Forgetting in CTL to Compute Necessary and Sufficient Conditions

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 problems arises on how to subtract such piece of information without altering the relevant system behaviour or violating the existing specifications.

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. We study its theoretical properties and also show that our notion of forgetting satisfies existing essential postulates. Furthermore, we analyse the computational complexity of basic tasks, including various results for the relevant fragment CTL_{AF}.

1 Introduction

Consider a car-manufacturing company which produces two types of automobiles: a sedan car (basically a four-doored classical passenger car) and a sports car. No matter a sedan or a sports car, both production lines are subject to a single standard criterion which is indispensable: safety restrictions. Complementing such a shard feature, there are also differences aligned with these types in general. That is, a sedan car is produced with a small engine, while a sport car is produced with a large one. Moreover, due to its large amount of production, the sedan car is subject to some very restrictive low-carbon emission regulations, while a sports car is not. In the verge of shifting to an upcoming new engine technology, the company aims to adapt the sedan production to electrical engines. Such major shift in production also comes with one in regulations; electric sedans are not subject to low-carbon emission restrictions any more. In fact, due to the difference in its underlying technology, a sedan car drastically emit much less carbon, hence such standard is obsolete, and can be dropped. Yet, dropping some restrictions in a large and complex production system (like the one in automative industry) without affecting the working system components or violating dependent specifications is a non-trivial task.

Similar situations may arise in many different domains such as business-process modelling, software development, concurrent systems and more [Baier and Katoen, 2008]. In general, some information content of such property might become irrelevant for the system due to various reasons e.g., it might become obsolete by time like in the above example, or perhaps becomes infeasible due to practical difficulties. Then, the problem arises on how to subtract such piece of information without altering the behaviour of the relevant system components or violating the existing specifications. Moreover, in such a scenario, two logical notions introduced by E. Dijkstra in [Dijkstra, 1978] are very informative: the *strongest necessary condition* (SNC) and the *weakest sufficient condition* (WSC) of a given specification. These correspond to *most general consequence* and the *most specific abduction* of such specification, respectively.

To address such a scenario in a principled way, we employ a method based on formal verification.[1] In particular, we introduce a *forgetting*-based approach in Computation Tree Logic (CTL) [Clarke and Emerson, 1981] a central formalism in formal verification, and show that it can be used to compute SNC and WSC, in the same spirit of [Lin, 2001].

The scenario we mentioned concerning car-engine manufacturing can be easily represented as a small example by the following Kripke structure $\mathcal{M} = (S, R, L, s_0)$ in Figure 1 on $V = \{sl, sr, se, le, lc\}$ whose elements correspond to *select*, *safety restrictions*, *small engine*, *large engine* and *low-carbon emission requirements*, respectively. Moreover, s_0 is the initial state where we select either choose producing an engine for *sedan* which corresponds to the state s_1 or a *sports car* which corresponds to the state s_2. Since only a single-type of production is possible at a time, after each production state (s_1 or s_2), we turn back to the initial state (s_0) to start over.

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]. On the *forgetting* side, it was first

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

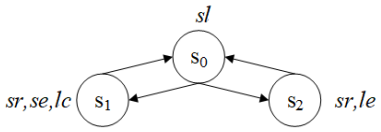

Figure 1: Car Engine Manufacturing Scenario

formally defined in propositional and first order logics by Lin and Reiter [Lin and Reiter, 1994]. Over the last decades, researchers have developed forgetting notions and theories not only in classical logic but also in non-classical logic systems [Eiter and Kern-Isberner, 2019], such as forgetting in logic programs under answer-set semantics [Zhang and Foo, 2006; Eiter and Wang, 2008; Wong, 2009; Wang *et al.*, 2012; Wang *et al.*, 2013], description logics [Wang *et al.*, 2010; Lutz and Wolter, 2011; Zhao and Schmidt, 2017] and knowledge forgetting in modal logic [Zhang and Zhou, 2009; Su *et al.*, 2009; Liu and Wen, 2011; Fang *et al.*, 2019]. It also has been considered in planning [Lin, 2003] and conflict solving [Lang and Marquis, 2010; Zhang *et al.*, 2005], creating restricted views of ontologies [Zhao and Schmidt, 2017], strongest and weakest definitions [Lang and Marquis, 2008], SNC (WSC) [Lin, 2001], among others.

