

Dynamic Description Logic: Embracing Actions into Description Logic

Liang Chang^{1,2}, Zhongzhi Shi¹, Lirong Qiu^{1,2}, and Fen Lin^{1,2}

¹ Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China,

² Graduate University of Chinese Academy of Sciences, Beijing, China
{changl,shizz,qiulr,linf}@ics.ict.ac.cn

Abstract. We present a dynamic description logic $\mathcal{D}\text{-}\mathcal{ALCO}$ @ for representing knowledge about dynamic application domains. $\mathcal{D}\text{-}\mathcal{ALCO}$ @ is a combination of a typical action theory and the description logic \mathcal{ALCO} @, in such a way that actions are treated as citizens of the logic. Actions of $\mathcal{D}\text{-}\mathcal{ALCO}$ @ are explicitly specified with the help of formulas, and are then used in the construction of concepts and formulas. Based on a regression operator introduced to deal with actions, we provide a tableau-based decision algorithm for this logic.

1 Introduction

Description logics are successfully used for representing knowledge about static application domains in a structured way. In order to describe knowledge about dynamic application domains, temporal extension of description logics has been extensively studied for more than ten years [2]. With a general temporal description logic, the concept *Mortal* can be described as $Mortal = LivingBeing \sqcap (LivingBeing \mathcal{U} \sqcap \neg LivingBeing)$, which states that a mortal is a living being that eventually will not be alive any more. A temporal evolution is embodied in this description, but actions that fulfilling the evolution are not referred.

In [9], Wolter proposed a dynamic description logic named \mathcal{PDLC} , by combining the description logic \mathcal{ALC} with propositional dynamic logic PDL. With \mathcal{PDLC} , the concept *Mortal* can be described as $Mortal = LivingBeing \sqcap \langle die \rangle \neg LivingBeing$, in which actions that bring the change are presented. Although atomic actions of \mathcal{PDLC} can be combined using PDL-like operators, these atomic actions are still short of descriptions. Moreover, efficient decision algorithm for this logic is still an open problem

In this paper, we propose a dynamic description logic $\mathcal{D}\text{-}\mathcal{ALCO}$ @, by combining a typical action theory with the description logic \mathcal{ALCO} @. On the one hand, actions are generated from atomic actions with the help of many constructs, and each atomic action is specified by its preconditions P and conditional effects E , where P and E are described with formulas. On the other hand, actions could be used in the construction of concepts and formulas. Therefore, not only concepts with dynamic meaning, but also actions happened in dynamic domains, can all be described and reasoned with this formalism.

We first define the syntax and semantics of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ in section 2, then introduce a regression operator to deal with actions in section 3, and provide a tableau based decision algorithm for a restricted form of the logic in section 4. Section 5 concludes the paper.

2 Syntax and Semantics of Dynamic Description Logic

The primitive symbols of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ are a set N_C of concept names, a set N_R of role names, and a set N_I of individual names.

Concepts C, C' of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ are formed with the following syntax rules:

$$C, C' \longrightarrow D|\@_p C|C \sqcup C'|\exists R.C|< \pi > C \quad (1)$$

$$D, D' \longrightarrow C_i|\{p\}|\@_p D|\neg D|D \sqcup D'|\exists R.D$$

where $C_i \in N_C, p \in N_I, R \in N_R, \pi$ is an action. These syntax rules are designed to ensure that no concepts of the form $\neg < \pi > C$ are constructed.

Formulas φ, φ' of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ are formed with the following syntax rules:

$$\varphi, \varphi' \longrightarrow \phi|\varphi \vee \varphi'|< \pi > \varphi \quad (2)$$

$$\phi, \phi' \longrightarrow C(p)|R(p, q)|\neg \phi|\phi \vee \phi'$$

where C is a concept, $p, q \in N_I, R \in N_R$, and π is an action. These syntax rules are designed to ensure that no formulas of the form $\neg < \pi > \varphi$ are constructed.

An *atomic action* of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ is a pair (P, E) , where,

- P is a finite set of formulas, used for describing the so-called pre-conditions,
- E is a finite set of conditional effects of the form ψ/ϕ , where ψ is a formula, ϕ is of form $A(p), \neg A(p), R(p, q)$, or $\neg R(p, q)$, with $A \in N_C, R \in N_R$, and $p, q \in N_I$,
- let $P = \{\varphi_1, \dots, \varphi_n\}$ and $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$, then P and E subject to the constraint that $\varphi_1 \wedge \dots \wedge \varphi_n \rightarrow \neg \phi_k$ for all k with $1 \leq k \leq m$.

