On forgetting in CTL with Finite States: a preliminary report

Renyan Feng[1*] , **Yisong Wang**[2] , **Stefan Schlobach**[3] and **Erman Acar**[4]
[1,2]GuiZhou University
[3,4]Vrije Universiteit Amsterdam
fengrenyan@gmail.com, yswang@gzu.edu.cn, {k.s.schlobach, erman.acar}@vu.nl

Abstract

Computation Tree Logic (CTL) is one of the main logical formalisms for program specification, which prescribes what a transition system has to do and what not, and verification. How to extract relevant specification from existing specification is a key problem when facing a big database of specification. Forgetting, as a technology distilling from a knowledge base only the part that is relevant to a subset of alphabet, just happens to solve the problem talked above. In this paper, we study forgetting in CTL from the sematic forgetting point of view by bisimulation to distill from a set of specifications only the part that is relevant to a subset of the alphabet. And we show that the CTL system is closed under forgetting. Besides, from the point of view of model and resolution, this paper discusses the algorithms of computing forgetting on CTL.

1 Introduction

Temporal logic has found numerous applications in computer science, ranging from the traditional and well-developed fields of program specification and verification [Manna and Pnueli, 1995; Pnueli and Manna, 1992; Pnueli, 1986], temporal databases [Chomicki, 1994; Chomicki and Niwinski, 1995; Abiteboul *et al.*, 1996; Sernadas, 1980], and distributed and multi-agent systems [Fagin *et al.*, 2004], to more recent uses in knowledge representation and reasoning [Schild, 1993; Baader and Ohlbach, 1995; Hodkinson *et al.*, 2000]. In temporal logic model checking, a kind of verification, temporal logic is mainly to use describe the properties of the system. Two important families of temporal logics have been considered: linear-time temporal logics (e.g LTL[Pnueli, 1977]) can be used to express properties of one single execution of the system under study, while branching-time temporal logics(e.g CTL[Clarke and Emerson, 1981] and CTL* [Emerson and Halpern, 1986])consider the execution tree [Laroussinie and Markey, 2014]. In this paper, we are interested to branching-time temporal logic-CTL (Computation Tree temporal Logic).

[*]Contact Author

A *specification* often refers to a set of documented requirements to be satisfied by a material, design, product, or service ?1?. In model checking, specification is mainly used to specify the properties that should be satisfied by system [Chaki *et al.*, 2005; Chaki *et al.*, 2004; Clarke *et al.*, 1994; Ghosh *et al.*, 2016]. The practical effectiveness of model checking is characterized by a trade-off between the expressive power of the specification formalism and the complexity of the corresponding model checking algorithm [Chaki *et al.*, 2004]. For software verification, this problem is even more acute, since software is harder to specify, and state explosion is exacerbated by the concurrent execution of multiple components. In order to solve the problem of states exploring, in the last few decades, researchers have put forward Modular Verification, rather than consider the whole states space [Chaki *et al.*, 2005; Chaki *et al.*, 2004]. However, the elements in a module is almost different with the other modules. If we can extract those knowledge related to the elements in this module, then it can be more efficient to verification. In addition, as the system has been updated, some of the elements previously considered are no longer considered, if we have had those specifications, we do not need to reproduce the specifications that different with the original one only in that the latter using lesser atoms. Besides, with the size of the system growing, not only has the number of available (proposition) increased considerably, but they are often large in size and are becoming more complex to manage. This leads to the specification being difficult to maintain and modify, and costly to reuse for later processing, where only a specific part of an specification is of interest. All of this working directly on the whole of the original specification and building a new sub-specification are unadvisable. Therefore, a strong demand for techniques and automated tools for obtaining the specific sub-specification.

Forgetting, as a technology distilling from a knowledge base only the part that is relevant to a subset of alphabet, just happens to solve the problem talked above. For example, the is a theory $T = \{a \vee \neg b, c \wedge b\}$ in Classical Proposition Logic (CPL), a theory that the $\{a, c\}$-alphabet related part of T is $\{a \vee c\}$. Although there have been a number of studies on similar methods of forgetting, such as variable elimination, irrelevance and independence[Bobrow *et al.*, 1997], forgetting has been put forward by Lin at first [Lin and Reiter, 1994]. 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 [Zhang and Foo, 2006; Eiter and Wang, 2008; Wong, 2009; Wang *et al.*, 2012; Wang *et al.*, 2013], forgetting in description logic [Wang *et al.*, 2010; Lutz and Wolter, 2011; Zhao and Schmidt, 2017] and knowledge forgetting in modal logic [Zhang and Zhou, 2009a; Su *et al.*, 2009; Liu and Wen, 2011; Fang *et al.*, 2019]. In application, forgetting has been used in planning [Lin, 2003], conflict solving [Lang and Marquis, 2010; Zhang *et al.*, 2005], knowledge compilation [Zhang and Zhou, 2009a; Bienvenu *et al.*, 2010], createing restricted views of ontologies [Zhao and Schmidt, 2018], strongest and weakest definitions [Lang and Marquis, 2008], strongest necessary and weakest sufficient conditions [Lin, 2001] and so on. Besides, (semantic) forgetting is studied by researchers in [Marquis, 2003; Eiter and Wang, 2008]. To our best knowledge, though forgetting has been extensively investigated from various aspects of different logical systems, in standard propositional logic, a general algorithm of forgetting and its computation-oriented investigation in CTL are still lacking. However, as we said above that this skill is essential in CTL. A simple example in the follows show the necessity of forgetting in CTL:

- *Let* $\varphi = \mathrm{A}((p \wedge q)\mathrm{U}(f \vee m)) \wedge r$, *then what is the* $\{q, f, m, r\}$*-alphabet related part of* φ*?*

Apparent that this is not easy to obtain even for this simple φ. More critical is existing methods cannot solve the problem. In this paper we will introduce a skill that can solve this problem.

It is know that forgetting can be defined in two closely related ways: it can be defined on the syntactic level as the dual of uniform interpolation [Konev *et al.*, 2009] and it can be defined model-theoretically as semantic forgetting [Zhao and Schmidt, 2017]. Besides, forgetting, is regarded as an abstract belief change operator, independent of the underlying logic, is reached by James P. [Delgrande, 2017]. In this paper, we explore the forgetting in CTL from a semantic forgetting point of view. Bisimulation, a concept weaker than homogeneous and stronger than homomorphic, make it possible to relate systems described at different levels of abstraction and to verify, for example, the correctness of an implementation with respect to a more abstract specification of a given system [Clarke *et al.*, 2018]. Bisimulation was first proposed by Benthem [Benthem, 1977], and then have been used in analyzing the expressive power of modal logics and in Model checking [Baier and Katoen, 2008]. The difference between our concept and these concepts include two aspects. On one hand, we have introduced the set V of atoms, which dose not mentioned in transition system. On the other hand, our V-bisimulation is on the set of Kripke structures, not on the set of states as before. Through this generalization, we were able to compare the differences among the Kripke structures on the set V, which allowed us to explore semantically the meaning of forgetting the atomic set V in the CTL.

The forgetting is close in the classical propositional logic (CPL), that is the result of forgetting a set V of atoms from the CPL formula is also a CPL formula. However, there are some logic systems, such as first-order logic and descrp logic, the forgetting is not close. Fortunately, our concept of forgetting is closed in the CTL and in order to prove this result, we introduced the characteristic formula. As its name suggests, the characteristic formula is a formula that describes a Kripke structure. I It describes the structure that the Kripke structure has and the meaning it conveys. The characteristic formula in CTL was originally proposed by [Browne *et al.*, 1988]. In this paper, it is modified on the original basis and combined with our v-bisimulation semantics. We don't care about the characteristic formula of the Kripke structure in the whole proposition space. We only consider its characteristic formula on a specified set.

Our Contributions: Focusing on forgetting in CTL, this article has the following contributions:

- Proposed a new set-based bisimulation for transition system;
- Given the definition of forgetting in CTL;
- Defined the characteristic formula of a Kripke structure on some set V of atoms;
- Proved that the forgetting in CTL is closed;
- Two algorithms have been proposed for computing forgetting in CTL.

The rest of the paper is organised as follows. Section 2 introduces the related notions for forgetting in CTL, including the syntax and semantics of CTL, the language we aimed for, and it's normal form. A formal definition of concept forgetting for CTL follows in Section 3. From the point of view of model and resolution, we propose two algorithms of computing forgetting on CTL in Section 4. We conclude in Section 5 with a summary of the work and an outline of directions of future work. Besides, in the appendix we give the proofs of main results in this paper.

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 [Zhang *et al.*, 2009].

2.1 Model structure in CTL

In general, a transition system [1] 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 $\mathcal{M} = (S, R, L)$ [Emerson, 1990], 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

[1]According to [Baier and Katoen, 2008], 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) $\to \subseteq S \times \mathrm{Act} \times S$ is a transition relation, (4) $I \subseteq S$ is a set of initial states, (5) AP is a set of atomic propositions, and (6) $L : S \to 2^{\mathrm{AP}}$ is a labeling function.

- 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 [Browne *et al.*, 1988], 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 [Zhang *et al.*, 2009]) 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 ($i \le j$), $(s_j, s_{j+1}) \in R$. 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 [Browne *et al.*, 1988], for $n \ge 0$,

- $\text{Tr}_0(s)$ 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 $\text{Tr}_n(s')$[2].

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

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) [Clarke *et al.*, 1986]. 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 $\neg$;
- 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:

$$\phi ::= \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \text{EX}\phi \mid \text{EG}\phi \mid \text{E}[\phi \text{ U } \phi] \qquad (1)$$

where $p \in \mathcal{A}$. The formulas $\phi \wedge \psi$ and $\phi \to \psi$ are defined in a standard manner of propositional logic. Intuitively, the formula EXϕ means that ϕ holds in some immediate successor of the current program state; the formula EGϕ means that for some computation path ϕ holds at every state along the path; and the formula E$[\phi \text{U} \psi]$ means that for some computation path there is an initial prefix of the path such that ψ holds at the last state of the prefix and ϕ holds at all other states along the prefix. The other form formulas of $\mathcal{L}$ are abbreviated as follows using the forms of (1):

$$\begin{aligned}
&\top =_{\text{def}} \neg\bot, \\
&\text{A}[\phi \text{ U } \psi] =_{\text{def}} \neg(\text{E}[\neg\psi \text{U} (\neg\phi \wedge \neg\psi)] \vee \text{EG}\neg\psi), \\
&\text{AF}\phi =_{\text{def}} \text{A}[\top \text{U } \phi], \\
&\text{EF}\phi =_{\text{def}} \text{E}[\top \text{U}\phi], \\
&\text{AG}\phi =_{\text{def}} \neg\text{EF}\neg\phi, \\
&\text{AX}\phi =_{\text{def}} \neg\text{EX}\neg\phi, \\
&\text{A}(\varphi \text{W} \psi) =_{\text{def}} \neg\text{E}(\neg\psi \text{U}(\neg\varphi \wedge \neg\psi)), \\
&\text{E}(\varphi \text{W} \psi) =_{\text{def}} \neg\text{A}(\neg\psi \text{U}(\neg\varphi \wedge \neg\psi)).
\end{aligned}$$

Notice that, according to the above definition for formulas of CTL, each of the CTL *temporal connectives* has the form XY where $X \in \{\text{A}, \text{E}\}$ and $Y \in \{\text{X}, \text{F}, \text{G}, \text{U}, \text{W}\}$. The priorities for the CTL connectives are assumed to be (from the highest to the lowest):

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

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 $(\mathcal{M}, s) \not\models \phi$;
- $(\mathcal{M}, s) \models \text{EX}\phi$ iff $(\mathcal{M}, s_1) \models \phi$ for some $s_1 \in S$ and $(s, s_1) \in R$;
- $(\mathcal{M}, s) \models \text{EG}\phi$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \dots)$ such that $(\mathcal{M}, s_i) \models \phi$ for each $i \ge 1$;
- $(\mathcal{M}, s) \models \text{E}[\phi_1 \text{U} \phi_2]$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \dots)$ such that, for some $i \ge 1$, $(\mathcal{M}, s_i) \models \phi_2$ and $(\mathcal{M}, s_j) \models \phi_1$ for each $j < i$.

Similar to the work in [Browne *et al.*, 1988; Bolotov, 1999], 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 IR(ϕ, V), if there is a formula ψ with *Var*$(\psi) \cap V = \emptyset$ such that $\phi \equiv \psi$.

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

2.3 The normal form of CTL

It has proved that any CTL formula φ can be transformed into a set T_φ of $\text{SNF}^g_{\text{CTL}}$ (Separated Normal Form with Global Clauses for CTL) clauses in polynomial time such that φ is satisfiable iff T_φ is satisfiable [Zhang *et al.*, 2008]. An important difference between CTL formulae and $\text{SNF}^g_{\text{CTL}}$ is that $\text{SNF}^g_{\text{CTL}}$ 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 $\text{SNF}^g_{\text{CTL}}$ 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: $\text{E}_{\langle ind\rangle}\text{X}$, $\text{E}_{\langle ind\rangle}\text{F}$, $\text{E}_{\langle ind\rangle}\text{G}$,$\text{E}_{\langle ind\rangle}\text{U}$ and $\text{E}_{\langle ind\rangle}\text{W}$.

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

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

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

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

$$\text{AG}(\textbf{start} \supset \bigvee_{j=1}^{k} m_j) \qquad (initial\ clause)$$

$$\text{AG}(true \supset \bigvee_{j=1}^{k} m_j) \qquad (global\ clause)$$

$$\text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{AX} \bigvee_{j=1}^{k} m_j) \qquad (\text{A} - step\ clause)$$

$$\text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{E}_{\langle ind\rangle}\text{X} \bigvee_{j=1}^{k} m_j) \qquad (\text{E} - step\ clause)$$

$$\text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{AF}l) \qquad (\text{A} - sometime\ clause)$$

$$\text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{E}_{\langle ind\rangle}\text{F}l) \qquad (\text{E} - sometime\ clause).$$

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 $\text{SNF}^g_{\text{CTL}}$ clause unless explicitly stated.

Formulae of $\text{SNF}^g_{\text{CTL}}$ 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 $j \geq i$, $(s_j, s_{j+1}) \in [ind]$. Let π be a path in Ind-model structure $\mathcal{M}$, by $s \in \pi$ we mean that s is a state in the path π.

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 $\text{SNF}^g_{\text{CTL}}$ 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 $\text{SNF}^g_{\text{CTL}}$ formulae and $\mathcal{M} = (S, R, L, [\_], s_0)$ be an Ind-model structure, the relation "$\models$" between $\text{SNF}^g_{\text{CTL}}$ formulae and $\mathcal{M}$ is defined recursively as follows:

- $(\mathcal{M}, s_i) \models$ **start** iff $s_i = s_0$;
- $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}\text{X}\psi$ iff for the path $\pi_{s_i}^{\langle ind\rangle}$, $(\mathcal{M}, s_{i+1}) \models \psi$;
- $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}\text{G}\psi$ iff for every $s_j \in \pi_{s_i}^{\langle ind\rangle}$, $(\mathcal{M}, s_j) \models \psi$;
- $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}[\varphi\text{U}\psi]$ 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 $(\mathcal{M}, s_k) \models \varphi$;
- $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}\text{F}\psi$ iff $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}[\top\text{U}\psi]$;
- $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}[\varphi\text{W}\psi]$ iff $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}\text{G}\varphi$ or $(\mathcal{M}, s_i) \models \text{E}_{\langle ind\rangle}[\varphi\text{U}\psi]$.

