Verification of Branch-Time Property Based on Dynamic Description Logic

Yaoguang Wang, Liang Chang, Fengying Li, Tianlong Gu

Verification of Branch-time Property Based on Dynamic Description Logic

Yaoguang Wang, Liang Chang, Fengying Li, Tianlong Gu

Guangxi Key Laboratory of Trusted Software,
Guilin University of Electronic Technology,Guilin 541004,China
wangyguang@qq.com,{changl,lfy,cctlgu}@guet.edu.cn

Abstract. The dynamic description logic DDL provides formalism for describing dynamic system in the semantic Web environment Model checking is a formal verification method based on state transition system. In this paper, we bring dynamic description logic into model checking. Firstly, state transition systems considered in model checking are modeled as complex actions in dynamic description logic. Secondly, a kind of temporal description logic DL-CTL is introduced to specify temporal properties on state transition systems, where DL-CTL is a DL-based extension of propositional branch-time temporal logic CTL. Finally, verification algorithm is presented with the help of reasoning mechanisms provided by description logic.

Keywords: dynamic description logic, verification, temporal description logic, action theory.

1 Introduction

Model checking [1] is a formal verification method widely used in recent years, which is based on state transition system. However, the traditional model checking has some limitations. Firstly, it does not consider what makes state change. Secondly, it just uses temporal logic based on proposition logic to specify the property, which limits the scope of specifying properties. So, researchers begin to combine the action theory [2] to verification problem and consider the action make the state change.

Verification problem with action theory has been addressed by some researchers. In [3], the author puts forward a method of verifying temporal properties based on infinite sequence of Golog program and checks whether the execution of program sequence can satisfy temporal properties. In [4], the author aims at the fully automated verification of non-terminating Golog programs and uses an extension of situation calculus by constructing the first-order temporal logic CTL*. However, the problems both of them consider are all undecidable. For this reason, [5] begins to use the action theory based on decidable description logic to check whether there is an execution sequence of action can satisfy the temporal property specified in linear temporal description logic DL-LTL [6].

However, [5] just considers the atom action and only the linear-time properties can be verified. Based on this limitation, we consider the action in dynamic description logic that contains action constructors like sequence, choice, iterator or test action. For the reason that the action in DDL contains choice action constructor, it becomes possible that the action can make the state change in branch structure. The verification problem becomes whether there is a model generated by an execution complex action that meets temporal formula specified in branch temporal description logic DL-CTL. Instead of considering the actual execution sequences of actions, we consider execution complex action sequences accepted by a given non- deterministic finite automaton NFA. If a NFA is an abstraction of the action, i.e. all possible execution sequences of the action are accepted by NFA, then any property that holds in all the actions accepted by NFA is also a property that is satisfied by any execution of the actions. Therefore, we not only add the action in dynamic description logic to verification problem, but also can verify the branch-time properties.

2 Preliminaries

2.1 Temporal Description Logic DL-CTL

Description logics [5-7] are a well-known family of formalisms to represent the knowledge, which offers the considerable expressive power going far beyond the propositional logic and the reasoning is still decidable. DL-CTL is the temporal extension to description logics, which extends the propositional branch-time logic (CTL) by allowing for the use of axioms of the basic description logic in place of propositional letters. The properties in this paper will be expressed in DL-CTL.

The concepts in DL-CTL is similar with DL, which are inductively defined a set N_C of concept name, a set N_R of role name, and a set N_I of individual name. The concept construction is the same as those do in DL and the formula is constructed with the temporal operators in CTL. At the same time, the propositional letters in CTL are replaced by ABox assertions of description logic.

Definition 1. DL-CTL formula is defined as follows：

$$\phi, \psi ::= C \sqsubseteq D \mid C(p) \mid R(p, q) \mid \neg\phi \mid \phi \wedge \psi \mid \mathrm{E}\boldsymbol{X}\,\phi \mid \mathrm{A}\boldsymbol{F}\,\phi \mid \mathrm{E}(\phi\mathbf{U}\psi)$$

where p, q$\in N_I$, R$\in N_R$, C,D are concept name. We can also introduce the formula such as false, true, $\phi\vee\psi$, $\phi\rightarrow\psi$, A**X** ϕ, E**F** ϕ, E**G** ϕ, A**G** ϕ, A(ϕ**U**ψ), E(ϕ**R**ψ), A(ϕ**R**ψ).

