Cognitive Robotics
AAAI Technical Report WS-12-06

Towards an Expressive Decidable Logical Action Theory

Wael Yehia
Department of Computer Science
York University
4700 Keele Street
Toronto, ON, M3J 1P3, Canada
Email: w2yehia@cse.yorku.ca

Mikhail Soutchanski
Department of Computer Science
Ryerson University
245 Church Street, ENG281
Toronto, ON, M5B 2K3, Canada
Email: mes@scs.ryerson.ca

Abstract

In the area of reasoning about actions, one of the key computational problems is the projection problem: to find whether a given logical formula is true after performing a sequence of actions. This problem is undecidable in the general situation calculus; however, it is decidable in some fragments. We consider a fragment P of the situation calculus and Reiter's basic action theories (BAT) such that the projection problem can be reduced to the satisfiability problem in an expressive description logic $\mathcal{ALCO}(\mathcal{U})$ that includes nominals (O), the universal role (U), and constructs from the well-known logic $\mathcal{ALC}$. It turns out that our fragment P is more expressive than previously explored description logic based fragments of the situation calculus. We explore some of the logical properties of our theories. In particular, we show that the projection problem can be solved using regression in the case where BATs include a general "static" TBox, i.e., an ontology that has no occurrences of fluents. Thus, we propose seamless integration of traditional ontologies with reasoning about actions. We also show that the projection problem can be solved using progression if all actions have only local effects on the fluents, i.e., in P, if one starts with an incomplete initial theory that can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept, then its progression resulting from execution of a ground action can still be expressed in the same language. Moreover, we show that for a broad class of incomplete initial theories progression can be computed efficiently.

1 Introduction

The projection problem is an important reasoning task in AI. It is a prerequisite to solving other computational problems including planning and high-level program execution. Informally, the projection problem consists in finding whether a given logical formula is true in a state that results from a sequence of transitions, when knowledge about an initial state is incomplete. In description logics (DLs) and earlier terminological systems, this problem was formulated using roles to represent transitions and concept expressions to represent states. This line of research as well as earlier applications of DLs to planning and plan recognition are discussed and reviewed in (Devanbu and Litman 1996). Using a somewhat related approach, the projection problem and a solution to the related frame problem (i.e., how to provide a concise axiomatization of non-effects of actions) have been explored using propositional dynamic logic, e.g., see (De Giacomo and Lenzerini 1995; De Giacomo et al. 1999) These papers discuss relations with the propositional fragment of the situation calculus and review previous work. A more recent work explores decidable combinations of several modal logics, or combining description logics with a modal logic of time or with a propositional dynamic logic (Artale and Franconi 2000; Wolter and Zakharyaschev 1998; Chang et al. 2012) The resulting logics are somewhat limited in terms of expressivity because to guarantee the decidability of the satisfiability problem in the combined logic, only atomic actions can be allowed. In applications, it is sometimes convenient to consider actions with arbitrary many arguments.

On the other hand, there are several proposals regarding the integration of DLs and reasoning about actions (Lutz and Sattler 2002; Baader et al. 2005; Liu et al. 2006; Calvanese et al. 2007; Gu and Soutchanski 2010). In (Gu and Soutchanski 2010), it is shown that the projection problem is decidable in a proposed fragment of the situation calculus (SC). However, the logical languages developed in these papers are not expressive enough to represent some of the action theories popular in AI or to solve the projection problem in a general case. For example, Gu& Soutchanski propose a DL based situation calculus (Gu and Soutchanski 2010), where the projection problem is reduced to the satisfiability problem in $\mathcal{ALCO}(\mathcal{U})$, a DL that adds nominals O and the universal (global) role U to the well known description logic $\mathcal{ALC}$. The universal role links any two individuals in the domain; it is introduced to add the usual unguarded $\forall$- and $\exists$-quantifiers which are handy to represent incomplete knowledge about an initial state and about conditional effects. They consider Reiter's basic action theories (BATs) (Reiter 2001), but impose syntactic constraints on the formulas that can appear in axioms by concentrating on a subset FO_{DL} of FO^2 formulas, where FO^2 is a fragment of first order logic (FOL) with only two variables. In the fragment of SC that they consider, action functions may have

Copyright © 2012, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

at most two object arguments, the formulas in the precondition axioms (PA) and context formulas in the successor state axioms (SSA) should be FO_{DL} formulas (if the situation argument is suppressed), where FO_{DL} formulas are those FO^2 formulas, which can be translated into a concept in $\mathcal{ALCO}(\mathcal{U})$ using the standard translation between DLs and fragments of FOL. They illustrate their proposal with several realistic examples of dynamic domains, but it turns out that some of the well-known examples, e.g., the Logistics domain from the first International Planning Competition (IPC) (McDermott 2000), cannot be represented due to syntactic restrictions on the language they consider. Here and subsequently, when we mention planning domain specifications, we consider them as FOL theories without making the Domain Closure Assumption (DCA) common in planning, i.e., without reducing them to purely propositional level. Later, (Gu 2010) introduces a possible extension, where the syntactic restrictions on the class of formulas FO_{DL} are relaxed, but stipulates SSAs for dynamic roles (fluents with two object arguments and one situational argument) to be context-free. She conjectures, but does not prove, that the projection problem in that extension can be reduced to satisfiability in $\mathcal{ALCO}(\mathcal{U})$.

In our paper, we consider an even more expressive fragment of SC, called $\mathcal{P}$, where all SSAs can be context dependent with context conditions formulated in a language $\mathcal{L}$ that includes FO_{DL} as a proper fragment. Manual translations of planning specifications (from IPC) into our language $\mathcal{P}$ show that $\mathcal{P}$ has expressive power sufficient to represent not only Blocks World and Logistics, but also many other popular benchmarks (Kudashkina 2011; Yehia 2012). In any case, reducing projection to satisfiability in $\mathcal{ALCO}(\mathcal{U})$ is justified by the fact that there are several off-the-shelf OWL2 reasoners that can be employed to solve the latter problem, since a DL $\mathcal{SROIQ}$ underlying the Web Ontology Language (OWL2) includes $\mathcal{ALCO}(\mathcal{U})$ as a fragment (Cuenca Grau et al. 2008). In our paper, we concentrate on foundational work and explore the logical properties of $\mathcal{P}$.

Our paper contributes to Cognitive Robotics and to reasoning about actions by formulating an expressive fragment of SC where the projection problem is decidable without the domain closure assumption (DCA) and closed world assumption (CWA), i.e., when an initial theory is incomplete and is not purely propositional.

2 Definition of $\mathcal{P}$

We assume that the reader is familiar with SC from (Pirri and Reiter 1999; Reiter 2001) and knows that a BAT $\mathcal{D} = \mathcal{D}_{AP} \cup \mathcal{D}_{SS} \cup UNA \cup \mathcal{D}_{S_0} \cup \Sigma$ consists of the precondition axioms (PAs) $\mathcal{D}_{AP}$, that use the binary predicate symbol $Poss$, successor state axioms (SSAs) $\mathcal{D}_{SS}$, a set of unique name axioms UNA, an initial theory $\mathcal{D}_{S_0}$ that specifies an incomplete theory of the initial situation S_0, and Σ - a set of domain independent foundational axioms about the relation $s_1 \preceq s_2$ of precedence between situations s_1 and s_2. In (Reiter 2001), axioms Σ are formulated in second-order logic, all other axioms are in many-sorted FOL, so we assume the usual definitions of sorts, terms, well-formed formulas, and so on. A fluent is a predicate with the last argument s of sort situation. As usual, we say that a situation calculus FOL formula $\psi(s)$ is *uniform* in s, if s is the only situation term mentioned in $\psi(s)$, the formula ψ has no occurrences of the predicates $Poss, \prec$, and has no quantifiers over variables of sort situation. The formula ψ obtained by deleting all arguments s from fluents in the formula $\psi(s)$ uniform in s is called the formula with *suppressed* situation argument; the interested reader can find details in (Pirri and Reiter 1999).

