Towards an Expressive Decidable Logical Action Theory

Wael Yehia[1] and Mikhail Soutchanski[2]

[1] Department of Computer Science and Engineering York University, 4700 Keele Street, Toronto, ON, M3J 1P3, Canada
w2yehia@cse.yorku.ca
[2] Department of Computer Science, Ryerson University, 245 Church Street, ENG281, Toronto, ON, M5B 2K3, Canada mes@scs.ryerson.ca

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 [5, 11] to mention a few only. 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 [10, 9]. 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 [1, 23, 7]. 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 [19, 4, 17, 6, 13]. In [13], 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 [13], 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 is interpreted as the binary relation that links any two domain elements.) They consider Reiter's basic action theories (BATs) [22], 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 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) [20], 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, [12] 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 her 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 fragment $\mathcal{P}$ show that $\mathcal{P}$ has expressive power sufficient to represent not only Blocks World and Logistics, but also many other domains. 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 [8]. In our paper, we concentrate on foundational work and explore the logical properties of $\mathcal{P}$. Our paper contributes by formulating an expressive fragment of SC where the projection problem is decidable without DCA and closed world assumption (CWA), i.e., when an initial theory is incomplete and is not purely propositional. We hope that research outlined in our paper will attract the description logic community to interesting research issues on the boundary between DLs and reasoning about actions.

Definition of $\mathcal{P}$

We assume that the reader is familiar with SC from [21, 22] 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 [22], axioms Σ are formulated in second-order logic, all other axioms are formulated in 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 [21].

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 $A(\boldsymbol{x})$. In particular, terms of sort object can be only constants or variables. Each action function can 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 $b_i, 1 \leq n \leq n$. 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 $\mathcal{L} = \boldsymbol{F}^x \cup \boldsymbol{F}^y$, and $\widetilde{\cdot}$ be a bijection between formulas in $\boldsymbol{F}^x$ and $\boldsymbol{F}^y$ as defined above, where the sets $\boldsymbol{F}^y$ and $\boldsymbol{F}^x$ are minimal sets constructed as follows. (We focus on $\boldsymbol{F}^x$, since $\boldsymbol{F}^y$ is similar.)*

1. *$\top$ and $\bot$ are in $\boldsymbol{F}^x$.*
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 $\boldsymbol{F}^x$.*
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 that 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 [13]

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 translated formula.*

This lemma is proved using the standard translation between DLs and FOL; the proof is similar to the proof of Lemma 1 in [13]. 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 $\boldsymbol{x}$ of object variables denote either x, or y, or $\langle x, y\rangle$; also, let $\boldsymbol{z}$ denote a finite vector of place holder variables.

Action precondition axioms $\mathcal{D}_{AP}$: For each action function $A(\boldsymbol{z})$, there is a single precondition axiom uniform in s:

$$(\forall \boldsymbol{z}, s).\ \ Poss(A(\boldsymbol{z}), s) \equiv \Pi_A(\boldsymbol{z}, s), \tag{1}$$

where $\Pi_A(\boldsymbol{z}, s)$ is uniform in s; it is an $\mathcal{L}$ formula with $\boldsymbol{z}$ as the only free variables, if any, when s is suppressed. When object arguments of $A(\boldsymbol{z})$ 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 the situation variable s is suppressed.

Successor state axioms $\mathcal{D}_{SS}$: There is a single SSA for each fluent $F(\boldsymbol{x}, do(a, s))$. 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:

$$(\forall \boldsymbol{x}, s, a).\ \ F(\boldsymbol{x}, do(a, s)) \equiv \gamma^+_F(\boldsymbol{x}, a, s) \vee F(\boldsymbol{x}, s) \wedge \neg\gamma^-_F(\boldsymbol{x}, a, s) \tag{2}$$

where each of the γ_F's are disjunctions either of the form

$[\exists \boldsymbol{z}].a = A(\boldsymbol{u}) \wedge \phi(x, \boldsymbol{z}, s)$, /* a set of variables $\boldsymbol{z} \subseteq \boldsymbol{u}$; may be $\{x\} \in \boldsymbol{u}$ */

if (2) is a SSA for a dynamic concept $F(x, s)$ with a single object argument x, or

