Journal of Artificial Intelligence Research 1 (1993) 1-15 Submitted 6/91; published 9/91

Computing Sufficient and Necessary Conditions in CTL: A Forgetting Approach

Renyan Feng RENYAN_FENG@163.COM
Guizhou University, P. R. China

Erman Acar* ERMAN.ACAR@VU.NL
Vrije Universiteit Amsterdam, The Netherlands
Civic AI Lab Amsterdam, The Netherlands

Yisong Wang* YSWANG@GZU.EDU.CN
Guizhou University, P. R. China

Wanwei Liu WWLIU@NUDT.EDU.CN
National University of Defense Technology, P. R. China

Stefan Schlobach K.S.SCHLOBACH@VU.NL
Vrije Universiteit Amsterdam, The Netherlands

Abstract

Computation tree logic (CTL) is one of the most common specification languages in the field of formal verification. In systems design and verification, it is often important to update existing knowledge with new attributes and subtract the irrelevant information without altering the relevant system behaviour or violating the existing specifications over a given signature. In such a scenario, although they do not always exist for CTL, the computation of two informative dual notions are very crucial: the strongest necessary condition (SNC) and the weakest sufficient condition (WSC) of a given property in which the former corresponds to the most general consequence and the latter corresponds to the most specific abduction. In this article, we address these scenarios for CTL in a principled way in terms of knowledge *forgetting*.[1] In particular, we propose a model-theoretic notion of forgetting by a generalized bisimilar equivalence (over a signature) and explore its general properties, i.e., modularity, commutativity, and homogeneity, as a knowledge distilling operator, and show how to obtain SNC, WSC and the knowledge update via forgetting. We present a representation theorem for forgetting in bounded CTL, based on Zhang-Zhou's postulates. Furthermore, we complement our model-theoretic approach with a more efficient syntactic one; a resolution-based method, to compute the forgetting in CTL. We run our Prolog implementation on both CTL-RP benchmark and a randomly generated CTL dataset. Our experimental results suggest that the number of atoms to be forgotten has a high impact on the performance of the algorithm.

1. Introduction

Computation tree logic (CTL) (Clarke & Emerson, 1981) is one of the central formalisms in formal verification. As a specification language, it is used to express a property that the system at hand is expected to satisfy. From both the verification and the system design points of view, there might be situations in which some information content of such property

1. This paper is the revised and extended version of a paper which appeared in the Proceedings of KR 2020 (Feng, Acar, Schlobach, Wang, & Liu, 2020).

might become irrelevant for the system due to various reasons, e.g., it might be discarded or become obsolete by time, or just become infeasible due to practical difficulties. As keeping such irrelevant information would be undesirable (such as waste of computational resources), the problem arises on how to remove it without altering the relevant system behavior or violating the existing system specifications over a given signature. Consider the following example.

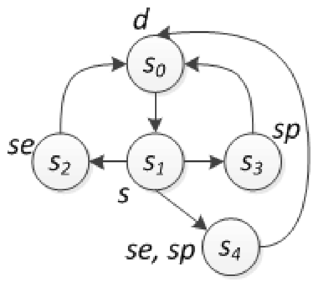


Figure 1: Car engine manufacturing scenario

Example 1 (Car-Manufacturing Company) *Assume a car-manufacturing company which produces two types of cars: a (se)dan car and a (sp)orts car. In each manufacturing cycle, the company has to (s)elect one of the three options: (1) produce se first, and then sp; (2) produce sp first, and then se; (3) produce se and sp at the same time. At the first of each selection, a (d)ecision is taken, i.e., given all the reasonable plans. This schematic view of this scenario is illustrated in Figure 1, as a Kripke structure* $\mathcal{M} = (S, R, L)$ *with the initial state* s_0 *(called labeled state transition graph), and the corresponding atomic variables* $V = \{d, s, se, sp\}$*. Now assume a situation in which due to some problems (e.g., economic crises or new environmental regulations on the engine technology) the company can no longer support the production of sports cars. This means that, all the manufacturing processes concerning sp are not necessary anymore. How should the company update the specification and the process (represented as a Kripke structure)?*

Similar scenarios may arise in many different application domains such as business-process modeling, software development, concurrent systems, and more (Baier & Katoen, 2008). Yet given a signature, dropping some restrictions in a large complex system without affecting the working system components, or violating dependent specifications is a non-trivial task. Moreover, in such a scenario, two logical notions are highly informative: the *strongest necessary condition* (SNC) and the *weakest sufficient condition* (WSC) of a given specification (Lin, 2001). These correspond to the *most general consequence* and the *most specific abduction* of such specification, respectively.

In formal verification, Dijkstra proposed a method, called the weakest precondition calculus, to compute the *weakest precondition* (WP) in verifying a 'Hoare triple' $\{\phi\}P\{\psi\}$, where P is a program and both ϕ and ψ are specifications (Dijkstra, 1975). In a similar scenario, given a model checking problem (i.e., deciding whether $\mathcal{M} \models \varphi$ for a given Kripke structure $\mathcal{M}$ and a specification $\varphi \in$ CTL), the problem of how to compute a *precondition* ψ over a given signature such that $\mathcal{M} \models \psi \rightarrow \varphi$ whenever $\mathcal{M} \not\models \varphi$ is raised. However, this cannot be computed by the method of Dijkstra since the given model $\mathcal{M}$ may be non-terminating. Besides, SP and the WP of a given specification are central to a wide variety of tasks and studies, e.g., in generating counterexamples (Dailler, Hauzar, Marché, & Moy, 2018) and refining system (Woodcock & Morgan, 1990) in verification. So to speak, if we have such a WP, then updating the model $\mathcal{M}$ with the WP (or WSC) is natural.

Updating logical databases traces back to Fagin et al.'s work in (Fagin, Kuper, Ullman, & Vardi, 1986). Various postulates are proposed for which arguably any plausible update function should satisfy. Among those postulates, Katsuno and Mendelzon's postulates (U1)-(U8) (Katsuno & Mendelzon, 1991), are well-studied. In (Baral & Zhang, 2005), the *knowledge update* for modal logic S5 has been introduced, using the modal operator K (for agent *knows*), and the proposed notion of update has been shown to satisfy Katsuno and Mendelzon's postulates.

Using a forgetting-based method to update databases has been studied for propositional logic (PL) formulas in (Fang, Wan, Liu, Fang, & Lai, 2018) and for atoms in modal logic S5 in (Zhang & Zhou, 2009). Nevertheless, to the best of our knowledge, the knowledge update of CTL including temporal operators has not been explored so far.

To address these scenarios and target the relevant notions of SNC and WSC in a principled way, we introduce a *forgetting*-based approach in CTL and show that it can be used to compute SNC and WSC on a restricted subset of the propositional variables, in the same spirit of (Lin, 2001; Doherty, Lukaszewicz, & Szalas, 2001). In addition, we demonstrate that the knowledge update can be computed by forgetting.

The **major contributions** of our paper are as follows:

- We define the notion of forgetting in CTL via the bisimulation on some set of atoms and explores its semantics properties, i.e., modularity, commutativity, and homogeneity.

- We explore a resolution-based method to compute forgetting in CTL by using the resolution rules proposed in (Zhang, Hustadt, & Dixon, 2014). To achieve this, we provide the notion of binary bisimulation (over a given signature and a set of indices) and an elimination process, including a generalized Ackermanns lemma. Moreover, the theorem shows that some atoms introduced in the calculational process can not be eliminated from its result for some CTL formulas, this follows that CTL is a failure of uniform interpolation.

- We outline a situation in which the forgetting is closed. To prove that it is closed, we define the characterizing formula (i.e., a CTL formula) of the (finite) initial structure and show that each CTL formula is equivalent to a disjunction of the characterizing formulas of its models. This fact means that the result of forgetting some atoms from a CTL formula always exists. Besides, we propose a model-based method to compute forgetting even if it is not very efficient, as one might expect. However, settling it is important from a theoretical point of view, to see how costly is the naive approach.

- We show that the WSC (SNC) and knowledge update can be obtained by using the forgetting technique. In particular, we prove that the WSC (SNC) of a specification (i.e., a CTL formula) on a given signature under a (finite) initial structure can be obtained by expressing this structure with its characterizing formula. Moreover, we exhibit the knowledge update defined by forgetting satisfies Katsuno and Mendelzon's postulates (U1)-(U8) proposed in (Katsuno & Mendelzon, 1991).

- We have run our resolution-based algorithm (implemented in Prolog) on a set of CTL formulas that both from the CTL-RP benchmarks and generated randomly to complement these theoretical insights with empirical findings and obtain the SNC of the given formula.

The rest of the paper is organized as follows. Section 2 reports on the related work. Section 3 introduces the notation and technical preliminaries. Section 4 offers the notion of forgetting and its semantics properties. Moreover, it provides a resolution-based method to compute forgetting in Section 5. Experimental results are shown in Section 7 to evaluate the Resolution-based method from an intuitive perspective. The conclusion closes the paper. To avoid hindering the flow of content, proofs and the wearying constructions are moved into the Appendix.

2. Related work

This section surveys previous works which are closely related to this paper and are important in artificial intelligence.

2.1 Forgetting

To progress the knowledge of cognitive robotics, *forgetting* was first formally presented in first order logic (FOL) (Lin & Reiter, 1994). It can be traced back to the work of Boole on propositional variable elimination and the seminal work of quantifier elimination (Ackermann, 1935). Noteworthy to mention is that forgetting is a dual notion of long-studied *uniform interpolation* (Visser, 2017; Konev, Walther, & Wolter, 2009).

In propositional logic (PL), *forgetting* a propositional variable p from a formula φ, written $\mathrm{F}(\varphi, \{p\})$, is the formula $\varphi[p/\bot] \vee \varphi[p/\top]$, which is equivalent to $\exists p\varphi$ (Lin & Reiter, 1994). *Forgetting* a finite set P of propositions from φ, written $\mathrm{F}(\varphi, P)$, is defined as:

$$\mathrm{F}(\varphi, \emptyset) = \varphi, \qquad \mathrm{F}(\varphi, P \cup \{q\}) = \mathrm{F}(\mathrm{F}(\varphi, \{q\}), P).$$

In the context of FOL, forgetting has often been studied as an instance of the *second-order quantifier elimination* (SOQE) problem. In particular, it has been shown that the result of (strong) forgetting of an n-ary predicate P from a FOL formula φ is the second-order formula $\exists R\varphi[P/R]$, in which R is an n-ary predicate variable and $\varphi[X/Y]$ is the result of replacing every occurrence of X in φ by Y (Lin & Reiter, 1994). Hence, the task of forgetting in FOL is to find a first-order formula that is equivalent to $\exists R\varphi[P/R]$. However, a second-order formula is possibly not expressible by a first-order one. As an ontology engineering technique to produce *views* of ontologies, forgetting has been widely explored in description logics (DLs), that are subclasses of first-order logic (Wang, Wang, Topor, & Pan, 2010; Lutz & Wolter, 2011; Konev, Ludwig, Walther, & Wolter, 2012; Zhao & Schmidt, 2017; Zhao, Schmidt, Wang, Zhang, & Feng, 2020).

Knowledge forgetting was firstly proposed in the propositional modal logic S5 to reason about agents' epistemic states (knowledge or belief). In doing so, four general postulates were presented to precisely characterize the notion of knowledge forgetting (Zhang & Zhou, 2009). Furthermore, in the scenario of non-monotonic reasoning, (Knowledge) *forgetting* in logic programs under answer set semantics has been extensively investigated from the perspective of various forgetting postulates (Zhang & Foo, 2006; Eiter & Wang, 2008; Wong, 2009; Wang, Zhang, Zhou, & Zhang, 2014; Wang, Wang, & Zhang, 2013; Delgrande, 2017).

2.2 The Weakest Sufficient Condition (WSC)

As aforementioned, the WSC and its dual notion of the strongest necessary condition (SNC) are highly crucial notions for formal verification in software engineering. Roughly speaking, the WSC corresponds to the *most specific abduction* and *the weakest precondition* (WP). Particularly, given a program S and a specification Q about states, the WSC for S w.r.t. Q is a specification that is true for precisely those initial states of which: (i) S must terminate, and (ii) executing S must produce a state satisfying Q. Four rules of computing the WP were firstly presented by Dijkstra (1975). Moreover, the notions of SNC and WSC were considered in the scope of formal verification, among others to refine system (Woodcock & Morgan, 1990) and to generate counterexamples (Dailler et al., 2018).

In the field of knowledge representation (KR), the WSC and SNC provide a method to generate successor state axioms from causal theories (Lin, 2003). Both WSC and SNC can be computed by forgetting (Lin, 2001; Doherty et al., 2001).

Theorem 1 (Theorem 2 of (Lin, 2001)) *Let T be a formula of PL, P a set of propositions, and q a proposition in T but not in P. Let P' be the set of propositions that are in T but not in $P \cup \{q\}$.*

- *The SNC of q on P is $\mathrm{F}(T[q/\top], P')$;*
- *The WSC of q on P is $\neg\mathrm{F}(T[q/\bot], P')$.*

The SNC and WSC were generalized from PL to FOL, and a direct method based on the SOQE technique has been proposed to automatically generate SNC and WSC by Doherty et al. (2001).

Theorem 2 (Lemma 4.1 of (Doherty et al., 2001)) *For any formula α, any set P of relation symbols and a closed theory Th:*

- *the SNC of α on P is defined by $\exists\overline{\Phi}.[Th \wedge \alpha]$,*
- *the WSC of α on P is defined by $\forall\overline{\Phi}.[Th \rightarrow \alpha]$,*

where $\overline{\Phi}$ is the set of all relation symbols appearing in Th and α but not in P.

Very recently, we have studied the forgetting in bounded[2] CTL and showed that both WSC and SNC can be computed via forgetting (Feng et al., 2020). However, the forgetting results are computed according to characterization formulas of Kripke structures, which is very expensive and maybe not feasible in practice. To practically compute forgetting results for CTL, we extend the proposed forgetting for bounded CTL to CTL with indexes and employ its sound and complete resolution procedure for our purpose.

2. With finite states in Kripke structures.

3. Preliminaries

In the section we recall the basic syntax, semantics of CTL (with indexes) (Clarke, Emerson, & Sistla, 1986; Emerson, 1990; Baier & Katoen, 2008) and its resolution procedure for CTL in normal form, called *separated normal form with global clauses* (Zhang et al., 2014).

We assume an underlying signature $\mathcal{A}$ of propositional variables (called *atoms*) and a countably infinite set $\mathcal{I}$ of indices. A *literal* is an atom p or its negation $\neg p$.p We denote $\overline{V} = \mathcal{A} - V$ for $V \subseteq \mathcal{A}$, $\bigwedge S$ (resp. $\bigvee S$) the conjunction (resp. disjunction) of the formulas in S for a finite set S of formulas.

3.1 Syntax of Indexed CTL

The *indexed CTL formulas* is formally defined as below:

$$\phi ::= \mathbf{start} \mid \bot \mid p \mid \neg\phi \mid \phi \vee \phi \mid \mathrm{EX}\phi \mid \mathrm{EG}\phi \mid \mathrm{E}(\phi \ \mathrm{U} \ \phi) \mid \mathrm{E}_{\langle ind \rangle}\mathrm{X}\phi \mid \mathrm{E}_{\langle ind \rangle}\mathrm{G}\phi \mid \mathrm{E}_{\langle ind \rangle}(\phi \mathrm{U} \phi)$$

where

- $p \in \mathcal{A}$, **start** and $\bot$ are propositional constants, $ind \in \mathcal{I}$;
- E and $\mathrm{E}_{\langle ind \rangle}$ are *existential path quantifiers* meaning that there is a path (with index);
- the *temporal operators* X, F, G, U and W means 'neXt state', 'some Future state', 'all future states (Globally)', 'Until' and 'unless (W)' respectively.

The other connectives $\wedge$ and $\rightarrow$ are defined in a standard manner of classical logic in terms of $\neg$ and $\wedge$. The *universal path quantifiers* A is defined using existential path quantifier. The (indexed) temporal operators F (future) and W (unless) can be defined using these (indexed) temporal operators X, U and G:

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

Formulas without index and the propositional constant **start** are called (CTL) *formulas* for simplicity. The set of atoms occurring in ϕ is denoted by $Var(\phi)$.

The priorities among connectives and modalities are assumed to be in order as below, the leftmost (rightmost) has the highest (lowest) priority:

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

Given a formula φ containing no '$\rightarrow$' and an atom p, an occurrence of p in φ is *positive* if it is preceded by even number of negative connectives $\rightarrow$ or $\neg$, it is *negative* otherwise. The formula φ is *positive* (resp. *negative*) w.r.t. p if every occurrence of p in φ is positive (resp. negative).

3.2 Semantics of Indexed CTL formula

An *initial $\mathcal{I}$-Kripke structure* is a quintuple $\mathcal{M} = (S, R, L, [\_], s_0)$, where

- S is a nonempty set of states, $s_0 \in S$ is a initial state of $\mathcal{M}$ (defined below);
- $R \subseteq S \times S$ and, there is $s' \in S$ s.t. $(s, s') \in R$ for each $s \in S$;
- $L : S \to 2^{\mathcal{A}}$ is a labeling function;
- $[\_] : \mathcal{I} \to 2^{S \times S}$ is a function which maps every index $ind \in \mathcal{I}$ to a successor function $[ind]$ *s.t.* for every $s \in S$ there exists exactly a state $s' \in S$ s.t. $(s, s') \in [ind] \cap R$.

A *path* of a Kripke structure $\mathcal{M} = (S, R, L)$ is an infinite sequence $\pi = (s_0, s_1, s_2, \dots)$ of states of $\mathcal{M}$ with $(s_j, s_{j+1}) \in R$ for every $j \geq 0$. By $s' \in \pi$, we mean that s' is a state occurring in the path π. In particular, we denote π_s a path of $\mathcal{M}$ starting from s. A state s is *initial* if there is a path π_s of $\mathcal{M}$ *s.t.* $s' \in \pi_s$ for each state $s' \in S$. An *indexed path* $\pi_s^{\langle ind \rangle}$ of an $\mathcal{I}$-Kripke structure $\mathcal{M} = (S, R, L, [\_])$ is a path $(s_0(= s),\ s_1,\ s_2,\ \dots)$ *s.t.* $(s_j, s_{j+1}) \in [ind]$ for every $j \geq 0$, where $ind \in \mathcal{I}$.

The *initial $\mathcal{I}$-Kripke structure* $\mathcal{M}$ is called an *initial structure* if it has no $[\_]$; it is called an *$\mathcal{I}$-Kripke structure* if it has no the initial state s_0; it is called a *Kripke structure* if it has no s_0 and $[\_]$. An *($\mathcal{I}$-)structure* is a tuple $\mathcal{K} = (\mathcal{M}, s)$ where $\mathcal{M}$ is an ($\mathcal{I}$-)Kripke structure and s is a state of $\mathcal{M}$. It is called an *initial $\mathcal{I}$-structure* if s is a initial state of $\mathcal{M}$. The above notion of (indexed) path for these structures are similarly defined.

Definition 1 *Let $\mathcal{M} = (S, R, L, [\_], s_0)$ be an initial $\mathcal{I}$-Kripke structure, $s \in S$ and ϕ be an indexed CTL formula. The* satisfiability *between $(\mathcal{M}, s)$ and ϕ, written $(\mathcal{M}, s) \models \phi$, is defined as follows:*

- $(\mathcal{M}, s) \models \boldsymbol{start}$ *iff* $s = s_0$;
- $(\mathcal{M}, s) \not\models \bot$; $(\mathcal{M}, s) \models p$ *iff* $p \in L(s)$;
- $(\mathcal{M}, s) \models \phi_1 \vee \phi_2$ *iff* $(\mathcal{M}, s) \models \phi_1$ *or* $(\mathcal{M}, s) \models \phi_2$;
- $(\mathcal{M}, s) \models \neg\phi$ *iff* $(\mathcal{M}, s) \not\models \phi$;
- $(\mathcal{M}, s) \models \mathrm{EX}\phi$ *iff* $(\mathcal{M}, s_1) \models \phi$ *for some* $(s, s_1) \in R$;
- $(\mathcal{M}, s) \models \mathrm{EG}\phi$ *iff* $\mathcal{M}$ *has a path* $(s_1 = s, s_2, \dots)$ *such that* $(\mathcal{M}, s_i) \models \phi$ *for each* $i \geq 1$;
- $(\mathcal{M}, s) \models \mathrm{E}(\phi_1 \mathrm{U} \phi_2)$ *iff* $\mathcal{M}$ *has a path* $(s_1 = s, s_2, \dots)$ *such that, for some* $i \geq 1$, $(\mathcal{M}, s_i) \models \phi_2$ *and* $(\mathcal{M}, s_j) \models \phi_1$ *for each* j $(1 \leq j < i)$;
- $(\mathcal{M}, s) \models \mathrm{E}_{\langle ind \rangle}\mathrm{X}\psi$ *iff for the path* $\pi_s^{\langle ind \rangle}$, $(\mathcal{M}, s') \models \psi$ *with* $(s, s') \in [ind]$;
- $(\mathcal{M}, s) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\psi$ *iff for every* $s' \in \pi_s^{\langle ind \rangle}$, $(\mathcal{M}, s') \models \psi$;
- $(\mathcal{M}, s) \models \mathrm{E}_{\langle ind \rangle}(\psi_1 \mathrm{U} \psi_2)$ *iff there exists* s_j $(1 \leq j)$ *occurring in* $\pi_s^{\langle ind \rangle} = (s = s_1, s_2, \dots)$ *s.t.* $(\mathcal{M}, s_j) \models \psi_2$ *and for every* $s_k \in \pi_s^{\langle ind \rangle}$, *if* $1 \leq k < j$, *then* $(\mathcal{M}, s_k) \models \psi_1$.

An initial $\mathcal{I}$-structure $\mathcal{K}$ is a *model* of a formula ϕ whenever $\mathcal{K} \models \phi$. By $Mod(\phi)$ we denote the set of models of ϕ. The formula ϕ is *satisfiable* if $Mod(\phi) \neq \emptyset$. Please note there that only initial $\mathcal{I}$-structures are considered to be models here. It is the case in (Browne, Clarke, & Grümberg, 1988; Bolotov, 1999; Zhang et al., 2014).

