Dynamic Logic for the Semantic Web

Liang Chang, Qicheng Zhang, Tianlong Gu, Zhongzhi Shi

Dynamic Logic for the Semantic Web

Liang Chang[1], Qicheng Zhang[1], Tianlong Gu[1], Zhongzhi Shi[2]

[1] Guangxi Key Laboratory of Trusted Software,
Guilin University of Electronic Technology, Guilin 541004, China
[2] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology,
Chinese Academy of Sciences, Beijing 100190, China

changl@guet.edu.cn, cctlgu@guet.edu.cn, shizz@ics.ict.ac.cn

Abstract. The propositional dynamic logic PDL is one of the most successful variants of modal logic; it plays an important role in many fields of computer science and artificial intelligence. As a logical basis for the W3C-recommended Web ontology language OWL, description logic provides considerable expressive power going far beyond propositional logic as while as the reasoning is still decidable. In this paper, we bring the power and character of description logic into PDL and present a dynamic logic *ALC*-DL for the semantic Web. The logic *ALC*-DL inherits the knowledge representation ability of both the description logic ALC and the logic PDL. With an approach based on Buchi tree automaton, we prove that the satisfiability problem of *ALC*-DL formulas is still decidable and is EXPTIME-complete. The logic *ALC*-DL is suitable for modeling and reasoning about dynamic knowledge in the semantic Web environment.

Keywords. dynamic logic; description logic; satisfiability problem; Buchi tree automaton; semantic Web

1 Introduction

The propositional dynamic logic PDL is one of the most successful variants of modal logic[1]. It is widely used in many fields of computer science and artificial intelligence. In order to enhance the knowledge representation capability of PDL and correspondingly extend the application domain, many extended logics were proposed by introducing action constructors into PDL[2-4].

Description logic is a family of languages for describing and reasoning about the knowledge of static domains. It plays an important role in the semantic Web, acting as the logic basis of the W3C-recommended Web ontology language OWL. A feature of description logic is that it provides considerable expressive power going far beyond propositional logic, while reasoning is still decidable.

In this paper, we bring the power and character of description logic into dynamic logic. This work is motivated by the fact that, in order to model Web services, actions, or intelligent agents in the semantic Web, one has to deal with lots of knowledge represented as ontologies. Since the Web ontology language OWL is based on description logic, the combination of description logic and dynamic logic will provide the capability

to deal with both the static knowledge of ontology and the dynamic knowledge on actions and services.

For the simplicity of presentation, this paper take the typical logic ALC as an example of description logics and adopt it to extend the propositional dynamic logic PDL. As a result, an extended dynamic logic named *ALC*-DL is presented, which inherits the knowledge representation capability of both the description logic ALC and the propositional dynamic logic PDL. With the help of the Buchi tree automaton, the satisfiability problem of *ALC*-DL formulas is investigated and demonstrated to be EXPTIME-complete.

2 Dynamic Logic *ALC*-DL

The dynamic logic *ALC*-DL is constructed by embracing the description logic ALC into the propositional dynamic logic PDL. From the point of view of syntax, *ALC*-DL is constructed by replacing all the atomic propositions of PDL with ABox assertions and TBox axioms of ALC.

Primitive symbols of *ALC*-DL are a set N_C of concept names, a set N_R of role names, and a set N_I of individual names. Starting from these symbols, with the help of a set of constructors, the concepts, formulas and actions of *ALC*-DL can be constructed inductively.

Definition 1. Concepts of *ALC*-DL are formed according to the following syntax rule:

$$C, D ::= C_i \mid \neg C \mid C \sqcup D \mid \forall R.\, C$$

where $C_i \in N_C$, $R \in N_R$. Furthermore, concepts of the forms $C \sqcap D$ and $\exists R.\, C$ can also be introduced as abbreviations of concepts $\neg(\neg C \sqcup \neg D)$ and $\neg(\forall R.\, \neg C)$ respectively.

