On Sufficient and Necessary Conditions in Bounded CTL: A Forgetting Approach

Renyan Feng[1,2] , **Erman Acar**[2,*] , **Stefan Schlobach**[2] , **Yisong Wang**[1,*] , **Wanwei Liu** [3]
[1]Guizhou University, P. R. China
[2]Vrije Universiteit Amsterdam, The Netherlands
[3]National University of Defense Technology, P. R. China
fengrenyan@gmail.com, {Erman.Acar, k.s.schlobach}@vu.nl, yswang@gzu.edu.cn, wwliu@nudt.edu.cn

Abstract

Computation Tree Logic (CTL) 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, some information content of such property might become irrelevant for the system due to various reasons, e.g., it might become obsolete by time, or perhaps infeasible due to practical difficulties. Then, the problem arises on how to subtract such piece of information without altering the relevant system behaviour or violating the existing specifications over a given signature. Moreover, in such a scenario, two crucial notions are informative: the strongest necessary condition (SNC) and the weakest sufficient condition (WSC) of a given property. To address such a scenario in a principled way, we introduce a forgetting-based approach in CTL and show that it can be used to compute SNC and WSC of a property under a given model and over a given signature. We study its theoretical properties and also show that our notion of forgetting satisfies existing essential postulates of knowledge forgetting. Furthermore, we analyse the computational complexity of some basic reasoning tasks for the fragment CTL_{AF} in particular.

1 Introduction

Computation Tree Logic (CTL) (Clarke and 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 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 information would be highly space-inefficient, the problem arises on how to remove it without altering the relevant system behaviour or violating the existing system specifications over a given signature. Consider the following example.

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 end of each selection, a final (d)ecision is taken.*

In Figure 1, this scenario is represented by the Kripke structure $\mathcal{M} = (S, R, L)$ with the initial state s_0 (called labelled 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) company can no longer support the production of sports cars. This means, all the manufacturing processes concerning sp are no more necessary and should be dropped from both the specifications and the Kripke structure for simplification.

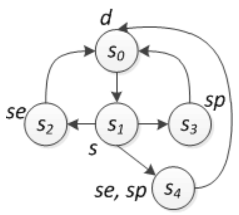

Figure 1: Car Engine Manufacturing Scenario

Similar scenarios like the one presented in Example 1 may arise in many different domains such as business-process modelling, software development, concurrent systems and more (Baier and Katoen 2008). Yet dropping some restrictions in a large and complex system or specification, without affecting the working system components or violating dependent specifications over a given signature, is a non-trivial task. Moreover, in such a scenario, two logical notions introduced by E. Dijkstra in (Dijkstra 1975) are highly informative: the *strongest necessary condition* (SNC) and the *weakest sufficient condition* (WSC) of a given specification. These correspond to the *most general consequence* and the *most specific abduction* of such specification, respectively.

To address these scenarios and to target the relevant notions SNC and WSC in a principled way, we employ a method based on formal verification.[1] In particular, we in-

[*]Corresponding author(s).

[1]This is especially useful for abstracting away the domain-dependent problems, and focusing on conceptual ones.

troduce 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, and Szalas 2001).

The rest of the paper is organised as follows. Next section reports about the related work. Section 3 introduces the notation and technical preliminaries. As key contributions, Section 4, introduces the notion of forgetting in bounded CTL. Moreover, it provides a model-theoretic characterization of CTL for (initial) Kripke structures, and studies the semantic properties of forgetting. In addition, a complexity analysis, concerning a relevant fragment $\mathrm{CTL}_{\mathrm{AF}}$, is carried out. Section 5 explores the relation between forgetting and SNC (WSC). Section 6 gives a model-based algorithm for computing forgetting in CTL and outline its complexity. Conclusion closes the paper.

Due to space restrictions, for most of the technical results, the actual proof is moved to the supplementary material [2], and instead an intuitive justification is put in place.

2 Related Work

The notions of SNC and WSC were considered in the scope of formal verification among others, in generating counterexamples (Dailler et al. 2018) and refinement of system (Woodcock and Morgan 1990). In addition, the WSC and SNC provide a method to generate successor state axioms from causal theories. In (Lin 2001), the SNC and WSC for a proposition q on a restricted subset of the propositional variables under a propositional theory T are computed based on the notion of forgetting. Besides, the SNC and WSC are generalized to first order logic (FOL) and a direct method that is based on *Second-Order Quantifier Elimination* (SOQE) technique has been proposed to automatically generate SNC and WSC in (Doherty, Lukaszewicz, and Szalas 2001).

Forgetting, which is a dual concept of *uniform interpolation* (Visser 1996; Konev, Walther, and Wolter 2009) and was first formally defined in propositional and FOL by Lin and Reiter (Lin and Reiter 1994; Eiter and Kern-Isberner 2019), can be traced back to the work of Boole on propositional variable elimination and the seminal work of Ackermann (Ackermann 1935). Usually, the definition of forgetting can be defined from the perspective of Strong/Semantic Forgetting and Weak Forgetting respectively (Zhang and Zhou 2010). In FOL, forgetting has often been studied as an instance of the SOQE problem. It is shown in (Lin and Reiter 1994) that the result of (strongly) forgetting an n-ary predicate P from a FOL formula φ is $\exists R \varphi[P/R]$, in which R is an n-ary predicate variable and $\varphi[X/Y]$ is a result of replacing every occurrence of X in φ by Y. The task of forgetting in FOL is to find a first-order formula that is equivalent to $\exists R \varphi[P/R]$. It is obvious that this is a SOQE problem. Similarly, the forgetting in description logics (DL) are also explored to create restricted views of ontologies by eliminating concept and role symbols from DL-based ontologies (Wang et al. 2010; Lutz and Wolter 2011; Zhao and Schmidt 2017).

[2]https://github.com/fengrenyan/proof-of-CTL.git

In propositional logic (PL), forgetting has often been studied under the name of variable elimination. In particular, the solution of forgetting a propositional variable p from a PL formula φ is $\varphi[p/\bot] \vee \varphi[p/\top]$ (Lin and Reiter 1994). In (Zhang and Zhou 2009), the authors define the knowledge forgetting of **S5** modal logic from the *strong forgetting* point of view to explore the relation between knowledge forgetting and knowledge update. They propose four general postulates (as we will revisit) for knowledge forgetting and show that these four postulates precisely characterize the notion of knowledge forgetting described above in **S5**. Furthermore, forgetting in logic programs under answer-set semantics are considered in (Zhang and Foo 2006; Eiter and Wang 2008; Wong 2009; Wang et al. 2014; Wang, Wang, and Zhang 2013).

