

# Decidable Reasoning in a Modified Situation Calculus

**Yilan Gu**

Dept. of Computer Science  
University of Toronto  
10 King's College Road  
Toronto, ON, M5S 3G4, Canada  
Email: yilan@cs.toronto.edu

**Mikhail Soutchanski**

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

## Abstract

We consider a modified version of the situation calculus built using a two-variable fragment of the first-order logic extended with counting quantifiers. We mention several additional groups of axioms that can be introduced to capture taxonomic reasoning. We show that the regression operator in this framework can be defined similarly to regression in the Reiter's version of the situation calculus. Using this new regression operator, we show that the projection and executability problems are decidable in the modified version even if an initial knowledge base is incomplete and open. For an incomplete knowledge base and for context-dependent actions, we consider a type of progression that is sound with respect to the classical progression. We show that the new knowledge base resulting after our progression is definable in our modified situation calculus if one allows actions with local effects only. We mention possible applications to formalization of Semantic Web services.

## 1 Introduction

The situation calculus is a popular and well understood predicate logic language for reasoning about actions and their effects [Reiter, 2001]. It is used to provide a well-defined semantics for Web services and a foundation for a high-level programming language Golog [Reiter, 2001; McIlraith and Son, 2002]. However, because the situation calculus is formulated in a general predicate logic, reasoning about effects of sequences of actions is undecidable (unless some restrictions are imposed on the theory that axiomatizes the initial state of the world). The first motivation for our paper is intention to overcome this difficulty. We propose to use a two-variable fragment  $FO^2$  of the first-order logic (FOL) as a foundation for a modified situation calculus. Because the satisfiability problem in this fragment is known to be decidable (it is in  $NEXPTIME$ ), we demonstrate that by reducing reasoning about effects of actions to reasoning in this fragment, one can always guarantee decidability. The second motivation for our paper comes from description logics. Description Logics (DLs) [Baader *et al.*, 2003] are a well-known family

of knowledge representation formalisms, which play an important role in providing the formal foundations of several widely used Web ontology languages including OWL in the area of the Semantic Web. Many expressive DLs can be translated to  $FO^2$  and offer considerable expressive power going far beyond propositional logic, while ensuring that reasoning is decidable [Borgida, 1996]. DLs have been mostly used to describe static knowledge bases. However, several research groups consider formalization of actions using DLs or extensions of DLs. Following the key observation that reasoning about complex actions can be carried in a fragment of the propositional situation calculus, [Giacomo *et al.*, 1999] give an epistemic extension of DLs to provide a framework for the representation of dynamic systems. However, the representation and reasoning about actions in this framework are strictly propositional, which reduces the representation power of this framework. In [Baader *et al.*, 2005], Baader *et al.* provide another proposal for integrating description logics and action formalisms. They take the well known description logic *ALCOIO* (and its sub-languages) as foundation and show that the complexity of executability and projection problems (two basic reasoning problems for possibly sequentially composed actions) coincides with the complexity of standard DL reasoning. However, actions (services) are represented in their paper meta-theoretically, not as first-order (FO) terms. This can potentially lead to some complications when specifications of other reasoning tasks (e.g., planning) will be considered because it is not possible to quantify over actions in their framework. In our paper, we take a different approach and represent actions as FO terms, but achieve integration of taxonomic reasoning and reasoning about actions by restricting the syntax of the situation calculus and by introducing additional axioms to represent a taxonomy.

Because after doing longer and longer sequences of actions, solving projection problems becomes increasingly more difficult, it is beneficial to progress the initial incomplete knowledge base (KB) to represent the current state of the world. Then, the subsequent projection problems can be solved with respect to a new progressed KB. The task of computing a progressed KB is called the progression problem. Our paper is structured as follows. In Section 2, we briefly review the Reiter's situation calculus and the extension of  $FO^2$  with counting quantifiers. In Section 3 we discuss details of our proposal: a modified situation calculus. In Section

4 we consider an extension of regression (the main reasoning mechanism in the situation calculus). Finally, in Section 5 we discuss the progression problem and in Section 6 we discuss briefly other related approaches to reasoning about actions.

## 2 Background

The situation calculus (SC)  $\mathcal{L}_{sc}$  is a predicate language for axiomatizing dynamic systems. All dialects of the SC  $\mathcal{L}_{sc}$  include three disjoint sorts: *actions*, *situations* and *objects*. **Actions** are FO terms consisting of an action function symbol and its arguments. Actions change the world. **Situations** are FO terms which denote world histories. A distinguished constant  $S_0$  is used to denote the *initial situation*, and function  $do(a, s)$  denotes the situation that results from performing action  $a$  in situation  $s$ . Every situation corresponds uniquely to a sequence of actions. **Objects** are FO terms other than actions and situations that depend on the domain of application. **Fluents** are relations or functions whose values may vary from one situation to the next. Normally, a fluent is denoted by a predicate or function symbol whose last argument has the sort situation. For example,  $F(\vec{x}, do([\alpha_1, \dots, \alpha_n], S_0))$  represents a relational fluent in the situation  $do(\alpha_n, do(\dots, do(\alpha_1, S_0) \dots))$  resulting from execution of ground action terms  $\alpha_1, \dots, \alpha_n$  in  $S_0$ . We do not consider functional fluents in this paper.

The SC includes the distinguished predicate  $Poss(a, s)$  to characterize actions  $a$  that are possible to execute in  $s$ . For any SC formula  $\phi$  and a term  $s$  of sort situation, we say  $\phi$  is a formula *uniform* in  $s$  iff it does not mention the predicates  $Poss$ , it does not quantify over variables of sort situation, it does not mention equality on situations, and whenever it mentions a term of sort situation in the situation argument position of a fluent, then that term is  $s$  (see [Reiter, 2001]). If  $\phi(s)$  is a uniform formula and the situation argument is clear from the context, sometimes we suppress the situation argument and write this formula simply as  $\phi$ , and also introduce a notation  $\phi[s]$  to represent the SC formula obtained by restoring situation  $s$  back to all the fluents and/or  $Poss$  predicates (if any) in  $\phi$ . It is obvious that  $\phi[s]$  is uniform in  $s$ .

A *basic action theory* (BAT)  $\mathcal{D}$  in the SC is a set of axioms written in  $\mathcal{L}_{sc}$  with the following five classes of axioms to model actions and their effects [Reiter, 2001]: action precondition axioms  $\mathcal{D}_{ap}$ , successor state axioms (SSAs)  $\mathcal{D}_{ss}$ , initial theory  $\mathcal{D}_{S_0}$ , unique name axioms  $\mathcal{D}_{una}$ , domain independent foundational axioms for situations  $\Sigma$ .

