On Sufficient and Necessary Conditions in transition system with Finite States: a preliminary report

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

Abstract

Model checking is an automatic, model-based, property-verification approach. It is intended to be used for concurrent, reactive systems and originated as a post-development methodology. Specification, which prescribes what the system has to do and what not, is used to product the properties that a system should satisfy. Computation Tree Logic (CTL) is one of the main logical formalisms for program specification and verification. In order to compute the SNC (WSC) of a given property (a CTL formula) under a given transition system, we study forgetting in CTL from the sematic forgetting point of view. We show that a CTL system is closed under our definition of forgetting, and this definition satisfies those four postulates of forgetting at first. And then we show the SNC and WSC can be computed by using the technology of forgetting. Besides, an algorithm model-based has been put forward to compute forgetting in CTL.

1 Introduction

Model checking is an automatic, model-based, property-verification approach. It is intended to be used for concurrent, reactive systems and originated as a post-development methodology. In model checking, specification is mainly used to specify the properties that should be satisfied by a system. CTL, one of the main logical formalisms for program specification and verification, has been proposed by Clarke and Emerson [Clarke and Emerson, 1981].

Weakest precondition, we also called weakest sufficient condition (WSC), is introduced by Dijkstra in [Dijkstra, 1978]. Strongest postcondition (we also called strongest necessary condition (SNC)), a dual concept, was introduced subsequently. The SNC is the most general consequent and the WSC is the most general precondition. In program correctness methods, SNC and WSC meet their toughest challenge when they deal with iterative constructs. It has been extensively studied in active system [Mraihi *et al.*, 2011; Bruggink *et al.*, 2011]. However, the method to compute the necessary and sufficient conditions under the give transition system T, set of atoms V and a formula (or property) φ, *i.e.* a formula ψ on V such that $T \models \psi \supset \varphi$ and $T \models \varphi \supset \psi$ respectively, is still lacking.

In the past decades, the SNC and WSC have been investigated in Classical Logic (CL) [Lin, 2001a] and it has many applications in knowledge representation and reasoning. In general, it is difficult to obtain such conditions for satisfying some properties under the given transition system. Inspired by the forgetting-based method to compute SNC (WSC) [Lin, 2001a], in this paper, we explore the forgetting of a branching-time temporal logic-CTL (Computation Tree temporal Logic) to compute SNC and WSC.

As a logical notion, forgetting was first formally defined in propostional and first order logics by Lin and Reiter [Lin and Reiter, 1994]. Over the last twenty years, researchers have developed forgetting notions and theories not only in classical logic but also in other non-classical logic systems, such as forgetting in logic programs under answer set/stable model semantics [Zhang and Foo, 2006; Eiter and Wang, 2008; Wong, 2009; Wang *et al.*, 2012; Wang *et al.*, 2013], forgetting in description logic [Wang *et al.*, 2010; Lutz and Wolter, 2011; Zhao and Schmidt, 2017] and knowledge forgetting in modal logic [Zhang and Zhou, 2009a; Su *et al.*, 2009; Liu and Wen, 2011; Fang *et al.*, 2019]. In application, forgetting has been used in planning [Lin, 2003], conflict solving [Lang and Marquis, 2010; Zhang *et al.*, 2005], createing restricted views of ontologies [Zhao and Schmidt, 2018], strongest and weakest definitions [Lang and Marquis, 2008], SNC (WSC) [Lin, 2001b] and so on. To our best knowledge, though forgetting has been extensively investigated from various aspects of different logical systems, in standard propositional logic, a general algorithm of forgetting and its computation-oriented investigation in CTL are still lacking.

However, the existing forgetting method in propositional logic, answer set programming, description logic and modal logic are not directly applicable in CTL. Similar with that in [Zhang and Zhou, 2009a], we study forgetting in CTL from the sematic forgetting point of view. And it is shown that our definition of forgetting satisfies those four postulates of forgetting.

The rest of the paper is organised as follows. Section 2 introduces the related notions for forgetting in CTL, including the syntax and semantics of CTL, the language we aimed for.

[*]Contact Author
[†]Contact Author

A formal definition of concept forgetting and its properties for CTL follows in Section 3. Section 4 explores the relation between forgetting and SNC (WSC). From the point of view of model, we propose an algorithm for computing forgetting on CTL in Section 5.

