_H)$ is a non-DL-atom.

We define the *doubled Hybrid MKNF knowledge base* $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$.

Intuitively, we use an atom based on the original predicate A to represent truth of A in the original knowledge base, while the atom based on a newly introduced predicate A^d represents non-falsity of A in the original knowledge base, i.e., if we want to know whether some atom is (non-monotonically) false, then we query using the auxiliary predicate. The new atom $NH(\vec{t}_H)$ appearing in (2b.i), is used as a marker to distinguish between rules that may be affected by the derivability of the

classical negation of its head (as in Γ'_K) and the others (as in Γ_K). Note that this process of doubling the knowledge base has no impact on the (at best) polynomial data complexity of computing the well-founded MKNF model since it only alters the computation by the constant factor 2.

Example 3.2 Consider \mathcal{K} from Example 2.4. We obtain \mathcal{K}^d as follows.

$$\begin{array}{ll}
 \mathcal{C} \sqsubseteq \mathcal{D} & \mathcal{C}^d \sqsubseteq \mathcal{D}^d \\
 \mathcal{C} \sqcap \mathcal{E} \sqsubseteq \perp & \mathcal{C}^d \sqcap \mathcal{E}^d \sqsubseteq \perp \\
 \mathcal{C}(\mathbf{b}) & \mathcal{C}^d(\mathbf{b}) \\
 \mathbf{p}(\mathbf{x}) \leftarrow \text{not } \mathcal{D}^d(\mathbf{x}), \mathbf{o}(\mathbf{x}) & \mathbf{p}^d(\mathbf{x}) \leftarrow \text{not } \mathcal{D}(\mathbf{x}), \mathbf{o}^d(\mathbf{x}) \\
 \mathcal{E}(\mathbf{x}) \leftarrow \text{not } \mathcal{E}^d(\mathbf{x}), \mathbf{o}(\mathbf{x}) & \mathcal{E}^d(\mathbf{x}) \leftarrow \text{not } \mathcal{E}(\mathbf{x}), \mathbf{o}^d(\mathbf{x}), \text{not } \mathcal{N}\mathcal{E}(\mathbf{x}) \\
 \mathbf{o}(\mathbf{a}) \leftarrow & \mathbf{o}^d(\mathbf{a}) \leftarrow \\
 \mathbf{o}(\mathbf{b}) \leftarrow & \mathbf{o}^d(\mathbf{b}) \leftarrow
 \end{array}$$

Only the rule with head $\mathcal{E}^d(\mathbf{x})$ contains the marker in the body as the original rule is the only one in \mathcal{K} with a DL-atom in the head. Note that the atoms based on the predicate \mathbf{o} , which ensure DL-safety, could be excluded from the doubling for efficiency reasons.

The marker has to be referenced in the modified transform but, before that, we define a slightly different operator for doubled, positive Hybrid MKNF knowledge bases that takes into account the parallel computations on the two renamings of the ontology.

Definition 3.3 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, positive, ground Hybrid MKNF knowledge base. The operators $R_{\mathcal{K}^d}$, $D_{\mathcal{K}^d}$, and $T_{\mathcal{K}^d}$ are defined on subsets of $\text{KA}(\mathcal{K}^d)$ as follows:

$$\begin{aligned}
 R_{\mathcal{K}^d}(S) &= \{H \mid \mathcal{P}^d \text{ contains a rule of the form } H \leftarrow A_1, \dots, A_n \\
 &\quad \text{such that, for all } i, 1 \leq i \leq n, A_i \in S\} \\
 D_{\mathcal{K}^d}(S) &= \{\xi \mid \xi \in \text{KA}(\mathcal{K}) \text{ and } \text{OB}_{\mathcal{O}, S} \models \xi\} \cup \\
 &\quad \{\xi \mid \xi \in \text{KA}(\mathcal{K}^d) \setminus \text{KA}(\mathcal{K}) \text{ and } \text{OB}_{\mathcal{O}^d, S} \models \xi\} \\
 T_{\mathcal{K}^d}(S) &= R_{\mathcal{K}^d}(S) \cup D_{\mathcal{K}^d}(S)
 \end{aligned}$$

These operator definitions are the same as those of Definition 2.7 apart from two differences. First, the doubled knowledge base \mathcal{K}^d is considered and, consequently, atoms from $\text{KA}(\mathcal{K}^d)$ appear. Second, the operator $D_{\mathcal{K}^d}$ computes consequences from \mathcal{O} and \mathcal{O}^d in parallel but limited to the corresponding set of atoms appearing in each of the two renamings, thus preventing an inconsistency, e.g., in \mathcal{O} , from affecting the consistency of \mathcal{O}^d .

Next, we present a slightly altered version of the MKNF-coherent transform (cf. Definition 2.10) taking into account the doubled Hybrid MKNF knowledge base and the new negative literals of the form $\mathcal{N}H(\vec{t}_H)$ that serve as markers.

Definition 3.4 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, ground Hybrid MKNF knowledge base and $S \subseteq \text{KA}(\mathcal{K}^d)$. The *MKNF^d-coherent transform* $\mathcal{K}^d // S$ is defined as $\mathcal{K}^d // S = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d // S)$, where $\mathcal{P}^d // S$ contains all rules $H \leftarrow A_1, \dots, A_n$ for which there exists a rule

$$H \leftarrow A_1, \dots, A_n, \text{not } B_1, \dots, \text{not } B_m$$

in \mathcal{P}^d with

- (1) $B_j \notin S$ for all $1 \leq j \leq m$; and
- (2) $\text{OB}_{\mathcal{O}, S} \not\models \neg H_1(t_{H_1}^{\vec{\tau}})$ if $\text{not } NH_1(t_{H_1}^{\vec{\tau}})$ appears in the body, where $H = H_1^d(t_{H_1}^{\vec{\tau}})$.

This definition is almost identical to the transformation of Definition 2.10 with the only difference that the removal due to classical negation is only possible in marked rules. Given Definition 3.1, this means that only rules whose head is a DL-atom and built by means of a doubled predicate may be eliminated that way (case 2b.i of Definition 3.1). Additionally, as we will see in Section 4, the marker itself can be used to actually trigger a query to the ontology for the classical negation of the atom in the head (using the original predicate for the atom).

We can now define a new operator $\Gamma_{\mathcal{K}}^d$ for ground knowledge bases \mathcal{K} similar to the ones in Definition 2.12, but that operates on atoms of $\text{KA}(\mathcal{K}^d)$.

Definition 3.5 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, ground Hybrid MKNF knowledge base and $S \subseteq \text{KA}(\mathcal{K}^d)$. We define:

$$\Gamma_{\mathcal{K}^d}(S) = \text{lfp}(T_{\mathcal{K}^d // S}).$$

We can show that this operator is antitonic just as its two predecessors.

Lemma 3.6 Let \mathcal{K}^d be a doubled, ground Hybrid MKNF knowledge base and $S_1 \subseteq S_2 \subseteq \text{KA}(\mathcal{K}^d)$. Then $\Gamma_{\mathcal{K}^d}(S_2) \subseteq \Gamma_{\mathcal{K}^d}(S_1)$.

PROOF. We have to show that $\text{lfp}(T_{\mathcal{K}^d // S_2}) \subseteq \text{lfp}(T_{\mathcal{K}^d // S_1})$. Since \mathcal{K} is finite, \mathcal{K}^d is also finite, and we prove by induction on n that $T_{\mathcal{K}^d // S_2} \uparrow n \subseteq T_{\mathcal{K}^d // S_1} \uparrow n$ holds.

The base case for $n = 0$ is trivial since $\emptyset \subseteq \emptyset$.

Assume that $T_{\mathcal{K}^d // S_2} \uparrow n \subseteq T_{\mathcal{K}^d // S_1} \uparrow n$ holds, consider $H \in T_{\mathcal{K}^d // S_2} \uparrow (n+1)$. Then $H \in T_{\mathcal{K}^d // S_2}(T_{\mathcal{K}^d // S_2} \uparrow n)$ and there are two cases to consider:

- (1) $\mathcal{K}^d // S_2$ contains a rule of the form $H \leftarrow A_1, \dots, A_n$ such that $A_i \in T_{\mathcal{K}^d // S_2} \uparrow n$ for each $1 \leq i \leq n$. In this case, since $S_1 \subseteq S_2$ holds, we also have $H \leftarrow A_1, \dots, A_n$ in $\mathcal{K}^d // S_1$ and, by the induction hypothesis, $A_i \in T_{\mathcal{K}^d // S_1} \uparrow n$ holds for each $1 \leq i \leq n$. Hence, $H \in T_{\mathcal{K}^d // S_1} \uparrow (n+1)$.
- (2) H is a consequence obtained from $D_{\mathcal{K}^d}$. But $D_{\mathcal{K}^d}$ derives only consequences from the unchanged DL renamings \mathcal{O} and \mathcal{O}^d together with $T_{\mathcal{K}^d // S_2} \uparrow n$. By the induction hypothesis, we conclude that $H \in T_{\mathcal{K}^d // S_1} \uparrow (n+1)$.

This finishes the proof. \square

Since this new operator is antitonic, we can define its iteration similar to Definition 2.13, but now with just one operator.

Definition 3.7 Let \mathcal{K}^d be a doubled, ground Hybrid MKNF knowledge base. We define:

$$\begin{aligned} \mathbf{P}_0^d &= \emptyset & \mathbf{N}_0^d &= \text{KA}(\mathcal{K}^d) \\ \mathbf{P}_{n+1}^d &= \Gamma_{\mathcal{K}^d}(\mathbf{N}_n^d) & \mathbf{N}_{n+1}^d &= \Gamma_{\mathcal{K}^d}(\mathbf{P}_n^d) \\ \mathbf{P}_\omega^d &= \bigcup \mathbf{P}_n^d & \mathbf{N}_\omega^d &= \bigcap \mathbf{N}_n^d \end{aligned}$$

The correspondence between Definitions 2.13 and 3.7 can be established with a precise relation between the atoms in the doubled knowledge base \mathcal{K}^d and those in \mathcal{K} . To ease the proof of the corresponding property, we rely on an adaptation of the notion of unfounded sets to doubled Hybrid MKNF KBs. For that purpose, we also adapt the notion of dependency.

Definition 3.8 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, ground Hybrid MKNF knowledge base, H an atom with $H \in \text{KA}(\mathcal{K}^d)$, and S a (possibly empty) set of atoms with $S \subseteq \text{KA}(\mathcal{K}^d)$. We say that H *depends* on S if and only if, for $\mathcal{O}' = \mathcal{O}$ or $\mathcal{O}' = \mathcal{O}^d$:

- (i) $\text{OB}_{\mathcal{O}', S} \models H$ and
- (ii) there is no S' with $S' \subset S$ such that $\text{OB}_{\mathcal{O}', S'} \models H$.

Based on this notion of dependency, the notion of an unfounded set for Hybrid MKNF is extended from Definition 2.19 to include \mathcal{O} and \mathcal{O}^d of the doubled KB.

Definition 3.9 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, ground Hybrid MKNF knowledge base and (T, F) a pair of sets such that $T, F \subseteq \text{KA}(\mathcal{K}^d)$. We say that $U \subseteq \text{KA}(\mathcal{K}^d)$ is an *unfounded set* (of \mathcal{K}^d) *with respect to* (T, F) if, for each atom $H \in U$, the following conditions are satisfied:

- (Ui) for each rule $H \leftarrow \mathcal{B}$ in \mathcal{P} at least one of the following holds.
 - (Uia) Some atom A appears in \mathcal{B} and in $U \cup F$.
 - (Uib) Some negative literal **not** B appears in \mathcal{B} and in T .
 - (Uic) $\text{OB}_{\mathcal{O}, T} \models \neg H_1(t_{H_1})$ and **not** $NH_1(t_{H_1}) \in \mathcal{B}$, where $H = H_1(\vec{t}_{H_1})$
- (Uii) for each (possibly empty) S on which H depends, with $S \subseteq \text{KA}(\mathcal{K})$ and $\text{OB}_{\mathcal{O}, S}$ consistent, there is at least one atom A such that $\text{OB}_{\mathcal{O}, S \setminus \{A\}} \not\models H$ and A in $U \cup F$.
- (Uii^d) for each (possibly empty) S on which H depends, with $S \subseteq \text{KA}(\mathcal{K}^d)$ and $\text{OB}_{\mathcal{O}^d, S}$ consistent, there is at least one atom A such that $\text{OB}_{\mathcal{O}^d, S \setminus \{A\}} \not\models H$ and A in $U \cup F$.

The union of all unfounded sets of \mathcal{K}^d w.r.t. (T, F) is called the *greatest unfounded set* of \mathcal{K}^d w.r.t. (T, F) and denoted $U_{\mathcal{K}^d}(T, F)$.

Of course, (Uii^d) is just a copy of (Uii) to deal with \mathcal{O}^d , the copy of \mathcal{O} .

The correspondence to the sequence \mathbf{N}_i^d for all i can now be established.

Lemma 3.10 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled, ground Hybrid MKNF knowledge base and $(\mathbf{P}_i^d, \mathbf{N}_i^d)$ a pair of sets such that $\mathbf{P}_i^d, \mathbf{N}_i^d \in \text{KA}(\mathcal{K}^d)$ in the computation of the alternating fixpoint of \mathcal{K}^d (Definition 3.7). Then the following holds:

$$\text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d = U_{\mathcal{K}^d}(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$$

PROOF. We show both inclusions from which the equality follows.

$\text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d \subseteq U_{\mathcal{K}^d}(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$: Let $H \in \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d$. Then $H \notin \mathbf{N}_{i+1}^d$, i.e., $H \notin \Gamma_{\mathcal{K}^d}(\mathbf{P}_i^d)$ and $H \notin \text{lfp}(T_{\mathcal{K}^d // \mathbf{P}_i^d})$. Thus, two conditions hold. First, for all rules of the form $H \leftarrow \mathcal{B}^+ \wedge \mathcal{B}^-$ in \mathcal{P}^d , there is at least one $A \in \mathcal{B}^+$ with $A \in \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d$, or at least one **not** $B \in \mathcal{B}^-$ with $B \in \mathbf{P}_i^d$, or $\text{OB}_{\mathcal{O}, \mathbf{P}_i} \models \neg H_1(\vec{t}_{H_1})$ and **not** $H_1(\vec{t}_{H_1}) \in \mathcal{B}^-$, where $H = H_1^d(\vec{t}_{H_1})$. Second, neither $\text{OB}_{\mathcal{O}, \mathbf{N}_{i+1}^d} \models H$ nor $\text{OB}_{\mathcal{O}^d, \mathbf{N}_{i+1}^d} \models H$ holds. The first condition corresponds exactly to (Ui) of Definition 3.9 w.r.t. $(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$. We derive from the second condition that, for all S with $S \subseteq \text{KA}(\mathcal{K}^d)$ on which H depends, there is at least one atom A such that $\text{OB}_{\mathcal{O}, S \setminus \{A\}} \not\models H$, respectively $\text{OB}_{\mathcal{O}^d, S \setminus \{A\}} \not\models H$, and A in $\text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d$. This matches condition (Uii) (resp. condition (Uii^d) of Definition 3.9 w.r.t. $(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$, and we conclude that $H \in U_{\mathcal{K}^d}(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$.

$U_{\mathcal{K}^d}(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d) \subseteq \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d$: Let $H \in U_{\mathcal{K}^d}(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$. Then H occurs in the greatest unfounded set w.r.t. $(\mathbf{P}_i^d, \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_i^d)$. It follows from Definition 3.9 that $H \notin \mathbf{N}_{i+1}^d$. Consequently, $H \in \text{KA}(\mathcal{K}^d) \setminus \mathbf{N}_{i+1}^d$. \square

We can now show the correspondence between atoms of \mathcal{K} in the fixed point of Definition 2.13 and those of \mathcal{K}^d in the fixed point of Definition 3.7.

Proposition 3.11 *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground Hybrid MKNF knowledge base. Then the following holds:*

- $A \in \mathbf{P}_\omega$ if and only if $A \in \mathbf{P}_\omega^d$.
- $B \notin \mathbf{N}_\omega$ if and only if $B^d \notin \mathbf{N}_\omega^d$.

