COMBINING RULES AND ONTOLOGIES INTO CLOPEN KNOWLEDGE BASES

Labinot Bajraktari        Magdalena Ortiz        Mantas Šimkus

INFSYS RESEARCH REPORT 1843-17-04
NOV 2017
**Combining Rules and Ontologies into Clopen Knowledge Bases**

Labinot Bajraktari, Magdalena Ortiz, Mantas Šimkus

**Abstract.** We propose *Clopen Knowledge Bases (CKBs)* as a new formalism combining Answer Set Programming (ASP) with ontology languages based on first-order logic. CKBs generalize the prominent *r-hybrid* and *DL+LOG* languages of Rosati, and are more flexible for specification of problems that combine open-world and closed-world reasoning. We argue that the *guarded negation fragment* of first-order logic (GNFO)—a very expressive fragment that subsumes many prominent ontology languages like Description Logics (DLs) and the *guarded fragment*—is an ontology language that can be used in CKBs while enjoying decidability for basic reasoning problems. We further show how CKBs can be used with expressive DLs of the *ALC* family, and obtain worst-case optimal complexity results in this setting. For DL-based CKBs, we define a fragment called *separable CKBs* (which still strictly subsumes r-hybrid and *DL+LOG* knowledge bases), and show that they can be rather efficiently translated into standard ASP programs. This approach allows us to perform basic inference from separable CKBs by reusing existing efficient ASP solvers. We have implemented the approach for separable CKBs containing ontologies in the DL *ALCH*, and present in this paper some promising empirical results for real-life data. They show that our approach provides a dramatic improvement over a naive implementation based on a translation of such CKBs into *dl-programs*.

---

1Institute of Information Systems, Vienna University of Technology, Austria. E-mail: lbajrakt@kr.tuwien.ac.at.
2Institute of Information Systems, Vienna University of Technology, Austria. E-mail: ortiz@kr.tuwien.ac.at.
3Institute of Information Systems, Vienna University of Technology, Austria. E-mail: simkus@dbai.tuwien.ac.at.

**Acknowledgements:** This work was supported by the Austrian Science Fund (FWF) via projects P30360, P25207, P25518, and W1255.

Copyright © 2017 by the authors
1 Introduction

Answer Set Programming (ASP) and ontology languages like Description Logics (DLs) play leading roles in Knowledge Representation and Reasoning (KR&R). ASP and DLs have largely orthogonal features because they make very different assumptions regarding the completeness of information, and thus reasoning techniques and algorithms that are deployed in ASP are significantly different from the ones used in DLs. Combining ASP, which makes the closed-world assumption (CWA), with DLs, which make the open-world assumption (OWA), into expressive hybrid languages that would enjoy the positive features of both has received significant attention in the last decade (see, e.g., [21, 22, 6, 19, 18]). However, the progress on understanding the relationship between different hybrid languages, and their relationship with more standard languages like plain ASP, has been limited, as has the development of efficient reasoning algorithms and implementations.

These and related problems are investigated in this paper for a new hybrid language called Clopen Knowledge Bases (CKBs), which generalizes and improves the prominent r-hybrid language [21], and DL+LOG [22]. Each CKB is a triple $H = (P, \varphi, \Sigma)$, where $P$ is a disjunctive Datalog program with “not” literals in rule bodies (Datalog$^\neg$), $\varphi$ is a theory (e.g., in first-order logic), and $\Sigma$ is a set of predicate symbols. Intuitively, $\Sigma$ specifies the predicates that should be interpreted under the OWA; the remaining predicates should be interpreted under the CWA. Our contributions can be summarized as follows:

- We introduce CKBs and define for them a stable model semantics, inspired by the semantics given by Rosati to r-hybrid and DL+LOG KBs. In a nutshell, the major difference between the latter formalisms and CKBs is that CKBs allow to use CWA predicates in the theory. This allows for more convenient knowledge representation, but also causes technical challenges.
- We study automated reasoning in CKBs. To this end, we first provide a general decidability result for checking entailment of ground atoms and consistency testing in CKBs $H = (P, \varphi, \Sigma)$, where $\varphi$ is expressed in the guarded negation fragment of FO (GNFO) [3]. This is a very expressive fragment that subsumes the more prominent guarded fragment of FO, as well as many expressive DLs. We give a NExpTime$^{2\text{ExpTime}}$ upper bound for inference from GNFO-based CKBs (we note that satisfiability of GNFO formulas is 2ExpTime-hard).
- We next study the reasoning in CKBs $H = (P, \varphi, \Sigma)$ where $\varphi$ is expressed in the very expressive DL ALC$^H$OT, which extends the basic DL ALC with role hierarchies, inverse roles, and nominals. We show that the (combined) complexity of reasoning in such CKBs is not higher than in standard (non-ground) ASP. If we assume bounded predicate arities in rules, the basic reasoning problems are ExpTime-complete, which coincides with the complexity of standard problems in plain ALC$^H$OT.
- We study the relationship between CKBs and other existing hybrid languages. We define a restricted class of separable CKBs, and show that they can be transformed in polynomial time into the so-called dl-programs [6]. These CKBs still generalize r-hybrid KBs, thus we establish a connection between r-hybrid KBs and dl-programs that is interesting in its own right. The dl-programs resulting from this transformation effectively implement a naive algorithm for reasoning in CKBs. However, our experiments with the dlvhex suite (an implementation of dl-programs; see [20]) show that this approach is not suitable for a practical implementation of CKBs.
- We address the above mentioned inefficiency by developing translations from separable CKBs into standard ASP programs, thus enabling the reuse of existing ASP solvers. Roughly, the necessary knowledge about the ontology is compiled into a set of disjunctive Datalog rules. Together with the original rules of the CKB, they form an ASP program whose stable models are in close correspondence with the stable models of the input CKB. We define two translations. The first data-independent one establishes a connection to
ASP, showing that ASP is as expressive as separable CKBs. The other data-dependent translation is geared towards implementation, exploiting the structure of the data in the input CKB to reduce non-deterministic choices.

- We have implemented the data-dependent translation for separable CKBs with \( ALC\mathcal{H} \) ontologies, and present here some promising empirical results. In particular, our approach provides a dramatic improvement over the naive implementation based on a direct encoding into dl-programs.

An extended version of this paper containing selected proofs can be found here: http://www.kr.tuwien.ac.at/research/reports/rr1704.pdf

2 Preliminaries

In this paper we talk about logics which are, in general, sets of theories, and our results are for specific logics that are fragments of standard FO. We start by introducing the notions of (relational) interpretations, as usual in FO, and Herbrand interpretations, as usual in rule languages.

Interpretations and models. Assume a countably infinite set \( S_{\text{const}} \) of constants, and a countably infinite set \( S_{\text{pred}} \) of predicate symbols. Each \( r \in S_{\text{pred}} \) is associated with a non-negative integer, called the arity of \( r \). An interpretation is a pair \( I = (\Delta^I, \cdot^I) \) that consists of a non-empty set \( \Delta^I \) (called \textit{domain}), and a valuation function \( \cdot^I \) that maps (i) each constant \( c \in S_{\text{const}} \) to an element \( c^I \in \Delta^I \), and (ii) each predicate symbol \( r \) to a set \( r^I \subseteq (\Delta^I)^n \), where \( n \) is the arity of \( r \).

We assume a countably infinite set \( T \) of theories. Each theory \( \varphi \in T \) is associated with a set \( \text{mods}(\varphi) \) of interpretations. Each \( I \in \text{mods}(\varphi) \) is called a model of \( \varphi \). We assume that \( \top \in T \), and we let \( \text{mods}(\top) \) be the set of all interpretations. A logic is simply a set of theories \( \mathcal{L} \subseteq T \). As concrete logics we will consider various fragments of FO; the notion of a model for a theory \( \varphi \) in FO is the standard one.

Atoms and Herbrand interpretations. We assume a countably infinite set \( S_{\text{var}} \) of variables. The elements of \( S_{\text{const}} \cup S_{\text{var}} \) are called terms. An atom is an expression of the form \( r(t_1, \ldots, t_n) \), where \( r \in S_{\text{pred}} \), \( n \) is the arity of \( r \), and \( t_1, \ldots, t_n \) are terms. An atom is called \textit{ground} if no variables occur in it. An \textit{Herbrand interpretation} \( I \) is any set of ground atoms. An Herbrand interpretation \( I \) can be seen as an ordinary interpretation \( \tilde{I} = (\Delta^{\tilde{I}}, \cdot^{\tilde{I}}) \), where we let (i) \( \Delta^{\tilde{I}} = S_{\text{const}} \), and (ii) \( r^{\tilde{I}} = \{ \tilde{u} \mid r(\tilde{u}) = I \} \) for all \( r \in S_{\text{pred}} \).

3 Clopen Knowledge Bases

We now formally define our new hybrid language.