However, existing forgetting definitions in PL and answer set programming are not directly applicable in modal logics. Moreover, existing forgetting techniques are not directly applicable in CTL due to its failure of uniform interpolation (Maksimova 1991). Similar to (Zhang and Zhou 2009), we research forgetting in bounded CTL from the semantic forgetting point of view and show that the result of forgetting some propositions from a CTL formula is always expressible in CTL. Furthermore, we show that our notion of forgetting satisfies those four postulates of forgetting presented in (Zhang and Zhou 2009). And last, we demonstrate how forgetting can be used to compute the SNC and WSC on a set of the propositions.

3 Notation and Preliminaries

Throughout this paper, we fix a finite set $\mathcal{A}$ of propositional variables (or atoms or propositions), use V, V' for subsets of $\mathcal{A}$ and $\overline{V} = \mathcal{A} - V$.

3.1 Kripke Structures in CTL

In general, a transition system can be described by a *Kripke structure* (see (Baier and Katoen 2008) for details). A Kripke structure is a triple $\mathcal{M} = (S, R, L)$ (Emerson 1990), where

- S is a finite nonempty set of states,[3],
- $R \subseteq S \times S$ and, for each $s \in S$, there is $s' \in S$ such that $(s, s') \in R$,
- $L : S \to 2^{\mathcal{A}}$ is a labeling function.

Given a Kripke structure $\mathcal{M} = (S, R, L)$, a *path* π of $\mathcal{M}$ is an infinite sequence $\pi = (s_0, s_1 s_2, \dots)$ of states 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 call 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 $s' \in S$. If s_0 is an initial state of $\mathcal{M}$, then we denote this Kripke structure $\mathcal{M}$ as (S, R, L, s_0) and call it an *initial structure*.

[3]Since CTL has finite model property (Emerson and Halpern 1985) we assume that the signature of states is fixed and finite, i.e., $S \subseteq \mathcal{S}$ with $\mathcal{S} = \{b_1, \dots, b_m\}$, such that any CTL formula with bounded length is satisfiable if and only if it is satisfiable in a such Kripke structure. Thus, there are only finite number of Kripke structures.

For a given initial structure $\mathcal{M} = (S, R, L, s_0)$ and $s \in S$, the *computation tree* $\text{Tr}_n^{\mathcal{M}}(s)$ of $\mathcal{M}$ (or simply $\text{Tr}_n(s)$), that has depth n and is rooted at s, is recursively defined as in (Browne, Clarke, and Grümberg 1988), for $n \geq 0$,

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

A K-*structure* (or K-*interpretation*) $\mathcal{K}$ consists of an initial structure $\mathcal{M} = (S, R, L, s_0)$ and a state $s \in S$, i.e., $\mathcal{K} = (\mathcal{M}, s)$. If in addition $s = s_0$ (i.e., $\mathcal{K} = (\mathcal{M}, s_0)$), then the K-structure is called an *initial* K-structure.

3.2 Syntax and Semantics of CTL

In the following we briefly review the basic syntax and semantics of the CTL (Clarke, Emerson, and Sistla 1986). The *signature* of the language $\mathcal{L}$ of CTL includes:

- a finite set of Boolean variables, called *atoms* of $\mathcal{L}$: $\mathcal{A}$;
- constant symbols: $\perp$ and $\top$;
- the classical connectives: $\vee$ and $\neg$;
- the path quantifiers: A and E;
- the temporal operators: X, F, G and U, that means 'neXt state', 'some Future state', 'all future states (Globally)' and 'Until', respectively;
- parentheses: (and).

The priorities for the CTL connectives are assumed to be in order as follows:

$$\neg, \text{EX}, \text{EF}, \text{EG}, \text{AX}, \text{AF}, \text{AG}, \wedge, \vee, \text{EU}, \text{AU}, \rightarrow,$$

where the leftmost (rightmost) symbol has the highest (lowest) priority. Then the *existential normal form (or ENF in short) formulas* of $\mathcal{L}$ are inductively defined via a Backus Naur form:

$$\phi ::= \perp \mid \top \mid p \mid \neg\phi \mid \phi \vee \phi \mid \text{EX}\phi \mid \text{EG}\phi \mid \text{E}(\phi \,\text{U}\, \phi) \quad (1)$$

where $p \in \mathcal{A}$. The formulas $\phi \wedge \psi$ and $\phi \rightarrow \psi$ are defined in a standard manner of propositional logic. The other form formulas of $\mathcal{L}$ are abbreviated using the forms of (1).

Throughout this article we shall assume that every formula of $\mathcal{L}$ has bounded size, where the size $|\varphi|$ of formula φ is its length over the alphabet of $\mathcal{L}$ (Emerson and Halpern 1985). As we will see later, this constraint will enable us to express the result of forgetting in CTL in the form of a (disjunctive) CTL formula. A *theory* of $\mathcal{L}$ is a finite set of formulas of $\mathcal{L}$. By abusing the notation, we identify a theory Π as the formula $\bigwedge \Pi$ whenever the context is clear.

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

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

Similar to the work in (Browne, Clarke, and Grümberg 1988; Bolotov 1999), only initial K-structures are considered to be candidate models in the following, unless otherwise noted. Formally, an initial K-structure $\mathcal{K}$ is a *model* of a formula ϕ whenever $\mathcal{K} \models \phi$. We denote $\mathit{Mod}(\phi)$ the set of models of ϕ. The formula ϕ is *satisfiable* if $\mathit{Mod}(\phi) \neq \emptyset$. Given two formulas ϕ_1 and ϕ_2, by $\phi_1 \models \phi_2$ we mean $\mathit{Mod}(\phi_1) \subseteq \mathit{Mod}(\phi_2)$, by $\phi_1 \equiv \phi_2$ we mean $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$. In this case, ϕ_1 is *equivalent* to ϕ_2. The set of atoms occurring in ϕ_1 is denoted by $\mathit{Var}(\phi_1)$. The formula ϕ_1 is *irrelevant to* the atoms in a set V (or simply V-*irrelevant*), written $\text{IR}(\phi_1, V)$, if there is a formula ψ with $\mathit{Var}(\psi) \cap V = \emptyset$ such that $\phi_1 \equiv \psi$.

4 Forgetting in CTL

In this section, we present the notion of forgetting in CTL and report its properties. First, we give a general definition of *bisimulation* between K-structures, called V-bisimulation, to define forgetting in CTL. The notion of bisimulation captures the idea that the computation trees of two structures are behaviourally same.