PROOF. We show by induction on n that two conditions hold.

- (i) $A \in \mathbf{P}_n$ if and only if $A \in \mathbf{P}_n^d$
- (ii) $B \notin \mathbf{N}_n$ if and only if $B^d \notin \mathbf{N}_n^d$

This is sufficient since the grounded knowledge base is finite, which means that the iteration is finite and stops for some natural number n , i.e., the two fixpoints coincide on the relevant atoms as in (i) and (ii).

The base case for $n = 0$ is straightforward since \mathbf{P}_0 and \mathbf{P}_0^d are empty while \mathbf{N}_0 and \mathbf{N}_0^d both contain their entire Herbrand base.

(1) So, suppose that (i) and (ii) hold for n and let $A \in \mathbf{P}_{n+1}$ and $B \notin \mathbf{N}_{n+1}$. We show that $A \in \mathbf{P}_{n+1}^d$ and $B^d \notin \mathbf{N}_{n+1}^d$. The other direction of the equivalence follows from an identical argument.

- (i) First, suppose that $A \in \mathbf{P}_{n+1}$ but $A \notin \mathbf{P}_n$ (otherwise we obtain the result by the induction hypothesis (1) immediately). We show that $A \in \mathbf{P}_{n+1}^d$. If $A \in \mathbf{P}_{n+1}$, then $A \in \Gamma_{\mathcal{K}}(\mathbf{N}_n)$, by Definition 2.13, and, thus, $A \in \text{lfp}(T_{\mathcal{K}/\mathbf{N}_n})$ by Definition 2.12. So $A \in T_{\mathcal{K}/\mathbf{N}_n} \uparrow m$ for some m and we show by induction on m that $A \in T_{\mathcal{K}^d // \mathbf{N}_n} \uparrow m$ (2), which implies that $A \in \mathbf{P}_{n+1}^d$. The base case for $m = 0$ holds immediately. Assume the claim (2) holds for m , we show it for $m + 1$. Suppose that $A \in T_{\mathcal{K}/\mathbf{N}_n} \uparrow (m + 1)$ then $A \in T_{\mathcal{K}/\mathbf{N}_n}(T_{\mathcal{K}/\mathbf{N}_n} \uparrow m)$. Then either $A \in R_{\mathcal{K}/\mathbf{N}_n}(T_{\mathcal{K}/\mathbf{N}_n} \uparrow m)$ or $A \in D_{\mathcal{K}/\mathbf{N}_n}(T_{\mathcal{K}/\mathbf{N}_n} \uparrow m)$. We start with the first case, i.e., there is a rule $A \leftarrow A_1, \dots, A_n, \text{not } B_1, \dots, \text{not } B_m$ with

$A_i \in T_{\mathcal{K}/\mathbf{N}_n} \uparrow m$ and **not** $B_j \notin \mathbf{N}_n$ for all i and j . For each such rule $A \leftarrow A_1, \dots, A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m$ in \mathcal{K} there is, according to Definition 3.1, a rule $A \leftarrow A_1, \dots, A_n, \mathbf{not} B_1^d, \dots, \mathbf{not} B_m^d$ in \mathcal{P}^d . Since, by the induction hypothesis (1), we have that $B_i \notin \mathbf{N}_n$ if and only if $B_i^d \notin \mathbf{N}_n^d$ we obtain that each rule in \mathcal{K}/\mathbf{N}_n has its correspondent in $\mathcal{K}^d//\mathbf{N}_n^d$. We obtain by the nested induction hypothesis of (2) that $A \in T_{\mathcal{K}^d//\mathbf{N}_n^d} \uparrow (m+1)$. Otherwise, $A \in D_{\mathcal{K}/\mathbf{N}_n}(T_{\mathcal{K}/\mathbf{N}_n} \uparrow m)$ holds, and $A \in D_{\mathcal{K}^d//\mathbf{N}_n^d}(T_{\mathcal{K}^d//\mathbf{N}_n^d} \uparrow m)$ is obtained immediately by the induction hypothesis (2) and the identical ontologies \mathcal{O} contained in \mathcal{K}^d and \mathcal{K} .

(ii) To prove (ii) we suppose as well that $B \notin \mathbf{N}_{n+1}$ but $B \in \mathbf{N}_n$. We show that $B^d \notin \mathbf{N}_{n+1}^d$. If $B \notin \mathbf{N}_{n+1}$, then $B \in U_{\mathcal{K}}(\mathbf{P}_n, \mathbf{KA}(\mathcal{K}) \setminus \mathbf{N}_n)$. By Definitions 3.9, and 3.1, we obtain that $B \in U_{\mathcal{K}^d}(\mathbf{P}_n^d, \mathbf{KA}(\mathcal{K}^d) \setminus \mathbf{N}_n^d)$. Hence, by Lemma 3.10, $B \notin \mathbf{N}_{n+1}^d$.

This finishes the proof. \square

It follows immediately from this proposition that we can use this alternative computation to compute the well-founded MKNF model. Formally we obtain the following theorem, which shows the adapted well-founded MKNF model.

Theorem 3.12 *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a ground, MKNF-consistent Hybrid MKNF knowledge base and let $\mathbf{P}_{\mathcal{K}}^d, \mathbf{N}_{\mathcal{K}}^d \subseteq \mathbf{KA}(\mathcal{K}^d)$ with*

$$\mathbf{P}_{\mathcal{K}}^d = \{A \mid A \in \mathbf{P}_{\omega}^d \text{ and } A \in \mathbf{KA}(\mathcal{K})\}, \mathbf{N}_{\mathcal{K}}^d = \{A^d \mid A^d \in \mathbf{N}_{\omega}^d \text{ and } A^d \in \mathbf{KA}(\mathcal{K}^d)\}.$$

Then

$$M_{WF} = \{A \mid A \in \mathbf{P}_{\mathcal{K}}^d\} \cup \{\pi(\mathcal{O})\} \cup \{\mathbf{not} A \mid A^d \in (\mathbf{KA}(\mathcal{K}^d) \setminus \mathbf{KA}(\mathcal{K})) \setminus \mathbf{N}_{\mathcal{K}}^d\}$$

is the well-founded MKNF model of \mathcal{K} .

PROOF. The result is an immediate consequence of Proposition 3.11 and Definition 2.16. \square

The two sets $\mathbf{P}_{\mathcal{K}}^d$ and $\mathbf{N}_{\mathcal{K}}^d$ are just used to remove superfluous atoms, e.g., the atoms based on doubled predicates for $\mathbf{P}_{\mathcal{K}}^d$. We note that for practical purposes we also derive from this theorem and Proposition 3.11 that we have to use the new predicates A^d if we query for negative literals.

To better illustrate how each of the two computations work, we finish the section with a technical example.

Example 3.13 Consider the knowledge base \mathcal{K} .

$$\begin{aligned} Q &\sqsubseteq \neg R \\ p(a) &\leftarrow \mathbf{not} p(a) \\ Q(a) &\leftarrow \\ R(a) &\leftarrow \mathbf{not} R(a) \end{aligned}$$

We now show how both computation work in this example, yielding (as expected from Proposition 3.11) the same results.

We can compute the two sequences \mathbf{P}_i and \mathbf{N}_i and obtain:

$$\begin{array}{ll}
\mathbf{P}_0 = \emptyset & \mathbf{N}_0 = \{p(a), q(a), r(a)\} \\
\mathbf{P}_1 = \{q(a)\} & \mathbf{N}_1 = \mathbf{N}_0 \\
\mathbf{P}_2 = \mathbf{P}_1 & \mathbf{N}_2 = \{p(a), q(a)\} \\
\mathbf{P}_3 = \{p(a), q(a), r(a)\} & \mathbf{N}_3 = \mathbf{N}_2 \\
\mathbf{P}_4 = \mathbf{P}_3 & \mathbf{N}_4 = \emptyset
\end{array}$$

The knowledge base is obviously MKNF-inconsistent since we derive that everything is true and false at the same time.

Now we apply the alternative computation using the doubled set of rules \mathcal{P}^d and the ontology \mathcal{O} and its renaming \mathcal{O}^d including the special marker predicates NR and NQ .

$$\begin{array}{ll}
Q \sqsubseteq \neg R & Q^d \sqsubseteq \neg R^d \\
p(a) \leftarrow \text{not } p^d(a) & p^d(a) \leftarrow \text{not } p(a) \\
Q(a) \leftarrow & Q^d(a) \leftarrow \text{not } \text{NQ}(a) \\
R(a) \leftarrow \text{not } R^d(a) & R^d(a) \leftarrow \text{not } R(a), \text{not } \text{NR}(a)
\end{array}$$

We compute the two sequences for the transformed knowledge base \mathcal{K}^d and obtain:

$$\begin{array}{ll}
\mathbf{P}_0^d = \emptyset & \mathbf{N}_0^d = \{p(a), p^d(a), q(a), q^d(a), r(a), r^d(a), \text{NQ}(a), \text{NR}(a)\} \\
\mathbf{P}_1^d = \{q(a), q^d(a)\} & \mathbf{N}_1^d = \mathbf{N}_0^d \\
\mathbf{P}_2^d = \mathbf{P}_1^d & \mathbf{N}_2^d = \{p(a), p^d(a), q(a), q^d(a), r(a)\} \\
\mathbf{P}_3^d = \{p(a), q(a), r(a)\} & \mathbf{N}_3^d = \mathbf{N}_2^d \\
\mathbf{P}_4^d = \mathbf{P}_3^d & \mathbf{N}_4^d = \{p(a), q(a), r(a)\}
\end{array}$$

All atoms based on original predicates are true while all doubled atoms are false. This indicates again that the knowledge base is MKNF-inconsistent.

Note that the inconsistency in R ensures that everything in the knowledge base is considered inconsistent. This does not always hold (consider adding a fact $p(a) \leftarrow$ to the rules, then $p^d(a)$ is not false but true). However, whenever we encounter an atom such that P is true while P^d is false, then we know that the KB is MKNF-inconsistent. Adapting Theorem 2.14 to this alternative computation of the well-founded MKNF model is not trivial, since the computation is now more intertwined. But this does not constitute a problem. The purpose of this computation is to provide a link to top-down querying, where, for reasons of efficiency, we do not want to test whether the entire KB is MKNF-consistent: we only consider the portion of the KB used in the derivation of the considered query.

4. TABLED SLG(\mathcal{O})-RESOLUTION FOR HYBRID MKNF

We present **SLG(\mathcal{O})** for Hybrid MKNF knowledge bases which extends SLG resolution from [Chen and Warren 1996] with an oracle to capture first-order deduction in DLs. SLG evaluation models well-founded computation for logic programs at an operational level, ensuring goal-directedness, termination and optimal complexity

for a large class of programs (cf. [Chen and Warren 1996]). At the same time it has motivated the design of modern tabling engines, and captures many aspects of their behavior. When SLG is extended with an oracle in $\mathbf{SLG}(\mathcal{O})$, several of the definitions of SLG are affected. In this section we present the definitions of $\mathbf{SLG}(\mathcal{O})$, as well as defining when an oracle is suitable for use in an evaluation. As the $\mathbf{SLG}(\mathcal{O})$ definitions are presented, we make clear how they differ from those of SLG. For the definition of $\mathbf{SLG}(\mathcal{O})$, we follow and extend the model of [Swift 1999].

Briefly, an $\mathbf{SLG}(\mathcal{O})$ evaluation is a sequence of forests (sets) of program trees. Program trees themselves correspond to subgoals that have been encountered in an evaluation. The nodes in these trees contain sets of literals divided into those literals that have not been examined, and others that have been examined, but their resolution delayed (cf. Definition 4.2). The need to delay some literals arises for the following reason. Modern Prolog engines rely on a fixed order for selecting literals in a rule, e.g., always left-to-right. However, well-founded computations cannot be performed using a fixed-order literal selection function.⁹ Hence, in $\mathbf{SLG}(\mathcal{O})$ the DELAY operation may postpone evaluation of some literals, which may be later resolved through an operation called SIMPLIFICATION. In addition to modeling the operational behavior of Prolog, the use of delay and simplification supports the termination and complexity results of $\mathbf{SLG}(\mathcal{O})$ discussed in Section 5, analogous to those presented for SLG in [Chen and Warren 1996].

Example 4.1 To ease the understanding of $\mathbf{SLG}(\mathcal{O})$, we present a concrete example of an $\mathbf{SLG}(\mathcal{O})$ evaluation that does not use an oracle. Consider the following Hybrid MKNF knowledge base \mathcal{K} with empty \mathcal{O} .

$$p(b) \leftarrow \quad (2)$$

$$p(c) \leftarrow \mathbf{not} p(a) \quad (3)$$

$$p(X) \leftarrow t(X, Y, Z), \mathbf{not} p(Y), \mathbf{not} p(Z) \quad (4)$$

$$p(a) \leftarrow p(b), p(a) \quad (5)$$

$$t(a, a, b) \leftarrow \quad (6)$$

$$t(a, b, a) \leftarrow \quad (7)$$

We consider the query $p(c)$ to \mathcal{K} in which none of the atoms is a DL-atom, i.e., no oracle needs to be used. The $\mathbf{SLG}(\mathcal{O})$ forest at the end of this evaluation is shown in Figure 1 where each node is labeled with a number indicating the order in which it was created in the $\mathbf{SLG}(\mathcal{O})$ evaluation. Nodes consist of either the symbol *fail*, or of a head representing the bindings made to an atomic subgoal and a body with a set of *Delays*, followed by the $|$ symbol, followed by *Goals* that are still to be examined. The evaluation begins by creating a tree for the initial query with root $p(c) \leftarrow | p(c)$ in node 1. Children of root nodes are created via the operation PROGRAM CLAUSE RESOLUTION just as in the SLD resolution of Prolog.

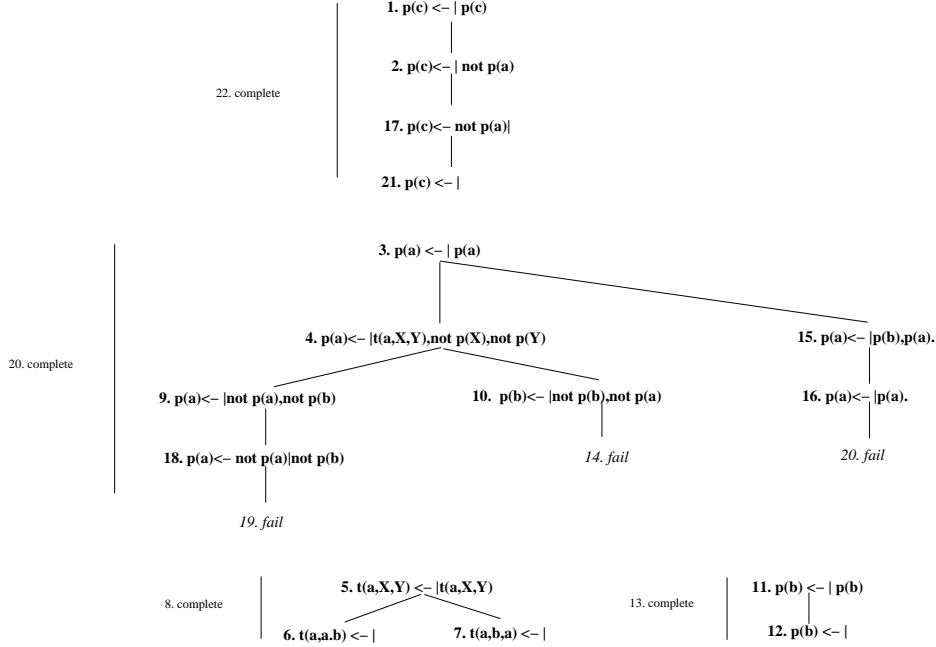
⁹A literal selection function is employed to choose the next literal to resolve in the body of a rule. In $\mathbf{SLG}(\mathcal{O})$, the only requirement for a selection function is that DL-atoms are not selected until they are ground, which is always possible given DL-safety of conjunctive queries and the rules appearing in the knowledge base (cf. Definition 2.3).

