Expressivity of Planning with Horn Description Logic Ontologies

Stefan Borgwardt[1], Jörg Hoffmann[2], Alisa Kovtunova[1], Markus Krötzsch[1], Bernhard Nebel[3], Marcel Steinmetz[2]

[1]Faculty of Computer Science, Technische Universität Dresden, Germany,
[2]Saarland University, Saarland Informatics Campus, Germany
[3]Faculty of Engineering, University of Freiburg, Germany
stefan.borgwardt@tu-dresden.de, hoffmann@cs.uni-saarland.de, alisa.kovtunova@tu-dresden.de, markus.kroetzsch@tu-dresden.de, nebel@informatik.uni-freiburg.de, steinmetz@cs.uni-saarland.de

Abstract

State constraints in AI Planning globally restrict the legal environment states. Standard planning languages make closed-domain and closed-world assumptions. Here we address open-world state constraints formalized by planning over a description logic (DL) ontology. Previously, this combination of DL and planning has been investigated for the light-weight DL DL-Lite. Here we propose a novel compilation scheme into standard PDDL with derived predicates, which applies to more expressive DLs and is based on the rewritability of DL queries into Datalog with stratified negation. We also provide a new rewritability result for the DL Horn-$\mathcal{ALCHOIQ}$, which allows us to apply our compilation scheme to quite expressive ontologies. In contrast, we show that in the slight extension Horn-$\mathcal{SROIQ}$ no such compilation is possible unless the weak exponential hierarchy collapses. Finally, we show that our approach can outperform previous work on existing benchmarks for planning with DL ontologies, and is feasible on new benchmarks taking advantage of more expressive ontologies.

Introduction

AI planning is concerned with sequential decision making problems where an agent needs to choose actions to achieve a goal, or to maximize reward (Ghallab, Nau, and Traverso 2004). Such problems are compactly described in a declarative language. Specifically, in the most basic ("classical") version of planning, a planning task describes an initial state of the agent's environment, a set of actions that can affect that environment, and a goal formula that is to be satisfied. In order to reach the goal, actions can be applied whenever their preconditions are satisfied in the current state. Here we are interested in *state constraints*, constraints that should hold globally, i.e. at every state, in difference to preconditions which merely need to hold locally. Moreover, standard planning formalisms (based on variants of the PDDL language (McDermott et al. 1998; Haslum et al. 2019)) follow closed-domain and closed-world assumptions, in which absent facts are assumed to be false and no new objects can be created. In particular, these assumptions underly state constraints as can be specified in PDDL3 (Gerevini et al. 2009). Here we instead target open-domain, open-world reasoning.

One way to do this is via *explicit-input Knowledge and Action Bases (eKABs)* (Calvanese et al. 2016), where states (sets of ground atoms) are interpreted using open-world semantics. All states are subject to a *background ontology*, which describes high-level concepts and global state constraints. Together, a state and an ontology describe a multitude of possible worlds, which leaves room for unknown information about existing and unknown objects. For example, an ontology could express that "everyone operating a machine works for an engineering department" and "everyone who works for a department is an employee" without explicitly identifying the department of each person or even stating that they are employees. Action preconditions contain *queries* that are evaluated under open-world semantics, e.g. the query for all "employees" would return all machine operators among other people. Finally, action effects add or remove atoms in the state, e.g. reassign machines or departments. This allows for a clean separation of what is directly observed (i.e. the state, which contains, e.g. operational data and sensor data) from what is indirectly inferred (using the ontology).

Calvanese et al. (2016) investigated eKABs with ontologies and queries formulated in the description logic DL-Lite, which is a popular formalism for conceptual modeling (Calvanese et al. 2007b). Queries in this logic enjoy *first-order rewritability*, which means that the queries and the ontology can be compiled into first-order (FO) formulas that are then evaluated under closed-world semantics. Based on this property, the authors described a compilation of DL-Lite eKABs into classical PDDL planning tasks. Later, this compilation was further optimized to enable practical planning with DL-Lite background ontologies (Borgwardt et al. 2021).

The goal of this paper is to extend the expressivity of state constraints in eKABs from the light-weight DL-Lite to more powerful description logics (DLs) (Baader et al. 2017). We mainly consider *Horn* DLs, which are fragments of Horn-FOL (Krötzsch, Rudolph, and Hitzler 2013; Jung et al. 2019). We investigate for which Horn DLs compilations into PDDL exist. For this purpose, we adapt the notion of *compilation schemes* (Nebel 2000; Thiébaux, Hoffmann, and Nebel 2005), which relate the expressivity of two formalisms. On the one hand, polynomial compilation schemes show that the expressivity of DL eKABs is not higher than that of PDDL. On the other hand, although such eKABs could be considered syntactic sugar, they represent powerful tools that allow domain engineers to add open-world constraints to planning tasks.

Our first contribution is a generic compilation scheme for any DL and queries that enjoy *Datalog*$^\neg$*-rewritability*, which essentially allows us to compile them into a set of PDDL *derived predicates*. Using this, we can immediately employ many existing rewritability results from the DL literature for AI planning (Ortiz, Rudolph, and Šimkus 2010; Eiter et al. 2012; Bienvenu and Ortiz 2015). We continue by describing a novel polynomial Datalog$^\neg$-rewriting for queries in the very expressive DL Horn-$\mathcal{ALCHOIQ}$ (Ortiz, Rudolph, and Šimkus 2011), which allows us to extend the previous compilability result even further. In contrast to this, we then show that such a compilation cannot exist (under a reasonable complexity-theoretic assumption) for the slightly more expressive DL Horn-$\mathcal{SROIQ}$ (Ortiz, Rudolph, and Šimkus 2011). For this, we follow the idea of a previous non-compilability result for PDDL with derived predicates into PDDL without derived predicates (Thiébaux, Hoffmann, and Nebel 2005). This more or less draws a line between the standardized ontology languages OWL 1,[1] which results in planning tasks of equal expressivity as PDDL, and OWL 2,[2] where queries are strictly more expressive than PDDL.

While polynomial rewritings are nice in theory, they are often not practical due to a polynomial increase in the arity of predicates. We therefore conclude the paper with an experimental evaluation that combines an existing practical implementation of an (exponential) Datalog$^\neg$-rewriting for Horn-$\mathcal{SHIQ}$ (Eiter et al. 2012) with our generic compilation scheme, compares this against previous approaches for DL-Lite eKABs on existing benchmarks (Calvanese et al. 2016; Borgwardt et al. 2021), and also introduces new benchmarks exploiting the newly increased expressivity.

Preliminaries

We first introduce description logics, Datalog$^\neg$, planning with derived predicates and ontologies, and compilations between these formalisms. As usual, we use the symbol $\models$ with two different meanings: open-world *entailment* of formulas from sets of formulas, where all possible interpretations of arbitrary (even infinite) size are considered, and closed-world *satisfaction* of a formula in a fixed, finite interpretation. It should always be clear from the context which one is used.