$[\exists \boldsymbol{z}].a = A(\boldsymbol{u}) \wedge \phi(x, \boldsymbol{z}, s) \wedge \phi(y, \boldsymbol{z}, s)$, /* $\boldsymbol{z} \subseteq \boldsymbol{u}$, possibly $\{x, y\} \cap \boldsymbol{u} \neq \emptyset$ */

if (2) is a SSA for a dynamic role $F(x, y, s)$, where $\phi(\boldsymbol{x}, \boldsymbol{z}, s)$ is a *context condition* uniform in s saying when an action A can have an effect on the fluent F. The formula $\phi(x, \boldsymbol{z}, s) \in \mathbf{F}^x$, the formula $\phi(y, \boldsymbol{z}, s) \in \mathbf{F}^y$, when s is suppressed. A set of variables $\boldsymbol{z}$ in a context condition $\phi(\boldsymbol{x}, \boldsymbol{z}, s)$ must be a subset of object variables $\boldsymbol{u}$. If $\boldsymbol{u}$ in an action function $A(\boldsymbol{u})$ does not include any z variable, then there is no $\exists \boldsymbol{z}$ quantifier.

If not all variables from $\boldsymbol{x}$ are included in $\boldsymbol{u}$, then it is said that $A(\boldsymbol{u})$ has a *global* effect, since the fluent F experiences changes beyond the objects explicitly named in $A(\boldsymbol{u})$ (e.g., driving a truck between two locations changes location of *all* boxes loaded in the truck). When a vector of object variables $\boldsymbol{u}$ contains both $\boldsymbol{x}$ and $\boldsymbol{z}$, we say that the

action $A(\boldsymbol{u})$ 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 $A(\boldsymbol{b_x}, \boldsymbol{b_z})$ for a variable a in the formula $[\exists \boldsymbol{z}].a = A(\boldsymbol{x}, \boldsymbol{z}) \wedge \phi(\boldsymbol{x}, \boldsymbol{z}, s)$, applying UNA for action terms yields $[\exists \boldsymbol{z}].\boldsymbol{x} = \boldsymbol{b_x} \wedge \boldsymbol{z} = \boldsymbol{b_z} \wedge \phi(\boldsymbol{x}, \boldsymbol{z}, s)$, and applying $\exists z(z = b \wedge \phi(z)) \equiv \phi(b)$ repeatedly results in the equivalent formula $\boldsymbol{x} = \boldsymbol{b_x} \wedge \phi(\boldsymbol{x}, \boldsymbol{b_z}, s)$.

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 basic action theory $\mathcal{D}$ that satisfies all of the above requirements is called an action theory $\mathcal{P}$. We note that BATs proposed in [13] 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 [13]. 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 [13], 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 ϕ is an $\mathcal{L}$ formula.

Due to space limitations, we skip introduction to DLs, but the reader can find one in [2]. Recall that the satisfiability problem of a concept and/or the consistency problem of an ABox in the DL language $\mathcal{ALCO}(\mathcal{U})$ can be solved in EXPTIME.

Example 1. As an example of $\mathcal{P}$, imagine searching for a given file in a depth-first search (DFS) like manner through directories. An action $forward(z_1, z_2, z_3)$ makes forward transition from a current directory z_1 to its child directory z_2 while searching for a file z_3. It 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 $backtrack(z_1, z_2, z_3)$ transition 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 $dirChild(x, y)$ 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 $perm(z_1, x, s)$, 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).

$$
\begin{aligned}
&Poss(forward(z_1, z_2, z_3), s) \equiv dir(z_1) \wedge dir(z_2) \wedge z_1 \neq z_2 \wedge file(z_3) \wedge \\
&\qquad dirChild(z_2, z_1) \wedge \neg vis(z_2, z_3, s) \wedge at(z_1, z_3, s) \\
&Poss(backtrack(z_1, z_2, z_3), s) \equiv dir(z_1) \wedge dir(z_2) \wedge file(z_3) \wedge dirChild(z_1, z_2) \wedge \\
&\qquad at(z_1, z_3, s) \wedge \neg \exists y \, (\, dirChild(y, z_1) \wedge dir(y) \wedge \neg vis(y, z_3, s) \,) \\
&Poss(find(z_1, z_2), s) \equiv file(z_1) \wedge dir(z_2) \wedge dirChild(z_1, z_2) \wedge at(z_2, z_1, s) \\
&Poss(chmod(z_1, z_2), s) \equiv dir(z_1) \wedge (z_2 = on \vee z_2 = off) \wedge \\
&\qquad\qquad\qquad\qquad \exists x.(perm(z_1, x, s) \wedge x \neq z_2).
\end{aligned}
$$