Accordingly, the evaluation uses rule (3) to create node 2. The (only possible) literal **not** $p(a)$ in node 2 is selected. This literal has an underlying subgoal $p(a)$ that does not correspond to the root of any tree in the forest so far. Thus, the $\mathbf{SLG}(\mathcal{O})$ operation NEW SUBGOAL creates a new tree for $p(a)$ (node 3), whose child, node 4, is created by PROGRAM CLAUSE RESOLUTION using rule (4). The NEW SUBGOAL operation is again used to create a new tree for the selected literal $t(a, x, y)$ (node 5), and children nodes 6 and 7 are created by PROGRAM CLAUSE RESOLUTION from rules (6) and (7). These latter nodes have empty *Goals* and are termed *answers*; moreover, since they also have empty *Delays*, they are *unconditional* answers.¹⁰ Any atom in the ground instantiation of an unconditional answer is true in the well-founded MKNF model, cf. Theorem 5.3. The $\mathbf{SLG}(\mathcal{O})$ operation POSITIVE RETURN is used to resolve the first of these answers against the selected literal of node 4, producing node 9. The selected literal of this latter node has $p(a)$ as its underlying subgoal, but there is already a tree for $p(a)$ in the forest and there are no answers for $p(a)$ to return. Since there is another unconditional answer for $t(a, x, y)$ (in node 7), POSITIVE RETURN can be used to produce node 10. The underlying subgoal $p(b)$ is selected, by NEW SUBGOAL the tree for $p(b)$ is created, and it is eventually determined that the subgoal $p(b)$ has an unconditional answer (node 12); accordingly, using the NEGATION FAILURE operation, the *failure node*, node 14, is created. Then, the computation, via PROGRAM CLAUSE RESOLUTION and program rule (5), produces another child for $p(a)$, node 15, and resolves $p(b)$ (node 16). At this stage the subgoal $p(a)$ is neither true, as no unconditional answers have been derived for it, nor false as one of its possible derivations, node 9, effectively has a loop through negation. However, in $\mathbf{SLG}(\mathcal{O})$ it is possible to apply the DELAY operation to the selected negative literal, by moving it from the *Goals* to the right of the $|$ symbol into the *Delays* to the left of the $|$ symbol. This DELAY operation produces node 17, which is termed a *conditional* answer, as it has empty *Goals* but non-empty *Delays*.¹¹ DELAY also produces node 18 whose new selected literal **not** $p(b)$ now fails (given the unconditional answer in node 12), producing the failure node 19. At this stage, all possible operations for non-answer nodes in $p(a)$ and the trees it depends on have been performed so that $p(a)$ may be *completed* (step 20). The completed subgoal $p(a)$ has no answers, and so is termed *failed* and is false in the well-founded MKNF model of \mathcal{K} . This failed literal can be removed from the delay list of node 18 through the SIMPLIFICATION operation producing the unconditional answer node 21.

Example 4.1 covers most of the main aspects of $\mathbf{SLG}(\mathcal{O})$, more precisely the main aspect of the underlying formalism, SLG, that is applicable to normal logic programs. SLG does not especially differ from other Prolog-like tabling formalisms in the case of programs that do not use default negation (**not**). However, for nega-

¹⁰In a practical program, a predicate defined by simple facts would not be evaluated using tabling, but rather would use SLD resolution as in Prolog.

¹¹Choosing delay in this order is not optimal and is made for purposes of illustrating the operations of $\mathbf{SLG}(\mathcal{O})$. This does not affect the result of the query itself since $\mathbf{SLG}(\mathcal{O})$ is shown to be confluent in Theorem 5.2.

Fig. 1. Final forest for the query $p(c)$ to P_1 .

tion it introduces the concept of delaying literals in order to be able to find witnesses of failure anywhere in a rule, along with the concept of simplifying these delayed literals whenever their truth value becomes known. **SLG**(\mathcal{O}) allows additionally the usage of an oracle to incorporate reasoning in the DL part \mathcal{O} , and we present its definitions in the following.

An **SLG**(\mathcal{O}) evaluation proceeds by constructing a forest according to the set of **SLG**(\mathcal{O}) operations. Such a forest, and the trees and nodes it contains are defined as follows:

Definition 4.2 A *node* has the form

$$\text{AnswerTemplate} \leftarrow \text{Delays} | \text{Goals} \quad \text{or} \quad \text{fail}.$$

In the first form, *AnswerTemplate* is an atom or a classically negated atom, while *Delays* and *Goals* are sequences of literals. The second form is called a *failure node*. A *program tree* T is a tree of nodes whose root is of the form $S \leftarrow | S$ for some atom S or a classically negated atom $S = \neg S_1$: we call S *the root node for* T and T *the tree for* S . An **SLG**(\mathcal{O}) *forest* \mathcal{F} is a set of program trees. A node N is an *answer* when it is a leaf node for which *Goals* is empty. If the *Delays* of an answer is empty, it is termed an *unconditional answer*, otherwise, it is a *conditional answer*. A program tree T may be marked with the symbol *complete*.

The notions in Definition 4.2 are almost identical to previous formulations of SLG resolution. The only difference is that **SLG**(\mathcal{O}) allows for the appearance of classically negated atoms as roots to incorporate possible calls for the classical

negation as required by the bottom-up computation in Definition 3.4. Such a literal $\neg A$ only appears in an *AnswerTemplate* or as the only goal in the root node, and is only used to query \mathcal{O} .

An **SLG**(\mathcal{O}) evaluation of a query Q starts with an initial forest with just one node $Q \leftarrow |Q$ and creates a sequence of forests. Each forest is obtained from the previous one by applying one **SLG**(\mathcal{O}) operation. If no further **SLG**(\mathcal{O}) operation is applicable, then the final forest for the evaluation of the query has been reached. We introduce these **SLG**(\mathcal{O}) operations incrementally, in Definitions 4.5, 4.7, 4.11, and 4.14. But before we present the first set of operations, we need two auxiliary definitions.

The definition of answer resolution in **SLG**(\mathcal{O}) (and SLG) differs from resolution in Horn rules in order to take into account delay literals in conditional answers.

Definition 4.3 Let N be a node $A \leftarrow D|L_1, \dots, L_n$, where $n > 0$. Let $Ans = A' \leftarrow D'|$ be an answer whose variables are disjoint from N . N is **SLG**(\mathcal{O}) *resolvable* with Ans if $\exists i, 1 \leq i \leq n$, such that L_i and A' are unifiable with an mgu¹² θ . The **SLG**(\mathcal{O}) resolvent of N and Ans on L_i has the form:

$$(A \leftarrow D|L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n)\theta$$

if D' is empty; otherwise the resolvent has the form:

$$(A \leftarrow D, L_i|L_1, \dots, L_{i-1}, L_{i+1}, \dots, L_n)\theta$$

Note that this form of resolution delays L_i rather than propagating the answer's delay list D' . This is necessary, as shown in [Chen and Warren 1996], to ensure polynomial data complexity.¹³

Next, we relate different types of literals to their underlying subgoals.

Definition 4.4 The *underlying subgoal* of L is 1) L if L is a positive literal or $L = \neg S$; 2) is S if $L = \mathbf{not} S$ (and S is not based on one of the new predicates NH introduced in Definition 3.4); or 3) is $\neg H(t_H^{\vec{t}})$ if $L = \mathbf{not} NH(t_H^{\vec{t}})$.

The first set of operations that we present deals with the creation of new trees and with resolution with program rules and with answers in other trees.

Definition 4.5 (SLG(\mathcal{O}) Operations – 1) Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base. Further assume that a fixed selection function is used to select a literal from the *Goals* in a node.

Given a forest \mathcal{F}_n of an SLG(\mathcal{O}) evaluation of \mathcal{K}^d , \mathcal{F}_{n+1} may be produced by one of the following operations.

- (1) NEW SUBGOAL: Let \mathcal{F}_n contain a tree with non-root node

$$N = Ans \leftarrow Delays|G, Goals$$

¹²most general unifier

¹³If delay lists were propagated directly, then delay lists could contain all derivations which could be exponentially many in the worst case.

- where S is the underlying subgoal of G . Assume \mathcal{F}_n contains no tree with root S . Then add the tree $S \leftarrow |S$ to \mathcal{F}_n .
- (2) **PROGRAM CLAUSE RESOLUTION**: Let \mathcal{F}_n contain a tree with root node $N = S \leftarrow |S$ and C be a rule $Head \leftarrow Body$ such that $Head$ unifies with S with mgu θ . Assume that in \mathcal{F}_n , N does not have a child $N_{child} = (S \leftarrow |Body)\theta$. Then add N_{child} as a child of N .
- (3) **POSITIVE RETURN**: Let \mathcal{F}_n contain a tree with non-root node N whose selected literal S is positive. Let Ans be an answer for S in \mathcal{F}_n and N_{child} be the **SLG**(\mathcal{O}) resolvent of N and Ans on S . Assume that in \mathcal{F}_n , N does not have a child N_{child} . Then add N_{child} as a child of N .

As illustrated in Example 4.1, the operation **NEW SUBGOAL** creates a new tree in the forest \mathcal{F} for a selected literal in the *Goals* of some (non-root) node in a tree in \mathcal{F} . Once a root node N for a positive literal is created, the **PROGRAM CLAUSE RESOLUTION** operation can create children for N , given the rules in the knowledge base. **POSITIVE RETURN** resolves positive literals in nodes, with answers already in the forest, according to Definition 4.3. Contrary to **SLG**, the **NEW SUBGOAL** operation may also create new trees for classically negated literals to which only the operation **ORACLE RESOLUTION**, defined below, applies.

Now, if a sequence of **SLG**(\mathcal{O}) operations yields a (possibly intermediate) forest containing an unconditional answer, then this answer is considered to be true. Likewise, if no more operations are applicable to a set of trees, and none of them contains an unconditional answer, i.e., the set of literals associated to these trees is completely evaluated (see Definition 4.12), then we can interpret all these literals as false. Expanding on this correspondence, we may associate an **SLG**(\mathcal{O}) forest with a partial interpretation, taking into consideration that, besides atoms and default negated atoms, **SLG**(\mathcal{O}) also allows classically negated literals as the roots of trees. This interpretation is shown to correspond to M_{WF} (cf. Theorem 5.3 below).

Definition 4.6 Let \mathcal{F} be a forest. Then the *interpretation induced by \mathcal{F}* , $I_{\mathcal{F}}$, is the smallest set such that:

- A (ground) atom $A \in I_{\mathcal{F}}$ iff A is in the ground instantiation of some unconditional answer $Ans \leftarrow |$ in \mathcal{F} .
- A (ground) negated atom $\neg A \in I_{\mathcal{F}}$ iff $\neg A$ is in the ground instantiation of some unconditional answer $Ans \leftarrow |$ in \mathcal{F} .
- A (ground) literal **not** $A \in I_{\mathcal{F}}$ iff A is in the ground instantiation of a literal whose tree in \mathcal{F} is marked as complete, and A is not in the ground instantiation of any answer in a tree in \mathcal{F} .

An atom S is *successful* (resp. *failed*) in \mathcal{F} if S' (resp. **not** S') is in $I_{\mathcal{F}}$ for every S' in the ground instantiation of S .

Whenever an atom A is successful, we can fail its default negation **not** A . If an atom A is failed, then we can simplify away **not** A . Ground default negated literals that are neither failed nor successful may be delayed and be simplified later. More precisely:

Definition 4.7 (SLG(\mathcal{O}) Operations – 2) Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base, and assume a selection function as in Definition 4.5.

Given a forest \mathcal{F}_n of an SLG(\mathcal{O}) evaluation of \mathcal{K}^d , \mathcal{F}_{n+1} may be further produced by one of the following operations.

- (4) **NEGATIVE RETURN:** Let \mathcal{F}_n contain a tree with a leaf node, whose selected literal **not** S is ground

$$N = \text{Ans} \leftarrow \text{Delays} | \text{not } S, \text{Goals}.$$

- (a) **NEGATION SUCCESS:** If S is failed in \mathcal{F}_n then create a child for N of the form: $\text{Ans} \leftarrow \text{Delays} | \text{Goals}$.
 (b) **NEGATION FAILURE:** If S succeeds in \mathcal{F}_n , then create a child for N of the form *fail*.

- (5) **DELAYING:** Let \mathcal{F}_n contain a tree with leaf node

$$N = \text{Ans} \leftarrow \text{Delays} | \text{not } S, \text{Goals}$$

such that S is ground in \mathcal{F}_n , but S is neither successful nor failed in \mathcal{F}_n . Then create a child for N of the form $\text{Ans} \leftarrow \text{Delays}, \text{not } S | \text{Goals}$.

- (6) **SIMPLIFICATION:** Let \mathcal{F}_n contain a tree with leaf node

$$N = \text{Ans} \leftarrow \text{Delays} |$$

and let $L \in \text{Delays}$

- (a) If L is failed in \mathcal{F} then create a child *fail* for N .
 (b) If L is successful in \mathcal{F} , then create a child $\text{Ans} \leftarrow \text{Delays}' |$ for N , where $\text{Delays}' = \text{Delays} - L$.

In Hybrid MKNF knowledge bases, an atom S is true if it is derivable from the rules or from the DL part of the knowledge base. So far, we have presented the operation **PROGRAM CLAUSE RESOLUTION** that handles the former case. We now introduce the **ORACLE RESOLUTION** operation to deal with the latter.

The next definition characterizes the behavior of an abstract oracle, \mathcal{O} ,¹⁴ that computes entailment according to the DL knowledge base \mathcal{O} , to be used in the **ORACLE RESOLUTION** operation. For that purpose, we define an oracle transition function that, given an interpretation induced by a forest, computes in a single step all possible atoms required to prove a goal S . In other words, such an oracle, when presented with S and a forest \mathcal{F} , non-deterministically returns in one step a set of ground atoms \mathcal{L} such that: for each $L \in \mathcal{L}$ there is at least one rule with L in the head in ground \mathcal{P}_G , and if \mathcal{L} were added to \mathcal{O} augmented with $I_{\mathcal{F}}$, the extended theory would immediately entail S . We only have to take into account that we appropriately query \mathcal{O} or its renaming \mathcal{O}^d in a doubled Hybrid MKNF knowledge base, and that we extend \mathcal{O} only with the positive part of $I_{\mathcal{F}}$.

Definition 4.8 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base, S a ground goal, \mathcal{L} a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable

¹⁴We overload \mathcal{O} syntactically to represent the oracle and the ontology, i.e., its underlying DL knowledge base, since from the viewpoint of **SLG(\mathcal{O})** they are the same.

with at least one rule head in \mathcal{P}^d , and $I_{\mathcal{F}}^+ = I_{\mathcal{F}} \setminus \{\text{not } A \mid \text{not } A \in I_{\mathcal{F}}\}$. The *complete oracle* for \mathcal{O} , denoted $\text{comp}T_{\mathcal{O}}$, is defined by

$$\text{comp}T_{\mathcal{O}}(I_{\mathcal{F}}, S, \mathcal{L}) \text{ iff } \mathcal{O} \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \models S \text{ or } \mathcal{O}^d \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \models S$$

Example 4.9 Consider the Hybrid MKNF knowledge base \mathcal{K} containing \mathcal{O} .

$$\mathbf{C(a)} \qquad \qquad \qquad \mathbf{C \sqcap F \sqsubseteq E}$$

Assume that $I_{\mathcal{F}}$ is empty, and that there is at least one rule whose head unifies with $\mathbf{F(a)}$. We query for $\mathbf{E(a)}$. In this case, $\text{comp}T_{\mathcal{O}}(\emptyset, \mathbf{E(a)}, \{\mathbf{F(a)}\})$ holds because $\mathcal{O} \cup \{\mathbf{F(a)}\} \models \mathbf{E(a)}$. Thus, deriving $\mathbf{F(a)}$ from the rules would be enough to conclude that $\mathbf{E(a)}$ is true in the well-founded MKNF model.

The set $\mathcal{O} \cup I_{\mathcal{F}}^+ \cup \mathcal{L}$ (and likewise $\mathcal{O}^d \cup I_{\mathcal{F}}^+ \cup \mathcal{L}$) may be inconsistent even though the well-founded MKNF model of \mathcal{K} exists. Consequently, such a complete oracle potentially allows us to obtain a large number of entailments that are eventually useless to derive S if \mathcal{K} is MKNF-consistent.

