Contribution Title*

Renyan Feng[1[0000−1111−2222−3333]], Erman Acar[3[2222−−3333−4444−5555]], Stefan Schlobach[3[2222−−3333−4444−5555]], and Yisong Wang[2,3[1111−2222−3333−4444]]

[1] Princeton University, Princeton NJ 08544, USA
[2] Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany lncs@springer.com
http://www.springer.com/gp/computer-science/lncs
[3] ABC Institute, Rupert-Karls-University Heidelberg, Heidelberg, Germany
{abc,lncs}@uni-heidelberg.de

Abstract. This paper proved a method to computing the forgetting in CTL which has been submitted to IJCAI, from the resolution proposed by Zhang at all by extending the resolution rules.

Keywords: Forgetting · CTL · Model checking.

1 Introduction

As a logical notion, *forgetting* was first formally defined in propostional and first order logics by Lin and Reiter [12]. Over the last twenty years, researchers have developed forgetting notions and theories not only in classical logic but also in other non-classical logic systems [**?**], such as forgetting in logic programs under answer set/stable model semantics [22,5,19,17,16], forgetting in description logic [18,14,25] and knowledge forgetting in modal logic [24,15,13,7]. In application, forgetting has been used in planning [11], conflict solving [9,23], createing restricted views of ontologies [25], strongest and weakest definitions [8], SNC (WSC) [10] and so on.

Though forgetting has been extensively investigated from various aspects of different logical systems. However, the existing forgetting method in propositional logic, answer set programming, description logic and modal logic are not directly applicable in CTL. Similar with that in [24], we research forgetting in CTL from the semantic forgetting point of view. And it is shown that our definition of forgetting satisfies those four postulates of forgetting.

2 Preliminaries

We start with some technical and notational preliminaries. Throughout this paper, we fix a finite set $\mathcal{A}$ of propositional variables (or atoms), and use V, V' for subsets of $\mathcal{A}$. In the following several parts, we will introduce the structure we use for CTL, syntactic and semantic of CTL and the normal form $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ (Separated Normal Form with Global Clauses for CTL) of CTL [21].

* Supported by organization x.

2.1 Model structure in CTL

In general, a transition system [4] is described as a *model structure* (or *Kripke structure*)(in this article, we treat transition system and model structure as the same thing), and a model structure is a triple $\mathcal{M} = (S, R, L)$ [6], where

- S is a set of states,
- $R \subseteq S \times S$ is a total binary relation over S, *i.e.* , for each state $s \in S$ there is a state $s' \in S$ such that $(s, s') \in R$, and
- L is an interpretation function $S \to 2^{\mathcal{A}}$ mapping every state to the set of atoms true at that state.

In this article, the same as [3], all of our results apply only to finite Kripke structures. Besides, we restrict ourselves to model structure $\mathcal{M} = (S, R, L, s_0)$ (similar with that in [21]) such that

- there exists a state s_0, called the *initial state*, such that for every state $s \in S$ there is a path π_{s_0} s.t. $s \in \pi_{s_0}$.

We call a model structure $\mathcal{M}$ on a set V of atoms if $L : S \to 2^V$, *i.e.* , the labeling function L map every state to V (not the $\mathcal{A}$). A *path* π_{s_i} start from s_i of $\mathcal{M}$ is a infinite sequence of states $\pi_{s_i} = (s_i, s_{i+1}s_{i+2}, \dots)$, where for each j $(i \leq j)$, $(s_j, s_{j+1}) \in R$. By $s' \in \pi_{s_i}$ we mean that s' is a state in the path π_{s_i}.

For a given model structure (S, R, L, s_0) and $s \in S$, the *computation tree* $\text{Tr}_n^{\mathcal{M}}(s)$ of $\mathcal{M}$(or simply $\text{Tr}_n(s)$), that has depth n and is rooted at s, is recursively defined as [3], for $n \geq 0$,

- $\text{Tr}_0(s)$ consists of a single node s with label s.
- $\text{Tr}_{n+1}(s)$ has as its root a node m with label s, and if $(s, s') \in R$ then the node m has a subtree $\text{Tr}_n(s')$[5].