Description Logics. Description logics are a family of KR formalisms (Baader et al. 2017) that describe open-world knowledge using *axioms* in restricted first-order logic over unary and binary predicates. Members of this family differ in their expressivity and complexity. A *TBox* (*ontology*) is a finite set of *DL axioms*, which can be seen as first-order sentences. The precise syntax of these axioms depends on the specific DL that is used, but this is not important for most of the paper. A *state* (*ABox*) is a finite set of *ground atoms* $p(\vec{c})$, where p is a predicate and $\vec{c}$ is a sequence of *objects* (*constants*). Since we use standard planning formalisms, we also allow predicates of arity higher than 2 in states, but those cannot occur in TBoxes, so essentially have a closed-world semantics. For a state s, $\mathcal{O}(s)$ is the set of all objects occurring in s. Two special unary predicates are $\top$ and $\bot$, which always evaluate to *true* and *false*, respectively.

Queries. A *conjunctive query (CQ)* is a formula of the form $q(\vec{x}) = \exists\vec{y}.\phi(\vec{x},\vec{y})$, where ϕ is a conjunction of unary and binary atoms. A *union of CQs (UCQ)* is a disjunction of CQs. An *instance query (IQ)* is a CQ of the form $p(x)$, where p is unary. The central reasoning problem is to decide whether $s,\mathcal{T},\theta \models q$, where s is a state, $\mathcal{T}$ a TBox, and θ an assignment of objects from $\mathcal{O}(s)$ to the free variables in the (U)CQ q. In an abuse of notation, we may denote with $\bot$ the CQ $\exists x.\bot(x)$. *ECQs* were introduced to combine open-world and closed-world reasoning (Calvanese et al. 2007a). We further extend ECQs by "closed-world atoms" that can also be of higher arity. ECQs are defined by the grammar $Q ::= p(\vec{x}) \mid [q] \mid \neg Q \mid Q \wedge Q \mid \exists y.Q$, where p is a predicate, $\vec{x}$ are terms, q is a UCQ, and y is a variable. The semantics of ECQs is defined as follows:

$$
\begin{array}{ll}
s,\mathcal{T},\theta \models p(\vec{x}) & \text{iff } s \models p(\theta(\vec{x})) \\
s,\mathcal{T},\theta \models [q] & \text{iff } s,\mathcal{T},\theta \models q \\
s,\mathcal{T},\theta \models \neg Q_1 & \text{iff } s,\mathcal{T},\theta \not\models Q_1 \\
s,\mathcal{T},\theta \models Q_1 \wedge Q_2 & \text{iff } s,\mathcal{T},\theta \models Q_1 \text{ and } s,\mathcal{T},\theta \models Q_2 \\
s,\mathcal{T},\theta \models \exists y.Q_1 & \text{iff } \exists o \in \mathcal{O}(s) : s,\mathcal{T},\theta[y \mapsto o] \models Q_1
\end{array}
$$

There is a difference between the ECQs $B(x)$, which is answered directly in the (closed-world) model described by s, and $[B(x)]$, which is evaluated w.r.t. the TBox as well. For example, if $s = \{C(a)\}$ and $\mathcal{T} = \{C \sqsubseteq B\}$, then $B(x)$ is not satisfied by any instantiations, but $[B(x)]$ is.

Datalog$^\neg$. A *Datalog*$^\neg$ *rule* is a formula $p(\vec{x}) \leftarrow \Phi(\vec{x},\vec{y})$ whose *body* $\Phi(\vec{x},\vec{y})$ is a conjunction of literals and whose *head* $p(\vec{x})$ is an atom. A set of Datalog$^\neg$ rules $\mathcal{R}$ is *stratified* if the set of its predicates can be partitioned into $\mathcal{P}_1,\dots,\mathcal{P}_n$ such that, for all $p_i \in \mathcal{P}_i$ and $p_i(\vec{x}) \leftarrow \Phi(\vec{x},\vec{y}) \in \mathcal{R}$,

- if $p_j \in \mathcal{P}_j$ occurs in $\Phi(\vec{x},\vec{y})$, then $j \leq i$, and
- if $p_j \in \mathcal{P}_j$ occurs negated in $\Phi(\vec{x},\vec{y})$, then $j < i$.

In the following, all sets of Datalog$^\neg$ rules are stratified. Datalog is the restriction of Datalog$^\neg$ to positive rule bodies.

All variables in Datalog$^\neg$ rules are implicitly universally quantified. Given a state s and a set of Datalog$^\neg$ rules $\mathcal{R}$, we denote by $\mathcal{R}(s)$ the minimal Herbrand model of $s \cup \mathcal{R}$.

Definition 1. *A TBox $\mathcal{T}$ and a UCQ $q(\vec{x})$ are* Datalog$^\neg$-rewritable *if there is a set of Datalog$^\neg$ rules $\mathcal{R}_{\mathcal{T},q}$ with a predicate P_q such that, for all states s and substitutions θ of $\vec{x}$ in $\mathcal{O}(s)$, we have $s,\mathcal{T},\theta \models q(\vec{x})$ iff $\mathcal{R}_{\mathcal{T},q}(s) \models P_q(\theta(\vec{x}))$.*

A Datalog$^\neg$-rewriting may use additional predicates and constants, but only needs to be correct for the original symbols. We talk about *Datalog-rewritability* if the set $\mathcal{R}_{\mathcal{T},q}$ does not contain negation. A variety of such rewritability results for DLs exist, for example a very complex (but polynomial-size) Datalog-rewriting for IQs over Horn-$\mathcal{ALCHOIQ}$ (Ortiz, Rudolph, and Šimkus 2010), a polynomial-size Datalog-rewriting for IQs in $\mathcal{EL}^{++}$ (Krötzsch 2011), or an exponential-size Datalog-rewriting for UCQs over Horn-$\mathcal{SHIQ}$ TBoxes implemented in the CLIPPER system[3] (Eiter

[1] http://www.w3.org/TR/owl-features/

[2] http://www.w3.org/TR/owl2-overview/

[3] https://github.com/ghxiao/clipper

et al. 2012), which is polynomial for the sublogic $\mathcal{ELH}_\bot$ (Bienvenu and Ortiz 2015).

Datalog$^\neg$-rewritability naturally extends to ECQs Q: take the disjoint union $\mathcal{R}_{\mathcal{T},Q}$ of all $\mathcal{R}_{\mathcal{T},q}$ for UCQs q occurring in Q and construct an FO formula $Q_\mathcal{T}$ by replacing each UCQ atom $[q(\vec{x})]$ in Q with $P_q(\vec{x})$. Then $s, \mathcal{T}, \theta \models Q(\vec{x})$ is equivalent to $\mathcal{R}_{\mathcal{T},Q}(s) \models Q_\mathcal{T}(\theta(\vec{x}))$ (Calvanese et al. 2007a).

PDDL with Derived Predicates. We recall PDDL 2.1 extended with derived predicates (Fox and Long 2003; Hoffmann and Edelkamp 2005). In this context, states are viewed under the closed-world assumption and all sets are finite.