The semantic of DL-CTL is similar with CTL and it is based on the structure, in which their states are organized by branch structure. However, the different from CTL is that the state in DL-CTL is not mapped to a set of propositional letters but mapped to DL interpretations. Thus, the state change in state transition system can be viewed as an interpretation change in DL-CTL.

Definition 2. DL-CTL structure is a tetrad $M = (\mathcal{S}, \mathrm{T}, \Delta, I)$:

(1) $\mathcal{S}$ is a set of all states;

(2) $T \subseteq \mathcal{S} \times \mathcal{S}$ is binary relation of state which means the transition between two states;

(3) Δ is the interpretation domain;

(4) For every state $s \in \mathcal{S}$, the function I gives s a DL interpretation $I(s) = (\Delta, \cdot^{I(s)})$ and the interpretation function $\cdot^{I(s)}$ must meet the following conditions:

(i) For every concept $C_i \in N_C$, there is $C_i^{I(s)} \subseteq \Delta$;

(ii) For every role name $R_i \in N_R$, there is $R_i^{I(s)} \subseteq \Delta \times \Delta$;

(iii) For every individual name $p_i \in N_I$ there is $p_i^{I(s)} \in \Delta$, and for every state $s' \in \mathcal{S}$ there is $p_i^{I(s)} = p_i^{I(s')}$.

Definition 3. A DL-CTL structure is M=($\mathcal{S}$, T,Δ, I), the semantic of concept and formula in DL-CTL are inductively defined as follows.

Firstly, for every states $s \in \mathcal{S}$, a concept C is interpreted $C^{I(s)}$, which is a subset of Δ.

(1) $(\neg C)^{I(s)} := \Delta \setminus C^{I(s)}$;

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

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

Secondly, for every states $s \in \mathcal{S}$, M, $s \vDash \phi$ means ϕ holds at state s of the structure M.

(4) $(M, s) \vDash C \sqsubseteq D$ iff $C^{I(s)} \subseteq D^{I(s)}$;

(5) $(M, s) \vDash C(p)$ iff $p^{I(s)} \in C^{I(s)}$;

(6) $(M, s) \vDash R(p, q)$ iff $(p^{I(s)}, q^{I(s)}) \in R^{I(s)}$;

(7) $(M, s) \vDash \neg\phi$ iff $(M, s) \nvDash \phi$;

(8) $(M, s) \vDash \phi \wedge \psi$ iff $(M, s) \vDash \phi$ and $(M, s) \vDash \psi$;

(9) $(M, s) \vDash \mathrm{E}\boldsymbol{X}\,\phi$ iff there is a state s' and (s, s') $\in$T and $(M, s') \vDash \phi$;

(10) $(M, s) \vDash \mathrm{A}\boldsymbol{F}\,\phi$ iff for every path starting with s, there is always a state s' so that sT*s' and $(M, s') \vDash \phi$;

(11) $(M, s) \vDash \mathrm{E}(\phi \mathbf{U} \psi)$ iff there exists a path starting with s, and there exists an integer $k \geq 0$ so that $(M, s+k) \vDash \psi$ and for every $0 \leq i < k$, $(M, s+i) \vDash \phi$.

In this paper, we consider that the state transition is caused by the application of action. For the reason that every state in DL-CTL structure is mapped to a DL interpretation, we can say that the change of the interpretation of one state to the next is also caused by action.

2.2 Dynamic Description Logic DDL

The action theory based on description logic is inherited the advantage of the action theory and the description logic, which not only has the more expressive power but also makes the reasoning tasks decidable. Dynamic description logic (DDL) is a kind of action theory based on description logic, which is proposed by Chang. L et al [8], who has introduced complex action to the action theory based on description logic.

In this paper, we consider the action theory based on dynamic description logic and the basic definitions of action will be given below.

Definition 4. Let $\mathcal{T}$ is an acyclic TBox. An action for $\mathcal{T}$ is generated below.

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

where α is an atom action and φ is an assertion. $\varphi?$, $\pi \cup \pi'$, π,π', π^* are respectively called test action, choice action, sequential action and iterated action.

The complex action is composed by these actions sequence. For example, (φ?), a∪b, (c, d)* is the complex action and a, b, c, d are respectively the atom action and φ is an assertion.

An atom action is a triple (**pre**, **occ**, **post**), and the detail description of atom action can be found in [7]. In this paper, we just consider the action without **occ** for convenience.