The direct 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.

$$
\begin{aligned}
at(x,y,do(a,s)) \equiv\ & \exists z_1(a=forward(z_1,x,y) \wedge perm(x,on,s)) \vee \\
& \exists z_1(a=backtrack(z_1,x,y)) \vee \\
& at(x,y,s) \wedge \neg\exists z_1(a=forward(x,z_1,y) \wedge perm(z_1,on,s)) \wedge \\
& \neg\exists z_1(a=backtrack(x,z_1,y)) \\
vis(x,y,do(a,s)) \equiv\ & \exists z_1(a=forward(z_1,x,y) \wedge perm(x,on,s)) \ \vee vis(x,y,s) \\
perm(x,y,do(a,s)) \equiv\ & a=chmod(x,y) \wedge \exists y(at(x,y,s) \wedge y=y\,) \vee \\
& perm(x,y,s) \wedge \neg\exists z_1(a=chmod(x,z_1) \wedge y \neq z_1 \wedge \exists y.at(x,y,s) \wedge y=y\,) \\
found(x,do(a,s)) \equiv\ & \exists z_1(a=find(x,z_1) \wedge perm(z_1,on,s)) \vee found(x,s).
\end{aligned}
$$

These SSAs satisfy syntactic constraints in $\mathcal{P}$, but they cannot be formulated as FO_{DL}-based SSAs considered in [13] since SSAs for the dynamic roles at and $perm$ have context conditions and mention action functions with more than 2 arguments. Clearly, neither PAs, nor SSAs can be translated to a DL, but nevertheless, there are instances of the projection problem in this BAT that can be reduced to SAT in a DL.

The Projection Problem in $\mathcal{P}$

Let $\mathcal{D}$ be a description logic based BAT defined in [13], $\alpha_1, \cdots, \alpha_n$ 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 [22]. A recursive definition of the *modified regression operator* $\mathcal{R}$ on any regressable formula $Goal(S)$ is given in [13]. 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 [13] 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 [13] 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 [13], $\mathcal{D}_{S_0}$ 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 [4].

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 [21], $\Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ssa} \cup UNA \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{T,st}$ 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 [21]. 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 [13] that is proved for a FO_{DL}-based BAT. However, this does not follow directly from [13, 12] because SSA for dynamic roles may have context conditions in $\mathcal{P}$, but in [13, 12] 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 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 $\mathcal{ALCO}(\mathcal{U})$:*

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

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. This theorem is important because it shows that any static $\mathcal{ALCO}(\mathcal{U})$ ontology can be seamlessly integrated with reasoning about actions in $\mathcal{P}$. Also, one can add an acyclic dynamic TBox $\mathcal{D}_{T,dyn}$ to $\mathcal{P}$ without any difficulties, as in [13]. However, [4, 17, 6] and others argue that a general dynamic TBox leads to serious difficulties. While [4] 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 [4], because the universal role U is required for this trick to work, but U was missing in [4].

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 static TBox has the following general concept inclusions:

$$dir \sqsubseteq \forall dirChild^{-}.(dir \sqcup file) \sqcap \leq 1\ dirChild.dir$$
$$file \sqsubseteq \neg dir \sqcap \forall dirChild^{-}.\bot$$

Let an initial $\mathcal{D}_{S_0}$ be the following theory (written as $\mathcal{L}$ formula for brevity):
$dir(home) \wedge dir(mes) \wedge dir(root) \wedge dir(wyehia) \wedge file(f1) \wedge file(f2)$
$dirChild(f1, mes) \wedge dirChild(f2, wyehia) \wedge dirChild(home, root) \wedge$
$dirChild(mes, home) \wedge dirChild(wyehia, home) \wedge$
$at(wyehia, f1, S_0) \wedge \forall x.(\neg(dir(x) \vee file(x)) \vee perm(x, on, S_0))$
The UNA for object constants (represented as nominals in $\mathcal{ALCO}(\mathcal{U})$):
$\{f1\} \neq \{f2\} \neq \{home\} \neq \{mes\} \neq \{off\} \neq \{on\} \neq \{root\} \neq \{wyehia\}$