Fluents with a single object argument, $F(x, s)$, are called *dynamic concepts*, and fluents with two object arguments, $F(x, y, s)$, are called *dynamic roles*. In the signature of a BAT $\mathcal{D}$, any predicate that is not a fluent must have either one or two arguments, and is called either a *(static) concept*, or a *(static) role*, respectively. Subsequently, we consider only BATs with relational fluents, and do not allow any other function symbols except $do(a, s)$ and action functions. In particular, terms of sort object can be only constants or variables. Actions may have any number of object arguments.

To specify syntactic constraints on $\mathcal{D}_{ap}$ and $\mathcal{D}_{ssa}$, we consider a language $\mathcal{L}$, that has at most $n + 2$ object variables $x, y, z_1, \ldots, z_n$, for some integer $n > 0$. We assume $\mathcal{L}$ has at least n constants $b_i, 1 \leq i \leq n$. The purpose of the variables z_i is to serve as place-holders to be instantiated with constants b_i that occur as named object arguments of ground action terms. This language $\mathcal{L}$ consists of two related sets of formulas: $\mathbf{F}^x$ and $\mathbf{F}^y$. Formulas $\phi(x)$ from the set $\mathbf{F}^x$ can have as free variables either x, or some of the place-holder variables z_i, $1 \leq i \leq n$, but cannot have free occurrences of y. Formulas $\phi(y)$ from the set $\mathbf{F}^y$ can have free occurrences of either y, or some of the place holders z_i, $1 \leq i \leq n$, but cannot have free occurrences of x. Note the formulas ϕ may have free variables z_i that are not shown explicitly, but it will be always clear from the context which variables are free in the formulas. We use the symbol $\widetilde{\cdot}$ to denote a bijection between $\mathbf{F}^x$ and $\mathbf{F}^y$. If $\phi(x) \in \mathbf{F}^x$, then $\widetilde{\phi}(y)$ is the *dual* formula of $\phi(x)$, obtained by renaming in $\phi(x)$ every occurrence of x (both free and bound) with y and every bound occurrence of y with x. Similarly, if $\phi(y) \in \mathbf{F}^y$, then $\widetilde{\phi}(x)$ is the *dual* formula to $\phi(y)$ obtained by replacing every occurrence of y with variable x, and every bound occurrence of x with y. The sets $\mathbf{F}^x$ and $\mathbf{F}^y$ have a non-empty intersection. For example, sentences that mention constants only, and $\mathbf{F}^x$ formulas that have only occurrences of z variables belong to both $\mathbf{F}^x$ and to $\mathbf{F}^y$. Each formula ϕ without x, y variables is mapped by bijection $\widetilde{\phi}$ to itself. We are ready to give the following inductive definition.

Definition 1 *Let $\mathcal{L}$ be the set of first-order logic formulas such that $\mathcal{L} = \mathbf{F}^x \cup \mathbf{F}^y$, and $\widetilde{\cdot}$ be a bijection between formulas in $\mathbf{F}^x$ and $\mathbf{F}^y$ as defined above, where the sets $\mathbf{F}^y$ and $\mathbf{F}^x$ are minimal sets constructed as follows. (We focus on $\mathbf{F}^x$, since $\mathbf{F}^y$ is similar.)*

1. *$\top$ and $\bot$ are in $\mathbf{F}^x$.*
2. *If AC is a unary predicate symbol, z is a variable distinct from x and y, and b is a constant, then the formulas $AC(x)$, $AC(z)$, and $AC(b)$ are in $\mathbf{F}^x$.*

3. *If b is a constant, and z is a variable that is distinct from x and y, then the formulas $x = x$, $x = b$, $x = z$ are in $\boldsymbol{F}^x$.*
4. *If R is a binary predicate symbol, b_1 and b_2 are constants, and z_1 and z_2 are variables that are distinct from x and y, then $R(z_1, z_2)$, $R(b_1, b_2)$, $R(b_1, z_2)$, $R(z_1, b_2)$, $R(x, b_2)$ and $R(x, z_2)$ are formulas in $\boldsymbol{F}^x$.*
5. *If $\phi \in \boldsymbol{F}^x$, then also $\neg\phi \in \boldsymbol{F}^x$.*
6. *If $\phi, \psi \in \boldsymbol{F}^x$, then both $(\phi \wedge \psi) \in \boldsymbol{F}^x$ and $(\phi \vee \psi) \in \boldsymbol{F}^x$.*
7. *If $\phi(x) \in \boldsymbol{F}^x$, R is a binary predicate symbol, b is any constant, z is any variable distinct from x and y, and $\widetilde{\phi}(y)$ is the formula dual to $\phi(x)$, then all of the following formulas with quantifiers guarded by R belong to $\boldsymbol{F}^x$: $\exists y.R(x, y) \wedge \widetilde{\phi}(y)$, $\exists y.R(b, y) \wedge \widetilde{\phi}(y)$, $\exists y.R(z, y) \wedge \widetilde{\phi}(y)$, as well as $\forall y.R(x, y) \supset \widetilde{\phi}(y)$, $\forall y.R(b, y) \supset \widetilde{\phi}(y)$, $\forall y.R(z, y) \supset \widetilde{\phi}(y)$.*
8. *If $\phi \in \boldsymbol{F}^x$, $\widetilde{\phi}$ is the formula dual to ϕ, then $[\exists x].\phi(x)$, $[\forall x].\phi(x)$ as well as $[\exists y.]\widetilde{\phi}(y)$, $[\forall y.]\widetilde{\phi}(y)$ belong to $\boldsymbol{F}^x$, where $[\exists]$ ($[\forall]$, respectively) means quantifiers are optional and applied only when a formula has a free variable.*

The intuition behind the definition of $\mathcal{L}$ is that any variable z other than x and y has to be free in a formula from $\mathcal{L}$. The set of formulas $FO_{DL} = FO^x_{DL} \cup FO^y_{DL}$ defined in (Gu and Soutchanski 2010) is a proper subset of $\mathcal{L}$ because the set of formulas FO^x_{DL} (FO^y_{DL}, respectively) is a proper subset of $\mathbf{F}^x$ ($\mathbf{F}^y$, respectively): no place holder variables $z_1, \ldots, z_n$ are allowed in FO^x_{DL} and FO^y_{DL}. We say a formula $\phi \in \mathcal{L}$ is a *z-free $\mathcal{L}$ formula*, if all occurrences of variables z (if any), other than x and y, in ϕ are instantiated with constants.