We say that α is executable in an interpretation I if I is a model of **pre**. If the execution of action can change the interpretation I to I′, we can say that α makes I⇒I′.

In the existing researches, action is considered as the cause of the state transition, which also means that action is the binary relation between states. For the reason that the action in dynamic description logic contains choice constructor with which the action can make the state change in branch-time state transition system, it becomes possible to verify the branch-time property based on dynamic description logic. Next, some necessary definitions corresponding to this case will be given below.

Definition 5. For every complex action π, let Σ be the set of the atom action or test action in π. Every element in Σ can be view as a word and let L(π) be the minimal set of string that defined by the following rules:

(1) If π is an atom action or test action then, L (π) = π;

(2) L ($\pi \cup \pi'$) =L (π)∪L(π');

(3) L (π, π') = $\{l_1 l_2 | l_1 \in L(\pi)$ and $l_2 \in L(\pi')\}$;

(4) $L(\pi^*)=L(\pi^0)\cup L(\pi^1)\cup L(\pi^2)\cdots$, where $L(\pi^0)$ is an empty string and for every i ≥1,there is $L(\pi^i)=L(\pi^{i-1},\pi)$.

So, each string in L(π) corresponds one of the action execution sequence of π.

Definition 6. For a complex action π and one of its action execution sequence l_i, where $l_i \in L(\pi)$, we use $|l_i|$ to denote the length of action execution sequence, $l_i(j)$ to denote the j-th atom or test action in i-th action execution sequence.

For example, for a complex action $\pi=a,b,c,d$, L(π)=$\{l_1\}$.That is to say, l_1 is the only action execution sequence and $l_1=abcd$; For another complex action$\pi=(a,b)\cup(c,d,e)$, L(π)=$\{l_1,l_2\}$, l_1 and l_2 are the two action execution sequences of π and $l_1=ab$, $l_2=cde$, $l_1(1)=a$, $l_2(2)=d$.

Definition 7. Let $\mathcal{T}$be an acyclic TBox, $\mathcal{A}$ an ABox, and π a complex action for $\mathcal{T}$. For the interpretation of a state s_0, there is $I(s_0)\vDash\mathcal{A}$, then for an execution action sequence l_i of π and a path starting from s_0, if $l_i(j)$ is executable in $I(s_j)$ and it makes $I(s_j)\Longrightarrow I(s_{j+1})$, then, we call this path is a path generated by l_i. A DL-CTL structure generated by π is a tree structure where s_0 is the root and it contains all paths generated by each corresponding execution action sequence of π.

In this paper, for verification problem based on action, we consider whether there is a model M generated by an execution complex action that meets the property specified in DL-CTL. According to automaton theory, every action π can be constructed to a non-deterministic finite automaton (NFA). Instead of considering the actual execution sequences of actions, we abstract the complex action to non-deterministic finite state automaton (NFA) and just consider the action accepted by a NFA.

Definition 8. A= (Q, Σ, δ, q_0, F) is a non-deterministic finite automaton and Q is a set of state, Σ is the alphabet, δ: $Q\times \Sigma \rightarrow 2^Q$ is a transition function, $q_0 \in Q$ is initial state and $F \subseteq Q$ is the final state set. We can say that A is a NFA for Σ and Σ is the set of the atom action or test action. The language accepted by A is L (A), which also can be treated as L (π).

According to the theory of formalism, the verification problem can be considered as satisfiability problem, which asks whether there is a complex action π accepted by NFA that satisfies the property specified in DL-CTL. The formal definition of this problem is shown in Definition 9 and an actual example will be given to this problem in the following section.

Definition 9. Let $\mathcal{T}$ be an acyclic TBox, $\mathcal{A}$ an ABox, and Σ a finite set of action for $\mathcal{T}$. $\mathcal{B}$ is a NFA for the alphabet Σ, and φ a DL-CTL formula. φ is satisfiable w.r.t $\mathcal{T}$, $\mathcal{A}$, and $\mathcal{B}$ if there is a DL-CTL structure M generated by π from $\mathcal{A}$ w.r.t $\mathcal{T}$ such that M, $s_0 \vDash \varphi$.

2.3 An example

An example of buying a book will be given to show the problem we have discussed above. Assume the fact is: Jim wants to buy book A and B in bookstore, if the bookstore does not have the two books, it has to order them and then Jim can buy the two books at the same time.