Suppose that  $\mathcal{D} = \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \Sigma$  is a BAT,  $\alpha_1, \dots, \alpha_n$  is a sequence of ground action terms, and  $G(s)$  is a uniform formula with one free variable  $s$ . One of the most important reasoning tasks in the SC is the projection problem, that is, to determine whether  $\mathcal{D} \models G(do([\alpha_1, \dots, \alpha_n], S_0))$ . Another basic reasoning task is the executability problem. Let  $executable(do([\alpha_1, \dots, \alpha_n], S_0))$  be an abbreviation of the formula  $Poss(\alpha_1, S_0) \wedge \bigwedge_{i=2}^n Poss(\alpha_i, do([\alpha_1, \dots, \alpha_{i-1}], S_0))$ . Then, the executability problem is to determine whether  $\mathcal{D} \models executable(do([\alpha_1, \dots, \alpha_n], S_0))$ . Planning and high-level program execution are two important settings where the executability and projection problems arise natu-

rally. *Regression* is a central computational mechanism that forms the basis for automated solution to the executability and projection tasks in the SC [Reiter, 2001]. A recursive definition of the regression operator  $\mathcal{R}$  on any *regressible formula*  $\phi$  is given in [Reiter, 2001]. We use notation  $\mathcal{R}[\phi]$  to denote the formula that results from eliminating  $Poss$  atoms in favor of their definitions as given by action precondition axioms and replacing fluent atoms about  $do(\alpha, s)$  by logically equivalent expressions about  $s$  as given by SSAs repeatedly until it cannot make such replacement any further. The formula  $G(do([\alpha_1, \dots, \alpha_n], S_0))$  is a particularly simple example of a regressible formula because it is uniform in  $do([\alpha_1, \dots, \alpha_n], S_0)$ , but generally, regressible formulas can mention several different ground situation terms. The regression theorem shows that one can reduce the evaluation of a regressible formula  $\phi$  to a FOL theorem proving task in the initial theory together with unique names axioms for actions:

$$\mathcal{D} \models \phi \text{ iff } \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \models \mathcal{R}[\phi].$$

This fact is the key result for our paper: it demonstrates that an executability or a projection task can be reduced to a FOL theorem proving task. However, because  $\mathcal{D}_{S_0}$  is an arbitrary FO theory, this type of reasoning is undecidable. Two of the most common ways to overcome this difficulty is to introduce the closed world assumption or introduce the domain closure assumption (i.e., assume the domain is finite). In many application domains these assumptions are unrealistic. Therefore, we consider a version of the SC formulated in  $FO^2$ , or in  $C^2$ .

*Two-variable FO logic*  $FO^2$  is the fragment of ordinary FO logic (with equality), whose formulas only use no more than two variable symbols  $x$  and  $y$  (free or bound). *Two-variable FO logic with counting*  $C^2$  extends  $FO^2$  by allowing FO counting quantifiers  $\exists^{\geq m}$  and  $\exists^{\leq m}$  for all  $m \geq 1$ . [Pacholski *et al.*, 1997] show that satisfiability problem for  $C^2$  is decidable and recently [Pratt-Hartmann, 2005] proves that this problem is in NEXPTIME even when counting quantifiers are coded succinctly. See additional background on DLs and discussion of connections between DLs with  $C^2$  in [Baader *et al.*, 2003; Borgida, 1996; Gu and Soutchanski, 2006].

## 3 Modeling Dynamic Systems in a Modified Situation Calculus

In this section, we consider dynamic systems formulated in a modification of the language of the SC so that it can be considered as an extension to  $C^2$  (with an additional situation argument).<sup>1</sup> The key idea is to consider a syntactic modification of the SC such that the executability and projection problems are guaranteed to be decidable as a consequence of the decidability of the satisfiability problem in  $C^2$ . Moreover, since the modified SC has strong connections with description logics, which will be explained in detail below, we will denote this language as  $\mathcal{L}_{sc}^{DL}$ .

First of all, the three sorts in  $\mathcal{L}_{sc}^{DL}$  (i.e., actions, situations and objects) are the same as those in  $\mathcal{L}_{sc}$ , except that they

<sup>1</sup>The reason that we call it a "modified" SC rather than a "restricted" SC is that we extend the SC with other features, such as adding acyclic TBox axioms to basic action theories.

obey the following restrictions: (1) all terms of sort *object* are variables ( $x$  and  $y$ ) or constants, i.e., object functional symbols are not allowed; (2) all action functions include no more than two arguments. Each argument of any term of sort *action* is either a constant or an *object* variable ( $x$  or  $y$ ); (3) variable  $s$  of sort *situation* and/or variable  $a$  of sort *action* are the only additional variables being allowed in  $\mathcal{D} - \Sigma - \mathcal{D}_{una}$  in addition to variables  $x, y$ .

Second, any fluent in  $\mathcal{L}_{sc}^{DL}$  is a predicate either with two or with three arguments (including the one of sort situation). We call fluents with two arguments (*dynamic*) *concepts*, and call fluents with three arguments (*dynamic*) *roles*. In  $\mathcal{L}_{sc}^{DL}$ , (*static*) *concepts* (i.e., unary predicates with no situation argument) and (*static*) *roles* (i.e., binary predicates with no situation argument), if any, are considered as unchangeable taxonomic properties and unchangeable classes of an application domain. Moreover, each concept (static or dynamic) can be either *primitive* or *defined*.

Third, apart from the standard FO logical symbols  $\wedge, \vee$  and  $\exists$ , with the usual definition of a full set of connectives and quantifiers,  $\mathcal{L}_{sc}^{DL}$  also includes counting quantifiers  $\exists \geq^m$  and  $\exists \leq^m$  for all  $m \geq 1$ . Equality  $=$  is allowed in  $\mathcal{L}_{sc}^{DL}$ .

The dynamic systems we are dealing with here satisfy the *open world assumption* (OWA): what is not stated explicitly is currently unknown rather than false. In this paper, the dynamic systems we are interested in can be formalized as a *basic action theory* (BAT)  $\mathcal{D}$  using the following seven groups of axioms in  $\mathcal{L}_{sc}^{DL}$ :  $\mathcal{D} = \Sigma \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss} \cup \mathcal{D}_T \cup \mathcal{D}_R \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ . Five of them ( $\Sigma, \mathcal{D}_{ap}, \mathcal{D}_{ss}, \mathcal{D}_{una}, \mathcal{D}_{S_0}$ ) are similar to those groups in a BAT in  $\mathcal{L}_{sc}$ , and the other two ( $\mathcal{D}_T, \mathcal{D}_R$ ) are introduced to axiomatize description logic related facts and properties (see below). However, because  $\mathcal{L}_{sc}^{DL}$  allows only two object variables, all axioms must conform to the following additional requirements.