Definition 2. *A* PDDL domain description *is a tuple* $(\mathcal{P}, \mathcal{P}_{\mathsf{der}}, \mathcal{A}, \mathcal{R})$*, where* $\mathcal{P}, \mathcal{P}_{\mathsf{der}}$ *are disjoint sets of predicates;* $\mathcal{A}$ *is a set of* actions*; and* $\mathcal{R}$ *is a set of* rules*. An* action *is of the form* $(\vec{x}, \mathsf{pre}, \mathsf{eff})$*, with* parameters $\vec{x}$*,* precondition pre*, and a finite set* eff *of* effects*. The precondition is an FO formula over* $\mathcal{P} \cup \mathcal{P}_{\mathsf{der}}$ *with free variables from* $\vec{x}$*, and an* effect *is of the form* $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del})$*, where* $\vec{y}$ *are variables,* cond *is an FO formula over* $\mathcal{P} \cup \mathcal{P}_{\mathsf{der}}$ *with free variables from* $\vec{x} \cup \vec{y}$*,* add *is a finite set of atoms over* $\mathcal{P}$ *(without* $\mathcal{P}_{\mathsf{der}}$*) with free variables from* $\vec{x} \cup \vec{y}$*, and* del *is a finite set of such negated atoms.* Rules *are of the form* $P(\vec{x}) \leftarrow \phi(\vec{x})$ *with* $P \in \mathcal{P}_{\mathsf{der}}$ *and a first-order formula* ϕ *over* $\mathcal{P} \cup \mathcal{P}_{\mathsf{der}}$*. The set* $\mathcal{R}$ *must be* stratified*, i.e. fulfill the same condition as sets of Datalog*$^\neg$ *rules when considering the rule bodies* $\phi(\vec{x})$ *in NNF.*

A PDDL task *is a tuple* $(\Delta, \mathcal{O}, I, G)$*, where* Δ *is a PDDL domain description;* $\mathcal{O}$ *is a finite set of* objects *including the ones in* Δ*;* I *is the* initial state*; and* G *is the* goal*, a closed FO formula over* $\mathcal{P} \cup \mathcal{P}_{\mathsf{der}}$ *and constants from* $\mathcal{O}$*.*

Derived predicates are not allowed to be modified by actions, i.e. they are only determined by the current state and the rules. The semantics of rules is defined similarly to Datalog$^\neg$, i.e. for a state s, $\mathcal{R}(s)$ is the minimal Herbrand model obtained by exhaustively applying the rules in $\mathcal{R}$, stratum by stratum, to the facts in s in order to populate the derived predicates $\mathcal{P}_{\mathsf{der}}$. In fact, all such rule sets $\mathcal{R}$ can be reformulated into Datalog$^\neg$ rule sets (Abiteboul, Hull, and Vianu 1995; Thiébaux, Hoffmann, and Nebel 2005). Although the definition of derived predicates requires the head and body to have the same free variables, this is compatible with the semantics of Datalog$^\neg$ since additional body variables can be viewed as implicitly existentially quantified.

For an action $a = (\vec{x}, \mathsf{pre}, \mathsf{eff})$ and $\theta\colon \vec{x} \to \mathcal{O}$, the *ground action* $\theta(a)$ has no parameters. A ground action $a = (\mathsf{pre}, \mathsf{eff})$ is *applicable* in a state s if $\mathcal{R}(s) \models \mathsf{pre}$ and its *application* yields a new state $s[\![a]\!]$ that contains a ground atom α iff (1) there are $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del}) \in \mathsf{eff}$ and θ such that $\mathcal{R}(s) \models \theta(\mathsf{cond})$ and $\alpha \in \theta(\mathsf{add})$; or (2) $\alpha \in s$ and for all $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del}) \in \mathsf{eff}$ and θ it holds that $\mathcal{R}(s) \not\models \theta(\mathsf{cond})$ or $\neg\alpha \notin \theta(\mathsf{del})$. A *plan* π is a sequence of ground actions such that π is applicable in I and $\mathcal{R}(I[\![\pi]\!]) \models G$.

eKABs. We recall *explicit-input action and knowledge bases* (Calvanese et al. 2016), but slightly adapt the notation to be consistent with PDDL notation.

Definition 3. *An* eKAB domain description *is a tuple* $(\mathcal{P}, \mathcal{A}, \mathcal{T})$*, where* $\mathcal{P}$ *is a finite set of predicates;* $\mathcal{A}$ *is a finite set of* DL actions*; and* $\mathcal{T}$ *is a TBox over the unary and binary predicates in* $\mathcal{P}$*. A* DL action *is of the form* $(\vec{x}, \mathsf{pre}, \mathsf{eff})$*, where* pre *is an ECQ over* $\mathcal{P}$ *with free variables from* $\vec{x}$*, and* eff *consists of* DL effect *of the form* $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del})$*, where* cond *is an ECQ over* $\mathcal{P}$ *with free variables from* $\vec{x} \cup \vec{y}$*, and* add *and* del *are as in PDDL.*

An eKAB (task) *is a tuple* $(\Delta, \mathcal{O}, \mathcal{O}_0, I, G)$*, where* Δ *is an eKAB domain description;* $\mathcal{O}$ *is a possibly infinite set of* objects*;* $\mathcal{O}_0$ *is a finite subset of* $\mathcal{O}$ *including the objects from* Δ*;* I *is the* initial state *(over* $\mathcal{O}_0$*), which is consistent with* $\mathcal{T}$*; and* G *is the* goal*, a closed ECQ over* $\mathcal{P}$ *and constants from* $\mathcal{O}_0$*.*

A ground action a is *applicable* in s if $s \models \mathsf{pre}$ and $s[\![a]\!] \cup \mathcal{T}$ is consistent. The *application* $s[\![a]\!]$ contains a fact α iff (1') there are $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del}) \in \mathsf{eff}$ and $\theta\colon \vec{y} \to \mathcal{O}(s) \cup \mathcal{O}_0$ such that $s, \mathcal{T}, \theta \models \mathsf{cond}$ and $\alpha \in \theta(\mathsf{add})$; or (2') $\alpha \in s$ and for all $(\vec{y}, \mathsf{cond}, \mathsf{add}, \mathsf{del}) \in \mathsf{eff}$ and θ as above it holds that $s, \mathcal{T}, \theta \not\models \mathsf{cond}$ or $\neg\alpha \notin \theta(\mathsf{del})$. A *plan* π must be applicable in I and satisfy $I[\![\pi]\!], \mathcal{T} \models G$. Substitutions for effects range over $\mathcal{O}(s) \cup \mathcal{O}_0$ since the TBox may contain objects from $\mathcal{O}_0$. In the following, we assume w.l.o.g. that $\mathcal{O}_0 \subseteq \mathcal{O}(s)$.

Additional Assumptions. Actions can refer to new objects (parameters $\vec{x}$ that are not in the precondition), and thereby increase the number of objects in a state. To obtain manageable state transition systems, in the literature the assumption of *state-boundedness* is often considered for eKAB-like formalisms; it requires that there exists a bound b such that any state reachable from I contains at most b objects (Calvanese et al. 2013; De Giacomo et al. 2014). For a fixed b, b-boundedness of an eKAB is decidable (De Giacomo et al. 2014). Even if a given eKAB is not state-bounded, one could instead ask for the existence of a b-bounded plan (Ahmetaj et al. 2017). An abstraction result implies that any b-bounded eKAB can be reformulated into one where $|\mathcal{O}| = |\mathcal{O}_0| + n + b$, where n is the maximum number of parameters of any action (Calvanese et al. 2013, 2016). Any plan of the original eKAB can still be encoded using this finite set of objects. Conversely, any abstract plan can be reformulated into a plan of the original eKAB by replacing the $n + b$ abstract objects by fresh objects from the original set $\mathcal{O}$ where necessary (i.e. if those objects did not occur in the previous state). We will also make this assumption here and assume for simplicity that $\mathcal{O} = \mathcal{O}_0$ is finite and denote both sets by $\mathcal{O}$.

