The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Combining Rules and Ontologies into Clopen Knowledge Bases*

Labinot Bajraktari, Magdalena Ortiz, Mantas Šimkus
Institute of Information Systems, TU Wien, Austria
lbajrakt@kr.tuwien.ac.at ortiz@kr.tuwien.ac.at simkus@dbai.tuwien.ac.at

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 $\mathcal{DL}$+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 $\mathcal{ALC}$ 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 $\mathcal{DL}$+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 $\mathcal{ALCH}$, 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*.

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., (Rosati 2005; 2006; Eiter et al. 2008; Motik and Rosati 2010; Knorr, Alferes, and Hitzler 2011)). 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 (Rosati 2005), and $\mathcal{DL}$+LOG (Rosati 2006). Each CKB is a triple $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$, where $\mathcal{P}$ is a disjunctive Datalog program with "*not*" literals in rule bodies ($\mathrm{Datalog}^{\neg,\vee}$), φ is a theory (e.g., in first-order logic), and Σ is a set of predicate symbols. Intuitively, Σ 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 $\mathcal{DL}$+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 $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$, where φ is expressed in the *guarded negation fragment of FO (GNFO)* (Bárány, Cate, and Segoufin 2015). This is a very expressive fragment that subsumes the more prominent *guarded fragment* of FO, as well as many expressive DLs. We give a $\textsc{NExpTime}^{2\textsc{ExpTime}}$ 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 $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ where φ is expressed in the very expressive DL $\mathcal{ALCHOI}$, which extends the basic DL $\mathcal{ALC}$ 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 $\mathcal{ALCHOI}$.
- We study the relationship between CKBs and other existing hybrid languages. We define a restricted class of

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

separable CKBs, and show that they can be transformed in polynomial time into the so-called *dl-programs* (Eiter et al. 2008). 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 (Redl 2017)) 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 $\mathcal{ALCH}$ 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

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

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

Atoms and Herbrand interpretations. We assume a countably infinite set $\mathbf{S}_{\mathsf{var}}$ of *variables*. The elements of $\mathbf{S}_{\mathsf{const}} \cup \mathbf{S}_{\mathsf{var}}$ are called *terms*. An *atom* is an expression of the form $r(t_1, \ldots, t_n)$, where $r \in \mathbf{S}_{\mathsf{pred}}$, n is the arity of r, and $t_1, \ldots, t_n$ are terms. An atom is called *ground* if no variables occur in it. An *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) $\Delta^{\tilde{I}} = \mathbf{S}_{\mathsf{const}}$, and (ii) $r^{\tilde{I}} = \{\vec{u} \mid r(\vec{u}) = I\}$ for all $r \in \mathbf{S}_{\mathsf{pred}}$.

Clopen Knowledge Bases

We now formally define our new hybrid language.

Syntax. A *rule* ρ is an expression of the form

$$p_1 \vee \ldots \vee p_k \leftarrow p_{k+1}, \ldots, p_l, not\ p_{l+1}, \ldots, not\ p_m \quad (1)$$

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

A *program* $\mathcal{P}$ is a set of rules. A *Clopen Knowledge Base (CKB)* is a triple $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$, where $\mathcal{P}$ is a program, $\varphi \in \mathbf{T}$ is a theory, and $\Sigma \subseteq \mathbf{S}_{\mathsf{pred}}$. The predicate symbols in Σ (resp., in $\mathbf{S}_{\mathsf{pred}} \setminus \Sigma$) 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 ρ appears in some atom $r(\vec{u}) \in body^+(\rho)$ 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 $r(\vec{u}) \leftarrow$ is called a *fact*. We write $r(\vec{u}) \in \mathcal{P}$ in case the fact $r(\vec{u}) \leftarrow$ is present in the program $\mathcal{P}$.