A *general concept inclusion axiom* is an expression of the form $C \sqsubseteq D$, where C, D are any concepts. A *concept assertion* (resp., *role assertion*) is of the form $C(p)$ (resp., $R(p,q)$), where C is a concept, $R \in N_R$, and $p,q \in N_I$.

Definition 2. Formulae of *ALC*-DL are formed according to the following syntax rule:

$$\varphi, \psi ::= C \sqsubseteq D \mid C(p) \mid R(p,q) \mid <\pi>\varphi \mid \neg\varphi \mid \varphi \wedge \psi$$

where $p, q \in N_I$, $R \in N_R$, C, D are concepts, and π is an action.

Formulas of the forms $[\pi]\varphi$, $\varphi \vee \psi$, $\varphi \rightarrow \psi$, *false* and *true* can also be introduced as abbreviations of formulas $\neg<\pi>\neg\varphi$, $\neg(\neg\varphi \wedge \neg\psi)$, $\neg(\varphi \wedge \neg\psi)$, $\varphi \wedge \neg\varphi$ and $\neg(\varphi \wedge \neg\varphi)$ respectively. Formulas of the form $<\pi>\varphi$ are called *action existence assertions*.

Definition 3. Actions of *ALC*-DL are formed according to the following syntax rule:

$$\pi, \pi' ::= \alpha \mid \varphi? \mid \pi \cup \pi' \mid \pi ; \pi' \mid \pi^*$$

where $\alpha \in N_A$, and φ is a formula. Actions of the forms α, $\varphi?$, $\pi \cup \pi'$, $\pi;\pi'$ and π^* are respectively called *atomic actions*, *test actions*, *sequential actions* and *iterated actions*.

The interpretation structure of *ALC*-DL is a combination of the interpretation structure of PDL and the interpretation of ALC. In such a structure, each action is

interpreted as a binary relation between states, while each state is mapped to a classical interpretation of description logic.

Definition 4. A *ALC*-DL interpretation structure is of the form $M=(W,T,\Delta,I)$, where,

(1) W is a non-empty finite set of states;

(2) T is a function which maps each action name $\alpha_i \in N_A$ to a binary relation $T(\alpha_i) \subseteq W \times W$;

(3) Δ is a non-empty set of individuals; and

(4) I is a function which associates with each state $w \in W$ a description logic interpretation $I(w)=(\Delta, \bullet^{I(w)})$, where the function $\bullet^{I(w)}$

(i) maps each concept name $C_i \in N_C$ to a set $C_i^{I(w)} \subseteq \Delta$,

(ii) maps each role name $R_i \in N_R$ to a binary relation $R_i^{I(w)} \subseteq \Delta \times \Delta$, and

(iii) maps each individual name $p_i \in N_I$ to an individual $p_i^{I(w)} \in \Delta$, with the constraints that $p_i^{I(w)}=p_i^{I(w')}$ for any state $w' \in W$. The interpretation $p_i^{I(w)}$ is also represented as p_i^I, since it is not effected by the state w.

Definition 5. Given an interpretation structure $M=(W,T,\Delta,I)$, the semantics of concepts, formulas and actions of *ALC*-DL are defined inductively as follows.

Firstly, for each state $w \in W$, a concept C is interpreted as a set $C^{I(w)} \subseteq \Delta$ which is defined inductively as follows:

(1) $(\neg C)^{I(w)} := \Delta_M \setminus C^{I(w)}$;

(2) $(C \sqcup D)^{I(w)} := C^{I(w)} \cup D^{I(w)}$;

(3) $(\forall R.C)^{I(w)} := \{ x \mid \text{for every } y \in \Delta\text{: if } (x, y) \in R^{I(w)}, \text{ then } y \in C^{I(w)} \}$.