Second, the characterizing formula of an initial K-structure on some set V of propositions will be given. Then we will show that each initial K-structure can be captured by a CTL formula, and hence the result of forgetting V from formula φ can be expressed as a disjunction of the characterizing formulas of initial K-structures which are V-bisimilar with some models of φ. And last, the related properties, which include representation theorem, algebraic properties (i.e., Modularity, Commutativity and Homogeneity) of the forgetting operator, and the complexity results on the fragment CTL_{AF}, will be explored.

4.1 V-Bisimulation

In forgetting, one needs to express bisimulation w.r.t. different sets of atomic variables explicitly under a single setting (Zhang and Zhou 2009). Therefore, in this subsection, we define a notion of V-bisimulation $\mathcal{B}^V$ which is a *bisimulation w.r.t. a set V of atomic propositions* for our aims.

In order to introduce the actual notion, we start with the construction of V-bisimulation up to a certain degree (of depth) $n \in \mathbb{N}$ in the computation trees (denoted by $\mathcal{B}_n^V$) which we will introduce next: Let $V \subseteq \mathcal{A}$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ with $i \in \{1, 2\}$ and $\mathcal{M}_i = (S_i, R_i, L_i, s_0^i)$.

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

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

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

In the rest of the paper, by bisimulation, we shall only refer to V-bisimulation. So to ease the notation, from now on we will omit the superscript V in $\mathcal{B}_i^V$ and write $\mathcal{B}_i$ instead.

Now, we are ready to define the notion of V-bisimulation between K-structures.

Definition 1 (V-bisimulation)**.** *Let $V \subseteq \mathcal{A}$. Given two K-structures $\mathcal{K}_1$ and $\mathcal{K}_2$ are* V-bisimilar*, denoted $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$, if and only if $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_n$ for all $n \geq 0$. Moreover, let $i \in \{1, 2\}$, then two paths $\pi_i = (s_{i,1}, s_{i,2}, \ldots)$ of $\mathcal{M}_i$ are* V-bisimilar *if $\mathcal{K}_{1,j} \leftrightarrow_V \mathcal{K}_{2,j}$ for every $j \in \mathbb{N}_{\geq 1}$ where $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$.*

On the one hand, this notion can be considered as a simple generalization of the classical bisimulation-equivalence of Definition 7.1 in (Baier and Katoen 2008) when $V = \mathcal{A}$ and there is only one initial state (as in our case).

On the other hand, our definition of $\mathcal{B}_n$ is similar to the state equivalence (i.e., E_n) in (Browne, Clarke, and Grümberg 1988), yet it is different in the sense that ours is defined on K-structures, while state-equivalence is defined on states. Moreover, our notion is also different from the state-based bisimulation notion of Definition 7.7 in (Baier and Katoen 2008), which is defined for states of a given K-structure. [4] Note that if we defined V-bisimulation on states instead, then it would not be an equivalence relation anymore (as it will be shown in Lemma 1).

Example 2 (cont'd from Example 1)**.** *Let us call 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,[5] 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 and Katoen 2008) with either $\mathcal{K}_2$ or $\mathcal{K}_3$.*

When the underlying initial structures are clear from the context, we shall adopt the simplified notation $s_1 \leftrightarrow_V s_2$ emphasising states, to denote $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$.

Lemma 1. *The relation $\leftrightarrow_V$ is an equivalence relation.*

Next, we give some further key properties of $\leftrightarrow_V$ w.r.t. different Vs.

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

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

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

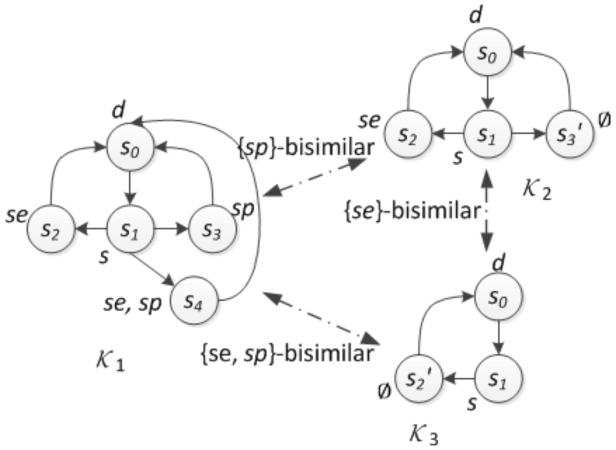

Figure 2: V-bisimulation between K-structures

(iii) for each path π_{s_1} of $\mathcal{M}_1$ there is a path π_{s_2} of $\mathcal{M}_2$ such that $\pi_{s_1} \leftrightarrow_{V_1} \pi_{s_2}$, and vice versa;

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

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

In Proposition 1, properties (i) to (iii) are the standard properties for V-bisimulation. Property (iv) shows that if a K-structure is V_1 and V_2-bisimilar with the other two K-structures, respectively, then those two K-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 K-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 K-structures are V-bisimilar, then they satisfy the same formula φ that does not contain any atoms in V, i.e., $\mathrm{IR}(\varphi, V)$. This idea has been formalized and shown in the following theorem.

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

Below, we illustrate this idea over an example.

Example 3 (cont'd from Example 2)**.** *Let $\varphi_1 = d \wedge \mathrm{EF}se \wedge \mathrm{AG}(se \rightarrow \mathrm{AX}d)$ and $\varphi_2 = d \wedge \mathrm{AX}se$ be two CTL formulae. They are $\{sp\}$-irrelevant. One can see that $\mathcal{K}_1$ and $\mathcal{K}_2$ in Figure 2 satisfy φ_1, but not φ_2.*

Next, we define the V-bisimulation between computation trees (of two initial structures). This construction will become useful when we define the characterizing formula of an initial K-structure using the characterizing formula of a computation tree.

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

- $L_1(s_1) - V = L_2(s_2) - V$,
- For every subtree $\mathrm{Tr}_{n-1}(s_i')$ of $\mathrm{Tr}_n(s_i)$, $\mathrm{Tr}_n(s_{(i \bmod 2)+1})$ has a subtree $\mathrm{Tr}_{n-1}(s'_{(i \bmod 2)+1})$ such that $\mathrm{Tr}_{n-1}(s_i') \leftrightarrow_V \mathrm{Tr}_{n-1}(s'_{(i \bmod 2)+1})$.