Example 4.10 Consider the Hybrid MKNF knowledge base \mathcal{K} containing \mathcal{O} .

$$\mathbf{C(a)} \qquad \qquad \mathbf{C \sqsubseteq \neg D} \qquad \qquad \mathbf{E \sqsubseteq F}$$

Assume that $I_{\mathcal{F}}$ is empty and we query for $\mathbf{E(a)}$. If \mathcal{O} were extended with $\mathbf{D(a)}$, \mathcal{O} would become inconsistent, so that all statements would be derivable from the extended \mathcal{O} , including $\mathbf{E(a)}$. Hence $\text{comp}T_{\mathcal{O}}(\emptyset, \mathbf{E(a)}, \{\mathbf{D(a)}\})$ holds because $\mathcal{O} \cup \{\mathbf{D(a)}\}$ is inconsistent. However, as \mathcal{K} is MKNF-consistent, $\mathbf{D(a)}$ cannot be derived so that the corresponding tree eventually fails. In Section 5, we provide the definition of a partial oracle which overcomes this lack of efficiency, and upon which concrete oracles can be based.

Complete oracles are applied to define the next **SLG**(\mathcal{O}) operation, which has no correspondence in SLG:

Definition 4.11 (SLG(\mathcal{O}) Operations – 3) Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base. Given a forest \mathcal{F}_n of an **SLG**(\mathcal{O}) evaluation of \mathcal{K}^d , \mathcal{F}_{n+1} may be produced by:

- (7) ORACLE RESOLUTION: Let \mathcal{F}_n contain a tree with root node $N = S \leftarrow |S$, and suppose that $\text{comp}T_{\mathcal{O}}(I_{\mathcal{F}_n}, S, \text{Goals})$ holds. Assume that N does not have a child $N_{\text{child}} = S \leftarrow |\text{Goals}$ in \mathcal{F}_n .¹⁵ Then add N_{child} as a child of N .

SLG(\mathcal{O}) also includes an operation that marks a set of trees as complete if the corresponding set of literals is completely evaluated. Completed trees can be used in **SLG**(\mathcal{O}) to simplify other trees and to augment the interpretation associated with the forest with default negated literals

Definition 4.12 A set \mathcal{S} of literals in a forest \mathcal{F} is *completely evaluated* if at least one of the conditions holds for each $S \in \mathcal{S}$

¹⁵For that comparison, we consider the sequences *Goals* as sets to avoid that one root node has several children whose sequences *Goals* are merely permutations.

- (1) The tree for S contains an answer $S \leftarrow |$; or
- (2) For each node N in the tree for S :
 - (a) The underlying subgoal of the selected literal of N is marked as complete; or
 - (b) The underlying subgoal of the selected literal of N is in \mathcal{S} and there are no applicable NEW SUBGOAL, PROGRAM CLAUSE RESOLUTION, POSITIVE RETURN (Definition 4.5), NEGATIVE RETURN, DELAYING (Definition 4.7) or ORACLE RESOLUTION (Definition 4.11) operations for N .

Once a set of literals is determined to be completely evaluated, a COMPLETION operation marks the trees for each literal (Definition 4.2). If a subgoal S is completed due to condition 1 holding, we say that S is *early completed*. If condition 1 does not hold, condition 2a of the above definition prevents the COMPLETION operation from being applied to one of a set of trees if certain other operations are applicable to those trees. This notion of completion is incremental in the sense that once a set \mathcal{S} of mutually dependent subgoals is fully evaluated, the derivation need not be concerned with the trees for \mathcal{S} apart from any answers they contain. In an actual implementation resources for such trees can be reclaimed.

In certain cases the propagation of conditional answers through resolution (Definition 4.3) can lead to a set of *unsupported answers* — conditional answers that are false in the well founded model (see, e.g., Example 1 of [Swift et al. 2009]).¹⁶ Intuitively, these answers, which have positive mutual dependencies, correspond to an unfounded set, but their technical definition is based on the form of conditional answers.

Definition 4.13 Let \mathcal{F} be an $\mathbf{SLG}(\mathcal{O})$ forest, and $Answer$ be an atom that occurs in the head of some answer in a tree with root S . Then $Answer$ is supported in \mathcal{F} if and only if:

- (1) S is not completely evaluated; or
- (2) there exists an answer node $Answer' \leftarrow Delays|$ in S such that $Answer'$ subsumes $Answer$ and for every positive literal $L \in Delays$, L is supported in \mathcal{F} .

We are now able to characterize the last two operations of $\mathbf{SLG}(\mathcal{O})$: one allows the completion of trees, and the other removes unsupported answers.

Definition 4.14 ($\mathbf{SLG}(\mathcal{O})$ Operations – 4) Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base. Given a forest \mathcal{F}_n of an $\mathbf{SLG}(\mathcal{O})$ evaluation of \mathcal{K}^d , \mathcal{F}_{n+1} may also be produced by one of the following operations.

- (8) COMPLETION: Given a completely evaluated set \mathcal{S} of literals (Definition 4.12), mark the trees for all literals in \mathcal{S} as complete.
- (9) ANSWER COMPLETION: Given a set of unsupported answers \mathcal{UA} , create a failure node as a child for each answer $Ans \in \mathcal{UA}$.

¹⁶As an aside, we note that unsupported answers appear to be uncommon in practical evaluations which minimize the use of delay such as [Sagonas et al. 2000].

Each of the operations (1)–(9), in Definitions 4.5, 4.7, 4.11 and 4.14, can be seen as a function that associates a forest with a new forest by adding a new tree, adding a new node to an existing tree, or marking a set of trees as complete. The only thing missing to complete the description of the procedure is the formalization of the initialization of an **SLG**(\mathcal{O}) evaluation, i.e., how the initial (DL-safe) conjunctive query is defined.

Definition 4.15 Let \mathcal{K}^d be a doubled Hybrid MKNF knowledge base and let q be a query of the form $q(X_i) \leftarrow A_1, \dots, A_n, \text{not } B_1^d, \dots, \text{not } B_m^d$ where X_i is the (possibly empty) set of requested variables. We set $\mathcal{F}_0 = \{q(X_i) \leftarrow q(X_i)\}$ to be the initial forest of an **SLG**(\mathcal{O}) evaluation of \mathcal{K}^d for q and add q itself to \mathcal{K}^d .

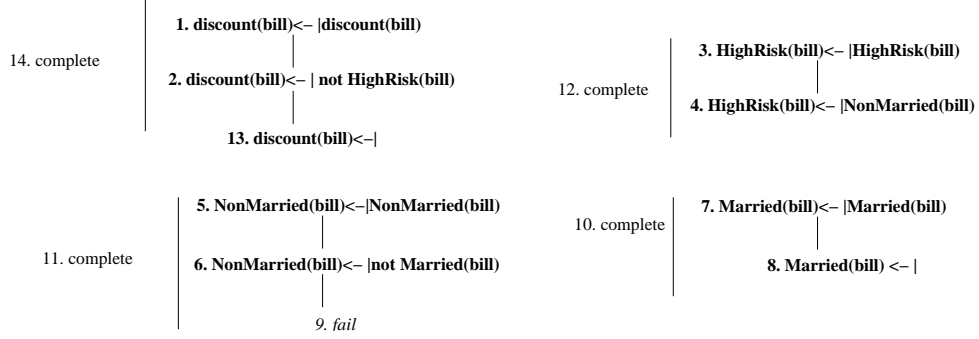
Of course, if the query is atomic we can simply start with the query itself, i.e., with the root containing the queried literal itself. Since the derivation uses \mathcal{K}^d (the doubled knowledge base), the technically correct way to query negative literals is to use **not** B^d instead of **not** B for any atom B which is why we use the doubled predicates for negative literals in the query.

Finally, note that if \mathcal{O} represents an expressive DL, then \mathcal{O} may derive equalities between different individuals because the unique names assumption (UNA) is not applied. Hybrid MKNF accounts for that using the standard names assumption (see [Motik and Rosati 2010; Knorr et al. 2011]), thus adapting reasoning with equalities as well. As such, equalities allow us to derive further information in the sense that, for example, if $\mathcal{C}(\mathbf{a})$ and $\mathbf{a} \approx \mathbf{b}$ hold, then $\mathcal{C}(\mathbf{b})$ is derivable. If $\mathcal{C}(\mathbf{a})$ is a DL-atom, then the DL reasoner of the oracle takes care of the problem internally. Only if $\mathcal{C}(\mathbf{a})$ is a non-DL-atom, then we specifically have to query for equalities in \mathcal{O} which is why ORACLE RESOLUTION is not restricted to DL-atoms.

In the next section, we show that **SLG**(\mathcal{O}) always terminates (Theorem 5.1) and, even though some orders of application of the possible operations are more efficient than others, that the procedure is confluent (Theorem 5.2). We also show that the procedure is sound and complete w.r.t. the well-founded MKNF model (Theorem 5.3) and that it is sound w.r.t. the semantics of two-valued MKNF (Corollary 5.4). Finally, under some assumptions, we maintain the computational complexity of the bottom-up procedure (Theorem 5.7), which is actually an improvement since we do not have to consider the entire knowledge base but only the part relevant for a concrete query. But before showing these results, we finish the presentation of **SLG**(\mathcal{O}) with an example illustrating its behavior.

Example 4.16 In order to illustrate the actions of **SLG**(\mathcal{O}) we consider a derivation of an answer to the query **discount**(Bill) using a KB \mathcal{K} from [Motik and Rosati 2007].¹⁷

¹⁷For ease of reading and since neither an MKNF-inconsistency nor an issue related to coherence occurs, we operate on \mathcal{K} directly instead of on \mathcal{K}^d .

Fig. 2. Final Forest for the query `discount(Bill)` to \mathcal{K} .

$$\text{NonMarried} \equiv \neg \text{Married} \quad (8)$$

$$\neg \text{Married} \sqsubseteq \text{HighRisk} \quad (9)$$

$$\exists \text{Spouse.T} \sqsubseteq \text{Married} \quad (10)$$

$$(\exists \text{Spouse.}\{\text{Michelle}\})(\text{Bill}) \quad (11)$$

$$\text{NonMarried}(x) \leftarrow \text{not Married}(x) \quad (12)$$

$$\text{discount}(x) \leftarrow \text{not HighRisk}(x) \quad (13)$$

First, note that TBox and ABox information are each distributed over both the DL KB and the rules. Figure 2 shows the final forest for this evaluation, where elements are marked in the order they are created. The initial forest for the evaluation consists of node 1 only. Given the selected literal of node 1, `discount(Bill)`, we can only apply PROGRAM CLAUSE RESOLUTION, so we use rule (13) to produce node 2, followed by NEW SUBGOAL to produce node 3. No rules are applicable for node 3, `HighRisk(Bill)`, but an ORACLE RESOLUTION operation can be applied to derive from axioms (8) and (9) that if `NonMarried(Bill)` can be proven (node 4), then this suffices to prove `HighRisk(Bill)`. Then, via a NEW SUBGOAL operation, node 5 is obtained. For the selected literal in node 5, `NonMarried(Bill)`, PROGRAM CLAUSE RESOLUTION produces node 6 from (12) and NEW SUBGOAL produces node 7. The selected literal of node 7, `Married(Bill)`, is not the head of a rule, so the only possibility is to use ORACLE RESOLUTION, and the answer `Married(Bill)` is derived from axioms (10) and (11). Using this answer, the tree for `Married(Bill)` can be early completed and a NEGATIVE RETURN operation produces node 10. The tree for `NonMarried(Bill)`, which does not have an answer, must be completed (step 11), and the same holds for `HighRisk(Bill)` (step 12). Once this occurs, a NEGATIVE RETURN operation is enabled to produce node 13.

The evaluation in the example illustrates two main points. First, the evaluation makes use of classical negation in the ontology along with closed world negation in the rules. Second, the actions of the DL part and the program part are interleaved, with the program “calling” the oracle by ORACLE RESOLUTION, and the oracle “calling” the program back with the answers of that operation.

5. PROPERTIES OF TABLED $\mathbf{SLG}(\mathcal{O})$ -RESOLUTION

We now present several properties of $\mathbf{SLG}(\mathcal{O})$ -resolution. The first property we can ensure is that our extension of SLG resolution terminates for the evaluation of any query, generating a final forest.

Theorem 5.1 *Let $q = L$ be a query to a doubled Hybrid MKNF knowledge base \mathcal{K}^d . Then any $\mathbf{SLG}(\mathcal{O})$ evaluation of \mathbf{K}^d for q terminates after finitely many steps, producing a finite final forest.*

PROOF. The proof is straightforward since we know already that SLG, i.e., $\mathbf{SLG}(\mathcal{O})$ without ORACLE RESOLUTION and the extended NEW SUBGOAL operation, terminates finitely for programs with bounded term-depth, and transfinitely otherwise (cf. Theorem 5.10 of [Chen and Warren 1996]). Since Definition 2.2 ensures that Hybrid MKNF knowledge bases do not contain recursive terms, i.e., non-nullary functors, they have bounded term depth, and so does the doubled knowledge base \mathcal{K}^d . Accordingly, we only have to ensure that the new operation ORACLE RESOLUTION and the extension of NEW SUBGOAL do not invalidate finite termination.

The operation ORACLE RESOLUTION can be applied in the same situation as PROGRAM CLAUSE RESOLUTION, namely when creating a new child for a root of a tree, so that each operation can be applied only once to a given node (for each of the finitely many rules, respectively for each of the finitely many possible answers of the complete oracle), and this creates one child per successful application. Now, since the knowledge base \mathcal{K}^d is finite, the number of (ground) rule heads is finite. Thus, 1) the number of children possibly created with ORACLE RESOLUTION for any arbitrary root is finite; and 2) the size of the nodes created is also finite.

The extension of the operation NEW SUBGOAL creates even in the worst case finitely many more trees with roots to which only ORACLE RESOLUTION is applicable, which in its turn is finitely many, as just demonstrated.

We conclude that termination holds for $\mathbf{SLG}(\mathcal{O})$. \square

As $\mathbf{SLG}(\mathcal{O})$ is defined, there is no prescribed order in which to apply the operations possible in a forest \mathcal{F}_i . For SLG some orders of application are in general more efficient than others but, as shown in [Chen and Warren 1996], any order yields the same outcome for any query. This same sort of confluence also holds for $\mathbf{SLG}(\mathcal{O})$:

Theorem 5.2 *Let \mathcal{E}_1 and \mathcal{E}_2 be two $\mathbf{SLG}(\mathcal{O})$ evaluations of a query $q = L$ to a doubled Hybrid MKNF knowledge base \mathcal{K}^d , \mathcal{F}_1 the final forest of \mathcal{E}_1 , and \mathcal{F}_2 the final forest of \mathcal{E}_2 . Then, $I_{\mathcal{F}_1} = I_{\mathcal{F}_2}$.*

PROOF. This is a well-known property for SLG as defined using the operations of Definition 4.5 excluding the extension of NEW SUBGOAL to classical negation, and the operations of Definitions 4.7 and 4.14 (cf. Theorem 5.7 of [Chen and Warren 1996]). Accordingly, we consider cases in which \mathcal{E}_1 and \mathcal{E}_2 make use of the operations that have been introduced/extended in $\mathbf{SLG}(\mathcal{O})$. However, PROGRAM CLAUSE RESOLUTION is used in SLG, and if we just consider the created children,

then PROGRAM CLAUSE RESOLUTION and ORACLE RESOLUTION are not distinguishable. Thus, we can consider that ORACLE RESOLUTION is a syntactic variant of PROGRAM CLAUSE RESOLUTION. The same holds for NEW SUBGOAL and the treatment of default negated atoms **not** S that create a tree with root S and those special literals **not** $NH(t_i)$ that may allow us to create a tree with root $\neg H(t_i)$: both its children are not distinguishable and only one of the two is applicable in each case. Thus, confluence of **SLG**(\mathcal{O}) follows directly from confluence of SLG (see Theorem 5.7 of [Chen and Warren 1996]). \square

The above theorem is also helpful to prove that **SLG**(\mathcal{O}) is a correct query procedure for $MKNF_{WF}$ and terminates within the same complexity bounds as the semantics defined in [Knorr et al. 2011]. First, we show that **SLG**(\mathcal{O}) coincides with $MKNF_{WF}$. Intuitively, what we have to show is that the well-founded MKNF model, as presented in Section 2 and based on the computation presented in Section 3, coincides with the interpretation $I_{\mathcal{F}}$ induced by the final forest \mathcal{F}_n for some query q to \mathcal{K}^d in all the ground literals involved in the query.¹⁸ We can further simplify that by showing, for each literal L appearing in \mathcal{K}_G^d , that $L \in M_{WF}$ (Definition 2.16) if and only if $L \in I_{\mathcal{F}}$ with ground query $q = L$ and \mathcal{F}_n for some n . Note that this correspondence also holds for atoms and classically negated atoms only appearing in the ontology.