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 $\text{SNF}^g_{\text{CTL}}$ 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 $\bigwedge_{\psi \in T} \psi \wedge \varphi$. Other notations is similar with those notations in section 2.2.

3 Forgetting on CTL

In this section we will propose the concepts of V-bisimulation and $\langle V, I\rangle$-bisimulation, which are called the set-based bisimulations, between K-structures and the characterizing formula of an initial structure. After that the definition of forgetting will be proposed from a sematic forgetting point of view. Besides, some properties of forgetting are also explored.

3.1 Set-based bisimulation

Let $V \subseteq \mathcal{A}$. Inspired by the notion of bisimulation in [Browne *et al.*, 1988], we define the relations $\mathcal{B}_0, \mathcal{B}_1, \ldots$ between K-structures as follows: let $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ and $\mathcal{M}_i = \langle S_i, R_i, L_i, s_i\rangle$ with $i \in \{1, 2\}$,

- $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_0$ if $L_1(s_1) - V = L_2(s_2) - V$;
- for $n \geq 0$, $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_{n+1}$ if
  - $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_0$,
  - for every $(s_1, s_1') \in R_1$, there is $(s_2, s_2') \in R_2$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}_n$, and

– for every $(s_2, s_2') \in R_2$, there is $(s_1, s_1') \in R_1$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}_n$

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

In order to distinguish the relations $\mathcal{B}_0, \mathcal{B}_1, \ldots$ for different set $V \subseteq \mathcal{A}$, by $\mathcal{B}_i^V$ we mean the relation $\mathcal{B}_1, \mathcal{B}_2, \ldots$ for $V \subseteq \mathcal{A}$. Denote as $\mathcal{B}_0, \mathcal{B}_1, \ldots$ when the underlying set V is clear from their contexts or there is no confusion.

Lemma 1 *Let $\mathcal{B}_0, \mathcal{B}_1, \ldots$ be the ones in the above definition. Then, for each $i \geq 0$,*

(i) $\mathcal{B}_{i+1} \subseteq \mathcal{B}_i$;

(ii) there is the leat number $k \geq 0$ such that $\mathcal{B}_{k+1} = \mathcal{B}_k$;

(iii) $\mathcal{B}_i$ is reflexive, symmetric and transitive.

Now, we define the notion of V-bisimulation between K-structures:

Definition 1 (V-bisimulation) *Let $V \subseteq \mathcal{A}$. The V-bisimular relation $\mathcal{B}$ between* K*-structures is defined as:*

$$(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B} \text{ if and only if } (\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_i \text{ for all } i \geq 0.$$

In this case, $\mathcal{K}_1$ and $\mathcal{K}_2$ are called *V-bisimular*. It seems that two K-structures $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2)$ are V-bisimular if s_1 have the same labels as s_2 and for each successor s_1' (s_2') of s_1 (s_2) there is a successor s_2' (s_1') of s_2 (s_1) such that $(\mathcal{M}_1, s_1')$ and $(\mathcal{M}_2, s_2')$ are V-bisimular. Formally:

Proposition 1 *Let $V \subseteq \mathcal{A}$, $\mathcal{M}_i = (S_i, R_i, L_i, s_i)$ $(i = 1, 2)$ be model structures and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2)$ with $s_i \in S_i$ be* K*-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 $(s_1, s_1') \in R_1$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}$

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

On the one hand, the above set-based bisimulation is an extension of the bisimulation-equivalence of Definition 7.1 in [Baier and Katoen, 2008] in the sense that if $V = \mathcal{A}$ then our bisimulation is almost same to the latter. On the other hand, the above set-based bisimulation notion is similar to the s-tate equivalence in [Browne *et al.*, 1988]. But it is different in the sense that ours is defined on K-structures, while it is defined on states in [Browne *et al.*, 1988]. What's more, the set-based bisimulation notion is also different from the state-based bisimulation notion of Definition 7.7 in [Baier and Katoen, 2008], which is defined for states of a given K-structure.

Two pathes $\pi_i = (s_{i,1}, s_{i,2}, \ldots)$ of $\mathcal{M}_i$ with $i \in \{1, 2\}$ are *V-bisimular* if

$$(\mathcal{K}_{1,j}, \mathcal{K}_{2,j}) \in \mathcal{B} \text{ for every } j \geq 0$$

where $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$.

In the following we abbreviated $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}$ by $(s_1, s_2) \in \mathcal{B}$ when the underlying model structures of states s_1 and s_2 are clear from their contexts or there is no confusion. The V-bisimular relation is uniformly abbreviated as $\leftrightarrow_V$ for convenience.The next lemma easily follows from the above definition,

Lemma 2 *The relation $\leftrightarrow_V$ is reflexive, symmetric and transitive.*

Proof: It is clear from Lemma 1 due to there is the leat number $k \geq 0$ such that $\mathcal{B}_k = \mathcal{B}$. ■

Besides, the V-bisimulation has the union property on the sets of atoms, that is if two K-structures are V_i-bisimular $(i = 1, 2)$ respectively then they are $(V_1 \cup V_2)$-bisimular. Formally:

Proposition 2 ***(union)*** *Let $i \in \{1, 2\}$, $V_i \subseteq \mathcal{A}$, s_is be two states and π_is be two pathes. 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 $\pi_1 \leftrightarrow_{V_1 \cup V_2} \pi_2$.

Except the union property, the V-bisimulation has also another important property, called transitivity. That is:

Proposition 3 ***(transitivity)*** *Let $V_1, V_2 \subseteq \mathcal{A}$ $(V_1 \cap V_2 = \emptyset)$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2, 3)$ be* K*-structures such that $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$ and $\mathcal{K}_2 \leftrightarrow_{V_2} \mathcal{K}_3$. Then:*

(i) for each path $\pi_1 = (s_1 = s_{1,0}, s_{1,1}, s_{1,2}, \ldots)$ of $\mathcal{M}_1$ there is a path $\pi_2 = (s_2 = s_{2,0}, s_{2,1}, s_{2,2}, \ldots)$ of $\mathcal{M}_2$ such that $\pi_1 \leftrightarrow_{V_1} \pi_2$, and vice versa;

(ii) $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$.

This is different with the transitivity of $\leftrightarrow_V$, which show the transitivity between structures on the same set of atoms.

Proposition 4 ***(V-bisimular expansion)****. Let $V \subseteq W \subseteq \mathcal{A}$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2)$ with $\mathcal{M}_i = (S_i, R_i, L_i, s_i)$ be two* K*-structures. If $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ then $\mathcal{K}_1 \leftrightarrow_W \mathcal{K}_2$.*

Proof: It apparent that if two K-structures are bisimular in a small set then it must be bisimular in a big set. ■

Let $\mathcal{M} = (S, R, L, s_0)$ be a model structure on a finite set $\mathcal{A}$ of atoms, $V \subseteq \mathcal{A}$ and $\mathcal{B} = \{(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}, s') | s, s' \in S\}$. For $s \in S$, $[s]_\mathcal{B}$ denotes the equivalence class of state s under $\mathcal{B}$, *i.e.* , $[s]_\mathcal{B} = \{s' \in S | (s, s') \in \mathcal{B}\}$. Note that for $s' \in [s]_\mathcal{B}$ we have $[s']_\mathcal{B} = [s]_\mathcal{B}$. The set $[s]_\mathcal{B}$ is referred to as the $\mathcal{B}$-equivalence class of s. The V-quotient space of S under $\mathcal{B}$, denoted by $S/\mathcal{B} = \{[s]_\mathcal{B} | s \in S\}$, is the consisting of all $\mathcal{B}$-equivalence classes.

Definition 2 *For* K*-structure $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ a model structure on a finite set $\mathcal{A}$ of atoms, $V \subseteq \mathcal{A}$ and $\mathcal{B} = \{(\mathcal{M}, s) \leftrightarrow_{V'} (\mathcal{M}, s') | s, s' \in S\}$ where $V' = \mathcal{A} - V$, the V-quotient structure is $\mathcal{K}_{|V} = (\mathcal{M}^*, s_0^*)$ with $\mathcal{M}^* = (S^*, R^*, L^*, s_0^*)$ on V, where*

- *s_0^* is an element of $[s_0]_\mathcal{B}$,*
- *$S^* = S/\mathcal{B}$,*
- *$R^* = \{([s]_\mathcal{B}, [s']_\mathcal{B}) | \exists s_1 \in [s]_\mathcal{B}$ s.t. $\exists s_2 \in [s']_\mathcal{B}$ and $(s_1, s_2) \in R\}$ and*
- *$L^*([s]_\mathcal{B}) = L([s]_\mathcal{B}) \cap V$.*

Proposition 5 *For any* K*-structure $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ a model structure on a finite set $\mathcal{A}$ of atoms and $V \subseteq \mathcal{A}$, it holds that $\mathcal{K} \leftrightarrow_{V'} \mathcal{K}_{|V}$ where $V' = \mathcal{A} - V$.*

Proof: Base. It is apparent that $L(s_0) - V' = L*(s_0^*) - V'$; Step. (i) For any $(s_0, s_1) \in R$ there is $s_1' \in [s_1]_{\mathcal{B}}$ such that $([s_0^*]_{\mathcal{A}}, [s_1']_{\mathcal{A}}) \in R^*$ and $s_1 \leftrightarrow_{V'} [s_1']_{\mathcal{B}}$ by the Definition 2; (ii) Similarly, for any $([s_0^*]_{\mathcal{A}}, [s_1']_{\mathcal{B}}) \in R^*$ there is $(s_0, s_1) \in R$ such that $s_1 \in [s_1']_{\mathcal{B}}$ and $s_1 \leftrightarrow_{V'} [s_1']_{\mathcal{B}}$. ■

Intuitively, if two K-structures are V-bisimular, then they satisfy the same formula φ that dose not contain any atoms in V, *i.e.* IR(φ, V).

Theorem 1 *Let $V \subseteq \mathcal{A}$, $\mathcal{K}_i$ $(i = 1, 2)$ be two* K*-structures such that $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ and ϕ a formula with IR(ϕ, V). Then $\mathcal{K}_1 \models \phi$ if and only if $\mathcal{K}_2 \models \phi$.*

Let $V \subseteq \mathcal{A}$, $\mathcal{M}_i = (S_i, R_i, L_i, s_i)$ $(i = 1, 2)$ be model structures. A computation tree $\text{Tr}_n(s_1)$ of $\mathcal{M}_1$ is V-*bisimular* to a computation tree $\text{Tr}_n(s_2)$ of $\mathcal{M}_2$, written $(\mathcal{M}_1, \text{Tr}_n(s_1)) \leftrightarrow_V (\mathcal{M}_2, \text{Tr}_n(s_2))$ (or simply $\text{Tr}_n(s_1) \leftrightarrow_V \text{Tr}_n(s_2)$), if

- $L_1(s_1) - V = L_2(s_2) - V$,
- for every subtree $\text{Tr}_{n-1}(s_1')$ of $\text{Tr}_n(s_1)$, $\text{Tr}_n(s_2)$ has a subtree $\text{Tr}_{n-1}(s_2')$ such that $\text{Tr}_{n-1}(s_1') \leftrightarrow_V \text{Tr}_{n-1}(s_2')$, and
- for every subtree $\text{Tr}_{n-1}(s_2')$ of $\text{Tr}_n(s_2)$, $\text{Tr}_n(s_1)$ has a subtree $\text{Tr}_{n-1}(s_1')$ such that $\text{Tr}_{n-1}(s_1') \leftrightarrow_V \text{Tr}_{n-1}(s_2')$.

Please note that the last two conditions in the above definition hold trivially for $n = 0$.

Proposition 6 *Let $V \subseteq \mathcal{A}$ and $(\mathcal{M}_i, s_i)$ $(i = 1, 2)$ be two* K*-structures. Then*

$$(s_1, s_2) \in \mathcal{B}_n \textit{ if and only if } Tr_j(s_1) \leftrightarrow_V Tr_j(s_2) \textit{ for every } 0 \le j \le n.$$

This means that $\text{Tr}_j(s_1) \leftrightarrow_V \text{Tr}_j(s_2)$ for all $j \ge 0$ if $s_1 \leftrightarrow_V s_2$, otherwise there is some number k such that $\text{Tr}_k(s_1)$ and $\text{Tr}_k(s_2)$ are not V-bisimular.

Proposition 7 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be a model structure and $s, s' \in S$ such that $(s, s') \notin \mathcal{B}$. There exists a least number k such that $Tr_k(s)$ and $Tr_k(s')$ are not V-bisimular.*

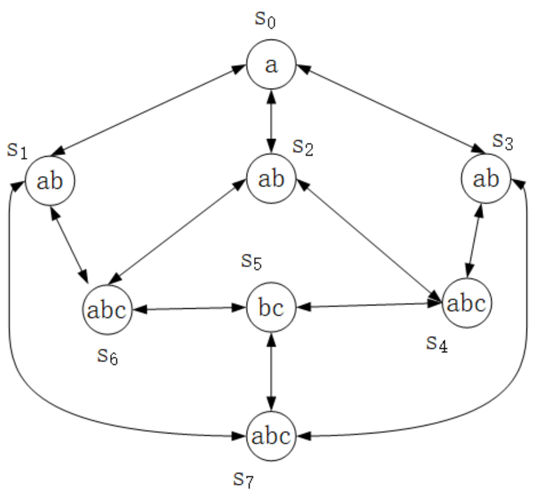
In this case the model structure $\mathcal{M}$ is called V-*distinguishable* (by states s and s' at the least depth k), which is denoted by $\text{dis}_V(\mathcal{M}, s, s', k)$. It is evident that $\text{dis}_V(\mathcal{M}, s, s', k)$ implies $\text{dis}_V(\mathcal{M}, s, s', k')$ whenever $k' \ge k$. The V-*characterization number* of $\mathcal{M}$, written $ch(\mathcal{M}, V)$, is defined as

$$ch(\mathcal{M}, V) = \begin{cases} \max\{k \mid s, s' \in S \,\&\, \text{dis}_V(\mathcal{M}, s, s', k)\}, & \\ \qquad\qquad \mathcal{M} \text{ is } V\text{-distinguishable}; & \\ \min\{k \mid \mathcal{B}_k = \mathcal{B}\}, & \text{otherwise.} \end{cases}$$

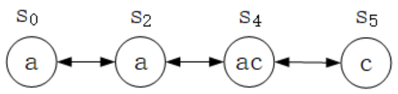
The following example shows the V-bisimulation intuitively:

Example 1 Let $\mathcal{M}$ as Fig. 1(a), $A = \{a, b, c\}$, $V = \{b\}$ and $\mathcal{K} = (\mathcal{M}, s_0)$. Then compute $\mathcal{K}_{|V}$.