Syntax. A rule \( \rho \) is an expression of the form

\[
p_1 \lor \ldots \lor p_k \leftarrow p_{k+1}, \ldots, p_l, \text{not } p_{l+1}, \ldots, \text{not } p_m
\]

where \( p_1, \ldots, p_m \) are atoms. An expression \( \text{not } p \), with \( p \) an atom, is a negated atom. We let \( \text{head}(\rho) = \{ p_1, \ldots, p_k \} \), \( \text{body}^+(\rho) = \{ p_{k+1}, \ldots, p_l \} \), and \( \text{body}^-(\rho) = \{ p_{l+1}, \ldots, p_m \} \).

A program \( \mathcal{P} \) is a set of rules. A \textit{Clopen Knowledge Base (CKB)} is a triple \( \mathcal{H} = (\mathcal{P}, \varphi, \Sigma) \), where \( \mathcal{P} \) is a program, \( \varphi \in T \) is a theory, and \( \Sigma \subseteq S_{\text{pred}} \). The predicate symbols in \( \Sigma \) (resp., in \( S_{\text{pred}} \setminus \Sigma \)) are called the open predicates (resp., closed predicates) w.r.t. \( \mathcal{H} \). The CKB \( \mathcal{H} \) is called safe if the following holds for every rule \( \rho \in \mathcal{P} \): each variable occurring in \( \rho \) appears in some atom \( r(\tilde{u}) \in \text{body}^+(\rho) \) with \( r \notin \Sigma \). Unless stated otherwise, all considered CKBs are safe.
A rule or program is called ground (resp., positive) if no variables (resp., negated atoms) occur in it. A ground rule \( r(\bar{u}) \leftarrow \) is called a fact. We write \( r(\bar{u}) \in \mathcal{P} \) in case the fact \( r(\bar{u}) \leftarrow \) is present in the program \( \mathcal{P} \).

As usual, \( \text{dom}(f) \) and \( \text{ran}(f) \) denote the domain and range of a function \( f \), respectively. A substitution \( \sigma \) is any partial function from \( \text{S}_{\text{var}} \) to \( \text{S}_{\text{const}} \). For a rule \( \rho \) and a substitution \( \sigma \), we use \( \sigma(\rho) \) to denote the rule that is obtained from \( \rho \) by replacing every variable \( X \in \text{dom}(\sigma) \) with \( \sigma(X) \). The grounding of a program \( \mathcal{P} \), denoted \( \text{ground}(\mathcal{P}) \), is the ground program that consists of all ground rules \( \rho' \) such that \( \rho' = \sigma(\rho) \) for some \( \rho \in \mathcal{P} \) and some substitution \( \sigma \). Note that \( \text{ground}(\mathcal{P}) \) is infinite in case \( \mathcal{P} \) has at least one variable.

**Semantics.** An Herbrand interpretation \( I \) is called a model of a ground positive program \( \mathcal{P} \) if \( \text{body}^+(\rho) \subseteq I \) implies \( \text{head}(\rho) \cap I \neq \emptyset \) for all \( \rho \in \mathcal{P} \). Furthermore, \( I \) is a minimal model of the program \( \mathcal{P} \) if, in addition, there is no \( J \subseteq I \) such that \( J \) is a model of \( \mathcal{P} \).

Given a program \( \mathcal{P} \), an Herbrand interpretation \( I \), and \( \Sigma \subseteq \text{S}_{\text{pred}} \), the reduct \( \mathcal{P}^{I,\Sigma} \) of \( \mathcal{P} \) w.r.t. \( I \) and \( \Sigma \) is the ground positive program obtained from \( \text{ground}(\mathcal{P}) \) in two steps:

1. First, delete every rule \( \rho \) that contains
   - (a) \( r(\bar{u}) \in \text{body}^+(r) \) with \( r \in \Sigma \) and \( r(\bar{u}) \notin I \),
   - (b) \( r(\bar{u}) \in \text{head}(r) \) with \( r \in \Sigma \) and \( r(\bar{u}) \in I \), or
   - (c) \( r(\bar{u}) \in \text{body}^-(r) \) with \( r(\bar{u}) \in I \).

2. In the remaining rules, delete all negated atoms, and all ordinary atoms \( r(\bar{u}) \) with \( r \in \Sigma \).

An Herbrand interpretation \( I \) is a stable model of a CKB \( \mathcal{H} = (\mathcal{P}, \varphi, \Sigma) \) if the following two conditions are satisfied:
- \( \{r(\bar{u}) \mid r(\bar{u}) \in I, r \notin \Sigma\} \) is a minimal model of \( \mathcal{P}^{I,\Sigma} \), and
- \( \bar{I} \) is model of \( \varphi \).

**Reasoning problems.** As usual in hybrid languages (see, e.g., [21]), the basic reasoning task for CKBs is entailment of ground atoms. That is, given a CKB \( \mathcal{H} = (\mathcal{P}, \varphi, \Sigma) \) and a ground atom \( R(\bar{u}) \), the problem is to decide whether \( R(\bar{u}) \in I \) holds for all stable models \( I \) of \( \mathcal{H} \). This problem can be reduced to checking the non-existence of a stable model for the CKB \( \mathcal{H}' = (\mathcal{P} \cup \{ \leftarrow R(\bar{u}) \}, \varphi, \Sigma) \). Thus in the rest of the paper we focus on checking the stable model existence for a given CKB. Note that in general a CKB may have infinitely many stable models.

**Example 1.** The CKB \( \mathcal{H} = (\mathcal{P}, \varphi, \Sigma) \) contains information on the local transport network (provided by the city’s transport authority and assumed to be complete) and on hotels and relevant locations (extracted from the web and not necessarily complete). We have \( \mathcal{P} = \mathcal{P}_1 \cup \mathcal{P}_2 \cup \mathcal{P}_3 \), where \( \mathcal{P}_1 \) and \( \mathcal{P}_2 \) contain facts. The network, which is depicted by solid lines at the bottom of Figure 1, is described in \( \mathcal{P}_1 \). Facts of the form \( \text{RouteTable}(\ell, s, s') \leftarrow \text{store that on the line } \ell, \text{ station } s \text{ is followed by station } s' \). The constants \( t_1 \) and \( t_2 \) represent tram lines, while \( \ell_1 \) represents a metro line; we have corresponding facts \( \text{MetroLine}(\ell_1) \), \( \text{TramLine}(t_1) \), \( \text{TramLine}(t_2) \). \( \mathcal{P}_2 \) contains facts related to locations, including the following (for convenience, \( \text{CloseTo} \) is depicted with dotted lines).

\[
\begin{align*}
\text{CloseTo}(c_1, s_1) & \leftarrow \text{Hotel}(h_1) \leftarrow \text{TramConn}(h_1) \leftarrow \\
\text{CloseTo}(h_2, s_4) & \leftarrow \text{Hotel}(h_2) \leftarrow \\
\end{align*}
\]

The (self-explanatory) rules in \( \mathcal{P}_3 \) and the theory \( \varphi \) are shown in Figure 1 (URailConn stands for urban rail connection). If \( h \) is a hotel with direct connection to the point of interest \( c_1 \), then \( Q(h) \) holds for it.
\[ P_3 = \{ \text{MetroStation}(Y_1) \leftarrow \text{RouteTable}(X, Y_1, Y_2), \text{MetroLine}(X) \]
\[ \text{TramStation}(Y_2) \leftarrow \text{RouteTable}(X, Y_1, Y_2), \text{TramLine}(X) \]
\[ \text{ReachOnLine}(X, Y_1, Y_2) \leftarrow \text{RouteTable}(X, Y_1, Y_2) \]
\[ \text{ReachOnLine}(X, Y_1, Y_2) \leftarrow \text{ReachOnLine}(X, Y_1, Y_2), \text{RouteTable}(X, Y_2, Y_3) \]
\[ \text{TramOnly}(X) \leftarrow \text{TramConn}(X), \text{not} \text{MetroConn}(X) \]
\[ Q(X) \leftarrow \text{Hotel}(X), \text{CloseTo}(X, Y), \text{ReachOnLine}(Z, Y, Y'), \text{CloseTo}(c_1, Y') \]
\[ Q'(X) \leftarrow Q(X), \text{not} \text{TramOnly}(X) \}

\[ \varphi = \{ \forall x. (\text{MetroStation}(x) \lor \text{TramStation}(x) \leftrightarrow \text{Station}(x)), \]
\[ \forall x. (\text{TramConn}(x) \leftrightarrow \exists y \text{CloseTo}(x, y) \land \text{TramStation}(y)), \]
\[ \forall x. (\text{MetroConn}(x) \leftrightarrow \exists y \text{CloseTo}(x, y) \land \text{MetroStation}(y)), \]
\[ \forall x. (\text{URailConn}(x) \leftrightarrow \exists y \text{CloseTo}(x, y) \land \text{Station}(y)) \} \]

Figure 1: Example CKB

In this case, it holds for both \( h_1 \) and \( h_2 \) (note that we do not know which station \( h_1 \) is close to). We can use negation as failure to further exclude hotels for which a tram connection is explicitly mentioned, but no metro connection, hence we can assume that it is only reachable by tram, like \( h_1 \). For this reason, \( Q' \) only holds for \( h_2 \). The predicates that describe the network, and those that occur in the heads of the rules in \( P_3 \) are closed. The remaining ones are open, i.e. \( \Sigma = \{ \text{Hotel}, \text{CloseTo}, \text{Station}, \text{TramConn}, \text{MetroConn}, \text{URailConn} \} \).