Theorem 5.3 *Let \mathcal{K}^d be a doubled Hybrid MKNF knowledge base, and $I_{\mathcal{F}}$ the interpretation induced by the final forest \mathcal{F} of an **SLG**(\mathcal{O}) evaluation of \mathcal{K}^d for a ground query $q = L$ where L is a literal or a classically negated atom. **SLG**(\mathcal{O}) resolution is sound and complete w.r.t. M_{WF} , which is obtained from \mathbf{P}_{ω}^d and \mathbf{N}_{ω}^d , i.e.,*

- for $L \in \text{KA}(\mathcal{K}_G^d)$:
 - $L \in \mathbf{P}_{\omega}^d$ if and only if $L \in I_{\mathcal{F}}$ and
 - $L_1^d \notin \mathbf{N}_{\omega}^d$ if and only if $L = \text{not } L_1^d \in I_{\mathcal{F}}$.
- for $L \notin \text{KA}(\mathcal{K}_G^d)$: $\mathcal{O} \cup \mathbf{P}_{\omega}^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_{\omega}^d \models L$ if and only if $L \in I_{\mathcal{F}}$.

PROOF. (**Completeness**): We show by induction on n that

- for $L \in \text{KA}(\mathcal{K}_G^d)$, if L is a positive literal, then $L \in \mathbf{P}_n^d$ implies that $L \in I_{\mathcal{F}}$; and if $L = \text{not } L_1^d$ is a negative literal, then $L_1^d \notin \mathbf{N}_n^d$ implies that $\text{not } L_1^d \in I_{\mathcal{F}}$
- for $L \notin \text{KA}(\mathcal{K}_G^d)$, if $\mathcal{O} \cup \mathbf{P}_n^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_n^d \models L$ then $L \in I_{\mathcal{F}}$.

The induction base holds immediately, for $L \in \text{KA}(\mathcal{K}_G^d)$, since \mathbf{P}_0^d is empty and \mathbf{N}_0^d contains all literals appearing in $\text{KA}(\mathcal{K}_G^d)$. For $L \notin \text{KA}(\mathcal{K}_G^d)$, we obtain that $\mathcal{O} \models L$ or $\mathcal{O}^d \models L$, so we can create a tree $L : - \mid L$ and with ORACLE RESOLUTION an answer $L : - \mid$, which shows $L \in I_{\mathcal{F}}$.

(*Induction Hypothesis 1*) Now suppose the claim holds for n . We have to show the induction step for $n + 1$. For $L \in \text{KA}(\mathcal{K}_G^d)$, let L be a positive literal, and suppose that $L \in \mathbf{P}_{n+1}^d$ but $L \notin \mathbf{P}_n^d$ (otherwise the claim would immediately follow by the induction hypothesis). Therefore, $L \in \Gamma_{\mathcal{K}_G^d}(\mathbf{N}_n^d)$ and so $L \in T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow \omega$

¹⁸Without loss of generality, we can restrict that statement to ground queries, a non-ground query would simply require to check all the ground instances.

(Definition 3.5). We show by induction on m that $L \in T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m$ implies that $L \in I_{\mathcal{F}}$.

Inner Induction

The base case is void since $T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow 0$ is empty.

(*Induction Hypothesis 2*) Suppose the property holds for m , we show it for $m+1$. So, assume that $L \in T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow (m+1)$ but $L \notin T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m$ (otherwise the property would immediately follow by the induction hypothesis (2)). Then $L \in T_{\mathcal{K}_G^d // \mathbf{N}_n^d}(T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m)$ and either $L \in R_{\mathcal{K}_G^d // \mathbf{N}_n^d}(T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m)$ (i.e., L is a consequence of rule deduction, Definition 3.3) or $L \in D_{\mathcal{K}_G^d // \mathbf{N}_n^d}(T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m)$ (L is a consequence of deduction in the ontology). In the first case, for L to be the consequence of a rule derivation, there must be a rule $L \leftarrow A_1, \dots, A_n, \mathbf{not} B_1, \dots, \mathbf{not} B_m$ in \mathcal{K}_G^d such that all $B_j \notin \mathbf{N}_n^d$ and all $\mathbf{K} A_i \in T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m$. Such a rule gives rise to the rule $L \leftarrow A_1, \dots, A_n$ in the the MKNF-coherent reduction, $\mathcal{K}_G^d // \mathbf{N}_n^d$. We thus know by the two induction hypotheses that all A_i and all $\mathbf{not} B_j$ appear in $I_{\mathcal{F}}$. From that we can construct a tree with root $L : - \mid L$ and a child obtained by applying PROGRAM CLAUSE RESOLUTION with the rule considered. In the resulting child the set of goals contains exactly all A_i that can be removed by POSITIVE RETURN and all $\mathbf{not} B_j$ that can be removed by NEGATIVE RETURN. The result is a leaf node $L : - \mid$ and we obtain that $L \in I_{\mathcal{F}}$ for this order of applying **SLG**(\mathcal{O}) operations. Since Theorem 5.2 ensures that we achieve the same result if we alter the order of such applications of **SLG**(\mathcal{O}) operations, we know that the statement holds in general. In the second case, i.e., for $L \in D_{\mathcal{K}_G^d // \mathbf{N}_n^d}(T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m)$, we construct a tree $L : - \mid L$ and apply ORACLE RESOLUTION as (finitely) many times as necessary. One of those children is the one actually allowing to derive L by means of the ontology, i.e., all goals in this child are positive literals that are true in $T_{\mathcal{K}_G^d // \mathbf{N}_n^d} \uparrow m$. We apply POSITIVE RETURN to these literals, and this, by the induction hypothesis (2), results in a leaf node $L : - \mid$. As before, Theorem 5.2 ensures that a different application order again yields eventually the same result.

Now, let L be a negative literal $\mathbf{not} L_1^d$, and suppose that $L_1^d \notin \mathbf{N}_{n+1}^d$ but $L_1^d \in \mathbf{N}_n^d$ (otherwise the claim would follow immediately by the induction hypothesis (1)). Then, $L_1^d \in U_{\mathcal{K}_G^d}(\mathbf{P}_n^d, \mathbf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$ by Lemma 3.10, i.e., L_1^d occurs in the greatest unfounded set w.r.t. $(\mathbf{P}_n^d, \mathbf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$. We construct a tree with root $L_1^d : - \mid L_1^d$. We proceed by creating all children of that root, applying PROGRAM CLAUSE RESOLUTION and ORACLE RESOLUTION as (finitely) many times as possible. By Definition 3.9, each such child (after finitely many subsequent operations) is either false or completely evaluated as in case 2.(b) of Definition 4.12, which means that another element of $U_{\mathcal{K}_G^d}(\mathbf{P}_n^d, \mathbf{KA}(\mathcal{K}_G^d) \setminus \mathbf{N}_n^d)$ has been encountered in the list of goals. Note that **SLG**(\mathcal{O}) selects literals in some order, while the greatest unfounded set U just refers to some other element in U . Consequently, it may happen that we have to evaluate some literals first whose evaluation is only known in an iteration step m with $m > n$. But this does not cause any problem. Such negative literals may be simply delayed (DELAYING), while both positive and negative literals are processed (NEW SUBGOAL and so on): if a literal can eventually be resolved, then it is removed from the list of goals of a child. Otherwise, we obtain an even larger unfounded set. In both cases, once no further operation can be applied, the set U

can be completed, and, by Definition 4.6, we derive $\mathbf{not} L_1^d \in I_{\mathcal{F}}$.

End of Inner Induction

The previous inner induction handled the case where L was derived as a consequence of a rule. Alternately, suppose that $L \notin \mathbf{KA}(\mathcal{K}_G^d)$ and $\mathcal{O} \cup \mathbf{P}_n^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_n^d \models L$. We can construct a tree starting with $L : - \mid L$ and apply ORACLE RESOLUTION until we get a child $L : - \mid Goals$ such that $Goals \subseteq \mathbf{P}_n^d$, which has to exist. We apply POSITIVE RETURN to all positive literals in $Goals$, which is possible by the induction hypothesis (1) thus deriving the answer $L : - \mid$, from which we conclude $L \in I_{\mathcal{F}}$.

(Soundness): We show by induction on n that:

- for $L \in \mathbf{KA}(\mathcal{K}_G^d)$, if L is a positive literal, then $L \in I_{\mathcal{F}_n}$ implies that $L \in \mathbf{P}_\omega^d$, and if $L = \mathbf{not} L_1^d$ is a negative literal, then $L \in I_{\mathcal{F}_n}$ implies that $L_1^d \notin \mathbf{N}_\omega^d$; and
- for $L \notin \mathbf{KA}(\mathcal{K}_G^d)$, if $L \in I_{\mathcal{F}}$, then $\mathcal{O} \cup \mathbf{P}_\omega^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_\omega^d \models L$.

The induction base holds trivially, since $I_{\mathcal{F}_0}$ is empty. So assume the property holds for n . We show that the property holds for all cases (1)–(9) of an **SLG**(\mathcal{O}) operation that may be applied to $I_{\mathcal{F}_n}$ yielding $I_{\mathcal{F}_{n+1}}$.

- (1) **NEW SUBGOAL:** This operation creates a new tree and does alone not alter $I_{\mathcal{F}}$, i.e., if $L \in I_{\mathcal{F}_{n+1}}$, then $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (2) **PROGRAM CLAUSE RESOLUTION:** A new child is created for the root $S \leftarrow \mid S$. If this child has an empty list of goals, then a rule with empty body was used to create this child. Now, if L is a positive literal with $L = S$, then $L \in I_{\mathcal{F}_{n+1}}$. But then, $L \in \mathbf{P}_\omega^d$ since there is a fact L in \mathcal{K}_G^d . Alternatively, if the list of children is not empty, then $L \in I_{\mathcal{F}_{n+1}}$ implies $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (3) **POSITIVE RETURN:** If the resolved goal is the last remaining, then the outcome of the operation is an unconditional answer. Suppose the answer template is equal to L . We can trace back this child to the immediate child of the root. All goals in this particular child have already been resolved, so that, by the induction hypothesis, all positive literals L appear in \mathbf{P}_ω^d and all negative literals $\mathbf{not} L_1^d$ do not appear in \mathbf{N}_ω^d . But then the property holds, no matter whether $L \in \mathbf{KA}(\mathcal{K}_G^d)$ or not. Alternatively, if the list of goals (including delayed ones) is not empty, then $L \in I_{\mathcal{F}_{n+1}}$ implies $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (4) **NEGATIVE RETURN**
 - (a) **NEGATION SUCCESS:** The argument is exactly the same as for POSITIVE RETURN, only now the last goal is a negative literal.
 - (b) **NEGATION FAILURE:** This operation fails one child. However, it does alone not contribute to $I_{\mathcal{F}}$, i.e., if $L \in I_{\mathcal{F}_{n+1}}$, then $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (5) **DELAYING:** This operation at best provides a conditional answer. As such it does not affect $I_{\mathcal{F}}$ alone. Therefore, if $L \in I_{\mathcal{F}_{n+1}}$, then $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (6) **SIMPLIFICATION:**

- (a) The first simplification case corresponds exactly to **NEGATION FAILURE**, only here the failure occurs in *Delays* and the failed literal may be positive or negative.
 - (b) The second simplification case corresponds exactly to **NEGATION SUCCESS**, only now the success occurs in *Delays* and the successful literal may be positive or negative.
- (7) **ORACLE RESOLUTION**: A new child is created for the root $S \leftarrow \mid S$ by means of the oracle. If the returned list of goals is empty, then the oracle allows us to derive the root directly, and $\mathcal{O} \cup I_{\mathcal{F}_n} \models L$ or $\mathcal{O}^d \cup I_{\mathcal{F}_n} \models L$. In this case, if L is a positive literal with $L = S$, then $L \in I_{\mathcal{F}_{n+1}}$. If $L \in \text{KA}(\mathcal{K}_G^d)$, then $L \in \mathbf{P}_\omega^d$, since the operator $D_{\mathcal{K}_G^d}$ together with all $L' \in I_{\mathcal{F}_n}$, for which $L' \in \mathbf{P}_\omega^d$ holds by the induction hypothesis, allows us to derive L . If $L \notin \text{KA}(\mathcal{K}_G^d)$, then $\mathcal{O} \cup \mathbf{P}_\omega^d \models L$ or $\mathcal{O}^d \cup \mathbf{P}_\omega^d \models L$ holds since we have that, for all $L' \in I_{\mathcal{F}_n}$, $L' \in \mathbf{P}_\omega^d$ holds by the induction hypothesis. Alternatively, if the list of goals is not empty, then $L \in I_{\mathcal{F}_{n+1}}$ implies $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (8) **COMPLETION**: This operation only affects $I_{\mathcal{F}}$ if some A is in the ground instantiation of a completely evaluated literal in \mathcal{F} and A is not in the ground instantiation of any answer in a tree in \mathcal{F} . In other words, this operation introduces **not** L' to $I_{\mathcal{F}}$. In particular, consider $L = \text{not } L_1^d$ as a negative literal and $L \in I_{\mathcal{F}_{n+1}}$. Thus, the tree for L_1^d does not contain any answer but also no further operation can be applied, i.e., in each child, there is (at least) one literal that either can not be resolved or it is failed. This matches the condition of the greatest unfounded set and we obtain that $L_1^d \notin \mathbf{N}_\omega^d$. For all other cases, if $L \in I_{\mathcal{F}_{n+1}}$, then $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.
- (9) **ANSWER COMPLETION**: This operation may affect $I_{\mathcal{F}}$ by adding failure nodes as children to conditional answers. Assume that one such answer occurs within some tree with root $S \leftarrow S$ in \mathcal{F}_n . In such a case, S may become false in $I_{\mathcal{F}_{n+1}}$ but was not false in $I_{\mathcal{F}_n}$. However, the notion of an answer that is not supported (Definition 4.13) captures the definition of an element of an unfounded set: in this case literals in the unfounded set may be in the *Delays* of an answer. As with the case of **COMPLETION**, we have that for any L in the ground instantiation of S , $L^d \notin \mathbf{N}_\omega^d$. For all other cases, if $L \in I_{\mathcal{F}_{n+1}}$, then $L \in I_{\mathcal{F}_n}$, and the property holds by the induction hypothesis.