Actions π, π' of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ are formed with the following syntax rule:

$$\pi, \pi' \longrightarrow (P, E)|\varphi?|\pi \cup \pi'|\pi; \pi' \quad (3)$$

where (P, E) is an atomic action, φ is a formula.

Before introducing the semantics for this logic, we will give some intuitive examples. Firstly, we describe an atomic action named *load* as $(\{Gun(a), \neg loaded(a)\}, \{\top(a)/loaded(a)\})$, where \top is an abbreviation of $C \sqcup \neg C$ for any concept C . The description tells that the action could happen in the case that a is an unloaded gun, and the only change brought about by this action is that a is loaded. Similarly, an atomic action named *shoot* is described as $(\{Gun(a), LivingBeing(b)\}, \{loaded(a)/\neg LivingBeing(b)\})$, the conditional effect of it means that in the case a is loaded, b will be not alive after the execution of *shoot*.

These two atomic actions can then be used to form a formula $\langle load ; shoot \rangle$ ($\neg LivingBeing(b)$), which asserts that b might be not alive after the sequential execution of *load* and *shoot*. Furthermore, we can also construct a concept $\langle shoot \rangle LivingBeing$ to describe these individuals that might be alive after the execution of *shoot*.

A model of $\mathcal{D}\text{-ALCCO}@$ is a pair $M=(W, I)$, where W is a set of states, I associates with each state $w \in W$ an interpretation $I(w) = (\Delta^I, C_0^{I(w)}, \dots, R_0^{I(w)}, \dots, p_0^{I(w)}, \dots)$, with $C_i^{I(w)} \subseteq \Delta^I$ for each $C_i \in N_c$, $R_i^{I(w)} \subseteq \Delta^I \times \Delta^I$ for each $R_i \in N_R$, and $p_i^{I(w)} \in \Delta^I$ for each $p_i \in N_I$; furthermore, for any $p_i \in N_I$ and any $u, v \in W$, we have $p_i^{I(u)} = p_i^{I(v)}$. Based on the interpretations of all these states, each action π is interpreted as a binary relation $\pi^I \subseteq W \times W$.

Given a $\mathcal{D}\text{-ALCCO}@$ model $M=(W, I)$ and a state $w \in W$, the value $C^{I(w)}$ of a concept C , the truth-relation $(M, w) \models \varphi$ (or simply $w \models \varphi$ if M is understood) for a formula φ , and the relation π^I for an action π are defined inductively as follows:

- (1) $\{p\}^{I(w)} = \{p^{I(w)}\}$;
- (2) If $p^{I(w)} \in C^{I(w)}$ then $(@_p C)^{I(w)} = \Delta^I$, else $(@_p C)^{I(w)} = \{\}$;
- (3) $(\neg C)^{I(w)} = \Delta^I - C^{I(w)}$;
- (4) $(C \sqcup D)^{I(w)} = C^{I(w)} \cup D^{I(w)}$;
- (5) $(\exists R.C)^{I(w)} = \{x | \exists y. ((x, y) \in R^{I(w)} \wedge y \in C^{I(w)})\}$;
- (6) $\langle \pi > C \rangle^{I(w)} = \{p | \exists w' \in W. ((w, w') \in \pi^I \wedge p \in C^{I(w')})\}$;
- (7) $(M, w) \models C(p)$ iff $p^{I(w)} \in C^{I(w)}$;
- (8) $(M, w) \models R(p, q)$ iff $(p^{I(w)}, q^{I(w)}) \in R^{I(w)}$;
- (9) $(M, w) \models \neg \varphi$ iff $(M, w) \models \varphi$ not holds;
- (10) $(M, w) \models \varphi \vee \psi$ iff $(M, w) \models \varphi$ or $(M, w) \models \psi$;
- (11) $(M, w) \models \langle \pi > \varphi$ iff $\exists w' \in W. ((w, w') \in \pi^I \wedge (M, w') \models \varphi)$;
- (12) Let S be a formula set, then, $(M, w) \models S$ iff $(M, w) \models \varphi_i$ for all $\varphi_i \in S$;
- (13) Let (P, E) be an atomic action with $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$, then, $(P, E)^I = \{(w_1, w_2) \in W \times W \mid (M, w_1) \models P, C^{I(w_2)} = C^{I(w_1)} \cup C^+ - C^- \text{ for each concept name } C \in N_C, \text{ and } R^{I(w_2)} = R^{I(w_1)} \cup R^+ - R^- \text{ for each role name } R \in N_R\}$, where,
 - $C^+ = \{p^{I(w_1)} \mid \psi/\phi \in E_\alpha, \phi = C(p), \text{ and } (M, w_1) \models \psi\}$,
 - $C^- = \{p^{I(w_1)} \mid \psi/\phi \in E_\alpha, \phi = \neg C(p), \text{ and } (M, w_1) \models \psi\}$,
 - $R^+ = \{(p^{I(w_1)}, q^{I(w_1)}) \mid \psi/\phi \in E_\alpha, \phi = R(p, q), \text{ and } (M, w_1) \models \psi\}$,
 - $R^- = \{(p^{I(w_1)}, q^{I(w_1)}) \mid \psi/\phi \in E_\alpha, \phi = \neg R(p, q), \text{ and } (M, w_1) \models \psi\}$;
- (14) $(\varphi?)^I = \{(w_1, w_1) \in W \times W \mid (M, w_1) \models \varphi\}$;
- (15) $(\pi \cup \pi')^I = \pi^I \cup \pi'^I$;
- (16) $(\pi; \pi')^I = \pi^I \circ \pi'^I$.