Although forgetting has been extensively investigated from various aspects of different logical systems, the existing forgetting techniques are not directly applicable in CTL. For instance, in propositional forgetting theory, forgetting atom q from φ is equivalent to a formula $\varphi[q/\top] \vee \varphi[q/\ \bot]$, where $\varphi[q/X]$ is a formula obtained from φ by replacing each q with X ($X \in \{\top, \bot\}$). This method cannot be extended to a CTL formula. Consider a CTL formula $\psi = \text{AG}p \wedge \neg\text{AG}q \wedge \neg\text{AG}\neg q$. If we want to forget atom q from ψ by using the above method, we would have $\psi[q/\top] \vee \psi[q/\ \bot] \equiv \bot$. This is obviously not correct since after forgetting q this specification should not become inconsistent. Similar to [Zhang and Zhou, 2009], we research forgetting in CTL from the semantic forgetting point of view. And it is shown that our definition of forgetting satisfies those four postulates of forgetting presented in [Zhang and Zhou, 2009].

The rest of the paper is organised as follows. Section 2 introduces the notation and technical preliminaries. As key contributions, Section 3, introduces the notion of forgetting in CTL, via developing the notion of V-bisimulation. Such bisimulation is constructed through a set-based bisimulation and more general than the classical bisimulation. Moreover, it provides a CTL characterization for model structures (with the initial state), and studies the semantic properties of forgetting. In addition, a complexity analysis, including a relevant fragment CTL_{AF}, is carried out. Section 4 explores the relation between forgetting and SNC (WSC). Section 5 gives a model-based algorithm for computing forgetting in CTL and outline its complexity. Conclusion closes the paper.

Remark Due to space restrictions and to avoid hindering the flow of content, we put all the proofs to the supplementary material (proof appendix).

2 Preliminaries

We start with some technical and notational preliminaries. Throughout this paper, we fix a finite set $\mathcal{A}$ of propositional variables (or atoms), and use V, V' for subsets of $\mathcal{A}$.

2.1 Model structure in CTL

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

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

We call a model structure $\mathcal{M}$ on a set V of atoms if $L : S \to 2^V$, *i.e.* , the labelling function L map every state to V (not the $\mathcal{A}$). A *path* π_{s_i} start from s_i of $\mathcal{M}$ is an infinite sequence of states $\pi_{s_i} = (s_i, s_{i+1}s_{i+2}, \dots)$, where for each j ($0 \leq i \leq j$), $(s_j, s_{j+1}) \in R$. By $s' \in \pi_{s_i}$ we mean that s' is a state in the path π_{s_i}. A state $s \in S$ is *initial* if for any state $s' \in S$, there is a path π_s s.t $s' \in \pi_s$. We denote this model structure as (S, R, L, s_0), where s_0 is initial.

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

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

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

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

2.2 Syntax and semantics of CTL

In the following we briefly review the basic syntax and semantics of the CTL [Clarke *et al.*, 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: $\bot$ and $\top$;
- the classical connectives: $\vee$ and $\neg$;
- the path quantifiers: A and E;
- the temporal operators: X, F, G U and W, that means 'neXt state', 'some Future state', 'all future states (Globally)', 'Until' and 'Unless', respectively;
- parentheses: (and).

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

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

[2] Indeed, every state is identified by a configuration of atoms i.e., which holds in that state.

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).

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

- $(\mathcal{M}, s) \not\models \bot$ 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 $s_1 \in S$ and $(s, s_1) \in R$;
- $(\mathcal{M}, s) \models \text{EG}\phi$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \ldots)$ 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 $1 \leq j < i$.

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

3 Forgetting in CTL

In this section, we will define the forgetting in CTL by V-bisimulation constructed via a set-based bisimulation. Besides, some properties of forgetting are also explored. For convenience, let $\mathcal{M} = (S, R, L, s_0)$, $\mathcal{M}' = (S', R', L', s'_0)$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ with $\mathcal{M}_i = (S_i, R_i, L_i, s^i_0)$, $s_i \in S_i$ (is any state in S_i) and $i \in \mathbb{N}$.