Given two formulas ϕ_1 and ϕ_2, by $\phi_1 \models \phi_2$ we mean $Mod(\phi_1) \subseteq Mod(\phi_2)$, and by $\phi_1 \equiv \phi_2$ we denote $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$. In this case, ϕ_1 is *equivalent* to ϕ_2. In addition, for a set Π of formulas, we say $\mathcal{K} \models \Pi$ if $\mathcal{K} \models \varphi$ for each $\varphi \in \Pi$. The formula ϕ_1 is *irrelevant to* the atoms in a set $V \subseteq \mathcal{A}$ (or simply V*-irrelevant*), written $\mathrm{IR}(\phi_1, V)$, if there is a formula ψ with $Var(\psi) \cap V = \emptyset$ and $\phi_1 \equiv \psi$.

The following lemma is evident by the above definition.

Lemma 1 *Let φ and ψ be two formulas. Then the following holds.*

(i) $\mathrm{AG}(\varphi \wedge \psi) \equiv (\mathrm{AG}\varphi) \wedge (\mathrm{AG}\psi)$.

(ii) $\mathrm{AGAG}(\varphi) \equiv \mathrm{AG}(\varphi)$.

(iii) $\mathrm{AG}(\alpha \to \varphi) \equiv \varphi$ *where* $\alpha \in \{\textbf{start}, \top\}$.

Two indexed CTL formulas φ and ψ are *equi-satisfiable* whenever it holds that φ is satisfiable if and only if ψ is satisfiable. It is evident that equivalent formulas are always equi-satisfiable, but not vice versa. For instance, the two formulas $\mathrm{E}_{\langle 1\rangle}\mathrm{X}p$ and $\mathrm{E}_{\langle 2\rangle}\mathrm{X}p$ are clearly equi-satisfiable, but they are not equivalent. The next lemma easily follows.

Lemma 2 *Let φ be an indexed CTL formula.*

(i) φ and φ' are equi-satisfiable where φ' is obtained from φ by renaming some index with a fresh one, i.e. replacing the every occurrence of an index with a new index not occurring in the formula.

(ii) $\varphi \models \varphi''$ where φ'' is obtained from φ by removing all indices.

Please note that the inverse of (ii) in the above proposition does not hold. For instance, let $\varphi = \mathrm{E}_{\langle 1\rangle}\mathrm{X}p \wedge \mathrm{E}_{\langle 1\rangle}\mathrm{X}\neg p$. It is clear that $\varphi'' = \mathrm{EX}p \wedge \mathrm{EX}\neg p$ which is clearly satisfiable, while φ is not satisfiable. Thus, $\varphi'' \not\models \varphi$.

3.3 Separated Normal Form and Resolution

In the subsection we recall the clausal forms of CTL with index, the rules of transforming CTL formulas into the clausal forms and the sound and complete resolution refutation procedure for the clausal theories (Zhang et al., 2014).

A *separated normal form clause* of CTL with index, named $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause (or *clause* simply), has one of the following forms:

$\mathrm{AG}(\textbf{start} \to \bigvee_{j=1}^{k} m_j)$	(initial clause)
$\mathrm{AG}(\top \to \bigvee_{j=1}^{k} m_j)$	(global clause)
$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{AX}\bigvee_{j=1}^{k} m_j)$	(A-step clause)
$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{E}_{\langle ind\rangle}\mathrm{X}\bigvee_{j=1}^{k} m_j)$	(E-step clause)
$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{AF}l)$	(A-sometime clause)
$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \to \mathrm{E}_{\langle ind\rangle}\mathrm{F}l)$	(E-sometime clause)

Table 1: Transformation Rules

Trans(1) $\dfrac{q \to \mathrm{E}T\varphi}{q \to \mathrm{E}_{\langle ind \rangle}T\varphi}$;	**Trans(2)** $\dfrac{q \to \mathrm{E}(\varphi_1 \mathrm{U} \varphi_2)}{q \to \mathrm{E}_{\langle ind \rangle}(\varphi_1 \mathrm{U} \varphi_2)}$;	**Trans(3)** $\dfrac{q \to \varphi_1 \wedge \varphi_2}{q \to \varphi_1, q \to \varphi_2}$;
Trans(4) $\dfrac{q \to \varphi_1 \vee \varphi_2 \text{ (if } \varphi_2 \text{ is not a disjunct)}}{q \to \varphi_1 \vee p, p \to \varphi_2}$;		**Trans(5)** $\dfrac{q \to D}{\top \to \neg q \vee D}$; $\dfrac{q \to \bot}{\top \to \neg q}$; $\dfrac{q \to \top}{\{\}}$
Trans(6) $\dfrac{q \to Q\mathrm{X}\varphi \text{ (if } \varphi \text{ is not a disjunct)}}{q \to Q\mathrm{X}p, p \to \varphi}$;		**Trans(7)** $\dfrac{q \to Q\mathrm{F}\varphi \text{ (if } \varphi \text{ is not a literal)}}{q \to Q\mathrm{F}p, p \to \varphi}$;
Trans(8) $\dfrac{q \to Q(\varphi_1 \mathrm{U} \varphi_2) \text{ (if } \varphi_2 \text{ is not a literal)}}{q \to Q(\varphi_1 \mathrm{U} p), p \to \varphi_2}$;		**Trans(10)** $\dfrac{q \to Q\mathrm{G}\varphi}{q \to p, p \to \varphi, p \to Q\mathrm{X}p}$;
Trans(9) $\dfrac{q \to Q(\varphi_1 \mathrm{W} \varphi_2) \text{ (if } \varphi_2 \text{ is not a literal)}}{q \to Q(\varphi_1 \mathrm{W} p), p \to \varphi_2}$;		
Trans(11) $\dfrac{q \to Q(\varphi \mathrm{U} l)}{q \to l \vee p, p \to \varphi, p \to Q\mathrm{X}(l \vee p), q \to Q\mathrm{F}l}$;		**Trans(12)** $\dfrac{q \to Q(\varphi \mathrm{W} l)}{q \to l \vee p, p \to \varphi, p \to Q\mathrm{X}(l \vee p)}$.

Notations: $T \in \{\mathrm{X}, \mathrm{G}, \mathrm{F}\}$, $Q \in \{\mathrm{A}, \mathrm{E}_{\langle ind \rangle}\}$; *ind* is a fresh index; p is a fresh atom; q is an atom, l is a literal, D is a disjunction of literals; φ, φ_1, and φ_2 are CTL formulas.

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 $ind \in \mathcal{I}$. As all $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses are of the form $\mathrm{AG}(\varphi \to \psi)$, we shall shorten it as $\varphi \to \psi$ for convenience when it is clear from the context, where φ and ψ are formulas.

It has been shown that any CTL formula φ can be transformed in polynomial time into an equi-satisfiable set T_φ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses using transformation rules in Table 1.

Intuitively, Trans(1) and Trans(2) introduce a fresh index for each existential path quantifier, and no other rules will introduce index. Trans(3), Trans(4), and Trans(5) are *Boolean rules* to decompose complex formulae by introducing fresh atomic propositions. Trans(6) to Trans(12) are *temporal operator rules* to remove combinations of temporal operators which are not allowed to appear in $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses. These transformation rules translate a *premise* into a set of *consequents*. Formally, T_φ is obtained in the sequence of derivation:

$$T_0 = \{\mathrm{AG}(\mathbf{start} \to p), \mathrm{AG}(p \to \mathbf{simp}(\mathbf{nnf}(\varphi)))\}, T_1, \ldots, T_n = T_\varphi$$

where

- p is a fresh propositional atom, *i.e.* $p \notin \{\mathbf{start}\} \cup \mathit{Var}(\varphi)$;
- $\mathbf{nnf}(\varphi)$ is a formula equivalent to φ and is in negative normal form;
- $\mathbf{simp}(\varphi)$ is obtained from φ by simplifying it according to the equivalences in CTL;
- $T_{t+1} = (T_t - \{\psi\}) \cup R_t$ $(t \geq 0)$, where ψ is a formula in T_t not in clausal form $\mathrm{SNF}^g_{\mathrm{CTL}}$, and R_t is the result of applying a matching transformation rule to ψ; and
- Every formula in T_n is in $\mathrm{SNF}^g_{\mathrm{CTL}}$ clausal form.

In what follows we also denote $\mathrm{SNF}^g_{\mathrm{CTL}}(\varphi)$ the set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses obtained from φ by the above transformation rules. The below lemma is evident.

Lemma 3 *Let φ be a CTL formula.*

Table 2: Resolution rules

(SRES1) $\frac{P \to \mathrm{AX}(C \lor l),\, Q \to \mathrm{AX}(D \lor \neg l)}{P \land Q \to \mathrm{AX}(C \lor D)}$;	**(SRES2)** $\frac{P \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(C \lor l),\, Q \to \mathrm{AX}(D \lor \neg l)}{P \land Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(C \lor D)}$;
(SRES3) $\frac{P \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(C \lor l),\, Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(D \lor \neg l)}{P \land Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(C \lor D)}$;	**(SRES4)** $\frac{\mathbf{start} \to C \lor l,\, \mathbf{start} \to D \lor \neg l}{\mathbf{start} \to C \lor D}$;
(SRES5) $\frac{\top \to C \lor l,\, \mathbf{start} \to D \lor \neg l}{\mathbf{start} \to C \lor D}$;	**(SRES6)** $\frac{\top \to C \lor l,\, Q \to \mathrm{AX}(D \lor \neg l)}{Q \to \mathrm{AX}(C \lor D)}$;
(SRES7) $\frac{\top \to C \lor l,\, Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(D \lor \neg l)}{Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}(C \lor D)}$;	**(SRES8)** $\frac{\top \to C \lor l,\, \top \to D \lor \neg l}{\top \to C \lor D}$;
(RW1) $\frac{\bigwedge_{i=1}^{n} m_i \to \mathrm{AX}\perp}{\top \to \bigvee_{i=1}^{n} \neg m_i}$;	**(RW2)** $\frac{\bigwedge_{i=1}^{n} m_i \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}\perp}{\top \to \bigvee_{i=1}^{n} \neg m_i}$;
(ERES1) $\frac{\Lambda \to \mathrm{EXEG}l,\, Q \to \mathrm{AF}\neg l}{Q \to \mathrm{A}(\neg \Lambda \mathrm{W} \neg l)}$;	**(ERES2)** $\frac{\Lambda \to \mathrm{E}_{\langle ind \rangle}\mathrm{XE}_{\langle ind \rangle}\mathrm{G}l,\, Q \to \mathrm{E}_{\langle ind \rangle}\mathrm{F}\neg l}{Q \to \mathrm{E}_{\langle ind \rangle}(\neg \Lambda \mathrm{W} \neg l)}$.

Notations: P and Q are conjunction of literals; C and D are disjunction of literals; l is a literal (to be resolved); $\Lambda = \bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} P_j^i$ and P_j^is are conjunction of literals. The resolvent of **ERES1** and **ERES2** stands for an equi-satisfiable set of $\mathrm{SNF}_{\mathrm{CTL}}^{g}$ clauses respectively.

(i) *Exact one fresh index will be introduced in* $\mathrm{SNF}_{\mathrm{CTL}}^{g}(\varphi)$ *for each occurrence of the path quantifier* E *in* φ.

(ii) *There is no two* E*-sometime clauses in* $\mathrm{SNF}_{\mathrm{CTL}}^{g}(\varphi)$ *having the same index, viz., if* $P_i \to \mathrm{E}_{\langle j_i \rangle}\mathrm{F}l_i$ $(i = 1, 2)$ *are in* $\mathrm{SNF}_{\mathrm{CTL}}^{g}(\varphi)$ *then* $j_1 \neq j_2$.

(iii) *No atom* $p \in Var(\varphi)$ *occurs in the left-hand side of an implication in* $\mathrm{SNF}_{\mathrm{CTL}}^{g}(\varphi)$.

One should note that every premise φ and every consequent ψ in these transformation rules are indeed the formula $\mathrm{AG}\varphi$ and $\mathrm{AG}\psi$, respectively. It is possible that, in $\mathrm{SNF}_{\mathrm{CTL}}^{g}(\varphi)$, an E-sometimes clause and an E-step clause may have the same index according to the rule **Trans(11)**, and two E-sometimes clauses may have the same index as well.

Let us consider the following example. lemma中的第(ii)点

Example 2 *Let* $\varphi = \mathrm{A}((p \land q)\mathrm{U}(f \lor m)) \land r$. *The theory* T_φ *consists of the following formulas:*

$$1: \textbf{\textit{start}} \to z, \quad 2: \top \to \neg z \lor r, \quad 3: \top \to \neg x \lor f \lor m, \quad 4: \top \to \neg z \lor x \lor y,$$
$$5: \top \to \neg y \lor p, \quad 6: \top \to \neg y \lor q, \quad 7: z \to \mathrm{AF}x, \quad 8: y \to \mathrm{AX}(x \lor y)$$

where x, y, z *are the introduced new atoms.*

It is proved that the CTL formula φ and the clausal theory T_φ are equi-satisfiable, viz., φ is satisfiable if and only if T_φ is, cf. Theorem 5.5. Once a CTL formula φ is transformed into the equi-satisfiable clausal theory T_φ, a resolution-based decision procedure can be used to decide whether φ is satisfiable, where the resolution rules are presented in Table 2.

One should note that **SRES1-8** are called *step resolution rules*, **RW1-2** are called *rewrite rules*, **ERES1-2** are called *eventuality resolution rules*. In particular, the first premise $\Lambda \to \textsc{exeg}l$ of the resolution rule **ERES1** represents a set, $\Lambda_{\textsc{eg}}$, of $\text{SNF}^g_{\textsc{ctl}}$ clauses

$$\begin{array}{lcr} P^1_1 \to *C^1_1, & & P^n_1 \to *C^n_1, \\ \vdots & & \vdots \\ P^1_{m_1} \to *C^1_{m_1}, & \dots & P^n_{m_n} \to *C^n_{m_n}, \end{array}$$

where, for every i $(1 \leq i \leq n)$,

- there is some $i \in \mathcal{I}$ such that each $*$ is either empty or an operator in $\{\textsc{ax}, \textsc{e}_{\langle i \rangle}\textsc{x}\}$,
- $\left(\bigwedge_{j=1}^{m_i} C^i_j\right) \to l$ is provable, and
- $\left(\bigwedge_{j=1}^{m_i} C^i_j\right) \to \left(\bigvee_{i=1}^{n} \bigwedge_{j=1}^{m_i} P^i_j\right)$ is provable.

The above last two conditions ensure that the set $\Lambda_{\textsc{eg}}$ of $\text{SNF}^g_{\textsc{ctl}}$ clauses implies $\Lambda \to \textsc{exeg}l$. The first premise of **ERES2** is similar to that of **ERES1**. The resolvent of eventuality resolution rule **ERES1** can be translated into an equi-satisfiable set of global and A-step $\text{SNF}^g_{\textsc{ctl}}$ clauses, while the resolvent of **ERES2** can be translated into an equi-satisfiable set of global and E-step $\text{SNF}^g_{\textsc{ctl}}$ clauses in which a fresh atom will be introduced (Zhang et al., 2014). In this sense, every resolvent resulted from these resolution rules is a $\text{SNF}^g_{\textsc{ctl}}$ clause, and every premise of these resolution rules are $\text{SNF}^g_{\textsc{ctl}}$ clauses as well.

With these resolution rules, one can derive (logical) consequences. Formally speaking, a *derivation* from a set S of $\text{SNF}^g_{\textsc{ctl}}$ clauses is a sequence $S_0, S_1, S_2, \dots$, of sets of $\text{SNF}^g_{\textsc{ctl}}$ clauses such that

- $S_0 = S$, and
- $S_{i+1} = S_i \cup \{\alpha\}$ $(i \geq 0)$ where $\alpha \notin S_i$ is a resolvent obtained as the conclusion of an application of a resolution rule to the premises in S_i.

A *refutation* of a set S of $\text{SNF}^g_{\textsc{ctl}}$ clauses is a derivation $S_0, S_1, \dots S_i$ from S such that S_i contains a *contradiction* for some $i \geq 0$, where a contradiction is either the formula $\top \to \bot$ or **start** $\to \bot$.

To decide the satisfiability of a given CTL formula φ, the resolution based decision procedure is to check whether there is a refutation of T_φ. This decision procedure is sound, complete and in EXP, cf. Theorems 5.6, 5.30 and 6.1 of (Zhang et al., 2014) respectively. The below corollary follows easily.

Corollary 3 *Let φ and ψ be two CTL formulas. Then $\varphi \models \psi$ if and only if there is a refutation of $T_{\varphi \wedge \neg\psi}$.*

4. Semantic Forgetting

This part focuses on forgetting in CTL, exploring its concepts and properties. First, we give a general definition of *bisimulation* between $\mathcal{I}$-Kripke structures to define forgetting in CTL. The notion of bisimulation captures the idea that the computation trees of two structures are behaviourally the same. Second, the related properties, which include modularity, commutativity, and homogeneity of the forgetting operator, will be explored.

4.1 Generalized Bisimulation

In this subsection, we present a notion of generalized bisimulation w.r.t. a set V of atomic propositions and a set I of indices for our aims, and explore its properties.

Definition 2 (V-bisimulation) *Let $V \subseteq \mathcal{A}$, $I \subseteq \mathcal{I}$, and $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ $(i = 1, 2)$ be $\mathcal{I}$-Kripke structures. A relation $\mathcal{B}_V \subseteq S_1 \times S_2$ is a V-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ if, for every $s_1 \in S_1$ and every $s_2 \in S_2$, $(s_1, s_2) \in \mathcal{B}_V$ implies:*

(i) $L_1(s_1) - V = L_2(s_2) - V$;

(ii) $\forall r_1 \in S_1$, *if* $(s_1, r_1) \in R_1$, *then* $\exists r_2 \in S_2$ *s.t.* $(s_2, r_2) \in R_2$ *and* $(r_1, r_2) \in \mathcal{B}_V$;

(iii) $\forall r_2 \in S_2$, *if* $(s_2, r_2) \in R_2$, *then* $\exists r_1 \in S_1$ *s.t.* $(s_1, r_1) \in R_1$ *and* $(r_1, r_2) \in \mathcal{B}_V$;

Two $\mathcal{I}$-structures $\mathcal{K}_1 = (\mathcal{M}_1, s_1)$ and $\mathcal{K}_2 = (\mathcal{M}_2, s_2)$ are V-*bisimilar*, denoted as $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$, if there is a V-bisimulation $\mathcal{B}_V$ between $\mathcal{M}_1$ and $\mathcal{M}_2$ such that $(s_1, s_2) \in \mathcal{B}_V$. Let $i \in \{1, 2\}$. Two paths $\pi_i = (s_{i,1}, s_{i,2}, \ldots)$ of $\mathcal{M}_i$ are V-*bisimilar*, denoted as $\pi_1 \leftrightarrow_V \pi_2$, if $\mathcal{K}_{1,j} \leftrightarrow_V \mathcal{K}_{2,j}$ for every $j \geq 1$, where $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$.

Intuitively, two states are "bisimular" if they behave the same when discarding the propositions in the set $V \subseteq \mathcal{A}$. When $V = \emptyset$, the three conditions of V-bisimulation are the conditions of bisimulation in CTL. In the following we abbreviate $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ to $s_1 \leftrightarrow_V s_2$ when the underlying $\mathcal{I}$-Kripke structures are clear from the context.

Example 3 (Continued from Example 1) *Let us recall the model given in the previous example as $\mathcal{K}_1$ with initial state s_0, i.e., $\mathcal{K}_1 = ((S, R, L, [\_], s_0), s_0)$, as illustrated in Figure 2. Then, $\mathcal{K}_2$ is obtained from $\mathcal{K}_1$ by removing sp,*[3] *and $\mathcal{K}_3$ is obtained from $\mathcal{K}_2$ by removing se. Observe that $\mathcal{K}_1 \leftrightarrow_{\{sp\}} \mathcal{K}_2$, $\mathcal{K}_2 \leftrightarrow_{\{se\}} \mathcal{K}_3$, and $\mathcal{K}_1 \leftrightarrow_{\{sp,se\}} \mathcal{K}_3$. Besides, $\mathcal{K}_1$ is not bisimilar (Baier & Katoen, 2008) with either $\mathcal{K}_2$ or $\mathcal{K}_3$.*

The next proposition shows the similar properties of Proposition **??** for V-bisimulations.

Proposition 1 *Let $i \in \{1, 2\}$, $V_1, V_2 \subseteq \mathcal{A}$, s_1' and s_2' be two states, π_1' and π_2' be two paths, and $\mathcal{K}_j = (\mathcal{M}_j, s_j)$ $(j = 1, 2, 3)$ be structures s.t. $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$ and $\mathcal{K}_2 \leftrightarrow_{V_2} \mathcal{K}_3$. Then:*

(i) $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$;

(ii) *If* $V_1 \subseteq V_2$ *then* $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$;

(iii) $s_1' \leftrightarrow_{V_i} s_2'$ $(i = 1, 2)$ *implies* $s_1' \leftrightarrow_{V_1 \cup V_2} s_2'$;

(iv) $\pi_1' \leftrightarrow_{V_i} \pi_2'$ $(i = 1, 2)$ *implies* $\pi_1' \leftrightarrow_{V_1 \cup V_2} \pi_2'$;

(v) *for each path* π_{s_1} *of* $\mathcal{M}_1$ *there is a path* π_{s_2} *of* $\mathcal{M}_2$ *s.t.* $\pi_{s_1} \leftrightarrow_{V_1} \pi_{s_2}$, *and vice versa.*

3. It removes sp from $L(s)$ for every $s \in S$. Note that $L(s_4) - \{sp\} = L(s_2)$.

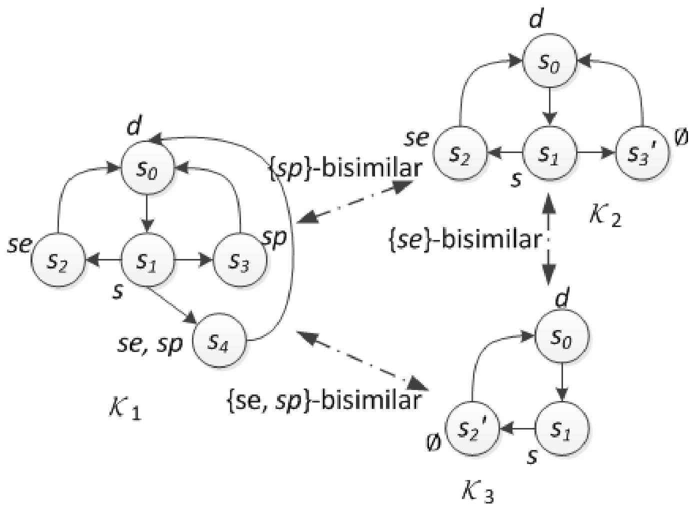


Figure 2: V-bisimulation between K-structures

In Proposition 1, properties (i) to (iii) are the standard properties for V-bisimulation. Property (iv) shows that if a structure is V_1 and V_2-bisimilar with the other two structures, respectively, then those two structures are $V_1 \cup V_2$-bisimilar. For an example, see Figure 2. This property is crucial for forgetting. And last, (v) says that if two structures are V_1-bisimilar, then they are V_2-bisimilar for any V_2 with $V_1 \subseteq V_2 \subseteq \mathcal{A}$.