By s_n we mean the node at the nth level in tree $\text{Tr}_m(s)$ $(m \geq n)$.

A K-*structure* (or K-*interpretation*) is a model structure $\mathcal{M} = (S, R, L, s_0)$ associating with a state $s \in S$, which is written as $(\mathcal{M}, s)$ for convenience in the following. In the case s is an initial state of $\mathcal{M}$, the K-structure is *initial*.

2.2 Syntactic and semantic of CTL

In the following we briefly review the basic syntax and semantics of the *Computation Tree Logic* (CTL in short) [4]. The *signature* of $\mathcal{L}$ includes:

- a finite set of Boolean variables, called *atoms* of $\mathcal{L}$: $\mathcal{A}$;
- the classical connectives: $\bot, \vee$ and $\neg$;

[4] According to [1], a *transition system* TS is a tuple $(S, Act, \to, I, AP, L)$ where (1) S is a set of states, (2) Act is a set of actions, (3) $\to \subseteq S \times \text{Act} \times S$ is a transition relation, (4) $I \subseteq S$ is a set of initial states, (5) AP is a set of atomic propositions, and (6) $L : S \to 2^{\text{AP}}$ is a labeling function.

[5] Though some nodes of the tree may have the same label, they are different nodes in the tree.

– the path quantifiers: A and E;
– the temporal operators: X, F, G U and W, that means 'neXt state', 'some Future state', 'all future states (Globally)', 'Until' and 'Unless', respectively;
– parentheses: (and).

The *(existential normal form or ENF in short) formulas* of $\mathcal{L}$ are inductively defined via a Backus Naur form:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathrm{EX}\phi \mid \mathrm{EG}\phi \mid \mathrm{E}[\phi\ \mathrm{U}\ \phi] \tag{1}$$

where $p \in \mathcal{A}$. The formulas $\phi \wedge \psi$ and $\phi \rightarrow \psi$ are defined in a standard manner of propositional logic. The other form formulas of $\mathcal{L}$ are abbreviated using the forms of (1). Notice that, according to the above definition for formulas of CTL, each of the CTL *temporal connectives* has the form XY where $X \in \{\mathrm{A}, \mathrm{E}\}$ and $Y \in \{\mathrm{X}, \mathrm{F}, \mathrm{G}, \mathrm{U}, \mathrm{W}\}$. The priorities for the CTL connectives are assumed to be (from the highest to the lowest):

$$\neg, \mathrm{EX}, \mathrm{EF}, \mathrm{EG}, \mathrm{AX}, \mathrm{AF}, \mathrm{AG} \prec \wedge \prec \vee \prec \mathrm{EU}, \mathrm{AU}, \mathrm{EW}, \mathrm{AW}, \rightarrow .$$

We are now in the position to define the semantics of $\mathcal{L}$. Let $\mathcal{M} = (S, R, L, s_0)$ be an model structure, $s \in S$ and ϕ a formula of $\mathcal{L}$. The *satisfiability* relationship between $\mathcal{M}$, s and ϕ, written $(\mathcal{M}, s) \models \phi$, is inductively defined on the structure of ϕ as follows:

– $(\mathcal{M}, s) \not\models \bot$;
– $(\mathcal{M}, s) \models p$ iff $p \in L(s)$;
– $(\mathcal{M}, s) \models \phi_1 \vee \phi_2$ iff $(\mathcal{M}, s) \models \phi_1$ or $(\mathcal{M}, s) \models \phi_2$;
– $(\mathcal{M}, s) \models \neg\phi$ iff $(\mathcal{M}, s) \not\models \phi$;
– $(\mathcal{M}, s) \models \mathrm{EX}\phi$ iff $(\mathcal{M}, s_1) \models \phi$ for some $s_1 \in S$ and $(s, s_1) \in R$;
– $(\mathcal{M}, s) \models \mathrm{EG}\phi$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \ldots)$ such that $(\mathcal{M}, s_i) \models \phi$ for each $i \geq 1$;
– $(\mathcal{M}, s) \models \mathrm{E}[\phi_1 \mathrm{U} \phi_2]$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \ldots)$ such that, for some $i \geq 1$, $(\mathcal{M}, s_i) \models \phi_2$ and $(\mathcal{M}, s_j) \models \phi_1$ for each $j < i$.