Secondly, for each state $w \in W$, the satisfaction-relation $(M,w) \models \varphi$ for any formula φ is defined as follows:

(4) $(M,w) \models C(p)$ iff $p^I \in C^{I(w)}$;

(5) $(M,w) \models R(p,q)$ iff $(p^I,q^I) \in R^{I(w)}$;

(6) $(M,w) \models C \sqsubseteq D$ iff $C^{I(w)} \subseteq D^{I(w)}$;

(7) $(M,w) \models \langle\pi\rangle\varphi$ iff some state $w' \in W$ exists with $(w,w') \in T(\pi)$ and $(M,w') \models \varphi$;

(8) $(M,w) \models \neg\varphi$ iff it is not the case that $(M,w) \models \varphi$;

(9) $(M,w) \models \varphi \wedge \psi$ iff $(M,w) \models \varphi$ and $(M,w) \models \psi$.

Finally, each action π is interpreted as a binary relation $T(\pi) \subseteq W \times W$ according to the following definitions:

(10) $T(\varphi?) := \{(w,w) \in W \times W \mid (M,w) \models \varphi\}$;

(11) $T(\pi \cup \pi') := T(\pi) \cup T(\pi')$;

(12) $T(\pi ; \pi') := \{(w,w') \in W \times W \mid$ there is some state $u \in W$ with $(w,u) \in T(\pi)$ and $(u,w') \in T(\pi')\}$;

(13) $T(\pi^*)$:= reflexive and transitive closure of $T(\pi)$.

Definition 6. An *ALC*-DL formula φ is *satisfiable* if and only if there is an interpretation structure $M=(W,T,\Delta,I)$ and a state $w \in W$ such that $(M,w) \models \varphi$.

The satisfiability problem of formulae is a primary inference problem for *ALC*-DL. Many other inference problems which we might concern can be reduced to this problem. For example, given a knowledge base $\mathcal{K}=(\mathcal{T}, \mathcal{A})$ of the description logic *ALC*, we want to know whether it is consistent or not. Since the TBox $\mathcal{T}$ is a finite set composed of general concept inclusion axioms, and the ABox $\mathcal{A}$ is a finite set composed of concept assertions, negations of concept assertions, role assertions, and negations of role assertions, it is obvious that both $\mathcal{T}$ and $\mathcal{A}$ are formula sets of the logic *ALC*-DL, and therefore the consistence problem of the knowledge base $\mathcal{K}$ can be decided by checking whether the conjunction of all the formulas contained in $\mathcal{T}$ and $\mathcal{A}$ is satisfiable or not.

If an *ALC*-DL formula is just a Boolean connection of general concept inclusion axioms, concept assertions and role assertions, then it is also called a *Boolean knowledge base* of the description logic *ALC*[5]. A Boolean knowledge base $\mathcal{B}$ is consistent if and only if there is an interpretation structure $M=(W,T,\Delta,I)$ and a state $w\in W$ such that $(M,w)\models\varphi_i$ for every $\varphi_i\in\mathcal{B}$. It is EXPTIME-complete to decide whether a Boolean knowledge base of the description logic *ALC* is consistent or not[5].

3 An Automata-based variant of *ALC*-DL

In order to adopt Buchi tree automaton for studying the satisfiability problem of *ALC*-DL formulae, be similar with the approach used in [6], we use automata on finite words rather than regular expressions to describe complex actions of *ALC*-DL.

Firstly, for any *ALC*-DL action π, taking it as a regular expression, then the language $L(\pi)$ represented by it is defined as following: ①$L(\pi):=\{\pi\}$ if π is an atomic action or a test action, ②$L(\pi\cup\pi'):=L(\pi)\cup L(\pi')$, ③$L(\pi\ ;\ \pi'):=\{l_1l_2 \mid l_1\in L(\pi)$ and $l_2\in L(\pi')\}$, and ④$L(\pi^*):=L(\pi^0)\cup L(\pi^1)\cup L(\pi^2)\cup\ldots$, where $L(\pi^0)$ is the empty word ε, and $L(\pi^i):= L(\pi^{i-1};\pi)$ for every $i\geq 1$. Obviously, all the possible executions of π are captured by the words in $L(\pi)$.

Secondly, we introduce some notations on nondeterministic finite automata.

