Clopen Knowledge Bases: Combining Description Logics and Answer Set Programming*

Labinot Bajraktari, Magdalena Ortiz and Mantas Šimkus

Institute of Information Systems, TU Wien, Austria

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., [23, 24, 7, 20, 19]). However, 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 [23] and $\mathcal{DL}$+LOG [24]. Each CKB is a triple $H = (P, \varphi, \Sigma)$, where P is a disjunctive Datalog program with "not" literals in rule bodies, φ is 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 (see Section 3 for details). The contributions of this paper can be summarized as follows:

- We introduce CKBs, and define for them a stable model semantics, inspired 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, in Section 4 we provide a general decidability result for checking entailment of ground atoms and consistency testing in CKBs $H = (P, \varphi, \Sigma)$, where φ is expressed in the *guarded negation fragment of FO (GNFO)* [4]. 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$^{\text{2EXPTIME}}$ upper bound for inference from GNFO-based CKBs (we note that satisfiability of GNFO formulas is 2EXPTIME-hard).
- In Section 5 we study reasoning in CKBs $H = (P, \varphi, \Sigma)$, where φ is expressed in the very expressive DL $\mathcal{ALCHIO}$. We show that the (combined) complexity of

* The work was supported by the Austrian Science Fund's projects P25207, P25518, and W1255.

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 $\mathcal{ALCHIO}$.

- We explore ways to implement reasoning in CKBs. To this end, we define a restricted class of *separable* CKBs, and present a *translation* from separable CKBs into standard ASP programs, thus enabling the reuse of existing ASP solvers. Roughly, the idea is to compile the necessary knowledge about the ontology into a set of disjunctive Datalog rules. Together with the original rules of the CKB, they form a plain ASP program whose stable models are in close correspondence with the stable models of the input CKB (see Section 5.2). The translation actively exploits the structure of the data in the input CKB in order to minimize non-deterministic choices.
- We have implemented our translation from separable CKBs with $\mathcal{ALCH}$ ontologies into plain ASP, and present here some promising empirical results. We pit our approach against an alternative implementation based on a polynomial time translation (given in the appendix of the extended version of this paper [3]) from separable CKBs into the so-called *dl-programs* [7]. Intuitively, a dl-program for a CKB effectively implements a naive algorithm for reasoning in CKBs. In particular, such a dl-program non-deterministically guesses a (relatively large) set of ground atoms, and then uses an external query (a *dl-atom*) to update the ontology that is checked for consistency by an external DL reasoner. Our experiments show that the translation into plain ASP provides a dramatic performance improvement over the implementation based on dl-programs.

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. We 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: we let $\tilde{I} = (\Delta^{\tilde{I}}, \cdot^{\tilde{I}})$ be the interpretation such that (i) $\Delta^{\tilde{I}} = \mathbf{S}_{\mathsf{const}}$, and (ii) $r^{\tilde{I}} = \{\boldsymbol{u} \mid r(\boldsymbol{u}) \in I\}$ for all $r \in \mathbf{S}_{\mathsf{pred}}$.

3 Clopen Knowledge Bases

We next define the syntax and the semantics of our hybrid language.

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

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

such that $p_1, \ldots, p_m$ are atoms. We define $head(\rho) = \{p_1, \ldots, p_k\}$, $body^+(\rho) = \{p_{k+1}, \ldots, p_l\}$, and $body^-(\rho) = \{p_{l+1}, \ldots, p_m\}$. We call the expression $not\ p$, where p is an atom, a *negated atom*. If $head(\rho) = \emptyset$, then ρ is a *constraint*.

A *program* P is a set of rules. A *Clopen Knowledge base (CKB)* is a triple $H = (P, \varphi, \Sigma)$, where 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. H. The CKB H is called *safe* if the following is satisfied for every rule $\rho \in P$: every variable that appears in ρ also appears in some atom $r(\boldsymbol{u}) \in body^+(\rho)$ 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 $r(\boldsymbol{u}) \leftarrow$ is called a *fact*. We write $r(\boldsymbol{u}) \in P$ in case the fact $r(\boldsymbol{u}) \leftarrow$ is present in a program P.

As usual, $\mathsf{dom}(f)$ denotes the *domain* of a function f, and $\mathsf{ran}(f)$ its *range*. A *substitution* σ is a 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 P (in symbols, $ground(P)$) is the ground program that consists of all ground rules ρ' such that $\rho \in P$ and $\rho' = \sigma(\rho)$ for some substitution σ. Note that $ground(P)$ is infinite when P has at least one variable.