Intuitively, if two structures are V-bisimilar, then they satisfy the same CTL formula φ that does not contain any atoms in V, i.e., $\text{IR}(\varphi, V)$. This idea has been formalized and shown in the following theorem.

Theorem 4 *Let $V \subseteq \mathcal{A}$, $\mathcal{K}_i$ $(i = 1, 2)$ be two structures s.t. $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ and ϕ be a CTL formula with $IR(\phi, V)$. Then $\mathcal{K}_1 \models \phi$ if and only if $\mathcal{K}_2 \models \phi$.*

Please note that the formula ϕ in the above theorem must have no index. Otherwise, the index function in $\mathcal{I}$-structures may affect the satisfiability. For instance, let $\phi = \text{E}_{\langle 1 \rangle}\text{X}p$, $\mathcal{K} = (\mathcal{M}, s)$ and $\mathcal{M} = (S, R, L, [\_], s_0)$ where $S = \{s_0, s_1\}$, $L(s_0) = \emptyset$, $L(s_1) = \{p\}$, $R = \{(s_0, s_1), (s_0, s_0), (s_1, s_1), (s_1, s_0)\}$ and $[1] = \{(s_0, s_1), (s_1, s_1)\}$. It is clear $\mathcal{K} \models \phi$. Let $\mathcal{K}' = (\mathcal{M}', s)$ and $\mathcal{M}' = (S, R, L, [\_]', s_0)$ with $[1]' = \{(s_0, s_0), (s_1, s_1)\}$. It is evident $\mathcal{K} \leftrightarrow_{\{q\}} \mathcal{K}'$ and $\text{IR}(\phi, \{q\})$. However, $\mathcal{K}' \not\models \phi$.

It is not difficult to check that, for the formulas $\varphi_1 = d \wedge \text{EF}se \wedge \text{AG}(se \rightarrow \text{AX}d)$ and $\varphi_2 = d \wedge \text{AX}se$, they are $\{sp\}$-irrelevant and the structures $\mathcal{K}_1$ and $\mathcal{K}_2$ in Figure 2 satisfy φ_1, but not φ_2.

Definition 3 (bisimular equivalence) *Let $V \subseteq \mathcal{A}$. Two formulas φ and ψ are V-*bisimularly equivalent*, written $\varphi \equiv_V \psi$, whenever, for every $\mathcal{K} \models \varphi$, there exists $\mathcal{K}' \models \psi$ such that $\mathcal{K} \leftrightarrow_V \mathcal{K}'$, and vice versa.*

The below lemma follows easily from Definitions 2 and 3, and Proposition 1.

Lemma 4 *For any $V \subseteq \mathcal{A}$, $\leftrightarrow_V$ and $\equiv_V$ are equivalent relations.*

Proof: The relation $\leftrightarrow_V$ is transitive by (i) of Proposition 1. It is obviously reflexive and symmetric. Thus, it is an equivalent relation.

The relation $\equiv_V$ is obviously reflexive and symmetric. Suppose that $\varphi \equiv_V \psi$ and $\psi \equiv_V \xi$. It follows that, for any $\mathcal{K} \models \varphi$, there exists $\mathcal{K}' \models \psi$ *s.t.* $\mathcal{K}' \leftrightarrow_V \mathcal{K}$ by $\varphi \equiv_V \psi$, and there exists $\mathcal{K}'' \models \xi$ *s.t.* $\mathcal{K}' \leftrightarrow_V \mathcal{K}''$ by $\psi \equiv_V \xi$. Since $\leftrightarrow_V$ is an equivalent relation, we have $\mathcal{K} \leftrightarrow_V \mathcal{K}''$. Similarly, for any $\mathcal{K}'' \models \xi$, there exists $\mathcal{K} \models \varphi$ *s.t.* $\mathcal{K}'' \leftrightarrow_V \mathcal{K}$. It implies the transitivity of $\equiv_V$. Thus, it is an equivalent relation. ■

p

The bellow corollary follows from the above definition and Proposition 1.

Corollary 5 *Let V, V_1, V_2 be subsets of $\mathcal{A}$, φ and ψ be formulas.*

(i) If $\varphi \equiv \psi$ then $\varphi \equiv_V \psi$.

(ii) If both φ and ψ contain no index and $\varphi \equiv_\emptyset \psi$ then $\varphi \equiv \psi$.

(iii) If $\varphi \equiv_{V_i} \psi$ $(i = 1, 2)$ then $\varphi \equiv_{V_1 \cup V_2} \psi$.

(iv) If $\varphi \equiv_{V_1} \psi$ and $V_1 \subseteq V_2$ then $\varphi \equiv_{V_2} \psi$.

Please note that the condition "φ and ψ contain no index" of (ii) is necessary. Otherwise let $\varphi = \mathrm{E}_{\langle 1 \rangle}\mathrm{X}p$ and $\psi = \mathrm{E}_{\langle 2 \rangle}\mathrm{X}p$. One can verify $\varphi \equiv_\emptyset \psi$, but $\varphi \not\equiv \psi$.

Proposition 2 *Let φ be a CTL formula. Then $\varphi \equiv_U T_\varphi$ where $T_\varphi = \mathrm{SNF}^g_{\mathrm{CTL}}(\varphi)$ and $U = Var(T_\varphi) - Var(\varphi)$.*

4.2 Semantic Forgetting

In this subsection, we present the notion of forgetting in CTL and investigate its semantic properties. Let us start with the formal definition.

Definition 4 (CTL-Forgetting) *Let $V \subseteq \mathcal{A}$ and ϕ be a CTL formula. A CTL formula ψ with $Var(\psi) \cap V = \emptyset$ is a* result of forgetting V from ϕ*, if*

$$Mod(\psi) = \{\mathcal{K} \mid \exists \mathcal{K}' \in Mod(\phi) \;\; s.t. \;\; \mathcal{K}' \leftrightarrow_V \mathcal{K}\}.$$

According to the above definition, if both formulas ψ and ψ' are results of forgetting V from ϕ, then $Mod(\psi) = Mod(\psi')$. In this sense, the result of forgetting V from ϕ is unique up to equivalence whenever the result exists. In this case we denote the forgetting result by $\mathrm{F}_{\mathrm{CTL}}(\phi, V)$, which means that the result of forgetting V from ϕ is expressible in the logic CTL, unless explicitly stated otherwise. In what follows, we write $\mathrm{F}_{\mathrm{CTL}}(\phi, \{p\})$ as $\mathrm{F}_{\mathrm{CTL}}(\phi, p)$ for convenience.

Please note that the above definition does not assert the existence of the forgetting result. In fact, there are CTL formulas whose forgetting results do not exist. For instance, let p and x be two distinct propositions and $\varphi(p, x)$[4] be the conjunction of the following formulas (Maksimova, 1991):

$$\mathrm{AG}(\neg x \wedge \neg \mathrm{AG}p \to \neg \mathrm{AX} \neg x), \qquad \mathrm{AG}(\neg \mathrm{AX} \neg x \to \mathrm{AX}x),$$
$$\mathrm{AG}(\mathrm{AX}x \to \neg x \wedge \neg \mathrm{AG}p), \qquad \mathrm{AG}(x \to \neg \mathrm{AG}p), \qquad \mathrm{AG}(\mathrm{AFAG}p).$$

4. $\varphi(p, x)$ means the formula φ with $Var(\varphi) = \{p, x\}$.

Maksimova showed that $\varphi(p,x) \wedge \varphi(p,y) \models x \leftrightarrow y$ and there is no CTL formula ψ with $Var(\psi) = \{p\}$ such that $\varphi(p,x) \models x \leftrightarrow \psi$, viz., the logic CTL has no explicitly definability (*Beth property*). This implies the below assertion.

Proposition 3 $\mathrm{F}_{\mathrm{CTL}}(x \wedge \varphi(p,x), \{x\})$ *is not expressible in CTL.*

Proof: Let $\psi(p) = \mathrm{F}_{\mathrm{CTL}}(x \wedge \varphi(p,x), \{x\})$ be a CTL formula. We have
$\varphi(p,x) \wedge \varphi(p,y) \vdash x \leftrightarrow y$
$\Rightarrow \varphi(p,x) \wedge x \vdash \varphi(p,y) \rightarrow y$
$\Rightarrow \varphi(p,x) \wedge x \vdash \psi(p)$ and $\psi(p) \vdash \varphi(p,y) \rightarrow p(y)$ (Theorem 8)
$\Rightarrow \varphi(p,x) \vdash x \rightarrow \psi(p)$, and $\varphi(p,y) \vdash \psi(p) \rightarrow p(y)$ which implies $\varphi(p,x) \vdash \psi(p) \rightarrow p(x)$
$\Rightarrow \varphi(p,x) \vdash x \leftrightarrow \psi(p)$, a contradiction. ■

In fact, the existence (or expressibility) of forgetting results correspond to the *uniform interpolation* (or *Craig interpolation*) property. Formally, a logic $\mathcal{L}$ has uniform interpolation (or Craig interpolation) property if, for any formulas φ and ψ with $\varphi \vdash_{\mathcal{L}} \psi$, there exists a formula ξ such that $\varphi \vdash_{\mathcal{L}} \xi$, $\xi \vdash_{\mathcal{L}} \psi$ and $Var(\xi) \subseteq Var(\varphi) \cap Var(\psi)$. It is well-known that the uniform interpolation property fails for the temporal logics LTL, CTL and CTL* (Maksimova, 1991; D'Agostino, 2008). This is different from the bounded CTL which has only finite signature and finite states. The bounded CTL has finite model property, which implies that every finite model can be characterized by a CTL formula. This follows that the forgetting result for bounded CTL always exists (Feng et al., 2020).

The next proposition shows that, for propositional formulas, the above CTL-forgetting agrees with the classical forgetting (Lin & Reiter, 1994).

Theorem 6 *Let φ be a formula of propositional logic and $V \subseteq \mathcal{A}$, then*

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

It is noteworthy that the V-irrelevant is of crucial importance for computing SNC and WSC. Consider the $\psi = \varphi \wedge (q \leftrightarrow \alpha)$. If $\mathrm{IR}(\varphi \wedge \alpha, \{q\})$, then the result of forgetting q from ψ is φ. This property is described in the following lemma, and as we will later see in Section 6, it will become important (in reducing the SNC (WSC) of any CTL formula to the one of a proposition).

Lemma 5 *Let φ and α be two CTL formulas and $q \notin (Var(\varphi) \cup Var(\alpha))$. Then* $\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge (q \leftrightarrow \alpha), q) \equiv \varphi$.

In what follows, we list other properties of the forgetting operator. According to the definition of forgetting, the set of atoms to be forgotten should be forgotten as a whole. The following property guarantees that this can be achieved modularly by applying forgetting one by one to the atoms to be forgotten.

Proposition 4 (Modularity) *Given a CTL formula φ, a set V of atoms, and an atom p s.t. $p \notin V$, then*

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

The next property follows from the above proposition.

Corollary 7 (Commutativity) *Let φ be a CTL formula and $V_i \subseteq \mathcal{A}$ $(i = 1, 2)$. Then:*

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

The following properties show that the forgetting respects the basic semantic notions of logic. Below we show that they are also satisfied in our notion forgetting in CTL.

Proposition 5 *Let φ, φ_i, ψ_i $(i = 1, 2)$ be formulas in CTL and $V \subseteq \mathcal{A}$. We have*

(i) $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$ *is satisfiable iff φ is;*

(ii) *If $\varphi_1 \equiv \varphi_2$, then* $\mathrm{F}_{\mathrm{CTL}}(\varphi_1, V) \equiv \mathrm{F}_{\mathrm{CTL}}(\varphi_2, V)$*;*

(iii) *If $\varphi_1 \models \varphi_2$, then* $\mathrm{F}_{\mathrm{CTL}}(\varphi_1, V) \models \mathrm{F}_{\mathrm{CTL}}(\varphi_2, V)$*;*

(iv) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \vee \psi_2, V) \equiv \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \vee \mathrm{F}_{\mathrm{CTL}}(\psi_2, V)$*;*

(v) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \wedge \psi_2, V) \models \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \wedge \mathrm{F}_{\mathrm{CTL}}(\psi_2, V)$*;*

(vi) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \wedge \psi_2, V) \equiv \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \wedge \psi_2$ *if $IR(\psi_2, V)$;*

The next property shows that forgetting a set $V \subseteq \mathcal{A}$ from a formula with path quantifiers is equivalent to quantify the result of forgetting V from the formula with the same path quantifiers.

Proposition 6 (Homogeneity) *Let $V \subseteq \mathcal{A}$ and ϕ a CTL formula.*

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

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

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

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

(v) $\mathrm{F}_{\mathrm{CTL}}(\mathrm{AG}\phi, V) \equiv \mathrm{AG}\mathrm{F}_{\mathrm{CTL}}(\phi, V)$.

(vi) $\mathrm{F}_{\mathrm{CTL}}(\mathrm{EG}\phi, V) \equiv \mathrm{EG}\mathrm{F}_{\mathrm{CTL}}(\phi, V)$.

Proof: (i) ($\Rightarrow$) $(\mathcal{M}, s) \models \mathrm{F}_{\mathrm{CTL}}(\mathrm{AX}\phi, V)$
$\Rightarrow$ There is $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$ and $(\mathcal{M}', s') \models \mathrm{AX}\phi$
$\Rightarrow$ There is $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$, and $(\mathcal{M}', s'') \models \phi$ for every $(s', s'') \in R'$ $(R' \in \mathcal{M}')$
$\Rightarrow$ For every $(s, s_1) \in R$, there is $(s', s_1') \in R'$ and $(\mathcal{M}, s_1) \leftrightarrow_V (\mathcal{M}', s_1')$, and $(\mathcal{M}', s'') \models \phi$ for every $(s', s'') \in R'$ $(R \in \mathcal{M})$
$\Rightarrow$ For every $(s, s_1) \in R$, there is $(\mathcal{M}, s_1) \leftrightarrow_V (\mathcal{M}', s_1')$ and $(\mathcal{M}', s_1') \models \phi$
$\Rightarrow$ For every $(s, s_1) \in R$, $(\mathcal{M}, s_1) \models \mathrm{F}_{\mathrm{CTL}}(\phi, V)$
$\Rightarrow$ $(\mathcal{M}, s) \models \mathrm{AX}\mathrm{F}_{\mathrm{CTL}}(\phi, V)$.

($\Leftarrow$) $(\mathcal{M}, s) \models \mathrm{AX}\mathrm{F}_{\mathrm{CTL}}(\phi, V)$
$\Rightarrow$ For every $(s, s') \in R$, $(\mathcal{M}, s') \models \mathrm{F}_{\mathrm{CTL}}(\phi, V)$ $(R \in \mathcal{M})$
$\Rightarrow$ For every $(s, s') \in R$, there is $(\mathcal{M}, s') \leftrightarrow_V (\mathcal{M}', s'')$ and $(\mathcal{M}', s'') \models \phi$
$\Rightarrow$ For every $i \geq 0$, there is $(\mathcal{M}, s_i') \leftrightarrow_V (\mathcal{M}_i', s_i'')$ and $(\mathcal{M}_i', s_i'') \models \phi$ where $\{s_0', s_1', \ldots\} = \{s' \mid (s, s') \in R\}$ and $\mathcal{M}_i' = (S_i', R_i', L_i', [\_]_i', s_i'')$ (we assume $S_i' \cap S_j' = \emptyset$ when $i \neq j$)
$\Rightarrow$ $(\mathcal{M}^*, s) \leftrightarrow_V (\mathcal{M}, s)$ and $(\mathcal{M}^*, s) \models \mathrm{AX}\phi$ where $\mathcal{M}^* = (S^*, R^*, L^*, [\_], s)$ and

- $S^* = \{s\} \cup \bigcup_{i \geq 0} S'_i$,
- $R^* = \{(s, s''_i) \mid i \geq 0\} \cup \bigcup_{i \geq 0} R'_i$,
- $L^* = \bigcup_{i \geq 0} L'_i$ and $L^*(s) = L(s)$ where $L \in \mathcal{M}$.

$\Rightarrow (\mathcal{M}, s) \models \mathrm{F}_{\text{CTL}}(\mathrm{AX}\phi, V)$. ■

We are now in the position to investigate the forgetting postulates in CTL. Let φ be a CTL formula, $V \subseteq \mathcal{A}$ and φ' be a result of forgetting V from φ. The *forgetting postulates* are as follows:

(**W**) Weakening: $\varphi \models \varphi'$;

(**PP**) Positive Persistence: for any formula η, if $\mathrm{IR}(\eta, V)$ and $\varphi \models \eta$ then $\varphi' \models \eta$;

(**NP**) Negative Persistence : for any formula η, if $\mathrm{IR}(\eta, V)$ and $\varphi \not\models \eta$ then $\varphi' \not\models \eta$;

(**IR**) Irrelevance: $\mathrm{IR}(\varphi', V)$.

Intuitively, the postulate (**W**) says, forgetting weakens the original formula; the postulates (**PP**) and (**NP**) say that forgetting result does not affect formulas that are irrelevant to forgotten atoms; the postulate (**IR**) states that forgetting result is irrelevant to forgotten atoms.

Theorem 8 (Representation Theorem) *Let φ and φ' be CTL formulas and $V \subseteq \mathcal{A}$. The following statements are equivalent:*

(i) $\varphi' \equiv \mathrm{F}_{\text{CTL}}(\varphi, V)$,

(ii) $\varphi' \equiv \{\phi \mid \varphi \models \phi$ *and* $IR(\phi, V)\}$,

*(iii) Postulates (**W**), (**PP**), (**NP**) and (**IR**) hold if φ, φ' and V are as in (i) and (ii).*

5. A Resolution-based Method for Computing Forgetting

In this section, we will explore how the resolution rules (in Table 2) can be used to compute forgetting in CTL when it exists. The main idea is that first, we turn the given CTL formula into a set of $\mathrm{SNF}^g_{\text{CTL}}$ clauses, and then we compute all the possible resolvents on the forgotten atoms and eliminate all the clauses containing forgotten atoms. Finally, we translate the remanning $\mathrm{SNF}^g_{\text{CTL}}$ clauses back to CTL formulas.

Let T be a finite set of $\mathrm{SNF}^g_{\text{CTL}}$ clauses and p an atom. The *unfolding* of T on p, written $\mathrm{UF}(T, p)$, is the set T together with

$$\{\alpha \mid \alpha \text{ is a resolvent obtained from } T \text{ where the resolved literal } l \in \{p, \neg p\}\}.$$

For a set V of atoms, we define $\mathrm{UF}(T, \emptyset) = T$ and $\mathrm{UF}(T, \{p\} \cup V) = \mathrm{UF}(\mathrm{UF}(T, \{p\}), V)$. Intuitively, unfolding T on an atom p is to do exhaustive resolution on p. In other words, no more new $\mathrm{SNF}^g_{\text{CTL}}$ clause can be generated using any of the step resolution rules **SRES1-8** to resolve the literal p or $\neg p$, even after the applying of rewrite rules and eventuality resolution rules.

The next proposition shows that, for any CTL formula φ, removing the clauses mentioning atoms in V from $\textsc{uf}(T_\varphi, V)$ remains bisimular equivalence when discarding these atoms in V. In what follows we denote $\mathit{ERes}(\varphi, V) = \{\alpha \in \textsc{uf}(T_\varphi, V) \mid \mathit{Var}(\alpha) \cap V = \emptyset\}$.

Proposition 7 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$. Then $T_\varphi \equiv_U \mathrm{ERes}(\varphi, V)$ where $\underline{U = \textsc{uf}(T_\varphi, V) - \mathit{Var}(\varphi).}$*

Example 4 (Continued from Example 2) *Let $V = \{p, r\}$. Then $\textsc{uf}(T_\varphi, V \cup \{x, y, z\})$ consists of the following clauses, in addition to the ones in Example 2:*

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

After removing clauses from $\textsc{uf}(T_\varphi, V \cup \{x, y, z\})$ that contain some atoms in V, we have $\mathrm{ERes}(\varphi, V)$ consisting of the following clauses:

$$\begin{gathered}\boldsymbol{start} \to z, \quad \boldsymbol{start} \to f \vee m \vee q, \quad \boldsymbol{start} \to x \vee y, \quad \boldsymbol{start} \to q \vee x, \quad \boldsymbol{start} \to f \vee m \vee y,\\ \top \to f \vee m \vee \neg x, \quad \top \to q \vee f \vee m \vee \neg z, \quad \top \to f \vee m \vee \neg z \vee y,\\ \top \to q \vee x \vee \neg z, \quad \top \to x \vee y \vee \neg z, \quad \top \to q \vee \neg y, \quad z \to \textsc{af}x,\\ y \to \textsc{ax}(q \vee f \vee m), \quad y \to \textsc{ax}(x \vee q), \quad y \to \textsc{ax}(x \vee y), \quad y \to \textsc{ax}(f \vee m \vee y).\end{gathered}$$

We see that $\mathrm{ERes}(\varphi, V)$ have some clauses mentioning the refreshed atoms in T_φ, though no index is mentioned in the example.