**Action precondition axioms**  $\mathcal{D}_{ap}$ : For each action  $A$  in  $\mathcal{L}_{sc}^{DL}$ , there is one axiom of the form  $Poss(A, s) \equiv \Pi_A[s]$  (or  $Poss(A(x), s) \equiv \Pi_A(x)[s]$ , or  $Poss(A(x, y), s) \equiv \Pi_A(x, y)[s]$ , respectively), if  $A$  is an action constant (or unary, or binary action term, respectively), where  $\Pi_A$  (or  $\Pi_A(x)$ , or  $\Pi_A(x, y)$ , respectively) is a  $C^2$  formula with no free variables (or with at most  $x$ , or with at most  $x, y$  as the only free variables, respectively). This set of axioms characterize the preconditions of all actions.

**Successor state axioms**  $\mathcal{D}_{ss}$ : There are two types of fluents in  $\mathcal{L}_{sc}^{DL}$ : *primitive dynamic concepts* of the form  $F(x, s)$  (fluents with exactly one non-situation argument) and *primitive dynamic roles* of the form  $F(x, y, s)$  (fluents with exactly two non-situation arguments). Let variable vector  $\vec{x}$  to be  $x$ , or  $y$ , or  $\langle x, y \rangle$ ; a SSA is specified 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 the axiom has the form

$$F(\vec{x}, do(a, s)) \equiv \psi_F(\vec{x}, a, s), \quad (1)$$

where the general structure of  $\psi_F(\vec{x}, a, s)$  is as follows:

$$F(\vec{x}, s) \wedge \neg(\bigvee_{j=1}^{m_1} [\exists x][\exists y](a = A_j^-(\vec{x}(j, 0, -)) \wedge \phi_j^-(\vec{x}(j, 1, -))][s]),$$

where each variable vector  $\vec{x}(i, n, b)$  (or  $\vec{x}(j, n, b)$  respectively) ( $i = 1..m_0, j = 1..m_1, n \in \{0, 1\}, b \in \{+, -\}$ ) represents a vector of object variables, which can be empty,  $x, y, \langle x, y \rangle$  or  $\langle y, x \rangle$ . Moreover,  $[\exists x]$  or  $[\exists y]$  represents that the quantifier included in  $[\ ]$  is optional; and each  $\phi_i^+(\vec{x}(i, 1, +))$ ,  $i = 1..m_0$  ( $\phi_i^-(\vec{x}(j, 1, -))$ ,  $j = 1..m_1$ , respectively), is a  $C^2$  formula with variables (both free and quantified) among  $x$  and  $y$ . Note that when  $m_0$  (or  $m_1$  respectively) is equal to 0, the corresponding disjunctive subformula is equivalent to *false*.

**Acyclic TBox axioms**  $\mathcal{D}_T$ : Similar to the TBox axioms in DL, we may define new concepts using TBox axioms. Any group of TBox axioms  $\mathcal{D}_T$  may include two sub-classes: static TBox  $\mathcal{D}_{T, st}$  and dynamic TBox  $\mathcal{D}_{T, dyn}$ . Every formula in static TBox is a *concept definition* formula of the form

$$G(x) \equiv \phi_G(x),$$

where  $G$  is a unary predicate symbol and  $\phi_G(x)$  is a  $C^2$  formula in the domain with free variable  $x$ , and there is no fluent in it. Every formula in dynamic TBox is a *concept definition* formula of the form  $G(x, s) \equiv \phi_G(x)[s]$ ,

where  $\phi_G(x)$  is a  $C^2$  formula with free variable  $x$ , and there is at least one fluent in it. All the concepts appeared in the left-hand side of TBox axioms are called *defined* concepts. We also require that the set of TBox axioms must be acyclic (acyclicity in  $\mathcal{D}_T$  is defined exactly as it is defined for TBox).

**RBox axioms**  $\mathcal{D}_R$ : Similar to the idea of RBox in DL, we may also specify a group of axioms, called RBox axioms below, to support a role taxonomy. Each role inclusion axiom is represented as

$$R_1(x, y)[s] \supset R_2(x, y)[s],$$

where  $R_1$  and  $R_2$  are primitive roles (either static or dynamic). If these axioms are included in the BAT  $\mathcal{D}$ , then it is assumed that  $\mathcal{D}$  is specified correctly in the sense that the meaning of any RBox axiom included in the theory is correctly compiled into SSAs. That is, one can prove by induction that  $(\mathcal{D} - \mathcal{D}_R) \models \forall s. R_1(x, y)[s] \supset R_2(x, y)[s]$ . Although RBox axioms are not used by the regression operator, they are used for taxonomic reasoning in the initial theory.

**Initial theory**  $\mathcal{D}_{S_0}$ : It is a finite set of  $C^2$  sentences (assuming that we suppress the only situation term  $S_0$  in all fluents). It specifies the incomplete information about the initial problem state and also describes all the facts that are not changeable over time in the domain of an application. In particular, it includes static TBox axioms  $\mathcal{D}_{T, st}$  as well as RBox axioms in the initial situation  $S_0$  (if any). In addition,  $\mathcal{D}_{S_0}$  also includes all unique name axioms for object constants.

The remaining two classes (foundational axioms for situations  $\Sigma$  and unique name axioms for actions  $\mathcal{D}_{una}$ ) are the same as those in the BATs of the usual SC. Note that these axioms (as well as  $\mathcal{D}_{ap}$  and  $\mathcal{D}_{ss}$ ) use more than two variables (e.g.,  $\mathcal{D}_{ss}$  use action and situation variables in addition to object variables), but we will see in the next section, that these axioms will be eliminated in the process of regressing a regressable formula to a sentence that will use no more than two object variables and no other variables.

## 4 Modified Regression with Lazy Unfolding

After giving the definition of what the BAT in  $\mathcal{L}_{sc}^{DL}$  is, we turn our attention to the reasoning tasks.