Even for a b-bounded eKAB, the TBox $\mathcal{T}$ can entail the existence of objects that are not mentioned in a state s. These objects are not affected by the bound b, because they are never explicitly materialized in s. Hence, the reasoning problems still employ standard DL semantics rather than fixed-domain reasoning (Gaggl, Rudolph, and Schweizer 2016).

We also assume that goals of eKAB and PDDL tasks consist of a single (closed-world) atom $g(\vec{c})$. If that is not the case, we can introduce a new action with the goal formula $G(\vec{x})$ as precondition that adds $g(\vec{x})$ to the state. The parameters $\vec{x}$ correspond to the constants $\vec{c}$ in the original goal formula $G(\vec{c})$. This assumption simplifies some of the formal definitions, but does not affect our main insights. Without this assumption, for example, the following definition of domain compilation f_δ would also need to depend on the goal formula since we later need to compile all UCQs (also those in the goal) into derived predicates.

Compilations. To study the relative expressivity of these formalisms, we adapt the notion of compilation schemes (Nebel 2000; Thiébaux, Hoffmann, and Nebel 2005).

Definition 4. *A* compilation scheme $\mathbf{f}$ *from eKABs to PDDL is a tuple of functions* $(f_\delta, f_o, f_i, f_g)$ *that induces a function* F *from eKAB tasks* $\Pi = (\Delta, \mathcal{O}, I, G)$ *to PDDL tasks*

$$F(\Pi) := \big(f_\delta(\Delta), \mathcal{O} \cup f_o(\Delta), f_i(\mathcal{O}, I), f_g(\mathcal{O}, G)\big)$$

such that (A) there exists a plan for Π *iff there exists a plan for* $F(\Pi)$*; and (B)* f_i *and* f_g *are polynomial-time computable. If* $\|f_\delta(\Delta)\|$ *and* $\|f_o(\Delta)\|$ *are bounded polynomially (exponentially) in* $\|\Delta\|$*, then* $\mathbf{f}$ *is* polynomial *(*exponential*).*

If for every plan P *solving an instance* Π *there exists a plan* P' *solving* $F(\Pi)$ *such that* $\|P'\| \leq c \cdot \|P\|^n + k$ *for positive integer constants* c, n, k*, we say that* $\mathbf{f}$ preserves plan size polynomially*. If* $n = 1$*, it* preserves plan size linearly*, and if additionally* $c = 1$*, then it* preserves plan size exactly.

To be considered of the same expressivity, there should be a polynomial compilation scheme between two formalisms that at least preserves plan size polynomially, but ideally exactly. Compilation schemes for specific DLs are restricted to TBoxes $\mathcal{T}$ formulated in the specified DL. For example, there exists an exponential compilation scheme for DL-Lite eKABs that rewrites ECQs in-situ into FO conditions (Calvanese et al. 2016). This compilation has been optimized by Borgwardt et al. (2021) by using derived predicates to simplify the conditions. In contrast, the compilations we investigate in the following directly use Datalog$^\neg$-rewritings to compile UCQs into derived predicates.

Compiling TBoxes into Derived Predicates

We start by describing a generic compilation that exploits Datalog$^\neg$-rewritability of specific DLs to compile open-world ECQs into closed-world formulas using derived predicates.

One restriction of PDDL derived predicates is that they cannot occur in action effects. However, Datalog$^\neg$-rewritings may derive new facts about the predicates occurring in the state and TBox, which can also occur in action effects. To circumvent this issue, we observe that query rewriting is only necessary for *evaluating conditions*, but does not affect the states themselves. Therefore, we separate the condition evaluation and action effects by using two disjoint signatures of predicates: we use the Datalog$^\neg$-rewriting on a copy s' of the state s in which each original predicate P has been replaced by a copy P'. This copying process can be simulated by making P' a derived predicate with the rule $P'(\vec{x}) \leftarrow P(\vec{x})$. In the following, we denote by $\mathcal{T}'$ the result of replacing each predicate P in $\mathcal{T}$ by P', and likewise for ECQs Q.

Another issue is that the rewriting may introduce additional constants, which are not allowed for instantiating actions, because that would change their behavior. We simulate this via two new predicates S and N and a new action $a_{S,N}$ with precondition $\neg S$ and unconditional effect S and $N(o)$ for all objects o that are not in the original domain description. All other action conditions are also extended by S and $\neg N(\vec{x}) := \bigwedge_{x \in \vec{x}} \neg N(x)$ for their parameters $\vec{x}$, to ensure that they can only be instantiated by the original objects.

Definition 5. *Let* $((\mathcal{P}, \mathcal{A}, \mathcal{T}), \mathcal{O}, I, G)$ *be a b-bounded eKAB for which all UCQs as well as* $\bot$ *are Datalog*$^\neg$*-rewritable w.r.t.* $\mathcal{T}$*. Let* $\mathcal{R}_{\mathcal{T}'}$ *be the disjoint union of* $\mathcal{R}_{\mathcal{T}',\bot}$ *and all* $\mathcal{R}_{\mathcal{T}',Q'}$ *for ECQs* Q *in the eKAB. Then the PDDL task* $((\mathcal{P} \cup \{S, N\}, \mathcal{P}', \mathcal{A}', \mathcal{R}'), \mathcal{O}', I, G')$ *is obtained as follows:*

- $\mathcal{P}'$ *consists of the predicates occurring in* $\mathcal{R}_{\mathcal{T}'}$*;*
- $\mathcal{A}'$ *contains* $a_{S,N}$ *and all actions obtained from* $\mathcal{A}$ *by replacing preconditions* pre *by* $S \wedge \neg N(\vec{x}) \wedge \neg P_\bot \wedge \mathsf{pre}'_{\mathcal{T}'}$ *and effect conditions* cond *by* $\mathsf{cond}'_{\mathcal{T}'}$*;*
- $\mathcal{R}' = \mathcal{R}_{\mathcal{T}'} \cup \{P'(\vec{x}) \leftarrow P(\vec{x}) \mid P \in \mathcal{P}\}$*;*
- $\mathcal{O}' = \mathcal{O} \cup \mathcal{O}(\mathcal{R}_{\mathcal{T}'})$*; and*
- $G' = S \wedge \neg P_\bot \wedge G$*.*

The goal does not need to be rewritten w.r.t. $\mathcal{T}$ since we assumed that it is a single (closed-world) atom.

Theorem 1. *Def. 5 is a compilation scheme from Datalog*$^\neg$*-rewritable eKABs to PDDL that preserves plan size exactly.*

Proof. The new goal G' can be computed in polynomial time since we only add $S \wedge \neg P_\bot$. Moreover, $a_{S,N}$ and the set of rules $\mathcal{R}_{\mathcal{T}'}$ depend only on the objects and conditions occurring in the original eKAB domain description. We show that all plans of either planning task are also plans for the other (modulo the initializing action $a_{S,N}$).

Consistency of $s \cup \mathcal{T}$ is equivalent to $\mathcal{R}_{\mathcal{T}',\bot}(s') \not\models P_\bot$, where s' is obtained from s by replacing each P with P'. Hence, the rules $P'(\vec{x}) \leftarrow P(\vec{x})$ and the conjuncts $\neg P_\bot$ in all conditions ensure that all states reached while executing a plan for the PDDL task are consistent with $\mathcal{T}$. This includes the initial state since I is consistent with $\mathcal{T}$ by assumption.