Let the projection query be whether $\mathcal{D} \cup$ TBox $\models found(f1, S)$, where S is $do([backtrack(wyehia, home, f1), forward(home, mes, f1), find(f1, mes)], S0))$. Then, it is easy to see that the regressed query is

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

This example demonstrates that we managed to solve the projection problem in the presence of a general expressive static TBox. This example has been implemented: axioms are implemented in XML and regression of a query was computed using a C++ program, see details at `http://www.scs.ryerson.ca/mes/dl2012.zip`

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 [15, 16, 18]. 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 [18] 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 our 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 $F(\boldsymbol{t}, S_\alpha) \equiv \Phi_F(\boldsymbol{t}, \alpha, S_0)$, which are denoted as $\mathcal{D}_{SS}[\Omega]$, where $S_\alpha = do(\alpha, S_0)$, and $\Phi_F(\boldsymbol{t}, \alpha, S_0)$ is a z-free $\mathcal{L}$ sentence representing the RHS of a SSA for the fluent F. $F(\boldsymbol{t}, S_\alpha)$ and $\Phi_F(\boldsymbol{t}, \alpha, S_0)$ mention different situation terms. However, $F(\boldsymbol{t}, S_\alpha)$ 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 $F(\boldsymbol{t}, S_\alpha)$ simply because it is the value of F in a different situation. Consequently, we can replace $F(\boldsymbol{t}, S_\alpha)$ 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 $P(\boldsymbol{t_j})$ 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 [18], means the result of replacing every occurrence of an atom $P(\boldsymbol{x})$ in ϕ by $\bigvee_{j=1}^{m}(\boldsymbol{x} = \boldsymbol{t_j} \wedge \theta[P(\boldsymbol{t_j})]) \vee (\bigwedge_{j=1}^{m} \boldsymbol{x} \neq \boldsymbol{t_j}) \wedge P(\boldsymbol{x})$. 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:*

$$\bigwedge UNA \ \wedge \bigvee_{\theta \in \mathcal{M}(\Omega(S_0))} (\bigwedge \mathcal{D}_{S_0} \wedge \bigwedge \mathcal{D}_{SS}[\Omega])[\theta] \ \ (S_\alpha / S_0) \tag{3}$$

Proof: This is a consequence of Lemmas (2), (3) and Theorem 3.6 from [18]. 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.

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 $\mathcal{D}_{S_0}$. It turns out that progression is computationally

tractable if an initial $\mathcal{D}_{S_0}$ is in $proper^+$ form [18], where $proper^+$ theories generalize databases by allowing incomplete disjunctive knowledge about some of the named elements of the domain [14]. 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 $\mathcal{ALCO}(\mathcal{U})$, i.e., a boolean $\mathcal{ALC}$ ABox , then progression of $\mathcal{D}_{S_0}$ in our normal form can be computed efficiently, and the normal form can be maintained without introducing any new variables. To achieve this, we introduce a new p^+ normal form. We 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. The fact that forgetting in our normal form KB can be accomplished without introducing new variables is novelty that does not follow from [18]. Also, we show that after progression the size of the progressed KB is linear wrt to the size of of the initial KB, i.e., progression can be computed efficiently. Once an initial theory $\mathcal{D}_{S_0}$ has been progressed to $\mathcal{D}_{S_\alpha}$, solving the projection problem can be done using any $\mathcal{ALCO}(\mathcal{U})$ satisfiability solver.

Due to lack of space we omit all technical details, but the interested readers can find them in the longer version at `http://www.scs.ryerson.ca/mes/dl2012.zip`

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 [12, 13]. 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 $\mathcal{P}$ when solving the projection problem. To the best of our knowledge, this seamless integration of DLs and reasoning about actions has never been proposed before. For example, [4, 13] 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 [18]. 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 (not included in this version) about maintaining our new p^+ normal form after forgetting are proved using new techniques. They don't follow from [18], where progression was studied in FOL.