2 Preliminaries

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

2.1 Model structure in CTL

In general, a transition system [1] is described as a *model structure* (or *Kripke structure*), and a model structure is a triple $\mathcal{M} = (S, R, L)$, where

- S is a finite nonempty set of states,
- $R \subseteq S \times S$ 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 labeling function L map every state to V (not the $\mathcal{A}$). A *path* π_{s_i} start from s_i of $\mathcal{M}$ is a infinite sequence of states $\pi_{s_i} = (s_i, s_{i+1}s_{i+2}, \dots)$, where for each j $(0 \le i \le j)$, $(s_j, s_{j+1}) \in R$. By $s' \in \pi_{s_i}$ we mean that s' is a state in the path π_{s_i}. A sate $s \in S$ 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* $\mathrm{Tr}_n^{\mathcal{M}}(s)$ of $\mathcal{M}$(or simply $\mathrm{Tr}_n(s)$), that has depth n and is rooted at s, is recursively defined as [Browne *et al.*, 1988], for $n \ge 0$,

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

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

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

2.2 Syntactic and semantic of CTL

In the following we briefly review the basic syntax and semantics of the *Computation Tree Logic* (CTL in short) [Clarke *et al.*, 1986]. The *signature* of $\mathcal{L}$ includes:

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

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

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

where $p \in \mathcal{A}$. The formulas $\phi \wedge \psi$ and $\phi \to \psi$ are defined in a standard manner of propositional logic. The other form formulas of $\mathcal{L}$ are abbreviated using the forms of (1). The priorities for the CTL connectives are assumed to be (from the highest to the lowest):

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

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

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

Similar to the work in [Browne *et al.*, 1988; Bolotov, 1999], only initial K-structures are considered to be candidate models in the following, unless explicitly stated. Formally, an initial K-structure $\mathcal{K}$ is a *model* of a formula ϕ whenever $\mathcal{K} \models \phi$. Let Π be a set of formulae, $\mathcal{K} \models \Pi$ if for each $\phi \in \Pi$ there is $\mathcal{K} \models \phi$. We denote *Mod*(ϕ) (*Mod*(Π)) the set of models of ϕ (Π). The formula ϕ (set Π of formulae) is *satisfiable* if *Mod*$(\phi) \neq \emptyset$ (*Mod*$(\Pi) \neq \emptyset$). Since signatures (atoms) in model structure is finite, *Mod*(ϕ) (*Mod*(Π)) is finite for any formula ϕ (set Π of formulae).

Let ϕ_1 and ϕ_2 be two formulas or set of formulas. By $\phi_1 \models \phi_2$ we denote *Mod*$(\phi_1) \subseteq$ *Mod*(ϕ_2). By $\phi_1 \equiv \phi_2$ we mean $\phi_1 \models \phi_2$ and $\phi_2 \models \phi_1$. In this case ϕ_1 is *equivalent* to ϕ_2. By *Var*(ϕ_1) we mean the set of atoms occurring in ϕ_1. ϕ_1 is V-*irrelevant*, written $\mathrm{IR}(\phi_1, V)$, if there is a formula ψ with *Var*$(\psi) \cap V = \emptyset$ such that $\phi_1 \equiv \psi$.

3 Forgetting in CTL

In this section, we will define the forgetting in CTL by V-bisimulation, called the set-based bisimulations. 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 $\mathcal{M}_i = (S_i, R_i, L_i, s_0^i)$, $s_i \in S_i$ and i is an integer.

[1] According to [Baier and Katoen, 2008], a *transition system* TS is a tuple $(S, Act, \to, I, AP, L)$ where (1) S is a set of states, (2) Act is a set of actions, (3) $\to \subseteq S \times \mathrm{Act} \times S$ is a transition relation, (4) $I \subseteq S$ is a set of initial states, (5) AP is a set of atomic propositions, and (6) $L : S \to 2^{\mathrm{AP}}$ is a labeling function.

3.1 Set-based bisimulation

To present a formal definition of knowledge forgetting, we need the concepts of V-bisimulation, which is similar with that in [Zhang and Zhou, 2009b]. Inspired by the notion of bisimulation in [Browne *et al.*, 1988], we define the relations $\mathcal{B}_0, \mathcal{B}_1, \ldots$ between K-structures as follows: let $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ and $\mathcal{M}_i = \langle S_i, R_i, L_i, s_0^i \rangle$ with $i \in \{1, 2\}$,

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

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

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

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

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

In this case, $\mathcal{K}_1$ and $\mathcal{K}_2$ are called V-bisimular. Our V-bisimulation is a more general relation than others. On the one hand, the above set-based bisimulation is an extension of the bisimulation-equivalence of Definition 7.1 in [Baier and Katoen, 2008] in the sense that if $V = \mathcal{A}$ then our bisimulation is almost same to the latter. On the other hand, the above set-based bisimulation notion is similar to the state equivalence in [Browne *et al.*, 1988]. But it is different in the sense that ours is defined on K-structures, while it is defined on states in [Browne *et al.*, 1988]. What's more, the set-based bisimulation notion is also different from the state-based bisimulation notion of Definition 7.7 in [Baier and Katoen, 2008], which is defined for states of a given K-structure.