Similarly, for any ECQ Q, we have $\mathcal{R}'(s) \models Q'_{\mathcal{T}'}(\theta(\vec{x}))$ iff $s', \mathcal{T}', \theta \models Q'(\vec{x})$, which is equivalent to $s, \mathcal{T}, \theta \models Q(\vec{x})$. Due to $a_{S,N}$, the substitutions for instantiating action and effect conditions range over the same objects in both tasks. Hence, both formalisms allow equivalent action applications in each state and can reach a goal state by the same plans. $\square$

For example, this immediately implies that eKABs with Horn-$\mathcal{SHIQ}$ TBoxes have exponential compilations into PDDL with derived predicates (without negation) and for $\mathcal{ELH}_\bot$ and DL-Lite we even obtain polynomial compilations (Eiter et al. 2012; Bienvenu and Ortiz 2015). The latter also holds for Horn-$\mathcal{SHOIQ}$ if all conditions are restricted to EIQs (Ortiz, Rudolph, and Šimkus 2010). In general, our construction applies to any ontology language where UCQs (or IQs) are Datalog$^\neg$-rewritable, and to any specific TBoxes and queries that happen to be Datalog$^\neg$-rewritable.

A Polynomial Rewriting for Horn-$\mathcal{ALCHOIQ}$

To extend the compilability results, we develop a polynomial-size rewriting for UCQs over Horn-$\mathcal{ALCHOIQ}$ into Datalog$^\neg$. It is based on a query answering approach developed by Carral, Dragoste, and Krötzsch (2018) and encodes the relevant definitions from their paper into Datalog$^{S,\neg}$ rules, which extend Datalog$^\neg$ by *set terms* that denote sets of objects. We then adapt a known polynomial translation to obtain a set of Datalog$^\neg$ rules (Ortiz, Rudolph, and Šimkus 2010).

The full details can be found in a technical report (Borgwardt et al. 2022), but we describe the main ideas here. The rewriting starts by translating the Horn-$\mathcal{ALCHOIQ}$ axioms of the given TBox $\mathcal{T}$ into Datalog rules (Carral, Dragoste, and Krötzsch 2018). However, since the resulting rule set is exponential, we here reformulate it into polynomially many Datalog$^{S,\neg}$ rules, loosely following ideas from Ortiz, Rudolph, and Šimkus (2010). The original approach uses exponentially many constants t_X, where X is a set of unary predicates, to describe anonymous objects that satisfy X. In Datalog$^{S,\neg}$, these individuals can directly be described by sets $\{X\}$ that can be used as arguments to predicates. For example, our rewriting introduces a new predicate $\mathsf{role}(r, X, Y)$ to express that the objects represented by X and Y are connected by the binary predicate r, which is now viewed as an additional element of $\mathcal{O}$. Using such additional predicates, the translation of the original Datalog rules by Carral, Dragoste, and Krötzsch (2018) is straightforward.

The remainder of the rewriting encodes the *filtration phase* from that paper into several strata of Datalog$^{S,\neg}$ rules. We use bespoke predicates to encode the constructions of expanded states and graphs in Definitions 7 and 8 in (Carral, Dragoste, and Krötzsch 2018), e.g. to compute a partial expansion of the input state to obtain more query matches and an acyclicity check over a dependency graph between query variables to filter out spurious matches. After a translation from Datalog$^{S,\neg}$ into Datalog$^{\neg}$ (Ortiz, Rudolph, and Šimkus 2010), correctness of the rewriting follows mostly from Theorem 3 in the paper by Carral, Dragoste, and Krötzsch (2018).

Theorem 2. *UCQs over Horn-$\mathcal{ALCHOIQ}$ TBoxes are Datalog$^{\neg}$-rewritable with rewritings of polynomial size.*

By Theorem 1, we thus obtain a polynomial compilation scheme for Horn-$\mathcal{ALCHOIQ}$ eKABs into PDDL that preserves plan size exactly. Admittedly, this construction is rather complex, but theoretically very interesting, in particular in light of the next section.

Non-Compilability for Expressive eKABs

As a counterpoint to the previous section, we now prove that polynomial compilations cannot exist for Horn-$\mathcal{SROIQ}$, not even if we allow the plan size to increase polynomially. Horn-$\mathcal{SROIQ}$ differs from Horn-$\mathcal{ALCHOIQ}$ only in allowing one additional type of axiom, called *complex role inclusions*. The following result is inspired by a similar non-compilability result for PDDL with derived predicates (Thiébaux, Hoffmann, and Nebel 2005). We start with some observations about the complexity of the involved problems. The *polynomial-step planning problem* is to decide whether a given planning task has a plan of length polynomial (for some given polynomial), and the *1-step planning problem* is the special case where the polynomial is 1.

Theorem 3. *The polynomial-step planning problem for PDDL is* ExpTime*-complete.*

Proof. Hardness follows from the complexity of the 1-step planning problem for PDDL with derived predicates (Thiébaux, Hoffmann, and Nebel 2005, Theorem 1). Membership can be seen as follows. In exponential time, we can enumerate all plans of polynomial length (for a fixed polynomial). For each such plan, we can check whether each ground action was applicable, which facts were generated or deleted, and whether the goal is satisfied in the end. The most complex part of this check is the evaluation of the derived predicates after each action, which can be done in exponential time (Dantsin et al. 2001). □

Theorem 4. *The 1-step planning problem for Horn-$\mathcal{SROIQ}$ eKABs is* 2ExpTime*-complete.*

Proof. Hardness follows from the complexity of reasoning in Horn-$\mathcal{SROIQ}$ (Ortiz, Rudolph, and Šimkus 2010). Membership holds since we can enumerate all candidate 1-step plans in PSpace, the CQs in preconditions and the goal can be evaluated in 2ExpTime (Ortiz, Rudolph, and Šimkus 2011) and the remaining parts of the ECQs can be evaluated in PSpace (Abiteboul, Hull, and Vianu 1995). □

While these complexity results already indicate that reasoning in Horn-$\mathcal{SROIQ}$ is more powerful than (polynomial) planning in PDDL with derived predicates, they tell us nothing about the relative expressivity of these two formalisms. There could still exist a polynomial-size compilation scheme from the former to the latter that preserves plan size polynomially, because the compilation can use arbitrary computational resources as long as the result is of polynomial size.

To prove that such a compilation indeed cannot exist, we follow Thiébaux, Hoffmann, and Nebel (2005) by using the notion of *advice-taking* Turing machines (Karp and Lipton 1982). Such machines are equipped with an advice oracle a, which is a function from positive integers to bit strings. On input w, the machine receives the *advice* $a(\|w\|)$ and then starts its computation as usual. The advice depends only on the length of the input, but not on its contents. An advice oracle is *polynomial* if the length of $a(\|w\|)$ is bounded polynomially in $\|w\|$. ExpTime/poly (*non-uniform* ExpTime) is the class of problems that can be decided by Turing machines with polynomial advice and exponential time bound.

The following result shows that a polynomial compilation scheme from Horn-$\mathcal{SROIQ}$ eKABs to PDDL would imply that the weak exponential hierarchy collapses completely. The latter is considered to be unlikely; in particular, it would mean that one can eliminate any bounded quantifier prefix in second-order logic and Presburger arithmetic (Gottlob, Leone, and Veith 1995; Haase 2014).