Given a formula  $W$  of  $\mathcal{L}_{sc}^{DL}$  in the domain  $\mathcal{D}$ , the definition of  $W$  being regressable (called  $\mathcal{L}_{sc}^{DL}$  regressable below) is slightly different from the definition of  $W$  being regressable in  $\mathcal{L}_{sc}$  (see [Reiter, 2001]) by adding the following two conditions: (i) any variable (free or bounded) in  $W$  is either  $x$  or  $y$ ; (ii) every term of sort situation in  $W$  is ground. Moreover, in  $\mathcal{L}_{sc}^{DL}$  we have to be more careful with the definition of the regression operator  $\mathcal{R}$  for two main reasons. First, to deal with TBox we have to extend regression. For a  $\mathcal{L}_{sc}^{DL}$  regressable formula  $W$ , we *extend* below the regression operator defined in [Reiter, 2001] with the *lazy unfolding technique* (see [Baader *et al.*, 2003]) to expand defined dynamic concepts. We still denote such operator as  $\mathcal{R}$ . Second,  $\mathcal{L}_{sc}^{DL}$  uses only two object variables and we have to make sure that after regressing a fluent atom we still get a  $\mathcal{L}_{sc}^{DL}$  formula, i.e., that we never need to introduce new (free or bound) object variables. To deal with the two-variable restriction, we modify our regression operator  $\mathcal{R}$  in comparison to the conventional operator defined in [Reiter, 2001]. For example, when replacing  $Poss$  atom or fluent atoms about  $do(\alpha, \sigma)$ , the definition of the conventional regression operator in [Reiter, 2001] has the assumption that the quantified variables in the right-hand side of the corresponding axioms should be renamed to new variables different from the free variables in the atoms that to be replaced. This assumption of using *new* variables for renaming assures equivalence of original formula and the formula after regression. To avoid introducing new variables (as required by the Reiter's regression operator) and to assure defined dynamic concepts being handled, we modify the regression operator for each  $\mathcal{L}_{sc}^{DL}$  regressable formula. Possibility of reusing variables is guaranteed by the general format of the SSAs given in the previous section and the additional condition (ii) in the definition of the  $\mathcal{L}_{sc}^{DL}$  regressable formula.

The complete formal definition of our  $\mathcal{R}$  is as follows, where  $\sigma$  denotes the term of sort situation, and  $\alpha$  denotes the term of sort action.

- If  $W$  is not atomic, i.e.  $W$  is of the form  $W_1 \vee W_2$ ,  $W_1 \wedge W_2$ ,  $\neg W'$ ,  $Qv.W'$  where  $Q$  represents a quantifier (including counting quantifiers) and  $v$  represents a variable symbol, then

$$\begin{aligned} \mathcal{R}[W_1 \vee W_2] &= \mathcal{R}[W_1] \vee \mathcal{R}[W_2], & \mathcal{R}[\neg W'] &= \neg \mathcal{R}[W'], \\ \mathcal{R}[W_1 \wedge W_2] &= \mathcal{R}[W_1] \wedge \mathcal{R}[W_2], & \mathcal{R}[Qv.W'] &= Qv.\mathcal{R}[W']. \end{aligned}$$

- Otherwise,  $W$  is atom. There are several cases.
  - a. If  $W$  is of the form  $A_1(\vec{t}) = A_2(\vec{t}')$  for some action function symbols  $A_1$  and  $A_2$ , then by using axioms in  $\mathcal{D}_{una}$ ,<sup>2</sup> we define the regression of  $W$  as

$$\mathcal{R}[W] = \begin{cases} \perp & \text{if } A_1 \neq A_2, \\ \bigwedge_{i=1}^{|\vec{t}|} t_i = t'_i & \text{otherwise.} \end{cases}$$

Otherwise, if  $W$  is situation independent atom (including equality between object constants or variables), or  $W$  is a concept or role uniform in  $S_0$ , then

$$\mathcal{R}[W] = W.$$

- b. If  $W$  is a regressable  $Poss$  atom, so it has the form

<sup>2</sup>Notice that the action functions with different number of arguments always use different function symbols (i.e., different names).

$Poss(A(\vec{t}), \sigma)$ , for terms of sort action and situation respectively in  $\mathcal{L}_{sc}^{DL}$ . Then there must be an action precondition axiom for  $A$  of the form  $Poss(A(\vec{x}), s) \equiv \Pi_A(\vec{x}, s)$ , where the argument  $\vec{x}$  of sort object can either be empty (i.e.,  $A$  is an action constant), a single variable  $x$  or two-variable vector  $\langle x, y \rangle$ . Because of the syntactic restrictions of  $\mathcal{L}_{sc}^{DL}$ , each term in  $\vec{t}$  can only be a variable  $x, y$  or a constant  $C$ . Then,

$$\mathcal{R}[W] = \begin{cases} \mathcal{R}[(\exists y)(x = y \wedge \Pi_A(x, y, \sigma))] & \text{if } \vec{t} = \langle x, x \rangle, \\ \mathcal{R}[(\exists x)(y = x \wedge \Pi_A(x, y, \sigma))] & \text{else if } \vec{t} = \langle y, y \rangle, \\ \mathcal{R}[\Pi_A(\vec{t}, \sigma)] & \text{else if } \vec{t} = \langle x, C \rangle \text{ or} \\ & \vec{t} = \langle x, y \rangle \text{ or } \vec{t} = x, \\ \mathcal{R}[\widetilde{\Pi}_A(\vec{t}, \sigma)] & \text{otherwise,} \end{cases}$$

where  $C$  is a constant and  $\widetilde{\phi}$  denotes a *dual formula* for formula  $\phi$  obtained by replacing every variable symbol  $x$  (free or quantified) with variable symbol  $y$  and replacing every variable symbol  $y$  (free or quantified) with variable symbol  $x$  in  $\phi$ , i.e.,  $\widetilde{\phi} = \phi[x/y, y/x]$ .

- c. If  $W$  is a defined dynamic concept, so it has the form  $G(t, \sigma)$  for some object term  $t$  and situation term  $\sigma$ , and there must be a TBox axiom for  $G$  of the form  $G(x, s) \equiv \phi_G(x, s)$ . Because of the restrictions of the language  $\mathcal{L}_{sc}^{DL}$ , term  $t$  can only be a variable  $x, y$  or a constant. Then, we use lazy unfolding technique as follows:

$$\mathcal{R}[W] = \begin{cases} \mathcal{R}[\phi_G(t, \sigma)] & \text{if } t \text{ is not variable } y, \\ \mathcal{R}[\phi_G(y, \sigma)] & \text{otherwise.} \end{cases}$$