In the spirit of r-hybrid and \( DL+LOG \) KBs, the FO theory of a CKB can be seen as a set of integrity constraints on the inferences made using the rules of the CKB. Since we are not in classical logic, and in particular because double negation elimination is not valid, “moving” a fact from the program to its theory need not preserve the stable models.

Example 2. We let \( \Sigma = \{ \text{Edge} \} \) and
\[ \varphi = \{ \forall xy \text{Edge}(x, y) \rightarrow (\text{Node}(x) \land \text{Node}(y)) \} \]
\[ P = \text{Node}(v_1) \leftarrow \ldots \text{Node}(v_n) \leftarrow ; \]
\[ \text{Reach}(X, X) \leftarrow \text{Node}(X); \]
\[ \text{Reach}(X, Z) \leftarrow \text{Reach}(X, Y), \text{Edge}(Y, Z), \text{Node}(Z); \}

Then these CKBs are not equivalent:
\[ \mathcal{H}_1 = (P, \varphi \land \text{Reach}(v_1, v_2), \Sigma) \]
\[ \mathcal{H}_2 = (P \cup \{ \text{Reach}(v_1, v_2) \rightarrow \}, \varphi, \Sigma) \]

Indeed, each stable model of \( \mathcal{H}_1 \) correspond to a directed graph \( G \) over \( v_1, \ldots, v_n \) such that \( (v_1, v_2) \) is included in the reflexive transitive closure of the edge relation in \( G \). In contrast, a stable model of \( \mathcal{H}_2 \) consists of an arbitrary graph over \( v_1, \ldots, v_n \), together with the reflexive transitive closure of the edge relation augmented with the tuple \( (v_1, v_2) \).

Relationship to ASP. Assume a program \( P \) and an Herbrand interpretation \( I \). We call \( I \) a stable model of \( P \) if \( I \) is a stable model of the CKB \( \mathcal{H} = (P, T, \emptyset) \). It is not difficult to see that this definition yields precisely the stable models that can alternatively be computed using the standard definition of stable model.
semantics in ASP. Indeed, the program $\mathcal{P}^{I,\emptyset}$ boils down to the standard Gelfond-Lifschitz reduct $\mathcal{P}^I$ of $\mathcal{P}$ w.r.t. $I$ [12]. Observe that in a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \emptyset)$, the theory $\varphi$ plays the role of integrity constraints on the stable models of the plain program $\mathcal{P}$, i.e. $I$ is a stable model of $\mathcal{H}$ iff $I$ is a stable model of $\mathcal{P}$ such that $I \in \text{mods}(\varphi)$.

**Relationship to r-hybrid KBs.** Our CKBs are a close relative of the r-hybrid KBs of Rosati [21]. The safety restriction here is inspired by the safety condition in r-hybrid KBs, and so is our definition of the semantics via a generalization of the Gelfond-Lifschitz reduct that additionally reduces the program according to the truth value of atoms over open predicates. Intuitively, r-hybrid KBs are a special kind of CKBs in which the rule component can refer to both open and closed predicates, but the theory component can use open predicates only. More formally, an r-hybrid KB $\mathcal{H} = (\varphi, \mathcal{P})$, where $\varphi$ is a theory in FO and $\mathcal{P}$ is a Datalog$\neg\vee$ program, corresponds to the CKB $\mathcal{H}' = (\mathcal{P}, \varphi, \Sigma)$, where $\Sigma$ is the set of predicates symbols appearing in $\varphi$. One can verify that the stable models of $\mathcal{H}'$ are exactly the so-called NM-models of $\mathcal{H}$.

In generic CKBs $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$, the set $\Sigma$ need not contain all the predicate symbols that appear in $\varphi$. That is, closed predicates may occur in $\varphi$, and the extensions of these predicates in (the relevant) models of $\varphi$ must be justified by program rules. This feature causes technical challenges, but is very useful for declarative specification of problems: in our approach, predicates under the OWA and the CWA can be used both in the program and in the theory of a hybrid KB (see Example 1 for an illustration).

The $\mathcal{DL}+\mathcal{LOG}$ language is obtained from r-hybrid KBs by allowing only DLs for specifying theories, and relaxing the safeness condition to weak safeness [22]. In the extended version we show that, when sufficiently rich DLs are considered, CKBs also generalize $\mathcal{DL}+\mathcal{LOG}$.

**Active domain predicate.** For convenience, we assume the availability of a unary “built-in” predicate $\text{adom}$ that, intuitively, stores the constants that appear in a given program. More precisely, for any program $\mathcal{P}$ and each $n$-ary relation symbol $r$ with $r \neq \text{adom}$ that appears in $\mathcal{P}$, we assume that (i) $\mathcal{P}$ contains the rule $\text{adom}(X_j) \leftarrow r(X_1, \ldots, X_n)$ for every $1 \leq j \leq n$, and (ii) $\text{adom}$ is allowed to occur only in bodies of the remaining rules.

### 4 Decidable CKBs

We now turn to identifying useful settings in which the existence of a stable model for a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ is decidable. This naturally requires $\varphi$ to belong to a logic $\mathcal{L}$ in which satisfiability is decidable (i.e., the set $\{ \varphi \in \mathcal{L} \mid \text{mods}(\varphi) \neq \emptyset \}$ should be recursive). However, this alone is not enough, since we will in general be interested in models of $\varphi$ where a selected set of closed predicates have a concrete extension that is given as input. We will see that this calls for logics with a rather flexible support for equality reasoning.

Towards providing a quite general decidability result for checking stable model existence in CKBs, we first define a simple program that allows us to freely “guess” the extensions of open predicates of a given CKB $\mathcal{H}$. These extensions are restricted to constants that appear in $\mathcal{H}$.

**Definition 1 (Program Choose($\mathcal{H}$)).** Assume a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$. For every $n$-ary relation symbol $r \in \Sigma$, let $\tau$ be a fresh $n$-ary relation symbol that does not appear in $\mathcal{H}$. We let Choose($\mathcal{H}$) be the set that contains

$$r(Y_1, \ldots, Y_n) \lor \tau(Y_1, \ldots, Y_n) \leftarrow \text{adom}(Y_1), \ldots, \text{adom}(Y_n)$$

for each $n$-ary relation symbol $r \in \Sigma$ that appears in $\mathcal{P}$.

A stable model $I$ of $\mathcal{P} \cup \text{Choose}(\mathcal{H})$ can be seen as a (partially complete) candidate for a stable model of a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$. The following proposition, whose proof relies on the imposed CKB safety requirement, tells us when such an $I$ witnesses the existence of a stable model of $\mathcal{H}$.
Proposition 1. A CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ has a stable model iff $\mathcal{P} \cup \text{Choose}(\mathcal{H})$ has some stable model $I$ for which there exists some $I \in \text{mods}(\varphi)$ with the following properties:

1. $(e_1^T, \ldots, e_n^T) \in r^I$ for all $r(c_1, \ldots, c_n) \in I$,
2. $(e_1^T, \ldots, e_n^T) \notin r^I$ for all $\bar{r}(c_1, \ldots, c_n) \in I$, and
3. if $(e_1, \ldots, e_n) \in r^I$ and $r \notin \Sigma$, then there exists $r(c_1, \ldots, c_n) \in I$ with $c_1^T = e_1, \ldots, c_n^T = e_n$.

From Proposition 1, we obtain decidability of stable model existence for $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ whenever we can list the stable models of $\mathcal{P} \cup \text{Choose}(\mathcal{H})$ and test, for each of them, the existence of a model $I$ of the theory $\varphi$ satisfying conditions (C1–C3). Moreover, if the logic $\mathcal{L}$ in question is strong enough to express, for a fixed candidate $I$, conditions (C1–C3) as part of a theory in $\mathcal{L}$, then decidability of the underlying satisfiability problem suffices. This applies, in particular, to the guarded negation fragment (GNFO), which is among the most expressive FO fragments for which decidability has been established [3].