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

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

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

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

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

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

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

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

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

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

Besides, we have the following properties:

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

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

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

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

Proposition 3 *For any* K*-structure $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ a model structure on a finite set $\mathcal{A}$ of atoms and $V \subseteq \mathcal{A}$, it holds that $\mathcal{K} \leftrightarrow_{V'} \mathcal{K}_{|V}$ where $V' = \mathcal{A} - V$.*

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

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

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

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

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

Proposition 4 *Let $V \subseteq \mathcal{A}$ and $(\mathcal{M}_i, s_i)$ $(i = 1, 2)$ be two* K*-structures. Then*

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

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 number k such that $\text{Tr}_k(s_1)$ and $\text{Tr}_k(s_2)$ are not V-bisimular.

Proposition 5 *Let $V \subseteq \mathcal{A}$, $\mathcal{M}$ be a model structure and $s, s' \in S$ such that $(s, s') \notin \mathcal{B}$. There exists a least number k such that $Tr_k(s)$ and $Tr_k(s')$ are not V-bisimular.*

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

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

Now we give the formal definition of forgetting in CTL from the semantic forgetting 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*

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

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).

Intuitively, forgetting an atom results in a weaker theory which entails the same set of formulae that are irrelevant to the atom. To present the representation property of forgetting in CTL, we will give the Characterize formula of an initial K-structure on V in the next subsection.

3.2 Characterize formula of initial K-structure

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

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

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

$$\mathcal{F}_V(Tr_{k+1}(s)) = \bigwedge_{(s,s') \in R} \text{EX}T(s') \wedge \text{AX} \bigvee_{(s,s') \in R} T(s') \wedge \mathcal{F}_V(Tr_0(s))$$

for $k \geq 0$, where $T(s') = \mathcal{F}_V(Tr_k(s'))$.

The characterize formula of a computation tree formally exhibit the context of each node on V (atoms are true at this node if they are in V, else false) and the temporal relation between states recursively. In this way, we can know:

Lemma 2 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ and $\mathcal{M}' = (S', R', L', s'_0)$ be two model structures, $s \in S$, $s' \in S'$ and $n \geq 0$. If $Tr_n(s) \leftrightarrow_{\overline{V}} Tr_n(s')$, then $\mathcal{F}_V(Tr_n(s)) \equiv \mathcal{F}_V(Tr_n(s'))$.*

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

Let $V \subseteq \mathcal{A}$, $\mathcal{K} = (\mathcal{M}, s_0)$ be an initial K-structure and $T(s') = \mathcal{F}_V(\text{Tr}_c(s'))$. The *characterizing formula* of $\mathcal{K}$ on

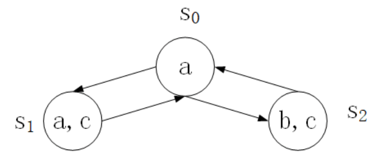

Figure 1: A simple Kripke structure

V, written $\mathcal{F}_V(\mathcal{M}, s_0)$ (or $\mathcal{F}_V(\mathcal{K})$), is defined as the conjunction of the following formulas:

$$\mathcal{F}_V(\text{Tr}_c(s_0)), \text{ and for each } s \in S$$

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

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

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

Example 1 Let $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ be a initial K-structure (in Fig. 1), in which $S = \{s_0, s_1, s_2\}$, $R = \{(s_0, s_1), (s_0, s_2), (s_1, s_0), (s_2, s_0)\}$, $L(s_0) = \{a\}$, $L(s_1) = \{a, c\}$ and $L(s_2) = \{b, c\}$. Let $V = \{a, b\}$, compute the characterizing formula of $\mathcal{K}$ on V.