Similar to the work in [3,2], only initial K-structures are considered to be candidate models in the following, unless explicitly stated. Formally, an initial K-structure $\mathcal{K}$ is a *model* of a formula ϕ whenever $\mathcal{K} \models \phi$. Let Π be a set of formulae, $\mathcal{K} \models \Pi$ if for each $\phi \in \Pi$ there is $\mathcal{K} \models \phi$. We denote *Mod*(ϕ) (*Mod*(Π)) the set of models of ϕ (Π). The formula ϕ (set Π of formulae) is *satisfiable* if *Mod*$(\phi) \neq \emptyset$ (*Mod*$(\Pi) \neq \emptyset$). Since both the underlying states in model structure and signatures are finite, *Mod*(ϕ) (*Mod*(Π)) is finite for any formula ϕ (set Π of formulae).

Let ϕ_1 and ϕ_2 be two formulas or set of formulas. By $\phi_1 \models \phi_2$ we denote *Mod*$(\phi_1) \subseteq$ *Mod*(ϕ_2). By $\phi_1 \equiv \phi_2$ we mean $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$. In this case ϕ_1 is *equivalent* to ϕ_2.

Let ϕ be a formula or set of formulas. By *Var*(ϕ) we mean the set of atoms occurring in ϕ. Let $V \subseteq \mathcal{A}$. The formula ϕ is V*-irrelevant*, written $\mathrm{IR}(\phi, V)$, if there is a formula ψ with *Var*$(\psi) \cap V = \emptyset$ such that $\phi \equiv \psi$.

2.3 The normal form of CTL

It has proved that any CTL formula φ can be transformed into a set T_φ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ (Separated Normal Form with Global Clauses for CTL) clauses in polynomial time such that φ is satisfiable iff T_φ is satisfiable [20]. An important difference between CTL formulae and $\mathrm{SNF}^g_{\mathrm{CTL}}$ is that $\mathrm{SNF}^g_{\mathrm{CTL}}$ is an extension of the syntax of CTL to use indices. These indices can be used to preserve a particular path context. The language of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses is defined over an extension of CTL. That is the language is based on: (1) the language of CTL; (2) a propositional constant **start**; (3) a countably infinite index set Ind; and (4) temporal operators: $\mathrm{E}_{\langle ind\rangle}\mathrm{X}$, $\mathrm{E}_{\langle ind\rangle}\mathrm{F}$, $\mathrm{E}_{\langle ind\rangle}\mathrm{G}$,$\mathrm{E}_{\langle ind\rangle}\mathrm{U}$ and $\mathrm{E}_{\langle ind\rangle}\mathrm{W}$.

The priorities for the $\mathrm{SNF}^g_{\mathrm{CTL}}$ connectives are assumed to be (from the highest to the lowest):

$$\begin{gathered}\neg, (\mathrm{EX}, \mathrm{E}_{\langle ind\rangle}\mathrm{X}), (\mathrm{EF}, \mathrm{E}_{\langle ind\rangle}\mathrm{F}), (\mathrm{EG}, \mathrm{E}_{\langle ind\rangle}\mathrm{G}), \mathrm{AX}, \mathrm{AF}, \mathrm{AG}\\ \prec \wedge \prec \vee \prec (\mathrm{EU}, \mathrm{E}_{\langle ind\rangle}\mathrm{U}), \mathrm{AU}, (\mathrm{EW}, , \mathrm{E}_{\langle ind\rangle}\mathrm{W}), \mathrm{AW}, \rightarrow .\end{gathered}$$

Where the operators in the same brackets have the same priority.