With the follow lemma, we can reduce some atoms while preserving equi-satisfiability.

Lemma 6 (Generalised Ackermann's Lemma) *Let x be an atom, $\Delta = \{\top \to \neg x \vee C_1, \ldots, \top \to \neg x \vee C_n, x \to B_1, \ldots, x \to B_m\}$ be a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses in which the unique occurrence of x is negative, and Γ be a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses in which x occurs positively. Then we have*

$$\Gamma \cup \Delta \equiv_{\{x\}} \Gamma \left[x / \bigwedge (\{C_i \mid 1 \le i \le n\} \cup \{B_i \mid 1 \le i \le m\}) \right]. \tag{1}$$

In this case we define $\textsc{gal}(\Gamma \cup \Delta, \{x\}) = \Gamma\, [x / \bigwedge (\{C_i \mid 1 \le i \le n\} \cup \{B_i \mid 1 \le i \le m\})]$, and $\textsc{gal}(\Gamma \cup \Delta, \{x\}) = \Gamma \cup \Delta$ otherwise. For a set V of atoms, we define

$$\textsc{gal}(\Gamma \cup \Delta, V \cup \{x\}) = \textsc{gal}(\textsc{gal}(\Gamma \cup \Delta, \{x\}), V).$$

Example 5 (Continued from Example 4) *Firstly, we consider the atom x and $\Delta = \{\top \to f \vee m \vee \neg x\}$ and $\Gamma = \text{ERes}(\varphi, V) - \Delta$. Any clause in Γ that mentions x is positive w.r.t. x. Then we have $\Gamma[x/(f \vee m)]$ consisting of the following clauses:*

$$\begin{aligned}&\textbf{\textit{start}} \to z, \quad \textbf{\textit{start}} \to f \vee m \vee q, \quad \textbf{\textit{start}} \to f \vee m \vee y,\\ &\top \to q \vee f \vee m \vee \neg z, \quad \top \to f \vee m \vee y \vee \neg z, \quad \top \to q \vee \neg y, \quad z \to \text{AF}(f \vee m),\\ &y \to \text{AX}(q \vee f \vee m), \quad y \to \text{AX}(f \vee m \vee y).\end{aligned}$$

Next, we consider the atom z, $\Delta' = \{\top \to q \vee f \vee m \vee \neg z, \top \to f \vee m \vee y \vee \neg z, z \to \text{AF}(f \vee m)\}$ and $\Gamma' = \Gamma[x/(f \vee m)] - \Delta'$ in which the atom z occurs positively. Then we have $\Gamma'' = \Gamma'[z/(q \vee f \vee m) \wedge (f \vee m \vee y) \wedge \text{AF}(f \vee m)]$ consisting of

$$\begin{aligned}&\textbf{\textit{start}} \to (q \vee f \vee m) \wedge (f \vee m \vee y) \wedge \text{AF}(f \vee m), \quad \textbf{\textit{start}} \to f \vee m \vee q, \quad \textbf{\textit{start}} \to f \vee m \vee y,\\ &\top \to q \vee \neg y, \quad y \to \text{AX}(q \vee f \vee m), \quad y \to \text{AX}(f \vee m \vee y).\end{aligned}$$

It is tedious but not difficult to check that $\text{ERes}(\varphi, V) \equiv_{\{x,z\}} \Gamma''$. Please note that one can not apply the above process to Γ'' and the atom y since Γ'' has a clause which is neither positive w.r.t. y, nor negative w.r.t. y.

The following lemma shows that, we can translate a set of $\text{SNF}^g_{\text{CTL}}$ clauses into a set of CTL formulas while keeping equi-satisfiability.

Lemma 7 *Let $j \in \mathcal{I}$, ψ_i, φ_i $(1 \le i \le n)$ be CTL formulas. We have*

(i) $\{\psi_i \to \text{E}_{\langle j \rangle}\text{X}\varphi_i \mid 1 \le i \le n\} \equiv_\emptyset \{(\bigwedge_{i \in S} \psi_i) \to \text{EX}\left(\bigwedge_{i \in S} \varphi_i\right) \mid S \subseteq \{1, \dots, n\}\}$*, and*

(ii) $\{(\psi_1 \to \text{E}_{\langle j \rangle}\text{F}\varphi_1), (\psi_2 \to \text{E}_{\langle j \rangle}\text{X}\varphi_2)\}$ *is $\emptyset$-bisimular equivalent to*

$$(\psi_1 \to \varphi_1 \vee \text{EXEF}\varphi_1) \wedge (\psi_2 \to \text{EX}\varphi_2) \wedge (\psi_1 \wedge \psi_2 \to ((\varphi_1 \wedge \text{EX}\varphi_2) \vee \text{EX}(\varphi_2 \wedge \text{EF}\varphi_1))). \quad (2)$$

We denote the right side of $\equiv_\emptyset$ in (i) and (ii) by $rxi(\{\alpha_i \mid 1 \le i \le n\})$ and $rfi(\{\beta_1, \alpha_2\})$, respectively, where $\alpha_i = \psi_i \to \text{E}_{\langle j \rangle}\text{X}\varphi_i$ $(1 \le i \le n)$ and $\beta_1 = \psi_1 \to \text{E}_{\langle j \rangle}\text{F}\varphi_1$.

We are now in the position to present the algorithm of compute CTL forgetting, Algorithm 1. It accepts a CTL formula φ and returns a CTL theory Σ which is bisimular equivalent to φ when discarding the atoms occurring in $Var(\Sigma) - Var(\varphi) \cup V$.

Theorem 9 *Let φ be a CTL formula, $V \subseteq \mathcal{A}$, $\Sigma =$CTL-forget(φ, V) and $U = Var(\Sigma) - Var(\varphi)$. Then*

(i) $\Sigma \equiv_{V \cup U} \varphi$*, and*

(ii) $\Sigma \equiv \text{F}_{\text{CTL}}(\varphi, V)$ *if $U = \emptyset$.*

Example 6 (Continued from Example 5) *It is easy to see that CTL-forget$(\varphi, \{p, r\})$ consists of the following formulas*

$$\begin{aligned}&(q \vee f \vee m) \wedge (f \vee m \vee y) \wedge \text{AF}(f \vee m), \quad \top \to q \vee \neg y,\\ &y \to \text{AX}(q \vee f \vee m), \quad y \to \text{AX}(f \vee m \vee y).\end{aligned}$$

There are some sub-classes of CTL formulas whose forgetting always exist.

Proposition 8 *Let φ be a CTL formula, which does not contain temporal operators $Pt\mathcal{T}$ with $Pt \in \{\text{A}, \text{E}\}$ and $\mathcal{T} \in \{\text{U}, \text{G}\}$, and for each atom $p \in V$ if p and $\neg p$ appear in the same domain of temporal operator, then CTL-forget$(\varphi, V) \equiv \text{F}_{\text{CTL}}(\varphi, V)$.*

Algorithm 1: CTL-forget(φ, V)

```
   Input: A CTL formula $\varphi$ and a set $V$ of atoms
   Output: A conjunction of formulas
 1 if $\varphi \equiv \bot$ then return $\bot$;
 2 if $V = Var(\varphi)$ then return $\top$;
 3 $T_\varphi \leftarrow \mathrm{SNF}^g_{\mathrm{CTL}}(\varphi)$ ;                // Transforming $\varphi$ into $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses
 4 $\Sigma \leftarrow \mathrm{UF}(T_\varphi, V \cup U)$ where $U = Var(T_\varphi) - Var(\varphi)$;          // Unfolding
 5 $\Sigma \leftarrow ERes(\Sigma, V)$ ;       // Removing clauses which mention some atom in $V$
 6 $\Sigma \leftarrow \mathrm{GAL}(\Sigma, Var(\Sigma) - Var(\varphi))$ ;      // Reducing the remaining fresh atoms
   /* removing indexes from $\Sigma$                                            */
 7 foreach A-sometime clause $\alpha \in \Sigma$ do
 8     $\Gamma \leftarrow \emptyset$;
 9     foreach E-step clause $\beta \in \Sigma$ having the same index as $\alpha$ do
10         $\Gamma \leftarrow \Gamma \cup \{rfi(\alpha, \beta)\}$;
11     end
12     $\Sigma \leftarrow \Sigma \cup \Gamma - \{\alpha\}$;
13 end
14 foreach $\Delta \subseteq \Sigma$ s.t. $\Delta$ consists of all the clauses having the same index do
15     $\Sigma \leftarrow \Sigma - \Delta \cup \{rxi(\Delta)\}$;
16 end
17 Replacing each initial clause "start $\rightarrow \varphi$" by $\varphi$;
18 return $\Sigma$;
```

5.1 Termination and Complexity of the Algorithm

We know that for each CTL formula, transformation and resolution processes terminate (Zhang et al., 2014). Moreover, the Remove_atoms and the Remove_index (include T_{CTL}) processes also terminate because the set of clauses obtained from the resolution process is finite. Therefore, for any given CTL formula and a set V of atoms, we can conclude that Algorithm 1 will eventually terminate. We report the complexity of the algorithm in the following result.

Proposition 9 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$. The time and space complexity of Algorithm 1 are $O((m+1)2^{4(n+n')})$ where $n = |Var(\varphi)|$, $n' = |V'|$ (V' is the set of atoms introduced in the transformation process) and m is the number of indices introduced during transformation.*

Observe that m is not greater than the number of temporal operators in φ. This observation tells us that the complexity of our algorithm only depends on the number of atoms and temporal operators in φ.

6. Necessary and Sufficient Conditions

In this part, we present one of the applications of our work: namely, the *strongest necessary condition* (SNC) and the *weakest sufficient condition* (WSC) of a given CTL specification.

As aforementioned in the introduction, these notions (introduced by E. Dijkstra in (Dijkstra, 1975)) correspond to the *most general consequence* and the *most specific abduction* of a specification, respectively, and have been central to a wide variety of tasks and studies (see Related Work). Our contribution, in particular, will be on computing SNC and WSC via forgetting under a given initial structure and a set V of atoms. Let us give the formal definition.

Definition 5 (Sufficient and Necessary condition) *Let ϕ be a CTL formula (or an initial structure), ψ be a CTL formula, $V \subseteq Var(\phi)$, $q \in Var(\phi) - V$ and $Var(\psi) \subseteq V$.*

- *ψ is a* necessary condition *(NC in short) of q on V under ϕ if $\phi \models q \rightarrow \psi$.*
- *ψ is a* sufficient condition *(SC in short) of q on V under ϕ if $\phi \models \psi \rightarrow q$.*
- *ψ is a* strongest necessary condition *(SNC in short) of q on V under ϕ if it is an NC of q on V under ϕ, and $\phi \models \psi \rightarrow \psi'$ for any NC ψ' of q on V under ϕ.*
- *ψ is a* weakest sufficient condition *(WSC in short) of q on V under ϕ if it is an SC of q on V under ϕ, and $\phi \models \psi' \rightarrow \psi$ for any SC ψ' of q on V under ϕ.*

Note that if both ψ and ψ' are SNC (WSC) of q on V under ϕ, then $Mod(\psi) = Mod(\psi')$, i.e., ψ and ψ' have the same models. In this sense, the SNC (WSC) of q on V under ϕ is unique (up to semantic equivalence). Besides, we say that a formula φ is defined on a set V of atoms if $Var(\varphi) \subseteq V$.The following result shows that the SNC and WSC are in fact dual notions.

Proposition 10 (Dual) *Let V, q, φ, and ψ are defined as in Definition 5. Then, ψ is an SNC (a WSC) of q on V under φ iff $\neg\psi$ is a WSC (an SNC) of $\neg q$ on V under φ.*

In order to generalise the atom q in Definition 5 to arbitrary formulas, one can replace q (in the definition) with any formula α, and redefine V as a subset of $Var(\alpha) \cup Var(\phi)$.

It turns out that the previous notions of SNC and WSC for an atomic variable can be lifted to any formula, or, conversely, the SNC and WSC of any formula can be reduced to that of an atomic variable, as the following result shows.

Proposition 11 *Let Γ and α be two formulas, $V \subseteq Var(\alpha) \cup Var(\Gamma)$ and q be a new proposition not occurring in Γ and α. Then, a formula φ of V is the SNC (WSC) of α on V under Γ iff it is the SNC (WSC) of q on V under $\Gamma' = \Gamma \cup \{q \leftrightarrow \alpha\}$.*

To give an intuition for WSC, we give the following example. The intuition for SNC is dual.

Example 7 (Continued from Example 3) *Recall $\mathcal{K}_2$ in Figure 2. Let $\psi = \text{EX}(s \wedge (\text{EX}se \vee \text{EX}\neg d))$, $\varphi = \text{EX}(s \wedge \text{EX}\neg d)$, $\mathcal{A} = \{d, s, se\}$ and $V = \{s, d\}$, then we can check that the WSC of ψ on V under $\mathcal{K}_2$ is φ.*

We verify this result by the following two steps:

(i) Observe that $\varphi \models \psi$ and $Var(\varphi) \subseteq V$. Besides, $(\mathcal{M}, s_0) \models \varphi \wedge \psi$, hence $\mathcal{K}_2 \models \varphi \rightarrow \psi$, which means φ is an SC of ψ on V under $\mathcal{K}_2$,

(ii) *We will show that for any SC φ' of ψ on V under $\mathcal{K}_2$, we have $\mathcal{K}_2 \models \varphi' \to \varphi$. It is easy to see that if $\mathcal{K}_2 \not\models \varphi'$, then $\mathcal{K}_2 \models \varphi' \to \varphi$, trivially. Now let's assume $\mathcal{K}_2 \models \varphi'$. In this case, we have $\varphi' \models \psi$ since φ' is an SC of ψ on V under $\mathcal{K}_2$. Therefore, there is $\varphi' \models \mathrm{EX}(s \wedge \phi)$, in which ϕ is a formula s.t. $\phi \models \mathrm{EX}se \vee \mathrm{EX}\neg d$. And then $\phi \models \mathrm{EX}\neg d$ since $IR(\varphi', \overline{V})$. Hence, $\varphi' \models \varphi$ and we get $\mathcal{K}_2 \models \varphi' \to \varphi$, as desired.*

The following result establishes the bridge between forgetting and the notion of SNC (WSC) which are central to our contribution.

Theorem 10 *Let φ be a formula, $V \subseteq Var(\varphi)$ and $q \in Var(\varphi) - V$.*

(i) $\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge q,\ (Var(\varphi) \cup \{q\}) - V)$ *is an SNC of q on V under φ.*

(ii) $\neg\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge \neg q,\ (Var(\varphi) \cup \{q\}) - V)$ *is a WSC of q on V under φ.*

Following Theorem 10, assume that $\beta = \mathrm{F}_{\mathrm{CTL}}(\varphi \wedge q, (Var(\varphi) \cup \{q\}) - V)$. Then, $\varphi \wedge q \models \beta$ by the definition of forgetting. Moreover, $\varphi \wedge q \models \beta$, and then β is an NC of q on V under φ.

In addition, for any ψ with $\mathrm{IR}(\psi, (Var(\varphi) \cup \{q\}) - V)$ and $\varphi \wedge q \models \psi$, we have $\beta \models \psi$ by Theorem 4. Therefore, β is the SNC of q on V under φ. This shows the intuition of how the SNC can be obtained from the forgetting.

Since any finite initial K-structure can be characterized by a CTL formula, by Theorem 10 one can obtain the SNC (and its dual WSC) of a target property (a formula) under an initial K-structure just by forgetting. This is shown in the following result.

Theorem 11 *Let $\mathcal{K} = (\mathcal{M}, s)$ be an initial structure with $\mathcal{M} = (S, R, L, s_0)$ on the set $\mathcal{A}$ of atoms, $V \subseteq \mathcal{A}$ and $q \in V' = \mathcal{A} - V$. Then,*

(i) *the SNC of q on V under $\mathcal{K}$ is $\mathrm{F}_{\mathrm{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge q, V')$.*

(ii) *the WSC of q on V under $\mathcal{K}$ is $\neg\mathrm{F}_{\mathrm{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge \neg q, V')$.*

7. Experiments

We have implemented the resolution-based algorithm (i.e., Algorithm 1) presented in Section 5, in Prolog. We have evaluated the performance of our algorithm on the task of computing the SNC and forgetting results on randomly generated CTL formulas and formulas from CTL-RP benchmarks [5], and a randomly generated set of CTL formulas. We ran our experiments on a Linux server running Fedora 21 with 8 Intel Cores i7 (4770 K, 3.50 GHz) and 32 GB RAM. The implementation and the experimental results are publicly available.[6]

In the experiments, the length (or size) of a 3-*CNF* formula φ, denoted by $|\varphi|$, refers to the number of clauses in φ.

5. `https://sourceforge.net/projects/ctlrp/`
6. `https://github.com/fengrenyan/forgetting-in-CTL/tree/main/Appendix`

Table 3: CPU time (in seconds) of computing $ERes(\varphi, V)$.

φ \ $\lvert V\rvert$	1	2	3	4
`s001`	0.0505	0.1053	0.2259	0.3680
`s002`	0.3645	1.0416	5.6372	10.0184
`s003`	97.5341	71.5396	190.1157	423.5793
`s004`	77.5086	77.4246	101.1284	118.7461
`s001-3`	681.2883	613.1859	1617.047	2356.949

7.1 General indicators

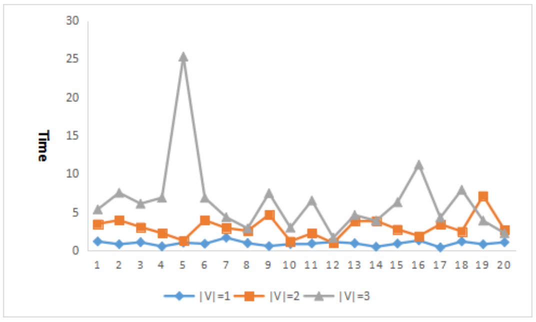
Many formulas in the benchmarks in the CTL-RP are unsatisfiable. In such a case, forgetting results are trivially $\bot$. To compute non-contradicting forgetting results, some subformulas of the benchmark are chosen. In particular, five CTL formulas `s001, s002, s003, s004`, and `s001-3`, each containing four atoms, are tested to forget a (random) number of atoms. In particular, the formula `s001` (resp., `s002, s003, s004)` is the conjunction of the first two subformulas of `s001.ctl` (resp., `s002.ctl, s003.ctl, s004.ctl`) from the benchmark `sample01` of CTL-RP. The formula `s001-3` is the conjunction of the first three subformulas of `s001.ctl`.

The CPU time cost (in seconds) of computing $ERes(\varphi, V)$ for the above formulas and a number of random atoms occurring in φ is reported in Table 3, where $|V|$ ranges from 1 to 4. Observe that the more complex the formula, and the more the number of atoms to be forgotten, and hence the harder the computation.

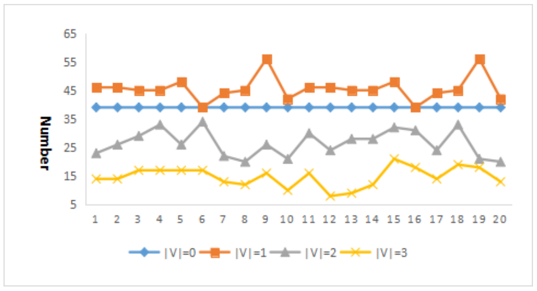
We also conducted experiments on formulas (on a set of atoms $\mathcal{A}$ with $|\mathcal{A}| = 4$) of the form $\varphi_1 \wedge \text{AX}\varphi_2 \wedge \text{EX}\varphi_3$ where φ_i $(i = 1, 2, 3)$ are randomly generated 3-*CNF* clauses, which have the same size, of propositional logic. It computes $\text{F}_{\text{CTL}}(\varphi, V)$ in a few seconds for $|V| \in \{1, 2, 3\}$ and $|\varphi_i| \in \{12, 16\}$. In particular, what is shown is the number of $\text{SNF}^g_{\text{CTL}}$ clauses after the Removing_atoms process. The horizontal axis (i.e., 1 to 20) denotes 20 formulas, $|V| = x$ denotes the number of atoms to be forgotten; that is, x ranges from 0 to 3 (with $|V| = 0$ expressing the result of the transformation process, i.e., the case no atom is being forgotten). The vertical axis stands for time in seconds in Figure 3(a) and Figure 4(a) needed for the removal of atoms, and the number in Figure 3(b) and Figure 4(b) stands for the number of $\text{SNF}^g_{\text{CTL}}$ clauses.

For the length 12 of φ_i, the CPU time cost (in seconds) and the remaining number of $\text{SNF}^g_{\text{CTL}}$ clauses after the Removing_atoms process of computing $ERes(\varphi, V)$ are reported in (a) and (b) of Figure 3, respectively, where $|V|$ ranges from 1 to 3. Figure 4 illustrates the same analysis for the length 16.

Observe that in all these figures, our Prolog program computes $\text{F}_{\text{CTL}}(\varphi, V)$ in a few seconds for $|V| \in \{1, 2, 3\}$. Moreover, as we forget more atoms from φ, the number of $\text{SNF}^g_{\text{CTL}}$ clauses obtained after the Removing_atoms process will also be decreasing.

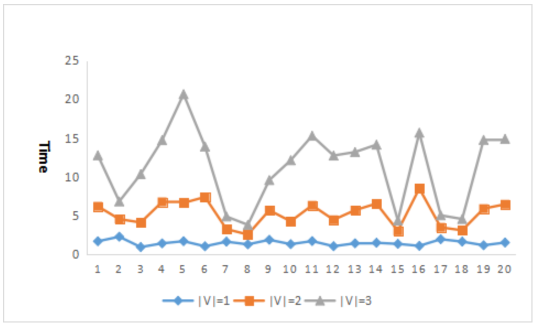


(a) Time used to compute forgetting

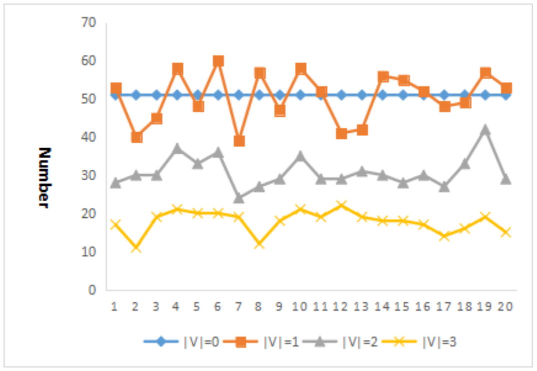


(b) The number of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses

Figure 3: The results of the cases for $\varphi_i = 12$.