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

$$\begin{aligned}
&\mathcal{F}_V(\text{Tr}_0(s_0)) = a \wedge \neg b,\\
&\mathcal{F}_V(\text{Tr}_0(s_1)) = a \wedge \neg b,\\
&\mathcal{F}_V(\text{Tr}_0(s_2)) = b \wedge \neg a,\\
&\mathcal{F}_V(\text{Tr}_1(s_0)) = \text{EX}(a \wedge \neg b) \wedge \text{EX}(b \wedge \neg a) \wedge \text{AX}((a \wedge \neg b) \vee\\
&\qquad (b \wedge \neg a)) \wedge (a \wedge \neg b),\\
&\mathcal{F}_V(\text{Tr}_1(s_1)) = \text{EX}(a \wedge \neg b) \wedge \text{AX}(a \wedge \neg b) \wedge (a \wedge \neg b),\\
&\mathcal{F}_V(\text{Tr}_1(s_2)) = \text{EX}(a \wedge \neg b) \wedge \text{AX}(a \wedge \neg b) \wedge (b \wedge \neg a).
\end{aligned}$$

Then it is easy to obtain $\mathcal{F}_V(\mathcal{M}, s_0)$.

Lemma 3 *Let φ be a formula. We have*

$$\varphi \equiv \bigvee_{(\mathcal{M}, s_0) \in Mod(\varphi)} \mathcal{F}_{\mathcal{A}}(\mathcal{M}, s_0). \quad (3)$$

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

Theorem 2 *Let $V \subseteq \mathcal{A}$, $\mathcal{M} = (S, R, L, s_0)$ be a model structure with initial state s_0 and $\mathcal{M}' = (S', R', L', s'_0)$ be a model structure with initial state s'_0. Then*

$$(\mathcal{M}', s'_0) \models \mathcal{F}_V(\mathcal{M}, s_0) \textit{ if and only if } (\mathcal{M}, s_0) \leftrightarrow_{\overline{V}} (\mathcal{M}', s'_0).$$

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

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

Proof: This is following Lemma 2 and the definition of the characterizing formula of initial K-structure $\mathcal{K}$ on V. ■

3.3 Semantic properties of forgetting in CTL

In this subsection we study essential semantic properties of forgetting. We will first show that our forgetting satisfy these postulates [Zhang and Zhou, 2009b] that precisely characterize the semantics of forgetting. We then discuss other semantic properties of forgetting.

By Lemma 3 and Definition 3, the result ψ of ϕ forget the set V of atoms always exists, which is equivalent to

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

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

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

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

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

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

(ii) $\psi \equiv \{\phi | \varphi \models \phi \& IR(\phi, V)\}$,

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

We can see from this theorem that our definition of forgetting under CTL is closed, *i.e.* for any CTL formula the result of forgetting is also a CTL formula, and captures the four postulates that forgetting should satisfy.

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

Proposition 6 *Let φ be a formula, 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 atom in V one by one. Similarly, a consequence of the previous proposition is:

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

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

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

Proposition 7 *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) $\mathrm{F}_{\mathrm{CTL}}(\psi_1 \wedge \psi_2, V) \models \mathrm{F}_{\mathrm{CTL}}(\psi_1, V) \wedge \mathrm{F}_{\mathrm{CTL}}(\psi_2, V)$;

(vi) If $IR(\psi_1, V)$, then $\mathrm{F}_{\mathrm{CTL}}(\varphi \wedge \psi_1, V) \equiv \mathrm{F}_{\mathrm{CTL}}(\varphi, V) \wedge \psi_1$.

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

Proposition 8 *Let $V \subseteq \mathcal{A}$ and ϕ a formula.*

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

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

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

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

4 SNC and WSC

In this section, we will give the definition of SNC (WSC) and show that the SNC (WSC) of a specification (a CTL formula) under a given initial K-structure and set V of atoms can be obtained from forgetting in CTL. The SNC (WSC) of a proposition will be given at first:

Definition 5 (sufficient and necessary condition) *Let ϕ be a formulas or an initial K-structure, ψ be formulas, $V \subseteq Var(\phi)$, $q \in Var(\phi) - V$ 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 9 *(**dual**) Let V, q, φ and ψ are the ones in Definition 5.*

(i) ψ is a SNC of q on V under φ iff $\neg\psi$ is a WSC of q on V under φ.

(ii) ψ is a WSC of q on V under φ iff $\neg\psi$ is a SNC of q on V under φ.

This show that the SNC and WSC are in fact dual conditions. Under the dual property, we can consider the SNC party only in sometimes, while the WSC part can be talked similarly.