We use $\varphi[\bar{x}]$ to indicate that a FO formula $\varphi$ has $\bar{x}$ as free variables. The fragment GNFO contains all formulas that can be built using the following grammar:

$$\varphi ::= r(v_1, \ldots, v_n) \mid v = u \mid \exists x \varphi \mid \varphi \land \varphi \mid \varphi \lor \varphi \mid \alpha \land \neg \varphi[\bar{x}],$$

where $u, v, v_1, \ldots, v_n$ are terms, and $\alpha$ is an atom or an equality statement such that all variables of $\bar{x}$ also occur in $\alpha$. Intuitively, in GNFO a subformula can be negated only if its free variables are “guarded” by an atom or an equality statement. Observe also that a subformula with a single free variable $x$ can always be guarded by an equality statement $x = x$. GNFO is flexible and natural for domain modelling; for instance, the theory $\varphi$ in Example 1 is in GNFO.

Theorem 1. Checking the stable model existence in CKBs $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$, where $\varphi$ is in GNFO, is decidable. The problem belongs to the class $\text{NExpTime}^{2\text{ExpTime}}$, and is $2\text{ExpTime}$-hard.

Proof. Assume a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ with $\varphi$ in GNFO. Let $\Sigma_c$ be the set of predicates that occur in $\mathcal{P}$ but not in $\Sigma$. For every $n$-ary predicate symbol $r \in \Sigma_c$, assume a tuple $\bar{x}_r = (x_1^r, \ldots, x_n^r)$ of variables. Assume a stable model $I$ of $\mathcal{P} \cup \text{Choose}(\mathcal{H})$. For such $I$, let $\psi(I)$ be the following formula:

$$\psi(I) = \bigwedge_{r(\bar{c}) \in I} r(\bar{c}) \land \bigwedge_{\bar{c}(\bar{c}) \in I} \neg r(\bar{c}) \land \bigwedge_{r \in \Sigma_c} \forall x_1^r \ldots \forall x_n^r \left( r(x_1^r, \ldots, x_n^r) \rightarrow \bigvee_{r(c_1, \ldots, c_n) \in I} \left( \bigwedge_{1 \leq i \leq n} (x_i^r = c_i) \right) \right)$$

One can check that the formula $\varphi \land \psi(I)$ is in GNFO. Note that the three conjuncts mimic the conditions (C1)–(C3); the third one relies on the availability of equality, and is essentially the same formula used in [4] for reasoning about visible and invisible tables in databases. The following holds: $\psi(I)$ is satisfiable iff there exists $I \in \text{mods}(\varphi)$ that satisfies the conditions (C1–C3) of Proposition 1. Overall, this means that $\mathcal{H}$ has a stable model iff $\mathcal{P} \cup \text{Choose}(\mathcal{H})$ has some stable model $I$ such that $\varphi \land \psi(I)$ is satisfiable. Keeping in mind that satisfiability in GNFO is $2\text{ExpTime}$-complete, this equivalence yields the $\text{NExpTime}^{2\text{ExpTime}}$ upper bound. Indeed, we can decide the existence of a stable model for $\mathcal{H}$ by non-deterministically guessing a candidate stable model $I$ of $\mathcal{P} \cup \text{Choose}(\mathcal{H})$, whose size is at most exponential in the size of $\mathcal{H}$, and then checking that (i) $I$ is a minimal model of $\mathcal{P} \cup \emptyset$, and (ii) that the formula $\psi(I)$ is satisfiable. The lower bound is carried over trivially from GNFO. □
5 CKBs and Description Logics

GNFO is very expressive and thus also computationally very expensive. In this section, we study DL-based CKBs, and show that such CKBs are (to a large extent) computationally not more expensive than plain ASP.

We first recall the syntax and semantics of the expressive DL $\mathcal{ALCHOI}$.

We assume a countably infinite set $S_{cn} \subseteq S_{pred}$ of unary relation symbols, called concept names, and a countably infinite set $S_{rn} \subseteq S_{pred}$ of binary relation symbols, called role names. If $R \in S_{rn}$, then $R$ and $R^{-}$ are roles. (Complex) concepts are defined as follows: (a) the symbols $\top$, $\bot$, and every concept name $A \in S_{cn}$ is a concept, (b) if $a \in S_{const}$, then $\{a\}$ is a concept (called nominal), and (c) if $C, D$ are concepts and $R$ is a role, then $C \sqcap D$, $C \sqcup D$, $\neg C$, $\forall R.C$, $\exists R.C$ are also concepts. Assume an interpretation $I = (\Delta^I, \mathcal{I})$, and observe that $A^I \subseteq \Delta^I$ for all concept names $A$, and $R^I \subseteq \Delta^I \times \Delta^I$ for all role names $R$. The semantics to all complex concepts and roles beyond concept and role names is given by extending the valuation function $\cdot^I$ in the usual way (see [2]; for convenience, we provide it in the extended version).

A TBox (or ontology) $\mathcal{T}$ is a finite set of axioms of the forms $C \sqsubseteq D$ (called concept inclusions), where $C$ and $D$ are concepts, and $R \sqsubseteq S$ (called role inclusions), where $R$ and $S$ are roles. Given a TBox $\mathcal{T}$, we define $\sqsubseteq^\mathcal{T}$ as the reflexive transitive closure of the relation $\sqsubseteq^\mathcal{T}$ that contains $R \sqsubseteq^\mathcal{T} S$ and $R^{-} \sqsubseteq^\mathcal{T} S^{-}$ for all role inclusions $R \sqsubseteq S$ in $\mathcal{T}$. An interpretation $I$ is a model of a TBox $\mathcal{T}$ if $C^I \subseteq D^I$ for each concept inclusion $C \sqsubseteq D \in \mathcal{T}$, and $R^I \subseteq S^I$ for each role inclusion $R \sqsubseteq S \in \mathcal{T}$. A TBox is satisfiable if it has some model. We note that satisfiability of $\mathcal{ALCHOI}$ TBoxes is EXPTime-complete [2].

Example 3. The theory $\varphi$ in Example 1 can be written in the syntax of $\mathcal{ALCHOI}$ as follows (we use the axiom $C \equiv D$ as a shortcut for the two inclusions $C \sqsubseteq D, D \sqsubseteq C$):

$$\mathcal{T} = \{ \text{MetroStation} \sqcup \text{TramStation} \equiv \text{Station}, \text{TramConn} \equiv \exists \text{CloseTo}. \text{TramStation}, \text{MetroConn} \equiv \exists \text{CloseTo}. \text{MetroStation}, \text{URailConn} \equiv \exists \text{CloseTo}. \text{Station} \}$$

The following theorem can be proven using (well) known complexity results from DLs and ASP, in combination with an encoding of condition (C3) of Proposition 1 by means of nominals, similarly to the encoding of DBoxes in [10] (see the extended version).

Theorem 2. Deciding stable model existence in CKBs $\mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma)$, where $\mathcal{T}$ is an $\mathcal{ALCHOI}$ TBox, is $\text{NEXPTime}^{\text{NP}}$-complete. If $\mathcal{P}$ is not disjunctive, the problem is $\text{NEXPTime}$-complete. The problem is $\text{EXPTime}$-complete, if (i) $\mathcal{P}$ is both non-disjunctive and positive, or (ii) the arity of predicate symbols in $\mathcal{P}$ is assumed to be bounded by a constant.

Proof. Assume a CKB $\mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma)$, where $\mathcal{T}$ is an $\mathcal{ALCHOI}$ TBox. Assume a stable model $I$ of $\mathcal{P} \cup \text{Choose}(\mathcal{H})$. For any such $I$, let $\text{TBBox}(\mathcal{H}, I)$ be the $\mathcal{ALCHOI}$ TBox that contains the following inclusions (in some axioms below we use $\{d_1, \ldots, d_n\}$ instead of $\{d_1\} \sqcup \ldots \sqcup \{d_n\}$):

- $\{c\} \sqsubseteq A$, for all $A(c) \in I$ with $A \in S_{cn}$, and $\{c\} \sqsubseteq \neg A$ for all $\overline{A}(c) \in I$ with $A \in S_{cn}$.
- $\{c\} \sqsubseteq \forall r.\{d\}$ for all $r(c, d) \in I$ with $r \in S_{rn}$, and $\{c\} \sqsubseteq \forall r.\neg \{d\}$ for all $\overline{r}(c, d) \in I$ with $r \in S_{rn}$.
- $A \sqsubseteq \bigcup_{A(c) \in I} \{c\}$, for all concept names $A \not\in \Sigma$,
- $\exists r \sqsubseteq \{d \mid \exists d' : r(d, d') \in I\}$, for all role names $r \not\in \Sigma$,
\[ \{ e \} \subseteq \forall r. \{ o | \exists e : r(e, o) \in I \} \] for all role names \( r \not\in \Sigma \) and all constants \( c \) that appear in \( I \).

The construction of \( \text{TBox}(\mathcal{H}, I) \) is inspired by a similar encoding in [10] where an expressive DL with the so-called \( \text{DBoxes} \) is translated into a standard DL with nominals.