Before talked about the sematic of this language, we introduce the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses at first. The $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses consists of formulae of the following forms.

$$\begin{array}{ll}
\mathrm{AG}(\textbf{start} \supset \bigvee_{j=1}^{k} m_j) & (initial\ clause)\\
\mathrm{AG}(true \supset \bigvee_{j=1}^{k} m_j) & (global\ clause)\\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{AX}\bigvee_{j=1}^{k} m_j) & (\mathrm{A}-step\ clause)\\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\bigvee_{j=1}^{k} m_j) & (\mathrm{E}-step\ clause)\\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{AF}l) & (\mathrm{A}-sometime\ clause)\\
\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{E}_{\langle ind\rangle}\mathrm{F}l) & (\mathrm{E}-sometime\ clause).
\end{array}$$

where $k \geq 0$, $n > 0$, **start** is a propositional constant, l_i $(1 \leq i \leq n)$, m_j $(1 \leq j \leq k)$ and l are literals, that is atomic propositions or their negation and ind is an element of Ind (Ind is a countably infinite index set). By clause we mean the classical clause or the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause unless explicitly stated.

Formulae of $\mathrm{SNF}^g_{\mathrm{CTL}}$ over $\mathcal{A}$ are interpreted in Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$, where S, R, L and s_0 is the same as our model structure talked in 2.1 and $[\_]$: Ind $\rightarrow 2^{(S*S)}$ maps every index $ind \in$ Ind to a successor function $[ind]$ which is a functional relation on S and a subset of the binary accessibility relation R, such that for every

$s \in S$ there exists exactly a state $s' \in S$ such that $(s, s') \in [ind]$ and $(s, s') \in R$. An infinite path $\pi_{s_i}^{\langle ind \rangle}$ is an infinite sequence of states $s_i, s_{i+1}, s_{i+2}, \ldots$ such that for every $j \geq i$, $(s_j, s_{j+1}) \in [ind]$.

Similarly, an *Ind-structure* (or *Ind-interpretation*) is a Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$ associating with a state $s \in S$, which is written as $(\mathcal{M}, s)$ for convenience in the following. In the case s is an initial state of $\mathcal{M}$, the Ind-structure is *initial*.

The semantics of $\mathrm{SNF}^g_{\mathrm{CTL}}$ is an extension of the semantics of CTL defined in Section 2.2 except using the Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$ replace model structure, $(\mathcal{M}, s_i) \models$ **start** iff $s_i = s_0$ and for all $\mathrm{E}_{\langle ind \rangle} \Gamma$ are explained in the path $\pi_{s_i}^{\langle ind \rangle}$, where $\Gamma \in \{\mathrm{X}, \mathrm{G}, \mathrm{U}, \mathrm{W}\}$. The semantics of $\mathrm{SNF}^g_{\mathrm{CTL}}$ is then defined as shown next as an extension of the semantics of CTL defined in Section 2.2. Let φ and ψ be two $\mathrm{SNF}^g_{\mathrm{CTL}}$ formulae and $\mathcal{M} = (S, R, L, [\_], s_0)$ be an Ind-model structure, the relation "$\models$" between $\mathrm{SNF}^g_{\mathrm{CTL}}$ formulae and $\mathcal{M}$ is defined recursively as follows:

- $(\mathcal{M}, s_i) \models$ **start** iff $s_i = s_0$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{X}\psi$ iff for the path $\pi_{s_i}^{\langle ind \rangle}$, $(\mathcal{M}, s_{i+1}) \models \psi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\psi$ iff for every $s_j \in \pi_{s_i}^{\langle ind \rangle}$, $(\mathcal{M}, s_j) \models \psi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{U} \psi]$ iff there exists $s_j \in \pi_{s_i}^{\langle ind \rangle}$ such that $(\mathcal{M}, s_j) \models \psi$ and for every $s_k \in \pi_{s_i}^{\langle ind \rangle}$, if $i \leq k < j$, then $(\mathcal{M}, s_k) \models \varphi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{F}\psi$ iff $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\top \mathrm{U} \psi]$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{W} \psi]$ iff $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\varphi$ or $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{U} \psi]$.

The semantics of the remaining operators is analogous to that given previously but in the extended Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$. A $\mathrm{SNF}^g_{\mathrm{CTL}}$ formula φ is satisfiable, iff for some Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$, $(\mathcal{M}, s_0) \models \varphi$, and unsatisfiable otherwise. And if $(\mathcal{M}, s_0) \models \varphi$ then $(\mathcal{M}, s_0)$ is called a Ind-model of φ, and we say that $(\mathcal{M}, s_0)$ satisfies φ. By $T \wedge \varphi$ we mean $\bigwedge_{\psi \in T} \psi \wedge \varphi$, where T is a set of formulae. Other terminologies are similar with those in section 2.2.