According to the fact, the property can be described like this: though the bookstore does not have the two books, Jim can eventually get them. Obviously, if Jim wants to buy the two books, the bookstores must have the two books; if not, the bookstore must order the two books. If the bookstores just order one of them, Jim still can't buy the two books. Based on these facts, we give a formal definition of this case.

Firstly, we give a basic definition and symbol in this case, where the concept set N_C={student, book, instore}, the role set N_R={bought, has}, the individual name set N_I={Jim,book_a,book_b},The action set:{buyBook_a,buyBook_b,order_a,order_b}.

Based on these definitions, the background knowledge can be described below.

$$\textit{student} \equiv \textit{person} \sqcap \exists \textit{has.books}$$

Then, we define the actions given above.

buyBook_a ≡({student (Jim), book (book_a), instore (book_a)}, {¬ instore (book_a), bought (Jim, book_a)});
buyBook_a ≡({student (Jim), book (book_b), instore (book_b)}, {¬ instore(book_b), bought (Jim, book_b)});
order_a ≡({book (book_a), ¬instore (book_a)}, {instore (book_a)});
order_b ≡({book (book_b), ¬instore (book_b)}, {instore (book_b)});

The property described in this example can be specified in following DL-CTL formula φ.

$$EF(\neg instore(book\_a) \land \neg instore(book\_b) \rightarrow EF(bought(Jim,book\_a) \land bought(Jim,book\_b)))$$

From the fact we know that if Jim want to buy the two books, the bookstore must have the two books; if not, the bookstore must order them, and Jim can go to buy the book and the get the book. That is to say, as long as the action *order_a*, *order_b*, *buyBook_a*, *buyBook_b* exists in complex action π, the property φ will be satisfied.

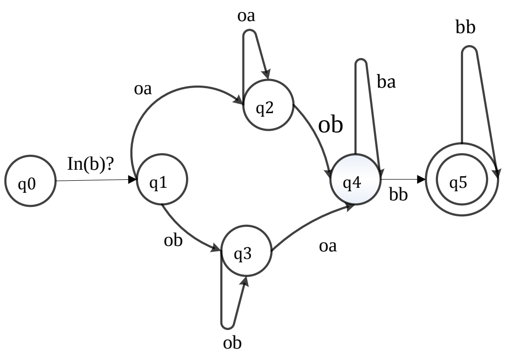

Fig. 1. a non-deterministic finite automaton $B_{buyBook}$

The NFA $B_{buyBook}$ is depicted in Figure 1. The alphabet of $B_{buyBook}$ is Σ, which consists of the actions defined above. For the convenience of describing the actions, the actions *instore (book)?*, *buyBook_a*, *buyBook_b*, *order_a*, *order_b* are abbreviated with *In(b)?, ba ,bb ,oa ,ob*.

For action π=In(b)?(((oa∪ob)(oa)*,(ob),(ba)*,(bb)*, it is easy to check that one of its action execution sequence *In(b)oa,oa(oa…)ob,ba,(bb…)*∈L($B_{buyBook}$) can generate a branch structure that can satisfy the property that Jim eventually get the two books described above.

3 Verification algorithm based on DDL

In this paper, we consider a restricted situation that action contains cyclic sequence of action, which is similar to the case given in [5]. We solve this problem defined above by the reduction from the satisfiability problem of DL-CTL formula to the consistency problem introduced in [7]. The detail algorithm is given in Algorithm 1 and the detail construction of each step will be given in the following section.

Algorithm 1. *Given an ABox $\mathcal{A}$, a TBox $\mathcal{T}$, an action π and a formula ϕ, whether there is a model M generated by π that meets ϕ can be decided by the following steps:*

Step 1. *Construct an acyclic TBox $\mathcal{T}_{red}$, and an ABox $\mathcal{A}_{red}$ from $\mathcal{A}$, $\mathcal{T}$, π and ϕ;*

Step 2. *Construct an ABox $\mathcal{A}_{pre}$ from π;*

Step 3. *Using tableau rules to compute an ABox $\mathcal{A}\phi$ from ϕ;*

Step 4. *Using the reasoning services provided by description logic to decide whether $\mathcal{A}_{red}\cup\mathcal{A}_{pre}\cup\mathcal{A}\phi$ is consistent w.r.t $\mathcal{T}_{red}$, if $\mathcal{A}_{red}\cup\mathcal{A}_{pre}\cup\mathcal{A}\phi$ is consistent w.r.t $\mathcal{T}_{red}$, return 'yes', else return 'no'.*

3.1 Construction of T$_{red}$

Firstly, we assume that there are no DL-CTL negation signs in φ, which is similar with the method in [10] when dealing with LTL negation signs. It allows DL-CTL signs occur only in front of ABox assertion rather than temporal operator.