For convenience, by $(s, s') \in B_i$ for $i \ge 0$ we mean $((\mathcal{M}, s), (\mathcal{M}, s')) \in B_i$ due to s and s' are under the same model structure. It is easy to check that:



(a) Transition system $\mathcal{M}$



(b) V-quotient Kripke structure of $\mathcal{M}$

Figure 1: Computing the V-quotient Kripke structure

(a) $(s_0, s_i) \in \mathcal{B}_0$, $(s_0, s_i) \notin \mathcal{B}_1$ with $i = 1, 2, 3$, then we have $\text{Tr}_0(s_0) \leftrightarrow_V \text{Tr}_0(s_i)$ and $\text{Tr}_1(s_0) \not\leftrightarrow_V \text{Tr}_1(s_i)$. Hence, $\mathcal{M}$ is V-distinguished by s_0 and s_i at the least depth 1, *i.e.* $\text{dis}_V(\mathcal{M}, s_0, s_i, 1)$;

(b) $(s_0, s_5) \notin \mathcal{B}_0$, then we have $\text{Tr}_0(s_0) \not\leftrightarrow_V \text{Tr}_0(s_5)$ and then $\mathcal{M}$ is V-distinguished by s_0 and s_5 at the least depth 0, *i.e.* $\text{dis}_V(\mathcal{M}, s_0, s_5, 0)$;

(c) $(s_0, s_i) \notin \mathcal{B}_0$ with $i = 4, 6, 7$, then we have $\text{Tr}_0(s_0) \not\leftrightarrow_V \text{Tr}_0(s_i)$ and then $\mathcal{M}$ is V-distinguished by s_0 and s_i at the least depth 0, *i.e.* $\text{dis}_V(\mathcal{M}, s_0, s_i, 0)$;

(d) $(s_i, s_j) \notin \mathcal{B}_0$ with $i = 1, 2, 3$ and $j = 4, 6, 7$, then we have $\text{Tr}_0(s_i) \not\leftrightarrow_V \text{Tr}_0(s_j)$ and then $\mathcal{M}$ is V-distinguished by s_i and s_j at the least depth 0, *i.e.* $\text{dis}_V(\mathcal{M}, s_i, s_j, 0)$;

(e) $(s_i, s_5) \notin \mathcal{B}_0$ with $i = 1, 2, 3$, then we have $\text{Tr}_0(s_i) \not\leftrightarrow_V \text{Tr}_0(s_5)$ and then $\mathcal{M}$ is V-distinguished by s_i and s_5 at the least depth 0, *i.e.* $\text{dis}_V(\mathcal{M}, s_i, s_5, 0)$;

(f) $(s_i, s_5) \notin \mathcal{B}_0$ with $i = 4, 6, 7$, then we have $\text{Tr}_0(s_i) \not\leftrightarrow_V \text{Tr}_0(s_5)$ and then $\mathcal{M}$ is V-distinguished by s_i and s_5 at the least depth 0, *i.e.* $\text{dis}_V(\mathcal{M}, s_i, s_5, 0)$.

Therefore, we have that $\mathcal{M}$ is V-distinguishable and $ch(\mathcal{M}, V) = \max\{k \mid s, s' \in S \,\&\, \text{dis}_V(\mathcal{M}, s, s', k)\} = 1$. In order to show that $(s, s') \in \mathcal{B}$, we only need to show that $(s, s') \in \mathcal{B}_i$ with $i = 0, 1$. It is easy to show that $(s_1, s_2) \in \mathcal{B}_0$ and $(s_1, s_2) \in \mathcal{B}_1$.

Hence, we can obtain $\mathcal{B} = \{(s_1, s_2), (s_2, s_1), (s_2, s_3), (s_3, s_2), (s_1, s_3), (s_3, s_1), (s_6, s_7), (s_7, s_6), (s_6, s_4), (s_4, s_6), (s_4, s_7), (s_7, s_4)\} \cup I_S$. Then we have the V-quotient structure is as Fig. 1(b).

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 $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_i)$ 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 Ind-structures is a set that satisfy $((\mathcal{M}_1, s_1), (\mathcal{M}_2, s_2)) \in \beta_{\langle V,I\rangle}$ if and only if:*

(i) $L_1(s_1) - V \equiv L_2(s_2) - V$;

(ii) $\forall (s_1, s_1') \in R_1$, $\exists (s_2, s_2') \in R_2$ *s.t.* $((\mathcal{M}_1, s_1'), (\mathcal{M}_2, s_2')) \in \beta_{\langle V,I\rangle}$;

(iii) $\forall (s_2, s_2') \in R_2$, $\exists (s_1, s_1') \in R_1$ *s.t.* $((\mathcal{M}_1, s_1'), (\mathcal{M}_2, s_2')) \in \beta_{\langle V,I\rangle}$;

(iv) $\forall i \notin I$ *there is* $[i]_1 = [i]_2$.

Apparently, this definition is similar with our concept V-bisimulation except that this $\langle V, I\rangle$-bisimulation has introduced the index. Besides, it is not difficult to prove $\langle V, I\rangle$-bisimulation possess those properties (talked-above) possessed by V-bisimulation.

3.2 Characterize formula of initial K-structure

Given a set $V \subseteq \mathcal{A}$, we can define a formula φ of V (that is $Var(\varphi) \subseteq V$) in CTL to equivalent uniquely describe a computation tree.

Definition 4 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be a model structure and $s \in S$. The* characterize formula *of the computation tree $Tr_n(s)$ on V, written $\mathcal{F}_V(Tr_n(s))$, is defined recursively as:*

$$\mathcal{F}_V(Tr_0(s)) = \bigwedge_{p \in V \cap L(s)} p \wedge \bigwedge_{q \in V - L(s)} \neg q,$$

$$\mathcal{F}_V(Tr_{k+1}(s)) = \left(\bigwedge_{(s,s') \in R} \text{EX}\mathcal{F}_V(Tr_k(s'))\right) \wedge$$

$$\text{AX}\left(\bigvee_{(s,s') \in R} \mathcal{F}_V(Tr_k(s'))\right) \wedge \mathcal{F}_V(Tr_0(s))$$

for $k \geq 0$.

It is apparent that $\text{IR}(\mathcal{F}_V(\text{Tr}_n(s)), \overline{V})$. As we will see that the characterize formula $\mathcal{F}_V(\text{Tr}_n(s))$ equivalent uniquely describe the computation tree $\text{Tr}_n(s)$ on V, that is for any formula φ of V, if φ is a characterize formula of $\text{Tr}_n(s)$ then $\varphi \equiv \mathcal{F}_V(\text{Tr}_n(s))$.

Lemma 3 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{M}' = (S', R', L', s_0')$ be two model structures, $s \in S$, $s' \in S'$ and $n \geq 0$.*

(i) $(\mathcal{M}, s) \models \mathcal{F}_V(Tr_n(s))$.

(ii) If $(\mathcal{M}, s) \models \mathcal{F}_V(Tr_n(s'))$ then $Tr_n(s) \leftrightarrow_{\overline{V}} Tr_n(s')$.

A consequence of the previous lemma is:

Lemma 4 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ a model structure, $k = ch(\mathcal{M}, V)$ and $s \in S$.*

- $(\mathcal{M}, s) \models \mathcal{F}_V(Tr_k(s))$, *and*
- *for each $s' \in S$, $(\mathcal{M}, s) \leftrightarrow_{\overline{V}} (\mathcal{M}, s')$ if and only if $(\mathcal{M}, s') \models \mathcal{F}_V(Tr_k(s))$.*

Proof: Let $\phi = \mathcal{F}_V(\text{Tr}_k(s))$, where c is the V-characteristic number of $\mathcal{M}$. $\mathcal{M}, s \models \phi$ by the definition of $\mathcal{F}$, and then $\forall s' \in S$, if $s \leftrightarrow_{\overline{V}} s'$ there is $\mathcal{M}, s' \models \phi$ by Theorem 1 due to $\text{IR}(\phi, \mathcal{A} \setminus V)$. If $s \not\leftrightarrow_{\overline{V}} s'$, then $\text{Tr}_c(s) \not\leftrightarrow_{\overline{V}} \text{Tr}_c(s')$, and then $\mathcal{M}, s \nvDash \phi$ by Lemma 3. ■

Lemma 5 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{M}' = (S', R', L', s_0')$ be two model structures, $s \in S$, $s' \in S'$ and $n \geq 0$. If $Tr_n(s) \leftrightarrow_{\overline{V}} Tr_n(s')$, then $\mathcal{F}_V(Tr_n(s)) \equiv \mathcal{F}_V(Tr_n(s'))$.*

Let $s' = s$, this show that for any formula φ of V, if φ is a characterize formula of $\text{Tr}_n(s)$ then $\varphi \equiv \mathcal{F}_V(\text{Tr}_n(s))$.

Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{K} = (\mathcal{M}, s_0)$ be an initial K-structure. The *characterizing formula* of $\mathcal{K}$ on V, written $\mathcal{F}_V(\mathcal{M}, s_0)$ (or $\mathcal{F}_V(\mathcal{K})$), is defined as the conjunction of the following formulas:

$\mathcal{F}_V(\text{Tr}_c(s_0))$, and

$$\text{AG}\left(\mathcal{F}_V(\text{Tr}_c(s)) \to \bigwedge_{(s,s') \in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s')) \wedge \text{AX} \bigvee_{(s,s') \in R} \mathcal{F}_V(\text{Tr}_c(s'))\right)$$

$, s \in S$

where $c = ch(\mathcal{M}, V)$. It is apparent that $\text{IR}(\mathcal{F}_V(\mathcal{M}, s_0), \overline{V})$.

Lemma 6 *Let φ be a formula. We have*

$$\varphi \equiv \bigvee_{(\mathcal{M}, s_0) \in Mod(\varphi)} \mathcal{F}_{\mathcal{A}}(\mathcal{M}, s_0). \qquad (2)$$

This means that any CTL formula can be described by the disjunction of the characterizing formulas of all the models of itself due to the number of models of a CTL formula is finite.

Theorem 2 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be a model structure with initial state s_0 and $\mathcal{M}' = (S', R', L', s_0')$ be a model structure with initial state s_0'. Then*

$(\mathcal{M}', s_0') \models \mathcal{F}_V(\mathcal{M}, s_0)$ *if and only if* $(\mathcal{M}, s_0) \leftrightarrow_{\overline{V}} (\mathcal{M}', s_0')$.

We will give an example to show the computing of characterizing formula:

Example 2 Let $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ be a initial K-structure (in Fig. 2), in which $S = \{s_0, s_1, s_2\}$, $R = \{(s_0, s_1), (s_0, s_2), (s_1, s_0), (s_2, s_0)\}$, $L(s_0) = \{a\}$, $L(s_1) = \{a, c\}$ and $L(s_2) = \{b, c\}$. Let $V = \{a, b\}$, compute the characterizing formula of $\mathcal{K}$ on V.

It is apparent that $\text{Tr}_0(s_0) \leftrightarrow_{\overline{V}} \text{Tr}_0(s_1)$ due to $L(s_0) - \overline{V} = L(s_1) - \overline{V}$, $\text{Tr}_1(s_0) \not\leftrightarrow_{\overline{V}} \text{Tr}_1(s_1)$ due to there is $(s_0, s_2) \in R$ such that for any $(s_1, s') \in R$ (there is only one immediate successor $s' = s_0$) there is $L(s_2) - \overline{V} \neq L(s') - \overline{V}$. Hence, we have that $\mathcal{M}$ is $\overline{V}$-distinguished by state s_0 and s_1 at the least depth 1, *i.e.* $\text{dis}_{\overline{V}}(\mathcal{M}, s_0, s_1, 1)$. Similarly, we have $\text{dis}_{\overline{V}}(\mathcal{M}, s_0, s_2, 0)$ and $\text{dis}_{\overline{V}}(\mathcal{M}, s_1, s_2, 0)$. Therefore, $ch(\mathcal{M}, \overline{V}) = \max\{k \mid s, s' \in S \;\&\; \text{dis}_{\overline{V}}(\mathcal{M}, s, s', k)\} = 1$.

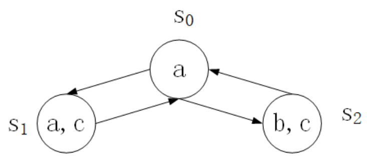


Figure 2: A simple Kripke structure

Then we have:

$\mathcal{F}_V(\text{Tr}_0(s_0)) = a \wedge \neg b,$
$\mathcal{F}_V(\text{Tr}_0(s_1)) = a \wedge \neg b,$
$\mathcal{F}_V(\text{Tr}_0(s_2)) = b \wedge \neg a,$
$\mathcal{F}_V(\text{Tr}_1(s_0)) = \text{EX}(a \wedge \neg b) \wedge \text{EX}(b \wedge \neg a) \wedge \text{AX}((a \wedge \neg b) \vee (b \wedge \neg a)) \wedge (a \wedge \neg b),$
$\mathcal{F}_V(\text{Tr}_1(s_1)) = \text{EX}(a \wedge \neg b) \wedge \text{AX}(a \wedge \neg b) \wedge (a \wedge \neg b),$
$\mathcal{F}_V(\text{Tr}_1(s_2)) = \text{EX}(a \wedge \neg b) \wedge \text{AX}(a \wedge \neg b) \wedge (b \wedge \neg a).$
Then it is easy to obtain $\mathcal{F}_V(\mathcal{M}, s_0)$.

By the following theorem we also have that given a set $V \subseteq \mathcal{A}$, the characterizing formula of an initial K-structure is equivalent uniquely describe this initial K-structure on V.

Theorem 3 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ a model structure with initial state s_0 and $\mathcal{M}' = (S', R', L', s_0')$ a model structure with initial state s_0'. If $(\mathcal{M}, s_0) \leftrightarrow_{\overline{V}} (\mathcal{M}', s_0')$ then $\mathcal{F}_V(\mathcal{M}, s_0) \equiv \mathcal{F}_V(\mathcal{M}', s_0')$.*

Proof: This is following Lemma 5 and the definition of the characterizing formula of initial K-structure $\mathcal{K}$ on V. ■

3.3 Forgetting on CTL

Having talked about the V-bisimulation between two K-structures and the characterizing formula of a K-structure, we will give the definition of forgetting under CTL from the sematic forgetting point of view.

Definition 5 (Forgetting) *Let $V \subseteq \mathcal{A}$ and ϕ a formula. A formula ψ with $Var(\psi) \cap V = \emptyset$ is a* result of forgetting V from ϕ, *if*

$$Mod(\psi) = \{\mathcal{K} \text{ is initial} \mid \exists \mathcal{K}' \in Mod(\phi) \ \&\ \mathcal{K}' \leftrightarrow_V \mathcal{K}\}. \quad (3)$$

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). By Lemma 6, such a formula ψ always exists, which is equivalent to

$$\bigvee_{\mathcal{K} \in \{\mathcal{K}' \text{ is an initial interpretation} \mid \exists \mathcal{K}'' \in Mod(\phi) \ \&\ \mathcal{K}'' \leftrightarrow_V \mathcal{K}'\}} \mathcal{F}_{\overline{V}}(\mathcal{K}).$$

For this reason, the forgetting result is denoted by $\text{F}_{\text{CTL}}(\phi, V)$. By the definition of forgetting, we have

Proposition 8 *Let φ be a CTL formula and V a set of atoms. If $V \cap Var(\varphi) = \emptyset$, then*

$$\text{F}_{\text{CTL}}(\varphi, V) \equiv \varphi.$$

Proof: $(\Rightarrow)\ \forall(\mathcal{M}, s_0) \in Mod(\text{F}_{\text{CTL}}(\varphi, V))$
$\Rightarrow$ there is $(\mathcal{M}', s_0')$ s.t. $(\mathcal{M}', s_0') \leftrightarrow_V (\mathcal{M}, s_0)$ and $(\mathcal{M}', s_0') \models \varphi$
$\Rightarrow (\mathcal{M}, s_0) \models \varphi$ $\qquad$ $(\text{IR}(\varphi, V))$

$(\Leftarrow)\ \forall(\mathcal{M}, s_0) \in Mod(\varphi)$ there is $(\mathcal{M}, s_0) \models \text{F}_{\text{CTL}}(\varphi, V)$ by (W). ■

In the case ψ is a result of forgetting V from ϕ, there are usually some expected properties (called *postulates*) for them [Zhang and Zhou, 2009b]:

- Weakening (**W**): $\varphi \models \psi$;
- Positive Persistence (**PP**): if $\text{IR}(\eta, V)$ and $\varphi \models \eta$, then $\psi \models \eta$;
- Negative Persistence (**NP**): if $\text{IR}(\eta, V)$ and $\varphi \not\models \eta$, then $\psi \not\models \eta$;
- Irrelevance (**IR**): $\text{IR}(\psi, V)$.