Definition 7. A nondeterministic finite automaton (NFA) $A=(Q, \Sigma, \rho, q_0, F)$ consists of a finite set of states Q, a finite input alphabet Σ, a transition function $\rho:Q\times\Sigma\rightarrow 2^Q$, an initial state $q_0\in Q$, and a set of final states $F\subseteq Q$.

The transition function ρ can also be inductively extended to a function ρ': $Q\times\Sigma^*\rightarrow 2^Q$, such that $\rho'(q,\varepsilon)=q$ and $\rho'(q,\omega a)=\rho(\rho'(q,\omega),a)$ for any $q\in Q$, $\omega\in\Sigma^*$ and $a\in\Sigma$, where ε is the empty word.

Given a NFA $A=(Q, \Sigma, \rho, q_0, F)$, a word $\omega\in\Sigma^*$ is accepted by it if and only if $\rho'(q_0,\omega)\in F$. The language accepted by A is the set of all the words accepted by A, and is denoted as $L(A)$.

For the simplicity of presentation, for any NFA $A=(Q, \Sigma, \rho, q_0, F)$, we use Q_A, Σ_A, ρ_A, q_A and F_A to denote the set of states, the alphabet, the transition function, the initial state, and the set of final states of this automaton respectively. Furthermore, for

any NFA A and any state $q \in Q_A$, we use A_q to denote the automaton which is constructed by setting the initial state of A as q, i.e., $A_q=(Q_A, \Sigma_A, \rho_A, q, F_A)$.

Now, for any *ALC*-DL action π, there must be a NFA A such that $L(A)=L(\pi)$[7]. On the contrary, given any NFA $A=(Q, \Sigma, \rho, q_0, F)$ whose input alphabet Σ is composed of atomic actions and test actions of *ALC*-DL, an *ALC*-DL action π can also be constructed such that $L(\pi)=L(A)$. Based on this fact, we can replace each action of *ALC*-DL by a corresponding NFA, and combine Definition 2 and Definition 3 to the following definition.

Definition 8. Let Φ^0 be a set composed of all the general concept inclusion axioms, concept assertions and role assertions of *ALC*-DL. Use $\Phi^{\mathbf{A}}$ (resp., $\Pi^{\mathbf{A}}$) to denote the sets of all the formulae of *ALC*-DL (resp., all the NFAs of *ALC*-DL). Then, $\Phi^{\mathbf{A}}$ and $\Pi^{\mathbf{A}}$ are the smallest sets satisfying the following conditions:

(1) $\Phi^0 \subseteq \Phi^{\mathbf{A}}$;

(2) if $\varphi, \psi \in \Phi^{\mathbf{A}}$, then $\{\neg\varphi, \varphi \wedge \psi\} \subseteq \Phi^{\mathbf{A}}$;

(3) if $A \in \Pi^{\mathbf{A}}$ and $\varphi \in \Phi^{\mathbf{A}}$, then $<A>\varphi \in \Phi^{\mathbf{A}}$;

(4) if $A=(Q, \Sigma, \rho, q_0, F)$ is a NFA with $\Sigma = N_A \cup \{\varphi? \mid \varphi \in \Phi^{\mathbf{A}}\}$, then $A \in \Pi^{\mathbf{A}}$.

We also call formulas of the form $<A>\varphi$ as *action existence assertions*.

Correspondingly, the semantics defined by Definition 5 should be updated. It can be realized by two steps.

Firstly, rules (11), (12) and (13) of Definition 5 are replaced by the following single rule:

(11') $T(A):= \{(w,w') \in W \times W \mid$ there exist some positive integer $m \geq 0$, some word $l_1 \ldots l_m \in L(A)$ and m+1 states $u_0, u_1, \ldots, u_m \in W$, such that $u_0=w$, $u_m=w'$ and $(u_{i-1},u_i) \in l_i^T$ for every $1 \leq i \leq m\}$.

Secondly, rule (7) of Definition 5 is updated as follows:

(7') $(M,w) \models <A>\varphi$ iff there is some state $w' \in W$ such that $(w,w') \in A^T$ and $(M,w') \models \varphi$.