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

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

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

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

An Herbrand interpretation I is a *stable model* of a CKB $H = (P, \varphi, \Sigma)$ if the following hold:

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

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 $H = (P, \top, \emptyset)$. 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 $P^{I,\emptyset}$ boils down to the standard Gelfond-Lifschitz reduct P^I of P w.r.t. I [13].

Observe that in a CKB $H = (P, \varphi, \emptyset)$, the theory φ plays the role of *integrity constraints* on the stable models of the plain program P, i.e. I is a stable model of H iff I is a stable model of 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 [23]. 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 $\mathcal{H} = (\mathcal{T}, \mathcal{P})$, where $\mathcal{T}$ is a theory in FO and $\mathcal{P}$ is a $\mathrm{Datalog}^{\neg,\vee}$ program as defined in [23], corresponds to the CKB $H' = (\mathcal{P}, \mathcal{T}, \Sigma)$, where Σ is the set of predicates symbols appearing in $\mathcal{T}$. One can verify that the stable models of H' are precisely the *NM-models* of $\mathcal{H}$.

In generic CKBs $H = (P, \varphi, \Sigma)$, where φ is an FO theory, the set Σ need not contain all the predicate symbols that appear in φ, i.e., closed predicates may occur in φ. Consequently, 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).

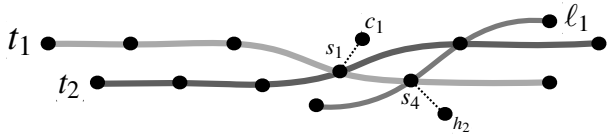
The $\mathcal{DL}$+LOG language is obtained from the r-hybrid language by allowing only DLs for specifying theories, and relaxing the safeness condition to *weak safeness* [24]. In the appendix of the extended version of this paper we show that, when sufficiently rich DLs are considered, CKBs also generalize $\mathcal{DL}$+LOG [3].

Reasoning problems. As usual in hybrid languages (see, e.g., [23]), the basic reasoning task for CKBs is *entailment of ground atoms*. That is, given a CKB $H = (P, \varphi, \Sigma)$ and a ground atom $R(\boldsymbol{u})$, the problem is to decide whether $R(\boldsymbol{u}) \in I$ holds for all stable models I of H. This problem can be reduced to checking the non-existence of a stable model for the CKB $H' = (P \cup \{\leftarrow R(\boldsymbol{u})\}, \varphi, \Sigma)$. 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 $H = (P, \varphi, \Sigma)$ 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 $P = P_1 \cup P_2 \cup P_3$, where P_1 and P_2 contain facts. The network, which is depicted by solid lines at the top of Figure 1, is described in 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 subway line; we have corresponding facts $\mathsf{SubwayLine}(\ell_1)$, $\mathsf{TramLine}(t_1)$, $\mathsf{TramLine}(t_2)$. P_2 contains facts related to locations, including the following (for convenience, $\mathsf{CloseTo}$ is depicted with dotted lines).

$$\mathsf{CloseTo}(c_1, s_1) \leftarrow, \mathsf{Hotel}(h_1) \leftarrow, \mathsf{TramConn}(h_1) \leftarrow, \mathsf{Hotel}(h_2) \leftarrow, \mathsf{CloseTo}(h_2, s_4) \leftarrow$$

The (self-explanatory) rules in P_1 and the theory φ are in Figure 1 ($\mathsf{URailConn}$ stands for urban rail connection). If h is a hotel with direct connection to the point of interest



$$
\begin{aligned}
P_3 = \{ \quad & \mathsf{SubwayStation}(Y_1) \leftarrow \mathsf{RouteTable}(X, Y_1, Y_2), \mathsf{SubwayLine}(X) \\
& \mathsf{TramStation}(Y_2) \leftarrow \mathsf{RouteTable}(X, Y_1, Y_2), \mathsf{TramLine}(X) \\
& \mathsf{ReachOnLine}(X, Y_1, Y_2) \leftarrow \mathsf{RouteTable}(X, Y_1, Y_2) \\
& \mathsf{ReachOnLine}(X, Y_1, Y_3) \leftarrow \mathsf{ReachOnLine}(X, Y_1, Y_2), \mathsf{RouteTable}(X, Y_2, Y_3) \\
& \mathsf{TramOnly}(X) \leftarrow \mathsf{TramConn}(X), \mathit{not}\ \mathsf{SubwayConn}(X) \\
& \mathsf{Q}(X) \leftarrow \mathsf{Hotel}(X), \mathsf{CloseTo}(X, Y), \mathsf{ReachOnLine}(Z, Y, Y') \\
& \qquad \mathsf{CloseTo}(c_1, Y') \\
& \mathsf{Q}'(X) \leftarrow \mathsf{Q}(X), \mathit{not}\ \mathsf{TramOnly}(X) \quad \}
\end{aligned}
$$