Lemma 1 *There are syntactic translations between the set of z-free formulas $\phi \in \mathcal{L}$ and the concept expressions from the language $\mathcal{ALCO}(\mathcal{U})$ in both directions, i.e., they are equally expressive. Moreover, such translations lead to no more than a linear increase in the size of the formula.*

This lemma is proved using the standard translation between DLs and FOL; the proof is similar to the proof of Lemma 1 in (Gu and Soutchanski 2010). Using the fluents $Loaded(box, s)$, $At(box, city, s)$, and $In(box, vehicle, s)$ from Logistics as an example, after suppressing s, a z-free $\mathcal{L}$ formula $Loaded(B_1) \vee \exists x(Box(x) \wedge x \neq B_1 \wedge In(x, T_1))$ is translated as $\exists U.(\{B_1\} \sqcap Loaded) \sqcup \exists U.(Box \sqcap \neg\{B_1\} \sqcap \exists In.\{T_1\}$, where $\{B_1\}, \{T_1\}$ are nominals (i.e., concepts interpreted as singleton sets), and $\forall x(\neg Box(x) \vee x = B_1 \vee At(x, Toronto))$, all boxes distinct from B_1 are in Toronto, is translated as $\forall U.(\neg Box \sqcup \{B_1\} \sqcup \exists At.\{Toronto\})$. Notice why nominals and U are important. Subsequently, we consider BATs that use in axioms $\mathcal{L}$-like formulas uniform in s. This motivates the following requirements. For brevity, let a vector $\tilde{\mathbf{x}}$ of object variables denote either x, or y, or $\langle x, y\rangle$; also, let $\tilde{\mathbf{z}}$ denote a vector of place holder variables.
Action precondition axioms $\mathcal{D}_{AP}$: For each action function $A(\vec{z})$, there is a single precondition axiom uniform in s:

$$(\forall \vec{z}, s).\ Poss(A(\vec{z}), s) \equiv \Pi_A(\vec{z})[s], \qquad (1)$$

where $\Pi_A(\vec{z}, s)$ is uniform in s; it is an $\mathcal{L}$ formula with $\vec{z}$ as the only free variables, if any, when s is suppressed. When object arguments of $A(\vec{z})$ are instantiated with constants, by Lemma 1, the RHS of each precondition axiom can be translated into a concept in $\mathcal{ALCO}(\mathcal{U})$, when s is suppressed.
Successor state axioms $\mathcal{D}_{SS}$: There is a single SSA for each fluent $F(\vec{x}, do(a, s))$. According to the general syntactic form of the SSAs provided in (Reiter 2001), without loss of generality, we can assume that each axiom is as follows:

$$F(\vec{x}, do(a,s)) \equiv \gamma^+_F(\vec{x}, a,s) \vee F(\vec{x}, s) \wedge \neg\gamma^-_F(\vec{x}, a,s) \qquad (2)$$

where each of the γ_F's are disjunctions either of the form
$[\exists \vec{z}].a = A(\vec{u}) \wedge \phi(x, \vec{z}, s)$,
/* a set of variables $\vec{z} \subseteq \vec{u}$; may be $\{x\} \in \vec{u}$ */
if (2) is a SSA for a dynamic concept $F(x, s)$ with a single object argument x, or
$[\exists \vec{z}].a = A(\vec{u}) \wedge \phi(x, \vec{z}, s) \wedge \phi(y, \vec{z}, s)$,
/* variables $\vec{z} \subseteq \vec{u}$, possibly $\{x, y\} \cap \vec{u} \neq \emptyset$ */
if (2) is a SSA for a dynamic role $F(x, y, s)$, where $\phi(\vec{x}, \vec{z}, s)$ is a *context condition* uniform in s saying when an action A can have an effect on the fluent F. The formula $\phi(x, \vec{z}, s) \in \mathbf{F}^x$, the formula $\phi(y, \vec{z}, s) \in \mathbf{F}^y$, when s is suppressed. A set of variables $\vec{z}$ in a context condition $\phi(\vec{x}, \vec{z}, s)$ must be a subset of object variables $\vec{u}$. If $\vec{u}$ in an action function $A(\vec{u})$ does not include any z variables, then there is no $\exists\vec{z}$ quantifier.

If not all variables from $\vec{x}$ are included in $\vec{u}$, then it is said that $A(\vec{u})$ has a *global* effect, since the fluent F experiences changes beyond the objects explicitly named in $A(\vec{u})$ (e.g., driving a truck between two locations changes location of *all* boxes loaded into the truck). When a vector of object variables $\vec{u}$ contains both $\vec{x}$ and $\vec{z}$, we say that the action $A(\vec{u})$ has a *local effect*. A BAT is called a *local-effect BAT* if all of its actions have only local effects. Observe that in a local-effect SSA, when one substitutes a ground action term $A(\vec{b_x}, \vec{b_z})$ for a variable a in the formula $[\exists \vec{z}].a = A(\vec{x}, \vec{z}) \wedge \phi(\vec{x}, \vec{z}, s)$, applying UNA for action terms yields $[\exists \vec{z}].\vec{x} = \vec{b_x} \wedge \vec{z} = \vec{b_z} \wedge \phi(\vec{x}, \vec{z}, s)$, and applying $\exists z(z = b \wedge \phi(z)) \equiv \phi(b)$ repeatedly results in the formula $\vec{x} = \vec{b_x} \wedge \phi(\vec{x}, \vec{b_z}, s)$.
Initial Theory $\mathcal{D}_{S_0}$: The $\mathcal{D}_{S_0}$ is an $\mathcal{L}$ sentence without z variables, i.e., it can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept.

A BAT $\mathcal{D}$ that satisfies all of the above requirements is called an action theory $\mathcal{P}$. We note that BATs proposed in (Gu and Soutchanski 2010) are less general than $\mathcal{P}$, because their axioms should be written using formulas from FO_{DL}, but FO_{DL} is a proper subset of $\mathcal{L}$. Sometimes, for clarity, when we talk about $\mathcal{P}$, we say that it is an $\mathcal{L}$-based BAT, in contrast to FO_{DL}-based BATs considered in (Gu and Soutchanski 2010). The Blocks World is an example of a FO_{DL}-based BAT, while Logistics is an example of $\mathcal{P}$. Logistics cannot be formulated as a FO_{DL}-based BAT because it includes actions, e.g., $drive(Truck, Loc_1, Loc_2, City)$, with more than 2 arguments, and the SSA for a dynamic role $At(obj, loc, s)$ uses as a context condition an $\mathbf{F}^x$ formula, while in (Gu and Soutchanski 2010), the SSAs for dynamic roles must be context-free. Subsequently, for brevity, instead of saying that $\phi(s)$ is a SC formula uniform in s that becomes an $\mathcal{L}$ formula when s is suppressed, we say simply that $\phi(s)$ is an $\mathcal{L}$ formula.

Due to space limitations, we skip introduction to DLs, but the reader can find one in (Baader, Horrocks, and Sattler 2007). Recall that the satisfiability problem (SAT) of a concept and/or the consistency problem of an ABox in the DL language $\mathcal{ALCO}(\mathcal{U})$ can be solved in EXPTIME. As a comparison, SAT in $\mathcal{SROIQ}$ (this DL includes $\mathcal{ALCO}(\mathcal{U})$ as a fragment) has high complexity 2NEXPTIME-complete. However, W3C recommends OWL2 for applications, because OWL2 solvers handle many large realistic instances reasonably fast, see (Cuenca Grau et al. 2008) for a related discussion of OWL2 and $\mathcal{SROIQ}$.