3 Problem Definition

In order to define our problem, *i.e.* forgetting in CTL, we review our definition of V-bisimulation (read ?? for more detials).

Definition 1. *Let $V \subseteq \mathcal{A}$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2)$ be* K*-structures (Ind-structures). Then $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}$ if and only if*

(i) $L_1(s_1) - V = L_2(s_2) - V$,
(ii) for every $(s_1, s_1') \in R_1$, there is $(s_2, s_2') \in R_2$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}$, and
(iii) for every $(s_2, s_2') \in R_2$, there is $(s_1, s_1') \in R_1$

where $\mathcal{K}_i' = (\mathcal{M}_i, s_i')$ with $i \in \{1, 2\}$.

Proposition 1. *Let $i \in \{1, 2\}$, $V_1, V_2 \subseteq \mathcal{A}$, s_i's be two states and π_i's be two pathes, and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2, 3)$ be* K*-structures (Ind-structures) such that $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$ and $\mathcal{K}_2 \leftrightarrow_{V_2} \mathcal{K}_3$. Then:*

(i) $s_1' \leftrightarrow_{V_i} s_2'$ $(i = 1, 2)$ *implies* $s_1' \leftrightarrow_{V_1 \cup V_2} s_2'$*;*
(ii) $\pi_1' \leftrightarrow_{V_i} \pi_2'$ $(i = 1, 2)$ *implies* $\pi_1' \leftrightarrow_{V_1 \cup V_2} \pi_2'$*;*
(iii) for each path π_{s_1} *of* $\mathcal{M}_1$ *there is a path* π_{s_2} *of* $\mathcal{M}_2$ *such that* $\pi_{s_1} \leftrightarrow_{V_1} \pi_{s_2}$*, and vice versa;*
(iv) $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$*;*
(v) If $V_1 \subseteq V_2$ *then* $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$*.*

Now we give the formal definition of forgetting in CTL from the semantic forgetting point view.

Definition 2 (Forgetting). *Let* $V \subseteq \mathcal{A}$ *and* ϕ *a CTL formula. A CTL formula* ψ *with* $Var(\psi) \cap V = \emptyset$ *is a* result of forgetting V from ϕ*, if*

$$Mod(\psi) = \{\mathcal{K} \text{ is initial} \mid \exists \mathcal{K}' \in Mod(\phi) \,\&\, \mathcal{K}' \leftrightarrow_V \mathcal{K}\}. \tag{2}$$

Where $\mathcal{K}$ *and* $\mathcal{K}'$ *are* K*-structures.*

Note that if both ψ and ψ' are results of forgetting V from ϕ then $Mod(\psi) = Mod(\psi')$, *i.e.* , ψ and ψ' have the same models. In the sense of equivalence the forgetting result is unique (up to equivalence).

Similar with the V-bisimulation between K-structures, we define the $\langle V, I\rangle$-bisimulation between Ind-structures as follows:

Definition 3. *(*$\langle V, I\rangle$*-bisimulation) Let* $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ *with* $i \in \{1, 2\}$ *be two Ind-structures,* V *be a set of atoms and* $I \subseteq Ind$*. The* $\langle V, I\rangle$*-bisimulation* $\beta_{\langle V,I\rangle}$ *between initial Ind-structures is a set that satisfy* $((\mathcal{M}_1, s_0^1), (\mathcal{M}_2, s_0^2)) \in \beta_{\langle V,I\rangle}$ *if and only if* $(\mathcal{M}_1, s_0^1) \leftrightarrow_V (\mathcal{M}_2, s_0^2)$ *and* $\forall j \notin I$ *there is*

(i) $\forall (s, s_1) \in [j]_1$ *there is* $(s', s_1') \in [j]_2$ *such that* $s \leftrightarrow_V s'$ *and* $s_1 \leftrightarrow_V s_1'$*, and*
(ii) $\forall (s', s_1') \in [j]_2$ *there is* $(s, s_1) \in [j]_1$ *such that* $s \leftrightarrow_V s'$ *and* $s_1 \leftrightarrow_V s_1'$*.*

Apparently, this definition is similar with our concept V-bisimulation except that this $\langle V, I\rangle$-bisimulation has introduced the index.