[4]As reported to us by an anonymous reviewer, there is also a notion of k-bisimulation (Kaushik et al. 2002) outside the realm of logic (but from database literature), which has a similar intuition to our $\mathcal{B}_n$, yet in the opposite direction: they consider bisimilarity through parents of a node (states), while we consider successors in relations. Again our notion is defined over K-structures.

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

The last condition in the above definition hold trivially for $n = 0$.

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

$$(s_1, s_2) \in \mathcal{B}_n \text{ iff } Tr_j(s_1) \leftrightarrow_V Tr_j(s_2) \text{ for every } 0 \leq j \leq n.$$

Proposition 2 says that a state s_1 of an initial structure is V-bisimilar to a state s_2 of another initial structure at a particular depth n if, and only if, all of the respective subtrees rooted at s_1 and s_2 until depth n are V-bisimilar.

Moreover, if two states s and s' from the same initial structure are not V-bisimilar, then the computation trees rooted at s and s', respectively, are not V-bisimilar at some depth $k \in \mathbb{N}$. This is shown in the following proposition.

Proposition 3. *Let $V \subseteq \mathcal{A}$, $\mathcal{M}$ be an initial structure and $s, s' \in S$ such that $s \not\leftrightarrow_V s'$. There exists a least k such that $Tr_k(s)$ and $Tr_k(s')$ are not V-bisimilar.*

4.2 Characterization of an Initial K-structure

In the following, we present characterizing formulas of initial K-structures over a signature to characterize the $\leftrightarrow_V$-class of an initial K-structure. [6]

To start with, we give the definition of characterizing formulas of computation trees.

Definition 2. *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be an initial structure and $s \in S$. The* characterizing formula *of the computation tree $Tr_n(s)$ on V, written $\mathcal{F}_V(Tr_n(s))$, is defined recursively as:*

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

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

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

for $k \geq 0$.

The characterizing formula of a computation tree formally exhibits the content of each node in V (i.e., atoms in V that are *true* if they are in the label of this node of the computation tree, and *false* otherwise) and the temporal relation between states recursively. Clearly, $\mathcal{F}_V(\text{Tr}_0(s))$ expresses the content of node s in terms of V, the conjunction with EX part guarantees that each direct successor s' of s is captured by a CTL formula until depth k, and the AX part guarantees that for each direct successor s' of s there exists another direct successor s'' of s such that s'' is V-bisimilar to s' until depth k.

The following result shows that the V-bisimulation between two computation trees implies the semantic equivalence of the corresponding characterizing formulas.

[6]Similar approaches has been taken in the literature e.g., in (Mycielski, Rozenberg, and Salomaa 1997), a class (namely, $\equiv_{\overline{k}}$-class) of structures of monadic formulas has been characterized by Hintikka formulae (Hintikka 1953). Another example is Yankov-Fine construction in (Yankov 1968).

Lemma 2. *Let $V \subseteq \mathcal{A}$, and $\mathcal{M}, \mathcal{M}'$ be two initial structures, $s \in S$, $s' \in S'$ and $n \geq 0$. If $Tr_n(s) \leftrightarrow_{\overline{V}} Tr_n(s')$, then $\mathcal{F}_V(Tr_n(s)) \equiv \mathcal{F}_V(Tr_n(s'))$.*

In Lemma 2, let $s' = s$. Then, it is easy to see that for any formula φ of V, if φ is a characterizing formula of $\text{Tr}_n(s)$ then $\varphi \equiv \mathcal{F}_V(\text{Tr}_n(s))$.

The notion of V-bisimulation and Proposition 3 naturally induce a complementary notion, so-called V-*distinguishability*, which will turn out to be useful in defining the characterizing formula of an initial K-structure. In particular, we will say that two states s and s' of $\mathcal{M}$ in Proposition 3 are V-*distinguishable* if $s \not\leftrightarrow_{\overline{V}} s'$, and write that $\text{dis}_V(\mathcal{M}, s, s', k)$, where we assume k to be the smallest natural number which makes s and s' V-distinguishable. Furthermore, we say that an initial structure $\mathcal{M}$ is V-distinguishable if there are two states s and s' in $\mathcal{M}$ that are V-distinguishable. Then given an initial structure $\mathcal{M}$ and a set V of atoms, the smallest value of k which ensures V-distinguishability is in question. We shall call such a k as the *characterization number* of $\mathcal{M}$ w.r.t. V and define it formally as

$$ch(\mathcal{M}, V) = \begin{cases} \max\{k \mid s, s' \in S \text{ and } \text{dis}_V(\mathcal{M}, s, s', k)\}, \\ \qquad\qquad\qquad\qquad \mathcal{M} \text{ is } V\text{-distinguishable;} \\ \min\{k \mid \mathcal{B}_k = \mathcal{B}_{k+1}\}, \qquad \text{otherwise.} \end{cases}$$

since it will be crucial in defining the characterization formula (for a given initial K-structure).

Observe that the $ch(\mathcal{M}, V)$ always exists for every initial structure $\mathcal{M}$ and $V \subseteq \mathcal{A}$: If there are two states s_1 and s_2 such that s_1 and s_2 are V-distinguishable, then the characterization number exists by definition. In the extreme case, if for all s, s' in $\mathcal{M}$, $((\mathcal{M}, s), (\mathcal{M}, s')) \in \mathcal{B}_k$ for all $k \geq 0$, and $\mathcal{B}_k = \mathcal{B}_{k+1}$ (since the set of states in $\mathcal{M}$ is always finite), then the characterization number is 0.

Intuitively, given a state $s \in S$ of $\mathcal{M}$, the characterization number c of $\mathcal{M}$ divides the states in $\mathcal{M}$ into two classes: The one which contains those states s' until depth c such that $(\mathcal{M}, s') \models \mathcal{F}_V(\text{Tr}_c(s))$, and the other which contains the remaining states. Now, we are finally ready to define the characterizing formula of an initial K-structure.

Definition 3 (Characterizing Formula)**.** *Let $V \subseteq \mathcal{A}$, and $\mathcal{K} = (\mathcal{M}, s_0)$ be an initial* K*-structure with $c = ch(\mathcal{M}, V)$, and for every state $s' \in S$ of $\mathcal{M}$, $T(s') = \mathcal{F}_V(Tr_c(s'))$. Then, the* characterizing formula *$\mathcal{F}_V(\mathcal{K})$ of $\mathcal{K}$ on V is:*

$$T(s_0) \wedge$$

$$\bigwedge_{s \in S} \text{AG}\left(T(s) \rightarrow \bigwedge_{(s,s') \in R} \text{EX}T(s') \wedge \text{AX}(\bigvee_{(s,s') \in R} T(s'))\right)$$

Here, $T(s_0)$ ensures that the K-structure starts from the initial state, and the remaining part ensures that we go deep enough in the computation tree (i.e., through all possible transitions from every state $s \in S$) to detect any two V-distinguishable states s and s' (which would then imply $T(s) \not\equiv T(s')$). As a remark on notation, sometimes we shall need to express the initial structure and the initial state explicitly, then we will use the rather transparent notation i.e., $\mathcal{F}_V(\mathcal{M}, s_0)$ (instead of $\mathcal{F}_V(\mathcal{K})$).