As usual, $\mathsf{dom}(f)$ and $\mathsf{ran}(f)$ denote the *domain* and *range* of a function f, respectively. A *substitution* σ is any partial function from $\mathbf{S}_{\mathsf{var}}$ to $\mathbf{S}_{\mathsf{const}}$. For a rule ρ and a substitution σ, we use $\sigma(\rho)$ to denote the rule that is obtained from ρ by replacing every variable $X \in \mathsf{dom}(\sigma)$ with $\sigma(X)$. The *grounding* of a program $\mathcal{P}$, denoted $ground(\mathcal{P})$, is the ground program that consists of all ground rules ρ' such that $\rho' = \sigma(\rho)$ for some $\rho \in \mathcal{P}$ and some substitution σ. Note that $ground(\mathcal{P})$ 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 $body^+(\rho) \subseteq I$ implies $head(\rho) \cap I \neq \emptyset$ for all $\rho \in \mathcal{P}$. Furthermore, I is a *minimal model* of the program $\mathcal{P}$ if, in addition, there is no $J \subsetneq I$ such that J is a model of $\mathcal{P}$.

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

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

(2) In the remaining rules, delete all negated atoms, and all ordinary atoms $r(\vec{u})$ 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:

- $\{r(\vec{u}) \mid r(\vec{u}) \in I, r \notin \Sigma\}$ is a minimal model of $\mathcal{P}^{I,\Sigma}$, and
- $\tilde{I}$ is model of φ.

Reasoning problems. As usual in hybrid languages (see, e.g., (Rosati 2005)), 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 $R(\vec{u})$, the problem is to decide whether $R(\vec{u}) \in I$ 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 $\mathcal{H}' = (\mathcal{P} \cup \{\leftarrow R(\vec{u})\}, \varphi, \Sigma)$. 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 form 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* $\mathsf{RouteTable}(\ell, s, s') \leftarrow$ *store that on the line ℓ, station s is followed by station s'. The constants t_1 and t_2 represent tram lines, while ℓ_1 represents a metro line; we have corresponding facts* $\mathsf{MetroLine}(\ell_1)$, $\mathsf{TramLine}(t_1)$, $\mathsf{TramLine}(t_2)$. *$\mathcal{P}_2$ contains facts related to locations, including the following (for convenience,* $\mathsf{CloseTo}$ *is depicted with dotted lines).*

$$\begin{array}{lll} \mathsf{CloseTo}(c_1, s_1) \leftarrow & \mathsf{Hotel}(h_1) \leftarrow & \mathsf{TramConn}(h_1) \leftarrow \\ \mathsf{CloseTo}(h_2, s_4) \leftarrow & \mathsf{Hotel}(h_2) \leftarrow & \end{array}$$

The (self-explanatory) rules in $\mathcal{P}_3$ and the theory φ are shown in Figure 1 ($\mathsf{URailConn}$ *stands for urban rail connection). If h is a hotel with direct connection to the point of interest c_1, then $\mathsf{Q}(h)$ holds for it. 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 $\mathcal{P}_3$ are closed. The remaining ones are open, i.e. $\Sigma = \{$*Hotel, $\mathsf{CloseTo}$, $\mathsf{Station}$, $\mathsf{TramConn}$, $\mathsf{MetroConn}$, $\mathsf{URailConn}\}$.

In the spirit of r-hybrid and $\mathcal{DL}$+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* $\Sigma = \{\mathsf{Edge}\}$ *and*

$$\begin{aligned} \varphi &= \{\forall xy\, \mathsf{Edge}(x,y) \to (\mathsf{Node}(x) \wedge \mathsf{Node}(y))\} \\ \mathcal{P} &= \mathsf{Node}(v_1) \leftarrow; \ldots \mathsf{Node}(v_n) \leftarrow; \\ &\quad \mathsf{Reach}(X,X) \leftarrow \mathsf{Node}(X); \\ &\quad \mathsf{Reach}(X,Z) \leftarrow \mathsf{Reach}(X,Y), \mathsf{Edge}(Y,Z), \mathsf{Node}(Z); \} \end{aligned}$$

Then these CKBs are not equivalent:

$$\begin{aligned} \mathcal{H}_1 &= (\mathcal{P}, \varphi \wedge \mathsf{Reach}(v_1, v_2), \Sigma) \\ \mathcal{H}_2 &= (\mathcal{P} \cup \{\mathsf{Reach}(v_1, v_2) \leftarrow \}, \varphi, \Sigma) \end{aligned}$$

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 $\mathcal{P}$ and an Herbrand interpretation I. We call I a *stable model* of $\mathcal{P}$ if I is a stable model of the CKB $\mathcal{H} = (\mathcal{P}, \top, \emptyset)$. 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 (Gelfond and Lifschitz 1988). Observe that in a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \emptyset)$, the theory φ 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 $\tilde{I} \in mods(\varphi)$.