Example 1 As an example of a $\mathcal{P}$ BAT, imagine searching for a given file in a depth-first search (DFS) like manner through directories. An action $forw(z_1, z_2, z_3)$ makes forward transition from a current directory z_1 to its child directory z_2 while searching for a file z_3 is possible in situation s, if z_2 has never been visited. This is represented using the fluent $vis(z_2, z_3, s)$. A backward transition $back(z_1, z_2, z_3)$ from z_1 back to its parent z_2 is possible only if all children of z_1 had been visited while searching for a file z_3. $\mathcal{P}$ also includes situation independent unary predicates $file(x)$, $dir(x)$, and the binary predicate $dirCh(x, y)$ meaning that x is a direct child of y in a file system. The search for a file f in a directory d succeeds when $find(d, f)$ is executed. This action is possible when d actually contains f. This is represented using the fluent $at(d, f, s)$. Using $chmod(z_1, z_2)$ one can toggle in situation s permissions of a directory z_1 between $z_2 = on$ and $z_2 = off$, if the current permission x for this directory z_1, represented using the fluent $p(z_1, x, s)$, is such that the values of x and z_2 are opposite. The following are precondition axioms (PA) for all actions (the variables z_i, s are $\forall$-quantified at front).

$$
\begin{aligned}
&Poss(forw(z_1,z_2,z_3),s) \equiv dir(z_1) \wedge dir(z_2) \wedge z_1 \neq z_2 \wedge \\
&\quad file(z_3) \wedge dirCh(z_2,z_1) \wedge \neg vis(z_2,z_3,s) \wedge at(z_1,z_3,s) \\
&Poss(back(z_1,z_2,z_3),s) \equiv dir(z_1) \wedge dir(z_2) \wedge file(z_3) \wedge \\
&\qquad dirCh(z_1, z_2) \wedge at(z_1, z_3, s) \wedge \\
&\qquad\quad \neg \exists y\, (\, dirCh(y, z_1) \wedge dir(y) \wedge \neg vis(y, z_3, s)\,) \\
&Poss(find(z_1,z_2),s) \equiv file(z_1) \wedge dir(z_2) \wedge \\
&\qquad dirCh(z_1, z_2) \wedge at(z_2, z_1, s) \\
&Poss(chmod(z_1,z_2),s) \equiv dir(z_1) \wedge (z_2 = on \vee z_2 = off) \wedge \\
&\qquad \exists x.(p(z_1, x, s) \wedge x \neq z_2).
\end{aligned}
$$

The direct effects and non-effects of actions are formulated using Successor State Axioms (SSA). The current DFS for a file y arrives at a directory x when either forward or backtracking transition leads to x; otherwise, if any other action is executed, it remains at x. Also, the directory x becomes visited as soon as DFS arrives there following some forward transition, but only if the current permission of x is *on* in situation s. Otherwise, forward transition has no effect. Changing permission of a directory x to y has an effect only when DFS for a file is currently located at x in situation s. A file f is found after doing $find(x, z_1$ in a directory z_1 only if permission is *on* for this directory in s.

$$
\begin{aligned}
&at(x,y,do(a, s)) \equiv \exists z_1(a = forw(z_1, x, y) \wedge p(x, on, s)) \vee \\
&\quad \exists z_1(a = back(z_1, x, y)) \vee \\
&\quad at(x,y,s) \wedge \neg \exists z_1(a = forw(x, z_1, y) \wedge p(z_1,on,s)) \wedge \\
&\qquad \neg \exists z_1(a = back(x, z_1, y)) \\
&vis(x,y,do(a, s)) \equiv \exists z_1(a = forw(z_1, x, y) \wedge p(x,on,s)) \vee \\
&\qquad vis(x, y, s) \\
&p(x, y, do(a, s)) \equiv a = chmod(x, y) \wedge \exists y\, at(x, y, s) \vee \\
&\qquad p(x, y, s) \wedge \neg \exists z_1(a = chmod(x, z_1) \wedge y \neq z_1 \wedge \\
&\qquad\quad \exists y.at(x, y, s)\,) \\
&found(x,do(a,s)) \equiv \exists z_1(a = find(x, z_1) \wedge p(z_1,on,s)) \vee \\
&\qquad found(x, s).
\end{aligned}
$$

Notice that the SSAs have syntactic forms required in $\mathcal{P}$.

3 The Projection Problem in $\mathcal{P}$

Let $\mathcal{D}$ be a description logic based BAT defined in (Gu and Soutchanski 2010), $\alpha_1, \cdots, \alpha_n$ be a sequence of ground action terms, and $Goal(s)$ be a query formula uniform in s such that it can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept, if s is suppressed. Subsequently, we call a query $Goal(S)$ a *regressable formula*, if S is a ground situation term. One of the most important reasoning tasks in the SC is the *projection problem*, that is, to determine whether $\mathcal{D} \models Goal(do([\alpha_1, \cdots, \alpha_n], S_0))$. Another basic reasoning task is the *executability problem*: whether all ground actions in $\alpha_1, \cdots, \alpha_n$ can be consecutively executed. This can be reduced to the projection problem using the precondition axioms, and for this reason we no longer consider it. Planning and high-level program execution are two important settings where the executability and projection problems arise naturally. *Regression* is a central computational mechanism that forms the basis of automated solutions to the executability and projection tasks in the SC (Reiter 2001). A recursive definition of the *modified regression operator* $\mathcal{R}$ on any regressable formula $Goal(S)$ is given in (Gu and Soutchanski 2010). The modified regression operator makes sure that the only two available object variables x, y are re-used when regressing a quantified formula in contrast to Reiter's regression, where new variables are introduced. For a regressable formula $Goal(S)$, we use notation $\mathcal{R}[Goal(S)]$ to denote the regressed formula uniform in S_0 that results from replacing repeatedly fluent atoms about $do(\alpha, s)$ by logically equivalent expressions about s as given by the RHS of SSAs, until such replacements no longer can be made; this is why the regressed formula is uniform in S_0. For any static concept $C(x)$ and role $R(x, y)$, by definition of regression $\mathcal{R}[C(x)] = C(x)$ and $\mathcal{R}[R(x, y)] = R(x, y)$.

The regression theorem (Theorem 8) proved in (Gu and Soutchanski 2010) shows that $\mathcal{R}[Goal(S)]$ is a FO_{DL} formula, when S_0 is suppressed and, as a consequence, one can reduce the projection problem for a regressable sentence $Goal(S)$ to the satisfiability problem in $\mathcal{ALCO}(\mathcal{U})$ as long as a BAT $\mathcal{D}$ satisfies syntactic restrictions due to using FO_{DL} formulas in axioms:

$$\mathcal{D} \models Goal \quad \text{iff} \quad \mathcal{D}_{S_0} \models \mathcal{R}[Goal(S)],$$

where it is assumed that $\mathcal{D}_{S_0}$ includes UNA, unique name axioms for objects. (Unique name axioms for actions are used by modified regression, and they are no longer required when regression terminates.) This statement is proved in

(Gu and Soutchanski 2010) for an extended BAT that additionally includes a set of axioms $\mathcal{D}_T = \mathcal{D}_{T,st} \cup \mathcal{D}_{T,dyn}$, where the static TBox $\mathcal{D}_{T,st}$ is an acyclic set of *concept definitions* that mentions only situation independent predicates (in (Gu and Soutchanski 2010), $\mathcal{D}_{S_0}$ includes $\mathcal{D}_{T,st}$), while dynamic TBox $\mathcal{D}_{T,dyn}$ is an acyclic set of definitions such that it has occurrences of fluents, but defined fluents are mentioned only in the RHS of SSAs, and they are eliminated by the modified regression operator using *lazy unfolding*. For example, $\mathcal{D}_{T,st}$ may include situation independent static definitions such as "vehicle is a truck or an airplane", while $\mathcal{D}_{T,dyn}$ may include convenient situation dependent abbreviations like $Movable(x, s) \equiv Loaded(x, s) \wedge \exists y In(x, y, s)$. The previously mentioned acyclicity assumption originates in (Baader et al. 2005).