Theorem 4 *Let φ and ψ be two formulas and $V \subseteq \mathcal{A}$. Then the following statements are equivalent:*

(i) $\psi \equiv \text{F}_{\text{CTL}}(\varphi, V)$,
(ii) $\psi \equiv \{\phi | \varphi \models \phi \& IR(\phi, V)\}$,
*(iii) Postulates (**W**), (**PP**), (**NP**) and (**IR**) hold.*

We can see from this theorem that the forgetting under CTL is closed, *i.e.* for any CTL formula the result of forgetting is also a CTL formula.

Lemma 7 *Let φ and α be two CTL formulae and $q \in \overline{Var(\varphi \cup \{\alpha\})}$. Then $\text{F}_{CTL}(\varphi \cup \{q \leftrightarrow \alpha\}, q) \equiv \varphi$.*

Proposition 9 *Let φ be a formula, V a set of atoms and p an atom such that $p \notin V$. Then:*

$$\text{F}_{\text{CTL}}(\varphi, \{p\} \cup V) \equiv \text{F}_{\text{CTL}}(\text{F}_{\text{CTL}}(\varphi, p), V).$$

This means that the result of forgetting V from φ can be obtained by forgetting atom in V one by one. Similarly, a consequence of the previous proposition is:

Corollary 5 *Let φ be a formula and $V_i \subseteq \mathcal{A}$ $(i = 1, 2)$. Then:*

$$\text{F}_{\text{CTL}}(\varphi, V_1 \cup V_2) \equiv \text{F}_{\text{CTL}}(\text{F}_{\text{CTL}}(\varphi, V_1), V_2).$$

The following results, which are satisfied in both classical proposition logic and modal logic **S5**, further illustrate other essential semantic properties of forgetting.

Proposition 10 *Let φ, φ_i, ψ_i $(i = 1, 2)$ be formulas and $V \subseteq \mathcal{A}$. We have*

(i) $\text{F}_{\text{CTL}}(\varphi, V)$ *is satisfiable iff φ is;*
(ii) If $\varphi_1 \equiv \varphi_2$, then $\text{F}_{\text{CTL}}(\varphi_1, V) \equiv \text{F}_{\text{CTL}}(\varphi_2, V)$;
(iii) If $\varphi_1 \models \varphi_2$, then $\text{F}_{\text{CTL}}(\varphi_1, V) \models \text{F}_{\text{CTL}}(\varphi_2, V)$;
(iv) $\text{F}_{\text{CTL}}(\psi_1 \vee \psi_2, V) \equiv \text{F}_{\text{CTL}}(\psi_1, V) \vee \text{F}_{\text{CTL}}(\psi_2, V)$;
(v) $\text{F}_{\text{CTL}}(\psi_1 \wedge \psi_2, V) \models \text{F}_{\text{CTL}}(\psi_1, V) \wedge \text{F}_{\text{CTL}}(\psi_2, V)$.

Another interest result is that the forgetting of the fragment $PT\varphi$ $(P \in \{\text{E}, \text{A}\}, T \in \{\text{F}, \text{X}\})$ on $V \subseteq \mathcal{A}$ can be computed by $PT\text{F}_{\text{CTL}}(\varphi, V)$. This give a convenient method to compute forgetting.

Proposition 11 *Let $V \subseteq \mathcal{A}$ and ϕ a formula.*

(i) $\mathrm{F}_{\text{CTL}}(\text{AX}\phi, V) \equiv \text{AX}\mathrm{F}_{\text{CTL}}(\phi, V)$.

(ii) $\mathrm{F}_{\text{CTL}}(\text{EX}\phi, V) \equiv \text{EX}\mathrm{F}_{\text{CTL}}(\phi, V)$.

(iii) $\mathrm{F}_{\text{CTL}}(\text{AF}\phi, V) \equiv \text{AF}\mathrm{F}_{\text{CTL}}(\phi, V)$.

(iv) $\mathrm{F}_{\text{CTL}}(\text{EF}\phi, V) \equiv \text{EF}\mathrm{F}_{\text{CTL}}(\phi, V)$.

4 Algorithm to compute forgetting

To compute the forgetting in CTL, we propose two methods, model-based and resolution-based, in this part. Literally speaking, the model-based method means that we can obtain the result of forgetting in CTL by obtain all the possible finite K-models of this result. How can we obtain all the K-models is what we will solved in this part. The resolution-based method obtain the result of forgetting by obtaining all the possible resolutions which can be implied by the original formula.

However, the resolution-based method is different from that in CPL [Wang, 2015] due to it is easy to transform any CPL formula into a CNF. However, it is difficult to transform a CTL formula into a form similar with CNF in CTL. Fortunately, there is an extension of CTL such that any CTL formula can be transformed into the extension in polynomial time. And we will give a detailed description later. Before that, let turn to the model-based method at first.

4.1 A model-based algorithm

As we have said that the set models of any formula φ is finite, hence if we can obtain all the models of φ then we can express this formula by the disjunction of those characteristic formulas of those models. By the definition of forgetting in CTL, the set of models of the result of forgetting is a finite set of initial K-structures. Then the model-based method is generated for forgetting in CTL.

Though the set of models of the result of forgetting is finite, while how many models is there? That's right we should given the bound of the number of the models. As it is said in Proposition– that if two initial K-structures are V-bisimulation, then their characteristic formulas is equal. Then let φ be a CTL formula, the $|Var(\varphi)| = m$ is a positive integer, we have the following theorem:

Theorem 6 *Let φ be a CTL formula, $V = Var(\varphi)$, $|V| = m$, and $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ be a initial* K*-structure. If $(\mathcal{M}, s_0) \models \varphi$, then there is an initial* K*-structure $\mathcal{K}' = (\mathcal{M}', s_0')$ with $\mathcal{M}' = (S', R', L', s_0')$ that satisfy:*

(i) $|S'|$ *is at most* 2^m,

(ii) $|R'|$ *is at most* $2^m * 2^m$,

(iii) $|L'|$ *is at most* $2^m * 2^m$,

(iv) $\mathcal{K} \leftrightarrow_{\mathcal{A}\setminus V} \mathcal{K}'$ *and* $(\mathcal{M}', s_0') \models \varphi$.

Proof: If $|S| \leq 2^m$, this result clearly holds. If $|S| > 2^m$, let $\mathcal{K}' = \mathcal{K}_{|V}$, then it is apparent that $\mathcal{K} \leftrightarrow_{\mathcal{A}\setminus V} \mathcal{K}'$ by Proposition 5 and $(\mathcal{M}', s_0') \models \varphi$. In the worst case, the number of states in S' is 2^m by the definition of $\mathcal{K}_{|V}$. Then the theorem is proved. ■

By this theorem, we can see that any initial K-structures $\mathcal{K}$ that satisfy φ can be transformed to an initial K-structure $\mathcal{K}'$ such that $\mathcal{K} \leftrightarrow_{\mathcal{A}\setminus V} \mathcal{K}'$ and $\mathcal{K} \models \varphi$ iff $\mathcal{K}' \models \varphi$ due to $\text{IR}(\varphi, \mathcal{A} \setminus V)$, in which $V = Var(\varphi)$. Therefore, the size of the model of φ is at most 2^m by Theorem 3(we only consider the number of states of this model).

Then we have the following model-based Algorithm 1 to computing the forgetting under CTL:

```
Algorithm 1: Model-based: Computing forgetting
   Input: A CTL formula $\varphi$ and a set $V$ of atoms
   Output: $\mathrm{F}_{\text{CTL}}(\varphi, V)$
1  $T = \emptyset$ // the set of models of $\varphi$ ;
2  $T' = \emptyset$ // the set of models of $\mathrm{F}_{\text{CTL}}(\varphi, V)$ ;
3  $m = |Var(\varphi)|$;
4  $n = |V|$;
5  for i=1, ..., $2^m$ do
6      Enumerating all possible initial K-structures $(\mathcal{M}, s_0)$
       with $\mathcal{M} = (S, R, L, s_0)$ and $|S| = i$;
7      For all initial K-structures $(\mathcal{M}, s_0)$ if $(\mathcal{M}, s_0) \models \varphi$
       then
8          $T = T \cup \{(\mathcal{M}, s_0)\}$;
9      end
10 end
11 for $\mathcal{K} = (\mathcal{M}, s_0) \in T$ do
12     Let $T' = T' \cup \{\mathcal{K}_{|V}\}$;
13 end
14 for i=1, ..., $2^{m-n}$ do
15     Enumerating all possible initial K-structures
       $(\mathcal{M}', s_0')$ with $\mathcal{M}' = (S', R', L', s_0')$ and $|S'| = i$;
16     For all initial K-structures $(\mathcal{M}', s_0')$ if $\exists(\mathcal{M}, s_0) \in T$
       s.t. $(\mathcal{M}, s_0) \leftrightarrow_V (\mathcal{M}', s_0')$ then
17         $T' = T' \cup \{(\mathcal{M}', s_0')\}$;
18     end
19 end
20 return $\bigvee_{(\mathcal{M}', s_0') \in T'} \mathcal{F}_{\overline{V}}(\mathcal{M}', s_0')$.
```

Proposition 12 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$. The time complexity of Algorithm 1 is $O(2^{2^m})$ and the space complexity is $O(2^{2m})$. Where $|Var(\varphi)| = m$ and $|V| = n$.*

Proof: The time and space spend by Algorithm 1 is mainly the **for** cycles from sentence 5 to 10. Under a given number i of states, there are i^i number of relations, i^m number of label functions and i number of possible initial states. In the case, we need the memory for the initial K-model in each time is $(i + i^2 + i * m + 1)$. Therefore, in the worst case is $i = 2^m$, that is we need $(2^m + 2^{2m} + m * 2^m + 1)$ memory to store the initial K-model.

For the time complexity, for each $1 \leq i \leq 2^m$, there is at most $i * i^i * i^m * i = i^2 * 2^{(i+m)}$ possible initial K-models. If we suppose that we can obtain an initial K-models in unit time, then in there will spend $(2^m)^2 * 2^{2^m+m}$ unit time in the worst case. Therefore, the time complexity is $O(2^{2^m})$. ■

4.2 A resolution-based algorithm

As said above that the time complexity of the model-based method is $O(2^{2^m})$, it is crazy even for $m = 4$. Dose there is an efficient time to do this work? The answer is yes! In this part, 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 [Zhang *et al.*, 2009].

The main idea of this method is made up of three parts: (1) Transform a CTL formula φ into a set T_φ of $\text{SNF}^g_{\text{CTL}}$ clauses; (2) Do all the possible resolutions on the set V of atoms we want to forget and the set V' introduced in (1); and (3) Eliminate those clauses that include at least one atom in $V \cup V'$. For (1) and (2) there are two good literals [Bolotov, 2000; Zhang *et al.*, 2009] to do it. In this paper we resolve the problem (3) and the problem of how to compute forgetting in CTL with the resolution in CTL with index. As we will see that the $\langle V, I\rangle$-bisimulation relation set up a bridge for CTL and $\text{SNF}^g_{\text{CTL}}$ language.

Proposition 13 *Let φ, P, C and D be four formulae, where m_j $(1 \le j \le k)$ are literals. Then*

(i) $\varphi \equiv \text{AG}(\textbf{\textit{start}} \supset \varphi)$,

(ii) $P \supset C \vee D \equiv \text{AG}(\textbf{\textit{start}} \wedge P \supset C \vee D)$.

Which means that we can change a set of $\text{SNF}^g_{\text{CTL}}$ to another set without **start**.

Let T be a set of $\text{SNF}^g_{\text{CTL}}$ clauses, we define T', called *nonInd*-$\text{SNF}^g_{\text{CTL}}$, as follows:

$$T' = \{C' | C' = P \supset \text{E}\urcorner D \text{ if } C = P \supset \text{E}_{\langle ind\rangle}\urcorner D \\ \text{else } C' = C,\ C \in T, \urcorner \in \{\text{X}, \text{F}\}\}.$$

Lemma 8 *(NI-BRemain) Let T be a set of $\text{SNF}^g_{\text{CTL}}$ clauses and T' be the nonInd-$\text{SNF}^g_{\text{CTL}}$ of T. Then we have $T \equiv_{\langle \emptyset, I\rangle} T'$, where I is the set of indexes in T.*

Similarly, let T be a set of $\text{SNF}^g_{\text{CTL}}$ clauses, then we define the following operator:

$$T_{\text{CTL}} = \{C | C' \in T' \text{ and } C = D \text{ if } C' \in T \\ \text{is the form AG}(\textbf{start} \supset D), \text{else } C = C'\}.$$

It is obvious that $T' \equiv T_{\text{CTL}}$ by Proposition 13.

The transformation of an arbitrary CTL formula into the set T_φ start with the set $T_0 = \{\text{AG}(\textbf{start} \supset p), \text{AG}(p \supset \textbf{simp}(\textbf{nnf}(\varphi)))\}$ (it has proved that $\textbf{simp}(\textbf{nnf}(\varphi)) \equiv \varphi$ [Bolotov, 2000]), where p is a new proposition that dose not occur in φ, **nnf** is a function that transforms an arbitrary CTL formula into its negation normal form (NNF) by pushing negations inwards and **simp** is a function simplifies an arbitrary CTL formula by simplification rules [Zhang *et al.*, 2009]. And then construct a sequence $T_0, T_1, \ldots, T_n = T_\varphi$ of formulea such that for every i $(0 \le i < n)$, $T_{i+1} = (T_i \setminus \{\psi\}) \cup R_i$, where ψ is a formula in T_i not in $\text{SNF}^g_{\text{CTL}}$ clause and R_i is the result set of applying a matching transformation rule to ψ. Besides, throughout the transformation formulae are kept in NNF. 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 $\forall(\mathcal{M}, s_0) \in \textit{Mod}(T)$ 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. Then we have:

Lemma 9 *(T0-VIB) Let φ be a CTL formula, then $\varphi \equiv_{\langle \{p\}, \emptyset\rangle} T_0$.*

Proof: $(\Rightarrow)$ $\forall(\mathcal{M}_1, s_1) \in \textit{Mod}(\varphi)$, *i.e.* $(\mathcal{M}_1, s_1) \models \varphi$. We can construct an Ind-model structure $\mathcal{M}_2$ is identical to $\mathcal{M}_1$ except $L_2(s_2) = L_1(s_1) \cup \{p\}$. It is apparent that $(\mathcal{M}_2, s_2) \models T_0$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset\rangle} (\mathcal{M}_2, s_2)$.

$(\Leftarrow)$ $\forall(\mathcal{M}_1, s_1) \in \textit{Mod}(T_0)$, it is apparent that $(\mathcal{M}_1, s_1) \models \varphi$ by the sematic of **start**. ■

By $\psi \to_t R_i$ we mean using transformation rules t on formula ψ (the formulae ψ as the premises of rule t) and obtaining the set R_i of transformation results. Let X be a set of formulas we have the following result.