We conclude that soundness holds. \square

Given the soundness of $MKNF_{WF}$ with respect to the semantics of MKNF knowledge bases of [Motik and Rosati 2010], it follows easily from [Knorr et al. 2011] that:

Corollary 5.4 *Let \mathcal{K} be a Hybrid MKNF knowledge base and L a literal that appears in \mathcal{K}_G^d . If $L \in I_{\mathcal{F}}$ ($L = \text{not } L_1^d \in I_{\mathcal{F}}$ respectively), where $I_{\mathcal{F}}$ is induced by the forest \mathcal{F} of an **SLG**(\mathcal{O}) evaluation of \mathcal{K}_G^d for query $q = L$, then L (**not** L_1 respectively) is derivable from all two-valued MKNF models of \mathcal{K} .*

In addition to the interpretation of the final forest $I_{\mathcal{F}}$ being sound with respect to the 2-valued MKNF model, the conditional answers in \mathcal{F} can be seen as a well-founded reduct of the rules in \mathcal{K} , augmented with conditional answers derived by

ORACLE RESOLUTION operations. As a result, the final forest can be seen as a *residual program*: a sound transformation not only of the rules, but of information from the oracle, and can be used to construct a partial 2-valued stable model.¹⁹

Regarding complexity, it is clear that the complexity of the whole procedure $\mathbf{SLG}(\mathcal{O})$ depends on the complexity of the oracle, and also on the number of results returned by each call to the oracle. Clearly, the complexity associated to the computation of one result of the oracle function is a lower-bound of the complexity of $\mathbf{SLG}(\mathcal{O})$. Moreover, even if the computation of one result of the oracle is tractable, the (data) complexity of $\mathbf{SLG}(\mathcal{O})$ may still be exponential if exponentially many solutions are generated by the oracle, e.g., if returning all supersets of a solution. This is so, because our definition of the oracle is quite general, and in order to prove interesting complexity results some assumptions must be made about the oracle. We start by defining a correct partial oracle:

Definition 5.5 Let $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$ be a doubled Hybrid MKNF knowledge base, S a goal, and \mathcal{L} a set of ground atoms such that each $L \in \mathcal{L}$ is unifiable with at least one rule head in \mathcal{P}^d (called program atoms). A *partial oracle* for \mathcal{O} , denoted $pT_{\mathcal{O}}$, is a relation $pT_{\mathcal{O}}(I_{\mathcal{F}}, S, \mathcal{L})$ such that if $pT_{\mathcal{O}}(I_{\mathcal{F}}, S, \mathcal{L})$, then

$$\begin{aligned} &\mathcal{O} \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \models S \text{ and } \mathcal{O} \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \text{ consistent; or} \\ &\mathcal{O}^d \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \models S \text{ and } \mathcal{O}^d \cup I_{\mathcal{F}}^+ \cup \mathcal{L} \text{ consistent.} \end{aligned}$$

A partial oracle $pT_{\mathcal{O}}$ is *correct* w.r.t. $compT_{\mathcal{O}}$ iff, for all MKNF-consistent \mathcal{K}^d , replacing $compT_{\mathcal{O}}$ in $\mathbf{SLG}(\mathcal{O})$ with $pT_{\mathcal{O}}$ succeeds for exactly the same set of queries.

Note that the complete oracle is indeed generating unnecessarily many answers, and it can be replaced by a partial one that assures correctness. E.g., consider a partial oracle that does not return supersets of other results. Such a partial oracle is obviously correct. A further improvement on efficiency is the restriction to consistent sets $\mathcal{O} \cup I_{\mathcal{F}_n}^+ \cup L$ and $\mathcal{O}^d \cup I_{\mathcal{F}_n}^+ \cup L$. If the knowledge base is MKNF-consistent, then looking for derivations based on inconsistencies is pointless anyway: we would just create a potentially large number of children none of which would result in an unconditional answer. In this sense, partial oracles are limited to meaningful derivations. In the case of an MKNF-inconsistent knowledge base, things get a bit more complicated.

Example 5.6 Consider again the already doubled knowledge base from Example 3.13.

$Q \sqsubseteq \neg R$ $p(a) \leftarrow \text{not } p^d(a)$ $Q(a) \leftarrow$ $R(a) \leftarrow \text{not } R^d(a)$	$Q^d \sqsubseteq \neg R^d$ $p^d(a) \leftarrow \text{not } p(a)$ $Q^d(a) \leftarrow \text{not } NQ(a)$ $R^d(a) \leftarrow \text{not } R(a), \text{not } NR(a)$
---	--

¹⁹[Chen and Warren 1996] discusses these transformational aspects of SLG resolution, which are preserved in $\mathbf{SLG}(\mathcal{O})$, while the XSB manual discusses how the residual program can serve as input to an ASP solver.

Cf. the computation in Example 3.13, $p(a)$, $Q(a)$, and $R(a)$ are true in the sequence \mathbf{P}_ω^d while $p^d(a)$, $Q^d(a)$, and $R^d(a)$ are false in the sequence \mathbf{N}_ω^d . The same results are derivable with a complete oracle. $Q(a)$ is derivable from the corresponding fact. From that $\neg R(a)$ is derivable and therefore **not** $R^d(a)$ as well. This allows us to obtain $R(a)$. Now, $Q(a)$ and $R(a)$ together with \mathcal{O} are inconsistent from which we can derive $p(a)$, but also $\neg Q(a)$ and $\neg p(a)$. Consequently, **not** $p^d(a)$ and **not** $Q^d(a)$ hold as well, i.e., everything is supposedly true and false at the same time.

If we limit ourselves to the partial (consistent) oracle, then we no longer derive $p(a)$, **not** $Q(a)$, or **not** $p(a)$. In this case, $R(a)$ is still true and false (inconsistent), but $Q(a)$ is true, and $p(a)$ is undefined.

Thus, the usage of such a partial oracle partially hides MKNF-inconsistencies and demonstrates a somewhat paraconsistent behavior instead.

This example also shows why correctness of a partial oracle is only defined w.r.t. MKNF-consistent knowledge bases. For MKNF-inconsistent knowledge bases the derivation relation is not identical in general.

By making assumptions on the complexity and number of results of an oracle, complexity results of $\mathbf{SLG}(\mathcal{O})$ are obtained.

Theorem 5.7 *Let \mathcal{K}^d be a doubled Hybrid MKNF knowledge base, and $pT_{\mathcal{O}}$ a correct partial oracle for \mathcal{O} , such that for every goal S , the cardinality of $pT_{\mathcal{O}}(I_{\mathcal{F}}, S, L)$ is bound by a polynomial on the number of program atoms. Moreover, assume that computing each element of $pT_{\mathcal{O}}$ is decidable with data complexity \mathcal{C} . Then, the $\mathbf{SLG}(\mathcal{O})$ evaluation of a query in \mathcal{K}^d is decidable with data complexity $P^{\mathcal{C}}$.*

PROOF. Decidability is guaranteed by Theorem 5.1. As for complexity, first note that, given the polynomial data complexity of SLG [Chen and Warren 1996], $\mathbf{SLG}(\mathcal{O})$ without calls to the oracle is of polynomial data complexity as well. Considering the oracle, since the cardinality of $pT_{\mathcal{O}}(I_{\mathcal{F}_n}, S, L)$ is bound by a polynomial, and each of the calls to the oracle can be seen as adding a new program rule (the result of ORACLE RESOLUTION operation), only polynomially many such rules are added. Hence, as such, the inclusion of oracle calls does not alter the polynomial data complexity of SLG. Now, computing each such rule amounts to a call to the oracle, which by hypothesis is decidable and with data complexity \mathcal{C} . So, the overall data complexity is $P^{\mathcal{C}}$. Note that the doubling of the knowledge base does not affect that since the factor 2 is subsumed by the (at least) polynomial complexity. \square

In particular, Theorem 5.7 means that if the partial oracle is tractable, and produces only polynomial many results, then $\mathbf{SLG}(\mathcal{O})$ is also tractable. Clearly, for an ontology part of the knowledge base that is tractable, it is possible to come up with a correct partial oracle that is also tractable. Basically, all that needs to be done is to proceed with the usual entailment method, assuming that all program atoms hold, and collecting them for the oracle result. To guarantee that the number of solutions of the oracle is bound by a polynomial, and still keeping with correctness, might be a bit more difficult. It amounts to finding a procedure that returns less results, and at the same time does not damage the completeness proof (similar to that of Theorem 5.3). At least for the tractable case this is

possible, albeit the oracle being the (polynomial complexity) bottom-up procedure that defines $MKNF_{WF}$. This approach is, however, somewhat counterproductive to the whole idea of a top-down querying mechanism: we could simply use the bottom-up procedure in the first place to compute the model and store the results in a database which we then query on demand. The following section defines a concrete oracle for the tractable description logic \mathcal{EL}^+ that maintains the desired data complexity and retains goal-orientation.

6. AN ORACLE FOR \mathcal{EL}^+

When defining an oracle on \mathcal{EL}^+ we could simply try to use the algorithm for subsumption presented in [Baader et al. 2005]: reduce instance checking to subsumption and return the desired set of atoms which, when proven, would ensure the derivability of the initial query. However, apart from the technical problems we would have to face, like how to obtain these sets of atoms whose truth allows us to prove the initial query, this would mean that we would have to run the entire subsumption algorithm for each query posed to the oracle in \mathcal{EL}^+ .

We therefore proceed differently. We still use the algorithm for subsumption from [Baader et al. 2005] to compute the complete class hierarchy of the \mathcal{EL}^+ TBox \mathcal{T} , but we use it only once, as a kind of preprocessing of the ontology. Then we take the obtained results together with the TBox \mathcal{T} and simplify them by removing all statements that are redundant when looking for instances of classes in a top-down manner. The result, together with the \mathcal{EL}^+ ABox \mathcal{A} , is then turned into a set of rules which can be used in a top-down manner, by using SLG alone, yielding the desired oracle. Moreover, this way we can straightforwardly combine these transformed rules with the ones in the knowledge base and, with the top-down querying system defined in Section 4, obtain a single top-down procedure querying an MKNF knowledge base where the ontology is described in \mathcal{EL}^+ .

6.1 Subsumption in \mathcal{EL}^+

In [Baader et al. 2005], a polynomial time algorithm for subsumption was described, and we recall important notions from it, restricted to \mathcal{EL}^+ . For a TBox \mathcal{T} , the notion $BC_{\mathcal{T}}$ represents the smallest set of concept descriptions that contains all concept names used in \mathcal{T} plus the top concept \top ; while $R_{\mathcal{T}}$ denotes the set of all role names used in \mathcal{T} . Using this notation, a normalized form of a TBox \mathcal{T} is defined.

Definition 6.1 [Baader et al. 2005] A TBox \mathcal{T} is in *normal form* if

- (1) all GCIs have one of the following forms, where $C_1, C_2 \in BC_{\mathcal{T}}$ and $D \in BC_{\mathcal{T}} \cup \{\perp\}$:

(1) $C_1 \sqsubseteq D$	(3) $\exists R.C_1 \sqsubseteq D$
(2) $C_1 \sqcap C_2 \sqsubseteq D$	(4) $C_1 \sqsubseteq \exists R.C_2$
- (2) all RI are of the form $R \sqsubseteq S$ or $R_1 \circ R_2 \sqsubseteq S$

By appropriately introducing new concept and role names, any TBox \mathcal{T} can be turned into normal form and, as shown in [Baader et al. 2005], this transformation

can be done in linear time. So, from now on, we assume that any TBox \mathcal{T} is in normal form.

The subsumption algorithm for \mathcal{EL}^+ ([Baader et al. 2005]) applies a set of completion rules to compute the entire class hierarchy, i.e. all subsumption relationships between all pairs of concept names occurring in \mathcal{T} . In detail, given a normalized TBox \mathcal{T} , the algorithm computes:

- a mapping S from $\text{BC}_{\mathcal{T}}$ to a subset of $\text{BC}_{\mathcal{T}} \cup \{\perp\}$; and
- a mapping T from $\text{R}_{\mathcal{T}}$ to a binary relation on $\text{BC}_{\mathcal{T}}$.

These mappings make implicit relations explicit in the following sense:

- (I1). $D \in S(C)$ implies that $C \sqsubseteq D$,
- (I2). $(C, D) \in T(R)$ implies that $C \sqsubseteq \exists R.D$.

The initialization of these mappings is the following:

- $S(C) := \{C, \top\}$ for each $C \in \text{BC}_{\mathcal{T}}$
- $T(R) := \emptyset$ for each $R \in \text{R}_{\mathcal{T}}$

Then the following completion rules are applied to extend S and T until no more rule applies.

- CR1 If $C' \in S(C)$, $C' \sqsubseteq D \in \mathcal{T}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$
- CR2 If $C_1, C_2 \in S(C)$, $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$, and $D \notin S(C)$
then $S(C) := S(C) \cup \{D\}$
- CR3 If $C' \in S(C)$, $C' \sqsubseteq \exists R.D \in \mathcal{T}$, and $(C, D) \notin T(R)$
then $T(R) := T(R) \cup \{(C, D)\}$
- CR4 If $(C, D) \in T(R)$, $D' \in S(D)$, $\exists R.D' \sqsubseteq E \in \mathcal{T}$, and $E \notin S(C)$
then $S(C) := S(C) \cup \{E\}$
- CR5 If $(C, D) \in T(R)$, $\perp \in S(D)$, and $\perp \notin S(C)$
then $S(C) := S(C) \cup \{\perp\}$
- CR6 If $(C, D) \in T(R)$, $R \sqsubseteq S \in \mathcal{T}$, and $(C, D) \notin T(S)$
then $T(S) := T(S) \cup \{(C, D)\}$
- CR7 If $(C, D) \in T(R_1)$, $(D, E) \in T(R_2)$, $R_1 \circ R_2 \sqsubseteq R_3 \in \mathcal{T}$, and $(C, E) \notin T(R_3)$
then $T(R_3) := T(R_3) \cup \{(C, E)\}$

Note that we omitted the four completion rules related to nominals and concrete domains.

It is shown in [Baader et al. 2005] that this algorithm terminates in polynomial time and that it is correct.

6.2 Simplifying the Ontology

Given a normalized \mathcal{EL}^+ TBox \mathcal{T} (Definition 6.1) and an \mathcal{EL}^+ ABox \mathcal{A} , the first step in transforming the ontology is to apply the subsumption algorithm to \mathcal{T} and obtain the mappings S and T computed by it. In particular, we obtain via S all the subsumption relationships implicitly or explicitly present in \mathcal{C} . In fact, it is easy to see that the initialization of $C \in S(C)$ for each $C \in \text{BC}_{\mathcal{T}}$ ensures that

each GCI of the form (1) of the normal form of Definition 6.1 ($C_1 \sqsubseteq D$) is also obtained by $D \in S(C_1)$, and each GCI of the form (4) ($C_1 \sqsubseteq \exists R.C_2$) is obtained by $(C_1, C_2) \in T(R)$.²⁰

It follows immediately from that, that we can actually ignore all GCIs of the form (1) and (4) as long as we have the complete mappings S and T of the subsumption algorithm available. But we can simplify even more.

Example 6.2 Consider the Hybrid MKNF knowledge base with \mathcal{O} in \mathcal{EL}^+ , containing one rule, and some facts.

$$\begin{array}{ll} C \sqsubseteq \exists R.D & G(x) \leftarrow D(x) \\ \exists R.C \sqsubseteq D & C(a). \quad C(b). \\ C_1 \sqcap C_2 \sqsubseteq D & R(a, b). \end{array}$$

Now consider that we want to know whether $G(a)$ holds. There is only one rule that allows us to derive $G(a)$, and this requires that $D(a)$ is derivable. Obviously, if we have $C_1(a)$ and $C_2(a)$ then $D(a)$ holds as well. But this information is currently not present in the knowledge base. If we check the second GCI then obtaining $D(a)$ requires finding $R(a, x)$ and $C(x)$ which appear as facts in the rule part, for $x = b$. Intuitively, we want the oracle to transform the query $D(a)$ into an **SLG**(\mathcal{O}) node $D(a) : - \mid R(a, x), C(x)$, the goals of which can then be resolved, leading to a derivation of $D(a)$.

Next, suppose we alternatively query for $G(b)$, and subsequently query the oracle for $D(b)$. Then the second GCI does not allow us to derive $D(b)$ because there is no $R(b, x)$ for some x derivable; the third does not allow us to derive $D(b)$ because there are no individuals known to hold in C_1 or C_2 . But even using the first GCI does not allow us to derive $D(b)$: while $C(a)$ holds and we know that there is an explicit relation $R(a, b)$ in the knowledge base, the semantics of \mathcal{O} (and descriptive first-order semantics in general) does not allow to derive $D(b)$, since $D(b)$ does not hold in all models of \mathcal{O} - there are models where $R(a, i)$ and $D(i)$ hold for some individual i not appearing in the knowledge base.

Clearly in a \mathcal{EL}^+ KB with a normalized TBox \mathcal{T} , GCIs of the form (3) ($\exists R.C_1 \sqsubseteq D$) and (2) ($C_1 \sqcap C_2 \sqsubseteq D$) – and therefore also of the form (1) – can be used to derive information when answering an (instance) query. On the other hand, the example implies that GCIs of the form (4) ($C_1 \sqsubseteq \exists R.C_2$) do not contribute to drawing this kind of conclusions. We now formalize this observation.