We would like to eliminate a previous assumption that $\mathcal{D}_{T,st}$ is acyclic. For simplicity, let us consider a case when $\mathcal{D}_{T,dyn} = \emptyset$. Let $\mathcal{D}$ be $\mathcal{P}$ such that its initial theory $\mathcal{D}_{S_0}$ is augmented with an arbitrary satisfiable static TBox $\mathcal{D}_{T,st}$ that may include *general concept inclusions* between $\mathcal{ALCO}(\mathcal{U})$ concepts. (This TBox can be expressed as an $\mathcal{ALCO}(\mathcal{U})$ concept.) Then, by the relative satisfiability theorem from (Pirri and Reiter 1999), $\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ is satisfiable iff $UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ is satisfiable, i.e., the presence of a static satisfiable ontology is harmless. Moreover, since regression does not affect the predicates without a situation term, in other words, since axioms in $\mathcal{D}_{T,st}$ are invariant wrt the regression operator, it can be used to answer "static" queries and to reduce the projection problem to the satisfiability in $\mathcal{ALCO}(\mathcal{U})$: $\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models Goal$ iff $UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models Goal$, when Goal is an $\mathcal{L}$ sentence without z-variables that has no occurrences of fluents (a "static" query), and UNA includes unique name axioms only for objects. This simple observation is a consequence of Lemma 1 and the regression theorem from (Pirri and Reiter 1999). In addition, in $\mathcal{P}$ we can prove that formulas from $\mathcal{L}$ remain to be in $\mathcal{L}$ after regression.

Theorem 1 *Let $\mathcal{D}$ be an $\mathcal{L}$-based BAT (a theory $\mathcal{P}$), ϕ be a regressable $\mathcal{L}$ formula, and α a ground action. The result of regressing $\phi[(do(\alpha, S_0)]$, denoted by $\mathcal{R}[\phi(do(\alpha, S_0)]$, is a formula uniform in situation S_0 that is an $\mathcal{L}$-formula if S_0 is suppressed.*

This can be proved similarly to Lemma 2 from Section 5.4 in (Gu and Soutchanski 2010) that is proved for a FO_{DL}-based BAT. However, this does not follow directly from (Gu and Soutchanski 2010; Gu 2010) because in $\mathcal{P}$, SSA for dynamic roles may have context conditions, but in (Gu and Soutchanski 2010; Gu 2010) it was assumed that SSA for dynamic roles are context free. Also, recall that FO_{DL} is a proper subset of $\mathcal{L}$. The proof is long and laborious because regression is a syntactic operation, and the SSAs in $\mathcal{P}$ may have several different syntactic forms, but we have to analyze all cases and show that if we start with a DL-like formula, then after a single step of regression we get a formula that remains DL-like. As a consequence, for the "dynamic" queries, we have the following.

Theorem 2 *Let $\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ be $\mathcal{P}$ augmented with a (static) general $\mathcal{ALCO}(\mathcal{U})$ TBox , $\phi(S)$ be a regressable z-free $\mathcal{L}$ sentence, and S be a ground situation. Then the projection problem can be reduced to satisfiability in $\mathcal{ALCO}(\mathcal{U})$:*

$$\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models \phi(S) \quad \textit{iff} \quad UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st} \models \mathcal{R}[\phi(S)]$$

This follows from Theorem 1 by induction on the length of the situation term S, from Lemma 1, and from the fact that $UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept. The Theorem 2 is important because it shows that any static $\mathcal{ALCO}(\mathcal{U})$ ontology can be seamlessly integrated with reasoning about actions in $\mathcal{P}$. To the best of our knowledge we are the first to propose this. Our contribution is important because one may expect that in applications a static TBox characterizing essential terminological connections between concepts does not change when actions are executed, but only fluents can change. Also, one can add an acyclic dynamic TBox $\mathcal{D}_{T,dyn}$ to $\mathcal{P}$ without any difficulties, as in (Gu and Soutchanski 2010). However, (Baader et al. 2005; Liu et al. 2006; Calvanese et al. 2007) and others argue that a general dynamic TBox leads to serious difficulties. While (Baader et al. 2005) does not consider a general static TBox $\mathcal{D}_{T,st}$, it could be added, e.g., by internalizing $\mathcal{D}_{T,st}$ into an $\mathcal{ALCO}(\mathcal{U})$ concept and including it as an ABox assertion wrt a dummy individual. This trick was not considered in (Baader et al. 2005), because the universal role U is required for this trick to work, but U was missing in (Baader et al. 2005).

Example 1 (Cont.) We would like to adapt for our purposes an example of a general TBox from the paper by Giuseppe De Giacomo, Maurizio Lenzerini "TBox and ABox Reasoning in Expressive Description Logics", KR 1996, pages 316-327). Suppose that a TBox has the following concept inclusions:

$$\begin{array}{lcl} dir & \sqsubseteq & \forall dirCh^{-}.(dir \sqcup file) \sqcap \leq 1\, dirCh.dir \\ file & \sqsubseteq & \neg dir \sqcap \forall dirCh^{-}.\bot \end{array}$$

Let $\mathcal{D}_{S_0}$ be the following incomplete theory (written in FOL syntax, but can be easily translated to $\mathcal{ALCO}(\mathcal{U})$):

$$\begin{array}{l} dir(home) \wedge dir(mes) \wedge dir(root) \wedge dir(wyehia) \wedge \\ file(f1) \wedge dirCh(f1, mes) \wedge dirCh(mes, home) \wedge \\ file(f2) \wedge dirCh(f2, wyehia) \wedge dirCh(wyehia, home) \wedge \\ dirCh(home, root) \wedge at(wyehia, f1, S_0) \wedge \\ \forall x.(\neg(dir(x) \vee file(x)) \vee p(x, on, S_0)) \end{array}$$

The UNA for object constants: $f1, f2, home, mes, off, on, root, wyehia$ are pairwise distinct. Let the projection query be whether $\mathcal{D} \cup \text{TBox} \models found(f1, S)$, $S = do([back(wyehia, home, f1), forw(home, mes, f1), find(f1, mes)], S0))$.
Then, it is easy to see that the regressed query is

$$(f1 = f1 \wedge p(mes, on, S0)) \vee found(f1, S0).$$

This example demonstrates that we managed to solve the projection problem in the presence of a general expressive static TBox. This example BAT has been implemented in XML, regression of a query was computed using a C++ program, SAT in $\mathcal{SROIQ}$ was solved using HERMIT; see at `http://www.scs.ryerson.ca/mes/dl2012.zip`

4 Progression in $\mathcal{P}$

In this section, we use the notion of forgetting about a sequence of ground atoms, the notion of progression in SC, the fact about definability of progression in FOL for local effect BATs, and notation introduced in (Lin and Reiter 1994; 1997; Liu and Lakemeyer 2009). Recall that $\mathcal{P}$ is any $\mathcal{L}$-based BAT. It is easy to give an example of $\mathcal{P}$ with global effect actions such that progression of $\mathcal{D}_{S_0}$ is not definable as a z-free $\mathcal{L}$ sentence. Subsequently, we consider only local-effect $\mathcal{P}$ action theories, and we talk about z-free $\mathcal{L}$ sentences that can be transformed into an $\mathcal{ALCO}(\mathcal{U})$ concept. Below, we prove that progression of a z-free $\mathcal{L}$ sentence $\mathcal{D}_{S_0}$ is still expressible as a z-free $\mathcal{L}$ sentence $\mathcal{D}_{S_\alpha}$ (here and subsequently, for brevity, we talk about situation-suppressed sentences). This does not follow from Theorem 3.6 in (Liu and Lakemeyer 2009) about definability of progression in FOL for local-effect BATs, since our initial theory $\mathcal{D}_{S_0}$ is formulated in a strict subset of FO^2 language, and it is not obvious at all whether in $\mathcal{P}$ progression $\mathcal{D}_{S_\alpha}$ of $\mathcal{D}_{S_0}$ can still be defined within same language. Since progression involves forgetting about old values of fluents and computing new values, we need a couple of intermediate lemmas. First, we show that new fluent values can be expressed in $\mathcal{L}$. Then, we prove that the result of forgetting about ground fluents in $\mathcal{D}_{S_0}$ affected by a ground action α remains to be a z-free $\mathcal{L}$ sentence.