(a) Time used to compute forgetting



(b) The number of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses

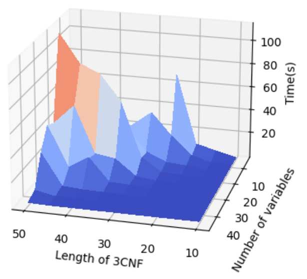
Figure 4: The results of the cases for $\varphi_i = 16$.

7.2 Computing SNC by Forgetting

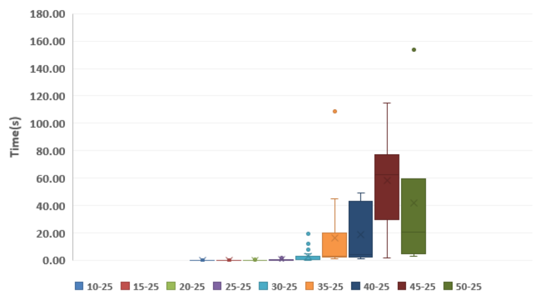
This section presents the empirical evaluation regarding *time performance* and *computing performance* of computing the SNC using forgetting. The time performance refers to how much time it takes to calculate SNC, while the computing performance refers to the percentage of the formulas that an SNC exists for, in a given set of formulas defined on a set $\mathcal{A}$ of atoms where $|\mathcal{A}| = 50$.

Let φ be a 3-*CNF* formula, V be a set of atoms, and q be an atom with $q \in Var(\varphi) - V$. It follows from Theorem 10 that if the result of forgetting the atoms in $Var(\varphi \wedge q) - V$ from $\varphi \wedge q$ exists, then the SNC of q on V under φ always exists. Accordingly, the SNC always exists in PL. Figure 5(a) shows the time performance.

In Figure 5(a), the numbers of atoms in V and clauses in φ are reported in Y-axis and X-axis, respectively: $|V| \in \{5, 10, 15, \ldots, 40, 45\}$ and $|\varphi| \in \{10, 15, \ldots, 45, 50\}$. Z-axis indicates the average time used to calculate 20 formulas. We see that the formulas with shorter lengths take less time. Moreover, the longer $|V|$ gets, i.e., the fewer atoms you have to forget, hence the lesser time it requires. The time spent to compute the SNC in the case of $|\varphi| \in \{10, \ldots, 25\}$ are around its median, while there are some singularities when $|\varphi| \in \{30, 35, 50\}$ (See Figure 6(b)).



(a) Time performance

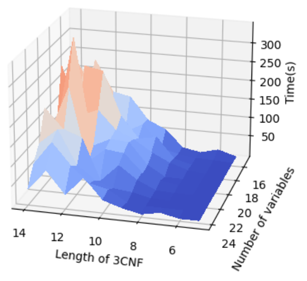


(b) The Boxplots of $|V| = 25$ and $|\varphi| \in \{10, 15, \ldots, 50\}$

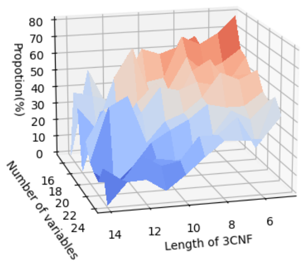
Figure 5: The time performance of computing SNC in PL.

Figure 6 shows the time performance (a) and the computing perormance (b) for a CTL formula φ of the form $\varphi_1 \wedge \text{AX}\varphi_2 \wedge \text{EX}\varphi_3$ where φ_i $(i = 1, 2, 3)$ are randomly generated 3-*CNF* clauses in PL, and X-axis corresponds to $|\varphi_i| \in \{5, 6, 7, \ldots, 14\}$, whereas Y-axis corresponds to $|V| \in \{15, 16, \ldots, 23, 24\}$. It is worth noting that 40 formulas have been calculated for every situation here, i.e., Z-axis indicates the average time used to calculate it.

In Figure 6(a), we observe the same behaviour and the intuition as in Figure 5(a). Figure 6(b), however, shows that the SNC exists for most formulas, especially, about 80% when $|\varphi_i| = 5$ and $|V| = 16$. Furthermore, the boxplots in Figure 7 indicate that given a formula, the more atoms that is needed to be forgotten, the more time it takes.



(a) Time performance



(b) Computing performance

Figure 6: The performances of computing SNC in CTL.

Overall, our algorithm can compute the SNC (WSC) in most of the cases and is efficient when the number of atoms that is to be forgotten is small, or the length of the formula is short. Moreover, the number of $\text{SNF}^g_{\text{CTL}}$ clauses decreases as we forget more atoms.

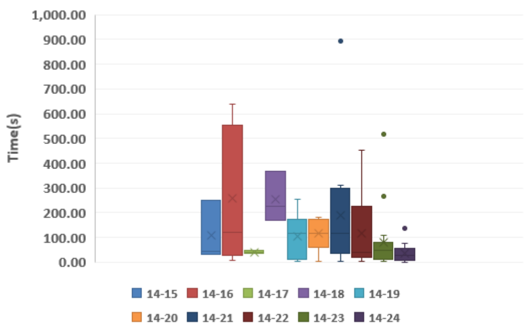


Figure 7: Time Boxplots of computing SNC in CTL

8. Concluding Remarks

8.1 Summary

In this paper, we presented the notion of forgetting for CTL which enables computing WSC (SNC) of specifications and knowledge update. To achieve this, we introduced the notions of V-bisimulation and $\langle V, I\rangle$-bisimulation, which can be considered a simple variable-based generalization of classical bisimulation. Furthermore, we studied formal properties (i.e., homogeneity, modularity, and commutativity) of forgetting and a resolution-based method to compute forgetting. In particular, we have shown that the notion of forgetting in bounded CTL satisfies the existing postulates of forgetting. Moreover, we proposed a model-based algorithm to compute the forgetting of a given formula and a set of variables in bounded CTL. We coupled our approach with a syntactic one and introduced a resolution-based algorithm which has been implemented in Prolog. Our experimental results show that it is relatively efficient to compute forgetting and can be used to compute the SNC in most cases.

8.2 Discussion and Future Avenues of Research

Our study explores how to abstract the information, which is critical to designing a system, from a specification inthe context of a CTL formula. In doing so, we have given a forgetting-based method from both syntax (resolution-based) and semantical (model-based) points. Based on this, we demonstrated how can forgetting be used to computed WSC (SNC) and knowledge update.

We have shown that our forgetting is not close in CTL. This analysis supports the theory that the CTL is a failure of uniform interpolation. Nevertheless, we show that there is a special fragment of CTL in which the forgetting is close. It is not trivial to find more fragments in the future since we believe most properties of a system can be expressed by those CTL fragments. And in this case, distilling some information from it by forgetting is easy. Moreover, the results indicate that forgetting is close in bounded CTL, which is the fact that the result of forgetting some atoms from a CTL formula can always be expressed by a CTL formula.

There remain some interesting directions which deserve further investigation: First, although the initial results for the complexity analysis of some problems (including model checking and Entailment of forgetting) for the $\mathrm{CTL}_{\mathrm{AF}}$ fragment was reported in our earlier work KR (Feng et al., 2020), a thorough investigation is still needed for the general case (namely, CTL). Second, we did not yet resolve whether the resolution-based method is complete for forgetting. Third, we still do not explore how can the WSC obtained by our algorithm be used to update the original model system. More specifically, when a transition system $\mathcal{M}$ does not satisfy a specification ϕ, one can evaluate the weakest sufficient condition ψ over a signature V under which $\mathcal{M}$ satisfies ϕ, viz., $\mathcal{M} \models \psi \to \phi$ and ψ mentions only atoms from V. It is worthwhile to explore how the condition ψ can guide the design of a new transition system $\mathcal{M}'$ satisfying ϕ. Last but not least, different fragments, in which the result of forgetting always exists, is of central interest for the future research avenue.

Bibliography

Ackermann, W. (1935). Untersuchungen über das eliminationsproblem der mathematischen logik. *Mathematische Annalen*, *110*(1), 390–413.

Baier, C., & Katoen, J.-P. (2008). *Principles of Model Checking*. The MIT Press.

Baral, C., & Zhang, Y. (2005). Knowledge updates: Semantics and complexity issues. *Artificial Intelligence*, *164*(1-2), 209–243.

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

Bolotov, A. (2000). *Clausal resolution for branching-time temporal logic*. Ph.D. thesis, Manchester Metropolitan University.

Browne, M. C., Clarke, E. M., & Grümberg, O. (1988). Characterizing finite Kripke structures in propositional temporal logic. *Theoretical Computer Science*, *59*(1-2), 115–131.

Clarke, E. M., & Emerson, E. A. (1981). Design and synthesis of synchronization skeletons using branching time temporal logic. In *Workshop on Logic of Programs*, pp. 52–71.

Clarke, E. M., Emerson, E. A., & Sistla, A. P. (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Program. Lang. Syst.*, *8*(2), 244–263.

D'Agostino, G. (2008). Interpolation in non-classical logics. *Synthese*, *164*(3), 421–435.

Dailler, S., Hauzar, D., Marché, C., & Moy, Y. (2018). Instrumenting a weakest precondition calculus for counterexample generation. *Journal of logical and algebraic methods in programming*, *99*, 97–113.

Delgrande, J. P. (2017). A knowledge level account of forgetting. *Journal of Artificial Intelligence Research*, *60*, 1165–1213.

Dijkstra, E. W. (1975). Guarded commands, Nondeterminacy and Formal Derivation of Programs. *Commun. ACM*, *18*(8), 453–457.

Doherty, P., Lukaszewicz, W., & Szalas, A. (2001). Computing strongest necessary and weakest sufficient conditions of first-order formulas. In Nebel, B. (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pp. 145–154. Morgan Kaufmann.

Eiter, T., & Wang, K. (2008). Semantic forgetting in answer set programming. *Artificial Intelligence*, *172*(14), 1644–1672.

Emerson, E. A. (1990). Temporal and modal logic. In *Formal Models and Semantics*, pp. 995–1072. Elsevier.

Fagin, R., Kuper, G. M., Ullman, J. D., & Vardi, M. Y. (1986). Updating logical databases. *Advanced Scientific Computing Research*, *3*, 1–18.

Fang, L., Wan, H., Liu, X., Fang, B., & Lai, Z. (2018). Dependence in propositional logic: Formula-formula dependence and formula forgetting - application to belief update and conservative extension. In McIlraith, S. A., & Weinberger, K. Q. (Eds.), *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pp. 1835–1844. AAAI Press.

Feng, R., Acar, E., Schlobach, S., Wang, Y., & Liu, W. (2020). On sufficient and necessary conditions in bounded CTL: A forgetting approach. In Calvanese, D., Erdem, E., & Thielscher, M. (Eds.), *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning, KR 2020, Rhodes, Greece, September 12-18, 2020*, pp. 361–370.

Katsuno, H., & Mendelzon, A. O. (1991). On the difference between updating a knowledge base and revising it. In Allen, J. F., Fikes, R., & Sandewall, E. (Eds.), *Proceedings of the 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR'91). Cambridge, MA, USA, April 22-25, 1991*, pp. 387–394. Morgan Kaufmann.

Konev, B., Ludwig, M., Walther, D., & Wolter, F. (2012). The logical difference for the lightweight description logic $\mathcal{EL}$. *Journal of Artificial Intelligence Research*, *44*, 633–708.

Konev, B., Walther, D., & Wolter, F. (2009). Forgetting and uniform interpolation in large-scale description logic terminologies. In Boutilier, C. (Ed.), *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pp. 830–835.

Lin, F. (2001). On strongest necessary and weakest sufficient conditions. *Artificial Intelligence*, *128*(1-2), 143–159.

Lin, F. (2003). Compiling causal theories to successor state axioms and strips-like systems. *Journal of Artificial Intelligence Research*, *19*, 279–314.

Lin, F., & Reiter, R. (1994). Forget it. In *Working Notes of AAAI Fall Symposium on Relevance*, pp. 154–159.

Lutz, C., & Wolter, F. (2011). Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *Proceedings of IJCAI'11*, pp. 989–995.

Maksimova, L. (1991). Temporal logics of "the next" do not have the beth property. *Journal of Applied Non-Classical Logics*, *1*, 73–76.

Visser, A. (2017). *Uniform Interpolation and Layered Bisimulation*, p. 139C164. Lecture Notes in Logic. Cambridge University Press.

Wang, Y., Wang, K., & Zhang, M. (2013). Forgetting for answer set programs revisited. In *Proceedings of IJCAI'13*, pp. 1162–1168, Beijing, China. IJCAI/AAAI.

Wang, Y., Zhang, Y., Zhou, Y., & Zhang, M. (2014). Knowledge forgetting in answer set programming. *Journal of Artificial Intelligence Research*, *50*, 31–70.

Wang, Z., Wang, K., Topor, R. W., & Pan, J. Z. (2010). Forgetting for knowledge bases in DL-Lite. *Annuals of Mathematics and Artificial Intelligence*, *58*(1-2), 117–151.

Wong, K.-S. (2009). *Forgetting in Logic Programs*. Ph.D. thesis, The University of New South Wales.

Woodcock, J. C., & Morgan, C. (1990). Refinement of state-based concurrent systems. In *International Symposium of VDM Europe*, pp. 340–351. Springer.

Zhang, L., Hustadt, U., & Dixon, C. (2009). A refined resolution calculus for CTL. In *International Conference on Automated Deduction*, pp. 245–260. Springer.

Zhang, L., Hustadt, U., & Dixon, C. (2014). A resolution calculus for the branching-time temporal logic CTL. *ACM Transactions on Computational Logic (TOCL)*, *15*(1), 1–38.

Zhang, Y., & Foo, N. Y. (2006). Solving logic program conflict through strong and weak forgettings. *Artificial Intelligence*, *170*(8-9), 739–778.

Zhang, Y., & Zhou, Y. (2009). Knowledge forgetting: Properties and applications. *Artificial Intelligence*, *173*(16-17), 1525–1537.

Zhao, Y., & Schmidt, R. A. (2017). Role forgetting for ALCOQH(Δ)-ontologies using an ackermann-based approach. In *Proceedings of IJCAI'17*, pp. 1354–1361.

Zhao, Y., Schmidt, R. A., Wang, Y., Zhang, X., & Feng, H. (2020). A practical approach to forgetting in description logics with nominals. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 3073–3079. AAAI Press.

Appendix A. Notations for the proofs

For convenience, we give the following notations:

(1) By $T \wedge \varphi$, we mean $\bigwedge_{\psi \in T} \psi \wedge \varphi$, where T is a finite set of formulas.

(2) Let C and C' be two formulas, we say C and C' are *resolvable* if there is a resolution rule using C and C' as the premises on some given atom.

(3) For any two $\mathcal{I}$-structures $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ with $i \in \{1,2\}$, if $(\mathcal{M}_1, s_0^1) \leftrightarrow_{\langle V,I\rangle} (\mathcal{M}_2, s_0^2)$, then we have $(\mathcal{M}'_1, s_0^1) \leftrightarrow_V (\mathcal{M}'_2, s_0^2)$ under CTL, where $\mathcal{M}'_i = (S_i, R_i, L_i, s_0^i)$ with $i \in \{1,2\}$.

(4) And when the $\mathcal{M}_i$ $(i = 1,2)$ is clear from the context, we also use $(s_1, s_2) \in \mathcal{B}_n$ to replace $((\mathcal{M}_1, s_1), (\mathcal{M}_2, s_2)) \in \mathcal{B}_n$.

Appendix B. Proofs for Section 4

Proposition ?? Let $V_1, V_2 \subseteq \mathcal{A}$, $I_1, I_2 \subseteq \mathcal{I}$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ with $i \in \{1,2,3\}$ be $\mathcal{I}$-structures s.t. $\mathcal{K}_1 \leftrightarrow_{\langle V_1, I_1\rangle} \mathcal{K}_2$ and $\mathcal{K}_2 \leftrightarrow_{\langle V_2, I_2\rangle} \mathcal{K}_3$. Then:

(i) $\mathcal{K}_1 \leftrightarrow_{\langle V_1 \cup V_2, I_1 \cup I_2\rangle} \mathcal{K}_3$;

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

Proof: (i) Let $\mathcal{M}_j = (S_i, R_i, L_i, [\_]_i, s_0^i)$ $(i = 1,2,3)$, $s_1 \leftrightarrow_{\langle V_1, I_1\rangle} s_2$ via a binary relation $\mathcal{B}$, and $s_2 \leftrightarrow_{\langle V_2, I_2\rangle} s_3$ via a binary relation $\mathcal{B}''$. Let $\mathcal{B}' = \{(w_1, w_3) \mid (w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''\}$. To prove $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$, we will prove that $\mathcal{B}'$ is a $\langle V_1 \cup V_2, I_1 \cup I_2\rangle$-bisimulation containing (s_1, s_3). It's evident that $(s_1, s_3) \in \mathcal{B}'$ due to $(s_1, s_2) \in \mathcal{B}$ and $(s_2, s_3) \in \mathcal{B}''$.

For all $(w_1, w_3) \in \mathcal{B}'$ and $j \notin (I_1 \cup I_2)$:

(a) there exists $w_2 \in S_2$ s.t. $(w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''$, and for all $q \notin V_1$, $q \in L_1(w_1)$ iff $q \in L_2(w_2)$ by $w_1 \leftrightarrow_{V_1} w_2$ and for all $q' \notin V_2$, $q' \in L_2(w_2)$ iff $q' \in L_3(w_3)$ by $w_2 \leftrightarrow_{V_2} w_3$. Then we have for all $r \notin V_1 \cup V_2$, $r \in L_1(w_1)$ iff $r \in L_3(w_3)$.

(b) $\forall u_1 \in S_1$, if $(w_1, u_1) \in R_1$, then $\exists u_2 \in S_2$ s.t. $(w_2, u_2) \in R_2$ 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 $\exists u_3 \in S_3$ s.t. $(w_3, u_3) \in R_3$ and $(u_2, u_3) \in \mathcal{B}''$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

(c) $\forall u_3 \in S_3$, if $(w_3, u_3) \in R_3$, then $\exists u_2 \in S_2$ s.t. $(w_2, u_2) \in R_2$ and $(u_2, u_3) \in \mathcal{B}_2$; and then $\exists u_1 \in S_1$ s.t. $(w_1, u_1) \in R_1$ and $(u_1, u_2) \in \mathcal{B}$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

(d) $\forall (w_1, r_1) \in [j]_1$, $\exists (w_2, r_2) \in [j]_2$ s.t. $(w_1, w_2) \in \mathcal{B}$ and $(r_1, r_2) \in \mathcal{B}$; and then $\exists (w_3, r_3) \in [j]_3$ s.t. $(w_2, w_3) \in \mathcal{B}''$ and $(r_2, r_3) \in \mathcal{B}''$. Therefore, $(w_1, w_3) \in \mathcal{B}''$ and $(r_1, r_3) \in \mathcal{B}''$.

(e) similar to (d), we can show that $\forall (w_3, r_3) \in [j]_3$, $\exists (w_1, r_1) \in [j]_1$ s.t. $(w_1, w_3) \in \mathcal{B}''$ and $(r_1, r_3) \in \mathcal{B}''$.

(ii) Supposing $\mathcal{B}_{\langle V_1, I_1\rangle}$ is a $\langle V_1, I_1\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ s.t. $(s_1, s_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$. We will show that $\mathcal{B}_{\langle V_1, I_1\rangle}$ is also a $\langle V_2, I_2\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ s.t. $(s_1, s_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$. For all $(w_1, w_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ and $j \notin I_2$, there is:

- $L_1(w_1) - V_2 = L_2(w_2) - V_2$ since $L_1(w_1) - V_1 = L_2(w_2) - V_1$ and $V_1 \subseteq V_2$;
- $\forall r_1 \in S_1$, if $(w_1, r_1) \in R_1$ then $\exists r_2 \in S_2$ s.t. $(w_2, r_2) \in R_2$ and $(r_1, r_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ since $\mathcal{B}_{\langle V_1, I_1\rangle}$ is a $\langle V_1, I_1\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$; and
- $\forall r_2 \in S_2$, if $(w_2, r_2) \in R_2$ then $\exists r_1 \in S_1$ s.t. $(w_1, r_1) \in R_1$ and $(r_1, r_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ since $\mathcal{B}_{\langle V_1, I_1\rangle}$ is a $\langle V_1, I_1\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$;
- $\forall (w_1, r_1) \in [j]_1$, $\exists (w_2, r_2) \in [j]_2$ s.t. $(w_1, w_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ and $(r_1, r_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ since $\mathcal{B}_{\langle V_1, I_1\rangle}$ is a $\langle V_1, I_1\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$; and
- $\forall (w_2, r_2) \in [j]_2$, $\exists (w_1, r_1) \in [j]_1$ s.t. $(w_1, w_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ and $(r_1, r_2) \in \mathcal{B}_{\langle V_1, I_1\rangle}$ since $\mathcal{B}_{\langle V_1, I_1\rangle}$ is a $\langle V_1, I_1\rangle$-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$.