- d. If  $W$  is a primitive concept (a primitive role, respectively), so it has the form  $F(t_1, do(\alpha, \sigma))$  or  $F(t_1, t_2, do(\alpha, \sigma))$  for some terms  $t_1$  (and  $t_2$ ) of sort object, term  $\alpha$  of sort action and term  $\sigma$  of sort situation. There must be a SSA for fluent  $F$  such as Eq. (1). Because of the restriction of the language  $\mathcal{L}_{sc}^{DL}$ , the term  $t_1$  and  $t_2$  can only be a variable  $x, y$  or a constant  $C$  and  $\alpha$  can only be an action function with no more than two arguments of sort object. Then, when  $W$  is a concept,

$$\mathcal{R}[W] = \begin{cases} \mathcal{R}[\psi_F(t_1, \alpha, \sigma)] & \text{if } t_1 \text{ is not variable } y, \\ \mathcal{R}[\psi_F(y, \alpha, \sigma)] & \text{otherwise, i.e., if } t_1 = y; \end{cases}$$

and, when  $W$  is a role,

$$\mathcal{R}[W] = \begin{cases} \mathcal{R}[(\exists y)(x = y \wedge \psi_F(x, y, \alpha, \sigma))] & \text{if } t_1 = x, t_2 = x; \\ \mathcal{R}[(\exists x)(y = x \wedge \psi_F(x, y, \alpha, \sigma))] & \text{if } t_1 = y, t_2 = y; \\ \mathcal{R}[\psi_F(y, x, \alpha, \sigma)] & \text{if } t_1 = y, t_2 = x; \\ & \text{or } t_1 = y, t_2 = C; \\ \mathcal{R}[\psi_F(t_1, t_2, \alpha, \sigma)] & \text{otherwise.} \end{cases}$$

Based on the above definition, we are able to prove the following theorems.

**Theorem 1** *Suppose  $W$  is a  $\mathcal{L}_{sc}^{DL}$  regressable formula, then the regression  $\mathcal{R}[W]$  defined above terminates in a finite number of steps.*

**Proof:** Immediately follows from conditions (i) and (ii) of the definition of  $\mathcal{L}_{sc}^{DL}$  regressable formula, acyclicity of the TBox axioms, and from the assumption that  $RBox$  axioms are compiled into the SSAs and consequently do not participate in regression.  $\square$

**Theorem 2** *Suppose  $W$  is a  $\mathcal{L}_{sc}^{DL}$  regressable formula with the background basic action theory  $\mathcal{D}$ . Then,  $\mathcal{R}[W]$  is a  $\mathcal{L}_{sc}^{DL}$  formula uniform in  $S_0$  with no more than two variables ( $x$*

and  $y$ ). Moreover,  $\mathcal{D} \models W \equiv \mathcal{R}[W]$ , and  
 $\mathcal{D} \models W$  iff  $\mathcal{D}_{S_0} \models \mathcal{R}[W]$ .

**Proof:** According to the definition of the modified regression operator, prove by induction over the structure of  $W$ . The first statement holds because all replacements done by  $\mathcal{R}$  transform  $W$  to logically equivalent formula. The second statement follows from the regression theorem in [Reiter, 2001].  $\square$

**Theorem 3** Suppose  $W$  is a  $\mathcal{L}_{sc}^{DL}$  regressive formula with the background basic action theory  $\mathcal{D}$ . Then, the problem whether  $\mathcal{D} \models W$  is decidable.

**Proof:** According to Theorem 2,  $\mathcal{D} \models W$  iff  $\mathcal{D}_{S_0} \models \mathcal{R}[W]$ , where  $\mathcal{R}[W]$  and the axioms in  $\mathcal{D}_{S_0}$  are  $C^2$  formulas. Therefore, the problem whether  $\mathcal{D} \models W$  is equivalent to whether  $\mathcal{D}_{S_0} \wedge \neg\mathcal{R}[W]$  is unsatisfiable or not, which is a decidable problem, according to the fact that the satisfiability problem in  $C^2$  is decidable.  $\square$

This theorem is important because it guarantees that the projection and executability problems in  $\mathcal{L}_{sc}^{DL}$  are decidable even if the initial KB  $\mathcal{D}_{S_0}$  is incomplete. [Gu and Soutchaniski, 2006] give some detailed examples that illustrate the basic reasoning tasks described above and reduction techniques for dealing with properties that need more than two variables, and show that using  $\mathcal{L}_{sc}^{DL}$ , one can model realistic dynamic domains such as school enrollment services and on-line shopping services.

We say that the SSA for a fluent  $F$  is *context-free* if the SSA for  $F$  has the form

$$F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a).$$

Then, we have the following theorem about the complexity analysis for reasoning about projection problem.

**Theorem 4** Given a basic action theory  $\mathcal{D}$  in  $\mathcal{L}_{sc}^{DL}$ , suppose that the SSA for a fluent  $F$  is context-free, then the computational complexity of answering the queries of the form  $F(\vec{X}, \sigma)$  is co-NEXPTIME, where  $\vec{X}$  is a vector of object constants and  $\sigma$  is a ground situation term.

**Proof:** The result follows from the complexity analysis of projection problem in [Reiter, 2001] (Chapter 4), Theorem 3, and the theorem in [Pratt-Hartmann, 2005] that the satisfiability problem in  $C^2$  is decidable in NEXPTIME.  $\square$

## 5 Progression of CNF-based KBs

The *progression* problem (also known as filtering and update) is how to compute the new theory in response to a given sequence of actions. In this section, we consider the progression problem for KBs in language  $\mathcal{L}_{sc}^{DL}$ . In this section, let  $\overline{\mathcal{D}} = \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \Sigma \cup \mathcal{D}_{una}$ .

A formal definition of (classical) progression is given in [Reiter, 2001]. A set of sentences  $\mathcal{D}_{S_\alpha}$  is the (classical) progression of the initial KB  $\mathcal{D}_{S_0}$  (wrt basic action theory  $\mathcal{D}$ ) after performing a ground action  $\alpha$  in the situation  $S_0$  iff  $\mathcal{D}_{S_\alpha}$  is uniform in  $do(\alpha, S_0)$ ,  $\mathcal{D} \models \mathcal{D}_{S_\alpha}$ , and for every model  $M_\alpha$  of  $\overline{\mathcal{D}} \cup \mathcal{D}_{S_\alpha}$ , there is a model  $M$  of  $\mathcal{D}$  such that  $M$  and  $M_\alpha$  have the same domain and interpret situation independent predicates, function symbols,  $Poss$  and all fluents about the future of  $do(\alpha, S_0)$  identically (in the sequel, we say that  $M$  and  $M_\alpha$

have a *progression relationship*). The progression can be iteratively repeated if the progressed KB has the same format as the initial KB and we can consider the computed progression as the new initial KB at the next iteration. [Lin and Reiter, 1997] shows that the (classical) progression of a finite FO KB is not always FOL definable (but it is always definable in the second-order logic). By using an example similar to [Lin and Reiter, 1997], one can prove

**Theorem 5** Progression of a theory in  $\mathcal{L}_{sc}^{DL}$  is not always FO definable, therefore it is definitely not definable in  $\mathcal{L}_{sc}^{DL}$ .