Lemma 10 *If $\psi \to_t R_i$ by an application of $t \in \{$Trans(1), ..., Trans(12)$\}$, then $T_i \equiv_{\langle \{p\}, \{ind\}\rangle} T_{i+1}$. Where $T_i = X \cup \{\psi\}$, $T_{i+1} = X \cup R_i$ and p (if any), ind (if any) are the atom and index introduced by using that rule respectively.*

Proposition 14 *Let φ be a CTL formula, then we have $\varphi \equiv_{\langle V,I\rangle} T_\varphi$. Where V be a set of new propositions introduced in the process of translate φ into T_φ, and I be the set of index appearing in T_φ.*

Proof: This can be proved from Lemma 9 and Lemma 10. ■

A *derivation* on a set V of atoms from a set T of $\text{SNF}^g_{\text{CTL}}$ clauses by using those resolution rules in [Zhang *et al.*, 2009] on V (*i.e.* do resolutions only on the atoms in V) is a sequence $T_0, T_1, T_2, \ldots, T_n = T^{r,V}$ of sets of $\text{SNF}^g_{\text{CTL}}$ clauses such that $T = T_0$ and $T_{i+1} = T_i \cup R_i$ where R_i is a set of clauses obtained as the conclusion of the application of a resolution rule to premises in T_i. And we call $T^{r,V}$ is the result of resolution of T on V by using resolution rules. Besides, if there is a T_i containing **start** $\supset \bot$ or $\top \supset \bot$, then T_i is unsatisfiable and then φ is unsatisfiable. In this case we do not need to compute other T_j $(j > i)$. By $\psi \to_r R_i$ we mean using resolution rules r on set ψ (the formulae in ψ as the premises of rule r) and obtaining the set R_i of resolution results. For resolution rules, we have the following results.

Lemma 11 *If $\psi \to_r R_i$ by an application of $r \in \{$**(SRES1)**, ..., **(SRES8)**, **RW1**, **RW2**$\}$, then $T_i \equiv_{\langle \{p\}, \emptyset\rangle} T_{i+1}$. Where $T_i = X \cup \psi$, $T_{i+1} = X \cup R_i$, p be the proposition corresponding with literal l used to do resolution in r.*

Proof: On one hand, it is apparent that $\psi \models R_i$ and then $T_i \models T_{i+1}$. On the other hand, $T_i \subseteq T_{i+1}$ and then $T_{i+1} \models T_i$. ■

Lemma 12 *If $\psi \to_r R_i$ by an application of $r =$**(ERES1)**, then $T_i \equiv_{\langle \{l, w^{\text{A}}_{\neg l}\}, \emptyset\rangle} T_{i+1}$. Where $T_i = X \cup \psi$, $T_{i+1} = X \cup R_i$, p be the proposition corresponding with literal l used to do resolution in r.*

Proof: It has been proved that $\psi \models R_i$ in [Bolotov, 2000], then there is $T_{i+1} = T_i \cup \Lambda^{\text{A}}_{\neg l}$ and then $\forall(\mathcal{M}_1, s_1) \in \textit{Mod}(T_i = X \cup \psi)$ there is a $(\mathcal{M}_2, s_2) \in \textit{Mod}(T_{i+1} =$

$T_i \cup \Lambda^{\mathrm{A}}_{\neg l})$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p, w^{\mathrm{A}}_{\neg l}\}, \emptyset\rangle} (\mathcal{M}_2, s_2)$ and vice versa by Proposition 14. ■

For rule **(ERES2)** we have the same result.

Proposition 15 *Let φ be a CTL formula and V be a set of atoms, then we have $T_\varphi \equiv_{\langle V\cup V', \emptyset\rangle} T_\varphi^{r, V\cup V'}$. Where V' be a set of new propositions introduced in the process of transformation.*

Proof: This is apparent from Lemma 11 and Lemma 12. ■

Let φ be a CTL formula, $V \subseteq \mathit{Var}(\varphi)$ be set of atoms and V' be a set of new propositions introduced in the process of translate φ into T_φ. By the transform and resolution rules, we have the following several important properties:

- **(GNA)** for all atom p in $\mathit{Var}(\varphi)$, p do not positively appear in the left hand of the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause;
- **(CNI)** for each global clause, there must be a atom $p \in V'$ appearing in the right hand negatively;
- **(PI)** for each atom $p \in V'$, if p appearing in the left hand of a $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause, then p is positively.

A key point to compute forgetting is eliminate those irrelevant atoms, for this purpose, we define the follow substitution to find out those atoms that do irrelevant. A *instantiate formula* of set V of atoms is a formula that the atoms in V do not appear in it.

Definition 6 *[substitution] Let φ be a CTL formula, $V \subseteq \mathit{Var}(\varphi)$ be a set of atoms and $V' = V'' \subseteq \mathcal{A}$ be a set of new propositions introduced in the process of translating φ into T_φ. Let $\Gamma = (T_\varphi^{r, V\cup V'})'$, then the process of substitution is as follows:*

(i) for each global clause $C = \top \supset D \vee \neg p \in \Gamma$, if there is one and on one atom $p = \mathit{Var}(l) \in V'' \cap \mathit{Var}(D)$ and $\mathit{Var}(D) \cap V = \emptyset$ then let $C = p \supset D$ and $V'' := V'' \setminus \{p\}$;

(ii) find out all the possible instantiate formulae $\varphi_1, ..., \varphi_m$ of $V \cup V''$ in the $p \supset \varphi_i \in \Gamma$ $(1 \leq i \leq m)$;

(iii) if there is $p \supset \varphi_i$ for some $i \in \{1, \dots, m\}$, then let $V'' := V'' \setminus \{p\}$, which means p is a instantiate formula;

(iv) for $\bigwedge_{j=1}^m p_j \supset \varphi_i \in \Gamma$ $(i \in \{1, \dots, m\})$, if there is $\alpha \supset p_1, \dots, \alpha \supset p_m \in \Gamma$ then let $\Gamma' := \Gamma \cup \{\alpha \supset \varphi\}$, if $\Gamma' \neq \Gamma$ then let $\Gamma := \Gamma'$ return to (i);

(v) for $p \supset \varphi_{j_1}, \dots, p \supset \varphi_{j_n} \in \Gamma$, $j_i \in \{1, \dots, m\}$ then replace all p in Γ with $\bigwedge_{i=1}^{j_n} \varphi_{j_i}$, where p do not appear in φ_{j_i}.

*Where p, p_i $(1 \leq i \leq m)$ are atoms and α is a conjunction of literals or **start**.*

We denote this process as $\mathrm{Sub}(\Gamma, V')$. It is apparent that $(T_\varphi^{r, V\cup V'})' \equiv_{\langle V', \emptyset\rangle} \mathrm{Sub}((T_\varphi^{r, V\cup V'})', V')$.

Example 3 Let $\varphi = \mathrm{A}((p \wedge q)\mathrm{U}(f \vee m)) \wedge r$ and $V = \{p\}$. Then we can compute $\mathrm{Sub}((T_\varphi^{r, V\cup V'})', V)$ as follows:

At first, we transform φ into a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ with $V' = \{x, y, z\}$, which is listed as:

1.**start** $\supset z$	2.$\top \supset \neg z \vee r$	3.$\top \supset \neg x \vee f \vee m$
4.$\top \supset \neg z \vee x \vee y$	5.$\top \supset \neg y \vee p$	6.$\top \supset \neg y \vee q$
7.$z \supset \mathrm{AF}x$	8.$y \supset \mathrm{AX}(x \vee y)$.	

In the second, we compute all the possible resolution on $V \cup V'$,

(1)**start** $\supset r$	$(1, 2, SRES5)$
(2)**start** $\supset x \vee y$	$(1, 4, SRES5)$
(3)$\top \supset \neg z \vee y \vee f \vee m$	$(3, 4, SRES8)$
(4)$y \supset \mathrm{AX}(f \vee m \vee y)$	$(3, 8, SRES6)$
(5)$\top \neg z \vee x \vee p$	$(4, 5, SRES8)$
(6)$\top \neg z \vee x \vee q$	$(4, 6, SRES8)$
(7)$y \supset \mathrm{AX}(x \vee p)$	$(5, 7, SRES6)$
(8)$y \supset \mathrm{AX}(x \vee q)$	$(5, 8, SRES6)$
(9)**start** $\supset f \vee m \vee y$	$(3, (2), SRES5)$
(10)**start** $\supset x \vee p$	$(5, (2), SRES5)$
(11)**start** $\supset x \vee q$	$(6, (2), SRES5)$
(12)$\top \supset p \vee \neg z \vee f \vee m$	$(5, (3), SRES8)$
(13)$\top \supset q \vee \neg z \vee f \vee m$	$(6, (3), SRES8)$
(14)$y \supset \mathrm{AX}(p \vee f \vee m)$	$(5, (4), SRES6)$
(15)$y \supset \mathrm{AX}(q \vee f \vee m)$	$(6, (4), SRES6)$
(16)**start** $\supset f \vee m \vee p$	$(5, (9), SRES5)$
(17)**start** $\supset f \vee m \vee q$	$(6, (9), SRES5)$

By the process of substitution we obtain that y is replaced by $q \wedge \mathrm{AX}(p \vee f \vee m)$, x is replaced by $f \vee m$ and z is replaced by r.

By Sub operator, we guarantee those atoms in $V \cup V'$ are really irrelevant atoms. Therefore, we can do the following elimination to eliminate them.

Definition 7 ***(Elimination)*** *Let T be a set of formulae, $C \in T$ and V a set of atoms, then the elimination operator, denoted as Elm, is defined as:*

$$\mathit{Elm}(C, V) = \begin{cases} \top, & \mathit{if}\ \mathit{Var}(C) \cap V \neq \emptyset \\ C, & \mathit{else}. \end{cases}$$

For convenience, we let $\mathit{Elm}(T, V) = \{\mathit{Elm}(r, V) | r \in T\}$.

Proposition 16 *Let φ be a CTL formula, $\Gamma = \mathit{Sub}((T_\varphi^{r, V\cup V'})', V')_{CTL}$ and $V \subseteq \mathit{Var}(\varphi)$ be set of atoms, then we have $\mathit{Elm}(\Gamma, V \cup V') \equiv_{\langle V\cup V', \emptyset\rangle} \Gamma$. Where V' is a set of new propositions introduced in the process of translate φ into T_φ.*

Proof: By the definition of substitution, we know that for each $p \in V'$ if p dose not be substituted by some instantiate formula then delete or add p to a state will not effect the satisfiability of the formula. On one hand, it easy to prove that due to for each Ind-model $(\mathcal{M}, s_0) \in \mathit{Mod}(\mathit{Elm}(\Gamma, V \cup V'))$ we can construct a Ind-model $(\mathcal{M}', s_0')$ that is identical with $(\mathcal{M}, s_0)$ except the for each state $s \in S$, the $L'(s)$ is obtained from $L(s)$ by adding or deleting those atoms in $V \cup V'$. On the other hand, there is $\Gamma \models \mathit{Elm}(\Gamma, V \cup V')$. ■

Corollary 7 *Let φ be CTL formula and $V \subseteq Var(\varphi)$ be a set of proposition, then we have $Elm(Sub((T_{\varphi}^{r,V\cup V'})', V')_{CTL}, V \cup V') \equiv_{\langle V\cup V', I\rangle} \varphi$. Where V' be a set of new propositions introduced in the process of translate φ into T_{φ}, and I be the set of index appearing in T_{φ}.*

In the case that formula dose not include index, we use model structure $\mathcal{M} = (S, R, L, s_0)$ to interpret formula instead of Ind-model structure. Therefore it is apparent that $\forall(\mathcal{M}, s_0) \in Mod(\varphi)$ there is a $(\mathcal{M}', s_0') \in Mod(Elm(\text{Sub}((T_{\varphi}^{r,V\cup V'})', V')_{CTL}, V \cup V'))$ such that $(\mathcal{M}, s_0) \leftrightarrow_{V\cup V'} (\mathcal{M}', s_0')$ from the Corollary 7 and vice versa.

Theorem 8 *Let φ be a CTL formula and $V \subseteq Var(\varphi)$, then we have:*

$$\mathrm{F}_{\mathrm{CTL}}(\varphi, V' \cup V) \equiv Elm(Sub((T_{\varphi}^{r,V\cup V'})', V')_{CTL}, V \cup V').$$

Where V' be a set of new propositions introduced in the process of translate φ into T_{φ}.

Then we have the following result:

Theorem 9 ***(Resolution-based CTL-forgetting)*** *Let φ be a CTL formula and V be a set of atoms. Then*

$$\mathrm{F}_{\mathrm{CTL}}(\varphi, V) \equiv \bigwedge_{\psi \in Elm(Sub((T_{\varphi}^{r,V\cup V'})', V')_{CTL}, V\cup V')} \psi.$$

Where V' is the set of atoms introduced in the process of transformation.

We can obtain that $\mathrm{F}_{\mathrm{CTL}}(\varphi, V) \equiv \mathrm{F}_{\mathrm{CTL}}(\varphi, V' \cup V)$ by Theorem 8, Proposition 8 and Proposition 9, where V' be a set of new propositions introduced in the process of translate φ into T_{φ}. Therefore, the Theorem 9 is proved.

Then we can obtain the result of forgetting of Example 3:

$$\mathrm{F}_{\mathrm{CTL}}(\varphi, \{p\}) \equiv r \wedge (f \vee m \vee q) \wedge$$
$$(f \vee m \vee (q \wedge \mathrm{AX}(f \vee m \vee q))) \wedge \mathrm{AG}((q \wedge \mathrm{AX}(f \vee m \vee q)$$
$$\supset \mathrm{AX}(f \vee m \vee (q \wedge \mathrm{AX}(f \vee m \vee q)))))$$

Given two clauses C and C', we call C and C' are resolvable, the result denote as $res(C, C')$, if there is a resolution rule using C and C' as the premises on some given atom. Then the pseudocode of algorithm resolution-based is as Algorithm 2.