Secondly, we refer to the method for solving the projection problem in [10] to the finite sequence of atom action to construct $\mathcal{T}_{red}$ and $\mathcal{A}_{red}$.

In order to define $\mathcal{T}_{red}$, we define $\mathcal{T}_N$, which contains all individual names in the input.

$$\mathcal{T}_N = \{N \equiv \sqcup\{a\},\ \text{for all } a \in N_I\} \qquad (1)$$

Let *Sub* be the set of the subconcepts in the input. For every C∈*Sub*, if C is not a defined concept name of T, then there is a concept definition of $\mathcal{T}_C^i$ and $\mathcal{T}_{Sub}^i$. Moreover, T_{Sub}^i contains only those concept definitions. The concept definition of T_C^i can be found in [10].

Now, according to the method given in [5], we are ready to assemble $\mathcal{T}_{red}$:

$$\mathcal{T}_{red} = \mathcal{T}_N \cup (\cup_i^{m+2n-1} \mathcal{T}_{Sub}^i) \cup \{\mathcal{T}_A^i \equiv \mathcal{T}_E^i | A \equiv E \in \mathcal{T},\ \text{for } i \leq m+2n-1\} \qquad (2)$$

TBox $\mathcal{T}_N$ and $\mathcal{T}_{Sub}^i$ can ensure that the interpretations of concept and role names remain unchanged by actions on the anonymous objects and the last part of $\mathcal{T}_{red}$ is to make sure that $\mathcal{T}$ is satisfied no matter how actions change an interpretation.

3.2 Construction of $\mathcal{A}_{red}$ and $\mathcal{A}_{pre}$

$\mathcal{A}_{red}$ is an ABox which record the changes by actions on the named objects. For every ABox assertion ϕ, we define $\phi^{(i)}$. If ϕ=C(a), $\phi^{(i)} = \mathcal{T}_C^i$ and if ϕ =r(a,b), $\phi^{(i)} = r^{(i)}$ (a,b).

In order to meet the semantic of action, we need to get the pre-definitions of A_{post}^i, $\mathcal{A}_{min}^i$, $\mathcal{A}_{ini}$,which can be found in [10], and then we can get ABox $\mathcal{A}_{red}$:

$$\mathcal{A}_{red} = \mathcal{A}_{ini} \cup (\cup_i^{m+2n-1} \mathcal{A}_{post}^i) \cup (\cup_i^{m+2n-1} \mathcal{A}_{min}^i) \qquad (3)$$

In [5], we know that from every model of $\mathcal{A}_{red}$ and $\mathcal{T}_{red}$, we can construct the cruial part of a DL-CTL structure generated by π from $\mathcal{A}$ and $\mathcal{T}$. We can also see that any finite sequence $I(s_0)....I(s_{m+2n-1})$ satisfy the property stated in the above items can be extended to an DL-CTL structure generated by $\pi = \pi_1 \cdots \pi_m (\pi'_1 \cdots \pi'_n)^\pi$ from $\mathcal{A}$ w.r.t $\mathcal{T}$ by setting $I(s_{m+kn+i}) = I(s_{m+n+i})$ for all k≥2 and $0 \leq i < n$.

To enforce the excitability of the execution action sequence l_i, which $l_i(j)$ is the j-th atom or test action in $l_i(j)$, we define ABox $\mathcal{A}_{pre}$, which is similar with the method given in [5] dealing with the atom actions.

$$\mathcal{A}_{pre} = \cup\{\gamma^{(j)} | \gamma \in pre_j,\ \text{for } 0 \leq j \leq m+2n-1\} \qquad (4)$$

where pre_j is the set of pre-conditions of l_i , the excution action sequence of π.

3.3 Construction of $\mathcal{A}\phi$

In order to ensure that the DL-CTL formula φ is satisfied, we generate additional ABox $\mathcal{A}\phi$ by applying a non-deterministic tableau algorithm. We have time-stamped copied $\phi^{(i)}$ for every subformula φ of ϕ, which means that ϕ holds at time point i.