**Proof:** We consider the theory  $\mathcal{D}_1$  obtained by modifying the theory  $\mathcal{D}$  given in [Lin and Reiter, 1997] as follows: (1) replace one constant symbol 0 in  $\mathcal{D}$  by an infinite set of constant symbols  $\{0, 1, 2, \dots\}$ ; (2) replace function symbol  $succ(x) = y$  in  $\mathcal{D}$  by predicate  $succ(x, y)$  which will be true iff  $y$  is the successor of  $x$ ; (3) replace the empty initial KB by the new  $\mathcal{D}_{S_0}$  which includes infinitely many axioms of the form  $c_1 \neq c_2$  for any non-identical constant symbols  $c_1$  and  $c_2$  given above and of the form  $succ(c, c')$  where constant  $c'$  is the successor of constant  $c$  in the sense of natural numbers. The rest of the proof is exactly the same as the proof given in [Lin and Reiter, 1997].  $\square$

Notice that the proof assumes that an  $\mathcal{L}_{sc}^{DL}$  theory  $\mathcal{D}_1$  is infinite. The problem whether progression of a finite theory in  $\mathcal{L}_{sc}^{DL}$  is always FO definable remains open.

Now, we consider a (weaker than classical) *modified progression* for certain type of incomplete KBs only. For this special case of incomplete KBs, we show below that a modified progression of a KB is in  $\mathcal{L}_{sc}^{DL}$  and it is sound wrt a classical progression of this KB.

First, we restrict the syntactic form of the KBs that are allowed. We use  $e$  to range over *ewffs*, that is, quantifier-free boolean formulas with equalities and inequalities only. For any vector  $\vec{x}$  that is  $\langle x, y \rangle$  (or a variable  $x$ , or  $y$ ) and any vector of object constants  $\vec{B}$  that is  $\langle B_1, B_2 \rangle$  (or a single constant  $B$ ), we write  $\vec{x} = \vec{B}$  as an abbreviation for  $x = B_1 \wedge y = B_2$  (or  $x = B$ , respectively). We use  $l(\vec{x}, S)$  to range over fluent *literals*, where  $S$  is a ground situation term. We call formulas of the form  $\forall \vec{x}. e(\vec{x}) \supset l(\vec{x}, S)$  *equality-based formulas*. We define a *CNF-based* KB  $\mathcal{D}_S = \mathcal{KB}_I \cup \mathcal{KB}_S$ , where  $\mathcal{KB}_I$  is a set of situation-independent formulas (including unique name axioms for object constants, i.e.,  $\mathcal{KB}_I$  is a subset of  $\mathcal{D}_{S_0}$ ), and  $\mathcal{KB}_S$  is a finite set of sentences uniform in  $S$ , where each sentence (also called *clause* below) is a disjunction of finitely many equality-based formulas. In particular,  $\mathcal{D}_{S_0} = \mathcal{KB}_I \cup \mathcal{KB}_{S_0}$ . A CNF formula composed from ground fluent literals uniform in  $S$  is a simple example of  $\mathcal{KB}_S$ .

Secondly, we consider an action theory  $\mathcal{D}$  that is *local-effect*. Let a SSA of a fluent  $F$  have the syntactic form  $F(\vec{x}, do(a, s)) \equiv \gamma_F^+(\vec{x}, a, s) \vee F(\vec{x}, s) \wedge \neg\gamma_F^-(\vec{x}, a, s)$ . This SSA is *local-effect* if  $\gamma_F^+(\vec{x}, a, s)$  and  $\gamma_F^-(\vec{x}, a, s)$  are disjunctions of formulas of the form  $\exists \vec{z}[a = A(\vec{y}) \wedge \phi(\vec{y}, s)]$ , where  $A$  is a action function,  $\vec{y}$  contains  $\vec{x}$ ,  $\vec{z} = \vec{y} - \vec{x}$ , and  $\phi$  is quantifier-free. An action theory  $\mathcal{D}$  is local-effect if each SSA in  $\mathcal{D}$  is local-effect. Let  $*$  be  $+$  or  $-$ . Consider a ground action  $\alpha$  and a fluent  $F_i$  in a local-effect action theory  $\mathcal{D}$  of language  $\mathcal{L}_{sc}^{DL}$ , then it is easy to see that the right-hand side (RHS) of SSA for  $F_i(\vec{x}, do(\alpha, s))$  has the following syntactic

form:  $\bigvee_{k=1}^{m_i^+} (\vec{x} = \vec{B}_{ik}^+ \wedge \phi_{ik}^+(s)) \vee F_i(\vec{x}, s) \wedge \neg \bigvee_{k=1}^{m_i^-} (\vec{x} = \vec{B}_{ik}^- \wedge \phi_{ik}^-(s))$ , where each  $m_i^*$  is a natural number, the variables in  $\vec{x}$  are among  $x$  and  $y$ , and for each  $k$ ,  $\phi_{ik}^*(s)$  is a propositional formula. This SSA is a special case of the generic SSA (1) from Section 3. Moreover, according to Theorem 3 the problem whether  $\mathcal{D} \models \phi_{ik}^*(S)$  is decidable for each  $k$  and any ground situation  $S$ . We define the following abbreviations.

$$\begin{aligned} Add(F_i, \alpha, S) &\stackrel{def}{=} \bigvee_{\{k \in 1..m_i^+ \mid D \models \phi_{ik}^+(S)\}} \vec{x} = \vec{B}_{ik}^+, \\ Add(\neg F_i, \alpha, S) &\stackrel{def}{=} \bigvee_{\{k \in 1..m_i^- \mid D \models \phi_{ik}^-(S)\}} \vec{x} = \vec{B}_{ik}^-, \\ Delete(F_i, \alpha, S) &\stackrel{def}{=} \bigvee_{k=1..m_i^-} \vec{x} = \vec{B}_{ik}^- \wedge \neg \bigvee_{\{k \mid D \models \neg \phi_{ik}^-(S)\}} \vec{x} = \vec{B}_{ik}^-, \\ Delete(\neg F_i, \alpha, S) &\stackrel{def}{=} \bigvee_{k=1..m_i^+} \vec{x} = \vec{B}_{ik}^+ \wedge \neg \bigvee_{\{k \mid D \models \neg \phi_{ik}^+(S)\}} \vec{x} = \vec{B}_{ik}^+ \end{aligned}$$

When there is no disjunct satisfying the condition on the context formula  $\phi_{ik}^*$ , the corresponding disjunction is equivalent to  $\perp$ . It is easy to see that all formulas above are ewffs.