For simplicity of notation, we start by transforming all the mappings obtained from the algorithm into GCIs, and then we remove all GCIs of the form (4).

Definition 6.3 Let \mathcal{T} be an \mathcal{EL}^+ TBox and S and T be the mappings obtained from the subsumption algorithm. We obtain the *completed* \mathcal{EL}^+ TBox \mathcal{T}' from \mathcal{T} by adding for each $D \in S(C)$ a GCI $C \sqsubseteq D$ to \mathcal{T}' and for each $(C, D) \in T(R)$ a GCI $C \sqsubseteq \exists R.D$ to \mathcal{T}' .

Let \mathcal{T} be a completed \mathcal{EL}^+ TBox. We define the *reduced* \mathcal{EL}^+ TBox \mathcal{T}' which is obtained from the completed TBox \mathcal{T} by removing all GCIs of form (4).

²⁰Cf. the completion rules *CR1* and *CR3* in Section 2 which precisely add each such explicit GCIs to the appropriate mapping.

It is straightforward to see that the transformation from the TBox \mathcal{T} to the completed TBox \mathcal{T}' simply allows us to disregard the mappings S and T obtained by the algorithm of subsumption without losing any of the subset relationships contained in these mappings.

Now we have to show that a reduced TBox, which in general does not preserve \mathcal{EL}^+ semantics, is still suitable for the query answering we are interested in, which restricts itself to queries of the form $C(a)$ or $R(a, b)$.

Proposition 6.4 *Let \mathcal{A} be an \mathcal{EL}^+ ABox, \mathcal{T} be a completed \mathcal{EL}^+ TBox, and \mathcal{T}' the reduced \mathcal{EL}^+ TBox obtained from \mathcal{T} . Then the following two conditions hold.*

- (i) *a is an instance of concept C in \mathcal{A} w.r.t. \mathcal{T} iff a is an instance of concept C in \mathcal{A} w.r.t. \mathcal{T}' .*
- (ii) *(a, b) is an instance of role R in \mathcal{A} w.r.t. \mathcal{T} iff (a, b) is an instance of role R in \mathcal{A} w.r.t. \mathcal{T}' .*

PROOF. For (i) we have to show that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} iff $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T}' ; for (ii) we have to show that $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} iff $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T}' . We are going to sketch the argument for (i); the case of (ii) follows analogously.

' \Leftarrow ': follows directly from monotonicity: adding GCIs of the form (4) will not invalidate any drawn conclusions, i.e. if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T}' then adding GCIs of the form (4) can only reduce the common models of \mathcal{I} of \mathcal{A} and never increase. We conclude $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} .

' \Rightarrow ': suppose that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} . If none of the GCIs of the form (4) contains the concept name C then we can remove them all and $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T}' . The same argument applies if C appears only on the left hand side of such GCIs. So assume C appears on the right hand side of at least one such GCI $C_1 \sqsubseteq \exists R.C$. However, even if there is an individual i such that $i^{\mathcal{I}} \in C_1^{\mathcal{I}}$ and $(i^{\mathcal{I}}, a^{\mathcal{I}}) \in R^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} then \mathcal{T} does not allow to conclude $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T} . We can thus conclude that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ for every common model \mathcal{I} of \mathcal{A} and \mathcal{T}' . \square

Having proven that TBox completion does not alter the derivability of instance queries, we can take a short cut: instead of completing the TBox we can directly remove all GCIs of the form (4) and discard the mapping T . We then complete the TBox only with respect to the mapping S and obtain the reduced TBox.

Corollary 6.5 *Let \mathcal{T} be a \mathcal{EL}^+ TBox and S and T be the mappings obtained from the subsumption algorithm. We obtain the reduced TBox \mathcal{T}' from \mathcal{T} by removing all GCIs of the form (4) from \mathcal{T} and by adding for each $D \in S(C)$ a GCI $C \sqsubseteq D$.*

6.3 Transformation into Rules

Now, we show how to transform the reduced \mathcal{EL}^+ KB into rules in such a way that running the SLG procedure on the obtained set of rules yields an oracle that can be used in **SLG**(\mathcal{O}). Special care must be taken with inconsistencies and with the

fact that if an atom is proven false in the ontology, then its negation also holds in the rules. Note that this is achieved in $\mathbf{SLG}(\mathcal{O})$ by querying for classically negated atoms, but these are outside the syntax of \mathcal{REL} even though a restricted form of negation is achievable via \perp .

Regarding inconsistencies, there are two kinds which can appear in the three-valued Hybrid MKNF semantics as presented in [Knorr et al. 2011]: either the ontology alone is inconsistent, or there is an inconsistency resulting from contradictory derivations in the rules and the ontology. In the first case, there is not much to be done. An inconsistent ontology has no models and we can simply derive anything from it, making reasoning over a combined knowledge base rather pointless. We therefore admit in the following an a-priori consistency check of the ontology alone, and proceed only if it succeeds, i.e., we limit ourselves in the following to a consistent ontology.²¹ For the second case, the bottom-up computation allows us to detect such problems, but in $\mathbf{SLG}(\mathcal{O})$ we are limited to finding atoms that are true and false at the same time, i.e., if for some $\mathbf{C}(\mathbf{a})$ both queries $\mathbf{C}(\mathbf{a})$ and $\mathbf{not} \mathbf{C}^d(\mathbf{a})$ ²² are answered with 'yes', then the combined KB is inconsistent. This can, of course, not be complete for a partial oracle, as shown in Example 5.6, so that we obtain a paraconsistent behavior. To carry over this behavior to a transformation into rules, we have to take into consideration the transformation presented in Definition 3.4 and their effect on the \mathcal{EL}^+ KB.

Regarding classical negation, we solve the problem in a specific way. In $\mathbf{SLG}(\mathcal{O})$, the special negative literals $\mathbf{not} NH(t_i)$ are used to call $\neg H(t_i)$. Since this is not expressible in \mathcal{EL}^+ we simply consider $\mathbf{not} NH(t_i)$ as normal negative literals, and transform \mathcal{O} into rules such that $\mathbf{not} NH(t_i)$ holds if $\neg H(t_i)$ holds. More precisely, if $H \sqsubseteq \perp$, then $NH(t_i)$ holds.

We are now ready to define the transformation of the ontology \mathcal{O} consisting of a reduced *TBox* and an *ABox* into a set of already doubled rules (see Definition 3.1).

Definition 6.6 Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a Hybrid MKNF knowledge base with a consistent \mathcal{EL}^+ KB \mathcal{O} . We define $\mathcal{P}_{\mathcal{O}}^d$ from \mathcal{O} , where C, D, C_1 , and C_2 are concept names, R, S, T are role names, and a, b are individual names, as the smallest set containing:

- (a1). for each $C(a) \in \mathcal{A}$: $C(a) \leftarrow$ and $C^d(a) \leftarrow \mathbf{not} NC(a)$.
- (a2). for each $R(a, b) \in \mathcal{A}$: $R(a, b) \leftarrow$ and $R^d(a, b) \leftarrow \mathbf{not} NR(a, b)$.
- (c1). for each $\text{GCI } C \sqsubseteq D \in \mathcal{T}$: $D(x) \leftarrow C(x)$ and $D^d(x) \leftarrow C^d(x), \mathbf{not} ND(x)$.
- (c2). for each $C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{T}$: $D(x) \leftarrow C_1(x), C_2(x)$ and $D^d(x) \leftarrow C_1^d(x), C_2^d(x), \mathbf{not} ND(x)$.
- (c3). for each $\exists R.C \sqsubseteq D \in \mathcal{T}$: $D(x) \leftarrow R(x, y), C(y)$ and $D^d(x) \leftarrow R^d(x, y), C^d(y), \mathbf{not} ND(x)$.
- (r1). for each $\text{RI } R \sqsubseteq S \in \mathcal{T}$: $S(x, y) \leftarrow R(x, y)$ and $S^d(x, y) \leftarrow R^d(x, y), \mathbf{not} NS(x, y)$.

²¹Note that ontologies in \mathcal{EL}^+ can in fact be inconsistent: consider a GCI $C \sqsubseteq \perp$ in the TBox and an assertion $C(a)$ in the ABox.

²²Recall that we use the doubled predicate for determining falsity.

- (r2). for each $R \circ S \sqsubseteq T \in \mathcal{T}$: $T(x, z) \leftarrow R(x, y), S(y, z)$ and $T^d(x, z) \leftarrow R^d(x, y), S^d(y, z)$, **not** $NT(x, z)$.
- (i1). for each $C \sqsubseteq \perp \in \mathcal{T}$: $NC(x) \leftarrow$.
- (i2). for each $C_1 \sqcap C_2 \sqsubseteq \perp \in \mathcal{T}$: $NC_2(x) \leftarrow C_1(x)$ and $NC_1(x) \leftarrow C_2(x)$.
- (i3). for each $\exists R.C \sqsubseteq \perp \in \mathcal{T}$: $NC(y) \leftarrow R(x, y)$ and $NR(x, y) \leftarrow C(y)$.

Note that the cases (i1) to (i3) are used to introduce truth of some $NH(t_i)$. Furthermore, these three cases only produce one rule, since atoms based on predicates of the forms NC^d or NR^d are not required anywhere.

Program $\mathcal{P}_{\mathcal{O}}^d$ can then be used as the basis for obtaining a correct partial oracle for \mathcal{EL}^+ , to be integrated in the general procedure of **SLG**(\mathcal{O}). Recall that an oracle receives a query S and the already derived (positive) information $I_{\mathcal{F}_n}^+$, and returns a set of atoms L , which if proven, ensure that S is derivable. The general idea of such an oracle for \mathcal{EL}^+ would be to use SLG to query Q in a program consisting of $\mathcal{P}_{\mathcal{O}}^d$ plus facts for all the atoms in $I_{\mathcal{F}_n}^+$, in such a way that any time an atom also defined in the rules is queried, the atom can succeed, i.e., is removed from the resolvent, and is collected in a set associated to the respective derivation branch.²³ Upon success, the so modified SLG procedure would return the set of collected atoms. The partial oracle would be defined by the relation with the query, the running forest, and the returned set of collected atoms. However, since both the rule part and the oracle itself would be evaluated by an SLG procedure, they can be combined: instead of collecting the atoms in the set, and then calling them in **SLG**(\mathcal{O}) after the oracle returns a result, one can simply immediately call the otherwise collected atoms, i.e., the atoms defined in the program. This way, correctness of the so defined partial oracle is equivalent to the correctness of the above transformation. We start by proving this for the consistent case:

Theorem 6.7 *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be an MKNF-consistent Hybrid MKNF knowledge base with \mathcal{O} in \mathcal{EL}^+ . Then $\mathcal{K}_{\mathcal{EL}^+} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}_{\mathcal{O}}^d))$ is semantically equivalent to $\mathcal{K}^d = (\mathcal{O}, \mathcal{O}^d, \mathcal{P}^d)$.*

PROOF. We have to show that $\mathcal{P}_{\mathcal{O}}^d$ is equivalent to \mathcal{O} and \mathcal{O}^d .

The transformations on ABox assertions, (a1) and (a2), on GCIs in \mathcal{C} , (c1), (c2), and (c3), and on role inclusions, (r1) and (r2), are semantically equivalent and can be found, e.g., in [Grosz et al. 2003]. Since \mathcal{O} contains the original GCIs and \mathcal{O}^d the doubled ones with new predicate names, we also create two rules, one for each of the two DL knowledge bases in \mathcal{K}^d . Note that the addition of predicates, such as $NC(x)$, to the body of a rule with head $C^d(x)$ is just done to enforce that whenever $NC(x)$ holds, i.e., $\neg C(x)$, then $C^d(x)$ cannot become true, which is used in the consistent case to enforce coherence. We only have to consider the transformations (i1) to (i3).

- (i1) $C \sqsubseteq \perp$: C is unsatisfiable, i.e., $\neg C(x)$ for all x ; \mathcal{O} contains a statement that allows us to infer $\neg C(x)$ which corresponds exactly to the fact $NC(x) \leftarrow$.

²³An alternative way of viewing this, would be to add to $\mathcal{P}_{\mathcal{O}}^d$ facts for all the atoms defined in the rules, run SLG as usual, but collecting all those facts that were used in the derivation.

- (i2) $C_1 \sqcap C_2 \sqsubseteq \perp$: the statement expresses disjointness of C_1 and C_2 , i.e., $\neg(C_1(x) \wedge C_2(x))$ for all x which is equivalent to $C_1(x) \rightarrow \neg C_2(x)$ and $C_2(x) \rightarrow \neg C_1(x)$; using the correspondences $\neg C_1(x) \rightarrow NC_1(x)$ and $\neg C_2(x) \rightarrow NC_2(x)$.
- (i3) $\exists R.C \sqsubseteq \perp$ follows the same argument as (i2).

This finishes the proof. \square

For the MKNF-inconsistent case, we point out that one result of the transformation into rules is that we obtain a somewhat paraconsistent approach: while an inconsistent ontology allows us to derive anything from it, the process of doubling the rules enables us to derive those consequences that do not depend on inconsistent information contained in the KB as presented in Example 5.6. We leave further details of this paraconsistency to future studies.

Finally, we have to show that the process of translating the ontology into rules and reasoning over the combined set of rules with **SLG**(\mathcal{O}) also preserves the intended polynomial data complexity.

Theorem 6.8 *Let $\mathcal{K} = (\mathcal{O}, \mathcal{P})$ be a Hybrid MKNF knowledge base with \mathcal{O} in \mathcal{EL}^+ . An **SLG**(\mathcal{O}) evaluation of a query in $\mathcal{K}_{\mathcal{EL}^+} = (\emptyset, (\mathcal{P}^d \cup \mathcal{P}_{\mathcal{O}}^d))$ is decidable with data complexity in P.*

PROOF. The \mathcal{EL}^+ oracle is in fact a transformation of the evaluation of the \mathcal{EL}^+ ontology into a set of rules so that evaluation of the Hybrid MKNF KB is made w.r.t. a combined set of rules. Note that the polynomial subsumption algorithm for \mathcal{EL}^+ and the linear transformations to obtain $\mathcal{K}_{\mathcal{EL}^+}$ together are in P.

We consider the data complexity to be the number of answers returned for a given atomic query w.r.t. the number ground facts in the rules, and the number of assertions in the ABox. Note that the transformation of the \mathcal{EL}^+ axioms introduces a number of facts in the rules at most linear in the size of the ABox (cases *a1* and *a2* of Definition 6.6). As a result the number of facts in the transformed system will be linear in the size of the rule facts plus the ABox of the original system.

Finally, note that Theorem 5.7 ensures polynomial data complexity of query evaluation in Hybrid MKNF. The transformed KB $\mathcal{K}_{\mathcal{EL}^+}$ can be considered a Hybrid MKNF KB with empty \mathcal{O} , so that by Theorem 5.7 the transformed KB has a data complexity in P. Since the transformed KB increases the size of the rule facts linearly this proves the statement. \square

7. DISCUSSION AND CONCLUSIONS

7.1 Related Work

Three other semantics define well-founded models for a combination of rules and ontologies, namely the works in [Eiter et al. 2011], [Lukasiewicz 2010], and [Drabent and Małuszyński 2007]. The approach of [Eiter et al. 2011] combines ontologies and rules in a modular way, keeping separate the semantics of both, and has identical data complexity to the well-founded MKNF semantics for a tractable DL. As such, it has similarities with **SLG**(\mathcal{O}) in terms of reasoning, in the sense that both treat reasoning in the DL separately. However, the approach of [Eiter et al. 2011], implemented using the dlv hex system [Eiter et al. 2006], has a looser integration, limiting

the way the ontology can call back program atoms. In [Eiter et al. 2011], the set of atoms occurring in rules and DLs are disjoint, and links must be established using specific interface atoms in the rules, which can only temporarily add information to the DL part. To the contrary, our semantics does not require any such restriction so that the flow of information between rules and DLs is not limited. The well-founded semantics for normal dl-programs [Lukasiewicz 2010] does not require any of these limitations either, but it requires that the ontology is decomposable into a positive and a negative part. This severely restricts the applicability to arbitrary DLs although it is shown in [Lukasiewicz 2010] that the approach is applicable to the *DL-Lite* family. This contrasts with **SLG**(\mathcal{O}), which can be applied to any decidable DL. Hybrid programs of [Drabent and Małuszyński 2007] are even more restrictive than [Eiter et al. 2011] in the combination: in fact it only allows to transfer information from the ontology to the rules and not the other way around. Moreover, the semantics of this approach differs from MKNF [Motik and Rosati 2010; Knorr et al. 2011] and also [Eiter et al. 2011; Lukasiewicz 2010] in that if an ontology expresses $B_1 \vee B_2$, then the semantics in [Drabent and Małuszyński 2007] derives p from rules $p \leftarrow B_1$ and $p \leftarrow B_2$, p while MKNF and [Eiter et al. 2011; Lukasiewicz 2010] do not. More generally, several well-founded models may exist, contrary to the more common definitions of well-founded models.