One can observe that $\mathrm{IR}(\mathcal{F}_V(\mathcal{M}, s_0), \overline{V})$. Besides, given a set of atomic propositions V, any initial K-structure has its own unique characterizing formula on V. As we will see later, the characterizing formula will play a crucial role in showing important properties of forgetting, as well as in our main contribution which is computing the SNC and WSC of a CTL formula under an initial K-structure.

The following example illustrates how one can compute a characterizing formula:

Example 4 (cont'd from Example 2)**.** *Reconsider the $\mathcal{K}_2 = (\mathcal{M}, s_0)$ in Figure 3, illustrated on the left side (originally introduced in Figure 2). The corresponding computation trees are listed on the right side: from left to right, they are rooted at s_0 with depth 0, 1, 2 and 3, respectively. For simplicity, the labels of the nodes in the trees are omitted (See Figure 2 for the actual labels). Let $V = \{d\}$ then $\overline{V} = \{s, se\}$.*

We can see that $Tr_0(s_1) \leftrightarrow_{\overline{V}} Tr_0(s_2)$, since $L(s_1) - \overline{V} = L(s_2) - \overline{V}$. Moreover, $Tr_1(s_1) \not\leftrightarrow_{\overline{V}} Tr_1(s_2)$, since there is $(s_1, s_2) \in R$ such that for any $(s_2, s') \in R$, it is the case that $L(s_2) - \overline{V} \neq L(s') - \overline{V}$ (because there is only one direct successor $s' = s_0$). Hence, we have s_1 and s_2 which are V-distinguishable and $dis_V(\mathcal{M}, s_1, s_2, 1)$. Similarly, we have $dis_V(\mathcal{M}, s_0, s_1, 0)$, $dis_V(\mathcal{M}, s_0, s_2, 0)$ and $dis_V(\mathcal{M}, s_0, s_3', 0)$. Furthermore, we can see that $s_2 \leftrightarrow_{\overline{V}} s_3'$. Therefore, $ch(\mathcal{M}, V) = \max\{k \mid s, s' \in S$ and $dis_V(\mathcal{M}, s, s', k)\} = 1$. And we have the following:

$$\begin{aligned}
&\mathcal{F}_V(Tr_0(s_0)) = d, \qquad \mathcal{F}_V(Tr_0(s_1)) = \neg d,\\
&\mathcal{F}_V(Tr_0(s_2)) = \neg d, \qquad \mathcal{F}_V(Tr_0(s_3')) = \neg d,\\
&\mathcal{F}_V(Tr_1(s_0)) = \mathrm{EX}\neg d \wedge \mathrm{AX}\neg d \wedge d \equiv \mathrm{AX}\neg d \wedge d,\\
&\mathcal{F}_V(Tr_1(s_1)) = \mathrm{EX}\neg d \wedge \mathrm{EX}\neg d \wedge \mathrm{AX}(\neg d \vee \neg d) \wedge \neg d\\
&\qquad \equiv \mathrm{AX}\neg d \wedge \neg d,\\
&\mathcal{F}_V(Tr_1(s_2)) = \mathrm{EX}d \wedge \mathrm{AX}d \wedge \neg d \equiv \mathrm{AX}d \wedge \neg d,\\
&\mathcal{F}_V(Tr_1(s_3')) \equiv \mathcal{F}_V(Tr_1(s_2)),\\
&\mathcal{F}_V(\mathcal{M}, s_0) \equiv \mathrm{AX}\neg d \wedge d \wedge\\
&\qquad \mathrm{AG}(\mathrm{AX}\neg d \wedge d \to \mathrm{AX}(\mathrm{AX}\neg d \wedge \neg d)) \wedge\\
&\qquad \mathrm{AG}(\mathrm{AX}\neg d \wedge \neg d \to \mathrm{AX}(\mathrm{AX}d \wedge \neg d)) \wedge\\
&\qquad \mathrm{AG}(\mathrm{AX}d \wedge \neg d \to \mathrm{AX}(\mathrm{AX}\neg d \wedge d)).
\end{aligned}$$

The following result shows that there is a correspondence between the semantic equivalence of characterizing formulae and the initial K-structures which are V-bisimilar. That is, two initial K-structures are V-bisimilar if, and only if their characterizing formulae are semantically equivalent. This means, characterizing formula characterizes initial K-structures which are equivalent up to V-bisimulation.

Theorem 2. *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{M}' = (S', R', L', s_0')$ be two initial structures. Then,*

(i) $(\mathcal{M}', s_0') \models \mathcal{F}_V(\mathcal{M}, s_0)$ iff $(\mathcal{M}, s_0) \leftrightarrow_{\overline{V}} (\mathcal{M}', s_0')$;

(ii) $s_0 \leftrightarrow_{\overline{V}} s_0'$ implies $\mathcal{F}_V(\mathcal{M}, s_0) \equiv \mathcal{F}_V(\mathcal{M}', s_0')$.

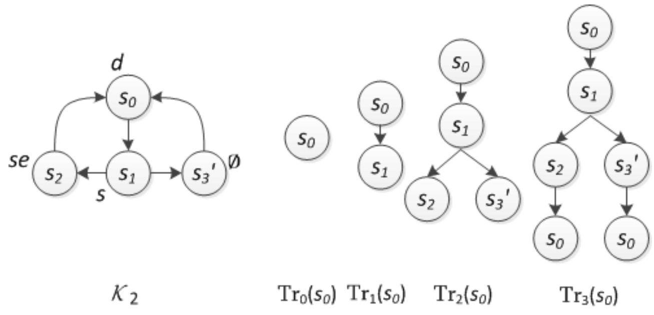

Figure 3: On the left side, $\mathcal{K}_2$ (aforementioned in Figure 2), and on the right side, the corresponding computation trees of depth 0, 1, 2 and 3, respectively. Labels of the nodes are omitted for simplicity.