The intuition behind these abbreviations is simple. For instance,  $Add(F_i, \alpha, S)$  is a collection of all those cases when  $F_i$  will become true in every model if the context  $\phi_{ik}^+(S)$  holds in  $S$ . Therefore, these cases provide support for adding the fluent  $F_i$  to the new KB. If one takes all those cases  $\vec{x} = \vec{B}_{ik}^-$  when  $F_i$  ceases to be true in some models (where contexts might or might not be entailed) and removes from them those cases when negations of contexts are known to be entailed (i.e., remove models where it is guaranteed that  $F_i$  will not cease to be true), then as a result one gets  $Delete(F_i, \alpha, S)$  that represents the collections of all those objects for which  $F_i$  has to be deleted from the current KB.

Now consider a ground action  $\alpha$  and a CNF-based KB  $\mathcal{D}_S = \mathcal{KB}_I \cup \mathcal{KB}_S$  as the current KB of a local-effect BAT  $\mathcal{D} = \overline{\mathcal{D}} \cup \mathcal{D}_S$ . Assume that  $\mathcal{D} \models Poss(\alpha, S)$ , otherwise there is no need in progression. We provide an algorithm to compute a variant of progression, called a *modified progression of  $\mathcal{D}_S$  wrt  $\overline{\mathcal{D}}$*  and the ground action  $\alpha$  executed in  $S$ , and denote the resulting KB as  $\mathcal{P}(\alpha, \mathcal{D}_S)$ .

Let  $\mathcal{P}(\alpha, \mathcal{D}_S)$  be the following set of sentences:

1. Initialize  $\mathcal{P}(\alpha, \mathcal{D}_S)$  to  $\mathcal{KB}_I \cup \{(\forall \vec{x}) Add(F_i, \alpha, S) \supset F_i(\vec{x}, do(\alpha, S)) \mid Add(F_i, \alpha, S) \neq \perp\} \cup \{(\forall \vec{x}) Add(\neg F_i, \alpha, S) \supset \neg F_i(\vec{x}, do(\alpha, S)) \mid Add(\neg F_i, \alpha, S) \neq \perp\}$ .
2. For each clause in  $\mathcal{KB}_S$  of the form  $C_j = p_1 \vee \dots \vee p_h \vee \dots$ , where each  $p_h$  is an equality-based formula  $(\forall \vec{x}) e_{j_h}(\vec{x}) \supset l_{j_h}(\vec{x}, S)$  and  $l_{j_h}$  is either  $F_i$  or its negation  $\neg F_i$ , we update this clause as follows.
  - (a) Initialize a temporary set  $T = \emptyset$ .
  - (b) For each  $p_h$ , if  $\mathcal{KB}_I \cup \{(\forall \vec{x}) e_{j_h}(\vec{x}) \wedge \neg Delete(l_{j_h}, \alpha, S)\} \not\models \perp$ , add  $(\forall \vec{x}) e_{j_h}(\vec{x}) \wedge \neg Delete(l_{j_h}, \alpha, S) \supset l_{j_h}(\vec{x}, do(\alpha, S))$  into the set  $T$ .
  - (c) If  $T \neq \emptyset$ , add a new clause  $C'_j = \bigvee_{p \in T} p$  to  $\mathcal{P}(\alpha, \mathcal{D}_S)$ ; otherwise, do nothing (i.e., replace  $C_j$  with  $\top$ ).

It is easy to see that the resulting KB  $\mathcal{P}(\alpha, \mathcal{D}_S)$  remains to be a CNF-based KB in  $\mathcal{L}_{sc}^{DL}$ . Therefore, such progression can be repeated for the next ground action, say  $\alpha'$ .

The intuition behind this algorithm is simple. The succes-

sor model of the KB after performing a ground action at the current situation should keep all the situation-independent information, add truth values for each fluent for those objects where it will definitely become true (or false), and also keep the remaining consistent information by removing conflicting knowledge for objects from the current KB. Note that we detect conflicts between cases supported by  $e_{j_h}$  and cases included in the  $Delete(l_{j_h}, \alpha, S)$  condition by using unique name axioms for constants in  $\mathcal{KB}_I$ , if necessary, to solve the entailment problem in (b).

For any given BAT  $\mathcal{D} = \overline{\mathcal{D}} \cup \mathcal{D}_S$  and a ground action  $\alpha$ , we say that a modified progression  $\mathcal{P}(\alpha, \mathcal{D}_S)$  is (*classically*) *sound* if any model of the classical progression of  $\mathcal{D}_S$  (wrt  $\alpha$  and  $\overline{\mathcal{D}}$ ) is a model of the modified progression. Also, we say that  $\mathcal{P}(\alpha, \mathcal{D}_S)$  is (*classically*) *complete* if every model of  $\overline{\mathcal{D}} \cup \mathcal{P}(\alpha, \mathcal{D}_S)$  is a model of the classical progression of  $\mathcal{D}_S$  (wrt  $\alpha$  and  $\overline{\mathcal{D}}$ ). The modified progression has the following nice properties.

**Theorem 6** *Given a BAT  $\mathcal{D}$  with the current KB  $\mathcal{D}_S$  and a ground action  $\alpha$  then*

- (1) *If  $\mathcal{D}$  is consistent and RHS of SSAs are consistent, then the modified progression  $\mathcal{P}(\alpha, \mathcal{D}_S)$  is also consistent.*
- (2) *The modified progression  $\mathcal{P}(\alpha, \mathcal{D}_S)$  is (classically) sound.*

**Proof:** (1) Prove by cases using the definition of  $\mathcal{P}(\alpha, \mathcal{D}_S)$ . (2) According to the definition of  $\mathcal{P}(\alpha, \mathcal{D}_S)$ , prove that  $\mathcal{D} \models \mathcal{P}(\alpha, \mathcal{D}_S)$  using the RHS of the SSAs for  $F_i(\vec{x}, do(\alpha, S))$ . Let  $M_\alpha$  be any model of the (classical) progression of  $\mathcal{D}_S$ . Then, there is a model  $M$  of  $\mathcal{D}$  such that  $M$  and  $M_\alpha$  have a progression relationship. Then  $M$  is also a model of  $\mathcal{P}(\alpha, \mathcal{D}_S)$  by  $\mathcal{D} \models \mathcal{P}(\alpha, \mathcal{D}_S)$ , and then since  $M$  and  $M_\alpha$  have a progression relationship we can prove that  $M_\alpha$  is a model of any sentence uniform in  $do(\alpha, S)$  iff  $M$  is a model of any sentence uniform in  $do(\alpha, S)$ . Therefore, we conclude that  $M_\alpha$  is a model of  $\mathcal{P}(\alpha, \mathcal{D}_S)$  because  $\mathcal{P}(\alpha, \mathcal{D}_S)$  is a set of sentences uniform in  $do(\alpha, S)$ .  $\square$

## 6 Discussion and Future Work

The major consequence of the results proved above for the problem of service composition is the following. If both atomic services and properties of the world that can be affected by these services have no more than two parameters, then we are guaranteed that even in the state of incomplete information about the world, one can always determine whether a sequentially composed service is executable and whether this composite service will achieve a desired effect. The previously proposed approaches made different assumptions: [McIlraith and Son, 2002] assumes that the complete information is available about the world when effects of a composite service are computed, and [Giacomo *et al.*, 1999] considers the propositional fragment of the SC.