Proposition 2. *Let* $i \in \{1, 2\}$*,* $V_1, V_2 \subseteq \mathcal{A}$*,* $I_1, I_2 \subseteq Ind$ *and* $\mathcal{K}_i = (\mathcal{M}_i, s_0^i)$ $(i = 1, 2, 3)$ *be Ind-structures such that* $\mathcal{K}_1 \leftrightarrow_{\langle V_1, I_1\rangle} \mathcal{K}_2$ *and* $\mathcal{K}_2 \leftrightarrow_{\langle V_2, I_2\rangle} \mathcal{K}_3$*. Then:*

(i) $\mathcal{K}_1 \leftrightarrow_{\langle V_1 \cup V_2, I_1 \cup I_2\rangle} \mathcal{K}_3$*;*
(ii) If $V_1 \subseteq V_2$ *and* $I_1 \subseteq I_2$ *then* $\mathcal{K}_1 \leftrightarrow_{\langle V_2, I_2\rangle} \mathcal{K}_2$*.*

Proof. (i) By Proposition 1 we have $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$. For (i) of Definition 3 we can prove it as follows: $\forall (s, s_1) \in [j]_1$ there is a $(s', s_1') \in [j]_2$ such that $s \leftrightarrow_{V_1} s'$ and $s_1 \leftrightarrow_{V_1} s_1'$ and there is a $(s'', s_1'') \in [j]_3$ such that $s' \leftrightarrow_{V_2} s''$ and $s_1' \leftrightarrow_{V_2} s_1''$, and then we have $\forall (s, s_1) \in [j]_1$ there is a $(s'', s_1'') \in [j]_3$ such that $s \leftrightarrow_{V_1 \cup V_2} s''$ and $s_1 \leftrightarrow_{V_1 \cup V_2} s_1''$. The (ii) of Definition 3 can be proved similarly.

(ii) This can be proved from (i).

4 The Calculus

Resolution in CTL is a method to decide the satisfiability of a CTL formula. In this paper, we will explore a resolution-based method to compute forgetting in CTL. In this part we use the transformation rules Trans(1) to Trans(12) and resolution rules (SRES1), ..., (SRES8), RW1, RW2, (ERES1), (ERES2) in [21].

The key problems of this method include (1) How to fill the gap between CTL and $\mathrm{SNF}^g_{\mathrm{CTL}}$; and (2) How to eliminate the irrelevant atoms in the formula. We will resolve these two problems by $\langle V, I\rangle$-bisimulation and *substitution* operator. For convenient, we use $V \subseteq \mathcal{A}$ denote the set we want to forget, $V' \subseteq \mathcal{A}$ with $V \cap V' = \emptyset$ the set of atoms (I be the set of index) introduced in the transformation process, φ the CTL formula, T_φ be the set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause obtained from φ by using transformation rules and $\mathcal{M} = (S, R, L, [\_], s_0)$ unless explicitly stated. Let T, T' be two set of formulae, I a set of indexes and $V'' \subseteq \mathcal{A}$, by $T \equiv_{\langle V'', I\rangle} T'$ we mean that $\forall(\mathcal{M}, s_0) \in Mod(T)$ there is a $(\mathcal{M}', s_0')$ such that $(\mathcal{M}, s_0) \leftrightarrow_{\langle V'', I\rangle} (\mathcal{M}', s_0')$ and $(\mathcal{M}', s_0') \models T'$ and vice versa.