Proposition 17 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$. The time and space complexity of Algorithm 2 are $O((m+1)2^{4(n+n')}$. Where $|Var(\varphi)| = n$, $|V'| = n'$ (V' is set of atoms introduced in transformation) and m is the number of the set Ind of indices introduced during transformation.*

Proof: It follows from that the lines 19-31 of the algorithm, which is to compute all the possible resolution. The possible number of $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clauses under the give V, V' and Ind is $(m+1)2^{4(n+n')} + (m*(n+n')+n+n'+1)2^{2(n+n')+1})$. ■

Algorithm 2: Computing forgetting - A resolution-based method

```
   Input: A CTL formula $\varphi$ and a set $V$ of atoms
   Output: $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$
1  $T = \emptyset$ // the initial set of $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clauses of $\varphi$ ;
2  $T' = \emptyset$ // the set of $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clauses without index;
3  $V' = \emptyset$ // the set of atoms introduced in the process of
   transforming $\varphi$ into $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clauses;
4  $OldT = \{\mathbf{start} \supset z, z \supset \varphi\}$;
5  $V' = \{z\}$;
6  while $OldT \neq T$ do
7      $OldT = T$;
8      $R = \emptyset$;
9      $X = \emptyset$;
10     if Chose a formula $\psi \in OldT$ that dose not a
       $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clause then
11         Using a match rule $Rl$ to transform $\psi$ into a set
           $R$ of $\mathrm{SNF}^{g}_{\mathrm{CTL}}$ clauses;
12         $X$ is the set of atoms introduced by using $Rl$;
13         $V' = V' \cup X$;
14         $T = OldT \setminus \{\psi\} \cup R$;
15     end
16 end
17 $S = \{C | C \in T$ and $Var(C) \cap V = \emptyset\}$;
18 $\Pi = T \setminus S$ ;
19 for ($p \in V \cup V'$) do
20     $\Pi' = \{C \in \Pi | p \in Var(C)\}$ ;
21     $\Sigma = \Pi \setminus \Pi'$;
22     for ($C \in \Pi'$ s.t. $p$ appearing in C positively) do
23         for ($C' \in \Pi'$ s.t. $p$ appearing in $C'$ negatively
           and C, $C'$ are resolvable) do
24             $\Sigma = \Sigma \cup \{res(C, C')\}$;
25             $\Pi' = \Pi' \cup \{C'' = res(C, C') | p \in Var(C'')\}$;
26         end
27     end
28     $\Pi = \Sigma$;
29 end
30 $Res = \Pi \cup S$;
31 return $Elm(\mathrm{Sub}(Res', V')_{CTL}, V \cup V')$.
```

5 Conclusion

6 Appendices Proofs

Proposition 1. **Proof:** ($\Rightarrow$) (a) It is apparent that $L_1(s_1) - V = L_2(s_2) - V$; (b) $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}$ iff $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_i$ for all $i \geq 0$, then for each $(s_1, s_1') \in R_1$, there is a $(s_2, s_2') \in R_2$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}_{i-1}$ for all $i > 0$ and then $L_1(s_1') - V = L_2(s_2') - V$. Therefore, $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}$. (c) This is similar with (b).

($\Leftarrow$) (a) $L_1(s_1) - V = L_2(s_2) - V$ implies that $(s_1, s_2) \in \mathcal{B}_0$; (b) Condition (ii) implies that for every $(s_1, s_1') \in R_1$, there is $(s_2, s_2') \in R_2$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}_i$ for all $i \geq 0$; (c) Condition (iii) implies that for every $(s_2, s_2') \in R_2$, there is $(s_1, s_1') \in R_1$ such that $(\mathcal{K}_1', \mathcal{K}_2') \in \mathcal{B}_i$ for all $i \geq 0$
$\Rightarrow (\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_i$ for all $i \geq 0$
$\Rightarrow (\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}$. ■

Proposition 3. **Proof:** (i) It is clear from Proposition 1.

(ii) Let $\mathcal{M}_i = (S_i, R_i, L_i, s_i)$ $(i = 1, 2, 3)$, $s_1 \leftrightarrow_{V_1} s_2$ via a binary relation $\mathcal{B}$, and $s_2 \leftrightarrow_{V_2} s_3$ via a binary relation $\mathcal{B}''$. Let $\mathcal{B}' \subseteq S_1 \times S_3$ and $\mathcal{B}' = \{(w_1, w_3) | (w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}_2\}$. It's apparent that $(s_1, s_3) \in \mathcal{B}'$. We prove $\mathcal{B}'$ is a $V_1 \cup V_2$-bisimulation between s_1 and s_3 from the three points of Proposition 1 of X-bisimulation (where X is a set of atoms). For all $(w_1, w_3) \in \mathcal{B}'$:

(1) there is $w_2 \in S_2$ such that $(w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''$, and $\forall q \notin V_1$, $q \in L_1(w_1)$ iff $q \in L_2(w_2)$ by $w_1 \leftrightarrow_{V_1} w_2$ and $\forall q' \notin V_2$, $q' \in L_2(w_2)$ iff $q' \in L_3(w_3)$ by $w_2 \leftrightarrow_{V_2} w_3$. Then we have $\forall r \notin V_1 \cup V_2$, $r \in L_1(w_1)$ iff $r \in L_3(w_3)$.

(2) if $(w_1, u_1) \in \mathcal{R}_1$, then $\exists u_2 \in S_2$ such that $(w_2, u_2) \in \mathcal{R}_2$ and $(u_1, u_2) \in \mathcal{B}$ (due to $(w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''$ by the definition of $\mathcal{B}'$); and then $\exists u_3 \in S_3$ such that $(w_3, u_3) \in \mathcal{R}_3$ and $(u_2, u_3) \in \mathcal{B}''$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

(3) if $(w_3, u_3) \in \mathcal{R}_3$, then $\exists u_2 \in S_2$ such that $(w_2, u_2) \in \mathcal{R}_2$ and $(u_2, u_3) \in \mathcal{B}''$; and then $\exists u_1 \in S_1$ such that $(w_1, u_1) \in \mathcal{R}_1$ and $(u_1, u_2) \in \mathcal{B}$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

■

Theorem 1. **Proof:** This theorem can be proved by inducting on the formula φ and supposing $Var(\varphi) \cap V = \emptyset$. Here we only prove the only-if direction. The other direction can be similarly proved.

Case $\varphi = p$ where $p \in \mathcal{A} - V$:
$(\mathcal{M}, s) \models \varphi$ iff $p \in L(s)$ (by the definition of satisfiability)
$\Leftrightarrow p \in L'(s')$ $(s \leftrightarrow_V s')$
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$

Case $\varphi = \neg\psi$:
$(\mathcal{M}, s) \models \varphi$ iff $(\mathcal{M}, s) \not\models \psi$
$\Leftrightarrow (\mathcal{M}', s') \not\models \psi$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$

Case $\varphi = \psi_1 \vee \psi_2$:
$(\mathcal{M}, s) \models \varphi$
$\Leftrightarrow (\mathcal{M}, s) \models \psi_1$ or $(\mathcal{M}, s) \models \psi_2$
$\Leftrightarrow (\mathcal{M}', s') \models \psi_1$ or $(\mathcal{M}', s') \models \psi_2$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$

Case $\varphi = \mathrm{EX}\psi$:
$\mathcal{M}, s \models \varphi$
$\Leftrightarrow$ There is a path $\pi = (s, s_1, ...)$ such that $\mathcal{M}, s_1 \models \psi$
$\Leftrightarrow$ There is a path $\pi' = (s', s_1', ...)$ such that $\pi \leftrightarrow_V \pi'$ ($s \leftrightarrow_V s'$, Proposition 3)
$\Leftrightarrow s_1 \leftrightarrow_V s_1'$ $(\pi \leftrightarrow_V \pi')$
$\Leftrightarrow (\mathcal{M}', s_1') \models \psi$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$

Case $\varphi = \mathrm{EG}\psi$:
$\mathcal{M}, s \models \varphi$
$\Leftrightarrow$ There is a path $\pi = (s = s_0, s_1, ...)$ such that for each $i \geq 0$ there is $(\mathcal{M}, s_i) \models \psi$
$\Leftrightarrow$ There is a path $\pi' = (s' = s_0', s_1', ...)$ such that $\pi \leftrightarrow_V \pi'$ ($s \leftrightarrow_V s'$, Proposition 3)
$\Leftrightarrow s_i \leftrightarrow_V s_i'$ for each $i \geq 0$ $(\pi \leftrightarrow_V \pi')$
$\Leftrightarrow (\mathcal{M}', s_i') \models \psi$ for each $i \geq 0$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$

Case $\varphi = \mathrm{E}[\psi_1 \mathrm{U} \psi_2]$:
$\mathcal{M}, s \models \varphi$
$\Leftrightarrow$ There is a path $\pi = (s = s_0, s_1, ...)$ such that there is $i \geq 0$ such that $(\mathcal{M}, s_i) \models \psi_2$, and for all $0 \leq j < i$, $(\mathcal{M}, s_j) \models \psi_1$
$\Leftrightarrow$ There is a path $\pi' = (s = s_0', s_1', ...)$ such that $\pi \leftrightarrow_V \pi'$ ($s \leftrightarrow_V s'$, Proposition 3)
$\Leftrightarrow (\mathcal{M}', s_i') \models \psi_2$, and for all $0 \leq j < i$ $(\mathcal{M}', s_j') \models \psi_1$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \varphi$ ■

Lemma5 Proof: This result can be proved by inducting on n.

Base. It is apparent that for any $s_n \in S$ and $s_n' \in S'$, if $\mathrm{Tr}_0(s_n) \leftrightarrow_{\overline{V}} \mathrm{Tr}_0(s_n')$ then $\mathcal{F}_V(\mathrm{Tr}_0(s_n)) \equiv \mathcal{F}_V(\mathrm{Tr}_0(s_n'))$ due to $L(s_n) - \overline{V} = L'(s_n') - \overline{V}$ by known.

Step. Supposing that for $k = m$ $(0 < m \leq n)$ there is if $\mathrm{Tr}_{n-k}(s_k) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k}(s_k')$ then $\mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k)) \equiv \mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k'))$, then we will show if $\mathrm{Tr}_{n-k+1}(s_{k-1}) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k+1}(s_{k-1}')$ then $\mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1})) \equiv \mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1}'))$. Apparent that:

$$\mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1})) = \left(\bigwedge_{(s_{k-1}, s_k) \in R} \mathrm{EX}\mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k))\right) \wedge \mathrm{AX}\left(\bigvee_{(s_{k-1}, s_k) \in R} \mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k))\right) \wedge \mathcal{F}_V(\mathrm{Tr}_0(s_{k-1}))$$

$$\mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1}')) = \left(\bigwedge_{(s_{k-1}', s_k') \in R} \mathrm{EX}\mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k'))\right) \wedge \mathrm{AX}\left(\bigvee_{(s_{k-1}', s_k') \in R} \mathcal{F}_V(\mathrm{Tr}_{n-k}(s_k'))\right) \wedge \mathcal{F}_V(\mathrm{Tr}_0(s_{k-1}'))$$

by the definition of characterize formula of the computation tree. Then we have for any $(s_{k-1}, s_k) \in R$ there is $(s_{k-1}', s_k') \in R'$ such that $\mathrm{Tr}_{n-k}(s_k) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k}(s_k')$ by $\mathrm{Tr}_{n-k+1}(s_{k-1}) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k+1}(s_{k-1}')$. Besides, for any $(s_{k-1}', s_k') \in R'$ there is $(s_{k-1}, s_k) \in R$ such that $\mathrm{Tr}_{n-k}(s_k) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k}(s_k')$ by $\mathrm{Tr}_{n-k+1}(s_{k-1}) \leftrightarrow_{\overline{V}} \mathrm{Tr}_{n-k+1}(s_{k-1}')$. Therefore, we have $\mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1})) \equiv \mathcal{F}_V(\mathrm{Tr}_{n-k+1}(s_{k-1}'))$ by induction hypothesis. ■

Theorem 2. **Proof:**

Let $\mathcal{F}_V(\mathcal{M}, s_0)$ be the characterizing formula of $(\mathcal{M}, s_0)$ on V. It is apparent that IR$(\mathcal{F}_V(\mathcal{M}, s_0), \overline{V})$. We will show that $(\mathcal{M}, s_0) \models \mathcal{F}_V(\mathcal{M}, s_0)$ at first.

It is apparent that $(\mathcal{M}, s_0) \models \mathcal{F}_V(\text{Tr}_c(s_0))$ by Lemma 3. We must show that $(\mathcal{M}, s_0) \models \bigwedge_{s \in S} G(\mathcal{M}, s)$. Let $\mathcal{X} = \mathcal{F}_V(\text{Tr}_c(s)) \rightarrow \left(\bigwedge_{(s,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))\right)$ $\wedge \text{AX}\left(\bigvee_{(s,s_1)\in R} \mathcal{F}_V(\text{Tr}_c(s_1))\right)$, we will show $\forall s \in S$, $(\mathcal{M}, s_0) \models G(\mathcal{M}, s)$. Where $G(\mathcal{M}, s) = \text{AG}\mathcal{X}$. There are two cases we should consider:

- If $(\mathcal{M}, s_0) \not\models \mathcal{F}_V(\text{Tr}_c(s))$, it is apparent that $(\mathcal{M}, s_0) \models \mathcal{X}$;
- If $(\mathcal{M}, s_0) \models \mathcal{F}_V(\text{Tr}_c(s))$:
  $(\mathcal{M}, s_0) \models \mathcal{F}_V(\text{Tr}_c(s))$
  $\Rightarrow s_0 \leftrightarrow_{\overline{V}} s$ by the definition of characteristic number and Lemma 4
  for each $(s, s_1) \in R$ there is $(\mathcal{M}, s_1) \models \mathcal{F}_V(\text{Tr}_c(s_1))$ $(s_1 \leftrightarrow_{\overline{V}} s_1)$
  $\Rightarrow (\mathcal{M}, s) \models \bigwedge_{(s,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))$
  $\Rightarrow (\mathcal{M}, s_0) \models \bigwedge_{(s,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))$ (IR$(\bigwedge_{(s,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1)), \overline{V})$, $s_0 \leftrightarrow_{\overline{V}} s$)
  for each (s, s_1) there is $\mathcal{M}, s_1 \models \bigvee_{(s,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))$
  $\Rightarrow (\mathcal{M}, s) \models \text{AX}\left(\bigvee_{(s,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))\right)$
  $\Rightarrow (\mathcal{M}, s_0) \models \text{AX}\left(\bigvee_{(s,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))\right)$ (IR$(\text{AX}\left(\bigvee_{(s,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))\right), \overline{V})$, $s_0 \leftrightarrow_{\overline{V}} s$)
  $\Rightarrow (\mathcal{M}, s_0) \models \mathcal{X}$.

For any other states s' which can reach from s_0 can be proved similarly, *i.e.* , $(\mathcal{M}, s') \models \mathcal{X}$. Therefore, $\forall s \in S$, $(\mathcal{M}, s_0) \models G(\mathcal{M}, s)$, and then $(\mathcal{M}, s_0) \models \mathcal{F}_V(\mathcal{M}, s_0)$.

We will prove this theorem from the following two aspects:

$(\Leftarrow)$ If $s_0 \leftrightarrow_{\overline{V}} s'_0$, then $(\mathcal{M}', s'_0) \models \mathcal{F}_V(M, s_0)$. S-ince $(\mathcal{M}, s_0) \models \mathcal{F}_V(\mathcal{M}, s_0)$ and IR$(\mathcal{F}_V(\mathcal{M}, s_0), \overline{V})$, hence $(\mathcal{M}', s'_0) \models \mathcal{F}_V(M, s_0)$ by Theorem 1.

$(\Rightarrow)$ If $(\mathcal{M}', s'_0) \models \mathcal{F}_V(M, s_0)$, then $s_0 \leftrightarrow_{\overline{V}} s'_0$. We will prove this by showing that $\forall n \geq 0$, $Tr_n(s_0) \leftrightarrow_{\overline{V}} Tr_n(s'_0)$.

Base. It is apparent that $Tr_0(s_0) \equiv Tr_0(s'_0)$.

Step. Supposing $\text{Tr}_k(s_0) \leftrightarrow_{\overline{V}} \text{Tr}_k(s'_0)$ $(k > 0)$, we will prove $\text{Tr}_{k+1}(s_0) \leftrightarrow_{\overline{V}} \text{Tr}_{k+1}(s'_0)$. We should only show that $\text{Tr}_1(s_k) \leftrightarrow_{\overline{V}} \text{Tr}_1(s'_k)$. Where $(s_0, s_1), (s_1, s_2), \ldots, (s_{k-1}, s_k) \in R$ and $(s'_0, s'_1), (s'_1, s'_2), \ldots, (s'_{k-1}, s'_k) \in R'$, *i.e.* s_{i+1} (s'_{i+1}) is an immediate successor of s_i (s'_i) for all $0 \leq i \leq k-1$.