3.1 Set-based bisimulation

To present a formal definition of forgetting, we need the concept of V-bisimulation. Inspired by the notion of bisimulation in [Browne *et al.*, 1988], we define the relations $\mathcal{B}_0, \mathcal{B}_1, \ldots$ between K-structures on V as follows: let $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ with $i \in \{1, 2\}$,

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

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

Now, we 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 $(\mathcal{K}_1, \mathcal{K}_2) \in \mathcal{B}_i$ for all $i \geq 0$. Moreover, two paths $\pi_i = (s_{i,1}, s_{i,2}, \ldots)$ of $\mathcal{M}_i$ with $i \in \{1, 2\}$ are* V-bisimilar *if $\mathcal{K}_{1,j} \leftrightarrow_V \mathcal{K}_{2,j}$ for every $j \geq 1$ where $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$.*

It is apparent that $\leftrightarrow_V$ is a binary relation. In the sequel, we abbreviate $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$ by $s_1 \leftrightarrow_V s_2$ whenever the underlying model structures of states s_1 and s_2 are clear from the context.

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

Besides, we have the following properties:

Proposition 1 *Let $i \in \{1, 2\}$, $V_1, V_2 \subseteq \mathcal{A}$, s'_is be two states and π'_is 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 $\pi'_1 \leftrightarrow_{V_1 \cup V_2} \pi'_2$;

(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 $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$.

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

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

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

- $L_1(s_1) - V = L_2(s_2) - V$,
- for every subtree $\text{Tr}_{n-1}(s'_1)$ of $\text{Tr}_n(s_1)$, $\text{Tr}_n(s_2)$ has a subtree $\text{Tr}_{n-1}(s'_2)$ such that $\text{Tr}_{n-1}(s'_1) \leftrightarrow_V \text{Tr}_{n-1}(s'_2)$, and vice versa.

Note that 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*

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

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

Proposition 3 *Let $V \subseteq \mathcal{A}$, $\mathcal{M}$ be a model 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.*

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

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

3.2 Characterization of Initial K-structure

In order to introduce our notion of forgetting, and to compute strongest necessary and weakest sufficient conditions, we need a formula that captures the initial K-structure on V syntactically. We call such formula as characterizing formula. In the following, we present such a characterization.

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

Definition 2 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be a model 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:*

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

$$\mathcal{F}_V(Tr_{k+1}(s)) = \bigwedge_{(s,s') \in R} \mathrm{EX}\mathcal{F}_V(Tr_k(s')) \wedge \mathrm{AX}\left(\bigvee_{(s,s') \in R} \mathcal{F}_V(Tr_k(s'))\right) \wedge \mathcal{F}_V(Tr_0(s))$$

for $k \geq 0$.

The characterizing formula of a computation tree formally exhibit the content of each node on V (i.e., atoms that are true at this node if they are in V, false otherwise) and the temporal relation between states recursively. The following result shows that the V-bisimulation between two computation trees imply the semantic equivalence of the corresponding characterizing formulas.

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

Let $s' = s$, it shows that for any formula φ of V, if φ is a characterizing formula of $\mathrm{Tr}_n(s)$ then $\varphi \equiv \mathcal{F}_V(\mathrm{Tr}_n(s))$.

Let $V \subseteq \mathcal{A}$, $\mathcal{K} = (\mathcal{M}, s_0)$ be an initial K-structure and $T(s') = \mathcal{F}_V(\mathrm{Tr}_c(s'))$. The *characterizing formula* of $\mathcal{K}$ on V, written $\mathcal{F}_V(\mathcal{M}, s_0)$ (or $\mathcal{F}_V(\mathcal{K})$ in short), is defined as the following formula:

$$\mathcal{F}_V(\mathrm{Tr}_c(s_0)) \wedge \bigwedge_{s \in S} \mathrm{AG}\left(\mathcal{F}_V(\mathrm{Tr}_c(s)) \rightarrow \bigwedge_{(s,s') \in R} \mathrm{EX}T(s') \wedge \mathrm{AX} \bigvee_{(s,s') \in R} T(s')\right)$$