Lemma 2 *Let $\mathcal{D}$ be a local effect $\mathcal{P}$, α a ground action, and $\Omega(S_0)$ be the characteristic set of α with respect to $\mathcal{D}$. Then $\mathcal{D}_{SS}[\Omega]$ is a set of $\mathcal{L}$ sentences without occurrences of z-variables, when the situation terms are suppressed.*

The characteristic set $\Omega(S_0)$ is a set of ground fluents affected by α. Because they change values, we have to forget their old values. To compute new values for them, we instantiate $\mathcal{D}_{SS}$ w.r.t. $\Omega(S_0)$, do simplification and obtain the set of sentences $F(\vec{t}, S_\alpha) \equiv \Phi_F(\vec{t}, \alpha, S_0)$, which are denoted as $\mathcal{D}_{SS}[\Omega]$, where $S_\alpha = do(\alpha, S_0)$, and $\Phi_F(\vec{t}, \alpha, S_0)$ is a z-free $\mathcal{L}$ sentence representing the RHS of a SSA for the fluent F. $F(\vec{t}, S_\alpha)$ and $\Phi_F(\vec{t}, \alpha, S_0)$ mention different situation terms. However, $F(\vec{t}, S_\alpha)$ can never occur in Φ_F or any RHS of SSA of other fluents because they are all uniform in S_0. Also, none of the ground fluents to be subsequently forgotten are relevant to $F(\vec{t}, S_\alpha)$ simply because it is the value of F in a different situation. Consequently, we can replace $F(\vec{t}, S_\alpha)$ temporarily by some atom F_t until forgetting of $\Omega(S_0)$ is completed, and then put it back while preserving logical equivalence. The next lemma shows that forgetting about ground atoms $\Omega(S_0)$ in an $\mathcal{L}$ formula results in an $\mathcal{L}$ formula.

Lemma 3 *Let ϕ be a $\boldsymbol{F}^x$ (or $\boldsymbol{F}^y$) formula and θ a truth assignment to some of the atoms $P(\vec{t_j})$ occurring in this formula (if any), then $\phi[\theta]$ remains a $\boldsymbol{F}^x$ ($\boldsymbol{F}^y$) formula.*

Notation $\phi[\theta]$ for forgetting about several ground atoms, introduced in (Liu and Lakemeyer 2009), means the result of replacing every occurrence of an atom $P(\vec{x})$ in ϕ by $\bigvee_{j=1}^m (\vec{x} = \vec{t_j} \wedge \theta[P(\vec{t_j})]) \vee (\bigwedge_{j=1}^m \vec{x} \neq \vec{t_j}) \wedge P(\vec{x})$. This Lemma is proved by induction over structure of ϕ.

Theorem 3 *Let $\mathcal{D}$ be a local-effect BAT based on $\mathcal{L}$ and α a ground action. Let $\Omega(s)$ be the characteristic set of α. Then the following formula is a progression of $\mathcal{D}_{S_0}$ w.r.t. α and this formula is an $\mathcal{L}$ sentence:*

$$\bigwedge UNA \wedge \bigvee_{\theta \in \mathcal{M}(\Omega(S_0))} (\bigwedge \mathcal{D}_{S_0} \wedge \bigwedge \mathcal{D}_{SS}[\Omega])[\theta] \ \ (S_\alpha / S_0)$$

Proof: This is a consequence of Lemmas (2), (3) and Theorem 3.6 from (Liu and Lakemeyer 2009). Note that the final formula is uniform in S_α. This theorem is important for our work because it shows for $\mathcal{P}$ that if an initial theory $\mathcal{D}_{S_0}$ is expressible as an $\mathcal{ALCO}(\mathcal{U})$-like concept, then progression $\mathcal{D}_{S_\alpha}$ is also expressible as an $\mathcal{ALCO}(\mathcal{U})$-like concept.

5 Efficient Progression in p^+ KB

Theorem 3 shows progression $\mathcal{D}_{S_\alpha}$ can be translated to $\mathcal{ALCO}(\mathcal{U})$, but in a general case, the size of progression can be much larger than the size of $\mathcal{D}_{S_0}$. If one wants to solve the projection problem by computing progression for a sequence of action, then one has to find special cases of an initial theory $\mathcal{D}_{S_0}$ such that the size of progression remains linear w.r.t. the size of $\mathcal{D}_{S_0}$. (Liu and Lakemeyer 2009) prove that progression is computationally tractable if an initial $\mathcal{D}_{S_0}$ is in $proper^+$ form, where $proper^+$ theories generalize databases by allowing incomplete disjunctive knowledge about some of the named elements of the domain (Lakemeyer and Levesque 2002). A $proper^+$ knowledge base (KB) is more general than a *proper* KB, which is equivalent to a possibly infinite consistent set of ground literals. We show that in $\mathcal{P}$, if $\mathcal{D}_{S_0}$ is a set of $proper^+$ formulas that can be translated into $\mathcal{ALCO}(\mathcal{U})$, then progression of $\mathcal{D}_{S_0}$ in our new normal form can be computed efficiently, and the normal form can be maintained without introducing any new variables. To achieve this, we show that a KB in our new normal form remains in the same normal form after forgetting about old values of fluents. The fact that forgetting in our normal form KB can be accomplished without introducing new variables is novelty that does not follow from (Liu and Lakemeyer 2009).

Let e be an *ewff*, a well-formed formula whose only predicate is equality, and let a clause d be a disjunction of literals. Recall that the universal closure $\forall(e \supset d)$ is called a *guarded clause*, or a $proper^+$-formula, and a KB is called $proper^+$ if it is a finite non-empty set of guarded clauses supplemented with the axioms of equality and the set of UNA for constants. We are going to use a p^+ normal form in which forgetting can be accomplished without introducing new variables. The new form is logically equivalent to the $proper^+$ normal form, but it is more handy for our purposes. In addition, there is no increase in the size when a $proper^+$ formula is transformed into p^+ form.

Definition 2 *We say that a disjunction of guarded clauses $\bigvee_i \forall(e_i \supset d_i)$ is a p^+ formula. A p^+ KB is a finite set of p^+ formulas (plus axioms of equality and UNA for constants).*

We would like to show that a KB in our p^+ normal form can be equivalently transformed into the same normal form after forgetting about old values of fluents, and none of the intermediate logical transformations require introducing new variables to preserve logical equivalence. Working with p^+ KB requires logically equivalent transformations using $\forall$-quantifiers applied to disjunctions. To ensure these transformations do not need fresh variables to preserve equivalence, we introduce auxiliary technical notions.