Theorem 5. *Unless* $\textsc{ExpTime}^{\mathrm{NP}} = \textsc{ExpTime}$*, there is no polynomial compilation scheme from Horn-$\mathcal{SROIQ}$ eKABs to PDDL preserving plan size polynomially.*

Proof sketch. Let M be a universal Turing machine with double-exponential time bound that can simulate all other such TMs. In the technical report (Borgwardt et al. 2022), we show how to construct a family of Horn-$\mathcal{SROIQ}$ eKAB domain descriptions Δ_n such that M accepts a word w of length n iff $(\Delta_n, \mathcal{O}, I_w, g)$ has a plan of length 1. Here, $\mathcal{O}$ contains only the two objects o and e, I_w is a state that can be computed from w in polynomial time, and g is a nullary predicate. This construction is based on the 2ExpTime-hardness proof for Horn-$\mathcal{SROIQ}$ (Ortiz, Rudolph, and Šimkus 2010).

Assume now that there is a compilation scheme $\mathbf{f} = (f_\delta, f_o, f_i, f_g)$ from Horn-$\mathcal{SROIQ}$ eKABs to PDDL preserving plan size polynomially. This scheme could be used as an advice oracle as follows. Let M' be a TM with double-exponential time bound. Then M' accepts w' iff M accepts $w = M'\#w'$ (in some fixed encoding). Let n be the size of w. The compilation of Δ_n to a PDDL domain description $\Delta'_n = f_\delta(\Delta_n)$ as well as $f_o(\Delta_n)$ can be used as polynomial advice for a Turing machine that, on input w, computes $\mathcal{O} = \{o, e\}$, I_w, and $(\Delta'_n, \mathcal{O} \cup f_o(\Delta_n), f_i(\mathcal{O}, I_w), f_g(\mathcal{O}, g))$, which can be done in polynomial time. It then decides polynomial-step plan existence for this PDDL task, which can be done in ExpTime by Theorem 3 and is equivalent to deciding whether M' accepts w' by Definition 5. Overall, this implies that $\text{ExpTime}^{\text{NP}} \subseteq$ 2ExpTime is included in ExpTime/poly, and therefore $\text{ExpTime}^{\text{NP}} = \text{ExpTime}$ (Buhrman and Homer 1992), which contradicts the assumption of the theorem. □

Corollary 1. *Unless* $\text{ExpTime}^{\text{NP}} = \text{ExpTime}$*, in general there can be no polynomial Datalog*$^\neg$*-rewritings for IQs over Horn-*$\mathcal{SROIQ}$ *TBoxes.*

Proof. By Theorem 1, such a rewriting would yield a polynomial compilation from Horn-$\mathcal{SROIQ}$ eKABs to PDDL preserving plan size exactly, which contradicts the assumption by Theorem 5. □

We obtain a similar result also for the non-Horn DLs $\mathcal{SH}$ and $\mathcal{ALCI}$, because for them similar 2ExpTime-hardness proofs can be adapted (Lutz 2008; Eiter et al. 2009).

Theorem 6. *Unless* $\text{ExpTime}^{\text{NP}} = \text{ExpTime}$*, there is no polynomial compilation scheme from* $\mathcal{SH}$ *or* $\mathcal{ALCI}$ *eKABs to PDDL preserving plan size polynomially.*

Experiments

While polynomial compilations are nice in theory, they have one major drawback: the size of the rules and in particular the arity of the new predicates grows polynomially with the input (Carral and Krötzsch 2020). In contrast, the existing exponential compilation from Horn-$\mathcal{SHIQ}$ to Datalog uses rules of constant size and in many cases does not exhibit an exponential blowup (Eiter et al. 2012). Moreover, from a pragmatic perspective, it is the only Datalog rewriting for CQs over Horn-DLs that has been implemented so far, in the Clipper system. We thus implemented our compilation from the *Compiling TBoxes into Derived Predicates* section using Clipper to answer the following questions: 1) Is the compilation feasible, i.e. can the generated classical planning tasks be handled by state-of-the-art planners? 2) How does our compilation perform against existing eKAB compilations?

For the experiments, we use the Fast Downward (FD) planning system (Helmert 2006) version 20.06 (the newest version as of August 2021), the main implementation platform for classical planning today. We ran FD with a dual-queue greedy best-first search using the h^{FF} heuristic, a commonly used baseline in the planning literature. All experiments were run on a computer with an Intel Core i5-4590 CPU@3.30GHz, and run time and memory cutoffs of 600s and 8GBs, respectively. The compiler and the benchmarks with a detailed description are available online.[4]

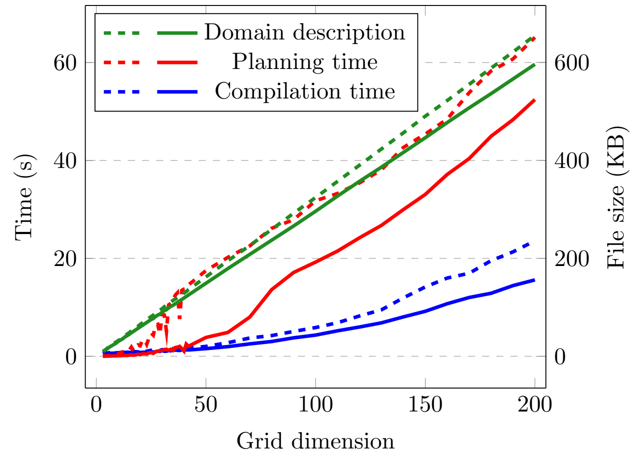

Figure 1: Comparison of the Robot (dashed) and RobotConj (solid) domains w.r.t. domain description size, compilation and planning times.

Implementation. We encode Horn-$\mathcal{SHIQ}$ eKAB tasks as ontology files accompanied by PDDL files whose syntax has been extended to allow conjunctive queries. The ontology file uses Turtle syntax, which can be processed by any off-the-shelf ontology tool. Our compiler reads the CQ-PDDL and ontology files, and generates a classical planning task in standard PDDL format. The Datalog rewriting is generated with Clipper (Eiter et al. 2012). The compilation additionally normalizes complex conditions via a Tseitin-like transformation (Tseitin 1983), which has been shown to be effective before (Borgwardt et al. 2021).

Benchmarks. Our benchmark collection consists of 125 instances adapted from existing DL-Lite eKAB benchmarks (Calvanese et al. 2016; Borgwardt et al. 2021), and 110 newly created instances. We manually translated the existing eKAB domains (Cats, Elevator, Robot, TaskAssign, TPSA, VTA, and VTA-Roles) into the format described above. Modifications almost exclusively pertained to extracting the ontology from the eKAB description into a separate Turtle file and moving so-called condition-action rules (Calvanese et al. 2016) into action preconditions. The translated instances are equivalent to the originals.