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

The following example show how to compute characterizing formula:

Example 1 Let $V = \{sr\}$ and $\overline{V} = \{sl, se, lc, le\}$ ($\mathcal{A} = V \cup \{sr\}$). We have $\mathrm{Tr}_0(s_0) \not\leftrightarrow_{\overline{V}} \mathrm{Tr}_0(s_1)$ and $\mathrm{Tr}_0(s_0) \not\leftrightarrow_{\overline{V}} \mathrm{Tr}_0(s_2)$, then $\mathrm{dis}_{\overline{V}}(\mathcal{M}, s_0, s_1, 0)$ and $\mathrm{dis}_{\overline{V}}(\mathcal{M}, s_0, s_2, 0)$. Besides, it is easy checking that $s_1 \leftrightarrow_{\overline{V}} s_2$ since they have the same direct successor s_0. Hence, $ch(\mathcal{M}, \overline{V}) = 0$. Therefore, we have:

$\mathcal{F}_V(\mathrm{Tr}_0(s_0)) = \neg sr$

$\mathcal{F}_V(\mathrm{Tr}_0(s_1)) = \mathcal{F}_V(\mathrm{Tr}_0(s_2)) = sr$, and then

$\mathcal{F}_V(\mathcal{M}, s_0) = \neg sr \wedge \mathrm{AG}(\neg sr \rightarrow \mathrm{AX}sr) \wedge \mathrm{AG}(sr \rightarrow \mathrm{AX}\neg sr)$.

By the following theorem, we also have that given $V \subseteq \mathcal{A}$, the characterizing formula of an initial K-structure is unique (up to semantic equivalence) which describes this initial K-structure on V.

Theorem 2 *Given $V \subseteq \mathcal{A}$, let $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{M}' = (S', R', L', s'_0)$ be two model 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 $\mathcal{F}_V(\mathcal{M}, s_0) \equiv \mathcal{F}_V(\mathcal{M}', s'_0)$.

Moreover, we know that any initial K-structure can be described as a CTL formula from the definition of characterizing formula. Then,

Lemma 3 *Let φ be a formula. We have*

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

It follows that any CTL formula can rewritten in the form of a characterizing formula (in the form of a disjunction of characterizing formulas; each corresponds to a single model.

3.3 Semantic Properties of Forgetting in CTL

In this subsection we will give the definition of forgetting in CTL and study it's semantic properties. We will first show via a representation theorem that our definition of forgetting correspond to the readily existing notion of forgetting which is characterised by several desirable properties (also called postulates) suggested in [Zhang and Zhou, 2009]. Next, we discuss various additional semantic properties of forgetting.

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

Definition 3 (Forgetting) *Let $V \subseteq \mathcal{A}$ and ϕ a formula. A formula ψ with $Var(\psi) \cap V = \emptyset$ is a* result of forgetting V from ϕ, *if*

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

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

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

For this reason, the forgetting result is denoted by $\mathrm{F}_{\mathrm{CTL}}(\phi, V)$.

Assume you are given a formula φ, and φ' is the formula after forgetting V, then we have the following desired properties, also called *postulates* of forgetting [Zhang and Zhou, 2009].

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

Intuitive enough, the postulate (**W**) says, forgetting weakens the original formula. (**PP**) and (**NP**) correspond to the fact that so long as forgotten atoms V are irrelevant to the remaining positive and the negative information, respectively, they do not affect such information. (**IR**) states that forgotten atoms V are not relevant for the final formula (i.e., φ' is V-irrelevant).

Theorem 3 (Representation Theorem) *Let φ, φ' and ϕ be CTL formulas and $V \subseteq \mathcal{A}$. Then the following statements are equivalent:*

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

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

*(iii) Postulates (**W**), (**PP**), (**NP**) and (**IR**) hold.*

The above theorem says that, the notion of forgetting we defined, satisfies (and entailed by) the four postulates of forgetting. Moreover, CTL is closed under our definition of forgetting, i.e., for any CTL formula the result of forgetting is also a CTL formula.