It is noteworthy that under our assumption of bounded size (of a CTL formula), say n, it will be sufficient to consider the models of formulas within a state space S satisfying $|S| = n8^n$ (Emerson and Halpern 1985). Any other model must be bisimilar to some model within the state space, and their characterizing formulas are equivalent due to Theorem 2. Therefore, given a formula of size within the bound, only a finite number of such initial K-structures need to be considered as the candidate models. This fact is expressed in the following lemma.

Lemma 3. *Let φ be a formula. We have*

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

Yet Lemma 3 has an additional message: Any CTL formula can be expressed in the form of a disjunction of the characterizing formulae of its models. This fact will be crucial in the results we present in next sections.

4.3 Semantic Properties of Forgetting in CTL

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 (Forgetting)**.** *Let $V \subseteq \mathcal{A}$ and ϕ be a formula. A formula ψ with $Var(\psi) \cap V = \emptyset$ is a* result of forgetting V from ϕ *(denoted as $\mathrm{F}_{\mathrm{CTL}}(\phi, V)$), if*

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

Realize that Definition 4 implies if both ψ and ψ' are results of forgetting V from ϕ, then $Mod(\psi) = Mod(\psi')$, i.e., ψ and ψ' have the same models. In this sense, the result of forgetting V from ϕ is unique (up to semantic equivalence). By Lemma 3, such a formula always exists, which is equivalent to

$$\bigvee_{\mathcal{K} \in \{\mathcal{K}' \mid \exists \mathcal{K}'' \in Mod(\phi) \text{ and } \mathcal{K}'' \leftrightarrow_V \mathcal{K}'\}} \mathcal{F}_{\overline{V}}(\mathcal{K}).$$

At this point, it is important to emphasize that, the notion of forgetting we have defined for CTL respects the classical forgetting defined for propositional logic (PL) (Lin and Reiter 1994). To see this, assume that φ is a PL formula and $p \in \mathcal{A}$, then $Forget(\varphi, p)$ is a result of forgetting p from φ; that is, $Forget(\varphi, p) \equiv \varphi[p/\bot] \vee \varphi[p/\top]$.

That way, given a set $V \subseteq \mathcal{A}$, one can recursively define *Forget*$(\varphi, V \cup \{p\}) =$ *Forget*(*Forget*$(\varphi, p), V)$, where *Forget*$(\varphi, \emptyset) = \varphi$. Using this insight, the following result shows that the classical notion of forgetting (for PL (Lin and Reiter 1994)) is a special case of forgetting in CTL.

Theorem 3. *Let φ be a PL formula and $V \subseteq \mathcal{A}$, then*

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

In (Zhang and Zhou 2009), authors give four postulates concerning knowledge forgetting in **S5** modal logic (also called *forgetting postulates*) which can be considered as desirable properties of such a notion. In the following, we first list these postulates, and then show that our notion of forgetting in CTL satisfies them.

Forgetting postulates (Zhang and Zhou 2009) are:

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

where $V \subseteq \mathcal{A}$, φ is a formula and φ' is a result of forgetting V from φ. Intuitively, the postulate (**W**) says, forgetting weakens the original formula; the postulates (**PP**) and (**NP**) say that forgetting results have no effect on formulas that are irrelevant to forgotten atoms; the postulate (**IR**) states that forgetting result is irrelevant to forgotten atoms. It is noteworthy that they are not all orthogonal e.g., (**NP**) is a consequence of (**W**) and (**PP**). Nonetheless, we prefer to list them all, in order to outline the basic intuition behind them.

Theorem 4 (Representation Theorem)**.** *Let φ and φ' be CTL formulas and $V \subseteq \mathcal{A}$. The following statements are equivalent:*

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

(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).*

Proof. $(i) \Leftrightarrow (ii)$. To prove this, it is enough to show that:

$$\begin{aligned}&\mathit{Mod}(\mathrm{F}_{\mathrm{CTL}}(\varphi, V)) = \mathit{Mod}(\{\phi|\varphi \models \phi, \mathrm{IR}(\phi, V)\})\\ &= \mathit{Mod}(\bigvee_{\mathcal{M}, s_0 \in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}-V}(\mathcal{M}, s_0)).\end{aligned}$$

First, suppose that $(\mathcal{M}', s_0')$ is a model of $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$. Then there exists an initial K-structure $(\mathcal{M}, s_0)$ which is a model of φ and $(\mathcal{M}, s_0) \leftrightarrow_V (\mathcal{M}', s_0')$. By Theorem 1, we have $(\mathcal{M}', s_0') \models \phi$ for all ϕ such that $\varphi \models \phi$ and $\mathrm{IR}(\phi, V)$. Thus, $(\mathcal{M}', s_0')$ is a model of the theory $\{\phi \mid \varphi \models \phi, \mathrm{IR}(\phi, V)\}$.

Second, suppose that $(\mathcal{M}', s_0')$ is a model of $\{\phi \mid \varphi \models \phi, \mathrm{IR}(\phi, V)\}$. Thus, $(\mathcal{M}', s_0') \models \bigvee_{(\mathcal{M}, s_0) \in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}-V}(\mathcal{M}, s_0)$ since $\bigvee_{(\mathcal{M}, s_0) \in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}-V}(\mathcal{M}, s_0)$ is irrelevant to V and $\varphi \models \bigvee_{(\mathcal{M}, s_0) \in \mathit{Mod}(\varphi)} \mathcal{F}_{\mathcal{A}-V}(\mathcal{M}, s_0)$ by Lemma 3.

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

$(ii) \Rightarrow (iii)$. This is rather straightforward, so we put it into the supplementary material.

$(iii) \Rightarrow (ii)$. By Positive Persistence, we have $\varphi' \models \{\phi \mid \varphi \models \phi, \mathrm{IR}(\phi, V)\}$. The $\{\phi \mid \varphi \models \phi, \mathrm{IR}(\phi, V)\} \models \varphi'$ can be obtained from (**W**) and (**IR**). Thus, φ' is equivalent to $\{\phi \mid \varphi \models \phi, \mathrm{IR}(\phi, V)\}$. ☐

It is noteworthy that the postulate **IR** 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 5, it will become important (in reducing the SNC (WSC) of any CTL formula to the one of a proposition).

Lemma 4. *Let φ and α be two CTL formulae and $q \in \overline{Var(\varphi) \cup Var(\alpha)}$. Then* $\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge (q \leftrightarrow \alpha), q) \equiv \varphi$.