■

Proposition 1 Let $i \in \{1, 2\}$, $V_1, V_2 \subseteq \mathcal{A}$, s_i's be two states, π_i's be two paths and $\mathcal{K}_j = (\mathcal{M}_j, s_j)$ $(j = 1, 2, 3)$ be$\mathcal{I}$-structures s.t. $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$ and $\mathcal{K}_2 \leftrightarrow_{V_2} \mathcal{K}_3$. Then:

(i) $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$;

(ii) If $V_1 \subseteq V_2$ then $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$;

(iii) $s_1' \leftrightarrow_{V_i} s_2'$ $(i = 1, 2)$ implies $s_1' \leftrightarrow_{V_1 \cup V_2} s_2'$;

(iv) $\pi_1' \leftrightarrow_{V_i} \pi_2'$ $(i = 1, 2)$ implies $\pi_1' \leftrightarrow_{V_1 \cup V_2} \pi_2'$;

(v) for each path π_{s_1} of $\mathcal{M}_1$ there is a path π_{s_2} of $\mathcal{M}_2$ s.t. $\pi_{s_1} \leftrightarrow_{V_1} \pi_{s_2}$, and vice versa.

Proof: (i) Let $\mathcal{M}_j = (S_j, R_j, L_j, [\_]_j, s_0^j)$ $(j = 1, 2, 3)$, $s_1 \leftrightarrow_{V_1} s_2$ via a binary relation $\mathcal{B}$, and $s_2 \leftrightarrow_{V_2} s_3$ via a binary relation $\mathcal{B}''$. Let $\mathcal{B}' = \{(w_1, w_3) \mid (w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''\}$. To prove $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$, we will prove that $\mathcal{B}'$ is a $V_1 \cup V_2$-bisimulation containing (s_1, s_3). It's evident that $(s_1, s_3) \in \mathcal{B}'$ due to $(s_1, s_2) \in \mathcal{B}$ and $(s_2, s_3) \in \mathcal{B}''$. For all $(w_1, w_3) \in \mathcal{B}'$:

(a) there exists $w_2 \in S_2$ s.t. $(w_1, w_2) \in \mathcal{B}$ and $(w_2, w_3) \in \mathcal{B}''$, and for all $q \notin V_1$, $q \in L_1(w_1)$ iff $q \in L_2(w_2)$ by $w_1 \leftrightarrow_{V_1} w_2$ and for all $q' \notin V_2$, $q' \in L_2(w_2)$ iff $q' \in L_3(w_3)$ by $w_2 \leftrightarrow_{V_2} w_3$. Then we have for all $r \notin V_1 \cup V_2$, $r \in L_1(w_1)$ iff $r \in L_3(w_3)$.

(b) $\forall u_1 \in S_1$, if $(w_1, u_1) \in R_1$, then $\exists u_2 \in S_2$ s.t. $(w_2, u_2) \in R_2$ 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 $\exists u_3 \in S_3$ s.t. $(w_3, u_3) \in R_3$ and $(u_2, u_3) \in \mathcal{B}''$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

(c) $\forall u_3 \in S_3$, if $(w_3, u_3) \in R_3$, then $\exists u_2 \in S_2$ s.t. $(w_2, u_2) \in R_2$ and $(u_2, u_3) \in \mathcal{B}_2$; and then $\exists u_1 \in S_1$ s.t. $(w_1, u_1) \in R_1$ and $(u_1, u_2) \in \mathcal{B}$, hence $(u_1, u_3) \in \mathcal{B}'$ by the definition of $\mathcal{B}'$.

(ii) Supposing $\mathcal{B}_{V_1}$ is a V_1-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ s.t. $(s_1, s_2) \in \mathcal{B}_{V_1}$. We will show that $\mathcal{B}_{V_1}$ is also a V_2-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$ s.t. $(s_1, s_2) \in \mathcal{B}_{V_1}$. For all $(w_1, w_2) \in \mathcal{B}_{V_1}$, there is:

- $L_1(w_1) - V_2 = L_2(w_2) - V_2$ since $L_1(w_1) - V_1 = L_2(w_2) - V_1$ and $V_1 \subseteq V_2$;

- $\forall r_1 \in S_1$, if $(w_1, r_1) \in R_1$ then $\exists r_2 \in S_2$ s.t. $(w_2, r_2) \in R_2$ and $(r_1, r_2) \in \mathcal{B}_{V_1}$ since $\mathcal{B}_{V_1}$ is a V_1-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$; and
- $\forall r_2 \in S_2$, if $(w_2, r_2) \in R_2$ then $\exists r_1 \in S_1$ s.t. $(w_1, r_1) \in R_1$ and $(r_1, r_2) \in \mathcal{B}_{V_1}$ since $\mathcal{B}_{V_1}$ is a V_1-bisimulation between $\mathcal{M}_1$ and $\mathcal{M}_2$.

(iii) is a special case of (ii) due to $V_i \subseteq (V_1 \cup V_2)$ with $i = 1, 2$. And then (iv) is evident from (iii).

(v) This is obvious from the definition of V-bisimulation due to $s_1 \leftrightarrow_{V_1} s_2$ (i.e., $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$).

■

Theorem 4 Let $V \subseteq \mathcal{A}$, $\mathcal{K}_i$ $(i = 1, 2)$ be two$\mathcal{I}$-structures s.t. $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ and ϕ a formula with $\text{IR}(\phi, V)$. Then $\mathcal{K}_1 \models \phi$ if and only if $\mathcal{K}_2 \models \phi$.

Proof: This theorem can be proved by inducting on the formula ϕ and supposing $Var(\phi) \cap V = \emptyset$. Let $\mathcal{K}_1 = (\mathcal{M}, s)$ and $\mathcal{K}_2 = (\mathcal{M}', s')$.

Base case. $\phi = p$ where $p \in \mathcal{A} - V$:

$(\mathcal{M}, s) \models \phi$ iff $p \in L(s)$ (by the definition of satisfiability)
$\Leftrightarrow p \in L'(s')$ $(s \leftrightarrow_V s')$
$\Leftrightarrow (\mathcal{M}', s') \models \phi$

Step case. (1) $\phi = \neg\psi$:

$(\mathcal{M}, s) \models \phi$ iff $(\mathcal{M}, s) \not\models \psi$
$\Leftrightarrow (\mathcal{M}', s') \not\models \psi$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \phi$

(2) $\phi = \psi_1 \vee \psi_2$:

$(\mathcal{M}, s) \models \phi$
$\Leftrightarrow (\mathcal{M}, s) \models \psi_1$ or $(\mathcal{M}, s) \models \psi_2$
$\Leftrightarrow (\mathcal{M}', s') \models \psi_1$ or $(\mathcal{M}', s') \models \psi_2$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \phi$

(3) $\phi = \text{EX}\psi$:

$(\mathcal{M}, s) \models \phi$
$\Leftrightarrow$ There is a path $\pi = (s, s_1, ...)$ s.t. $(\mathcal{M}, s_1) \models \psi$
$\Leftrightarrow$ There is a path $\pi' = (s', s'_1, ...)$ s.t. $\pi \leftrightarrow_V \pi'$ ($s \leftrightarrow_V s'$, Proposition 1(v))
$\Leftrightarrow s_1 \leftrightarrow_V s'_1$ $(\pi \leftrightarrow_V \pi')$
$\Leftrightarrow (\mathcal{M}', s'_1) \models \psi$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \phi$

(4) $\phi = \text{EG}\psi$:

$(\mathcal{M}, s) \models \phi$
$\Leftrightarrow$ There is a path $\pi = (s = s_0, s_1, ...)$ s.t. for each $i \geq 0$ there is $(\mathcal{M}, s_i) \models \psi$
$\Leftrightarrow$ There is a path $\pi' = (s' = s'_0, s'_1, ...)$ s.t. $\pi \leftrightarrow_V \pi'$ ($s \leftrightarrow_V s'$, Proposition 1(v))
$\Leftrightarrow s_i \leftrightarrow_V s'_i$ for each $i \geq 0$ $(\pi \leftrightarrow_V \pi')$
$\Leftrightarrow (\mathcal{M}', s'_i) \models \psi$ for each $i \geq 0$ (induction hypothesis)
$\Leftrightarrow (\mathcal{M}', s') \models \phi$

(5) $\phi = \text{E}(\psi_1 \text{U} \psi_2)$:

$(\mathcal{M}, s) \models \phi$
$\Leftrightarrow$ 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 1(v))
$\Leftrightarrow$ $(\mathcal{M}', s_i') \models \psi_2$, and for all $0 \leq j < i$ $(\mathcal{M}', s_j') \models \psi_1$ (induction hypothesis)
$\Leftrightarrow$ $(\mathcal{M}', s') \models \phi$ ■

Theorem 6 Let φ be a PL formula and $V \subseteq \mathcal{A}$, then

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

Proof: Let $\mathcal{M} = (S, R, L, [\_], s)$ and $\mathcal{M}' = (S', R', L', [\_]', s')$.

On the one hand, for each $(\mathcal{M}, s) \in Mod(\mathrm{F}_{\mathrm{CTL}}(\varphi, V))$, there exists a $(\mathcal{M}', s') \in Mod(\varphi)$ s.t. $s \leftrightarrow_V s'$. Thus, $L(s) - V = L'(s') - V$. Hence, $(\mathcal{M}, s)$ is a model of $\mathrm{F}(\varphi, V)$.

On the other hand, for each $(\mathcal{M}, s) \in Mod(\mathrm{F}(\varphi, V))$, there exists a $(\mathcal{M}', s') \in Mod(\varphi)$ s.t. $L(s) - V = L'(s') - V$. We can construct an initial$\mathcal{I}$-structure $(\mathcal{M}_1, s_1)$ s.t. $\mathcal{M}_1 = (S_1, R_1, L_1, [\_]_1, s_1)$ with $S_1 = (S - \{s\}) \cup \{s_1\}$, R_1 is the same as R, except replace s with s_1, and L_1 is the same as L, except $L_1(s_1) = L'(s')$, where L' is the label function of M'. It is clear that $(\mathcal{M}_1, s_1)$ is a model of φ and $s_1 \leftrightarrow_V s$. Hence, $(\mathcal{M}, s)$ is a model of $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$. ■

Lemma 5 Let φ and α be two CTL formulas and $q \notin (Var(\varphi) \cup Var(\alpha))$. Then $\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge (q \leftrightarrow \alpha), q) \equiv \varphi$.
Proof: Let $\varphi' = \varphi \wedge (q \leftrightarrow \alpha)$. For any model $(\mathcal{M}, s)$ of $\mathrm{F}_{\mathrm{CTL}}(\varphi', q)$ there is an initial$\mathcal{I}$-structure $(\mathcal{M}', s')$ s.t. $(\mathcal{M}, s) \leftrightarrow_{\{q\}} (\mathcal{M}', s')$ and $(\mathcal{M}', s') \models \varphi'$. It's evident that $(\mathcal{M}', s') \models \varphi$, and then $(\mathcal{M}, s) \models \varphi$ since $\mathrm{IR}(\varphi, \{q\})$ and $(\mathcal{M}, s) \leftrightarrow_{\{q\}} (\mathcal{M}', s')$ by Theorem 4.

Let $(\mathcal{M}, s) \in Mod(\varphi)$ with $\mathcal{M} = (S, R, L, [\_], s)$. We construct $(\mathcal{M}', s)$ with $\mathcal{M}' = (S, R, L', [\_], s)$ as follows:

$$L' : S \to \mathcal{A} \textit{ and } \forall s^* \in S, L'(s^*) = L(s^*) - \{q\} \text{ if } (\mathcal{M}, s^*) \not\models \alpha, else\ L'(s^*) = L(s^*) \cup \{q\},$$
$$L'(s) = L(s) \cup \{q\} \text{ if } (\mathcal{M}, s) \models \alpha, \textit{ and } L'(s) = L(s) - \{q\} \textit{ otherwise.}$$

It is clear that $(\mathcal{M}', s) \models \varphi$, $(\mathcal{M}', s) \models q \leftrightarrow \alpha$ and $(\mathcal{M}', s) \leftrightarrow_{\{q\}} (\mathcal{M}, s)$. Therefore $(\mathcal{M}', s) \models \varphi \wedge (q \leftrightarrow \alpha)$, and then $(\mathcal{M}, s) \models \mathrm{F}_{\mathrm{CTL}}(\varphi \wedge (q \leftrightarrow \alpha), q)$ by $(\mathcal{M}', s) \leftrightarrow_{\{q\}} (\mathcal{M}, s)$. ■

Proposition 4 **(Modularity)** Given a formula $\varphi \in \mathrm{CTL}$, V a set of atoms, and p an atom s.t. $p \notin V$, then

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

Proof: Let $(\mathcal{M}_1, s_1)$ with $\mathcal{M}_1 = (S_1, R_1, L_1, s_1)$ be a model of $\mathrm{F}_{\mathrm{CTL}}(\varphi, \{p\} \cup V)$. By the definition, there exists a model $(\mathcal{M}, s)$ with $\mathcal{M} = (S, R, L, [\_], s)$ of φ, such that $(\mathcal{M}_1, s_1)$ $\leftrightarrow_{\{p\} \cup V}$ $(\mathcal{M}, s)$. We construct an initial structure $(\mathcal{M}_2, s_2)$ with $\mathcal{M}_2 = (S_2, R_2, L_2, s_2)$ as follows:

(1) for s_2: let s_2 be the state such that:

- $p \in L_2(s_2)$ iff $p \in L_1(s_1)$,
- for all $q \in V$, $q \in L_2(s_2)$ iff $q \in L(s)$,

- for all other atoms q', $q' \in L_2(s_2)$ iff $q' \in L_1(s_1)$ iff $q' \in L(s)$.

(2) for another:

(i) for all pairs $w \in S$ and $w_1 \in S_1$ s.t. $w \leftrightarrow_{\{p\}\cup V} w_1$, let $w_2 \in S_2$ and

- $p \in L_2(w_2)$ iff $p \in L_1(w_1)$,
- for all $q \in V$, $q \in L_2(w_2)$ iff $q \in L(w)$,
- for all other atoms q', $q' \in L_2(w_2)$ iff $q' \in L_1(w_1)$ iff $q' \in L(w)$.

(ii) if $(w_1', w_1) \in R_1$, w_2 is constructed based on w_1 and $w_2' \in S_2$ is constructed based on w_1', then $(w_2', w_2) \in R_2$.

(3) delete duplicated states in S_2 and pairs in R_2.

Then we have $(\mathcal{M}, s) \leftrightarrow_{\{p\}} (\mathcal{M}_2, s_2)$ and $(\mathcal{M}_2, s_2) \leftrightarrow_V (\mathcal{M}_1, s_1)$. Thus, $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\text{CTL}}(\varphi, p)$. Therefore $(\mathcal{M}_1, s_1) \models \mathrm{F}_{\text{CTL}}(\mathrm{F}_{\text{CTL}}(\varphi, p), V)$.

On the other hand, suppose that $(\mathcal{M}_1, s_1)$ is a model of $\mathrm{F}_{\text{CTL}}(\mathrm{F}_{\text{CTL}}(\varphi, p), V)$, then there exists an initial structure $(\mathcal{M}_2, s_2)$ s.t. $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\text{CTL}}(\varphi, p)$ and $(\mathcal{M}_2, s_2) \leftrightarrow_V (\mathcal{M}_1, s_1)$, and there exists $(\mathcal{M}, s)$ s.t. $(\mathcal{M}, s) \models \varphi$ and $(\mathcal{M}, s) \leftrightarrow_{\{p\}} (\mathcal{M}_2, s_2)$. Therefore, $(\mathcal{M}, s) \leftrightarrow_{\{p\}\cup V} (\mathcal{M}_1, s_1)$ by Proposition 1(i), and consequently, $(\mathcal{M}_1, s_1) \models \mathrm{F}_{\text{CTL}}(\varphi, \{p\} \cup V)$. ■

Proposition 5 Let φ, φ_i, ψ_i $(i = 1, 2)$ be formulas in CTL and $V \subseteq \mathcal{A}$. We have

(i) $\mathrm{F}_{\text{CTL}}(\varphi, V)$ is satisfiable iff φ is;

(ii) If $\varphi_1 \equiv \varphi_2$, then $\mathrm{F}_{\text{CTL}}(\varphi_1, V) \equiv \mathrm{F}_{\text{CTL}}(\varphi_2, V)$;

(iii) If $\varphi_1 \models \varphi_2$, then $\mathrm{F}_{\text{CTL}}(\varphi_1, V) \models \mathrm{F}_{\text{CTL}}(\varphi_2, V)$;

(iv) $\mathrm{F}_{\text{CTL}}(\psi_1 \vee \psi_2, V) \equiv \mathrm{F}_{\text{CTL}}(\psi_1, V) \vee \mathrm{F}_{\text{CTL}}(\psi_2, V)$;

(v) $\mathrm{F}_{\text{CTL}}(\psi_1 \wedge \psi_2, V) \models \mathrm{F}_{\text{CTL}}(\psi_1, V) \wedge \mathrm{F}_{\text{CTL}}(\psi_2, V)$;

Proof: (i) $(\Rightarrow)$ Supposing $(\mathcal{M}, s)$ is a model of $\mathrm{F}_{\text{CTL}}(\varphi, V)$, then there is a model $(\mathcal{M}', s')$ of φ s.t. $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$ by the definition of F_{CTL}.

$(\Leftarrow)$ Supposing $(\mathcal{M}, s)$ is a model of φ, then there is an initial structure $(\mathcal{M}', s')$ s.t. $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$, and then $(\mathcal{M}', s') \models \mathrm{F}_{\text{CTL}}(\varphi, V)$ by the definition of F_{CTL}.

The (ii) and (iii) can be proved similarly.

(iv) $(\Rightarrow)$ For all$(\mathcal{M}, s) \in \mathit{Mod}(\mathrm{F}_{\text{CTL}}(\psi_1 \vee \psi_2, V))$, there exists $(\mathcal{M}', s') \in \mathit{Mod}(\psi_1 \vee \psi_2)$ s.t. $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$ and $(\mathcal{M}', s') \models \psi_1$ or $(\mathcal{M}', s') \models \psi_2$
$\Rightarrow$ there exists $(\mathcal{M}_1, s_1) \in \mathit{Mod}(\mathrm{F}_{\text{CTL}}(\psi_1, V))$ s.t. $(\mathcal{M}', s') \leftrightarrow_V (\mathcal{M}_1, s_1)$ or there exists $(\mathcal{M}_2, s_2) \in \mathit{Mod}(\mathrm{F}_{\text{CTL}}(\psi_2, V))$ s.t. $(\mathcal{M}', s') \leftrightarrow_V (\mathcal{M}_2, s_2)$
$\Rightarrow (\mathcal{M}, s) \models \mathrm{F}_{\text{CTL}}(\psi_1, V) \vee \mathrm{F}_{\text{CTL}}(\psi_2, V)$ by Theorem 4.

$(\Leftarrow)$ for all $(\mathcal{M}, s) \in \mathit{Mod}(\mathrm{F}_{\text{CTL}}(\psi_1, V) \vee \mathrm{F}_{\text{CTL}}(\psi_2, V))$
$\Rightarrow (\mathcal{M}, s) \models \mathrm{F}_{\text{CTL}}(\psi_1, V)$ or $(\mathcal{M}, s) \models \mathrm{F}_{\text{CTL}}(\psi_2, V)$
$\Rightarrow$ there is an initial structure $(\mathcal{M}_1, s_1)$ s.t. $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}_1, s_1)$ and $(\mathcal{M}_1, s_1) \models \psi_1$ or $(\mathcal{M}_1, s_1) \models \psi_2$

$\Rightarrow (\mathcal{M}_1, s_1) \models \psi_1 \vee \psi_2$
$\Rightarrow (\mathcal{M}, s) \models \mathrm{F}_{\mathrm{CTL}}(\psi_1 \vee \psi_2, V)$ due to $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}_1, s_1)$.
The (v) can be proved as (iv). ■

Proposition 6 **(Homogeneity)** Let $V \subseteq \mathcal{A}$ and $\phi \in \mathrm{CTL}$, then:

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

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

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

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

Proof: Let $\mathcal{M} = (S, R, L, s_0)$ with the initial state s_0 and $\mathcal{M}' = (S', R', L', s_0')$ with the initial state s_0', then we call $(\mathcal{M}', s_0')$ be a sub-structure of $(\mathcal{M}, s_0)$ if:

- $S' \subseteq S$ and $S' = \{s' \mid s'$ is reachable from $s_0'\}$,
- $R' = \{(s_1, s_2) \mid s_1, s_2 \in S'$ and $(s_1, s_2) \in R\}$,
- $L' : S' \to 2^{\mathcal{A}}$ and for all $s_1 \in S'$ there is $L'(s_1) = L(s_1)$, and
- s_0' is s_0 or a state reachable from s_0.

(i) To prove $\mathrm{F}_{\mathrm{CTL}}(\mathrm{AX}\phi, V) \equiv \mathrm{AX}(\mathrm{F}_{\mathrm{CTL}}(\phi, V))$, the only thing we need to prove is

$$\mathit{Mod}(\mathrm{F}_{\mathrm{CTL}}(\mathrm{AX}\phi, V)) = \mathit{Mod}(\mathrm{AX}\mathrm{F}_{\mathrm{CTL}}(\phi, V)).$$

($\Rightarrow$) For all $(\mathcal{M}', s') \in \mathit{Mod}(\mathrm{F}_{\mathrm{CTL}}(\mathrm{AX}\phi, V))$ there exists an initial structure $(\mathcal{M}, s)$ s.t. $(\mathcal{M}, s) \models \mathrm{AX}\phi$ and $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$
$\Rightarrow$ for any sub-structure $(\mathcal{M}_1, s_1)$ of $(\mathcal{M}, s)$, there is $(\mathcal{M}_1, s_1) \models \phi$, where s_1 is a directed successor of s
$\Rightarrow$ there is an initial structure $(\mathcal{M}_2, s_2)$ s.t. $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\mathrm{CTL}}(\phi, V)$ and $(\mathcal{M}_2, s_2) \leftrightarrow_V (\mathcal{M}_1, s_1)$
$\Rightarrow$ it is easy to construct an initial structure $(\mathcal{M}_3, s_3)$ by $(\mathcal{M}_2, s_2)$ s.t. $(\mathcal{M}_2, s_2)$ is a sub-structure of $(\mathcal{M}_3, s_3)$ with s_2 is a direct successor of s_3 and $(\mathcal{M}_3, s_3) \leftrightarrow_V (\mathcal{M}, s)$
$\Rightarrow (\mathcal{M}_3, s_3) \models \mathrm{AX}(\mathrm{F}_{\mathrm{CTL}}(\phi, V))$ and $(\mathcal{M}_3, s_3) \leftrightarrow_V (\mathcal{M}', s')$
$\Rightarrow (\mathcal{M}', s') \models \mathrm{AX}(\mathrm{F}_{\mathrm{CTL}}(\phi, V))$ by Theorem 4.