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

Proposition 4 *Given a formula $\varphi \in$ CTL, V a set of atoms and p an atom such that $p \notin V$. Then,*

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

This means that the result of forgetting V from φ can be obtained by forgetting atoms in V one by one. Moreover, the order of atoms does not matter (commutativity), which follows from Proposition 4.

Corollary 4 (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 results, which are satisfied in both classical propositional logic and modal logic **S5** [Zhang and Zhou, 2009], further illustrate other essential semantic properties of forgetting.

Proposition 5 *Let φ, φ_i, ψ_i $(i = 1, 2)$ be formulas 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) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \wedge \psi_2, V) \models \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \wedge \mathrm{F}_{\mathrm{CTL}}(\psi_2, V)$;

Another interesting result is that the forgetting of $QT\varphi$ ($Q \in \{\mathrm{E}, \mathrm{A}\}$, $T \in \{\mathrm{F}, \mathrm{X}\}$) on $V \subseteq \mathcal{A}$ can be computed by $QT\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$. This gives us a convenient method to compute forgetting, since we can push the forgetting operator to a subformula without affecting the semantics, and rather compute the forgetting on this subformula.

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

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

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

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

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

3.4 Complexity Results

In the following, we outline the computational complexity of the various tasks regarding the forgetting in CTL and its popular fragment $\mathrm{CTL}_{\mathrm{AF}}$. It turns out that the model-checking on forgetting without any restriction is NP-complete.

Proposition 7 (Model Checking on Forgetting) *Let $(\mathcal{M}, s_0)$ be an initial* K*-structure, φ be a CTL formula and V a set of atoms. Deciding whether $(\mathcal{M}, s_0)$ is a model of* $\mathrm{F}_{CTL}(\varphi, V)$ *is NP-complete.*

The fragment of CTL, in which each formula contains only AF temporal connective correspond to specifications that are desired 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]. In the following, we report various complexity results concerning forgetting and the logical entailment in this fragment.

Theorem 5 (Entailment on Forgetting) *Let φ and ψ be two $\mathrm{CTL}_{\mathrm{AF}}$ formulas and V a set of atoms. Then, results:*

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

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

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

The following results follow from Theorem 5 and extends them to semantic equivalence.

Corollary 6 *Let φ and ψ be two $\mathrm{CTL}_{\mathrm{AF}}$ formulas and V a set of atoms. Then*

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

(ii) deciding $\mathrm{F}_{CTL}(\varphi, V) \equiv^{?} \varphi$ *is co-NP-complete,*

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

4 Strongest Necessary and Weakest Sufficient Conditions

In this section, we will give the definition of strongest necessary and weakest sufficient conditions (SNC and WSC, respectively) of a specification in CTL and show that they can be obtained by forgetting under a given initial K-structure and set V of atoms. We first start with defining these notions of an atomic proposition.

Definition 4 (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 $\phi \models q \rightarrow \psi$.*
- *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 a 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 *(SNC in short) of q on V under ϕ if it is a 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 the sense of equivalence the SNC (WSC) is unique (up to equivalence).

Proposition 8 *(**dual**) Let V, q, φ and ψ are like in Definition 4. The 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.*

This shows that the SNC and WSC are in fact dual conditions. Under this dual property, we can bother ourselves with only one of them e.g., SNC, and WSC can be obtained easily.

In order to generalise Definition 4 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 notion of SNC and WSC for an atomic proposition can be lifted to any formula, or conversely the SNC and WSC of any formula can be reduced to that of a proposition.

Proposition 9 *Let Γ and α be two formulas, $V \subseteq Var(\alpha) \cup Var(\phi)$ and q is 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 $\Gamma' = \Gamma \cup \{q \leftrightarrow \alpha\}$.*

The following result establishes the bridge between these two notions which are central to the paper; basically SNC (WSC) and the forgetting in which the former is obtained through the latter.

Theorem 7 *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 a 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.*

As aforementioned, since any initial K-structure can be characterized by a CTL formula, we can obtain the SNC (and its dual WSC) of a target property (a formula) under an initial K-structure by forgetting.