Obviously, from the point of view of representation and reasoning ability, the automata-based variant of *ALC*-DL presented here is equivalent with the logic defined in former section. Furthermore, for any *ALC*-DL action π in a regular expression form, the NFA A satisfying $L(A)=L(\pi)$ can be constructed in polynomial time, with a property that the number of states in A is linearly bounded by the size of π[7]. Therefore, the satisfiability problem of *ALC*-DL formulae can be studied based on the variant presented in this section, and all the results from this study are also suitable for the logic defined in former section.

4 Buchi tree automaton for *ALC*-DL formulae

Taken a similar approach used in the literatures of [3] and [6], we study the satisfiability problem of *ALC*-DL formulae by Buchi tree automaton. Given any *ALC*-DL formula φ, we will construct a Buchi tree automaton B_φ such that there exists a one-to-one correspondence between the models of φ and the language accepted by B_φ,

and by which the satisfiability problem of φ can be decided by checking whether the language accepted by B_φ is empty or not. Before going to more details, we firstly introduce some notations.

Definition 9. For any finite set Σ and any natural number k, let [k] denote the set {1,…,k}. A function T: $[k]^* \to \Sigma$ is called a k-ary Σ-tree. The empty word ε is called the root of the tree. Furthermore, for any $x \in [k]^*$ and $i \in [k]$, the node xi is called the i^{th}-child of the node x.

For any infinite non-empty word $\gamma \in [k]^\omega$, we use $\gamma[0]$ to denote the empty word ε, and use $\gamma[n]$ (where $n \geq 1$) to denote the word formed by the former n characters of the word γ. The word γ is also called a path of a k-ary Σ-tree. It is obvious that the path γ starts from the root ε, and sequentially goes through notes γ[1], γ[2], γ[3] and so on.

Definition 10. A Buchi tree automaton B on k-ary Σ-trees is a quintuple $B=(Q, \Sigma, \rho, I, F)$, where Q is a finite set of states, Σ is a finite alphabet, $\rho \subseteq Q \times \Sigma \times Q^k$ is the transition relation, $I \subseteq Q$ is the set of initial states, and $F \subseteq Q$ is the set of accepting states.

Let T be a k-ary Σ-tree. Then, a run of a Buchi tree automaton B on T is a k-ary Q-tree r:$[k]^* \to Q$ such that $r(\varepsilon) \in I$, and $(r(x), T(x), r(x1), \ldots, r(xk)) \in \rho$ for all nodes $x \in [k]^*$. A run r of B on T is accepting if and only if $inf(r, \gamma) \cap F \neq \varnothing$ for every path $\gamma \in [k]^\omega$, where $inf(r, \gamma)$ is a set composed of the states in Q that occur infinitely often along the path γ, i.e., $inf(r, \gamma) = \{q \in Q \mid$ for every $n \geq 0$, there always exists some $m \geq n$ with $\gamma[m]=q\}$.

Let B be a Buchi tree automaton on k-ary Σ-trees. Then, a k-ary Σ-tree T is accepted by B if and only if there exists an accepting run of B on T. The language accepted by B, denoted as $L(B)$, is the set of all the k-ary Σ-trees accepted by B.

Let φ be an *ALC*-DL formula for deciding whether it is satisfiable or not. We will construct a Buchi tree automaton B_φ for it. Some notations used in the construction is defined as follows.

Firstly, for any *ALC*-DL formula ψ, let sub(ψ) be a set composed of all the sub-formulae of ψ. More precisely, sub(ψ) is defined inductively as follows:

(1) if ψ is a general concept inclusion axiom, concept assertion or role assertion, then sub(ψ):=$\{\psi\}$;

(2) if ψ is of the forms $\neg\psi'$ or $<\pi>\psi'$, then sub(ψ):=$\{\psi\} \cup$sub(ψ');

(3) if ψ is of the form $\psi' \wedge \psi''$, then sub(ψ):=$\{\psi\} \cup$sub(ψ')$\cup$sub(ψ'').