Due to Proposition 1, and due to the construction of the above TBoxes, \( \mathcal{H} \) has a stable model iff \( \mathcal{P} \cup \text{Choose}(\mathcal{H}) \) has a stable model \( I \) such that \( \mathcal{T} \cup \text{TBox}(\mathcal{H}, I) \) is consistent. In other words, the consistency of \( \mathcal{H} \) can be decided by traversing the stable models \( I \) of \( \mathcal{P} \cup \text{Choose}(\mathcal{H}) \), for each such \( I \) building \( \mathcal{T} \cup \text{TBox}(\mathcal{H}, I) \) and checking its satisfiability. We note that \( \text{TBox}(\mathcal{H}, I) \) is always of polynomial size in the size of \( \mathcal{H} \), and consequently checking the consistency of \( \text{TBox}(\mathcal{H}, I) \) is feasible in single exponential time. From this observation, and the complexity of standard ASP under the syntactic restriction mentioned in the theorem, the completeness results follow.

\[ \Box \]

6 Translations and Implementation

We focus here on DL-based CKBs as described in the previous section, and provide translations from such CKBs to other formalisms, in particular to dl-programs and to plain ASP. The translations are given for a large fragment of CKBs, which we call separable CKBs, and which in fact generalizes r-hybrid KBs. To define the fragment we need the notion of a positive occurrence and a negative occurrence of a concept or role name \( \alpha \) in a concept \( C \). These notions are defined inductively as follows. (A) Every concept name \( A \) occurs positively in \( A \). (B) Every role name \( S \) with \( R \sqsubseteq \top \) \( S \) occurs positively in \( \exists R.C \), for any concept \( C \). (C) Every role name \( S \) with \( R \sqsubseteq \top \) \( S \) occurs negatively in \( \forall R.C \), for any concept \( C \). (D) If a concept name \( A \) occurs positively (resp., negatively) in \( C \), then \( A \) occurs positively (resp., negatively) in \( C \sqcap D \), \( C \sqcup D \), \( \forall R.C \) and \( \exists R.C \), for any concept \( D \) and role \( R \). (E) If a concept or role name \( \alpha \) occurs positively (resp., negatively) in \( C \), then \( \alpha \) occurs negatively (resp., positively) in \( \neg C \).

Definition 2 (Separability). A CKB \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \) is separable if the concept \( \prod_{C \in \mathcal{D} \in \mathcal{T}} (\neg C \sqcup D) \) does not have a positive occurrence of concept or role name \( \alpha \) with \( \alpha \not\in \Sigma \).

Example 4. Take the CKB \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \) with \( \mathcal{P} = \{ Q(X, Y, Z) \leftarrow T(X, Y), P(Y, Z) \} \), \( \mathcal{T} = \{ \exists R.(\exists P.A) \subseteq B \} \), and \( \Sigma = \{ R, A, B \} \). Then \( \mathcal{H} \) is separable because \( P \) occurs only negatively in \( \neg(\exists R.(\exists P.A)) \sqcup B \).

Intuitively, in a separable CKB \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \) the inclusions in \( \mathcal{T} \) can be used to infer the extensions of open predicates from the extensions of closed predicates and other predicates, but these axioms simply cannot assert membership of a domain element (resp., pair of elements) in a closed concept name (resp., role name). More concretely, for separable CKBs one can show a version of Proposition 1 where the condition (C3) is omitted (the rest of the proposition remains the same). The omission of condition (C3) is a major change: recall that we relied heavily on the equality predicate in GNFO, and on nominals supported in \( \text{ALCHQI} \) in order to cope with (C3). We note that separable CKBs capture r-hybrid KBs \( \mathcal{H} = (\mathcal{T}, \mathcal{P}) \) with \( \mathcal{T} \) an \( \text{ALCHQI} \) TBox. Such KBs, as mentioned, correspond to CKBs \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \), where \( \Sigma \) is the set of predicates symbols that appear in \( \mathcal{T} \), and which trivially satisfy the separability condition. We remark that the pair \( (\mathcal{T}, \mathcal{P}) \) with \( \mathcal{T}, \mathcal{P} \) from Example 4 is not a safe r-hybrid KB (neither is it weakly safe in the spirit of \( \text{DL+LOG} \)), because the variable \( Z \) does not appear in a rule atom with a predicate symbol that does not occur in \( \mathcal{T} \).

6.1 Translation into DL-programs

We can now show how a separable CKB \( \mathcal{H} \) can be translated into a dl-program \( \Pi_\mathcal{H} \) while preserving the existence of a stable model. Please see [6] for the definition of dl-programs; for convenience, in the extended
version we provide the definition of a core fragment of dl-programs that is sufficient for the encoding. From a separable CKB \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \) we build a dl-program \( \Pi_{\mathcal{H}} = (\mathcal{T}', \mathcal{P}') \) as follows. For every concept name \( A \) (resp., role name \( r \)) that appears in \( \mathcal{T} \), let \( A' \) be a fresh concept name (resp., let \( r' \) be a fresh role name). Then the TBox \( \mathcal{T}' \) is obtained from \( \mathcal{T} \) simply by replacing every concept and role name \( S \) by \( S' \). For the construction of \( \mathcal{P}' \), let \( S_1, \ldots, S_n \) be an arbitrary enumeration of the concept and role names that appear both in \( \mathcal{P} \) and \( \Sigma \). Then the set \( \mathcal{P}' \) of dl-rules is defined as follows:

\[
\mathcal{P}' = \mathcal{P} \cup \text{Choose}(\mathcal{H}) \cup \{ \leftarrow \text{DL}[\lambda; \bot](X) \},
\]

where \( \lambda = S'_1 \cup S_1, \ldots, S'_n \cup S_n, S'_1 \cup S_1, \ldots, S'_n \cup S_n \).

Intuitively, given a stable model \( I \) of \( \mathcal{P} \cup \text{Choose}(\mathcal{H}) \), the expression \( \lambda \) above allows us to check the conditions (C1) and (C2) of Proposition 1 (see the construction of \( \Pi_{\mathcal{H}}(\mathcal{I}, I) \) as used in the proof of Theorem 2 in the extended version). The constraint \( \leftarrow \text{DL}[\lambda; \bot](X) \) is then used to discard \( I \) in case the built TBox is inconsistent. Thus from this encoding we get the following result.

**Theorem 3.** A separable CKB \( \mathcal{H} \) has a stable model iff the dl-program \( \Pi_{\mathcal{H}} \) has an answer set.

### 6.2 Translation into Plain ASP

We describe here our translations from separable CKBs \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \) into standard ASP. Intuitively, instead of using a richer language than plain ASP to perform the search for \( \mathcal{I} \in \text{mods}(\varphi) \) with properties (C1) and (C2) described in Proposition 1 (as we did above with dl-programs), we perform reasoning about the TBox of an input KB during the translation so that afterwards the TBox can effectively be forgotten. Unlike our translation into dl-programs, this translation is not polynomial and may take single exponential time in the size of the input. However, our experiments show that in practice the latter performs much better than the former. The below translations are inspired by existing translation from expressive DLs into disjunctive Datalog [16, 8, 5]. In fact, we provide a pair of translations: a generic modular translation that is independent from the concrete facts in the input KB, and a restricted translation that does take into account the data (the latter was implemented). We limit this approach to \( \text{ALCH} \) (i.e., we do not support inverses and nominals).

We assume here TBoxes in normal form, that is, each axiom is of one of the following forms:

\[
\begin{align*}
A_1 \sqcap \ldots \sqcap A_n & \sqsubseteq B & A & \sqsubseteq B_1 \sqcup \ldots \sqcup B_m & A & \sqsubseteq \exists R.B & (I) \\
\exists R.A & \sqsubseteq B & A & \sqsubseteq \forall R.B & R & \sqsubseteq S & (II)
\end{align*}
\]

where \( A, B, A_i, B_i \) are concept names, \( \top \) or \( \bot \), and \( R, S \) are role names. It is well known that any TBox \( \mathcal{T} \) can be normalized into a TBox \( \mathcal{T}' \) in polynomial time so that \( \mathcal{T} \) and \( \mathcal{T}' \) have the same models up to the original signature of \( \mathcal{T} \) (see, e.g., [23]).

**Definition 3** (Communication rules \( \text{Comm}(\mathcal{H}) \)). For a separable CKB \( \mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma) \), let \( \text{Comm}(\mathcal{H}) \) denote the set of the following rules:

\[
\begin{align*}
S(X, Y) & \leftarrow R(X, Y) & \text{for each } & R & \sqsubseteq S & \in \mathcal{T} \\
B(X) & \leftarrow r(X, Y), A(Y) & \text{for each } & \exists R.A & \sqsubseteq B & \in \mathcal{T} \\
B(Y) & \leftarrow A(X), r(X, Y) & \text{for each } & A & \sqsubseteq \forall R.B & \in \mathcal{T}
\end{align*}
\]

The program \( \text{Comm}(\mathcal{H}) \) simply contains the direct translation of inclusions listed in (II). To deal with the remaining inclusions (i.e. the ones listed in (I)), we employ types.
**Definition 4 (Types).** A type is any set \( \tau \subseteq S_{cn} \cup \{ \neg A \mid A \in S_{cn} \} \). A type \( \tau \) is consistent w.r.t. a TBox \( T \) if there exists a model \( I \) of \( T \) and an element \( e \in \Delta^I \) such that \( e \in (\bigcap_{C \in \tau} C)^I \). We use \( \text{types}(T) \) to denote the set of types \( \tau \) such that (i) \( \tau \) is consistent w.r.t. \( T \), and (ii) \( A \in \tau \) or \( \neg A \in \tau \) for each concept name \( A \) in \( T \).