Proposition 3. *Let P, P_i and φ_i be CTL formulas, then*

(i) $P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_1 \wedge \cdots \wedge P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_n \equiv_{\langle \emptyset, \{ind\}\rangle} P \supset \mathrm{EX} \bigwedge_{i\in\{0,\ldots,n\}} \varphi_i$,

(ii) $P_1 \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_1 \wedge \cdots \wedge P_n \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_n \in T \equiv_{\langle \emptyset, \{ind\}\rangle} \bigwedge_{e\in 2^{\{0,\ldots,n\}}\setminus\{\emptyset\}}(\bigwedge_{i\in e} P_i \supset \mathrm{EX}(\bigwedge_{i\in e}\varphi_i))$,

(iii) $P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{F}\varphi_1 \wedge \cdots \wedge P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{F}\varphi_n \in T \equiv_{\langle \emptyset, \{ind\}\rangle} P \supset \bigvee \mathrm{EF}(\varphi_{j_1} \wedge \mathrm{EF}(\varphi_{j_2} \wedge \mathrm{EF}(\cdots \wedge \mathrm{EF}\varphi_{j_n})))$, *where* $(j_1, \ldots, j_n)$ *are sequences of all elements in* $\{0, \ldots, n\}$,

(iv) $P \supset (C \vee \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_1) \wedge P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_2 \equiv_{\langle \emptyset, \{ind\}\rangle} P \supset ((C \wedge \mathrm{EX}\varphi_2) \vee \mathrm{EX}(\varphi_1 \wedge \varphi_2))$

Proof. It is easy to check.

The algorithm of computing the forgetting in CTL is as Algorithm 1.

Algorithm 1: Computing forgetting - A resolution-based method

Input: A CTL formula φ and a set V of atoms
Output: $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$
$T = \emptyset$ // the initial set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses of φ ;
$T' = \emptyset$ // the set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses without index;
$V' = \emptyset$ // the set of atoms introduced in the process of transforming φ into $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses;
$T, V' \leftarrow Tran(\varphi, V)$;
$Res \leftarrow Res(T, V')$;
$\mathit{\Gamma} = \mathrm{Sub}(Res, V')$;
$\mathit{\Omega} \leftarrow Elm(\mathrm{EF}(\mathit{\Gamma}), \mathit{\Gamma})$;
$\mathit{\Gamma}_1 \leftarrow \mathrm{NI}(\mathit{\Omega})$;
return $\bigwedge_{\psi\in R(\mathit{\Gamma}_1)_{\mathrm{CTL}}} \psi$.

Algorithm 2: $Tran(\varphi, V)$

Input: A CTL formula φ and a set V of atoms
Output: A set T of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses and a set V' of atoms

$T = \emptyset$ // the initial set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses of φ ;
$OldT = \{$**start** $\supset z, z \supset \varphi\}$;
$V' = \{z\}$;
while $OldT \neq T$ **do**
    $OldT = T$;
    $R = \emptyset$;
    $X = \emptyset$;
    if *Chose a formula* $\psi \in OldT$ *that dose not a* $\mathrm{SNF}^g_{\mathrm{CTL}}$ *clause* **then**
        Using a match rule Rl to transform ψ into a set R of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses;
        X is the set of atoms introduced by using Rl;
        $V' = V' \cup X$;
        $T = OldT \setminus \{\psi\} \cup R$;
    end
end

Algorithm 3: $Res(T, V')$

Input: A set T of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses and a set V' of atoms
Output: A set Res of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses

$S = \{C | C \in T$ and $Var(C) \cap V = \emptyset\}$;
$\Pi = T \setminus S$;
for $(p \in V \cup V')$ **do**
    $\Pi' = \{C \in \Pi | p \in Var(C)\}$;
    $\Sigma = \Pi \setminus \Pi'$;
    for *($C \in \Pi'$ s.t. p appearing in C positively)* **do**
        for *($C' \in \Pi'$ s.t. p appearing in C' negatively and C, C' are resolvable)* **do**
            $\Sigma = \Sigma \cup \{res(C, C')\}$;
            $\Pi' = \Pi' \cup \{C'' = res(C, C') | p \in Var(C'')\}$;
        end
    end
    $\Pi = \Sigma$;
end
$Res = \Pi \cup S$;

References

1. Baier, C., Katoen, J.: Principles of Model Checking. The MIT Press (2008)
2. Bolotov, A.: A clausal resolution method for ctl branching-time temporal logic. Journal of Experimental & Theoretical Artificial Intelligence **11**(1), 77–93 (1999). https://doi.org/10.1080/095281399146625