Relationship to r-hybrid KBs. Our CKBs are a close relative of the r-hybrid KBs of Rosati (Rosati 2005). 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 φ is a theory in FO and $\mathcal{P}$ is a $\mathrm{Datalog}^{\neg,\vee}$ program, corresponds to the CKB $\mathcal{H}' = (\mathcal{P}, \varphi, \Sigma)$, where Σ is the set of predicates symbols appearing in φ. 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 Σ need not contain all the predicate symbols that appear in φ. That is, closed predicates may occur in φ, and the extensions of these predicates in (the relevant) models of φ 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 $\mathcal{DL}$+LOG language is obtained from r-hybrid KBs by allowing only DLs for specifying theories, and relaxing the safeness condition to *weak safeness* (Rosati 2006). In the extended version we show that, when sufficiently rich DLs are considered, CKBs also generalize $\mathcal{DL}$+LOG.

Active domain predicate. For convenience, we assume the availability of a unary "built-in" predicate $adom$ 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 $r \neq adom$ that appears in $\mathcal{P}$, we assume that (i) $\mathcal{P}$ contains the rule $adom(X_j) \leftarrow r(X_1, \ldots, X_n)$ for every $1 \leq j \leq n$, and (ii) $adom$ is allowed to occur only in bodies of the remaining rules.

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 φ to belong to a logic $\mathcal{L}$

$$
\begin{aligned}
\mathcal{P}_3 = \{\ \mathsf{MetroStation}(Y_1) &\leftarrow \mathsf{RouteTable}(X,Y_1,Y_2), \mathsf{MetroLine}(X)\\
\mathsf{TramStation}(Y_2) &\leftarrow \mathsf{RouteTable}(X,Y_1,Y_2), \mathsf{TramLine}(X)\\
\mathsf{ReachOnLine}(X,Y_1,Y_2) &\leftarrow \mathsf{RouteTable}(X,Y_1,Y_2)\\
\mathsf{ReachOnLine}(X,Y_1,Y_3) &\leftarrow \mathsf{ReachOnLine}(X,Y_1,Y_2), \mathsf{RouteTable}(X,Y_2,Y_3)\\
\mathsf{TramOnly}(X) &\leftarrow \mathsf{TramConn}(X), \mathit{not}\ \mathsf{MetroConn}(X)\\
\mathsf{Q}(X) &\leftarrow \mathsf{Hotel}(X), \mathsf{CloseTo}(X,Y), \mathsf{ReachOnLine}(Z,Y,Y'), \mathsf{CloseTo}(c_1,Y')\\
\mathsf{Q}'(X) &\leftarrow \mathsf{Q}(X), \mathit{not}\ \mathsf{TramOnly}(X)\quad \}
\end{aligned}
$$

$$
\begin{aligned}
\varphi = \{\ &\forall x.\big(\mathsf{MetroStation}(x) \vee \mathsf{TramStation}(x) \leftrightarrow \mathsf{Station}(x)\big),\\
&\forall x.\big(\mathsf{TramConn}(x) \leftrightarrow \exists y\, \mathsf{CloseTo}(x,y) \wedge \mathsf{TramStation}(y)\big),\\
&\forall x.\big(\mathsf{MetroConn}(x) \leftrightarrow \exists y\, \mathsf{CloseTo}(x,y) \wedge \mathsf{MetroStation}(y)\big),\\
&\forall x.\big(\mathsf{URailConn}(x) \leftrightarrow \exists y\, \mathsf{CloseTo}(x,y) \wedge \mathsf{Station}(y)\big)\quad \}
\end{aligned}
$$