Since Horn-$\mathcal{SHIQ}$ is more expressive than DL-Lite, we also created 3 new domains (Drones, Queens, and RobotConj) in which we make use of conjunctions, qualified existential restrictions occurring with negative polarity, and symmetric and transitive relations, all of which are not supported by DL-Lite (Baader et al. 2017). Drones describes a complex 2D drone navigation problem, in which drones need to be moved to avoid critical situations; the latter are described in the ontology using axioms with qualified existential restrictions and symmetric relations. Queens generalizes the eight queens

[4]https://gitlab.perspicuous-computing.science/a.kovtunova/pddl-horndl

Domain	#	# solved Cal16	Bor21	Horn	# compiled Cal16	Bor21	Horn	planning time Cal16	Bor21	Horn	compilation time Cal16	Bor21	Horn
Cats	20	14	**20**	**20**	20	20	20	63.46	0.13	**0.03**	**0.16**	0.22	0.65
Elevator	20	20	20	20	20	20	20	0.36	0.30	**0.03**	**0.60**	0.73	0.66
Robot	20	4	12	**20**	12	12	**20**	15.05	10.10	**0.11**	138.52	138.55	**0.75**
TaskAssign	20	3	**20**	**20**	20	20	20	0.81	0.12	**0.06**	2.87	39.21	**0.66**
TPSA	15	14	5	**15**	**15**	5	**15**	2.01	2.42	**0.30**	0.84	25.37	**0.59**
VTA	15	**15**	13	**15**	15	15	15	23.06	371.56	**16.91**	**0.33**	1.21	0.65
VTA-Roles	15	**15**	5	**15**	**15**	5	**15**	2.25	11.61	**1.36**	**0.59**	95.53	0.66
$\sum$	125	85	95	**125**	117	97	**125**	19.99	77.33	**3.59**	18.01	31.84	**0.66**
Drones	24			20			24			101.42			0.69
Queens	30			15			30			21.66			0.69
RobotConj	56			56			56			8.14			2.77
$\sum$	110			91			110			30.87			1.75

Table 1: Per-domain aggregated statistics: "# solved" number of instances solved by the planner; "# compiled" number of instances for which the compilers could generate the PDDL input files for the planner; "planning time" average planner run time over the commonly solved instances; "compilation time" average time of generating the PDDL files.

puzzle to board sizes $n \in \{5, \ldots, 10\}$ and numbers of queens $m \in \{n-4, \ldots, n\}$. Queens are initially placed randomly on the board and need to be moved to a configuration where no queen threatens another. The ontology contains a symmetric, transitive relation to describe legal moves. RobotConj is a redesign of Robot that moves some of the complexity from actions into the ontology. The original Robot benchmark encodes static knowledge about 2D grid cell adjacency in the action descriptions, which can be encoded much more naturally in the ontology using conjunctions. Note that the original Robot benchmark consists of 20 instances (grid sizes 3×3 up to 22×22), whereas for the new RobotConj we included 56 instances (up to 200×200) since they could be easily handled by our compiler.

Scalability Study. We use Robot and RobotConj to analyze how our compilation performs as a function of domain description size (including the ontology). Figure 1 depicts the results for 56 instances of each domain, obtained by scaling the grid from 3×3 to 200×200. In both domains, the file size is directly proportional to size of the grid. Even the largest tested instance could be compiled and solved in less than 90 seconds, attesting the feasibility of our approach. The increased complexity of RobotConj's ontology does not affect the performance. On the contrary, both compilation and planning for RobotConj are actually consistently faster than for Robot, due to the simplified actions.

Comparison to DL-Lite Compilations. We compare to Cal16, the original DL-Lite eKAB compiler (Calvanese et al. 2016), and to Bor21, its recently introduced optimization using derived predicates to compile away complex formulas (Borgwardt et al. 2021). We refer to our compilation by Horn. Table 1 gives a summary of the results. Cal16 and Bor21 were only run on the original DL-Lite eKAB benchmarks.

Considering the DL-Lite benchmark part, Horn has similar or better performance than Cal16 and Bor21. In Robot, the Cal16 compilation (and hence also Bor21) could only process the 12 smallest instances (up to grid size 14×14), exceeding the 600 seconds time limit thereafter. While Cal16 and Bor21 both show a blow-up in compilation time in some domains, our new Horn compiler could process all instances in less than 1 second on average. The compilation time of Horn is almost consistently larger than 0.6 seconds, which can be attributed to the fixed overhead of calling CLIPPER. While the compilation time of Horn is very competitive with the previous DL-Lite compilers, the planner's performance statistics really substantiate this advantage. Regarding planning time and the number of solved instances, Horn significantly outperforms both alternatives on the DL-Lite benchmarks.

The more complex ontologies in the Horn-$\mathcal{SHIQ}$ benchmark part did not pose a challenge to Horn. All instances could still be translated within 3 seconds on average. The constructed instances of Drones and Queens are however much more challenging from a planning perspective. Contrary to the DL-Lite benchmarks before, average planning runtime is higher, and some instances could not be solved in time. The difficulty of the instances was chosen intentionally, with the purpose of creating challenging problems for future work.

Conclusion

We have shown that adding Horn-DL background ontologies often does not increase the expressivity of PDDL planning tasks. This is due to Datalog$^{\neg}$-rewritability, which allows us to reduce open-world to closed-world reasoning. However, adding more axiom types (Horn-$\mathcal{SROIQ}$) or using non-Horn DLs ($\mathcal{SH}$ or $\mathcal{ALCI}$) increases the expressivity beyond PDDL, unless the weak exponential hierarchy collapses. An evaluation of our generic compilation approach using the CLIPPER system demonstrates the feasibility of using Datalog$^{\neg}$-rewritings, even compared to more specialized compilation schemes for the smaller logic DL-Lite. Moreover, we have contributed additional benchmarks to showcase the increased expressivity of our proposed approach.

In future work, we will investigate the existence of a polynomial compilation scheme for Horn-$\mathcal{SHOIQ}$, whose expressivity lies between that of Horn-$\mathcal{ALCHOIQ}$ and Horn-$\mathcal{SROIQ}$. We also want to investigate planning formalisms with different effect semantics, e.g. the one described by De Giacomo et al. (2021).

Acknowledgements

This work is supported by DFG grant 389792660 as part of TRR 248 – CPEC (https://perspicuous-computing.science).

References

Abiteboul, S.; Hull, R.; and Vianu, V. 1995. *Foundations of Databases*. Addison-Wesley.

Ahmetaj, S.; Calvanese, D.; Ortiz, M.; and Šimkus, M. 2017. Managing Change in Graph-Structured Data Using Description Logics. *ACM Transactions on Computational Logic*, 18(4): 27:1–27:35.

Baader, F.; Horrocks, I.; Lutz, C.; and Sattler, U. 2017. *An Introduction to Description Logic*. Cambridge University Press.

Bienvenu, M.; and Ortiz, M. 2015. Ontology-Mediated Query Answering with Data-Tractable Description Logics. In Faber, W.; and Paschke, A., eds., *Reasoning Web. 11th Int. Summer School*, volume 9203 of *Lecture Notes in Computer Science*, 218–307. Springer.

Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; Krötzsch, M.; Nebel, B.; and Steinmetz, M. 2022. Expressivity of Planning with Horn Description Logic Ontologies (Technical Report). arXiv:2203.09361.

Borgwardt, S.; Hoffmann, J.; Kovtunova, A.; and Steinmetz, M. 2021. Making DL-Lite Planning Practical. In Bienvenu, M.; and Lakemeyer, G., eds., *Proc. of the 18th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'21)*, 641–645. IJCAI.

Buhrman, H.; and Homer, S. 1992. Superpolynomial Circuits, Almost Sparse Oracles and the Exponential Hierarchy. In Shyamasundar, R., ed., *Proc. of the 12th Conf. on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'92)*, volume 652 of *Lecture Notes in Computer Science*, 116–127. Springer-Verlag.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007a. EQL-Lite: Effective First-Order Query Processing in Description Logics. In Veloso, M. M., ed., *20th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 274–279.