Secondly, let cl(φ) be the smallest set satisfying the following conditions:

(1) $\varphi \in$cl(φ);

(2) if $\psi \in$cl(φ), then sub(ψ)$\subseteq$cl(φ);

(3) if $\psi \in$cl(φ), then $\neg\psi \in$cl(φ);

(4) if $<A>\psi \in$cl(φ), then $\psi \in$cl(φ) for each test action $\psi? \in \Sigma_A$;

(5) if $<A>\psi \in$cl(φ), then $<A_q>\psi \in$cl(φ) for each state $q \in Q_A$.

Finally, for any set $h \subseteq cl(\varphi)$, it is called an *ALC-Hintikka set* of φ if it satisfies the following five conditions:

(H1) for every formula $\psi \in sub(\varphi)$: $\psi \in h$ if and only if $\neg\psi \notin h$;

(H2) if $\psi \wedge \psi \in h$, then both $\psi \in h$ and $\psi \in h$;

(H3) if $\neg(\psi \wedge \psi) \in h$, then either $\neg\psi \in h$ or $\neg\psi \in h$;

(H4) if $\neg<A>\psi \in h$ and $q_A \in F_A$, then $\neg\psi \in h$;

(H5) if $\neg<A>\psi \in h$, then for any state $q \in Q_A$ and any test action $\psi? \in \Sigma_A$: it must be $\neg\psi \in h$ or $\neg<A_q>\psi \in h$ whenever $q \in \rho_A(q_A, \psi?)$.

Algorithm 1. Given any *ALC*-DL formula φ, let $\mathcal{H}_\varphi$ be a set composed of all the *ALC*-Hintikka sets of φ, let Π_φ be a set composed of all the atomic actions occurring in φ, let $\epsilon_1, \epsilon_2, \ldots, \epsilon_k$ be all the action existence assertions contained in $cl(\varphi)$, and let $\Lambda_\varphi = \mathcal{H}_\varphi \times (\Pi_\varphi \cup \{\perp\}) \times \{0,\ldots,k\}$. Then, the corresponding Buchi tree automaton $B_\varphi = (Q, \Lambda_\varphi, \rho, I, F)$ is a Buchi tree automaton on k-ary Λ_φ-trees, and is constructed as follows:

(1) $Q := \Lambda_\varphi \times \{\varnothing, \uparrow\}$;

(2) $I := \{ ((h, \pi, l), d) \in Q \mid \varphi \in h \text{ and } d = \varnothing \}$;

(3) $F := \{ ((h, \pi, l), d) \in Q \mid d = \varnothing \}$; and

(4) $(((h_0, \pi_0, l_0), d_0), (h, \pi, l), ((h_1, \pi_1, l_1), d_1), \ldots, ((h_k, \pi_k, l_k), d_k)) \in \rho$ if and only if

(i) $(h_0, \pi_0, l_0) = (h, \pi, l)$;

(ii) let T be a conjunction of all the (positive or negative) general concept inclusion axioms, (positive or negative) concept assertions, and (positive or negative) role assertions contained in h, then, as a Boolean knowledge base, T is consistent;

(iii) for each $1 \le i \le k$, if $\epsilon_i \in h_0$ and ϵ_i is of the form $<A>\psi$, then there exist some integer $n \ge 0$, some word $w = \psi_1?\ldots\psi_n? \in \Sigma_A^*$ composed of test actions, and some state $q_1 \in Q_A$ such that $\{\psi_1,\ldots,\psi_n\} \subseteq h_0$, $q_1 \in \rho_A(q_A, w)$ and either ①$q_1 \in F_A$, $\psi \in h_0$, $\pi_i = \perp$, $l_i = 0$, or ②there is some atomic action $\alpha \in \Sigma_A$ and some state $q_2 \in Q_A$ with $q_2 \in \rho_A(q_1, \alpha)$, $\epsilon_j = <A_{q2}>\psi \in h_i$, $\pi_i = \alpha$ and $l_i = j$;

(iv) for each $1 \le i \le k$, if there is some formula $\neg<A>\psi \in h_0$, some state $q \in Q_A$ and some atomic action $\alpha \in \Sigma_A$ with $q \in \rho_A(q_A, \alpha)$ and $\pi_i = \alpha$, then it must be $\neg<A_q>\psi \in h_i$;

(v) for each $1 \le i \le k$, $d_i = \uparrow$ if and only if either ①$d_0 = \varnothing$, $l_i \ne 0$ and $\epsilon_i \in h$, or ② $d_0 = \uparrow$, $l_i \ne 0$ and $l_0 = i$.