Different from the method for solving the semantic the temporal operator of LTL given in [10], we use the approach given below to solve the problem of DL-CTL with these tableau rules. The tableau algorithm starts with an initial set $\mathcal{S}=\{\phi^{(0)}\}$, and then modifies this set by tableau rules until there are no more rules to apply, part of the tableau rules are described below and other rules can be inferred by these rules.

(1) $\neg\neg$rule: If $(\neg\neg\phi)^{(i)} \in \mathcal{S}$ and $(\phi)^{(i)}\notin \mathcal{S}$, let $(\phi)^{(i)} \in \mathcal{S}$;

(2) $\wedge$ rule: If $(\phi_1\wedge\phi_2)^{(i)} \in \mathcal{S}$and $\{(\phi_1)^{(i)}, (\phi_2)^{(i)}\}\notin \mathcal{S}$, let $\{(\phi_1)^{(i)}, (\phi_2)^{(i)}\} \in \mathcal{S}$;

(3) $\neg\wedge$ rule: If $\neg(\phi_1\wedge\phi_2)^{(i)} \in \mathcal{S}$ and $(\neg\phi_1)^{(i)}\notin \mathcal{S}$, let$(\neg\phi_1)^{(i)}\in \mathcal{S}$; If$\neg(\phi_1\wedge\phi_2)^{(i)} \in \mathcal{S}$and $(\neg\phi_2)^{(i)}\notin \mathcal{S}$,let $(\neg\phi_2)^{(i)}\in \mathcal{S}$;

(4) $\neg$EX rule: If $(\neg EX\phi_1)^{(i)} \in \mathcal{S}$ and $(AX\neg\phi_1)^{(i)}\notin \mathcal{S}$,let $(AX\neg\phi_1)^{(i)} \in \mathcal{S}$, and let $(\neg\phi_1)^{(i+1)} \in \mathcal{S}$ in all possible branches at the next time point ,then remove $(AX\neg\phi_1)^{(i)}$, $(\neg EX\phi_1)^{(i)}$;

(5) $\neg$AX rule: If $(\neg AX\phi_1)^{(i)} \in \mathcal{S}$ and $(EX\neg\phi_1)^{(i)}\notin \mathcal{S}$, let $(EX\neg\phi_1)^{(i)} \in \mathcal{S}$, and let $(\neg\phi_1)^{(i)}\in \mathcal{S}$ in next time point in all branches at i+1, and all the branches are respectively recorded by l_1,l_2...

(6) EG rule: If $(EG\phi)^{(i)} \in \mathcal{S}$ and $(\phi)^{(i)}\notin \mathcal{S}$,and use $(\phi)^{(i)}$, $(\phi)^{(i+1)}\cdots$to lable all the states aftertimeiandlet$\{(\phi)^{(i)},(\phi)^{(i+1)}\cdots(\phi)^{(j)}\} \in \mathcal{S}$ ($j\le m+2n-1$), then remove$(EG\phi)^{(i)}$ and all the possible branches are recorded by l_1,l_2...

(7) AG rule: If $(AG\varphi)^{(i)} \in \mathcal{S}$ and $(\varphi)^{(i)}\notin \mathcal{S}$, use $(\phi)^{(i)}$, $(\phi)^{(i+1)}\cdots$to lable all the states after time i and let $\{(\phi)^{(i)},(\phi)^{(i+1)}\cdots(\phi)^{(j)}\} \in \mathcal{S}$ （$j\le m+2n-1$） ,then remove$(EG\phi)^{(i)}$.

(8) EU rule: If $(E(\phi_1U\phi_2))^{(i)} \in \mathcal{S}$,for i there are two conditions: $i\le m+n$ and $i>m+n$. When $i>m+n$, use $(\phi)^{(i)}$, $\cdots(\phi)^{(k-1)}$,$(\phi)^{(k)}$ to lable all the states after time i and make $\{(\phi)^{(i)}...(\phi)^{(k-1)}\ (\phi)^{(k)}\} \in \mathcal{S}$, then remove $(E(\phi_1U\phi_2))^{(i)}$, when $i \le m+n$,use $(\phi)^{(i)}$, $\cdots (\phi)^{(m+2n-1)}$,$(\phi)^{(m+n)}$, $\cdots (\phi)^{(k-1)}$,$(\phi)^{(k)}$ to lable all the states after time i and make$\{(\phi)^{(i)}, \cdots(\phi)^{(m+2n-1)},(\phi)^{(m+n)}, \cdots(\phi)^{(k-1)},(\phi)^{(k)}\} \in \mathcal{S}$, then remove $(E(\phi_1U\phi_2))^{(i)}$. For the two conditions, use l_1,l_2...to record the existing branches.