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

Fig. 1. Example CKB

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 subway 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 the rules in P_3 infer from them, are closed. The remaining ones are open, that is:

$$\Sigma = \{\mathsf{Hotel}, \mathsf{CloseTo}, \mathsf{Station}, \mathsf{TramConn}, \mathsf{SubwayConn}, \mathsf{URailConn}\}.$$

4 Decidable CKBs

We now turn to identifying useful settings in which the existence of a stable model for a CKB $H = (P, \varphi, \Sigma)$ is decidable. This naturally requires that the theory φ belongs to a logic $\mathcal{L}$ in which satisfiability is decidable (i.e., the set $\{\varphi \in \mathcal{L} \mid \mathit{mods}(\varphi) \neq \emptyset\}$ 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. Intuitively, this means we are interested in 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 to freely "guess" the extensions of open predicates of a given CKB H, these extensions are restricted to constants that appear in H.

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

$$r(c_1, \ldots, c_n) \vee \overline{r}(c_1, \ldots, c_n) \leftarrow$$

for each n-ary $r \in \Sigma$ occurring in P, and each tuple $(c_1, \ldots, c_n)$ of constants from P.

A stable model I of $P \cup Choose(H)$ can be seen as (partially complete) candidate for a stable model of a CKB $H = (P, \varphi, \Sigma)$. 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 H.

Proposition 1. *A CKB $H = (P, \varphi, \Sigma)$ has a stable model iff $P \cup Choose(H)$ has some stable model I for which there exists some $\mathcal{I} \in mods(\varphi)$ with the following properties:*

(C1) $(c_1^{\mathcal{I}}, \ldots, c_n^{\mathcal{I}}) \in r^{\mathcal{I}}$ *for all* $r(c_1, \ldots, c_n) \in I$,
(C2) $(c_1^{\mathcal{I}}, \ldots, c_n^{\mathcal{I}}) \notin r^{\mathcal{I}}$ *for all* $\overline{r}(c_1, \ldots, c_n) \in I$, *and*
(C3) if $(e_1, \ldots, e_n) \in r^{\mathcal{I}}$ *and* $r \notin \Sigma$, *then there exists* $r(c_1, \ldots, c_n) \in I$ *with* $c_1^{\mathcal{I}} = e_1, \ldots, c_n^{\mathcal{I}} = e_n$.

From Proposition 1, we obtain decidability of stable model existence for $H = (P, \varphi, \Sigma)$ whenever we can list the stable models of $P \cup Choose(H)$ 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 satisfiability has been established [4].

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

$$\varphi ::= r(u_1, \ldots, u_n) \mid u_1 = u_2 \mid \exists x\, \varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \alpha \wedge \neg\varphi[\boldsymbol{x}],$$

where $u_1, \ldots, u_n$ are terms, and α is an atom or an equality statement such that all variables of $\boldsymbol{x}$ 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.

The following upper bound can be shown by employing Proposition 1, and an encoding of conditions (C1–C3) in GNFO (see the appendix in [3], as well as [5] for a similar trick).

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

5 CKBs and Description Logics

GNFO is very expressive and thus computationally very expensive. We study next CKBs based on DLs, and show that they are (to a large extent) computationally not more expensive than plain ASP. We concentrate here on the expressive DL $\mathcal{ALCHIO}$. To this end, 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*. We use the elements of $\mathbf{S}_{\mathsf{const}}$ as *individuals*. The syntax for $\mathcal{ALCHIO}$ *concepts*, *roles*, concept and role *inclusions*, and *TBoxes* is the usual. We also reuse interpretations $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ as defined in Section 2, and note that the semantics to all complex concepts and roles is given by extending $\cdot^{\mathcal{I}}$ in the standard way (see [2]). The notions of *models* and *satisfiability* of TBoxes are also standard.