($\Leftarrow$) For all $(\mathcal{M}_3, s_3) \in \mathit{Mod}(\mathrm{AX}(\mathrm{F}_{\mathrm{CTL}}(\phi, V)))$, then for any sub-structure $(\mathcal{M}_2, s_2)$ with s_2 is a directed successor of s_3, there is $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\mathrm{CTL}}(\phi, V)$
$\Rightarrow$ for any $(\mathcal{M}_2, s_2)$, there is an initial structure $(\mathcal{M}_1, s_1)$ s.t. $(\mathcal{M}_1, s_1) \models \phi$ and $(\mathcal{M}_1, s_1) \leftrightarrow_V (\mathcal{M}_2, s_2)$
$\Rightarrow$ it is easy to construct an initial structure $(\mathcal{M}, s)$ by $(\mathcal{M}_1, s_1)$ s.t. $(\mathcal{M}_1, s_1)$ is a sub-structure of $(\mathcal{M}, s)$ with s_1 is a direct successor of s and $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}_3, s_3)$
$\Rightarrow (\mathcal{M}, s) \models \mathrm{AX}\phi$ and then $(\mathcal{M}_3, s_3) \models \mathrm{F}_{\mathrm{CTL}}(\mathrm{AX}\phi, V)$.

(ii) In order to prove $\mathrm{F}_{\mathrm{CTL}}(\mathrm{EX}\phi, V) \equiv \mathrm{EX}\mathrm{F}_{\mathrm{CTL}}(\phi, V)$, the only thing we need to prove is

$$\mathit{Mod}(\mathrm{F}_{\mathrm{CTL}}(\mathrm{EX}\phi, V)) = \mathit{Mod}(\mathrm{EX}\mathrm{F}_{\mathrm{CTL}}(\phi, V)).$$

($\Rightarrow$) For all $(\mathcal{M}', s') \in \mathit{Mod}(\mathrm{F}_{\text{CTL}}(\text{EX}\phi, V))$ there exists an initial structure $(\mathcal{M}, s)$ s.t. $(\mathcal{M}, s) \models \text{EX}\phi$ and $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}', s')$
$\Rightarrow$ there is a sub-structure $(\mathcal{M}_1, s_1)$ of $(\mathcal{M}, s)$ s.t. $(\mathcal{M}_1, s_1) \models \phi$, where s_1 is a directed successor of s
$\Rightarrow$ there is an initial structure $(\mathcal{M}_2, s_2)$ s.t. $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\text{CTL}}(\phi, V)$ and $(\mathcal{M}_2, s_2) \leftrightarrow_V (\mathcal{M}_1, s_1)$
$\Rightarrow$ it is easy to construct an initial structure $(\mathcal{M}_3, s_3)$ by $(\mathcal{M}_2, s_2)$ s.t. $(\mathcal{M}_2, s_2)$ is a sub-structure of $(\mathcal{M}_3, s_3)$ that s_2 is a direct successor of s_3 and $(\mathcal{M}_3, s_3) \leftrightarrow_V (\mathcal{M}, s)$
$\Rightarrow (\mathcal{M}_3, s_3) \models \text{EX}(\mathrm{F}_{\text{CTL}}(\phi, V))$
$\Rightarrow (\mathcal{M}', s') \models \text{EX}(\mathrm{F}_{\text{CTL}}(\phi, V))$.

($\Leftarrow$) For all $(\mathcal{M}_3, s_3) \in \mathit{Mod}(\text{EX}(\mathrm{F}_{\text{CTL}}(\phi, V)))$, there exists a sub-structure $(\mathcal{M}_2, s_2)$ of $(\mathcal{M}_3, s_3)$ s.t. $(\mathcal{M}_2, s_2) \models \mathrm{F}_{\text{CTL}}(\phi, V)$
$\Rightarrow$ there is an initial structure $(\mathcal{M}_1, s_1)$ s.t. $(\mathcal{M}_1, s_1) \models \phi$ and $(\mathcal{M}_1, s_1) \leftrightarrow_V (\mathcal{M}_2, s_2)$
$\Rightarrow$ it is easy to construct an initial structure $(\mathcal{M}, s)$ by $(\mathcal{M}_1, s_1)$ s.t. $(\mathcal{M}_1, s_1)$ is a sub-structure of $(\mathcal{M}, s)$ that s_1 is a direct successor of s and $(\mathcal{M}, s) \leftrightarrow_V (\mathcal{M}_3, s_3)$
$\Rightarrow (\mathcal{M}, s) \models \text{EX}\phi$ and then $(\mathcal{M}_3, s_3) \models \mathrm{F}_{\text{CTL}}(\text{EX}\phi, V)$.

(iii) and (iv) can be proved as (i) and (ii), respectively. ■

Appendix C. Proofs for Section 5

Proposition ?? Let φ be a CTL formula and (T_φ, V', I) =Transform(φ). $\varphi \equiv_{\langle V', I\rangle} T_\varphi$.
Proof: For convenience, we define the transformation of any CTL formula φ into the set T_φ by a sequence $T_0, T_1, \ldots, T_n = T_\varphi$ of sets of formulas with $T_0 = \{\text{AG}(\mathbf{start} \to p), \text{AG}(p \to \mathbf{simp}(\mathbf{nnf}(\varphi)))\}$ such that for every i $(0 \leq i < n)$, $T_{i+1} = (T_i \setminus \{\psi\}) \cup R_i$ and all the formulas in T_φ are $\text{SNF}^g_{\text{CTL}}$ clauses, where p is a fresh atom, ψ is a formula in T_i which is not a $\text{SNF}^g_{\text{CTL}}$ clause, and R_i is the result set of applying a matching transformation rule to ψ. Note that throughout the transformation, formulas are kept in negation normal form (nnf).

Then the proposition can be proved from T_i to T_{i+1} $(0 \leq i < n)$ by using one transformation rule on T_i. We will prove this proposition from the following several aspects:

(1) $\varphi \equiv_{\langle \{p\}, \varnothing\rangle} T_0$.

($\Rightarrow$) For each $(\mathcal{M}_1, s_1) \in \mathit{Mod}(\varphi)$, we can construct an$\mathcal{I}$-Kripke structure $\mathcal{M}_2$ is identical to $\mathcal{M}_1$ except for $L_2(s_2) = L_1(s_1) \cup \{p\}$. It is evident that $(\mathcal{M}_2, s_2) \models T_0$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \varnothing\rangle} (\mathcal{M}_2, s_2)$.

($\Leftarrow$) For each $(\mathcal{M}_1, s_1) \in \mathit{Mod}(T_0)$, it is evident 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 formulas ψ as the premises of rule t) and obtaining the set R_i of its results. Let X be a set of formulas, we will show $T_i \equiv_{\langle V', I\rangle} T_{i+1}$ by using the transformation rule t. Where $T_i = X \cup \{\psi\}$, $T_{i+1} = X \cup R_i$, V' is the set of atoms introduced by t and I is the set of indices introduced by t. (We will prove this result for $t \in \{$**Trans(1), Trans(4), Trans(6)**$\}$ in Table 1, while other cases can be proved similarly.)

(2) t=**Trans(1)**.

($\Rightarrow$) For each $(\mathcal{M}_1, s_1) \in Mod(T_i)$, i.e., $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \text{EX}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$, 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 s.t. $(s_1^j, s_1^{j+1}) \in R_1$ and $(\mathcal{M}, s_1^{j+1}) \models \varphi$.

We construct an$\mathcal{I}$-Kripke structure $\mathcal{M}_2$, which is identical to $\mathcal{M}_1$ except for $[ind]_2 = \bigcup_{s\in S} R_s \cup R_y$. Where ind is the index introduced by using Trans(1) on clause $\text{AG}(q \rightarrow \text{EX}\varphi)$, $R_{s_1^j} = \{(s_1^j, s_1^{j+1}), (s_1^{j+1}, s_1^{j+2}), \dots\}$ (with if $(\mathcal{M}_1, s_1^j) \models q$ then $(\mathcal{M}_1, s_1^{j+1}) \models \varphi$) and $R_y = \{(s_x, s_y) \mid$ for all $s_x \in S$ if for all $(s_1', s_2') \in \bigcup_{s\in S} R_s, s_1' \neq s_x$ then find exactly one state $s_y \in S$ s.t. $(s_x, s_y) \in R\}$, which means for every $s \in S$ there exists exactly a state $s' \in S$ s.t. $(s, s') \in [ind]$ and $(s, s') \in R$. It is evident 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)
$\Rightarrow (\mathcal{M}_2, s_1) \models \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow (\mathcal{M}_2, s_1) \models X \wedge \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi)$

($\Leftarrow$) For all $(\mathcal{M}_1, s_1) \in Mod(T_{i+1})$, i.e., $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and $(\mathcal{M}_1, s_1) \models \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow$ 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' s.t. $(s_1^j, s') \in [ind]_1$ 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)
$\Rightarrow (\mathcal{M}_1, s_1) \models \text{AG}(q \rightarrow \text{EX}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \text{EX}\varphi)$
It is obivous that $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \emptyset, \{ind\}\rangle} (\mathcal{M}_1, s_1)$.

(3) t=**Trans(4)**.

($\Rightarrow$) For each $(\mathcal{M}_1, s_1) \in Mod(T_i)$, i.e., $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \varphi_1 \vee \varphi_2)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and $\forall s_1' \in S, (\mathcal{M}_1, s_1') \models q \rightarrow \varphi_1 \vee \varphi_2$
$\Rightarrow (\mathcal{M}_1, s_1') \models \neg q$ or $(\mathcal{M}_1, s_1') \models \varphi_1 \vee \varphi_2$

Then we construct an$\mathcal{I}$-Kripke structure $\mathcal{M}_2$ as follows: $\mathcal{M}_2$ is the same with $\mathcal{M}_1$ except for each state s_1' if $(\mathcal{M}_1, s_1') \models \neg q$ then $L_2(s_1') = L_1(s_1')$, else 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 evident that $(\mathcal{M}_2, s_1') \models (q \rightarrow \varphi_1 \vee p) \wedge (p \rightarrow \varphi_2)$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset\rangle} (\mathcal{M}_2, s_2)$, then $(\mathcal{M}_2, s_1) \models T_{i+1}$.

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

(4) t=**Trans(6)**. We prove it is correct for the $\text{E}_{\langle ind\rangle}\text{X}$, while it can be proved similarly for the AX case.

($\Rightarrow$) For each $(\mathcal{M}_1, s_1) \in Mod(T_i)$, *i.e.* $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi)$
$\Rightarrow (\mathcal{M}_1, s_1) \models X$ and for all $s_1' \in S, (\mathcal{M}_1, s_1') \models q \rightarrow \text{E}_{\langle ind\rangle}\text{X}\varphi$
$\Rightarrow (\mathcal{M}_1, s_1') \models \neg q$ or there exists a state s' s.t. $(s_1', s') \in [ind]$ and $(\mathcal{M}_1, s') \models \varphi$

We construct an$\mathcal{I}$-Kripke structure $\mathcal{M}_2$ as follows: $\mathcal{M}_2$ is the same with $\mathcal{M}_1$ except for each state s_1' if $(\mathcal{M}_1, s_1') \models \neg q$ then $L_2(s_1') = L_1(s_1')$, else if $(\mathcal{M}_1, s_1') \models q$ then $L_2(s') = L_1(s') \cup \{p\}$. It is evident that $(\mathcal{M}_2, s_1) \models \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}p) \wedge \text{AG}(p \rightarrow \varphi)$, $(\mathcal{M}_2, s_2) \models T_{i+1}$ and $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset\rangle} (\mathcal{M}_2, s_2)$ $(s_2 = s_1)$.

($\Leftarrow$) For each $(\mathcal{M}_1, s_1) \in Mod(T_{i+1})$, i.e., $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \rightarrow \text{E}_{\langle ind\rangle}\text{X}p) \wedge \text{AG}(p \rightarrow \varphi)$. It is apparent that $(\mathcal{M}_1, s_1) \models T_i$.

■

Proposition ?? Let φ be a CTL formula, then $T_\varphi \equiv_{\langle V\cup V', \emptyset\rangle}$ *Resolution*$(T_\varphi, V \cup V')$.
Proof: This result can be proved from T_i to T_{i+1} $(0 \le i < n)$ by using one resolution rule on T_i.

By $\Pi \to_r R_i$, we mean using resolution rules r on set Π (the formulas in Π as the premises of rule r) and obtaining the set R_i of resolution results. We will show $T_i \equiv_{\langle V\cup V', \emptyset\rangle} T_{i+1}$ by using the resolution rule r. Where $T_i = X \cup \Pi$, $T_{i+1} = X \cup R_i$, X is a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses, and p is the proposition corresponding with literal l used to resolution in r.

(1) If $\Pi \to_r R_i$ with $r \in \{$**(SRES1)**, ..., **(SRES8)**, (**RW1**), (**RW2**)$\}$ in Table 2, then $T_i \equiv_{\langle\{p\},\emptyset\rangle} T_{i+1}$.

On the one hand, it is evident that $\Pi \models R_i$ and then $T_i \models T_{i+1}$. On the other hand, $T_i \subseteq T_{i+1}$ and then $T_{i+1} \models T_i$.

(2) If $\Pi \to_r R_i$ with $r =$**(ERES1)**, then $T_i \equiv_{\langle\{l, w^{\mathrm{A}}_{\neg l}\},\emptyset\rangle} T_{i+1}$.

It has been proved that $\Pi \models R_i$ in (Bolotov, 2000), then there is $T_{i+1} = T_i \cup \Lambda^{\mathrm{A}}_{\neg l}$. And then for all $(\mathcal{M}_1, s_1) \in Mod(T_i = X \cup \Pi)$, there is a $(\mathcal{M}_2, s_2) \in Mod(T_{i+1} = T_i \cup \Lambda^{\mathrm{A}}_{\neg l})$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle\{p, w^{\mathrm{A}}_{\neg l}\},\emptyset\rangle} (\mathcal{M}_2, s_2)$, and for all $(\mathcal{M}_2, s_2) \in Mod(T_{i+1} = T_i \cup \Lambda^{\mathrm{A}}_{\neg l})$, there is a $(\mathcal{M}_1, s_1) \in Mod(T_i = X \cup \Pi)$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle\{p, w^{\mathrm{A}}_{\neg l}\},\emptyset\rangle} (\mathcal{M}_2, s_2)$ by Proposition **??**. Where $\Lambda^{\mathrm{A}}_{\neg l}$ is a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses obtained by using the transformation rules in Table 1 on the formula in R_i and $w^{\mathrm{A}}_{\neg l}$ is a fresh atom uniquely associated with clause $Q \to \mathrm{AF}\neg l$ (please see (Zhang, Hustadt, & Dixon, 2009) for more detail).

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

■

Proposition ?? *Resolution*$(T_\varphi, V \cup V') \equiv_{V\cup V'}$ *Removing_atoms*$($*Resolution*$(T_\varphi, V\cup V'), V)$.
Proof: For convenience, we let $Res =$ *Resolution*$(T_\varphi, V \cup V')$, $V = \{p\}$, C_i is a classical clause, and l is p or $\neg p$. It is evident that $Res \models$ *Removing_atoms*(Res, V), we will prove that for each $\mathcal{K} = (\mathcal{M}, s) \in Mod($*Removing_atoms*$(Res, V))$ with $\mathcal{M} = (S, R, L, s)$, there is an initial structure $\mathcal{K}' = (\mathcal{M}', s')$ s.t. $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models Res$.

As we can see that the p can only occur in the right of a clause, we will prove this proposition from the following several points.

(1) We consider there are global clauses in *Res* (the other cases are sub-cases of this one), then for each $C = \top \to C_1 \vee l \in Res$:

(a) If there does not exist a clause $C' \in Res$ s.t. C and C' are resolvable on p, this means there is no other clauses in *Res* except *Pt*-sometime clauses C' containing $\neg l$ with $Pt \in \{\mathrm{A}, \mathrm{E}\}$.

If $p \notin Var(C')$, for each $\mathcal{K} = (\mathcal{M}, s) \in Mod($*Removing_atoms*$(Res, V))$ we can construct $(\mathcal{M}', s')$ as follows: Let $\mathcal{M}' = (S, R, L', s)$ (i.e. $s' = s$), in which L' is the same as L except for each $s_1 \in S$ if $(\mathcal{M}, s_1) \not\models C_1 \vee l$ then let $L'(s_1) = L(s_1) \cup \{p\}$ if $l = p$ else $L'(s_1) = L(s_1) - \{p\}$.

If $C' = Q \to Pt\mathrm{F}\neg l$, without loss of generality, we assume $l = p$. For each $\mathcal{K} = (\mathcal{M}, s) \in$ *Mod*$($*Removing_atoms*$(Res, V))$, we construct $(\mathcal{M}', s')$ as follows: let $\mathcal{M}' = (S', R', L', s')$ with $S' = S$, $R' = R$, $s' = s$, and $L' = L$ except that for each $s \in S'$, we have $L'(s) = L(s) - \{Q\}$ if Q is an atom (if Q is a term, then we can delete the atoms which occurring in Q positively and add the atoms which occurring in Q negatively) and $L'(s) = L(s) \cup \{p\}$ if $(\mathcal{M}, s) \not\models C_1$ else $L'(s) = L(s)$. It is easy to check that $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models \mathit{Res}$.

(b) If there are some clauses $C' \in \mathit{Res}$ s.t. C and C' are resolvable on p:

(i) If $C' = Q \rightarrow Pt\mathrm{X}(C_2 \vee \neg l)$ (we let $Pt = \mathrm{G}$, we can prove similarly for $Pt = \mathrm{E}$), then we have $Q \rightarrow \mathrm{GX}(C_1 \vee C_2) \in \mathit{Res}$. And then for each $\mathcal{K} = (\mathcal{M}, s) \in \mathit{Mod}(\mathit{Removing\_atoms}(\mathit{Res}, V))$, we construct $(\mathcal{M}', s')$ as follows: Let $\mathcal{M}' = (S, R, L', s)$ (i.e., $s' = s$), in which L' is the same as L except for each $s_1 \in S$ if $(\mathcal{M}, s_1) \not\models Q$ then for each $(s_1, s_2) \in R$ if $(\mathcal{M}, s_2) \not\models C_1$ then let $L'(s_2) = L(s_2) \cup \{p\}$ if $l = p$ else $L'(s_2) = L(s_2) - \{p\}$, else if $(\mathcal{M}, s_2) \models C_1 \wedge \neg C_2$ then let $L'(s_2) = L(s_2) - \{p\}$ if $l = p$ else $L'(s_2) = L(s_2) \cup \{p\}$; else if $(\mathcal{M}, s_2) \models \neg C_1 \wedge C_2$ then let $L'(s_2) = L(s_2) \cup \{p\}$ if $l = p$ else $L'(s_2) = L(s_2) - \{p\}$. It is easy to check that $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models C' \wedge C$.

(ii) If $C' = Q \rightarrow Pt\mathrm{F}\neg l$. Without loss of generality, we assume $l = p$ for convenience. To make C and C' are resolvable on p, there must be a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses $\{P^1_1 \rightarrow *C^1_1, \ldots, P^1_{m_1} \rightarrow *C^1_{m_1}, P^n_1 \rightarrow *C^n_1, \ldots, P^1_{m_n} \rightarrow *C^1_{m_n}\}$ s.t. $*$ is either empty or an operator in $\{\mathrm{GX}, \mathrm{E}_{\langle ind\rangle}\mathrm{X}\}$, which include $\neg C_1 \rightarrow l$, such that $\bigvee_{i=1}^n \bigwedge_{j=1}^{m_i} P^i_j \rightarrow \mathrm{EXEG}l$. Therefore, we get clause $C'' = \top \rightarrow \neg Q \vee \neg p \vee C_1$ by using ERES1 (similar for ERES2) and then $\top \rightarrow \neg Q \vee C_1$ by using SRES8 on C and C''. In this case, for any $\mathcal{K} = (\mathcal{M}, s) \in \mathit{Mod}(\mathit{Removing\_atoms}(\mathit{Res}, V))$, we construct $(\mathcal{M}', s')$ as follows: Let $\mathcal{M}' = (S, R, L', s)$ (i.e., $s' = s$) in which L' is the same as L except for each $s_1 \in S$ if $(\mathcal{M}, s_1) \models Q$ then let $L'(s_1) = L(s_1) - \{p\}$, else $L'(s_1) = L(s_1) \cup \{p\}$. It is easy to check that $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models C' \wedge C$.

(iii) We can consider other clauses similarly and obtained that $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models \mathit{Res}$.

(2) We consider the Pt-step clauses. Let $C \in \mathit{Res}$ is $Q \rightarrow \mathrm{AX}(C_1 \vee \neg l)$. Without loss of generality, we assume there are some clauses $C' \in \mathit{Res}$ such that C and C' are resolvable on p and $l = p$.

If $C' = Q_1 \rightarrow Pt\mathrm{X}(C_2 \vee l)$ (Let $Pt = \mathrm{E}_{ind}$, we can prove similarly for $Pt = \mathrm{A}$), then we have $Q \wedge Q_1 \rightarrow \mathrm{E}_{ind}\mathrm{X}(C_1 \vee C_2) \in \mathit{Res}$. And then for each $\mathcal{K} = (\mathcal{M}, s) \in \mathit{Mod}(\mathit{Removing\_atoms}(\mathit{Res}, V))$, we construct $(\mathcal{M}', s')$ as follows: Let $\mathcal{M}' = (S, R, L', s)$ (i.e. $s' = s$), in which L' is the same as L except for each $s_1 \in S$

(i) if $(\mathcal{M}, s_1) \not\models Q \wedge Q_1$ then "if $(\mathcal{M}, s_1) \models \neg Q \wedge Q_1$ then (if $(\mathcal{M}, s'_2) \not\models C_2$ for $(s_1, s'_2) \in \pi_s^{\langle ind\rangle}$ then let $L'(s'_2) = L(s'_2) - \{p\}$ else $L'(s'_2) = L(s'_2)$), else if $(\mathcal{M}, s_1) \models Q \wedge \neg Q_1$ then for each $(s_1, s_2) \in R$ (if $(\mathcal{M}, s_2) \not\models C_1$ then let $L'(s_2) = L(s_2) \cup \{p\}$ else $L'(s'_2) = L(s'_2)$), else let $L'(s'_2) = L(s'_2)$".