In what follows, we list other interesting 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 formula $\varphi \in CTL$, V a set of atoms and p an atom such that $p \notin V$. 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 5 (Commutativity)**.** *Let φ be a 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. They hold in both classical propositional logic and modal logic **S5** (Zhang and Zhou 2009). 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) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \wedge \psi_2, V) \models \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \wedge \mathrm{F}_{\mathrm{CTL}}(\psi_2, V)$*;*

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 $\phi \in CTL$,*

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

4.4 Complexity Results

In the following, we analyze the computational complexity of the various tasks regarding the forgetting in the fragment CTL_{AF}. The fragment CTL_{AF} of CTL, in which each formula contains only AF temporal connective, corresponds to specifications that are expected to hold in all branches eventually. Such properties are of special interest in concurrent systems e.g., mutual exclusion and waiting events (Baier and Katoen 2008). Our first result shows that the problem of model checking for forgetting of V from φ is NP-complete, if $\varphi \in \text{CTL}_{\text{AF}}$.

Proposition 7 (Model Checking). *Given an initial* K*-structure* $(\mathcal{M}, s_0)$*,* $V \subseteq \mathcal{A}$ *and* $\varphi \in CTL_{\text{AF}}$*, deciding* $(\mathcal{M}, s_0) \models^? \mathrm{F}_{\text{CTL}}(\varphi, V)$ *is* NP*-complete.*

In the following, we investigate some complexity results concerning forgetting and the logical entailment in this fragment.

Theorem 6 (Entailment). *Let* φ *and* ψ *be two* CTL_{AF} *formulas and* V *be a set of atoms. Then,*

(i) deciding $\mathrm{F}_{\text{CTL}}(\varphi, V) \models^? \psi$ *is co*-NP*-complete,*
(ii) deciding $\psi \models^? \mathrm{F}_{\text{CTL}}(\varphi, V)$ *is* Π_2^{P}*-complete,*
(iii) deciding $\mathrm{F}_{\text{CTL}}(\varphi, V) \models^? \mathrm{F}_{\text{CTL}}(\psi, V)$ *is* Π_2^{P}*-complete.*

Proof. (i) and (iii) is moved to supplementary material due to space restrictions. (ii) Membership: We consider the complement of the problem. Guess an initial K-structure $(\mathcal{M}, s_0)$ which has polynomial size in the size of ψ satisfying ψ and check $(\mathcal{M}, s_0) \not\models \mathrm{F}_{\text{CTL}}(\varphi, V)$. By Proposition 7, it is in Σ_2^{P}. So the original problem is in Π_2^{P}. Hardness: Let $\psi \equiv \top$. Then the problem is reduced to decide the validity of $\mathrm{F}_{\text{CTL}}(\varphi, V)$. Since propositional forgetting is a special case by Theorem 3, the hardness follows from the proof of Proposition 24 in (Lang, Liberatore, and Marquis 2003). $\square$

The following results are implications of Theorem 6.

Corollary 7. *Let* φ *and* ψ *be two* CTL_{AF} *formulas and* V *a set of atoms. Then*

(i) deciding $\psi \equiv^? \mathrm{F}_{\text{CTL}}(\varphi, V)$ *is* Π_2^{P}*-complete,*
(ii) deciding $\mathrm{F}_{\text{CTL}}(\varphi, V) \equiv^? \varphi$ *is co*-NP*-complete,*
(iii) deciding $\mathrm{F}_{\text{CTL}}(\varphi, V) \equiv^? \mathrm{F}_{\text{CTL}}(\psi, V)$ *is* Π_2^{P}*-complete.*

5 Necessary and Sufficient Conditions

In this section, we present the final key notions 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 K-structure and a set V of atoms. Let us give the formal definition.

Definition 5 (sufficient and necessary condition). *Let* ϕ *be a formula (or an initial* K*-structure),* ψ *be a 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* $\phi \models q \to \psi$*.*
- ψ *is a* sufficient condition *(SC in short) of* q *on* V *under* ϕ *if* $\phi \models \psi \to q$*.*
- ψ *is a* strongest necessary condition *(SNC in short) of* q *on* V *under* ϕ*, if it is a NC of* q *on* V *under* ϕ*, and* $\phi \models \psi \to \psi'$ *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 a SC of* q *on* V *under* ϕ*, and* $\phi \models \psi' \to \psi$ *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). The following result shows that the SNC and WSC are in fact dual notions.

Proposition 8 (Dual). *Let* V, q, φ *and* ψ *are defined as in Definition 5. Then,* ψ *is a SNC (WSC) of* q *on* V *under* φ *iff* $\neg\psi$ *is a WSC (SNC) of* $\neg q$ *on* V *under* φ*.*

In order to generalise Definition 5 to arbitrary formulas, one can replace q (in the definition) by 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 9. *Let* Γ *and* α *be two formulas,* $V \subseteq Var(\alpha) \cup Var(\Gamma)$ *and* q *be a new proposition not 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 5 (cont'd from Example 2). *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* $\mathcal{K}_2 \models \varphi \to \psi$*, which means* φ *is a 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 a SC of* ψ *on* V *under* $\mathcal{K}_2$*. Therefore, there is* $\varphi' \models \text{EX}(s \wedge \phi)$*, in which* ϕ *is a formula such that* $\phi \models \text{EX}se \vee \text{EX}\neg d$*. And then* $\phi \models \text{EX}\neg d$ *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 8. *Let* φ *be a formula,* $V \subseteq Var(\varphi)$ *and* $q \in Var(\varphi) - V$*.*

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

Algorithm 1: A Model-based CTL Forgetting Procedure

```
Input: A CTLformula $\varphi$ and a set $V$ of atoms
Output: $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$
1 $\psi \leftarrow \bot$;
2 foreach initial K-structure $\mathcal{K}$ (over $\mathcal{A}$ and $\mathcal{S}$) do
3     if $\mathcal{K} \not\models \varphi$ then continue;
4     foreach initial K-structure $\mathcal{K}'$ with $\mathcal{K} \leftrightarrow_V \mathcal{K}'$ do
5         $\psi \leftarrow \psi \vee \mathcal{F}_{\overline{V}}(\mathcal{K}')$;
6     end
7 end
8 return $\psi$;
```

(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 8, assume that $\beta = \mathrm{F}_{\mathrm{CTL}}(\varphi \wedge q, (Var(\varphi) \cup \{q\}) - V)$. Then, $\varphi \wedge q \models \beta$ by (**W**). Moreover, $\varphi \wedge q \models \beta$, and then β is a 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 (**PP**). 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 initial K-structure can be characterized by a CTL formula, by Theorem 8 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 9. *Let* $\mathcal{K} = (\mathcal{M}, s)$ *be an initial* K*-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* $\neg\mathrm{F}_{\mathrm{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge \neg q, V')$.