The interpretation of atomic actions follows the possible models approach[8], and adopts the style introduced in[4].

A formula φ (or a formula set S) is satisfiable if and only if there is a model $M = (W, I)$ and a state $w \in W$ with $(M, w) \models \varphi$ (or $(M, w) \models S$).

The goal of the following sections is to develop an algorithm for checking the satisfiability of $\mathcal{D}\text{-ALCCO}@$ formulas. For simplicity, we take the unique name

assumption (UNA), i.e., $p_i^{I(w)} \neq p_j^{I(w)}$ for any $p_i, p_j \in N_I$ with $p_i \neq p_j$. Furthermore, we don't take into account TBoxes that composed of concept definitions[3].

3 Regression Operator

In this section we introduce a *regression operator* to deal with actions.

Definition 1 (Regression). For a formula of the form $\langle \pi \rangle \varphi$, ψ is the result of regressing $\langle \pi \rangle \varphi$, in symbols $\text{Regress}(\langle \pi \rangle \varphi) = \psi$, if

- (1) no actions occurred in ψ , i.e., ψ is an $\mathcal{ALCCO@}$ formula;
- (2) $\langle \pi \rangle \varphi \models \psi$, i.e., for any model $M = (W, I)$ and any state $w \in W$: if $(M, w) \models \langle \pi \rangle \varphi$, then $(M, w) \models \psi$;
- (3) for any model $M = (W, I)$ and any state $w \in W$: if $(M, w) \models \psi$, then we can construct a model $M' = (W', I')$ by introducing a world w' , with the constraint that $W' = W \cup \{w'\}$, $I'(w_i) = I(w_i)$ for each $w_i \in W$, $(w, w') \in \pi^I$, and $(M', w') \models \varphi$; therefore we have $(M', w) \models \langle \pi \rangle \varphi$.

Before presenting algorithms for the regression operator, we introduce another operator named *ABox updating triggered by atomic action*.

An ABox is a finite set of individual assertions of the form $C(p)$, $R(p, q)$, and $\neg R(p, q)$, where C is a concept, $R \in N_R$, and $p, q \in N_I$. An ABox \mathcal{A} entails a formula φ (written $\mathcal{A} \models \varphi$) if and only if for any model $M = (W, I)$ and any state $w \in W$: $(M, w) \models \mathcal{A}$ implies $(M, w) \models \varphi$. An ABox \mathcal{A} entails a formula set S (written $\mathcal{A} \models S$) if and only if $\mathcal{A} \models \varphi$ for any formula $\varphi \in S$.