3. Browne, M.C., Clarke, E.M., Grumberg, O.: Characterizing finite kripke structures in propositional temporal logic. Theor. Comput. Sci. **59**, 115–131 (1988). https://doi.org/10.1016/0304-3975(88)90098-9, https://doi.org/10.1016/0304-3975(88)90098-9
4. Clarke, E.M., Emerson, E.A., Sistla, A.P.: Automatic verification of finite-state concurrent systems using temporal logic specifications. ACM Trans. Program. Lang. Syst. **8**(2), 244–263 (1986). https://doi.org/10.1145/5397.5399, https://doi.org/10.1145/5397.5399
5. Eiter, T., Wang, K.: Semantic forgetting in answer set programming. Elsevier Science Publishers Ltd. (2008)
6. Emerson, E.A.: Temporal and modal logic. In: Formal Models and Semantics, pp. 995–1072. Elsevier (1990)
7. Fang, L., Liu, Y., Van Ditmarsch, H.: Forgetting in multi-agent modal logics. Artificial Intelligence **266**, 51–80 (2019)
8. Lang, J., Marquis, P.: On propositional definability. Artificial Intelligence **172**(8), 991–1017 (2008)
9. Lang, J., Marquis, P.: Reasoning under inconsistency: a forgetting-based approach. Elsevier Science Publishers Ltd (2010)
10. Lin, F.: On strongest necessary and weakest sufficient conditions. Artif. Intell. **128**(1-2), 143–159 (2001). https://doi.org/10.1016/S0004-3702(01)00070-4, https://doi.org/10.1016/S0004-3702(01)00070-4
11. Lin, F.: Compiling causal theories to successor state axioms and strips-like systems. Journal of Artificial Intelligence Research **19**, 279–314 (2003)
12. Lin, F., Reiter, R.: Forget it. In: Working Notes of AAAI Fall Symposium on Relevance. pp. 154–159 (1994)
13. Liu, Y., Wen, X.: On the progression of knowledge in the situation calculus. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 976–982. IJCAI/AAAI, Barcelona, Catalonia, Spain (2011)
14. Lutz, C., Wolter, F.: Foundations for uniform interpolation and forgetting in expressive description logics. In: IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence. pp. 989–995. IJCAI/AAAI, Barcelona, Catalonia, Spain (2011)
15. Su, K., Sattar, A., Lv, G., Zhang, Y.: Variable forgetting in reasoning about knowledge. Journal of Artificial Intelligence Research **35**, 677–716 (2009)
16. Wang, Y., Wang, K., Zhang, M.: Forgetting for answer set programs revisited. In: IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence. pp. 1162–1168. IJCAI/AAAI, Beijing, China (2013)
17. Wang, Y., Zhang, Y., Zhou, Y., Zhang, M.: Forgetting in logic programs under strong equivalence. In: Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference. pp. 643–647. AAAI Press, Rome, Italy (2012)
18. Wang, Z., Wang, K., Topor, R.W., Pan, J.Z.: Forgetting for knowledge bases in DL-Lite. Annuals of Mathematics and Artificial Intelligence **58**(1-2), 117–151 (2010)
19. Wong, K.S.: Forgetting in Logic Programs. Ph.D. thesis, The University of New South Wales (2009)
20. Zhang, L., Hustadt, U., Dixon, C.: First-order resolution for ctl. Tech. rep., Citeseer (2008)
21. Zhang, L., Hustadt, U., Dixon, C.: A refined resolution calculus for ctl. In: International Conference on Automated Deduction. pp. 245–260. Springer (2009)
22. Zhang, Y., Foo, N.Y.: Solving logic program conflict through strong and weak forgettings. Artificial Intelligence **170**(8-9), 739–778 (2006)
23. Zhang, Y., Foo, N.Y., Wang, K.: Solving logic program conflict through strong and weak forgettings. In: Ijcai-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, Uk, July 30-August. pp. 627–634 (2005)

24. Zhang, Y., Zhou, Y.: Knowledge forgetting: Properties and applications. Artificial Intelligence **173**(16-17), 1525–1537 (2009)
25. Zhao, Y., Schmidt, R.A.: Role forgetting for alcoqh (δ)-ontologies using an ackermann-based approach. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. pp. 1354–1361. AAAI Press (2017)

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017