Definition 3 *Let S_i be a set of variables that occur in a guarded clause $\forall(e_i \supset d_i)$. Let $\mathcal{S} = \{S_1, S_2, ...S_n\}$ be a collection of these sets such that they are pairwise disjoint. We call a p^+ formula $\phi = \bigvee_i \forall(e_i \supset d_i)$ separable w.r.t. $\mathcal{S}$ iff for each guarded clause $\forall(e_i \supset d_i)$ the free variables of $e_i \supset d_i$ are a subset of one and only one set in $\mathcal{S}$.*

Definition 4 *Let capacity($\mathcal{S}$) be the maximum number of variables in a set from $\mathcal{S}$.*

Our goal is to transform a KB into a form such that forgetting about a ground atom $P(\vec{c})$ becomes a simple syntactic operation. Note that the easiest case for forgetting about a ground atom $P(\vec{c})$ in a formula ϕ_{irr} is when $P(\vec{c})$ is *irrelevant* to ϕ_{irr}, or formally, when $forget(\phi_{irr}, P(\vec{c})) \equiv \phi_{irr}$. For example, this applies when a formula ϕ_{irr} has no occurrences of P. Otherwise, let's consider a ground (for the sake of simplicity) KB where all clauses mention a predicate symbol P at most once. Then, using distributivity $((a \vee P(\vec{c})) \wedge (b \vee P(\vec{c}))) \equiv (a \wedge b \vee P(\vec{c}))$, we can collect all sub-formulas from clauses that mention $P(\vec{c})$ into a single conjunction ϕ_{pos}. Similarly, we can collect all sub-formulas from clauses that mention $\neg P(\vec{c})$ into a single conjunction ϕ_{neg}. Then, we can use the following simple observation to forget about $P(\vec{c})$ easily. Let $P(\vec{c})$ be a ground atom, ϕ_{pos}, ϕ_{neg}, and ϕ_{irr} be sentences to which $P(\vec{c})$ is irrelevant, and KB be $(P(\vec{c}) \vee \phi_{pos}) \wedge (\neg P(\vec{c}) \vee \phi_{neg}) \wedge \phi_{irr}$, Then, $forget(KB, P(\vec{c})) = (\phi_{pos} \vee \phi_{neg}) \wedge \phi_{irr}$. Now, we would like to elaborate these observations in the context of p^+ KBs that have $\forall$-quantifiers. This preliminary discussion motivates the subsequent developments.

Proposition 1 *Let $\phi = \bigvee_i \forall(e_i \supset d_i)$ be a p^+ formula. If for each d_i and for every $P(\vec{t})$ in d_i the formula $(e_i \wedge \vec{t} = \vec{c})$ is unsatisfiable, then $P(\vec{c})$ is irrelevant to ϕ.*

We are ready to define a new normal form NF_+ that accommodates disjunctions of guarded clauses. Our definition takes into account separability of variables that is important for equivalent transformations with $\forall$ such as $\forall y(A(x) \vee B(y)) \equiv A(x) \vee \forall y B(y)$.

Definition 5 *Let $\mathcal{K}$ be a p^+ KB, $\mathcal{S}'$ be a collection of pairwise disjoint sets of variables and $P(\vec{c})$ be a ground atom. Then, $\mathcal{K}$ is in NF_+ normal form w.r.t $P(\vec{c})$, called $NF_+(\mathcal{K}, P(\vec{c}))$, if $\mathcal{K} = \{\phi \mid \phi = \bigvee_i \forall(e_i \supset d_i)\}$ such that*

1. *each formula $\phi \in \mathcal{K}$ is separable w.r.t. $\mathcal{S}'$,*
2. *in each $\phi \in \mathcal{K}$, for any $P(\vec{t})$ appearing in any d_i, either $\vec{t}$ is $\vec{c}$ or $(e_i \wedge \vec{t} = \vec{c})$ is unsatisfiable.*

The crucial fact is that a p^+ KB can be normalized without introducing new variables.

Theorem 4 *Let $\mathcal{S}$ be a collection of sets of variables such that these sets are pairwise disjoint, and $capacity(\mathcal{S}) = m$ for some integer $m > 0$. Let $\mathcal{K}$ be a KB of p^+ formulas that are separable w.r.t. $\mathcal{S}$, and let $P(\vec{c})$ be a ground atom. Then, $\mathcal{K}$ can be transformed into a KB in NF_+ normal form w.r.t $P(\vec{c})$, such that each constituent formula $\phi = \bigvee_i \forall(e_i \supset d_i)$ is separable w.r.t. a collection $\mathcal{S}'$ of pairwise disjoint sets of variables, $\mathcal{S} \subseteq \mathcal{S}'$, and $capacity(\mathcal{S}') = m$.*

For a given ground atom $P(\vec{c})$, converting a p^+ KB into NF_+ produces only three different types of p^+ formulas with sub-formulas ϕ_{pos}, ϕ_{neg}, ϕ_{irr} such that $P(\vec{c})$ is irrelevant to them: formula of structure $P(\vec{c}) \vee \phi_{pos}$ is called pos-type formula, formula of structure $\neg P(\vec{c}) \vee \phi_{neg}$ is called neg-type formula, irr-type formulas ϕ_{irr} have no occurrences of $P(\vec{c})$. As explained above, forgetting about $P(\vec{c})$ in NF_+ can be easily accomplished. Moreover, we prove that the KB resulting from forgetting about $P(\vec{c})$ can be easily transformed back into the p^+ KB while preserving capacity (i.e., without introducing new variables).

Theorem 5 *Let $\mathcal{S}$ be a collection of sets of variables such that the sets in each collection are pairwise disjoint, and $capacity(\mathcal{S}) = m$ for some integer $m > 0$. Let $\mathcal{K}$ be a p^+ KB of formulas that are separable w.r.t. $\mathcal{S}$, and $P(\vec{c})$ be a ground atom. Then, the result of forgetting $P(\vec{c})$ in $\mathcal{K}$ is a p^+ KB of formulas that are separable w.r.t. a collection of pairwise disjoint sets of variables $\mathcal{S}'$ such that $\mathcal{S} \subseteq \mathcal{S}$' and $capacity(\mathcal{S}') = m$.*

Since our proof shows that the transformations do not incur any increase in the sizes of the formulas, the complexity of forgetting is as in (Liu and Lakemeyer 2009): linear w.r.t. the size of a KB.

Once forgetting has been completed, the maximum number of variables in any guarded clause does not exceed the initial $capacity(\mathcal{S})$ of a KB that is transformed back into p^+ form. Consequently, we can accomplish forgetting in our action theory $\mathcal{P}$, if we start with an initial theory $\mathcal{D}_{S_0}$ that is both in p^+ normal form and that is a z-free $\mathcal{L}$ sentence. The intersection of these two languages should be expressive enough for applications.

Regarding the connection between $\mathcal{ALCO}(\mathcal{U})$ and p^+, it is clear that the intersection of $\mathcal{L}$ and the language for defining p^+ KBs restricts the capacity of guarded clauses to 2 since $\mathcal{ALCO}(\mathcal{U})$ is a fragment of FO^2. In $\mathcal{P}$, if context formulas $\mathbf{F}^x$ and $\mathbf{F}^y$ are *essentially quantifier free* (i.e., if context conditions can be simplified to quantifier free formulas), then $\mathcal{D}_{SS}[\Omega]$ can be converted to CNF of literals, and then into a p^+ KB. Therefore, by Theorem 3, since $\mathcal{D}_{SS}[\Omega]$ is $O(1)$ in size ($\mathcal{D}_{SS}$ is a fixed input), the only potentially large input that matters is $\mathcal{D}_{S_0}$. The number of affected ground fluent literals that should be forgotten is limited by the structure of the $\mathcal{D}_{SS}$, i.e., it can be considered a constant. Therefore, we would apply a constant number of forgetting operations, each operation increasing the size of $\mathcal{D}_{S_0}$ linearly. Overall, this yields progression $\mathcal{D}_{S_\alpha}$ that is linear w.r.t $\mathcal{D}_{S_0}$. Once an initial theory $\mathcal{D}_{S_0}$ has been progressed, the projection problem can be solved using any $\mathcal{ALCO}(\mathcal{U})$ satisfiability solver.