(9) AU rule: If $(A(\phi_1U\phi_2))^{(i)} \in \mathcal{S}$,for i there are two conditions: $i\le m+n$ and $i>m+n$. When $i>m+n$, use $(\phi)^{(i)}$, $\cdots(\phi)^{(k-1)}$,$(\phi)^{(k)}$ to lable all the states after time i and make $\{(\phi)^{(i)}\cdots\ (\phi)^{(k-1)}\ (\phi)^{(k)}\} \in \mathcal{S}$, then remove $(A(\phi_1U\phi_2))^{(i)}$,when $i \le m+n$, use $(\phi)^{(i)}$, $\cdots(\phi)^{(m+2n-1)}$,$(\phi)^{(m+n)}$, $\cdots(\phi)^{(k-1)}$,$(\phi)^{(k)}$ to lable all the states after time i and make$\{(\phi)^{(i)}, \cdots(\phi)^{(m+2n-1)},(\phi)^{(m+n)}, \cdots(\phi)^{(k-1)},(\phi)^{(k)}\} \in \mathcal{S}$,then remove $(A(\phi_1U\phi_2))^{(i)}$.

(10) $\neg$EU rule: If $(\neg E(\phi_1U\phi_2))^{(i)} \in \mathcal{S}$ and at the same time $(\neg\phi_1)^{(i)}\notin \mathcal{S}$,$(\neg\phi_2)^{(i)}\notin \mathcal{S}$, let $\{(\neg\phi_2)^{(i)},(\neg\phi_2)^{(i)}\phi_1\}\in \mathcal{S}$; If$(\neg E(\phi_1U\phi_2))^{(i)} \in \mathcal{S}$ and $(\neg\phi_2)^{(i)}\notin \mathcal{S}$,at the same time, $(AX\neg E(\phi_1U\phi_2))^{(i)} \notin \mathcal{S}$, let $\{(\neg\phi_2)^{(i)},(AX\neg E(\phi_1U\phi_2))^{(i)}\}\in \mathcal{S}$,then,use the rule AX and EU defined above.

(11) $\neg$AU rule: If $(\neg A(\phi_1U\phi_2))^{(i)} \in \mathcal{S}$ and $(\neg\phi_1)^{(i)} \notin \mathcal{S}$,$(\neg\phi_2)^{(i)} \notin \mathcal{S}$,let $\{(\neg\phi_2)^{(i)}$, $(\neg\phi_2)^{(i)}\phi_1\}\in \mathcal{S}$; If $(\neg A(\phi_1U\phi_2))^{(i)}\in \mathcal{S}$ and $(\neg\phi_2)^{(i)}\notin \mathcal{S}$,$(EX\neg A(\phi_1U\phi_2))^{(i)} \notin \mathcal{S}$, let $\{(\neg\phi_2)^{(i)}$, $(EX\neg A(\phi_1U\phi_2))^{(i)}\}\in \mathcal{S}$, then,use the rule EX and AU defined above .

(12) $\neg$EG rule: If $(\neg EG\phi_1)^{(i)} \in \mathcal{S}$ and $(\neg\phi_1)^{(i)}\notin \mathcal{S}$,let $(\neg\phi_1)^{(i)}\in \mathcal{S}$; If $(\neg EG\phi_1)^{(i)}\in \mathcal{S}$ and $(AX\neg EG\phi_1)^{(i)}\notin \mathcal{S}$,let $(AX\neg EG\phi_1)^{(i)} \in \mathcal{S}$, then,use the rule AX and EG defined

above and use l_1,l_2...to record the existing branches..

(13) $\neg$AG rule: If $(\neg AG\phi_1)^{(i)} \in \mathcal{S}$and $(\neg\phi_1)^{(i)} \notin \mathcal{S}$,let $(\neg\phi_1)^{(i)}\in \mathcal{S}$; If $(\neg AG\phi_1)^{(i)}\in \mathcal{S}$ and $(EX\neg AG\phi_1)^{(i)} \notin \mathcal{S}$, let$(EX\neg AG\phi_1)^{(i)} \in \mathcal{S}$, then use the rule EX and AG defined above.