Theorem 8 *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 $\neg\mathrm{F}_{\mathrm{CTL}}(\mathcal{F}_{\mathcal{A}}(\mathcal{K}) \wedge \neg q, V')$.

5 An Algorithm to Compute Forgetting

To compute the forgetting in CTL, we propose a model-based method. Intuitively, the model-based method means that we can compute the forgetting applied to a formula simply by considering all the possible models of that formula.

Next, we give the model-based algorithm i.e., Algorithm 1 to compute the forgetting in CTL. The correctness of this algorithm follows from Lemma 3 and Theorem 2.

Example 2 Consider the model given in Figure 1. Assume that we are given a property $\alpha = \mathrm{AGEF}(lc \wedge sr)$ and $\{lc\}$. Then, it is easy to check that $\mathrm{F}_{\mathrm{CTL}}(\alpha, \{lc\}) \equiv \mathrm{AGEF}sr$.

Proposition 10 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$ with $|\mathcal{A}| = m$ and $|V| = n$. The time and space complexity of Algorithm 1 are $O(2^{m*2^m})$.*

Algorithm 1: A Model-based Forgetting Procedure

Input: A CTL formula φ and a set V of atoms
Output: $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$

```
T = ∅ // the set of models of φ ;
T' = ∅ // the set of possible initial K-structures ;
n = |A|;
for i = 1, ..., 2^n do
    for s_j ∈ {s_1, ..., s_i} do
        Let s_j be an initial state, construct
          M = (S, R, L, s_j) by the definition of model
          structure with S = {s_1, ..., s_i};
        for K ∈ T' do
            if (M, s_j) ↮_{Var(φ)} K then
                Let T' ← T' ∪ {(M, s_j)};
            end
        end
    end
    for (M, s_0) ∈ T' do
        if (M, s_0) ⊨ φ then
            T ← T ∪ {(M, s_0)};
        end
    end
end
return ⋁_{(M', s'_0) ∈ T} F_{\overline{V}}(M', s'_0).
```

6 Concluding Remarks

Summary In this article, we generalized the notion of bisimulation from model structures of Computation Tree Logic (CTL) to model structures with initial states over a given signature V, named *V-bisimulation*. Based on this new bisimulation, we presented the notion of forgetting for CTL, which enables computing weakest sufficient and strongest necessary conditions of specifications. Further properties and complexity issues in this context were also explored. 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 investigated the model checking of forgetting, which turned out to be NP-Complete, and the relevant fragment of $\mathrm{CTL}_{\mathrm{AF}}$ which ranged from co-NP to Π_2^p-completeness. 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., $\mathcal{M} \models \psi \supset \phi$ 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 ψ.

Moreover, a further study regarding the the computational complexity for other general fragments is required and part of the future research agenda. Such investigation can be coupled with fine-grained parameterized analysis.

References

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

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

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

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

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

[Dailler *et al.*, 2018] Sylvain Dailler, David Hauzar, Claude Marché, and Yannick Moy. Instrumenting a weakest precondition calculus for counterexample generation. *Journal of logical and algebraic methods in programming*, 99:97–113, 2018.

[Dijkstra, 1978] Edsger W Dijkstra. Guarded commands, nondeterminacy, and formal derivation of programs. In *Programming Methodology*, pages 166–175. Springer, 1978.

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

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

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

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

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

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

[Lin, 2001] Fangzhen Lin. On strongest necessary and weakest sufficient conditions. *Artif. Intell.*, 128(1-2):143–159, 2001.

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

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

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

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

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

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

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

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

[Woodcock and Morgan, 1990] Jim CP Woodcock and Carroll Morgan. Refinement of state-based concurrent systems. In *International Symposium of VDM Europe*, pages 340–351. Springer, 1990.

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

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

[Zhang *et al.*, 2005] Yan Zhang, Norman Y. Foo, and Kewen Wang. Solving logic program conflict through strong and weak forgettings. In *Ijcai-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, Uk, July 30-August*, pages 627–634, 2005.

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