Definition 2 (ABox updating triggered by atomic action). Let $\mathcal{A}, \mathcal{A}'$ be ABoxes, (P, E) be an atomic action. Then, \mathcal{A}' is the result of updating \mathcal{A} with (P, E) , in symbols $\mathcal{A} * (P, E) = \mathcal{A}'$, if

- (1) $\mathcal{A} \models P$;
- (2) For any model $M = (W, I)$ and any states $w, w' \in W$: if $(w, w') \in (P, E)^I$ and $(M, w) \models \mathcal{A}$, then $(M, w') \models \mathcal{A}'$;
- (3) For any model $M = (W, I)$ and any state w in W : if $(M, w) \models \mathcal{A}'$, then we can construct a model $M' = (W', I')$ by introducing a world w' , with the constraint that $W' = W \cup \{w'\}$, $I'(w_i) = I(w_i)$ for each $w_i \in W$, $(w', w) \in (P, E)^{I'}$ and $(M', w') \models \mathcal{A}$.

This operator is similar to the ABox updating introduced by Liu et al[6]. Therefore, based on the ABox update algorithm by Liu et al[6], we develop the following algorithm to calculate $\mathcal{A} * (P, E)$, by adding step (1) to decide whether the action is executable on \mathcal{A} , adding step (2) to select these changes that will take place, and adding the seventh case in step (3) to calculate $(\langle \pi \rangle C)^{Ea}$ for concepts of the form $\langle \pi \rangle C$.

Algorithm 1 ($\mathcal{A} * (P, E)$) Let \mathcal{A} be an ABox, (P, E) be an atomic action with $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$. Construct an ABox \mathcal{A}' with the following steps:

- (1) If $\mathcal{A} \models P$ not holds, exit the algorithm with the result “ (P, E) is not executable on \mathcal{A} ”;

(2) Construct a set $\text{Eff}(\mathcal{A}, E) := \{\phi \mid \psi/\phi \in E \text{ and } \mathcal{A} \models \psi\}$; If $\text{Eff}(\mathcal{A}, E)$ is empty, return the ABox $\mathcal{A}' := \mathcal{A}$, exit the algorithm;

(3) Let $\text{ObjE}(\mathcal{A}, E)$ be all the individual names occurred in $\text{Eff}(\mathcal{A}, E)$; Construct an ABox $\mathcal{A}^E := \{C^E(p) \mid C(p) \in \mathcal{A}\} \cup \{R(p, q) \mid R(p, q) \in \mathcal{A} \text{ and } \neg R(p, q) \notin \text{Eff}(\mathcal{A}, E)\} \cup \{\neg R(p, q) \mid \neg R(p, q) \in \mathcal{A} \text{ and } R(p, q) \notin \text{Eff}(\mathcal{A}, E)\}$, where C^E is constructed inductively as follows:

- For concept name C_i , $C_i^E := C_i \sqcup \bigsqcup_{\neg C_i(p) \in \text{Eff}(\mathcal{A}, E)} \{p\} \sqcap \prod_{C_i(p) \in \text{Eff}(\mathcal{A}, E)} \neg\{p\}$;
- $\{p\}^E := \{p\}$;
- $(@_p D)^E := @_p D^E$;
- $(\neg D)^E := \neg D^E$;
- $(D_1 \sqcup D_2)^E := D_1^E \sqcup D_2^E$;
- $(\exists R.D)^E := (\prod_{p \in \text{ObjE}(\mathcal{A}, E)} \neg\{p\} \sqcap \exists R.D^E) \sqcup (\prod_{p \in \text{ObjE}(\mathcal{A}, E)} \{p\} \sqcap \exists R.(\prod_{q \in \text{ObjE}(\mathcal{A}, E)} \neg\{q\} \sqcap D^E)) \sqcup \prod_{p, q \in \text{ObjE}(\mathcal{A}, E), R(p, q) \notin \text{Eff}(\mathcal{A}, E), \neg R(p, q) \notin \text{Eff}(\mathcal{A}, E)} (\{p\} \sqcap \exists R.(\{q\} \sqcap D^E)) \sqcup \prod_{\neg R(p, q) \in \text{Eff}(\mathcal{A}, E)} (\{p\} \sqcap @_q D^E)$;
- $(\langle \pi > D)^E := \langle \{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}; \pi > C$, where $(\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})$ is an atomic action constructed according to the elements of E ;

(4) Return the ABox $\mathcal{A}' := \mathcal{A}^E \cup \text{Eff}(\mathcal{A}, E)$.