An approach to integrating DLs and reasoning about actions proposed in [4] inspired a number of subsequent papers including [13], where the reader can find extensive comparison and discussion. The approach proposed in [4] 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 [4] can answer projection queries only in local effect BATs. In any case, it is important to compare our implementations with an implementation based on [4] for the common classes of queries and theories. The first step in this direction is taken in [3]. Due to lack of space we couldn't discuss other related work, but all related publications are very extensively discussed in [4, 6, 13, 17].

References

1. Artale, A., Franconi, E.: A survey of temporal extensions of description logics. Ann. Math. Artif. Intell. 30(1-4), 171–210 (2000)
2. Baader, F., Horrocks, I., Sattler, U.: Description logics. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, pp. 135–179. Elsevier (2007)
3. Baader, F., Lippmann, M., Liu, H., Soutchanski, M., Yehia, W.: Experimental results on solving the projection problem in action formalisms based on description logics. In: Proc. of the 25th Intern. Workshop on Description Logics (2012)
4. Baader, F., Lutz, C., Miliĉić, M., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proceedings of the 20th AAAI Conference. pp. 572–577. Pittsburgh, PA, USA (2005), extended version is available as LTCS-Report-05-02 at http://lat.inf.tu-dresden.de/research/reports.html
5. Badea, L.: Planning in description logics: Deduction versus satisfiability testing. In: Proc. of the Intern. Workshop on Description Logics (1998)
6. Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Actions and programs over description logic ontologies. In: Proc. of the Intern. Workshop on Description Logics (2007)
7. Chang, L., Shi, Z., Qiu, L., Lin, F.: Dynamic description logic: Embracing actions into description logic. In: Proc. of the Intern. Workshop on Description Logics (2007)
8. Cuenca Grau, B., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P.F., Sattler, U.: OWL 2: The next step for OWL. J. Web Sem. 6(4), 309–322 (2008)
9. De Giacomo, G., Iocchi, L., Nardi, D., Rosati, R.: A theory and implementation of cognitive mobile robots. J. Log. Comput. 9(5), 759–785 (1999)
10. De Giacomo, G., Lenzerini, M.: PDL-based framework for reasoning about actions. In: Gori, M., Soda, G. (eds.) AI*IA. Lecture Notes in Computer Science, vol. 992, pp. 103–114. Springer (1995)
11. Devanbu, P.T., Litman, D.J.: Taxonomic plan reasoning. Artif. Intell. 84(1-2), 1–35 (1996)
12. Gu, Y.: Advanced Reasoning about Dynamical Systems. Ph.D. thesis, Department of Computer Science, University of Toronto, Canada (2010)
13. Gu, Y., Soutchanski, M.: A description logic based situation calculus. Ann. Math. Artif. Intell. 58(1-2), 3–83 (2010)
14. Lakemeyer, G., Levesque, H.J.: Evaluation-based reasoning with disjunctive information in first-order knowledge bases. In: Proc. of KR-02. pp. 73–81 (2002)
15. Lin, F., Reiter, R.: Forget it! In: Proceedings of the AAAI Fall Symposium on Relevance. pp. 154–159 (1994)
16. Lin, F., Reiter, R.: How to progress a database. Artificial Intelligence 92, 131–167 (1997)
17. Liu, H., Lutz, C., Milii, M., Wolter, F.: Reasoning about actions using description logics with general TBoxes. In: Logics in Artificial Intelligence, Lecture Notes in Computer Science, vol. 4160, pp. 266–279. Springer Berlin / Heidelberg (2006)
18. Liu, Y., Lakemeyer, G.: On first-order definability and computability of progression for local-effect actions and beyond. In: Boutilier, C. (ed.) IJCAI. pp. 860–866 (2009)
19. Lutz, C., Sattler, U.: A proposal for describing services with dls. In: Proc. of the 15th Intern. Workshop on Description Logics (2002)
20. McDermott, D.V.: The 1998 AI planning systems competition. AI Magazine 21(2), 35–55 (2000)
21. Pirri, F., Reiter, R.: Some contributions to the metatheory of the situation calculus. Journal of the ACM 46(3), 325–364 (1999)
22. Reiter, R.: Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems. The MIT Press (2001)
23. Wolter, F., Zakharyaschev, M.: Dynamic description logics. In: Advances in Modal Logic. pp. 431–446. CSLI Publications (1998)