As we mentioned in Introduction, [McIlraith and Son, 2002] propose to use Golog for composition of Semantic Web services. It is surprisingly straightforward to define almost all Golog operators starting from our  $C^2$  based SC. The only restriction in comparison with the original Golog [Reiter, 2001] is that we cannot define the operator  $(\pi x)\delta(x)$ , non-deterministic choice of an action argument, because  $\mathcal{L}_{sc}^{DL}$

regressable formulas cannot have occurrences of non-ground action terms in situation terms.

The recent paper [Baader *et al.*, 2005] proposes integration of description logics *ALCO* (and its sub-languages) with an action formalism for reasoning about Web services. We discuss that paper in [Gu and Soutchanski, 2006].

An interesting paper [Liu and Levesque, 2005] aims to achieve computational tractability of solving projection and progression problems. The theory of the initial KB is assumed to be in the so-called *proper form* (i.e., conjunctions of equality-based formulas) and the query used in the projection problem is expected to be in a certain *normal form*. They consider a weaker type of progression defined for proper KBs with local effect actions only and show that their progression is sound (and sometimes complete) wrt the classical progression. We also consider local effect action theories, but our CNF-based KB is a set of disjunctions of equality-based formulas. However, [Liu and Levesque, 2005] considers a SC formulated in a general FOL (where the entailment problem is undecidable) and impose no restrictions on arity of fluents. Because of these significant differences in our approaches, it is not possible to compare them.

It is obvious that all cases from [Lin and Reiter, 1997; Shirazi and Amir, 2005; Liu and Levesque, 2005] when the progression is FOL definable also can be applied to our case simply because we restrict the language to two object variables only. However, we do the disjunctive case that nobody did before. Also, [Liu *et al.*, 2006] considers update of an ABox in a DL following [Winslett, 1990] and also mentions that update can be applied to a boolean ABox formulated in  $C^2$ , but their update is defined in terms of a conjunction of primitive fluent literals, i.e., it is different from our progression because our update is defined in terms of changes in the theory due to a ground action. [Giacomo *et al.*, 2006] generalized the update approach of [Liu *et al.*, 2006] from *ALC* to a more expressive DL language *DL-Lite* which allows general inclusion assertions in TBox, showed that the result of an update is always expressible by a *DL-Lite* ABox and provided a polynomial-time algorithm that computes the update over a *DL-Lite* KB.

The most important direction for future research is efficient implementation of practical scenarios of reasoning in  $\mathcal{L}_{sc}^{DL}$ . Although in general, the worst-case computational complexity for the reasoning problems in  $\mathcal{L}_{sc}^{DL}$  is high, some practical scenarios may facilitate empirically efficient solutions to the projection and executability problems. We are planning to consider also the progression problem for a more general class of incomplete KBs and conditions when the modified progressions is (classically) complete.

## Acknowledgments

Thanks to the Natural Sciences and Engineering Research Council of Canada and to the Department of Computer Science of the University of Toronto for partial financial support.

## References

[Baader *et al.*, 2003] Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook:*

*Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[Baader *et al.*, 2005] Franz Baader, Carsten Lutz, Maja Miličić, Ulrike Sattler, and Frank Wolter. Integrating description logics and action formalisms: First results. In *Proc. of AAAI-05*, pages 572–577, Pittsburgh, USA, July 2005. An extended version: LTCS-Report-05-02 from <http://lat.inf.tu-dresden.de/research/reports.html>.

[Borgida, 1996] Alexander Borgida. On the relative expressiveness of description logics and predicate logics. *Artificial Intelligence*, 82(1-2):353–367, 1996.

[Giacomo *et al.*, 1999] Giuseppe De Giacomo, Luca Iocchi, Daniele Nardi, and Riccardo Rosati. A theory and implementation of cognitive mobile robots. *J. of Logic and Computation*, 9(5):759–785, 1999.

[Giacomo *et al.*, 2006] Giuseppe De Giacomo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Posati. On the update of description logic ontologies at the instance level. In *Proc. of AAAI-06*, pages 1271–1276, Boston, MA, USA, July 2006. AAAI Press.

[Gu and Soutchanski, 2006] Yilan Gu and Mikhail Soutchanski. A logic for decidable reasoning about services. In *Proceedings of AAAI-06 workshop on AI-Driven Technologies for Services-Oriented Computing*, Boston, MA, USA, 2006. <http://www.cs.toronto.edu/~yilan/publications/papers/aaai06.pdf>.

[Lin and Reiter, 1997] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, 92:131–167, 1997.

[Liu and Levesque, 2005] Yongmei Liu and Hector J. Levesque. Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In *Proc. of IJCAI-05*, pages 522–527, Edinburgh, Scotland, UK, August 2005.

[Liu *et al.*, 2006] H. Liu, C. Lutz, M. Milicic, and F. Wolter. Updating description logic ABoxes. In *Proc. of KR2006*, pages 46–56, Lake District, UK, June, 2-5 2006.

[McIlraith and Son, 2002] Sheila McIlraith and Tran Son. Adapting Golog for composition of semantic web services. In *Proc. of KR2002*, pages 482–493, Toulouse, 2002.

[Pacholski *et al.*, 1997] Leszek Pacholski, Wiesław Szwań, and Lidia Tendera. Complexity of two-variable logic with counting. In *Proc. of LICS-97*, pages 318–327, Warsaw, Poland, 1997. A journal version: *SIAM Journal on Computing*, v 29(4), 1999, p. 1083–1117.

[Pratt-Hartmann, 2005] Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *Journal of Logic, Lang. and Inf.*, 14(3):369–395, 2005.

[Reiter, 2001] Raymond Reiter. *Knowledge in Action: Logical Foundations for Describing and Implementing Dynamical Systems*. The MIT Press, 2001.

[Shirazi and Amir, 2005] Afsaneh Shirazi and Eyal Amir. First-order logical filtering. In *Proc. of IJCAI-05*, pages 589–595, Edinburgh, Scotland, UK, 2005.

[Winslett, 1990] Marianne S. Winslett. *Updating logical databases*. The Academic Press, 1990.