In this algorithm, rules used for constructing C^E are technically designed to guarantee the following property:

Property 1. Let $M = (W, I)$ be a $\mathcal{D}\text{-}\mathcal{ALCCO}@$ model, $w, w' \in W$, and $(w, w') \in (P, E)^I$. Then, for any $\mathcal{D}\text{-}\mathcal{ALCCO}@$ concept C and any individual name x , $w' \models C^E(x)$ if and only if $w \models C(x)$.

Since $x^{I(w')} = x^{I(w)}$, we only need to demonstrate $(C^E)^{I(w')} = C^{I(w)}$, by structural induction on C . In these cases that C is C_i , $\{p\}$, $@_p D$, $\neg D$, $D_1 \sqcup D_2$, and $\exists R.D$, the proof is similar to those given in[5]. In the case that C is $\langle \pi > D$, we have $((\langle \pi > D)^E)^{I(w')} = (\langle \{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}; \pi > D)^{I(w')} = \{p \mid \exists w_1 \in W. ((w', w_1) \in ((\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\}); \pi)^I \wedge p \in D^{I(w_1)})\} = \{p \mid \exists w_1 \in W. \exists w_2 \in W. ((w', w_2) \in (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})^I \wedge (w_2, w_1) \in \pi^I \wedge p \in D^{I(w_1)})\}$. According to the semantics of $\mathcal{D}\text{-}\mathcal{ALCCO}@$ actions, $I(w_2)$ and $I(w)$ are equivalent and $(w', w) \in (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})^I$, therefore we can continue these equations as $((\langle \pi > D)^E)^{I(w')} = \{p \mid \exists w_1 \in W. ((w, w_1) \in \pi^I \wedge p \in D^{I(w_1)})\} = (\langle \pi > D)^{I(w)}$.

The following property is an easy consequence:

Property 2. Algorithm 1 is terminable, the returned \mathcal{A}' satisfies $\mathcal{A} * \alpha = \mathcal{A}'$.

Utilizing the algorithm of $\mathcal{A} * (P, E)$, we develop the following algorithm for the regression operator:

Algorithm 2 (Regress($\langle \pi \rangle \varphi$)) Let π be an action, φ be a formula. Calculate $\text{Regress}(\langle \pi \rangle \varphi)$ recursively with the following steps:

- (1) If φ contains a subformula of the form $\langle \pi_i \rangle \varphi_i$, then replace all the occurrence of $\langle \pi_i \rangle \varphi_i$ in φ with $\text{Regress}(\langle \pi_i \rangle \varphi_i)$; Repeat this step, until no such subformulas contained in φ ;
- (2) If π is an atomic action (P, E) with $E = \{\psi_1/\phi_1, \dots, \psi_m/\phi_m\}$, then,
 - (i) Construct an atomic action $\alpha' = (\{\}, \{\phi_1/\neg\phi_1, \dots, \phi_m/\neg\phi_m\})$;
 - (ii) Translate φ into a disjunction normal form $\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_k$, where each φ_i is a conjunction of individual assertions;
 - (iii) For each φ_i , let it be $\varphi_{i1} \wedge \varphi_{i2} \wedge \dots \wedge \varphi_{im}$, construct an ABox $\mathcal{A}_i := \{\varphi_{i1}, \varphi_{i2}, \dots, \varphi_{im}\}$, and construct the ABox \mathcal{A}'_i that satisfying $\mathcal{A}_i * \alpha' = \mathcal{A}'_i$;
 - (iv) Return the formula $(\text{Set2F}(\mathcal{A}'_1) \vee \dots \vee \text{Set2F}(\mathcal{A}'_k)) \wedge \text{Set2F}(P)$, where $\text{Set2F}(S)$ represents the conjunction of all the elements of S , e.g., if $S = \{\varphi_1, \dots, \varphi_n\}$, then $\text{Set2F}(S) := \varphi_1 \wedge \dots \wedge \varphi_n$.
- (3) If π is $\phi?$, then return the formula $\phi \wedge \varphi$;
- (4) If π is $\pi_1 \cup \pi_2$, then return $\text{Regress}(\langle \pi_1 \rangle \varphi) \vee \text{Regress}(\langle \pi_2 \rangle \varphi)$;
- (5) If π is $\pi_1 ; \pi_2$, then return $\text{Regress}(\langle \pi_1 \rangle \text{Regress}(\langle \pi_2 \rangle \varphi))$.

Property 3. Algorithm 2 is terminable, the returned formula ψ satisfies $\text{Regress}(\langle \pi \rangle \varphi) = \psi$.