The following property holds for the Buchi tree automaton constructed above:

Theorem 1. An *ALC*-DL formula φ is satisfiable if and only if the language accepted by the corresponding Buchi tree automaton B_φ is not empty.

This theorem can be proved with a similar process presented in [3]. Due to space limitations, we omitted the proof here.

Let n be the length of φ. In the rest of this section, we give an analyses for the complexity of the satisfiability problem.

Firstly, according to the constructions presented above, the cardinality of cl(φ) is polynomial in n, and the cardinality of $\mathcal{H}_\varphi$ is at most exponential in n. Therefore, for the Buchi tree automaton B_φ= (Q, Λ_φ, ρ, I, F), cardinalities of both Λ_φ and Q are at most exponential in n. At the same time, since the number of action existence assertions contained in cl(φ) is polynomial in n, the cardinality of the relationship ρ is at most exponential in n. So, the size of the Buchi tree automaton B_φ is at most exponential in n.

Secondly, during the process of constructing the Buchi tree automaton B_φ, majority of the time is spent on step (4) for checking condition (ii). Since the cardinality of each *ALC*-Hintikka set h is polynomial in n, the size of the Boolean knowledge base T is also polynomial in n. So, the time for checking condition (ii) on step (4) is at most exponential in n. Since the cardinality of the relationship ρ is at most exponential in n too, to sum up, the time used for constructing B_φ is at most exponential in n.

Finally, since the emptiness problem for Buchi tree automaton is PTIME[6], the time used for deciding whether the language accepted by B_φ is empty or not is at most exponential in n.

To sum up, the complexity upper-bound for deciding whether an *ALC*-DL formula is satisfiable or not is EXPTIME. Together with the fact that the satisfiability problem for the propositional dynamic logic is already EXPTIME-complete, we get the following result:

Theorem 2. Satisfiability of *ALC*-DL formulae is EXPTIME-complete.

5 Related works

As a combination of the description logic ALC, the propositional dynamic logic PDL and an action theory based on possible models approach, a dynamic description logic named DDL is presented in [8] for describing and reasoning about actions. Compared with DDL, the logic *ALC*-DL is also capable for describing the preconditions and effects of atomic actions. For example, the formula $<\alpha>true\rightarrow\psi$ states that the formula ψ must be true for the action α to be executed; the formula $[\alpha]\varphi$ states that the formula φ will be true after the execution of the action α. However, in order to set up a complete action theory, some mechanisms out of *ALC*-DL should be introduced to deal with the frame problem and the ramification problem for action theories. On the other hand, general concept inclusion axioms contained in TBoxes of DDL are restricted to be invariable[9]. However, in *ALC*-DL, no restrictions are put on these knowledge described by the description logic ALC. To sum up, DDL can be treated as a special case of the logic *ALC*-DL.

By embracing the description logic ALC into the linear temporal logic LTL, a linear temporal description logic named *ALC*-LTL was proposed by Baader et al[5]. From the point of view of syntax, *ALC*-LTL was constructed by replacing all the atomic propositions of LTL with ABox assertions and TBox axioms of ALC. The satisfiability problem of *ALC*-LTL was investigated by Baader et al and proved to be decidable. In another paper, results on the complexity of this inference problem was

also demonstrated with the help of the general Buchi automaton[10]. Compared with *ALC*-LTL, the logic *ALC*-DL is constructed with a similar approach. However, since the interpretation structure of *ALC*-DL is more complex than that of *ALC*-LTL, we have to adopt Buchi tree automaton rather than general Buchi automaton to study the satisfiability problem.