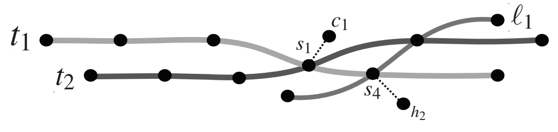

Figure 1: Example CKB

in which satisfiability is decidable (i.e., the set $\{\varphi \in \mathcal{L} \mid mods(\varphi) \neq \emptyset\}$ should be recursive). However, this alone is not enough, since we will in general be interested in models of φ where a selected set of 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 $\overline{r}$ be a fresh n-ary relation symbol that does not appear in $\mathcal{H}$. We let $Choose(\mathcal{H})$ be the set that contains*

$$r(Y_1,\dots,Y_n) \vee \overline{r}(Y_1,\dots,Y_n) \leftarrow adom(Y_1),\dots,adom(Y_n)$$

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

A stable model I of $\mathcal{P} \cup Choose(\mathcal{H})$ 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 $\mathcal{P} \cup Choose(\mathcal{H})$ has some stable model I for which there exists some $\mathcal{I} \in mods(\varphi)$ with the following properties:*
(C1) *$(c_1^{\mathcal{I}},\dots,c_n^{\mathcal{I}}) \in r^{\mathcal{I}}$ for all $r(c_1,\dots,c_n) \in I$,*
(C2) *$(c_1^{\mathcal{I}},\dots,c_n^{\mathcal{I}}) \notin r^{\mathcal{I}}$ for all $\overline{r}(c_1,\dots,c_n) \in I$, and*
(C3) *if $(e_1,\dots,e_n) \in r^{\mathcal{I}}$ and $r \notin \Sigma$, then there exists $r(c_1,\dots,c_n) \in I$ with $c_1^{\mathcal{I}} = e_1,\dots,c_n^{\mathcal{I}} = e_n$.*

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 $\mathcal{P} \cup Choose(\mathcal{H})$ and test, for each of them, the existence of a model $\mathcal{I}$ of the theory φ 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 (Bárány, Cate, and Segoufin 2015).

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

$$\varphi ::= r(v_1,\dots,v_n) \mid v = u \mid \exists x\, \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \alpha \wedge \neg\varphi[\vec{x}],$$

where $u, v, v_1, \dots, v_n$ are terms, and α is an atom or an equality statement such that all variables of $\vec{x}$ also occur in α. 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 φ in Example 1 is in GNFO.

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

Proof. Assume a CKB $\mathcal{H} = (\mathcal{P}, \varphi, \Sigma)$ with φ in GNFO. Let Σ_c be the set of predicates that occur in $\mathcal{P}$ but not in Σ. For every n-ary predicate symbol $r \in \Sigma_c$, assume a tuple $\vec{x}_r = (x_r^1,\dots,x_r^n)$ of variables. Assume a stable model I of $\mathcal{P} \cup Choose(\mathcal{H})$. For such I, let $\psi(I)$ be the following formula:

$$\psi(I) = \bigwedge_{r(\vec{c}) \in I} r(\vec{c}) \wedge \bigwedge_{\overline{r}(\vec{c}) \in I} \neg r(\vec{c}) \wedge \bigwedge_{r \in \Sigma_c} \forall x_r^1 \dots \forall x_r^n \Bigg(r(x_r^1,\dots,x_r^n) \rightarrow \bigvee_{r(c_1,\dots,c_n) \in I} \big(\bigwedge_{1 \leq i \leq n} (x_r^i = c_i) \big) \Bigg)$$

One can check that the formula $\varphi \wedge \psi(I)$ 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 (Benedikt et al. 2016) for reasoning about *visible* and *invisible* tables in databases. The following holds: $\psi(I)$ is satisfiable iff there exists $\mathcal{I} \in mods(\varphi)$ that satisfies the conditions (C1-C3) of Proposition 1. Overall, this means that $\mathcal{H}$ has a stable model iff $\mathcal{P} \cup Choose(\mathcal{H})$ has some stable model I such that $\varphi \wedge \psi(I)$ is satisfiable. Keeping in mind that satisfiability in GNFO is 2ExpTime-complete, this equivalence yields the $\text{NExpTime}^{\text{2ExpTime}}$ 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 $\mathcal{P} \cup Choose(\mathcal{H})$, whose size is at most exponential in the size of $\mathcal{H}$, and then checking that (i) I is a minimal model of $\mathcal{P}^{I,\emptyset}$, and (ii) that the formula $\psi(I)$ is satisfiable. The lower bound is carried over trivially from GNFO. □