6 An Algorithm for Forgetting in CTL

The technical developments we have presented in previous sections naturally induce a procedure to compute forgetting in CTL. We think that it is useful to outline such a procedure explicitly in the form of an algorithm. It is a model-based approach (presented in Algorithm 1); that is, it will compute the forgetting applied to a formula, simply by considering all the possible models of that formula. Its correctness is guaranteed by Lemma 3 and Theorem 2.

The example we give below echoes the initial example which was given in the introduction, and finalizes the running example with a simple intuition of forgetting.

Example 6. *Recall the* K*-structure* $\mathcal{K}_1$ *given in Figure 2, and assume that we are given a property* $\alpha = \mathrm{EF}(se \wedge sp)$. *It is easy to see that* $\mathcal{K}_1$ *in Figure 2 satisfy* α. *If sp is intended to be removed, i.e., forgetting sp from* α, *then* $\mathrm{F}_{\mathrm{CTL}}(\alpha, \{sp\}) \equiv \mathrm{EF}se$. *Hence, the company can use the new specification* $\mathrm{EF}se$ *to guide the new production process (which guarantees that the sedan car is eventually produced).*

As we will show below, computing the forgetting by going through all the models is not very efficient, as one might expect. However, settling it is important from a theoretical point of view i.e., to see how costly is the naive approach.

Proposition 10. *Let* φ *be a CTL formula and* $V \subseteq \mathcal{A}$ *with* $|\mathcal{S}| = m$, $|\mathcal{A}| = n$ *and* $|V| = x$. *Then the space complexity is* $O((n-x)m^{2(m+2)}2^{nm}\log m)$ *and the time complexity of Algorithm 1 is at least the same as the space.*

As expected, Algorithm 1 has a high cost; namely, EXPSPACE complexity in the size of the state space and $\mathcal{A}$, which does not look encouraging. However, we believe that settling this result is important both from a theoretical and a practical point of view. Theoretically, it gives us a picture about the worst case, and urges us to come up with more efficient syntactical approaches which is a part of our future agenda. Moreover, we believe that model-based investigation and some of the structural observations we have made provide us with informative valuable insights, which in turn could be useful in designing future algorithms which can exploit these observations, and potentially could lead to even efficient approximations with provably good bounds. Such future developments might prove important in developing practical algorithms as well.

7 Concluding Remarks

Summary In this paper, we have presented the notion of forgetting for CTL which enables computing weakest sufficient and strongest necessary conditions of specifications. In doing so, we introduced and employed the notion of V-bisimulation which can be considered as a simple variable based generalisation of classical bisimulation. Furthermore, we have studied formal properties of forgetting, among them, homogeneity, modularity and commutativity. In particular, we have shown that our notion of forgetting satisfies the existing postulates of forgetting, which means it faithfully extends the notion of forgetting from classical propositional logic and modal logic **S5** to CTL. On the complexity theory side, we have investigated the model checking and the entailment problems of forgetting in the fragment $\mathrm{CTL}_{\mathrm{AF}}$, which turn out to be NP-complete and range from co-NP to Π_2^{P}-completeness, respectively. And finally, we proposed a model-based algorithm which computes the forgetting of a given formula and a set of variables, and outlined its complexity.

Future work Note that, 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.

Moreover, a further study regarding the computational complexity for other general fragments is required and part of the future research agenda. As mentioned in Section 6, these high complexity results are encouraging for other syntactic approaches e.g., proof-theoretic. Such investigation can be coupled with fine-grained parameterized analysis, as well as a search for approximation algorithms with provably good accuracy bounds.

Acknowledgements

We kindly thank all the anonymous reviewers whose comments improved this work to a great extent. Renyan Feng is funded by China Scholarship Council (CSC) grant number 201906670007. Renyan Feng and Yisong Wang is supported by the National Natural Science Foundation of P.R. China under Grants 61976065, 61370161 and U1836105. Erman Acar's research is funded by MaestroGraph research programme (NWO) with project number 612.001.552.

References

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

Baier, C., and Katoen, J. 2008. *Principles of Model Checking*. The MIT Press.

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

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

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

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

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

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

Doherty, P.; Lukaszewicz, W.; and Szalas, A. 2001. Computing strongest necessary and weakest sufficient conditions of first-order formulas. In *IJCAI'01*, 145–154.

Eiter, T., and Kern-Isberner, G. 2019. A brief survey on forgetting from a knowledge representation and reasoning perspective. *KI-Künstliche Intelligenz* 33(1):9–33.

Eiter, T., and Wang, K. 2008. Semantic forgetting in answer set programming. *Artif. Intell.* 172(14):1644–1672.

Emerson, E. A., and Halpern, J. Y. 1985. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of computer and system sciences* 30(1):1–24.

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

Hintikka, J. 1953. Distributive normal forms in the calculus of predicates.

Kaushik, R.; Naughton, J. F.; Bohannon, P.; and Shenoy, P. 2002. Updates for structure indexes. In *Proceedings of VLDB'02*, 239–250. Elsevier.

Konev, B.; Walther, D.; and Wolter, F. 2009. Forgetting and uniform interpolation in large-scale description logic terminologies. In *Twenty-First International Joint Conference on Artificial Intelligence*.

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional independence: Formula-variable independence and forgetting. *Journal of Artificial Intelligence Research* 18:391–443.

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

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

Lutz, C., and Wolter, F. 2011. Foundations for Uniform Interpolation and Forgetting in Expressive Description Logics. In *Proceedings of IJCAI'11*, 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.

Mycielski, J.; Rozenberg, G.; and Salomaa, A., eds. 1997. *Structures in Logic and Computer Science, A Selection of Essays in Honor of Andrzej Ehrenfeucht*, volume 1261 of *Lecture Notes in Computer Science*. Springer.

Visser, A. 1996. Uniform interpolation and layered bisimulation. In *Gödel'96 (Brno, 1996)*. 139–164.

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

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

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

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

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

Yankov, V. A. 1968. Three sequences of formulas with two variables in the positive propositional logic. *Izvestiya Rossiiskoi Akademii Nauk. Seriya Matematicheskaya* 32(4):880–883.

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

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

Zhang, Y., and Zhou, Y. 2010. Forgetting revisited. In *Proceedings of KR 2010*, 602–604. AAAI Press.

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