(ii) else if $(\mathcal{M}, s_1) \models Q \wedge Q_1$ then we have $(\mathcal{M}, s'_2) \models C_1 \vee C_2$ for $(s_1, s_2) \in \pi_s^{\langle ind\rangle}$. Therefore, if $(\mathcal{M}, s'_2) \models C_1 \wedge \neg C_2$ then $L'(s'_2) = L(s'_2) - \{p\}$, else if $(\mathcal{M}, s'_2) \models \neg C_1 \wedge C_2$ then let $L'(s_2) = L(s_2) \cup \{p\}$ else $L'(s'_2) = L(s'_2)$. For other state s_2 with $(s_1, s_2) \in R$ and $s_2 \neq s'_2$, if $(\mathcal{M}, s_1) \models Q$ and $(\mathcal{M}, s_2) \models \neg C_1$ then let $L'(s_2) = L(s_2) \cup \{p\}$ else $L'(s'_2) = L(s'_2)$.

It is easy to check that $\mathcal{K} \leftrightarrow_{V''} \mathcal{K}'$ and $\mathcal{K}' \models C' \wedge C$, in which $\mathcal{K}' = (\mathcal{M}', s')$. ■

Lemma 8 *Let* $\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi$ *be a* $\mathrm{SNF}^g_{\mathrm{CTL}}$ *formula, then we have*

$$\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi \equiv \varphi \vee \mathrm{E}_{\langle ind \rangle}\mathrm{X}\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi.$$

Proof: ($\Rightarrow$) Let $(\mathcal{M}, s_0) \in \mathit{Mod}(\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi)$, then there exists a path $\pi_{s_0}^{\langle ind \rangle} = (s_0, s_1, \dots)$ s.t. $(\mathcal{M}, s_j) \models \varphi$ for some $s_j \in \pi_s^{\langle ind \rangle}$ with $j \geq 0$. In this case, we can see either $j = 0$ or $j > 0$, then we have $(\mathcal{M}, s_0) \models \varphi \vee \mathrm{E}_{\langle ind \rangle}\mathrm{X}\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi$.

($\Leftarrow$) Let $(\mathcal{M}, s_0) \in \mathit{Mod}(\varphi \vee \mathrm{E}_{\langle ind \rangle}\mathrm{X}\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi)$, then we have $(\mathcal{M}, s_0) \models \varphi$ or there exists a path $\pi_{s_0}^{\langle ind \rangle} = (s_0, s_1, \dots)$ s.t. $(\mathcal{M}, s_1) \models \mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi$. Therefore, we have $(\mathcal{M}, s_0) \models \mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi$ by the semantic of $\mathrm{E}_{\langle ind \rangle}\mathrm{F}$. ■

Lemma 9 *Let* P, P_i *and* φ_i *be CTL formulas, then*

(i) $\bigwedge_{i=1}^n (P \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}\varphi_i) \equiv_{\langle \emptyset, \{ind\} \rangle} P \to \mathrm{EX} \bigwedge_{i=1}^n \varphi_i$,

(ii) $\bigwedge_{i=1}^n (P_i \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}\varphi_i) \equiv_{\langle \emptyset, \{ind\} \rangle} \bigwedge_{e \in 2^{\{1,\dots,n\}} \setminus \{\emptyset\}} (\bigwedge_{i \in e} P_i \to \mathrm{EX}(\bigwedge_{i \in e} \varphi_i))$,

(iii) $\bigwedge_{i=1}^n (P \to \mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi_i) \equiv_{\langle \emptyset, \{ind\} \rangle} P \to \bigvee \mathrm{EF}(\varphi_{j_1} \wedge \mathrm{EF}(\varphi_{j_2} \wedge \mathrm{EF}(\cdots \wedge \mathrm{EF}\varphi_{j_n})))$, *where* $(j_1, \dots, j_n)$ *are sequences of all elements in* $\{1, \dots, n\}$,

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

(v) $(P_1 \to \varphi \vee \mathrm{E}_{\langle ind \rangle}\mathrm{X}\mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi_1) \wedge (P_2 \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}\varphi_2) \equiv_{\langle \emptyset, \{ind\} \rangle} (P_1 \to \varphi \vee \mathrm{EXEF}\varphi_1) \wedge (P_2 \to \mathrm{EX}\varphi_2) \wedge (P_1 \wedge P_2 \to ((\varphi \wedge \mathrm{EX}\varphi_2) \vee \mathrm{EX}(\varphi_2 \wedge \mathrm{EF}\varphi_1)))$.

Proof: (i) For each $(\mathcal{M}, s_0) \in \mathit{Mod}(\bigwedge_{i=1}^n (P \to \mathrm{E}_{\langle ind \rangle}\mathrm{X}\varphi_i))$, if $(\mathcal{M}, s_0) \models P$, then there exists $(s_0, s_1) \in [ind]$ s.t. $(\mathcal{M}, s_1) \models \varphi_1$, $\dots$, $(\mathcal{M}, s_1) \models \varphi_n$, and then there is $(s_0, s_1) \in R$ s.t. $(\mathcal{M}, s_1) \models \bigwedge_{i=1}^n \varphi_i$, i.e. $(\mathcal{M}, s_0) \models P \to \mathrm{EX} \bigwedge_{i=1}^n \varphi_i$.

For each $(\mathcal{M}, s_0) \in \mathit{Mod}(P \to \mathrm{EX} \bigwedge_{i=1}^n \varphi_i)$, we suppose there is $(s_0, s_1) \in R$ s.t. $(\mathcal{M}, s_1) \models \bigwedge_{i=1}^n \varphi_i$. It is easy to construct an initial$\mathcal{I}$-structure $(\mathcal{M}', s_0)$ s.t. $(\mathcal{M}', s_0)$ is identical to $(\mathcal{M}, s_0)$ except for the $(s_0, s_1) \in [ind]$, i.e. $(\mathcal{M}, s_0) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}', s_0)$.

(ii) (If part) For any model $(\mathcal{M}, s_0)$ of the left side of the equation, if there is $(\mathcal{M}, s_0) \models \bigwedge_{i=1}^m P_{j_i}$ with $j_i \in \{1, \dots, n\}$ and $1 \leq m \leq n$, then there is a next state s_1 of s_0 with $(s_0, s_1) \in [ind]$ s.t. $(\mathcal{M}, s_1) \models \bigwedge_{i=1}^m \varphi_{j_i}$. By the definition of $[ind]$, we have $(s_0, s_1) \in R$ and then $(\mathcal{M}, s_0) \models \bigwedge_{i=1}^m P_{j_i} \to \mathrm{EX}(\bigwedge_{i=1}^m P_{j_i}\varphi_{j_i})$. The other side can be similarly proved as (i).

(iii) (Only if part) For any model $(\mathcal{M}, s_0)$ of the right side of the equation, if there is $(\mathcal{M}, s_0) \models P$, then there exists a path $\pi_{s_0} = (s_0, s_1, \dots)$ s.t. $(\mathcal{M}, s_{j_i}) \models \varphi_{j_i}$ $(1 \leq i \leq n)$. It means we can construct an initial$\mathcal{I}$-structure $(\mathcal{M}', s_0)$ s.t. $(\mathcal{M}', s_0)$ is identical to $(\mathcal{M}, s_0)$ except for each (s_j, s_{j+1}) of π_{s_0} there is $(s_j, s_{j+1}) \in [ind]$ $(0 \leq j)$. It is easy to check $(\mathcal{M}', s_0) \models \bigwedge_{i=1}^n (P \to \mathrm{E}_{\langle ind \rangle}\mathrm{F}\varphi_i)$ and $(\mathcal{M}, s_0) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}', s_0)$. The other side can be shown similarly as in (ii).

Other results can be proved similarly. ■

Proposition ?? Let Π be a set of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses such that, for any index i, it contains at most one E-sometime clause whose index is i. Then $\Pi \equiv_{\langle \emptyset, I \rangle} \mathit{Removing\_index}(\Pi)$ where I is the set of indices in Π.

Proof: We know that there are no two E-sometime clauses that have the same index, and the Resolution process will not produce any E-sometime clauses by the transformation rules in Table 1 and Resolution rules in Table 2. Therefore, there are two steps to eliminate indices: We can use Lemma 8 to transform a E-sometime clause into a similar E-step clause at first, and then combine all the formulas which have the same index by the expressions, which are logically equivalent on tuple $\langle \emptyset, \{ind\} \rangle$, in Lemma 9 to eliminate the indices. It can be easily proved by Lemma 8 and Lemma 9. ■

Theorem 6 [Generalised Ackermanns Lemma] Let $\Gamma = \Delta \cup \Gamma'$ with $\Delta = \{\top \to \neg x \vee C_1, \dots, \top \to \neg x \vee C_n, x \to B_1, \dots, x \to B_m\}$ and Γ' is a subset of NI in Algorithm 1, where $x \in V'$, C_i $(1 \le i \le n)$ are classical propositional clauses that do not contain x, and B_j $(1 \le j \le m)$ are formulas of the disjunction (or conjunction) of formulas of form $Qt\mathcal{T}C$ with Qt $(\mathcal{T})$ is empty or $Qt \in \{\text{A}, \text{E}\}$, $\mathcal{T} \in \{\text{X}, \text{F}\}$ and C is a CTL formula that also do not contain x. If Γ' is positive w.r.t. x (i.e. each clause in Γ' is positive w.r.t. x), then $\Gamma'[x/\varphi] \equiv_{\langle \{x\}, \emptyset \rangle} \Gamma$ with $\varphi = \bigwedge_{i=1}^{n} C_i \wedge \bigwedge_{j=1}^{m} B_j$, where $\Gamma'[x/\varphi]$ is obtained from Γ' by replacing all x with φ.
Proof: $(\Rightarrow)$ For any model $(\mathcal{M}, s_0)$ of Γ, it is obvious that $(\mathcal{M}, s_0) \models \Gamma'[x/\varphi]$ by considering $\mathcal{T}'$ as a monotone function over x.

$(\Leftarrow)$ For any models $(\mathcal{M}, s_0)$ of $\Gamma'[x/\varphi]$ with $\mathcal{M} = (S, R, L, s_0)$, we construct an$\mathcal{I}$-initial structure $\mathcal{M}' = (S', R', L', s_0')$ with $S' = S$, $R' = R$, $s_0' = s_0$, and L' is the same with L except that for each $s' \in S'$, if $(\mathcal{M}', s') \models \varphi$ then let $L'(s') = L(s) \cup \{x\}$, else let $L'(s') = L(s) - \{x\}$ (i.e. $x \leftrightarrow \varphi$).

It is easy to check that $(\mathcal{M}, s_0) \leftrightarrow_{\langle \{x\}, \emptyset \rangle} (\mathcal{M}', s_0')$ and $(\mathcal{M}', s_0') \models \Gamma$. ■

Theorem ?? Let φ be a CTL formula and V be a set of atoms. Then, $\mathit{ERes}(\varphi, V) \equiv_{\langle V', \emptyset \rangle} \text{F}_{\text{CTL}}(\varphi, V)$, where V' is the set of fresh atoms in $\mathit{ERes}(\varphi, V)$.
Proof: Apparently, *RemA* is a special set that satisfies the one in Proposition **??**. Therefore, it is easy to show $\varphi \equiv_{\langle V' \cup V, \emptyset \rangle} \mathit{ERes}(\varphi, V)$ by Proposition **??**, **??**, **??**, and Theorem 6. And then we have $\mathit{ERes}(\varphi, V) \equiv_{\langle V', \emptyset \rangle} \text{F}_{\text{CTL}}(\varphi, V)$ due to $\varphi \equiv_{\langle V, \emptyset \rangle} \text{F}_{\text{CTL}}(\varphi, V)$, $\text{IR}(\mathit{ERes}(\varphi, V), V)$, and $\text{IR}(\text{F}_{\text{CTL}}(\varphi, V), V)$. ■

Lemma 10 *Let φ be a Classical propositional logic formula and V a set of atoms, then* $\text{ERes}(\varphi, V) \equiv \text{F}_{\text{CTL}}(\varphi, V)$.

Proof: Without loss of generality, we assume $V = \{p\}$ and φ a CNF, i.e., $\bigwedge_{i=1}^{n} C_i$ that contains two clauses (i.e., $C_k = p \vee C'$ and $C_j = \neg p \vee C''$), where $k, j \in \{1, 2, \dots, n\}$ and C', C'' are clauses. By using the transformation process, there is a set Π of $\text{SNF}^{g}_{\text{CTL}}$ clauses, with $\Pi = \{\mathbf{start} \to z, \top \to \neg z \vee C_1, \dots, \top \to \neg z \vee C_n\}$ (note that there is only an atom z is introduced in the transformation process).

In this case, we have a resolvent $\top \to \neg z \vee C' \vee C''$. We can easily check that $\mathit{ERes}(\varphi, V) \equiv \text{F}_{\text{CTL}}(\varphi, V)$ by Theorem **??** since z can be eliminated by Theorem 6. ■

Proposition 8 Let φ be a CTL formula, which does not contain temporal operators $Pt\mathcal{T}$ with $Pt \in \{\text{A}, \text{E}\}$ and $\mathcal{T} \in \{\text{U}, \text{G}\}$, and for each atom $p \in V$ if p and $\neg p$ appear in the same domain of temporal operator, then $\mathit{ERes}(\varphi, V) \equiv \text{F}_{\text{CTL}}(\varphi, V)$.
Proof: Without loss of generality, we assume $V = \{p\}$. For each CTL formula φ with this

form in the result, (for convenience, supposing $\varphi = \varphi_1 \wedge \text{AXEF}\varphi_2$ with $p \notin \mathit{Var}(\varphi_1)$ and φ_2 is a CNF that contains clauses $C_1 = \neg p \vee C'$ and $C_2 = p \vee C'$) it can be transformed into a set Π of $\text{SNF}^g_{\text{CTL}}$ clauses with $\Pi = \{\top \to \neg x \vee p \vee C', \top \to \neg x \vee \neg p \vee C', \dots\}$, in which x is a new atom introduced in this process. In this case, x can be eliminated. Hence, the other atoms introduced can also be eliminated since for each atom if it contains x then it can be replaced by $C_1 \vee C_2$ by Lemma 10. Therefore, we have $\mathit{ERes}(\varphi, V) \equiv \text{F}_{\text{CTL}}(\varphi, V)$ by Theorem **??**. ■

Proposition 9 Let φ be a CTL formula and $V \subseteq \mathcal{A}$. The time and space complexity of Algorithm 1 are $O((m+1)2^{4(n+n')})$. Where $n = |\mathit{Var}(\varphi)|$, $n' = |V'|$ (V' is set of atoms introduced in the Transform process) and m is the number of indices introduced during transformation.
Proof: It is mainly decided by the resolution process and the process of removing indices. The possible number of $\text{SNF}^g_{\text{CTL}}$ clauses under the given V, V' and Ind is $(m+1)2^{4(n+n')} + (m*(n+n')+n+n'+1)2^{2(n+n')+1})$. ■

Proposition 10 **(dual)** Let V, q, φ, and ψ are like in Definition 5. The ψ is an SNC (a WSC) of q on V under φ iff $\neg\psi$ is a WSC (an SNC) of $\neg q$ on V under φ.
Proof: (i) Suppose ψ is the SNC of q. Then $\varphi \models q \to \psi$. Thus $\varphi \models \neg\psi \to \neg q$. So $\neg\psi$ is an SC of $\neg q$. Suppose ψ' is any other SC of $\neg q$: $\varphi \models \psi' \to \neg q$. Then $\varphi \models q \to \neg\psi'$, this means $\neg\psi'$ is an NC of q on V under φ. Thus $\varphi \models \psi \to \neg\psi'$ by the assumption. So $\varphi \models \psi' \to \neg\psi$. This proves that $\neg\psi$ is the WSC of $\neg q$. The proof of the other part of the proposition is similar.

(ii) The WSC case can be proved similarly with SNC case. ■

Proposition 11 Let Γ and α be two formulas, $V \subseteq \mathit{Var}(\alpha) \cup \mathit{Var}(\Gamma)$ and q be a new proposition not occurring in Γ and α. Then, a formula φ of V is the SNC (WSC) of α on V under Γ iff it is the SNC (WSC) of q on V under $\Gamma' = \Gamma \cup \{q \leftrightarrow \alpha\}$.
Proof: We prove this for SNC. The case for WSC is similar. Let $\mathit{SNC}(\varphi, \alpha, V, \Gamma)$ denote that φ is the SNC of α on V under Γ, and $\mathit{NC}(\varphi, \alpha, V, \Gamma)$ denote that φ is the NC of α on V under Γ.

($\Rightarrow$) We will show that if $\mathit{SNC}(\varphi, \alpha, V, \Gamma)$ holds, then $\mathit{SNC}(\varphi, q, V, \Gamma')$ will be true. According to $\mathit{SNC}(\varphi, \alpha, V, \Gamma)$ and $\alpha \equiv q$, we have $\Gamma' \models q \to \varphi$, which means φ is an NC of q on V under Γ'. Suppose φ' is any NC of q on V under Γ', then $\text{F}_{\text{CTL}}(\Gamma', q) \models \alpha \to \varphi'$ due to $\alpha \equiv q$, $\mathit{IR}(\alpha \to \varphi', \{q\})$, and Theorem 4, i.e., $\Gamma \models \alpha \to \varphi'$ by Lemma 5, this means $\mathit{NC}(\varphi', \alpha, V, \Gamma)$. Therefore, $\Gamma \models \varphi \to \varphi'$ by the definition of SNC and $\Gamma' \models \varphi \to \varphi'$. Hence, $\mathit{SNC}(\varphi, q, V, \Gamma')$ holds.

($\Leftarrow$) We will show that if $\mathit{SNC}(\varphi, q, V, \Gamma')$ holds, then $\mathit{SNC}(\varphi, \alpha, V, \Gamma)$ will be true. According to $\mathit{SNC}(\varphi, q, V, \Gamma')$, it's not difficult to know that $\text{F}_{\text{CTL}}(\Gamma', \{q\}) \models \alpha \to \varphi$ due to $\alpha \equiv q$, $\mathit{IR}(\alpha \to \varphi, \{q\})$, and Theorem 4, i.e., $\Gamma \models \alpha \to \varphi$ by Lemma 5, this mean-s $\mathit{NC}(\varphi, \alpha, V, \Gamma)$. Suppose φ' is any NC of α on V under Γ. Then $\Gamma' \models q \to \varphi'$ since $\alpha \equiv q$ and $\Gamma' = \Gamma \cup \{q \equiv \alpha\}$, which means $\mathit{NC}(\varphi', q, V, \Gamma')$. According to $\mathit{SNC}(\varphi, q, V, \Gamma')$, $\mathit{IR}(\varphi \to \varphi', \{q\})$ and Theorem 4, we have $\text{F}_{\text{CTL}}(\Gamma', \{q\}) \models \varphi \to \varphi'$, and $\Gamma \models \varphi \to \varphi'$ by Lemma 5. Hence, $\mathit{SNC}(\varphi, \alpha, V, \Gamma)$ holds. ■

Theorem 10 Let φ be a formula, $V \subseteq \mathit{Var}(\varphi)$ and $q \in \mathit{Var}(\varphi) - V$.

(i) $\mathrm{F}_{\text{CTL}}(\varphi \wedge q, (\mathit{Var}(\varphi) \cup \{q\}) - V)$ is an SNC of q on V under φ.

(ii) $\neg \mathrm{F}_{\text{CTL}}(\varphi \wedge \neg q, (\mathit{Var}(\varphi) \cup \{q\}) - V)$ is a WSC of q on V under φ.

Proof: We will prove the SNC part, while it is not difficult to prove the WSC part according to Proposition 10. Let $\mathcal{F} = \mathrm{F}_{\text{CTL}}(\varphi \wedge q, (\mathit{Var}(\varphi) \cup \{q\}) - V)$.

The "NC" part: It's easy to see that $\varphi \wedge q \models \mathcal{F}$ by the definition of forgetting in CTL. Hence, $\varphi \models q \rightarrow \mathcal{F}$, this means $\mathcal{F}$ is an NC of q on V under φ.

The "SNC" part: We will show that for all NC ψ' of q on V under φ (i.e $\varphi \models q \rightarrow \psi'$), there is $\varphi \models \mathcal{F} \rightarrow \psi'$. We know that if $\varphi \wedge q \models \psi'$ then $\mathcal{F} \models \psi'$ by Theorem 4 and $\mathit{IR}(\psi', (\mathit{Var}(\varphi) \cup \{q\}) - V)$. Therefore, we have $\varphi \wedge \mathcal{F} \models \psi'$ since ψ' is an NC of q on V under φ and then $\varphi \models \mathcal{F} \rightarrow \psi'$, i.e. $\mathcal{F}$ is the SNC of q on V under φ. ∎

Theorem 11 Let $\mathcal{K} = (\mathcal{M}, s)$ be an initial structure with $\mathcal{M} = (S, R, L, s_0)$ on the set $\mathcal{A}$ of atoms, $V \subseteq \mathcal{A}$ and $q \in V' = \mathcal{A} - V$. Then:

(i) the SNC of q on V under $\mathcal{K}$ is $\mathrm{F}_{\text{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge q, V')$.

(ii) the WSC of q on V under $\mathcal{K}$ is $\neg \mathrm{F}_{\text{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge \neg q, V')$.

Proof: (i) As we know that any initial structure $\mathcal{K}$ can be described as a characterizing formula $\mathcal{F}_{\mathcal{A}}(\mathcal{K})$, then the SNC of q on V under $\mathcal{F}_{\mathcal{A}}(\mathcal{K})$ is $\mathrm{F}_{\text{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge q, \mathcal{A} - V)$.

(ii) This is proved by the dual property. ∎