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 $\mathbf{S}_{\mathsf{cn}} \subseteq \mathbf{S}_{\mathsf{pred}}$ of unary relation symbols, called *concept names*, and a countably infinite set $\mathbf{S}_{\mathsf{rn}} \subseteq \mathbf{S}_{\mathsf{pred}}$ of binary relation symbols, called *role names*. If $R \in \mathbf{S}_{\mathsf{rn}}$, then R and R^- are *roles*. (Complex) *concepts* are defined as follows: (a) the symbols $\top, \bot$, and every concept name $A \in \mathbf{S}_{\mathsf{cn}}$ is a concept, (b) if $a \in \mathbf{S}_{\mathsf{const}}$, 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 $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$, and observe that $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ for all concept names A, and $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ 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 $\cdot^{\mathcal{I}}$ in the usual way (see (Baader et al. 2003); 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 $\mathcal{I}$ is a *model* of a TBox $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for each concept inclusion $C \sqsubseteq D \in \mathcal{T}$, and $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$ 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 (Baader et al. 2003).

Example 3. *The theory φ 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 $C \sqsubseteq D$, $D \sqsubseteq C$);*

$$\begin{aligned}\mathcal{T} = \{ \quad & \mathsf{MetroStation} \sqcup \mathsf{TramStation} \equiv \mathsf{Station},\\ & \mathsf{TramConn} \equiv \exists \mathsf{CloseTo}.\mathsf{TramStation},\\ & \mathsf{MetroConn} \equiv \exists \mathsf{CloseTo}.\mathsf{MetroStation},\\ & \mathsf{URailConn} \equiv \exists \mathsf{CloseTo}.\mathsf{Station} \quad \}\end{aligned}$$

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 (Franconi, Ibáñez-García, and Seylan 2011) (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* NExpTime*-complete. The problem is* 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.*

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 α 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 $R \sqsubseteq^*_{\mathcal{T}} S$ occurs positively in $\exists R.C$, for any concept C. (C) Every role name S with $R \sqsubseteq^*_{\mathcal{T}} S$ 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 α occurs positively (resp., negatively) in C, then α occurs negatively (resp., positively) in $\neg C$.

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

Example 4. *Take the CKB $\mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma)$ with $\mathcal{P} = \{Q(X, Y, Z) \leftarrow T(X, Y), P(Y, Z)\}$, $\mathcal{T} = \{\exists R.(\exists P.A) \sqsubseteq B\}$, 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 $\mathcal{ALCHOI}$ 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 $\mathcal{ALCHOI}$ TBox. Such KBs, as mentioned, correspond to CKBs $\mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma)$, where Σ 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 $\mathcal{DL}$+LOG), because the variable Z does not appear in a rule atom with a predicate symbol that does not occur in $\mathcal{T}$.

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 (Eiter et al. 2008) 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 Σ. Then the set $\mathcal{P}'$ of dl-rules is defined as follows:

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

where $\lambda = S_1' \uplus S_1, \ldots, S_n' \uplus S_n, S_1' \uplus \overline{S}_1, \ldots, S_n' \uplus \overline{S}_n$.

Intuitively, given a stable model I of $\mathcal{P} \cup \mathit{Choose}(\mathcal{H})$, the expression λ above allows us to check the conditions (C1) and (C2) of Proposition 1 (see the construction of $\mathsf{TBox}(\mathcal{H}, I)$ as used in the proof of Theorem 2 in the extended version). The constraint $\leftarrow DL[\lambda; \bot](X)$ 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.*

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 $\mathcal{I} \in \mathit{mods}(\varphi)$ 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 (Hustadt, Motik, and Sattler 2007; Eiter, Ortiz, and Šimkus 2012; Bienvenu et al. 2014). 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 $\mathcal{ALCH}$ (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:

$$\begin{array}{cccc} A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B & A \sqsubseteq B_1 \sqcup \ldots \sqcup B_m & A \sqsubseteq \exists R.B & \text{(I)} \\ \exists R.A \sqsubseteq B & A \sqsubseteq \forall R.B & R \sqsubseteq S & \text{(II)} \end{array}$$

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., (Simančík, Kazakov, and Horrocks 2011)).