It can be shown that the tableau rules always terminate with a finite set $\mathcal{S}$, which contains only time-stamped DL-assersions and the final $\mathcal{S}$ is an ABox. Since there exists the branch time operator, it will generate not only one ABox, depending by the choices made in the rules. We say that $\mathcal{A}\phi$ is induced by ϕ w.r.t π if it is one of the ABoxes produced by applying the above rules to $\{\phi^{(0)}\}$.

In this case, we introduce the verification problem based on dynamic description logic and conisder its dual, the satisfiability problem ,which is introduced in Definition 7. Finally, we reduce this problem to consistency of an ABox w.r.t an acyclic Tbox.

Theorem 1. The DL-CTL formula ϕ is satisfiable w.r.t $\mathcal{T}, \mathcal{A}$ and an NFA $\mathcal{B}$ iff there is an ABox $\mathcal{A}\phi$ induced by ϕ w.r.t π such that $\mathcal{A}_{red}\cup\mathcal{A}_{pre}\cup\mathcal{A}\phi$ is consisitent with $\mathcal{T}_{red}$.

This theorem can be proved with a similar process presented in [10]. Due to space limitations, we omitted the proof here.

4 Conclusions

Traditional verification technology is based on transition system and the property is specified in propositional logic, which limits the scope of describing the property. So, the temporal description logic DL-LTL is put forward to specify the temporal property in [6]. Based on DL-LTL, F.Baader considers runtime verification problem in [9], which observes changes to the state without knowing how they are caused.

In order to explore the cause of state transition, [5] assumes the action can make the state change and then combine the decidable action theory based on description logic to decide whether there is an infinite execution atom actions that can satisfy the linear property specified in DL-LTL. At last, a specific approach is given for solving this problem.

However, [5] just considers the atom actions and just can verify the liner property. For such limitations, we consider complex action based on dynamic description logic and use the temporal description logic DL-CTL to specify the properties. For the verification problem whether the property specified in DL-CTL holds in a model generated by a complex action, we consider its dual, the satisfiability problem whether there is an execution of complex action that can satisfy the property. For the convenience of this problem, we abstract complex action to a non-deterministic finite state automaton (NFA) and consider whether there is a complex action accepted by NFA that satisfy the temporal property. Finally, we reduce this verification problem to consistency problem in description logic and give an approach to it.

In this paper, we abstract the action to a non-deterministic finite state automaton and every possible execution actions are accepted by this NFA. A future work will consider the actual action rather than its abstraction.

Acknowledgements. This work is supported by the National Natural Science Foundation of China (Nos. 61363030, 61262030), the Natural Science Foundation of Guangxi Province (No.2012GXNSFBA053169, 2012GXNSFAA053220) and the Science Foundation of Guangxi Key Laboratory of Trusted Software.

References

1. Clarke, E.M., Grumberg, O., Peled, D.A.: Model Checking. MIT Press, 1999.
2. Reiter, R.: Knowledge in action: logical foundations for describingand implementing dynamical systems. Cambridge, MA: MIT Press, 2001.
3. Giacomo, G.D., Ternovskaia, E., Reiter, R.: Non-terminating processes in the situation calculus. In: Proceedings of the AAAI'97 Workshop on Robots, Softwoods, Immobots: Theories of Action, Planning and Control, 1997.
4. Claβen, J., Lakemeyer, G.: A Logic for non-terminating Golog programs. In: Proceedings KR 2008, pp. 589–599. AAAI Press, 2008.
5. Baader, F., Liu, H.K., ul Mehdi, A.: Verifying properties of infinite sequences of description logic actions. In: Proceedings of ECAI'10, 2010.
6. Baader, F., Ghilardi, S., Lutz, C.: LTL over description logic axioms. In: Proceedings of KR 2008, pp. 684–694. AAAI Press, Cambridge, 2008.
7. Baader, F., Lutz, C., Sattler, U., Wolter, F.: Integrating description logics and action formalisms: First results. In: Proceedings AAAI'05, 2005.
8. Chang L, Shi ZZ, Gu TL, Zhao LZ.: A family of dynamic description logics for representing and reasoning about actions. Journal of Automated Reasoning 49(1), 19–70, 2012.
9. Baader, F., Bauer, A., Lippmann, M.: Runtime verification using a temporal description logic. In: Proceedings of FroCoS 2009, pp. 149–164, 2009.
10. Baader, F, Liu, H.K.: Intergrate Action Formalisms into Linear Temporal Description Logic. LTCS-Report 09-03, TU Dresden, Germany, 2009.