For the case of formula, we have that the SCN (WSC) of any formula can be defined as follows:

Definition 6 *Let Γ be a formula or an initial* K*-structure, α be a formula and $P \subseteq (Var(\Gamma) \cup Var(\alpha))$. A formula φ of P is said to be a NC (SC) of α on P under Γ iff $\Gamma \models \alpha \to \varphi$. It is said to be a SNC (WSC) if it is a NC (SC), and for any other NC (SC) φ', we have that $\Gamma \models \varphi \to \varphi'$ ($\Gamma \models \varphi' \to \varphi$).*

It is seems that the SNC (WSC) of any formula can be obtained by changing to that of a proposition. Formally:

Proposition 10 *Let Γ be a formula, P, and α be as in Definition 6. A formula φ of P is the SNC (WSC) of α on P under Γ iff it is the SNC (WSC) of q on P under $\Gamma' = \Gamma \cup \{q \equiv \alpha\}$, where q is a new proposition not in Γ and α.*

We propose the theorem of computing the SNC (WSC) of an atom due to the SNC (WSC) of a formula can be change to the SNC (WSC) of an atom by Proposition 10.

Theorem 6 *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 we have said before that any initial K-structure can be characterized by a CTL formula, we can obtain the SNC (WSC) of an initial K-structure for satisfy some needed property (formula) by forgetting.

Theorem 7 *Let $\mathcal{K} = (\mathcal{M}, s)$ be a initial* K*-structure with $\mathcal{M} = (S, R, L)$ on the finite set $\mathcal{A}$ of atoms, $V \subseteq \mathcal{A}$ and $q \in V'$ ($V' = \mathcal{A} - V$). Then:*

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

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

5 Algorithm to compute forgetting

To compute the forgetting in CTL, we propose a model-based method in this part. Literally speaking, the model-based method means that we can obtain the result of forgetting in CTL by obtain all the possible finite models of this result.

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

Though the set of models of the result of forgetting is finite, while how many models is there? As it is said in Theorem 3 that if two initial K-structures are V-bisimulation, then their characteristic formulas is equal. Then let φ be a CTL formula, the $|Var(\varphi)| = m$ is a positive integer, we have the following theorem:

Algorithm 1: Model-based: Computing forgetting

Input: A CTL formula φ and a set V of atoms
Output: $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$
$T = \emptyset$ // the set of models of φ ;
$T' = \emptyset$ // the set of models of $\mathrm{F}_{\mathrm{CTL}}(\varphi, V)$;
$m = |Var(\varphi)|$;
$n = |V|$;
for *i=1, ...,* 2^m **do**
  Enumerating all possible initial K-structures $(\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ and $|S| = i$;
  For all initial K-structures $(\mathcal{M}, s_0)$ **if** $(\mathcal{M}, s_0) \models \varphi$ **then**
    $T = T \cup \{(\mathcal{M}, s_0)\}$;
  end
end
return $\bigvee_{(\mathcal{M}', s_0') \in T} \mathcal{F}_{\overline{V}}(\mathcal{M}', s_0')$.

Theorem 8 *Let φ be a CTL formula, $V = Var(\varphi)$, $|V| = m$, and $\mathcal{K} = (\mathcal{M}, s_0)$ with $\mathcal{M} = (S, R, L, s_0)$ be a initial* K*-structure. If $(\mathcal{M}, s_0) \models \varphi$, then there is an initial* K*-structure $\mathcal{K}' = (\mathcal{M}', s_0')$ with $\mathcal{M}' = (S', R', L', s_0')$ that satisfy:*

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

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

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

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

Proof: If $|S| \leq 2^m$, this result clearly holds. If $|S| > 2^m$, let $\mathcal{K}' = \mathcal{K}_{|V}$, then it is apparent that $\mathcal{K} \leftrightarrow_{\mathcal{A}\setminus V} \mathcal{K}'$ by Proposition 3 and $(\mathcal{M}', s_0') \models \varphi$. In the worst case, the number of states in S' is 2^m by the definition of $\mathcal{K}_{|V}$. Then the theorem is proved. ■

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

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

Proposition 11 *Let φ be a CTL formula and $V \subseteq \mathcal{A}$. The time complexity of Algorithm 1 is $O(2^{2^m})$ and the space complexity is $O(2^{2m})$. Where $|Var(\varphi)| = m$ and $|V| = n$.*

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.

[Bruggink *et al.*, 2011] HJ Bruggink, Raphaël Cauderlier, Mathias Hülsbusch, and Barbara König. Conditional reactive systems. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2011)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2011.

[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.

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

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

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

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

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

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

[Mraihi *et al.*, 2011] Olfa Mraihi, Wided Ghardallou, Asma Louhichi, Lamia Labed Jilani, Khaled Bsaies, and Ali Mili. Computing preconditions and postconditions of while loops. In *International Colloquium on Theoretical Aspects of Computing*, pages 173–193. Springer, 2011.

[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.

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

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

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

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

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

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