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

$$\begin{array}{rll} S(X,Y) \leftarrow R(X,Y) & \textit{for each} & R \sqsubseteq S \in \mathcal{T} \\ B(X) \leftarrow r(X,Y), A(Y) & \textit{for each} & \exists R.A \sqsubseteq B \in \mathcal{T} \\ B(Y) \leftarrow A(X), r(X,Y) & \textit{for each} & A \sqsubseteq \forall R.B \in \mathcal{T} \end{array}$$

The program $\mathit{Comm}(\mathcal{H})$ 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 $\tau \subseteq \mathbf{S}_{\mathsf{cn}} \cup \{\neg A \mid A \in \mathbf{S}_{\mathsf{cn}}\}$. A type τ is* consistent *w.r.t. a TBox $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ and an element $e \in \Delta^\mathcal{I}$ such that $e \in (\bigsqcap_{C \in \tau} C)^\mathcal{I}$. We use $\mathit{types}(\mathcal{T})$ to denote the set of types τ such that (i) τ is consistent w.r.t. $\mathcal{T}$, and (ii) $A \in \tau$ or $\neg A \in \tau$ for each concept name A in $\mathcal{T}$.*

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

(i) the rule $\bigvee_{\tau \in \mathit{types}(\mathcal{T})} \mathsf{Type}_\tau(X) \leftarrow \mathit{adom}(X)$

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

$$\begin{array}{rl} A(X) \leftarrow \mathsf{Type}_\tau(X) & \text{for each } A \in \tau \cap \mathbf{S}_{\mathsf{cn}} \\ \leftarrow \mathsf{Type}_\tau(X), A(X) & \text{for each } \neg A \in \tau \end{array}$$

The program $\mathsf{ASP}(\mathcal{H})$ 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 $\mathcal{P}$ depend only on $\mathcal{T}$ and Σ, and thus the translation is data-independent. Note that the set $\mathit{types}(\mathcal{T})$ can be computed in single exponential time in the size of $\mathcal{T}$, and for this a standard DL reasoner can be used. Indeed, a type τ is consistent w.r.t. $\mathcal{T}$ iff $\mathcal{T} \cup \{A(c) \mid A \in \tau\} \cup \{\neg A(c) \mid \neg A \in \tau\}$ has a model, for a fresh constant c.

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

Data-dependent translation. Since $|\mathit{types}(\mathcal{T})|$ is often exponential in the size of $\mathcal{T}$, the program $\mathsf{ASP}(\mathcal{H})$ 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 $\mathcal{H} = (\mathcal{P}, \mathcal{T}, \Sigma)$. For every constant c that appears in $\mathcal{H}$, let $\mathfrak{t}(c, \mathcal{H})$ be the set of types returned by the *non-failing* runs of the following non-deterministic procedure:

(1) Let $\tau = \{A \mid \mathcal{P}$ has the fact $A(c) \leftarrow\}$.

(2) Close τ under the following inference rules:

  (a) If $A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B \in \mathcal{T}$ and $\{A_1, \ldots, A_n\} \subseteq \tau$, then add B to τ.

  (b) If $\exists S.\top \sqsubseteq B \in \mathcal{T}$, $R \sqsubseteq^*_\mathcal{T} S$, and $\mathcal{P}$ has the fact $R(c,d) \leftarrow$ for some d, then add B to τ.

  (c) If $\top \sqsubseteq \forall S.B \in \mathcal{T}$, $R \sqsubseteq^*_\mathcal{T} S$, and $\mathcal{P}$ has the fact $R(d,c) \leftarrow$ for some d, then add B to τ.

  If τ is inconsistent w.r.t. $\mathcal{T}$, then return *failure*.