**Data-independent translation.** Assume a separable CKB \( \mathcal{H} = (P, T, \Sigma) \). For each \( \tau \in \text{types}(T) \), let \( \text{Type}_\tau \) be a fresh unary predicate symbol. We let \( \text{ASP}(\mathcal{H}) \) be the extension of \( P \cup \text{Choose}(\mathcal{H}) \cup \text{Comm}(\mathcal{H}) \) with the following rules:

(i) the rule \( \bigvee_{\tau \in \text{types}(T)} \text{Type}_\tau(X) \leftarrow \text{adom}(X) \)

(ii) for each type \( \tau \in \text{types}(T) \), the following constraints

\[
A(X) \leftarrow \text{Type}_\tau(X) \quad \text{for each } A \in \tau \cap S_{cn}
\]

\[
\neg \text{Type}_\tau(X), A(X) \quad \text{for each } \neg A \in \tau
\]

The program \( \text{ASP}(\mathcal{H}) \) above built from a CKB \( \mathcal{H} \) yields a tool to decide consistency of \( \mathcal{H} \). In fact, the rules additional to the original program \( P \) depend only on \( T \) and \( \Sigma \), and thus the translation is data-independent. Note that the set \( \text{types}(T) \) can be computed in single exponential time in the size of \( T \), and for this a standard DL reasoner can be used. Indeed, a type \( \tau \) is consistent w.r.t. \( T \) iff \( T \cup \{ A(c) \mid A \in \tau \} \cup \{ \neg A(c) \mid \neg A \in \tau \} \) has a model, for a fresh constant \( c \).

**Theorem 4.** The CKB \( \mathcal{H} = (P, T, \Sigma) \) has a stable model iff \( \text{ASP}(\mathcal{H}) \) has a stable model. In fact, for any set \( F \) of facts, \( \mathcal{H} = (P \cup F, T, \Sigma) \) has a stable model iff \( \text{ASP}(\mathcal{H}) \cup F \) has a stable model.

**Data-dependent translation.** Since \( |\text{types}(T)| \) is often exponential in the size of \( T \), the program \( \text{ASP}(\mathcal{H}) \) can be prohibitively large to be used in practice. We next present an optimized way to obtain a desired ASP program, by sacrificing data independence.

Assume a separable CKB \( \mathcal{H} = (P, T, \Sigma) \). For every constant \( c \) that appears in \( \mathcal{H} \), let \( t(c, \mathcal{H}) \) be the set of types returned by the non-failing runs of the following non-deterministic procedure:

1. Let \( \tau = \{ A \mid P \text{ has the fact } A(c) \leftarrow \} \).
2. Close \( \tau \) under the following inference rules:
   (a) If \( A_1 \land \cdots \land A_n \subseteq B \in T \) and \( \{ A_1, \ldots, A_n \} \subseteq \tau \), then add \( B \) to \( \tau \).
   (b) If \( \exists S.T \subseteq B \in T, R \sqsubseteq_S^+ S \), and \( P \) has the fact \( R(c, d) \leftarrow \) for some \( d \), then add \( B \) to \( \tau \).
   (c) If \( T \subseteq \forall S.B \in T, R \sqsubseteq_S^+ S \), and \( P \) has the fact \( R(d, c) \leftarrow \) for some \( d \), then add \( B \) to \( \tau \).
3. If \( \tau \) is inconsistent w.r.t. \( T \), then return failure.
4. Pick a concept name \( B \) such that \( \{ B, \neg B \} \cap \tau = \emptyset \), and \( B \) appears in one of the following:
   (a) in a non-fact rule of \( P \),
   (b) in some \( \exists R.A \subseteq B \in T \) or \( A \sqsubseteq \exists R.B \in T \) such that \( R \) appears in a non-fact rule of \( P \),
   (c) in some \( \exists S.A \subseteq B \in T \) such that \( P \) has the fact \( R(c, d) \leftarrow \) for some \( d \), and \( R \sqsubseteq_S^+ S \), or
   (d) in some \( A \sqsubseteq \forall R.B \in T \) such that \( P \) has the fact \( R(d, c) \leftarrow \) for some \( d \), and \( R \sqsubseteq_T^+ S \).
If the above $B$ does not exist, then return $\tau$. Otherwise, non-deterministically add to $\tau$ either $B$ or $\neg B$, and go to step (2).

Take a fresh unary predicate symbol $Type_\tau$ for each $\tau \in t(c, H)$ such that $c$ occurs in $H$. We let $ASP^{dd}(H)$ be the extension of $P \cup Comm(H)$ with the following rules:

(i) for all roles $R \in \Sigma$ that appear in a non-fact rule in $P$, the rule $R(X, Y) \lor \neg R(X, Y) \leftarrow \text{adom}(X), \text{adom}(Y)$, where $\neg R$ is a fresh relation symbol

(ii) for each constant $c$ of $H$, the rule $\bigvee_{\tau \in t(c, H)} Type_\tau(c) \leftarrow$

(iii) for each constant $c$ of $H$ and type $\tau \in t(c, H)$, the following constraints

$$A(c) \leftarrow Type_\tau(c) \quad \text{for each } A \in \tau \cap S_{cn}$$

$$\neg A(c) \leftarrow Type_\tau(c) \quad \text{for each } \neg A \in \tau$$

The translation allows us to decide stable model existence:

**Theorem 5.** The CKB $\mathcal{H} = (P, T, \Sigma)$ has a stable model iff $ASP^{dd}(H)$ has a stable model.

### 6.3 Implementation and Experiments

We present here some experiments that demonstrate the advantages of translating a separable CKB $\mathcal{H}$ into a plain program $ASP^{dd}(H)$. We have implemented our approach in a prototype reasoner. In particular, to build the function $t$ described previously, instead of relying on an external DL reasoner, we have implemented our own algorithm for testing consistency of types w.r.t. a TBox. It is designed in such a way that the consistency of several types can be tested simultaneously, using caching to avoid recomputation. Consistent types are stored in a database and can be reused for other hybrid knowledge bases over the same ontology.

Our implementation is written in Java and PostgreSQL 9.5.5 database, and uses OWLAPI [15] to manage ontologies. The ASP program resulting from the translation is evaluated with Clingo 4.2.1 [11]. The experiments were run on a PC with Intel Core i7 CPU and 16GB RAM running 64bit Linux-Mint 17. We compared the performance of our implementation with the direct encoding to dl-programs, as presented on page 5. The latter is implemented in dlvhex, which also uses Clingo.

For benchmarking, we used real-world OpenStreetMap data, transformed into Datalog facts following [9]. The data, describing the city of Vienna, is available as database dumps at BBBike. The extracted data contains facts about 19517 geographical points in the map treated as constants. Concept assertions were extracted from tags in the mapping data, for points of interest like Hotel, Restaurant, Shop, Hospital, MetroStation etc. There are also facts about relations between these points and other constants representing objects of interest such as metro lines, types of cuisine, dishes etc. Among plain Datalog relations, we extracted next, relating pairs of points whose distance is below a certain threshold set in meters. By considering different thresholds, ranging from 50 to 250 meters, we obtained sets of facts of different sizes. Other Datalog relations extracted to describe the Vienna metro network are locatedAlong and nextStation. The former relates a metro station to the corresponding metro line, and the latter relates pairs of consecutive stations on the same line. The extracted relations that also occur in $T$ include roles like hasCuisine and serves, which relate a Restaurant to a Cuisine or a Dish, respectively. As TBox for our CKBs, we used the DL-Lite$_R$ ontology from the MyITS Project [7], enriched with $ALCH$ axioms.