Buchi tree automaton was firstly proposed by Vardi et al[6] and proved to have a PTIME complexity on the emptiness problem. The satisfiability problem on formulae of the deterministic propositional dynamic logic DPDL was also reduced to the emptiness problem by Vardi et al[6], and demonstrated to be EXPTIME. With a similar approach, Lutz et al[3] transformed the satisfiability problem of the propositional dynamic logic extended with negation of atomic actions into the emptiness problem of Buchi tree automaton, and by which demonstrated that the satisfiability problem was still decidable with the complexity of EXPTIME-complete. This paper is inspired by both Vardi et al.'s work and Lutz et al.'s work. In fact, the Buchi tree automaton constructed for *ALC*-DL formula can be treated as a variant of the Buchi tree automaton constructed by Lutz et al[3]. On the one hand, mechanism for dealing with negation of atomic actions is deleted from Lutz et al.'s Buchi tree automaton. On the other hand, some mechanisms for handling the knowledge described by the description logic ALC is introduced here.

6 Conclusion

The representation ability of *ALC*-DL is reflected in two aspects. On the one hand, propositions used in the propositional dynamic logic PDL are upgraded to formulas of the description logic ALC. Therefore, *ALC*-DL offers considerable expressive power going far beyond PDL, while reasoning is still decidable. On the other hand, *ALC*-DL extends the representation ability of the description logic ALC from static domains to dynamic domains. With *ALC*-DL, knowledge based described by ALC can be connected by actions and therefore evolutions of these knowledge based can be investigated and well organized.

One of our future work is to design efficient decision algorithms for *ALC*-DL. Another work is to put *ALC*-DL into application, by studying the description and reasoning of semantic Web services, and the evolution and organization of ontologies.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (Nos. 60903079, 60963010, 61163041), the Natural Science Foundation of Guangxi Province (No.2012GXNSFBA053169), and the Science Foundation of Guangxi Key Laboratory of Trusted Software (No. KX201109).

References

[1] Harel D, Kozen D, Tiuryn J. Dynamic Logic. Cambridge, MA: MIT Press, 2000.

[2] Giacomo G D, Massacci F. Combining deduction and model checking into tableaux and algorithms for Converse-PDL. *Information and Computation*, 2000, 162(1-2): 117-137.
[3] Lutz C, Walther D. PDL with negation of atomic programs. *Journal of Applied Non-Classical Logic*, 2005, 15(2): 189-214.
[4] Göller S, Lohrey M, Lutz C. PDL with intersection and converse is 2EXP-complete. In: Proceedings of the 10th International Conference on Foundations of Software Science and Computation Structures. Berlin: Springer-Verlag, 2007: 198-212.
[5] Baader F, Ghilardi S, Lutz C. LTL over description logic axioms. In: Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning. Cambridge: AAAI Press, 2008. 684-694.
[6] Vardi M Y, Wolper P. Automata-theoretic techniques for modal logics of programs. *Journal of Computer and System Sciences*, 1986, 32(2): 183-221.
[7] Hromkovic J, Seibert S, Wilke T. Translating regular expressions into small ε-free nondeterministic finite automata. In: Proceedings of the 14th Annual Symposium on Theoretical Aspects of Computer Science. Belin: Springer, 1997: 55-66.
[8] Shi Z Z, Dong M K, Jiang Y C, Zhang H J. A logical foundation for the semantic Web. *Science in China, Ser. F: Information Sciences*, 2005, 48(2): 161-178.
[9] Chang L, Shi Z Z, Gu T L, Zhao L Z. A family of dynamic description logics for representing and reasoning about actions. *Journal of Automated Reasoning*, 2012, 49(1): 19-70.
[10] Baader F, Bauer A, Lippmann M. Runtime verification using a temporal description logic. In: Proceedings of the 7th International Symposium on Frontiers of Combining Systems. Berlin: Springer-Verlag, 2009: 149-164.