(3) 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 $\mathcal{P}$,

(b) in some $\exists R.A \sqsubseteq B \in \mathcal{T}$ or $A \sqsubseteq \forall R.B \in \mathcal{T}$ such that R appears in a non-fact rule of $\mathcal{P}$,

(c) in some $\exists S.A \sqsubseteq B \in \mathcal{T}$ such that $\mathcal{P}$ has the fact $R(c, d) \leftarrow$ for some d, and $R \sqsubseteq^*_{\mathcal{T}} S$, or

(d) in some $A \sqsubseteq \forall R.B \in \mathcal{T}$ such that $\mathcal{P}$ has the fact $R(d, c) \leftarrow$ for some d, and $R \sqsubseteq^*_{\mathcal{T}} S$.

If the above B does not exist, then return τ. Otherwise, non-deterministically add to τ either B or $\neg B$, and go to step (2).

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

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

(ii) for each constant c of $\mathcal{H}$, the rule $\bigvee_{\tau \in \mathfrak{t}(c,\mathcal{H})} \mathsf{Type}_\tau(c) \leftarrow$

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

$$\begin{array}{rll} A(c) \leftarrow & \mathsf{Type}_\tau(c) & \text{for each } A \in \tau \cap \mathbf{S}_{\mathsf{cn}} \\ \leftarrow & \mathsf{Type}_\tau(c), A(c) & \text{for each } \neg A \in \tau \end{array}$$

The translation allows us to decide stable model existence:

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

Implementation and Experiments

We present here some experiments that demonstrate the advantages of translating a separable CKB $\mathcal{H}$ into a plain program $\mathsf{ASP}^{\mathsf{dd}}(\mathcal{H})$. We have implemented our approach in a prototype reasoner. In particular, to build the function $\mathfrak{t}$ 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 (Horridge and Bechhofer 2011) to manage ontologies. The ASP program resulting from the translation is evaluated with Clingo 4.2.1 (Gebser et al. 2011). 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[1] data, transformed into Datalog facts following (Eiter et al. 2015). The data, describing the city of Vienna, is available as database dumps at BBBike[2]. 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 $\mathcal{T}$ 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 (Eiter, Krennwallner, and Schneider 2013), enriched with $\mathcal{ALCH}$ axioms.

	next50	*next100*	*next150*	*next200*	*next250*
Fact count	145014	263075	479283	743935	1053335
$\mathcal{P}_1$	19.6	30.1	44.6	60.2	87.6
$\mathcal{P}_2$	19.6	31.8	52.7	64.0	95.4
$\mathcal{P}_3$	19.6	32.8	56.1	64.7	98.2
$\mathcal{P}_4$	23.8	32.9	49.8	65.9	87.3

Table 1: Number of facts for different *next* relations, and running times in seconds for $\mathcal{P}_1$–$\mathcal{P}_4$

We considered 4 separable CKBs with the same TBox $\mathcal{T}$, but different programs $\mathcal{P}$. The programs are given in the extended version. Each program captures the potential information need of a tourist searching for a hotel. Programs $\mathcal{P}_1$–$\mathcal{P}_4$ ask for a reachable Hotel from the main station "*Hauptbahnhof*". Additionally $\mathcal{P}_1$–$\mathcal{P}_3$ ask for Hotels that are next to some LocRestaurant (a concept inferred from the ontology). $\mathcal{P}_4$ asks for Hotels that are in a quiet neighbourhood, by negating the computed relation LoudNeighbourhood. Note that $\mathcal{P}_1$ requires that the station close to the Hotel should be reachable without line changes, while $\mathcal{P}_2$ allows for at most one line change, whereas $\mathcal{P}_3$–$\mathcal{P}_4$ 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.