---

1. [https://www.openstreetmap.org](https://www.openstreetmap.org)
Table 1: Number of facts for different next relations, and running times in seconds for $P_1$–$P_4$

<table>
<thead>
<tr>
<th></th>
<th>next50</th>
<th>next100</th>
<th>next150</th>
<th>next200</th>
<th>next250</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fact count</td>
<td>145014</td>
<td>263075</td>
<td>479283</td>
<td>743935</td>
<td>1053335</td>
</tr>
<tr>
<td>$P_1$</td>
<td>19.6</td>
<td>30.1</td>
<td>44.6</td>
<td>60.2</td>
<td>87.6</td>
</tr>
<tr>
<td>$P_2$</td>
<td>19.6</td>
<td>31.8</td>
<td>52.7</td>
<td>64.0</td>
<td>95.4</td>
</tr>
<tr>
<td>$P_3$</td>
<td>19.6</td>
<td>32.8</td>
<td>56.1</td>
<td>64.7</td>
<td>98.2</td>
</tr>
<tr>
<td>$P_4$</td>
<td>23.8</td>
<td>32.9</td>
<td>49.8</td>
<td>65.9</td>
<td>87.3</td>
</tr>
</tbody>
</table>

We considered 4 separable CKBs with the same TBox $T$, but different programs $P$. The programs are given in the extended version. Each program captures the potential information need of a tourist searching for a hotel. Programs $P_1$–$P_4$ ask for a reachable Hotel from the main station “Hauptbahnhof”. Additionally $P_1$–$P_3$ ask for Hotels that are next to some LocRestaurant (a concept inferred from the ontology). $P_4$ asks for Hotels that are in a quiet neighbourhood, by negating the computed relation LoudNeighbourhood. Note that $P_1$ requires that the station close to the Hotel should be reachable without line changes, while $P_2$ allows for at most one line change, whereas $P_3$–$P_4$ allow for any number of changes as long as a station is reachable (achieved via recursion).

For each of the mentioned programs, we included the datasets of different sizes shown in Table 1, which have up to roughly a million facts. Our approach behaved well, as can be seen from the running times shown in Table 1. The dl-program encoding for dlvhex did not scale for any of the example programs provided, and failed to return answers because of memory exhaustion even for the smallest dataset shown in Table 1. We tried to test it against a smaller yet useful set of facts with approx 13000 Datalog facts, and it still reached the time out of 600s that was set.

7 Discussion

We have presented CKBs, a powerful generalization of r-hybrid and $\mathcal{DL}+\mathcal{LOG}$ KBs due to Rosati. In addition to decidability and complexity results for CKBs, we have provided an implementation for a rich fragment of CKBs. The implementation is based on a reduction to reasoning in plain ASP. Our experiments show that this is a promising approach that provides a dramatic improvement over a naive implementation based on a translation into dl-programs.

As shown in Example 1, the ability to use CWA predicates in the theory of a CKB adds significant power. This power is not readily available even in hybrid MKNF, a very rich formalism that captures r-hybrid and $\mathcal{DL}+\mathcal{LOG}$ KBs [19]. Roughly speaking, to capture CKBs we would need to extend hybrid MKNF to support modal operator $K$ inside FO theories. Another way to see a difference is using data complexity. Due to results on DLs with DBoxes (see [10]), satisfiability is already NP-hard in data complexity for CKBs based on basic $DL$-Lite TBoxes in combination with non-disjunctive positive rules. The same setting in hybrid MKNF is tractable.

Related Work. There are few other works on implementing reasoning over combinations of DL ontologies and rules. For expressive (non-Horn) DLs that go beyond the DL-Lite and $\mathcal{EL}$ families, dl-programs is the
richest formalism that has been implemented, in particular in the dlvhex suite. The HermiT system supports reasoning in expressive DLs enriched with positive rules under DL-safety [13]. The work in [16] enables query answering services over expressive DLs using a data-independent translation into disjunctive Datalog. For Horn DLs, Heymans et al. showed how dl-programs with external queries over Datalog-re writable DLs can be translated into Datalog with stable negation [14]. Redl recently presented a generalization of this rewriting approach to external atoms in general HEX-programs [20], still its applicability for reasoning with DL ontologies was demonstrated only using the lightweight logic DL-Lite. An implementation of reasoning in hybrid MKNF KBs (with lightweight ontologies) under the Well-Founded Semantics is also available [1, 17]. The work in [24] shows how reasoning about DL concepts, but not general TBoxes, can be implemented in ASP.

References


Appendix

8.1 CKBs and $\mathcal{DL+LOG}$

For r-hybrid KBs based on DLs, Rosati has generalized the DL-safety restriction to weak DL-safety, resulting in the $\mathcal{DL+LOG}$ language [22]. We now generalize safe CKBs to weakly safe CKBs so that the weak DL-safety condition is captured. We will show next that, when sufficiently expressive logics for theory specification are used, there is effectively no difference between weak and ordinary safeness. For this reason, in the paper we concentrate on the conceptually simpler r-hybrid language, instead of $\mathcal{DL+LOG}$.

**Definition 5** (Weak safety). A CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ is weakly safe if the following conditions are satisfied for every rule $\rho \in \mathcal{P}$:

(a) every variable $x$ of $\rho$ must appear in some atom from $\text{body}^+(\rho)$, and

(b) if $x$ appears in $\text{head}(\rho)$, then $x$ also appears in some atom $r(\vec{u}) \in \text{body}^+(\rho)$ with $r \notin \Sigma$. 


One can check that $DL+LOG$ KBs $H = (T, P)$, where $T$ is an $ALCHOI$ TBox, are subsumed by weakly safe CKBs. Indeed, for such an $H$ build the CKB $H' = (P, T, \Sigma)$, where $\Sigma$ is the set of concept and role names that appear in $T$. One can verify that $H'$ is weakly safe, and that the stable models of $H'$ are precisely the NM-models of $H$.

W.l.o.g., we assume that CKBs do not have a rule $\rho$ such that $r(\vec{u}) \in body^{-}(\rho)$ and $r \notin \Sigma$. A CKB that violates this requirement can be transformed in polynomial time into an equivalent CKB that does obey it. Indeed, if a CKB contains a rule $\rho$ as above, one can replace it by the rule $head(\rho) \lor r(\vec{u}) \leftarrow body(\rho) \setminus \{not \ r(\vec{u})\}$. It is not difficult to see that this transformation preserves the stable models of $H$.

We next show that if we consider a sufficiently expressive DL, there is effectively no difference between safety and weak safety. We need the notion of Boolean conjunctive queries (BCQs). A BCQ $Q$ is just a finite set of atoms of the form $R(\vec{u})$, where $R \in S_{\text{cn}} \cup S_{m}$. We say $Q$ is true in an interpretation $I$ if there exists a function $\pi : S_{\text{const}} \cup S_{\text{var}} \rightarrow \Delta^{I}$ such that (i) $\pi(c) = c^{I}$ for all $c \in S_{\text{const}}$, (ii) $\pi(t) \in A^{I}$ for all $A(t) \in Q$, and (iii) $(\pi(t), \pi(t')) \in r^{I}$ for all $r(t, t') \in Q$.

We consider an extension of $ALCHOI$ with BCQs, denoted $ALCHOI^{BCQ}$. In particular, every BCQ $Q$ is considered to be a concept, in addition to the usual rules used to build $ALCHOI$ concepts. The semantics of such concepts in an interpretation $I$ is defined as follows:

$$Q^{I} = \begin{cases} \Delta^{I} & \text{if } Q \text{ is true in } I; \\ \emptyset & \text{otherwise.} \end{cases}$$

**Definition 6.** Assume a weakly safe CKB $H = (T, P, \Sigma)$. By $\text{safe}(H)$ we denote a CKB that can be obtained from $H$ in two steps:

1. Replace each rule $\rho$ in $P$ by the set of all rules $\sigma(\rho)$, where $\sigma$ is a substitution with
   - $\text{dom}(\sigma) = \{x \in S_{\text{var}} \mid x \text{ occurs in } r(\vec{u}) \in body^{-}(\rho), r \notin \Sigma\}$, and
   - $\text{ran}(\sigma) = \{c \in S_{\text{const}} \mid c \text{ occurs in } P\}$.
   
2. For every rule $\rho$ of $P$ perform the following. Let $Q_{\rho} = \{r(\vec{u}) \mid r(\vec{u}) \in body(\rho), r \in \Sigma\}$. Take a fresh concept name $A$, and let $a$ be an arbitrary individual from $P$. Add $A$ to $\Sigma$, add the inclusions $Q_{\rho} \subseteq A$ and $A \sqsubseteq Q_{\rho}$ to $T$, and replace $\rho$ in $P$ by the rule
   $$head(\rho) \leftarrow (body(\rho) \setminus Q_{\rho}) \cup \{A(a)\}$$

It is not difficult to obtain the following:

**Theorem 6.** Assume a weakly safe CKB $H$. Then $\text{safe}(H)$ is a safe CKB such that $H$ has a stable model iff $\text{safe}(H)$ has a stable model.

In other words, a weakly safe CKB $H$ based on an $ALCHOI$ TBox can be effectively translated, while preserving the existence of a stable model, into a safe CKB $H'$ based on an $ALCHOI^{BCQ}$ TBox. We finally give the following remark on the complexity of reasoning in $ALCHOI^{BCQ}$ TBoxes.