(i) It is apparent that $L(s_k) - \overline{V} = L'(s'_k) - \overline{V}$ by inductive assumption.

Before talking about the other points, note the following fact that:
$(\mathcal{M}', s'_0) \models \mathcal{F}_V(\mathcal{M}, s_0)$
$\Rightarrow \forall s' \in S'$, $(\mathcal{M}', s') \models \mathcal{F}_V(\text{Tr}_c(s)) \rightarrow \left(\bigwedge_{(s,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))\right) \wedge$ $\text{AX}\left(\bigvee_{(s,s_1)\in R} \mathcal{F}_V(\text{Tr}_c(s_1))\right)$ for any $s \in S$. **(fact)**
(I) $(\mathcal{M}', s'_0) \models \mathcal{F}_V(\text{Tr}_c(s_0)) \rightarrow \left(\bigwedge_{(s_0,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))\right) \wedge$ $\text{AX}\left(\bigvee_{(s_0,s_1)\in R} \mathcal{F}_V(\text{Tr}_c(s_1))\right)$ **(fact)**
(II) $(\mathcal{M}', s'_0) \models \mathcal{F}_V(\text{Tr}_c(s_0)))$ (known)
(III) $(\mathcal{M}', s'_0) \models \left(\bigwedge_{(s_0,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))\right) \wedge$ $\text{AX}\left(\bigvee_{(s_0,s_1)\in R} \mathcal{F}_V(\text{Tr}_c(s_1))\right)$ ((I),(II))

(ii) We will show that for each $(s_k, s_{k+1}) \in R$ there is a $(s'_k, s'_{k+1}) \in R'$ such that $L(s_{k+1}) - \overline{V} = L'(s'_{k+1}) - \overline{V}$.
(1) $(\mathcal{M}', s'_0) \models \bigwedge_{(s_0,s_1)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_1))$ (III)
(2) $\forall (s_0, s_1) \in R$, $\exists (s'_0, s'_1) \in R'$ $(\mathcal{M}', s'_1) \models \mathcal{F}_V(\text{Tr}_c(s_1))$ (2)
(3) $\text{Tr}_c(s_1) \leftrightarrow_{\overline{V}} \text{Tr}_c(s'_1)$ ((2), Lemma 3)
(4) $L(s_1) - \overline{V} = L'(s'_1) - \overline{V}$ ((3), $c \geq 0$)
(5) $(\mathcal{M}', s'_1) \models \mathcal{F}_V(\text{Tr}_c(s_1)) \rightarrow \left(\bigwedge_{(s_1,s_2)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_2))\right) \wedge$ $\text{AX}\left(\bigvee_{(s_1,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))\right)$ **(fact)**
(6) $(\mathcal{M}', s'_1) \models \left(\bigwedge_{(s_1,s_2)\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_2))\right) \wedge$ $\text{AX}\left(\bigvee_{(s_1,s_2)\in R} \mathcal{F}_V(\text{Tr}_c(s_2))\right)$ ((2), (5))
(7) . . . . . .
(8) $(\mathcal{M}', s'_k) \models \left(\bigwedge_{(s_k,s_{k+1})\in R} \text{EX}\mathcal{F}_V(\text{Tr}_c(s_{k+1}))\right) \wedge$ $\text{AX}\left(\bigvee_{(s_k,s_{k+1})\in R} \mathcal{F}_V(\text{Tr}_c(s_{k+1}))\right)$ (similar with (6))
(9) $\forall (s_k, s_{k+1}) \in R$, $\exists (s'_k, s'_{k+1}) \in R'$ $(\mathcal{M}', s'_{k+1}) \models \mathcal{F}_V(\text{Tr}_c(s_{k+1}))$ (8)
(10) $\text{Tr}_c(s_{k+1}) \leftrightarrow_{\overline{V}} \text{Tr}_c(s'_{k+1})$ ((9), Lemma 3)
(11) $L(s_{k+1}) - \overline{V} = L'(s'_{k+1}) - \overline{V}$ ((10), $c \geq 0$)

(iii) We will show that for each $(s'_k, s'_{k+1}) \in R'$ there is a $(s_k, s_{k+1}) \in R$ such that $L(s_{k+1}) - \overline{V} = L'(s'_{k+1}) - \overline{V}$.
(1) $(\mathcal{M}', s'_k) \models \text{AX}\left(\bigvee_{(s_k,s_{k+1})\in R} \mathcal{F}_V(\text{Tr}_c(s_{k+1}))\right)$ (by (8) talked above)
(2) $\forall (s'_k, s'_{k+1}) \in R'$, $\exists (s_k, s_{k+1}) \in R$ $(\mathcal{M}', s'_{k+1}) \models \mathcal{F}_V(\text{Tr}_c(s'_{k+1}))$ (1)
(3) $\text{Tr}_c(s_{k+1}) \leftrightarrow_{\overline{V}} \text{Tr}_c(s'_{k+1})$ ((2), Lemma 3)
(4) $L(s_{k+1}) - \overline{V} = L'(s'_{k+1}) - \overline{V}$ ((3), $c \geq 0$)

■

Theorem 4. **Proof:** $(i) \Leftrightarrow (ii)$. To prove this, we will show that:

$$Mod(\text{F}_{\text{CTL}}(\varphi, V)) = Mod(\{\phi | \varphi \models \phi, \text{IR}(\phi, V)\})$$
$$= Mod(\bigvee_{\mathcal{M}, s_0 \in Mod(\varphi)} \mathcal{F}_{\mathcal{A}\setminus V}(\mathcal{M}, s_0)).$$

Firstly, suppose that $(\mathcal{M}', s'_0)$ is a model of $\text{F}_{\text{CTL}}(\varphi, V)$. Then there exists an an initial K-structure $(\mathcal{M}, s_0)$ such that $(\mathcal{M}, s_0)$ is a model of φ and $(\mathcal{M}, s_0) \leftrightarrow_V (\mathcal{M}', s'_0)$. By Theorem 1, we have $(\mathcal{M}', s'_0) \models \phi$ for all ϕ that $\varphi \models$

ϕ and IR(ϕ, V). Thus, $(\mathcal{M}', s_0')$ is a model of $\{\phi|\varphi \models \phi, \text{IR}(\phi, V)\}$.

Secondly, suppose that $(\mathcal{M}', s_0')$ is a models of $\{\phi|\varphi \models \phi, \text{IR}(\phi, V)\}$. Thus, $(\mathcal{M}', s_0') \models \bigvee_{(\mathcal{M},s_0)\in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}\setminus V}(\mathcal{M}, s_0)$ due to $\bigvee_{(\mathcal{M},s_0)\in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}\setminus V}(\mathcal{M}, s_0)$ is irrelevant to V.

Finally, suppose that $(\mathcal{M}', s_0')$ is a model of $\bigvee_{\mathcal{M},s_0\in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}\setminus V}(\mathcal{M}, s_0)$. Then there exists $(\mathcal{M}, s_0) \in \mathit{Mod}(\varphi)$ such that $(\mathcal{M}', s_0') \models \mathcal{F}_{\mathcal{A}\setminus V}(\mathcal{M}, s_0)$. Hence, $(\mathcal{M}, s_0) \leftrightarrow_V (\mathcal{M}', s_0')$ by Theorem 2. Thus $(\mathcal{M}', s_0')$ is also a model of $\text{F}_{\text{CTL}}(\varphi, V)$.

$(ii) \Rightarrow (iii)$. It is not difficult to prove it.

$(iii) \Rightarrow (ii)$. Suppose that all postulates hold. By Positive Persistence, we have $\psi \models \{\phi|\varphi \models \phi, \text{IR}(\phi, V)\}$. Now we show that $\{\phi|\varphi \models \phi, \text{IR}(\phi, V)\} \models \psi$. Otherwise, there exists formula ϕ' such that $\psi \models \phi'$ but $\{\phi|\varphi \models \phi, \text{IR}(\phi, V)\} \not\models \phi'$. There are three cases:

- ϕ' is relevant to V. Thus, ψ is also relevant to V, a contradiction to Irrelevance.
- ϕ' is irrelevant to V and $\varphi \models \phi'$. This contradicts to our assumption.
- ϕ' is irrelevant to V and $\varphi \not\models \phi'$. By Negative Persistence, $\psi \not\models \phi'$, a contradiction.

Thus, ψ is equivalent to $\{\phi|\varphi \models \phi, \text{IR}(\phi, V)\}$. ■

Lemma 8 **Proof:** $(\Rightarrow)$ $\forall \mathcal{M} = \langle S, R, L, [\_], s_0\rangle$, $(\mathcal{M}, s_0) \models T$
$\Rightarrow \forall C \in T$ there is $(\mathcal{M}, s_0) \models C$
$\Rightarrow \forall C' \in T'$ there are two aspects:

1. If C' is a clause from T that without index, then it is apparent that $(\mathcal{M}, s_0) \models C'$;
2. If C' is a clause from $C \in T$ by removing the index in C:
   (a) If C is an E-step clause, *i.e.* $C = \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k}(m_j)_{\langle ind\rangle})$ and then $C' = \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k} m_j)$.
   $\mathcal{M}, s_0 \models C$
   $\Rightarrow \forall \pi = (s_0, s_1, \dots)$ and s_m $(m \leq 0)$ there is $(\mathcal{M}, s_m) \models \bigwedge_{i=1}^{l} l_i \supset \text{E}_{\langle ind\rangle}\text{X}\bigvee_{j=1}^{k} m_j$
   If $(\mathcal{M}, s_m) \not\models \bigwedge_{i=1}^{l} l_i$, it is apparent that $(\mathcal{M}, s_m) \models \bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k}(m_j)$
   If $(\mathcal{M}, s_m) \models \bigwedge_{i=1}^{l} l_i$ then there is $(s_m, s_{m+1}) \in$ [ind] $\subseteq R\, \mathcal{M}, s_{m+1} \models \bigvee_{j=1}^{k} m_j$
   $\Rightarrow \mathcal{M}, s_m \models \bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k} m_j$
   Therefore, $(\mathcal{M}, s_0) \models C'$.
   (b) If C is an E-sometime clause, this can be proved similarly.

$(\Leftarrow)$ $\forall \mathcal{M} = (S, R, L, [\_], s_0)$, $(\mathcal{M}, s_0) \models T'$, we can construct a Ind-model structure $\mathcal{M}' = (S, R, L, [\_]', s_0)$ as follows:

1. if $(\mathcal{M}, s_0) \models C' = \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k} m_j)$, then for each state s' in S there are two cases: if $(\mathcal{M}, s') \models \neg(\bigwedge_{i=1}^{l} l_i)$ then it is apparent $(\mathcal{M}', s_0) \models C'$, we do not need two change the $[\_]$; if $(\mathcal{M}, s') \models \bigwedge_{i=1}^{l} l_i$ then there is a path $\pi = (s', s_1, \dots)$ s.t. $(\mathcal{M}, s_1) \models \bigvee_{j=1}^{k} m_j$, in this way, we let $R_{s'} = \{(s', s_1), (s_1, s_2), \dots\}$, $R_y = \{(s_x, s_y)|\forall s_x \in S$ if $\forall (s_1, s_2) \in \bigcup_{s\in S} R_s, s_2 \neq s_x$ then find a unique $s_y \in S$ such that $(s_x, s_y) \in R\}$ and $[ind]' = \bigcup_{s\in S} R_s \cup R_y$. Therefore, $(\mathcal{M}, s_0) \leftrightarrow_{\langle \emptyset, \{ind\}\rangle} (\mathcal{M}', s_0')$ and $(\mathcal{M}', s_0) \models \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EX}\bigvee_{j=1}^{k}(m_j)_{\langle ind\rangle})$.
2. if $(\mathcal{M}, s_0) \models C' = \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EF}\bigvee_{j=1}^{k} m_j)$, then for each state s' in S there are two cases: if $(\mathcal{M}, s') \models \neg(\bigwedge_{i=1}^{l} l_i)$ then it is apparent $(\mathcal{M}', s_0) \models C'$, we do not need two change the $[\_]$; if $(\mathcal{M}, s') \models \bigwedge_{i=1}^{l} l_i$ then there is a path $\pi = (s' = s^0, s^1, \dots)$ s.t. for some state s^i $(i \geq 0)$ in this path $(\mathcal{M}, s^i) \models \bigvee_{j=1}^{k} m_j$, in this way, we can construct the $[ind]'$ similarly as 1. Therefore, $(\mathcal{M}, s_0) \leftrightarrow_{\emptyset, \{ind\}} (\mathcal{M}', s_0')$ and $(\mathcal{M}', s_0) \models \text{AG}(\bigwedge_{i=1}^{l} l_i \supset \text{EF}\bigvee_{j=1}^{k}(m_j)_{\langle ind\rangle})$.

■

Lemma 10 **Proof:** We will prove this result in $t \in$ {Trans(1), Trans(4), Trans(6)}, other cases can be proved similarly.

(1) For t=Trans(1):
$(\Rightarrow)$ $\forall (\mathcal{M}_1, s_1) \in \mathit{Mod}(T_i)$ *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{EX}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and for every π starting from s_1 and every state $s_1^j \in \pi$, $(\mathcal{M}, s_1^j) \models \neg q$ or there exists a path π' starting from s_1^j, there exists a state s' such that $(s_1^j, s_1^{j+1}) \in R_1$ and $(\mathcal{M}, s_1^{j+1}) \models \varphi$
We can construct an Ind-model structure $\mathcal{M}_2$ is identical to $\mathcal{M}_1$ except $[ind]' = \bigcup_{s\in S} R_s \cup R_y$, where $R_{s'} = \{(s_1^j, s_1^{j+1}), (s_1^{j+1}, s_1^{j+2}), \dots\}$ and $R_y = \{(s_x, s_y)|\forall s_x \in S$ if $\forall (s_1', s_2') \in \bigcup_{s\in S} R_s, s_2' \neq s_x$ then find a unique $s_y \in S$ such that $(s_x, s_y) \in R\}$. It is apparent that $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \emptyset, \{ind\}\rangle} (\mathcal{M}_2, s_2)$ (let $s_2 = s_1$).
$\Rightarrow$ for every path starting from s_1 and every state s_1^j in this path, $(\mathcal{M}_2, s_1^j) \models \neg q$ or $(\mathcal{M}_2, s_1^j) \models \text{EX}\varphi_{\langle ind\rangle}$ (by the semantic of EX)
$\Rightarrow (\mathcal{M}_2, s_1) \models \text{AG}(q \supset \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow (\mathcal{M}_2, s_1) \models X \wedge \text{AG}(q \supset \text{E}_{\langle ind\rangle}\text{X}\varphi)$

$(\Leftarrow)$ $\forall (\mathcal{M}_1, s_1) \in \mathit{Mod}(T_{i+1})$ *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and $(\mathcal{M}_1, s_1) \models \text{AG}(q \supset \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow$ for every path starting from s_1 and every state s_1^j in this path, $(\mathcal{M}_1, s_1^j) \models \neg q$ or there exits a state s' such that $(s_1^j, s') \in [ind]$ and $(\mathcal{M}_1, s') \models \varphi$ (by the semantic of $\text{E}_{\langle ind\rangle}\text{X}$)
$\Rightarrow$ for every path starting from s_1 and every state s_1^j in this path, $(\mathcal{M}_1, s_1^j) \models \neg q$ or $(\mathcal{M}_1, s_1^j) \models \text{EX}\varphi$ (by the semantic of EX)
$\Rightarrow (\mathcal{M}_1, s_1) \models \text{AG}(q \supset \text{EX}\varphi)$

$\Rightarrow (\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{EX}\varphi)$
It is apparent that $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}_1, s_1)$.