Example 2. The theory φ in Example 1 can be written in the syntax of $\mathcal{ALCHIO}$ as follows (we use the axiom $C \equiv D$ as a shortcut for the two inclusions $C \sqsubseteq D$, $D \sqsubseteq C$):

$$\mathsf{SubwayStation} \sqcup \mathsf{TramStation} \equiv \mathsf{Station} \qquad \mathsf{TramConn} \equiv \exists \mathsf{CloseTo}.\mathsf{TramStation}$$
$$\mathsf{SubwayConn} \equiv \exists \mathsf{CloseTo}.\mathsf{SubwayStation} \qquad \mathsf{URailConn} \equiv \exists \mathsf{CloseTo}.\mathsf{Station}$$

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

The above theorem can be proven by employing an encoding of condition (C3) of Proposition 1 by means of nominals, similarly to the encoding of DBoxes in [11].

5.1 Separability

We provide next a translation from DL-based CKBs to plain ASP. The translation is 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 (complex) concept C. These notions are defined inductively as follows:

- Every concept name A occurs positively in A.
- Every role name R occurs positively in $\exists R.C$, for any concept C.
- Every role name R occurs negatively in $\forall R.C$, for any concept C.
- 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.
- 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 $H = (P, \mathcal{T}, \Sigma)$ is* separable *if $\bigsqcap_{C \sqsubseteq D \in \mathcal{T}} (\neg C \sqcup D)$ does not have a positive occurrence of concept or role name α with $\alpha \notin \Sigma$.*

Example 3. Take the CKB $H = (P, \mathcal{T}, \Sigma)$ with $\mathcal{T} = \{\exists R.(\exists P.A) \sqsubseteq B\}$, $P = \{Q(X, Y, Z) \leftarrow T(X, Y), P(Y, Z)\}$, and $\Sigma = \{R, A, B\}$. Then H is separable because P occurs only negatively in $\neg(\exists R.(\exists P.A)) \sqcup B$.

Intuitively, in a separable CKB $H = (P, \mathcal{T}, \Sigma)$ 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 $\mathcal{ALCHIO}$ 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 in Section 3, correspond to CKBs $H = (\mathcal{P}, \mathcal{T}, \Sigma)$, 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 $(\mathcal{T}, P)$ with $\mathcal{T}, P$ from Example 3 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 $\mathcal{T}$.

5.2 Translation into Plain ASP

We describe here our translation from separable CKBs $H = (P, \mathcal{T}, \Sigma)$ into standard ASP. Intuitively, we perform reasoning about $\mathcal{T}$ *during* the translation so that afterwards $\mathcal{T}$ can be effectively forgotten. This translation is not polynomial and may take single exponential time in the size of the input. However, our experiments show that in practice this translation performs much better than the translation into dl-programs (which is provided in [3]). The translation is inspired by existing translations from expressive DLs into disjunctive Datalog [17, 9, 6], however it actively exploits the structure of the data (i.e., the facts) and is not data-independent. We limit this approach to $\mathcal{ALCH}$ TBoxes (i.e., we do not support inverses and nominals).

We assume here TBoxes in *normal form*, that is, each axiom has the form

$$A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \qquad A \sqsubseteq B_1 \sqcup \ldots \sqcup B_m \qquad A \sqsubseteq \exists R.B \tag{2}$$

$$\exists R.A \sqsubseteq B \qquad A \sqsubseteq \forall R.B \qquad R \sqsubseteq S \tag{3}$$

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., [25]).

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

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

The program $Comm(H)$ contains the direct translation of inclusions listed in (3). To deal with the remaining inclusions (the ones listed in (2)), we need the notion of *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* $types(\mathcal{T})$ *to denote the set of types over the signature of a TBox* $\mathcal{T}$ *that are consistent w.r.t.* $\mathcal{T}$.

Definition of the translation. Assume a separable CKB $H = (P, \mathcal{T}, \Sigma)$. For a TBox $\mathcal{T}$, we use $\sqsubseteq_{\mathcal{T}}$ for the transitive closure of the relation $\{(R, S) \mid R \sqsubseteq S \in \mathcal{T}\}$. For every constant c that appears in H, let $\mathfrak{t}(c, H)$ be the set of types returned by the *non-failing* runs of the following non-deterministic procedure:

(1) Let $\tau = \{A \mid P$ 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 P has a fact $R(c, d) \leftarrow$ for some d, add B to τ.
   (c) If $\top \sqsubseteq \forall S.B \in \mathcal{T}$, $R \sqsubseteq_{\mathcal{T}} S$, and P has a fact $R(d, c) \leftarrow$ for some d, add B to τ.
   If τ in not consistent 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 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 P,
   (c) in some $\exists S.A \sqsubseteq B \in \mathcal{T}$ such that 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 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 $\tau \in \mathfrak{t}(c, H)$ such that c occurs in H. We let $\mathsf{ASP}^{\mathsf{dd}}(H)$ 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, and all constants c, d of P, the disjunctive fact $R(c, d) \vee \overline{R}(c, d) \leftarrow$, where $\overline{R}$ is a fresh relation symbol
(ii) for each constant c of H, the disjunctive fact $\bigvee_{\tau \in \mathfrak{t}(c,H)} \mathsf{Type}_\tau(c) \leftarrow$
(iii) for each constant c of H and type $\tau \in \mathfrak{t}(c, H)$, the following constraints

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

The above translation yields a tool to decide the existence of a stable model for H.

Theorem 3. *The CKB* H *has a stable model iff* $\mathsf{ASP}^{\mathsf{dd}}(H)$ *has a stable model.*

6 Implementation and Experiments

We next present some experiments that demonstrate the advantages of translating a separable CKB H into a plain program $\mathsf{ASP}^{\mathsf{dd}}(H)$. We have implemented our approach in a prototype reasoner. In particular, to build the function $\mathfrak{t}$ described in Section 5.2,

instead of relying on an external DL reasoner for testing consistency of types w.r.t. a TBox, we have implemented our own algorithm for this purpose. 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. The ASP program resulting from the translation is evaluated with Clingo 4.2.1 [12].

Our implementation is written in Java and PostgreSQL 9.5.5 database, and uses OWLAPI [16] to manage ontologies. 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 system with an implementation based on a direct encoding into dl-programs (given in the appendix of [3]), and implemented in dlvhex (an implementation of dl-programs, see [21]). We note that both encodings use Clingo. For benchmarking data, we used as source real-world OpenStreetMap data[1], and followed the approach described in [10] to transform it into Datalog facts. The data describes the city of Vienna and are available from BBBike as database dumps[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, SubwayStation, etc. There are also facts about relations between these points and other constants representing objects of interest such as subway lines, types of cuisine, dishes, etc. Among the 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 subway network include locatedAlong and nextStation. The former relates a subway station to the corresponding subway 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 the TBox of our separable CKBs, we used an ontology based on the geospatial ontology found in MyITS Project [8] (currently discontinued), which was a system for smart, semantically enriched route planning over real work data sources, including OpenStreetMap (OSM) data. More specifically, we increased the expressivness of the original ontology in DL-Lite$_R$, by adding $\mathcal{ALCH}$ inclusion axioms that were relevant for our use cases. We considered different separable CKBs with the same TBox, but different programs. Four of them are provided in the appendix of [3]. Each example captures the potential information need of a tourist planning to settle 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, achieved by using the negation of the computed relation LoudNeighbourhood. Note that P_1 requires that the station close to the Hotel should be reachable with no line changes starting from "*Hauptbahnhof*", 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).

[1] `https://www.openstreetmap.org`
[2] `http://download.bbbike.org/osm/bbbike/Wien/`

	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 evaluation of P_1–P_4 (including the translation of a CKB H into $\mathsf{ASP}^{\mathsf{dd}}(H)$)

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

In this paper, we have presented CKBs which is a powerful generalization of r-hybrid 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.

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 lightweight DLs of 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 [14]. The work in [17] enables query answering services over expressive DLs using a data-independent translation into disjunctive Datalog. For Horn DLs, Heymans et al. show how dl-programs with external queries over Datalog-rewritable DLs can be translated into Datalog with stable negation [15]. Redl recently presented a generalization of this rewriting approach to external atoms in general HEX-programs [22], 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, 18]. The work in [26] shows how reasoning about DL concepts, but not general TBoxes, can be implemented in ASP.

Future work. The main task for future work is to generalize our ASP translations from separable $\mathcal{ALCH}$-based CKBs to non-separable CKBs based on more expressive DLs like $\mathcal{ALCHIO}$. Another challenge is to understand the *data complexity* of CKBs.