6 Discussion and Future Work

Main contributions of our paper are as follows. First, we define a logical theory $\mathcal{P}$ integrating reasoning about action with DLs such that $\mathcal{P}$ is more expressive than theories from (Gu 2010; Gu and Soutchanski 2010). For example, in $\mathcal{P}$, there are no restrictions on arity of actions functions. Second, Theorem 2 (regression in $\mathcal{P}$) shouldn't be underestimated. It shows existing ontologies (with a general $\mathcal{ALCO}(\mathcal{U})$ static TBox) can be seamlessly integrated with $\mathcal{P}$. To the best of our knowledge, this seamless integration of DLs and reasoning about actions has never been proposed before. For example, (Baader et al. 2005; Gu and Soutchanski 2010) allowed only acyclic dynamic TBox (that can be easily added to $\mathcal{P}$ too). Third, Theorem 3 is a new non-trivial statement that doesn't follow from (Liu and Lakemeyer 2009). It is important because it guarantees that progression of $\mathcal{ALCO}(\mathcal{U})$ KBs can still be formulated in the same language, and consequently, one can continue computing progression for subsequent actions. Fourth, Theorems 4 and 5 are proved using new techniques. They don't follow from (Liu and Lakemeyer 2009), where progression was studied in FOL.

Our regression in $\mathcal{P}$ had been successfully implemented in XML and C++ (Yehia 2012) and extensively tested on half a dozen benchmark domains (Kudashkina 2011; Yehia 2012). In (Kudashkina 2011), ADL versions of several planning specifications (considered as FOL theories, without grounding) have been manually translated from PDDL (Planning Domains Definition Language) into XML encoding of $\mathcal{P}$. Note that STRIPS planning domains are trivial. When fluents have more than two object arguments, they can be rephrased using simpler fluents if arguments vary over finite ranges: see examples of such transformations in Section 5.2 of (Gu and Soutchanski 2010). An implementation of progression in $\mathcal{P}$ is ongoing: see the pseudo-code at `http://www.scs.ryerson.ca/mes/dl2012.zip`

An approach to integrating DLs and reasoning about actions proposed in (Baader et al. 2005) inspired a number of subsequent papers including (Gu and Soutchanski 2010), where the reader can find extensive comparison and discussion. The approach proposed in (Baader et al. 2005) is expressive, and it can be used to represent many popular AI action theories. However, one can answer only ground projection queries using their approach, but Theorem 2 shows we can use regression to answer projection queries with quantifiers over object arguments in fluents. Also, our regression can be used to solve the projection problem in a BAT where some actions have global effects, but the approach proposed in (Baader et al. 2005) can answer projection queries only in local effect BATs. In any case, it is important to compare our implementations with an implementation based on (Baader et al. 2005) for the common classes of queries and theories. An empirical assessment can use a few planning domains and a number of other benchmarks. The first step in this direction is taken in (Yehia et al. 2012).

All other related publications are very extensively discussed in (Baader et al. 2005; Calvanese et al. 2007; Gu and Soutchanski 2010; Liu et al. 2006).

Acknowledgements

Thanks to the Natural Sciences and Engineering Research Council of Canada (NSERC) and to the Department of Computer Science of York University for providing partial financial support. We are grateful to Giuseppe De Giacomo and Yilan Gu for discussions at an earlier stage of this research.

References

Artale, A., and Franconi, E. 2000. A survey of temporal extensions of description logics. *Ann. Math. Artif. Intell.* 30(1-4):171–210.

Baader, F.; Lutz, C.; Miliĉić, M.; Sattler, U.; and Wolter, F. 2005. Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the 20th AAAI Conference*, 572–577.

Baader, F.; Horrocks, I.; and Sattler, U. 2007. Description Logics. In *Handbook of Knowledge Representation*. Elsevier. 135–179.

Calvanese, D.; De Giacomo, G.; Lenzerini, M.; and Rosati, R. 2007. Actions and Programs over Description Logic Ontologies. In *Description Logics Workshop (DL-2007)*.

Chang, L.; Shi, Z.; Gu, T.; and Zhao, L. 2012. A Family of Dynamic Description Logics for Representing and Reasoning About Actions. *Journal of Automated Reasoning* 49(1):1–52.

Cuenca Grau, B.; Horrocks, I.; Motik, B.; Parsia, B.; Patel-Schneider, P. F.; and Sattler, U. 2008. OWL 2: The next step for OWL. *J. Web Sem.* 6(4):309–322.

De Giacomo, G., and Lenzerini, M. 1995. PDL-based framework for reasoning about actions. In Gori, M., and Soda, G., eds., *AI*IA*, volume 992 of *Lecture Notes in Computer Science*, 103–114. Springer.

De Giacomo, G.; Iocchi, L.; Nardi, D.; and Rosati, R. 1999. A theory and implementation of cognitive mobile robots. *J. Log. Comput.* 9(5):759–785.

Devanbu, P. T., and Litman, D. J. 1996. Taxonomic plan reasoning. *Artif. Intell.* 84(1-2):1–35.

Gu, Y., and Soutchanski, M. 2010. A Description Logic Based Situation Calculus. *Ann. Math. Artif. Intell.* 58(1-2):3–83.

Gu, Y. 2010. *Advanced Reasoning about Dynamical Systems*. Ph.D. Dissertation, Department of Computer Science, University of Toronto, Canada.

Kudashkina, E. 2011. *An Empirical Evaluation of the Practical Logical Action Theory (Undergrad. Thesis)*. Toronto, Canada: Dep. of Comp. Science, Ryerson University.

Lakemeyer, G., and Levesque, H. J. 2002. Evaluation-Based Reasoning with Disjunctive Information in First-Order Knowledge Bases. In *Proc. of KR-02*, 73–81.

Lin, F., and Reiter, R. 1994. Forget It! In *Proceedings of the AAAI Fall Symposium on Relevance*, 154–159.

Lin, F., and Reiter, R. 1997. How to Progress a Database. *Artificial Intelligence* 92:131–167.

Liu, Y., and Lakemeyer, G. 2009. On First-Order Definability and Computability of Progression for Local-Effect Actions and Beyond. In Boutilier, C., ed., *IJCAI*, 860–866.

Liu, H.; Lutz, C.; Milii, M.; and Wolter, F. 2006. Reasoning About Actions Using Description Logics with General TBoxes. In *Logics in Artificial Intelligence*, volume 4160 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg. 266–279.

Lutz, C., and Sattler, U. 2002. A Proposal for Describing Services with DLs. In *Proc. of the 15th Intl Workshop on Description Logics*.

McDermott, D. V. 2000. The 1998 AI Planning Systems Competition. *AI Magazine* 21(2):35–55.

Pirri, F., and Reiter, R. 1999. Some Contributions to the Metatheory of the Situation Calculus. *Journal of the ACM* 46(3):325–364.

Reiter, R. 2001. *Knowledge in Action: Logical Foundat. for Describing and Implementing Dynam. Systems*. MIT Press.

Wolter, F., and Zakharyaschev, M. 1998. Dynamic description logics. In *Advances in Modal Logic 2*, 431–446.

Yehia, W.; Liu, H.; Lippmann, M.; Baader, F.; and Soutchanski, M. 2012. Experimental Results on Solving the Projection Problem in Action Formalisms Based on Description Logics. In *Proc. of the 25th Intern. Workshop on Description Logics*.

Yehia, W. 2012. *MSc Thesis (forthcoming)*. Toronto, Canada: Dep. of Comp. Science and Engineer., York Univ.