In Section 6, we also presented a concrete oracle for **SLG**(\mathcal{O}) that allows the combination of non-monotonic rules with the DL \mathcal{EL}^+ . Using this oracle, **SLG**(\mathcal{O}) remains tractable w.r.t. data complexity, and permits the discovery of possible inconsistencies between the rules and the ontology. These results contribute to the work related to conjunctive query answering with respect to \mathcal{EL}^+ . Conjunctive query answering has been studied, e.g., for acyclic \mathcal{EL}^+ in [Mei et al. 2009] as an extension to [Lutz et al. 2009], where the limitation to acyclic TBoxes avoids general undecidability (see [Rosati 2007]). In contrast, our work limits the queries to be DL-safe but adds rules as an additional expressive means. As an additional point of comparison, since the concrete oracle operates in a kind of abductive way – by finding the set of atoms which together with the ontology prove the query – our work also bears some relation to [Bienvenu 2008] where general complexity results on abduction for \mathcal{EL}^{++} are established. Another concrete oracle for **SLG**(\mathcal{O}) was very recently presented in [Knorr and Alferes 2011] providing a top-down procedure for *DL-Lite_R*, the DL underlying the tractable OWL 2 profile, OWL 2 QL. As does the \mathcal{EL}^+ oracle, the *DL-Lite_R* oracle maintains the data complexity of the bottom-up approach.

7.2 Conclusions

Together with the alternate computation method of Section 3, **SLG**(\mathcal{O}) provides a sound and complete querying method for Hybrid MKNF knowledge bases. Further, **SLG**(\mathcal{O}) maintains the favorable computational complexity of the well-founded MKNF model and freely allows bidirectional calls between the ontology and the rules, unlike other approaches (as discussed in Section 7.1). As such it presents a significant step towards making Hybrid MKNF knowledge bases practically usable for the Semantic Web.

Future work with regard to concrete oracles includes the (non-trivial) extension of the \mathcal{EL}^+ oracle to \mathcal{EL}^{++} . A second potentially fruitful extension is the construction

of an oracle for ELP ([Krötzsch et al. 2008]), an approach based on rules that allow DL expressions instead of (negated) atoms and that covers \mathcal{EL}^{++} . Since the algorithmization of ELP, like that of the \mathcal{EL}^+ oracle, transforms its expressive rules into datalog rules it may benefit from the pre-processing step introduced for \mathcal{EL}^+ knowledge bases. A third concrete oracle to investigate is \mathcal{SROELV}_n [Krötzsch et al. 2011], a tractable fragment of \mathcal{SROIQ} enhanced with nominal schemas that covers not only datalog rules in DL syntax but also \mathcal{EL}^{++} .

Other future work will address the class of conjunctive queries that $\mathbf{SLG}(\mathcal{O})$ can answer. While $\mathbf{SLG}(\mathcal{O})$ queries posed to KBs without an ontology are handled in the same way as in SLG, the queries posed to the ontology, which are required to be ground, are not conjunctive queries in the sense of [Glimm et al. 2008], where boolean queries may contain anonymous variables that are interpreted existentially. The extension to such queries may possibly be supported by anonymous variables in XSB, the system in which $\mathbf{SLG}(\mathcal{O})$ is currently implemented.

Furthermore, we may take evolution and dynamics into consideration. In [Slota and Leite 2010b; Slota et al. 2011], updating Hybrid MKNF knowledge bases is considered, while [Slota and Leite 2010a; 2011] presents the problem from a more general perspective in SE-models. The extension of $\mathbf{SLG}(\mathcal{O})$ to such dynamic knowledge bases forms another line of future work.

Finally, we mention that a prototype implementation of $\mathbf{SLG}(\mathcal{O})$ exists [Gomes et al. 2010] based on XSB Prolog and its ontology management library CDF. Because CDF includes an \mathcal{ALCQ} prover written directly using XSB, the ORACLE RESOLUTION operation of Section 4 is more easily implemented than it would be using a separate prover, as is the detection of when a mutually dependent set of sub-goals is completely evaluated (Definition 4.12). Accordingly, the polynomial data complexity of the oracle is also more easily guaranteed. The resulting implementation will enable further study into how Hybrid MKNF knowledge bases can be practically used and will indicate needed optimizations and useful extensions. For instance, since XSB supports constraint processing, temporal or spatial constraints can be added to the ABox. From a systems perspective, the multi-threading of XSB can allow for the construction of Hybrid MKNF knowledge servers that make use of either Prolog rules or F-logic rules (via FLORA-2, which is implemented using XSB). As mentioned in Section 5 the final forest of a $\mathbf{SLG}(\mathcal{O})$ evaluation produces a well-founded reduct of the rules and oracle information. This reduct, which is materialized in XSB's tables, can be sent to a stable model generator through XSB's XASP library to obtain a partial stable MKNF model of [Motik and Rosati 2010].

REFERENCES

- ALFERES, J. J., KNORR, M., AND SWIFT, T. 2009. Queries to hybrid MKNF knowledge bases through oracular tabling. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds. Springer, 1–16.
- BAADER, F., BRANDT, S., AND LUTZ, C. 2005. Pushing the \mathcal{EL} envelope. In *IJCAI-05, Proceedings of the 19th International Joint Conference on Artificial Intelligence*, L. P. Kaelbling and A. Saffiotti, Eds. Morgan Kaufmann, 364–369.
- BAADER, F., CALVANESE, D., MCGUINNESS, D. L., NARDI, D., AND PATEL-SCHNEIDER, P. F., Eds.

2007. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2nd ed. Cambridge University Press.
- BIENVENU, M. 2008. Complexity of abduction in the \mathcal{EL} family of lightweight description logics. In *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR08)*. 220–230.
- BOLEY, H. AND KIFER, M., Eds. 2010. *RIF Overview*. W3C Candidate Recommendation, 22 June 2010. Available at <http://www.w3.org/TR/rif-overview/>.
- CHEN, W. AND WARREN, D. S. 1996. Tabled Evaluation with Delaying for General Logic Programs. *43*, 1, 20–74.
- DRABENT, W. AND MALUSZYŃSKI, J. 2007. Well-Founded semantics for hybrid rules. In *Proceedings of the First International Conference on Web Reasoning and Rule Systems (RR2007)*, M. M. Marchiori, J. Z. Pan, and C. de Sainte Marie, Eds. Springer, 1–15.
- EITER, T., IANNI, G., LUKASIEWICZ, T., AND SCHINDLAUER, R. 2011. Well-founded semantics for description logic programs in the Semantic Web. *ACM Transactions on Computational Logic* *12*, 11:1–11:41.
- EITER, T., IANNI, G., LUKASIEWICZ, T., SCHINDLAUER, R., AND TOMPITS, H. 2008. Combining answer set programming with description logics for the Semantic Web. *Artificial Intelligence* *172*, 12–13 (August), 1495–1539.
- EITER, T., IANNI, G., SCHINDLAUER, R., AND TOMPITS, H. 2006. Effective integration of declarative rules with external evaluations for semantic web reasoning. In *Proceedings of the 3rd European Conference on Semantic Web (ESWC 2006)*, Y. Sure and J. Domingue, Eds. Springer, 273–287.
- GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* *9*, 365–385.
- GLIMM, B., LUTZ, C., HORROCKS, I., AND SATTLER, U. 2008. Answering conjunctive queries in the *SHIQ* description logic. *Journal of Artificial Intelligence Research* *31*, 150–197.
- GOMES, A. S., ALFERES, J. J., AND SWIFT, T. 2010. Implementing query answering for hybrid MKNF knowledge bases. In *Practical Aspects of Declarative Languages, 12th International Symposium, PADL 2010, Madrid, Spain, January 18-19, 2010. Proceedings*, M. Carro and R. Peña, Eds. Springer, 25–39.
- GROSOFF, B. N., HORROCKS, I., VOLZ, R., AND DECKER, S. 2003. Description logic programs: Combining logic programs with description logics. In *Proceedings of the World Wide Web Conference (WWW2003)*, Budapest, Hungary. ACM, 48–57.
- HITZLER, P., KRÖTZSCH, M., PARSIA, B., PATEL-SCHNEIDER, P. F., AND RUDOLPH, S., Eds. 2009. *OWL 2 Web Ontology Language: Primer*. W3C Recommendation 27 October 2009. Available at <http://www.w3.org/TR/owl2-primer/>.
- KNORR, M. AND ALFERES, J. J. 2010. Querying in \mathcal{EL}^+ with nonmonotonic rules. In *Proceedings of the 19th European Conference on Artificial Intelligence, ECAI2010*, H. Coelho, R. Studer, and M. Wooldridge, Eds. IOS Press, 1079–1080.
- KNORR, M. AND ALFERES, J. J. 2011. Querying OWL 2 QL and nonmonotonic rules. In *The Semantic Web –2011–10th International Semantic Web Conference, ISWC 2011, Bonn, Germany, October 23-27*. to appear.
- KNORR, M., ALFERES, J. J., AND HITZLER, P. 2008. A coherent well-founded model for hybrid MKNF knowledge bases. In *Proceedings of the 18th European Conference on Artificial Intelligence, ECAI2008*, M. Ghallab, C. D. Spyropoulos, N. Fakotakis, and N. Avouris, Eds. IOS Press, 99–103.
- 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 (June), 1528–1554.
- KRÖTZSCH, M., MAIER, F., KRISNADHI, A. A., AND HITZLER, P. 2011. A better uncle for OWL: Nominal schemas for integrating rules and ontologies. In *Proceedings of the 20th International World Wide Web Conference, WWW2011, March/April 2011*. ACM, 645–654.
- KRÖTZSCH, M., RUDOLPH, S., AND HITZLER, P. 2008. ELP: Tractable rules for OWL 2. In *Proceedings of the 7th International Semantic Web Conference (ISWC-08)*, A. P. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin, and K. Thirunarayan, Eds. Springer, 649–664.

- LIFSCHITZ, V. 1991. Nonmonotonic databases and epistemic queries. In *Proceedings of the 12th International Joint Conferences on Artificial Intelligence, IJCAI'91*, J. Mylopoulos and R. Reiter, Eds. 381–386.
- LLOYD, J. W. 1987. *Foundations of Logic Programming*, 2nd ed. Springer.
- LUKASIEWICZ, T. 2010. A novel combination of answer set programming with description logics for the Semantic Web. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 22, 11 (November), 1577–1592.
- LUTZ, C., TOMAN, D., AND WOLTER, F. 2009. Conjunctive query answering in the description logic \mathcal{EL} using a relational database system. In *IJCAI'09: 21st International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 2070–2075.
- MEI, J., LIU, S., XIE, G. T., KALYANPUR, A., AND FOKOUE, A. 2009. A practical approach for scalable conjunctive query answering on acyclic \mathcal{EL}^+ knowledge base. In *The Semantic Web - ISWC 2009, 8th International Semantic Web Conference, ISWC 2009, Chantilly, VA, USA, October 25-29, 2009. Proceedings*, A. Bernstein, D. R. Karger, T. Heath, L. Feigenbaum, D. Maynard, E. Motta, and K. Thirunarayan, Eds. Springer, 408–423.
- MOTIK, B., GRAU, B. C., HORROCKS, I., WU, Z., FOKOUE, A., AND LUTZ, C., Eds. 2009. *Profiles*. W3C Recommendation 27 October 2009. Available at <http://www.w3.org/TR/owl2-profiles/>.
- MOTIK, B. AND ROSATI, R. 2007. A faithful integration of description logics with logic programming. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, M. M. Veloso, Ed. AAAI Press, Hyderabad, India, 477–482.
- MOTIK, B. AND ROSATI, R. 2010. Reconciling Description Logics and Rules. *Journal of the ACM* 57, 5, 93–154.
- PATEL, C., CIMINO, J. J., DOLBY, J., FOKOUE, A., KALYANPUR, A., KERSHENBAUM, A., MA, L., SCHONBERG, E., AND SRINIVAS, K. 2007. Matching patient records to clinical trials using ontologies. In *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11-15, 2007*, K. Aberer, K.-S. Choi, N. F. Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, Eds. Springer, 816–829.
- PEARCE, D. AND WAGNER, G. 1990. Reasoning with negative information I: Strong negation in logic programs. In *Language, Knowledge and Intentionality*, L. Haaparanta, M. Kusch, and I. Niiniluoto, Eds. Acta Philosophica Fennica 49, 430–453.
- PEREIRA, L. M. AND ALFERES, J. J. 1992. Well founded semantics for logic programs with explicit negation. In *10th European Conference on Artificial Intelligence, ECAI 92, Vienna, Austria, August 3-7, 1992*, B. Neumann, Ed. John Wiley and Sons, Chichester, 102–106.
- ROSATI, R. 2007. On conjunctive query answering in \mathcal{EL} . In *Description Logics 2007*, D. Calvanese, E. Franconi, V. Haarslev, D. Lembo, B. Motik, A.-Y. Turhan, and S. Tessaris, Eds. CEUR Electronic Workshop Proceedings.
- SAGONAS, K., SWIFT, T., AND WARREN, D. S. 2000. The limits of fixed-order computation. *Theoretical Computer Science* 254, 1-2, 465–499.
- SLOTA, M. AND LEITE, J. 2010a. On semantic update operators for answer-set programs. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*, H. Coelho, R. Studer, and M. Wooldridge, Eds. Frontiers in Artificial Intelligence and Applications, vol. 215. IOS Press, Lisbon, Portugal, 957–962.
- SLOTA, M. AND LEITE, J. 2010b. Towards Closed World Reasoning in Dynamic Open Worlds. *Theory and Practice of Logic Programming, 26th Int'l. Conference on Logic Programming (ICLP'10) Special Issue 10*, 4-6 (July), 547–564.
- SLOTA, M. AND LEITE, J. 2011. Back and forth between rules and SE-models. In *Proceedings of the 11th International Conference on Logic Programming and Nonmonotonic Reasoning (LPNMR-11)*, J. P. Delgrande and W. Faber, Eds. Lecture Notes in Computer Science, vol. 6645. Springer, Vancouver, Canada, 174–186.
- SLOTA, M., LEITE, J., AND SWIFT, T. 2011. Splitting and updating hybrid knowledge bases. *Theory and Practice of Logic Programming, 27th Int'l. Conference on Logic Programming (ICLP'11) Special Issue 11*, 4-5, 801–819.

- SWIFT, T. 1999. A new formulation of tabled resolution with delay. In *Recent Advances in Artificial Intelligence*. LNAI, vol. 1695. Springer, 163–177.
- SWIFT, T., PINTO, A. M., AND PEREIRA, L. M. 2009. Incremental answer completion. In *Logic Programming, 25th International Conference, ICLP 2009, Pasadena, CA, USA, July 14-17, 2009. Proceedings*, P. M. Hill and D. S. Warren, Eds. 519–524.
- TARSKI, A. 1955. Lattice-theoretic fixpoint theorem and its applications. *Pacific Journal of Mathematics* 5, 2, 285–309.
- VAN GELDER, A. 1989. The alternating fixpoint of logic programs with negation. In *Principles of Database Systems*. ACM Press, 1–10.
- VAN GELDER, A., ROSS, K. A., AND SCHLIPF, J. S. 1991. The well-founded semantics for general logic programs. *Journal of the ACM* 38, 3, 620–650.