**Proposition 2.** Testing consistency of $ALCHOI^{BCQ}$ TBoxes is $2\text{ExpTime}$-complete.

**Proof (sketch).** Assume an $ALCHOI^{BCQ}$ TBox $T$, and let $BCQ(T)$ denote the set of BCQs that appear in $T$. For every $Q \in BCQ(T)$ and every variable $x$ in $Q$, reserve a fresh individual $c_{Q,x}$. For every $Q \in BCQ(T)$, let $Q_{\text{Skolem}}$ be the BCQ that is obtained from $Q$ by replacing every variable $x$ with the individual $c_{Q,x}$, and let $\mathcal{T}_{Q}$ be the TBox that consists of inclusions (i) $\{c\} \sqsubseteq A$ for all $A(c) \in Q_{\text{Skolem}}$, and (ii) $\{c\} \sqsubseteq \exists r\{d\}$ for all $r(c, d) \in Q_{\text{Skolem}}$. A BCQ-valuation for $T$ is any set $V \subseteq BCQ(T)$. We let $T^{V}$ be the TBox that is obtained from $T$ by
- replacing with $\top$ every occurrence of $Q \in V$, and
- replacing with $\bot$ every occurrence of $Q \in BCQ(T) \setminus V$.

It is not difficult to see that $T$ has a model iff there exists $V \subseteq BCQ(T)$ such that $T^V \cup \bigcup_{Q \in V} T_Q$ has a model where each $Q' \in BCQ(T) \setminus V$ is false. Clearly, the later task corresponds to non-entailment of a union of conjunctive queries (UCQ), which is known to be 2ExpTime-complete for $\mathcal{ALCHOL}$. The lower bound can be inferred from this problem as well.

\section{DL-programs}

We briefly recall here the syntax and semantics of dl-programs [6]. Roughly speaking, dl-programs extend plain ASP with \textit{dl-atoms}, which are special atoms that correspond to queries over an external DL KB. For the encoding based on Proposition 1, we need only a relatively small fragment of dl-programs. In particular, here dl-atoms are only allowed to test consistency of DL knowledge bases, and can only use the operators $\uplus$ and $\cup$ to (“temporarily”) update it. More formally, a \textit{dl-atom} $\alpha$ is an expression of the form

$$DL[t \, op_1 \, R_1, \ldots, \, S_n \, op_n \, R_n; \bot](t),$$

where $t$ is a term, $\{op_1, \ldots, \, op_n\} \subseteq \{\uplus, \cup\}$, and each pair $S_i, R_i$ with $1 \leq i \leq n$ is such that (i) $S_i \in S_{cn}$, and $R_i \in S_{pred}$ is unary, or (ii) $S_i \in S_{rn}$, and $R_i \in S_{pred}$ is a binary. The notion of \textit{dl-rules} $\rho$ is defined exactly as the notion of ordinary rules, except that here dl-atoms can occur in the place of non-negated ordinary atoms. Each dl-rule $\rho$ must satisfy the next condition: every variable of $\rho$ must appear in a non-negated ordinary atom in the body of $\rho$. A \textit{dl-program} is a pair $\Pi = (T, \mathcal{P})$ with $T$ an $\mathcal{ALCHOL}$ TBox, and $\mathcal{P}$ a set of dl-rules. Here concept and role names that occur in $T$ are allowed to occur in $\mathcal{P}$ only in dl-atoms. We let $\text{ground}(\mathcal{P})$ be the set of ground dl-rules that can be obtained from the rules in $\mathcal{P}$ by replacing variables with constants from $\mathcal{P}$.

When building TBoxes next, we use $P(t)$, $P(t, v)$, $\neg P(t)$ and $\neg P(t, v)$ as abbreviations for inclusion axioms $\{t\} \subseteq P$, $\{t\} \subseteq \exists P \{v\}$, $\{t\} \subseteq \neg P$, and $\{t\} \subseteq \forall P \neg \{v\}$, respectively. For a TBox $T$, an Herbrand interpretation $I$, and a ground dl-atom $\alpha$ of form (2), we write $I \models_T \alpha$ if the TBox $T \cup T_1 \cup \cdots \cup T_n$ is inconsistent, where each $T_i$, $1 \leq i \leq n$, is defined as follows:

- $T_i = \{S_i(t) \mid R_i(t) \in I\}$ if $op_i = \uplus$ and $S_i \in S_{cn}$,
- $T_i = \{S_i(t, v) \mid R_i(t, v) \in I\}$ if $op_i = \uplus$ and $S_i \in S_{rn}$,
- $T_i = \{\neg S_i(t) \mid R_i(t) \in I\}$ if $op_i = \cup$ and $S_i \in S_{cn}$, and
- $T_i = \{\neg S_i(t, v) \mid R_i(t, v) \in I\}$ if $op_i = \cup$ and $S_i \in S_{rn}$.

Assume a dl-program $\Pi = (T, \mathcal{P})$ and an Herbrand interpretation $I$. We let $\Pi^I$ be the (plain) ground positive program that is obtained from $\text{ground}(\mathcal{P})$ by

- deleting every rule with a dl-atom $L$ such that $I \not\models_T L$,
- deleting every rule with a literal $\text{not} \, R(\overline{u})$ with $R(\overline{u}) \in I$,
- deleting all dl-atoms, and all negative literals in the remaining rules.

If $I$ is a minimal model of $\Pi^I$, then $I$ is called a \textit{(weak) answer set} of $\Pi$. 

\[
\begin{align*}
\top^I &= \Delta^I & (C \cap D)^I &= C^I \cap D^I & (\exists R. C)^I &= \{ e \in \Delta^I \mid \exists e' \in \Delta^I : (e, e') \in R^I \land e' \in C^I \} \\
\bot^I &= \emptyset & (C \cup D)^I &= C^I \cup D^I & (\forall R. C)^I &= \{ e \in \Delta^I \mid \forall e' \in \Delta^I : (e, e') \not\in R^I \lor e' \in C^I \} \\
(\neg C)^I &= \Delta^I \setminus C^I & \{ e \}^I &= \{ c \} & (R^-)^I &= \{ (e, e') \mid (e', e) \in R^I \}
\end{align*}
\]

Table 2: Extension of the valuation function for \( \mathcal{ALCHOL} \)
\( P_1 \): Find hotels that are close to stations reachable from Hauptbahnhof (main Station) with no metro line changes, and have local restaurants close by.

\[
\text{reachableWithNoChanges}(X, Y) \leftarrow \text{locatedAlong}(X, Z), \text{locatedAlong}(Y, Z), \quad X = \text{“Hauptbahnhof”}
\]

\[ q_1(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \text{reachableWithNoChanges}(Z, Y), \text{next}(X, V), \text{LocRestaurant}(V) \]

\( P_2 \): Find hotels that are close to stations reachable from Hauptbahnhof (main Station) with up to one metro line change, and have local restaurants close by.

\[
\text{reachableWithNoChanges}(X, Y) \leftarrow \text{locatedAlong}(X, Z), \text{locatedAlong}(Y, Z)
\]

\[
\text{reachableWithOneChange}(X, Z) \leftarrow \text{reachableWithNoChanges}(X, Y), \text{locatedAlong}(X, V), \quad \text{reachableWithNoChanges}(Y, Z), \text{locatedAlong}(Z, W), \quad X = \text{“Hauptbahnhof”}
\]

\[ q_2(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \text{reachableWithOneChange}(Z, Y), \text{next}(X, V), \text{LocRestaurant}(V) \]

\( P_3 \): Find hotels that are close to stations reachable from Hauptbahnhof (main Station) using metro lines and have local restaurants close by.

\[
\text{reachable}(X, Y) \leftarrow \text{nextStation}(X, Y), \quad X = \text{“Hauptbahnhof”}
\]

\[
\text{reachable}(X, Z) \leftarrow \text{nextStation}(X, Y), \text{reachable}(Y, Z)
\]

\[ q_3(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \text{reachable}(Z, Y), \text{next}(X, V), \text{LocRestaurant}(V) \]

\( P_4 \): Find hotels that are close to stations reachable from Hauptbahnhof (main Station) and are in a quiet neighbourhood.

\[
\text{reachable}(X, Y) \leftarrow \text{nextStation}(X, Y), \quad X = \text{“Hauptbahnhof”}
\]

\[
\text{reachable}(X, Z) \leftarrow \text{nextStation}(X, Y), \text{reachable}(Y, Z)
\]

\[
\text{LoudNeighbourhood}(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \quad \text{reachable}(Z, Y), \text{next}(X, V), \quad \text{Club}(V)
\]

\[
\text{LoudNeighbourhood}(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \quad \text{reachable}(Z, Y), \text{next}(X, V), \quad \text{Bar}(V)
\]

\[ q_4(X) \leftarrow \text{Hotel}(X), \text{next}(X, Y), \text{reachable}(Z, Y), \quad \text{not} \ \text{LoudNeighbourhood}(X) \]

Figure 2: Programs evaluated against both encodings. Note that the relation nextStation is symmetrical