Calvanese, D.; De Giacomo, G.; Lembo, D.; Lenzerini, M.; and Rosati, R. 2007b. Tractable Reasoning and Efficient Query Answering in Description Logics: The *DL-Lite* Family. *Journal of Automated Reasoning*, 39(3): 385–429.

Calvanese, D.; De Giacomo, G.; Montali, M.; and Patrizi, F. 2013. Verification and Synthesis in Description Logic Based Dynamic Systems. In Faber, W.; and Lembo, D., eds., *7th Int. Conf. Web Reasoning and Rule Systems (RR)*, 50–64. Springer.

Calvanese, D.; Montali, M.; Patrizi, F.; and Stawowy, M. 2016. Plan Synthesis for Knowledge and Action Bases. In Kambhampati, S., ed., *25th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 1022–1029. AAAI Press.

Carral, D.; Dragoste, I.; and Krötzsch, M. 2018. The Combined Approach to Query Answering in Horn-$\mathcal{ALCHOIQ}$. In Thielscher, M.; Toni, F.; and Wolter, F., eds., *Proc. of the 16th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'18)*, 339–348. AAAI Press.

Carral, D.; and Krötzsch, M. 2020. Rewriting the Description Logic $\mathcal{ALCHIQ}$ to Disjunctive Existential Rules. In Bessiere, C., ed., *Proc. of the 29th Int. Joint Conf. on Artificial Intelligence and the 17th Pacific Rim Int. Conf. on Artificial Intelligence (IJCAI-PRICAI'20)*, 1777–1783. IJCAI.

Dantsin, E.; Eiter, T.; Gottlob, G.; and Voronkov, A. 2001. Complexity and Expressive Power of Logic Programming. *ACM Computing Surveys*, 33(3): 374–425.

De Giacomo, G.; Lespérance, Y.; Patrizi, F.; and Vassos, S. 2014. Progression and Verification of Situation Calculus Agents with Bounded Beliefs. In amd Paul Scerri, A. L.; Bazzan, A.; and Huhns, M., eds., *13th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, 141–148. ACM.

De Giacomo, G.; Oriol, X.; Rosati, R.; and Savo, D. F. 2021. Instance-Level Update in DL-Lite Ontologies through First-Order Rewriting. *Journal of Artificial Intelligence Research*, 70: 1335–1371.

Eiter, T.; Lutz, C.; Ortiz, M.; and Šimkus, M. 2009. Query Answering in Description Logics with Transitive Roles. In Boutilier, C., ed., *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI'09)*, 759–764. AAAI Press.

Eiter, T.; Ortiz, M.; Šimkus, M.; Tran, T.-K.; and Xiao, G. 2012. Query Rewriting for Horn-$\mathcal{SHIQ}$ Plus Rules. In Hoffmann, J.; and Selman, B., eds., *Proc. of the 26th AAAI Conf. on Artificial Intelligence (AAAI'12)*, 726–733. AAAI Press.

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*, 20: 61–124.

Gaggl, S. A.; Rudolph, S.; and Schweizer, L. 2016. Fixed-Domain Reasoning for Description Logics. In Kaminka, G. A.; Fox, M.; Bouquet, P.; Hüllermeier, E.; Dignum, V.; Dignum, F.; and van Harmelen, F., eds., *Proc. of the 22nd Eur. Conf. on Artificial Intelligence (ECAI'16)*, volume 285 of *Frontiers in Artificial Intelligence and Applications*, 819–827. IOS Press.

Gerevini, A.; Haslum, P.; Long, D.; Saetti, A.; and Dimopoulos, Y. 2009. Deterministic planning in the fifth international planning competition: PDDL3 and experimental evaluation of the planners. *Artificial Intelligence*, 173(5-6): 619–668.

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning: Theory and Practice*. Morgan Kaufmann.

Gottlob, G.; Leone, N.; and Veith, H. 1995. Second Order Logic and the Weak Exponential Hierarchies. In Wiedermann, J.; and Hájek, P., eds., *Proc. of the 20th Int. Symp. on Mathematical Foundations of Computer Science (MFCS'95)*, volume 969 of *Lecture Notes in Computer Science*, 66–81. Springer-Verlag.

Haase, C. 2014. Subclasses of Presburger Arithmetic and the Weak EXP Hierarchy. In Henzinger, T.; and Miller, D., eds., *Proc. of the Joint Meeting of the 23rd EACSL Annual Conf. on Computer Science Logic (CSL) and the 29th Annual ACM/IEEE Symp. on Logic in Computer Science (LICS)*, 14:1–14:10. ACM.

Haslum, P.; Lipovetzky, N.; Magazzeni, D.; and Muise, C. 2019. An introduction to the planning domain definition language. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 13(2): 1–187.

Helmert, M. 2006. The Fast Downward Planning System. *Journal of Artificial Intelligence Research*, 26: 191–246.

Hoffmann, J.; and Edelkamp, S. 2005. The Deterministic Part of IPC-4: An Overview. *Journal of Artificial Intelligence Research*, 24: 519–579.

Jung, J. C.; Papacchini, F.; Wolter, F.; and Zakharyaschev, M. 2019. Model Comparison Games for Horn Description Logics. In Bouyer, P., ed., *Proc. of the 34th Annual IEEE Symp. on Logic in Computer Science (LICS'19)*, 1–14. IEEE.

Karp, R. M.; and Lipton, R. J. 1982. Turing Machines that Take Advice. *L'Enseignement Mathématique*, 28: 191–209.

Krötzsch, M. 2011. Efficient Rule-Based Inferencing for OWL EL. In Walsh, T., ed., *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, 2668–2773. AAAI Press.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2013. Complexities of Horn Description Logics. *ACM Transactions on Computational Logic*, 14(1): 2:1–2:36.

Lutz, C. 2008. The Complexity of Conjunctive Query Answering in Expressive Description Logics. In *Proc. of the 4th Int. Joint Conf. on Automated Reasoning (IJCAR'08)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, 179–193. Springer-Verlag.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. *The PDDL Planning Domain Definition Language*. The AIPS-98 Planning Competition Comitee.

Nebel, B. 2000. On the Compilability and Expressive Power of Propositional Planning Formalisms. *Journal of Artificial Intelligence Research*, 12: 271–315.

Ortiz, M.; Rudolph, S.; and Šimkus, M. 2010. Worst-case Optimal Reasoning for the Horn-DL Fragments of OWL 1 and 2. In Lin, F.; Sattler, U.; and Truszczynski, M., eds., *Proc. of the 12th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'10)*, 269–279. AAAI Press.

Ortiz, M.; Rudolph, S.; and Šimkus, M. 2011. Query Answering in the Horn Fragments of the Description Logics $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. In Walsh, T., ed., *Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI'11)*, 1039–1044. AAAI Press.

Thiébaux, S.; Hoffmann, J.; and Nebel, B. 2005. In Defense of PDDL Axioms. *Artificial Intelligence*, 168: 38–69.

Tseitin, G. S. 1983. On the Complexity of Derivation in Propositional Calculus. In Siekmann, J. H.; and Wrightson, G., eds., *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, 466–483. Springer Berlin Heidelberg. ISBN 978-3-642-81955-1.