This property can be proved with three steps. Firstly, in the case that π is an atomic action and φ is a formula containing no subformulas of the form $\langle \pi_i \rangle \varphi_i$, it is obvious that the algorithm will terminate, it is also easy to demonstrate that the returned formula ψ satisfies $\text{Regress}(\langle \pi \rangle \varphi) = \psi$. Secondly, we relax the π to be any actions, and demonstrate the same results by structural induction on π . Finally, we relax the φ to be any formulas and demonstrate the property. Due to space limitation, we omit the details here.

4 Tableau Algorithm

Based on the regression operator, we can develop a tableau-based procedure for deciding the satisfiability of $\mathcal{D}\text{-ALCO@}$ formulas.

Algorithm 3 (Deciding the satisfiability of a $\mathcal{D}\text{-ALCO@}$ formula) For a $\mathcal{D}\text{-ALCO@}$ formula φ , decide its satisfiability with the following steps:

- (1) Construct a formula set $S' := \{\varphi\}$. If S' contains clash, exit the algorithm with the result “ φ is unsatisfiable”.
- (2) Construct a set $SS := \{S'\}$;
- (3) Take out an element S from SS , apply one of the rules in table 1 to S ; For every new generated formula set, if it contains no clash, then add it into SS ;
- (4) Repeat step (3), until SS is empty or no rules can be applied to the formula set S that taken out from SS , in the former case return the result “ φ is unsatisfiable”, in the latter case return the result “ φ is satisfiable”.

A *clash* in a formula set S is one of the following cases: (1) $\varphi \in S$ and $\neg\varphi \in S$ for a formula φ ; (2) $C(p) \in S$ and $(\neg C)(p) \in S$ for a concept C and an

- individual name p ; (3) $\{q\}(p) \in S$ for two different individual names p and q ;
(4) $(\neg\{p\})(p) \in S$ for an individual name p .

Table 1. Tableau rules for $\mathcal{D}\text{-}\mathcal{ALCCO}$

Rules on concepts:

- $\mathbf{R}_{@}$: If $(@_q C)(x) \in S$ and $C(q) \notin S$, then $S_1 := \{C(q)\} \cup S$;
 $\mathbf{R}_{\neg@}$: If $(\neg@_q C)(x) \in S$ and $(\neg C)(q) \notin S$, then $S_1 := \{(\neg C)(q)\} \cup S$;
 \mathbf{R}_{\sqcup} : If $(C_1 \sqcup C_2)(x) \in S$, $C_1(x) \notin S$ and $C_2(x) \notin S$,
then $S_1 := C_1(x) \cup S$, $S_2 := C_2(x) \cup S$;
 $\mathbf{R}_{\neg\sqcup}$: If $(\neg(C_1 \sqcup C_2))(x) \in S$, and $(\neg C_1)(x) \notin S$ or $(\neg C_2)(x) \notin S$,
then $S_1 := \{(\neg C_1)(x), (\neg C_2)(x)\} \cup S$;
 \mathbf{R}_{\exists} : If $(\exists R.C)(x) \in S$, there is no y such that $R(x, y) \in S$ and $C(y) \in S$,
then $S_1 := \{C(z), R(x, z)\} \cup S$, where z is a new individual name;
 $\mathbf{R}_{\neg\exists}$: If $(\neg(\exists R.C))(x) \in S$, then $S_1 := \{(\neg C)(y) \mid R(x, y) \in S, (\neg C)(y) \notin S\}$;
 $\mathbf{R}_{<\pi>c}$: If $(\langle \pi \rangle C)(x) \in S$, and $\text{Regress}(\langle \pi \rangle C(x)) \notin S$,
then $S_1 := \{\text{Regress}(\langle \pi \rangle C(x))\} \cup S$;
 $\mathbf{R}_{\neg c}$: If $(\neg(\neg C))(x) \in S$, and $C(x) \notin S$, then $S_1 := \{C(x)\} \cup S$;

Rules on formulas:

- \mathbf{R}_{\vee} : If $\varphi \vee \psi \in S$, $\varphi \notin S$, and $\psi \notin S$, then $S_1 := \{\varphi\} \cup S$, $S_2 := \{\psi\} \cup S$;
 $\mathbf{R}_{\neg\vee}$: If $\neg(\varphi \vee \psi) \in S$, and $\neg\varphi \notin S$ or $\neg\psi \notin S$, then $S_1 := \{\neg\varphi, \neg\psi\} \cup S$;
 $\mathbf{R}_{<\pi>f}$: If $\langle \pi \rangle \varphi \in S$ and $\text{Regress}(\langle \pi \rangle \varphi) \notin S$, then $S_1 := \{\text{Regress}(\langle \pi \rangle \varphi)\} \cup S$;
 $\mathbf{R}_{\neg f}$: If $\neg(\neg\varphi) \in S$ and $\varphi \notin S$, then $S_1 := \{\varphi\} \cup S$.
-

It is easy to demonstrate that this algorithm holds the following properties:

- (*Termination*) The algorithm is terminable;
- (*Soundness*) For the set SS , let SS' be the result of executing step (3), then, there is a satisfiable formula set $S \in SS$ if and only if there is a satisfiable formula set $S' \in SS'$.
- (*Completeness*) For any finite formula set $S \in SS$, if no rules can be applied to S and S contains no clash, then S is satisfiable.

5 Discussion

TBoxes composed of concept definitions are not taken into account in previous sections. In fact, if a TBox is referred, the set N_C could be divided into two disjoint sets N_{CD} and N_{CP} , where N_{CD} is the set of defined concept names, N_{CP} is the set of primitive concept names [3]. In this case, adopting the idea of [4], we will add a constraint to the description of atomic actions: for every conditional effect ψ/ϕ , ϕ must be of form $A(p)$, $\neg A(p)$, $R(p, q)$, or $\neg R(p, q)$, with $A \in N_{CP}$. Then, if the TBox is acyclic and contains no general concept inclusion

axioms(GCIs), our algorithms are still effective, by adding processes to replace each occurrence of defined concept names with their corresponding definitions.

As actions are treated as citizens in $\mathcal{D}\text{-}\mathcal{ALCO}$, reasoning problems about actions could be introduced into this formalism, such as the executability, projection, and subsumption problems[1][4][7]. We think that the regression operator is useful for these reasoning tasks. For example, in order to decide that whether a formula φ is a consequence of applying a sequence of actions π_1, \dots, π_k in an ABox \mathcal{A} , we can calculate the formula $\psi := \neg \text{Regress}(\langle \pi_1; \dots; \pi_k \rangle \neg \varphi)$ and check $\mathcal{A} \models \psi$.

Another future work is to alleviate these syntactic restrictions posed on this logic, so that concepts of the form $\neg \langle \pi \rangle C$, formulas of the form $\neg \langle \pi \rangle \varphi$, and actions of the form π^* can be constructed.

Acknowledgments. This work was partially supported by the National Science Foundation of China (No. 90604017, 60435010), and National Basic Research Priorities Programme(No. 2003CB317004). We would like to thank these anonymous reviewers for their valuable suggestions.

References

1. Artale, A., Franconi, E.: A Temporal Description Logic for Reasoning about Actions and Plans. *Journal of Artificial Intelligence Research*, 1998, 9: 463-506.
2. Artale, A. Franconi, E.: A Survey of Temporal Extensions of Description Logics. *Annals of Mathematics and Artificial Intelligence*, 2000, 30(1-4):171-210.
3. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. F., eds.: *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.
4. Baader, F., Lutz, C., Milicic, M., Sattler, U., Wolter, F.: Integrating Description Logics and Action Formalisms: First Results. In *Proceedings of the 12th National Conference on Artificial Intelligence (AAAI'05)*, Pittsburgh, PA, USA, 2005.
5. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating Description Logic ABoxes. *LTCS-Report 05-10*, Chair for Automata Theory, Institute for Theoretical Computer Science, Dresden University of Technology, Germany, 2005.
6. Liu, H., Lutz, C., Milicic, M., Wolter, F.: Updating description logic ABoxes. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR'2006)*, 2006.
7. Lutz, C., Sattler, U.: A Proposal for Describing Services with DLs. In *Proceedings of the 2002 International Workshop on Description Logics (DL'2002)*, 2002.
8. Winslett, M.: Reasoning about action using a possible models approach. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'88)*, 1988.
9. Wolter, F., Zakharyashev, M.: Dynamic description logic. In *Advances in Modal Logic*, Vol 2. Stanford: CSLI Publications, 2000: 449-463