Discussion

We have presented CKBs, a powerful generalization of r-hybrid and $\mathcal{DL}$+LOG KBs due to Rosati. In addition to de-

[1]https://www.openstreetmap.org

[2]http://download.bbbike.org/osm/bbbike/Wien/

cidability 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 $\mathcal{DL}$+LOG KBs (Motik and Rosati 2010). Roughly speaking, to capture CKBs we would need to extend hybrid MKNF to support modal operator $\mathbf{K}$ inside FO theories. Another way to see a difference is using *data complexity*. Due to results on DLs with DBoxes (see (Franconi, Ibáñez-García, and Seylan 2011)), 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 (Glimm et al. 2014). The work in (Hustadt, Motik, and Sattler 2007) 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-rewritable DLs can be translated into Datalog with stable negation (Heymans, Eiter, and Xiao 2010). Redl recently presented a generalization of this rewriting approach to external atoms in general HEX-programs (Redl 2017), 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 (Alferes, Knorr, and Swift 2013; Ivanov, Knorr, and Leite 2013). The work in (Swift 2004) shows how reasoning about DL concepts, but not general TBoxes, can be implemented in ASP.

References

Alferes, J. J.; Knorr, M.; and Swift, T. 2013. Query-driven procedures for hybrid MKNF knowledge bases. *ACM Trans. Comput. Logic* 14(2):16:1–16:43.

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. New York, NY, USA: Cambridge University Press.

Bárány, V.; Cate, B. T.; and Segoufin, L. 2015. Guarded negation. *J. ACM* 62(3):22:1–22:26.

Benedikt, M.; Bourhis, P.; ten Cate, B.; and Puppis, G. 2016. Querying visible and invisible information. In *Proc. of LICS 2016*, 297–306. ACM.

Bienvenu, M.; ten Cate, B.; Lutz, C.; and Wolter, F. 2014. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.* 39(4):33:1–33:44.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *Artif. Intell.* 172(12-13):p. 1495.

Eiter, T.; Pan, J. Z.; Schneider, P.; Šimkus, M.; and Xiao, G. 2015. A rule-based framework for creating instance data from OpenStreetMap. In *Proc. of RR 2015*. Springer.

Eiter, T.; Krennwallner, T.; and Schneider, P. 2013. Lightweight spatial conjunctive query answering using keywords. In *Proc. of ESWC 2013*. Springer.

Eiter, T.; Ortiz, M.; and Šimkus, M. 2012. Conjunctive query answering in the description logic SH using knots. *J. Comput. Syst. Sci.* 78(1):47–85.

Franconi, E.; Ibáñez-García, Y. A.; and Seylan, I. 2011. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.* 278:71–84.

Gebser, M.; Kaufmann, B.; Kaminski, R.; Ostrowski, M.; Schaub, T.; and Schneider, M. 2011. Potassco: The potsdam answer set solving collection. *AI Comm.* 24(2):107–124.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP/SLP 1988*, 1070–1080. MIT Press.

Glimm, B.; Horrocks, I.; Motik, B.; Stoilos, G.; and Wang, Z. 2014. Hermit: An OWL 2 reasoner. *J. Autom. Reasoning* 53(3):245–269.

Heymans, S.; Eiter, T.; and Xiao, G. 2010. Tractable reasoning with dl-programs over datalog-rewritable description logics. In *Proc. of ECAI 2010*. IOS Press.

Horridge, M., and Bechhofer, S. 2011. The OWL API: A java API for OWL ontologies. *Semantic Web* 2(1):11–21.

Hustadt, U.; Motik, B.; and Sattler, U. 2007. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning* 39(3):351–384.

Ivanov, V.; Knorr, M.; and Leite, J. 2013. A query tool for *EL* with non-monotonic rules. In *Proc. of ISWC 2013*, 216–231.

Knorr, M.; Alferes, J. J.; and Hitzler, P. 2011. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.* 175(9-10):1528–1554.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *J. ACM* 57(5).

Redl, C. 2017. Efficient evaluation of answer set programs with external sources based on external source inlining. In *Proc. of AAAI 2017*. AAAI Press.

Rosati, R. 2005. On the decidability and complexity of integrating ontologies and rules. *J. Web Sem.* 3(1):61–73.

Rosati, R. 2006. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. of KR 2006*.

Simančík, F.; Kazakov, Y.; and Horrocks, I. 2011. Consequence-based reasoning beyond horn ontologies. In *Proc. of IJCAI 2011*, 1093–1098. AAAI Press.

Swift, T. 2004. Deduction in ontologies via ASP. In *Proc. of LPNMR 2004*. Springer.