References

1. José Júlio Alferes, Matthias Knorr, and Terrance Swift. Query-driven procedures for hybrid mknf knowledge bases. *ACM Trans. Comput. Logic*, 14(2):16:1–16:43, June 2013.
2. Franz Baader. *The description logic handbook: theory, implementation, and applications*. Cambridge university press, 2003.
3. Labinot Bajraktari, Magdalena Ortiz, and Mantas Šimkus. Clopen knowledge bases: Combining description logics and answer set programming (with appendix), 2017. Available at `http://dbai.tuwien.ac.at/staff/simkus/papers/clopen-kbs-extended.pdf`.
4. Vince Bárány, Balder Ten Cate, and Luc Segoufin. Guarded negation. *J. ACM*, 62(3):22:1–22:26, June 2015.
5. Michael Benedikt, Pierre Bourhis, Balder ten Cate, and Gabriele Puppis. Querying visible and invisible information. In *Proc. of LICS 2016*, pages 297–306. ACM, 2016.
6. Meghyn Bienvenu, Balder ten Cate, Carsten Lutz, and Frank Wolter. Ontology-based data access: A study through disjunctive datalog, csp, and MMSNP. *ACM Trans. Database Syst.*, 39(4):33:1–33:44, 2014.
7. Thomas Eiter, Giovambattista Ianni, Thomas Lukasiewicz, Roman Schindlauer, and Hans Tompits. Combining answer set programming with description logics for the semantic web. *Artif. Intell.*, 172(12-13):p. 1495, 2008.
8. Thomas Eiter, Thomas Krennwallner, and Patrik Schneider. Lightweight spatial conjunctive query answering using keywords. In *Proc. of ESWC 2013*. Springer, 2013.
9. Thomas Eiter, Magdalena Ortiz, and Mantas Šimkus. Conjunctive query answering in the description logic SH using knots. *J. Comput. Syst. Sci.*, 78(1):47–85, 2012.
10. Thomas Eiter, Jeff Z. Pan, Patrik Schneider, Mantas Šimkus, and Guohui Xiao. A rule-based framework for creating instance data from OpenStreetMap. In *Proc. of RR 2015*. Springer, 2015.
11. Enrico Franconi, Yazmin Angélica Ibáñez-García, and Inanç Seylan. Query answering with DBoxes is hard. *Electr. Notes Theor. Comput. Sci.*, 278:71–84, 2011.
12. Martin Gebser, Benjamin Kaufmann, Roland Kaminski, Max Ostrowski, Torsten Schaub, and Marius Thomas Schneider. Potassco: The potsdam answer set solving collection. *AI Commun.*, 24(2):107–124, 2011.
13. Michael Gelfond and Vladimir Lifschitz. The stable model semantics for logic programming. In *Proc. of ICLP/SLP 1988*, pages 1070–1080. MIT Press, 1988.
14. Birte Glimm, Ian Horrocks, Boris Motik, Giorgos Stoilos, and Zhe Wang. Hermit: An OWL 2 reasoner. *J. Autom. Reasoning*, 53(3):245–269, 2014.
15. Stijn Heymans, Thomas Eiter, and Guohui Xiao. Tractable reasoning with dl-programs over datalog-rewritable description logics. In *Proc. of ECAI 2010*. IOS Press, 2010.
16. Matthew Horridge and Sean Bechhofer. The OWL API: A java API for OWL ontologies. *Semantic Web*, 2(1):11–21, 2011.
17. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reasoning in description logics by a reduction to disjunctive datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
18. Vadim Ivanov, Matthias Knorr, and Joao Leite. A query tool for EL with non-monotonic rules. In *Proc. of ISWC 2013*. Springer, 2013.
19. Matthias Knorr, José Júlio Alferes, and Pascal Hitzler. Local closed world reasoning with description logics under the well-founded semantics. *Artif. Intell.*, 175(9-10):1528–1554, 2011.
20. Boris Motik and Riccardo Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.

21. Christoph Redl. The dlvhex system for knowledge representation: recent advances (system description). *TPLP*, 16(5-6):866–883, 2016.
22. Christoph Redl. Efficient evaluation of answer set programs with external sources based on external source inlining. In *Proc. of AAAI 2017*. AAAI Press, February 2017.
23. Riccardo Rosati. On the decidability and complexity of integrating ontologies and rules. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(1):61 – 73, 2005. Rules Systems.
24. Riccardo Rosati. DL+log: Tight integration of description logics and disjunctive datalog. In *Proc. of KR 2006*. AAAI Press, 2006.
25. František Simančík, Yevgeny Kazakov, and Ian Horrocks. Consequence-based reasoning beyond horn ontologies. In *Proc. of IJCAI 2011*, pages 1093–1098. AAAI Press, 2011.
26. Terrance Swift. Deduction in ontologies via ASP. In *Proc. of LPNMR 2004*. Springer, 2004.