(2) For t=Trans(4):
($\Rightarrow$) $\forall(\mathcal{M}_1, s_1) \in Mod(T_i)$, *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \varphi_1 \vee \varphi_2)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and $\forall s_1' \in S, (\mathcal{M}_1, s_1') \models q \supset \varphi_1 \vee \varphi_2$
$\Rightarrow (\mathcal{M}_1, s_1') \models \neg q$ or $(\mathcal{M}_1, s_1') \models \varphi_1 \vee \varphi_2$
The we can construct an Ind-model structure $\mathcal{M}_2$ as follows. $\mathcal{M}_2$ is the same with $\mathcal{M}_1$ when $(\mathcal{M}_1, s_1') \models \neg q$. When $(\mathcal{M}_1, s_1') \models q$, $\mathcal{M}_2$ is identical to $\mathcal{M}_1$ except if $(\mathcal{M}_1, s_1') \models \varphi_1$ then $L_2(s_1') = L_1(s_1')$ else $L_2(s_1') = L_1(s_1') \cup \{p\}$. It is apparent that $(\mathcal{M}_2, s_1') \models (q \supset \varphi_1 \vee p) \wedge (p \supset \varphi_2)$, then $(\mathcal{M}_2, s_1) \models T_{i+1}$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset \rangle} (\mathcal{M}_2, s_2)$.

($\Leftarrow$) $\forall(\mathcal{M}_1, s_1) \in Mod(T_{i+1})$, *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \varphi_1 \vee p) \wedge \text{AG}(p \supset \varphi_2)$. It is apparent that $(\mathcal{M}_1, s_1) \models T_i$.

(3) For t=Trans(6):
We prove for $\text{EX}_{\langle ind \rangle}$, while for the AX can be proved similarly.

($\Rightarrow$) $\forall(\mathcal{M}_1, s_1) \in Mod(T_i)$, *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{EX}\varphi_{\langle ind \rangle})$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and $\forall s_1' \in S, (\mathcal{M}_1, s_1') \models q \supset \text{EX}\varphi_{\langle ind \rangle}$
$\Rightarrow (\mathcal{M}_1, s_1') \models \neg q$ or there exists a state s' such that $(s_1', s') \in [ind]$ and $(\mathcal{M}_1, s') \models \varphi$
We can construct an Ind-model structure $\mathcal{M}_2$ as follows. $\mathcal{M}_2$ is the same with $\mathcal{M}_1$ when $(\mathcal{M}_1, s_1') \models \neg q$. When $(\mathcal{M}_1, s_1') \models q$, $\mathcal{M}_2$ is identical to $\mathcal{M}_1$ except for s' there is $L_2(s') = L_1(s') \cup \{p\}$. It is apparent that $(\mathcal{M}_2, s_1) \models \text{AG}(q \supset \text{EX}p) \wedge \text{AG}(p \supset \varphi)$, $(\mathcal{M}_2, s_2) \models T_{i+1}$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset \rangle} (\mathcal{M}_2, s_2)$ $(s_2 = s_1)$.

($\Leftarrow$) $\forall(\mathcal{M}_1, s_1) \in Mod(T_{i+1})$, *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \varphi_1 \vee p) \wedge \text{AG}(p \supset \varphi_2)$. It is apparent that $(\mathcal{M}_1, s_1) \models T_i$. ■

Theorem8 **Proof:** ($\Rightarrow$) $\forall(\mathcal{M}, s_0) \in Mod(\text{F}_{\text{CTL}}(\varphi, V' \cup V))$
$\Rightarrow \exists(\mathcal{M}', s_0') \in Mod(\varphi)$ s.t. $(\mathcal{M}, s_0) \leftrightarrow_{V' \cup V} (\mathcal{M}', s_0')$
$\Rightarrow \exists(\mathcal{M}_1, s_1) \in Mod(Elm(\text{Sub}((T_\varphi^{r, V \cup V'})', V)_{CTL}, V \cup V'))$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{V' \cup V} (\mathcal{M}', s_0')$
$\Rightarrow (\mathcal{M}, s_0) \leftrightarrow_{V' \cup V} (\mathcal{M}_1, s_1)$
$\Rightarrow (\mathcal{M}, s_0) \models Elm(\text{Sub}((T_\varphi^{r, V \cup V'})', V)_{CTL}, V \cup V')$ (IR($Elm(\text{Sub}((T_\varphi^{r, V \cup V'})', V)_{CTL}, V \cup V'), V' \cup V$))

($\Leftarrow$)$\forall(\mathcal{M}_1, s_1) \in Mod(Elm(\text{Sub}((T_\varphi^{r, V \cup V'})', V)_{CTL}, V \cup V'))$
$\Rightarrow \exists(\mathcal{M}', s_0') \in Mod(\varphi)$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{V' \cup V} (\mathcal{M}', s_0')$
$\Rightarrow (\mathcal{M}_1, s_1) \models \text{F}_{\text{CTL}}(\varphi, V' \cup V)$ (IR($\text{F}_{\text{CTL}}(\varphi, V' \cup V), V \cup V'$) and $\varphi \models \text{F}_{\text{CTL}}(\varphi, V' \cup V)$) ■

References

[Abiteboul *et al.*, 1996] Serge Abiteboul, Laurent Herr, and J Bussche. Temporal versus first-order logic to query temporal databases. Technical report, Stanford InfoLab, 1996.

[Baader and Ohlbach, 1995] Franz Baader and Hans Juürgen Ohlbach. A multi-dimensional terminological knowledge representation language. *Journal of Applied Non-Classical Logics*, 5(2):153–197, 1995.

[Baier and Katoen, 2008] Christel Baier and JoostPieter Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[Benthem, 1977] van J. F. A. K Benthem. Modal correspondence theory. 1977.

[Bienvenu *et al.*, 2010] Meghyn Bienvenu, Hélene Fargier, and Pierre Marquis. Knowledge compilation in the modal logic s5. In *Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, Usa, July*, pages 261–266, 2010.

[Bobrow *et al.*, 1997] Daniel G Bobrow, Devika Subramanian, Russell Greiner, and Judea Pearl. Special issue on relevance 97 (1-2). *Artificial Intelligence Journal*, 1997.

[Bolotov, 1999] Alexander Bolotov. A clausal resolution method for ctl branching-time temporal logic. *Journal of Experimental & Theoretical Artificial Intelligence*, 11(1):77–93, 1999.

[Bolotov, 2000] Alexander Bolotov. *Clausal resolution for branching-time temporal logic*. PhD thesis, Manchester Metropolitan University, 2000.

[Browne *et al.*, 1988] Michael C. Browne, Edmund M. Clarke, and Orna Grumberg. Characterizing finite kripke structures in propositional temporal logic. *Theor. Comput. Sci.*, 59:115–131, 1988.

[Chaki *et al.*, 2004] Sagar Chaki, Edmund Clarke, Orna Grumberg, Joël Ouaknine, Natasha Sharygina, Tayssir Touili, and Helmut Veith. An expressive verification framework for state/event systems. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE, 2004.

[Chaki *et al.*, 2005] Sagar Chaki, Edmund Clarke, Joël Ouaknine, Natasha Sharygina, and Nishant Sinha. Concurrent software verification with states, events, and deadlocks. *Formal Aspects of Computing*, 17(4):461–483, 2005.

[Chomicki and Niwinski, 1995] Jan Chomicki and Damian Niwinski. On the feasibility of checking temporal integrity constraints. *Journal of Computer and System Sciences*, 51(3):523–535, 1995.

[Chomicki, 1994] Jan Chomicki. Temporal query languages: a survey. In *International Conference on Temporal Logic*, pages 506–534. Springer, 1994.

[Clarke and Emerson, 1981] Edmund M Clarke and E Allen Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pages 52–71. Springer, 1981.

[Clarke *et al.*, 1986] Edmund M. Clarke, E. Allen Emerson, and A. Prasad Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, 8(2):244–263, 1986.

[Clarke *et al.*, 1994] Edmund M Clarke, Orna Grumberg, and David E Long. Model checking and abstraction. *ACM transactions on Programming Languages and Systems (TOPLAS)*, 16(5):1512–1542, 1994.

[Clarke *et al.*, 2018] Edmund M. Clarke, Thomas A. Henzinger, and Helmut Veith. Introduction to model checking. In *Handbook of Model Checking.* [2018], pages 1–26.

[Delgrande, 2017] James P. Delgrande. A knowledge level account of forgetting. *J. Artif. Intell. Res.*, 60:1165–1213, 2017.

[Eiter and Wang, 2008] Thomas Eiter and Kewen Wang. *Semantic forgetting in answer set programming*. Elsevier Science Publishers Ltd., 2008.

[Emerson and Halpern, 1986] E Allen Emerson and Joseph Y Halpern. "sometimes" and "not never" revisited: on branching versus linear time temporal logic. *Journal of the ACM (JACM)*, 33(1):151–178, 1986.

[Emerson, 1990] E Allen Emerson. Temporal and modal logic. In *Formal Models and Semantics*, pages 995–1072. Elsevier, 1990.

[Fagin *et al.*, 2004] Ronald Fagin, Joseph Y Halpern, Yoram Moses, and Moshe Vardi. *Reasoning about knowledge*. MIT press, 2004.

[Fang *et al.*, 2019] Liangda Fang, Yongmei Liu, and Hans Van Ditmarsch. Forgetting in multi-agent modal logics. *Artificial Intelligence*, 266:51–80, 2019.

[Ghosh *et al.*, 2016] Shalini Ghosh, Daniel Elenius, Wenchao Li, Patrick Lincoln, Natarajan Shankar, and Wilfried Steiner. Arsenal: automatic requirements specification extraction from natural language. In *NASA Formal Methods Symposium*, pages 41–46. Springer, 2016.

[Hodkinson *et al.*, 2000] Ian Hodkinson, Frank Wolter, and Michael Zakharyaschev. Decidable fragments of first-order temporal logics. *Annals of Pure and Applied logic*, 106(1-3):85–134, 2000.

[Konev *et al.*, 2009] Boris Konev, Dirk Walther, and Frank Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Twenty-First International Joint Conference on Artificial Intelligence*, 2009.

[Lang and Marquis, 2008] Jerome Lang and Pierre Marquis. On propositional definability. *Artificial Intelligence*, 172(8):991–1017, 2008.

[Lang and Marquis, 2010] Jerome Lang and Pierre Marquis. *Reasoning under inconsistency: a forgetting-based approach*. Elsevier Science Publishers Ltd, 2010.

[Laroussinie and Markey, 2014] François Laroussinie and Nicolas Markey. Quantified ctl: expressiveness and complexity. *arXiv preprint arXiv:1411.4332*, 2014.

[Lin and Reiter, 1994] Fangzhen Lin and Ray Reiter. Forget it. In *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159, 1994.

[Lin, 2001] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, 128(1):143–159, 2001.

[Lin, 2003] Fangzhen Lin. Compiling causal theories to successor state axioms and strips-like systems. *Journal of Artificial Intelligence Research*, 19:279–314, 2003.

[Liu and Wen, 2011] Yongmei Liu and Ximing Wen. On the progression of knowledge in the situation calculus. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 976–982, Barcelona, Catalonia, Spain, 2011. IJCAI/AAAI.

[Lutz and Wolter, 2011] Carsten Lutz and Frank Wolter. Foundations for uniform interpolation and forgetting in expressive description logics. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 989–995, Barcelona, Catalonia, Spain, 2011. IJCAI/AAAI.

[Manna and Pnueli, 1995] Zohar Manna and Amir Pnueli. *Temporal Verification of Reactive Systems: Safety*, volume 2. Springer Science & Business Media, 1995.

[Marquis, 2003] Pierre Marquis. *Propositional independence: formula-variable independence and forgetting*. AI Access Foundation, 2003.

[Pnueli and Manna, 1992] Amir Pnueli and Zohar Manna. The temporal logic of reactive and concurrent systems. *Springer*, 16:12, 1992.

[Pnueli, 1977] Amir Pnueli. The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, pages 46–57. IEEE, 1977.

[Pnueli, 1986] Amir Pnueli. Applications of temporal logic to the specification and verification of reactive systems: a survey of current trends. In *Current trends in Concurrency*, pages 510–584. Springer, 1986.

[Schild, 1993] Klaus Schild. Combining terminological logics with tense logic. In *Portuguese Conference on Artificial Intelligence*, pages 105–120. Springer, 1993.

[Sernadas, 1980] Amilcar Sernadas. Temporal aspects of logical procedure definition. *Information systems*, 5(3):167–187, 1980.

[Su *et al.*, 2009] Kaile Su, Abdul Sattar, Guanfeng Lv, and Yan Zhang. Variable forgetting in reasoning about knowledge. *Journal of Artificial Intelligence Research*, 35:677–716, 2009.

[Wang *et al.*, 2010] Zhe Wang, Kewen Wang, Rodney W. Topor, and Jeff Z. Pan. Forgetting for knowledge bases in DL-Lite. *Annuals of Mathematics and Artificial Intelligence*, 58(1-2):117–151, 2010.

[Wang *et al.*, 2012] Yisong Wang, Yan Zhang, Yi Zhou, and Mingyi Zhang. Forgetting in logic programs under strong equivalence. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*, pages 643–647, Rome, Italy, 2012. AAAI Press.

[Wang *et al.*, 2013] Yisong Wang, Kewen Wang, and Mingyi Zhang. Forgetting for answer set programs revisited. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 1162–1168, Beijing, China, 2013. IJCAI/AAAI.

[Wang, 2015] Yisong Wang. On forgetting in tractable propositional fragments. http://arxiv.org/abs/1502.02799, 2015.

[Wong, 2009] Ka-Shu Wong. *Forgetting in Logic Programs*. PhD thesis, The University of New South Wales, 2009.

[Zhang and Foo, 2006] Yan Zhang and Norman Y. Foo. Solving logic program conflict through strong and weak forgettings. *Artificial Intelligence*, 170(8-9):739–778, 2006.

[Zhang and Zhou, 2009a] Yan Zhang and Yi Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173(16):1525–1537, 2009.

[Zhang and Zhou, 2009b] Yan Zhang and Yi Zhou. Knowledge forgetting: Properties and applications. *Artificial Intelligence*, 173(16-17):1525–1537, 2009.

[Zhang *et al.*, 2005] Yan Zhang, Norman Y. Foo, and Kewen Wang. 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*, pages 627–634, 2005.

[Zhang *et al.*, 2008] Lan Zhang, Ullrich Hustadt, and Clare Dixon. First-order resolution for ctl. Technical report, Citeseer, 2008.

[Zhang *et al.*, 2009] Lan Zhang, Ullrich Hustadt, and Clare Dixon. A refined resolution calculus for ctl. In *International Conference on Automated Deduction*, pages 245–260. Springer, 2009.

[Zhao and Schmidt, 2017] Yizheng Zhao and Renate A Schmidt. Role forgetting for alcoqh (δ)-ontologies using an ackermann-based approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 1354–1361. AAAI Press, 2017.

[Zhao and Schmidt, 2018] Y. Zhao and R. A. Schmidt. Fame: An automated tool for semantic forgetting in expressive description logics. In D. Galmiche, S. Schulz, and R. Sebastiani, editors, *Automated Reasoning (IJCAR 2018)*, volume 10900 of *Lecture Notes in Artificial Intelligence*, pages 19–27. Springer, 2018.