

# A Resolution Calculus for Forgetting in CTL

## Abstract

Computation Tree Logic (CTL) is a popular logical formalism in computer science with a wide range of applications; it is used in formal verification in the context of representing and reasoning about high-level system information (or *specification*), but also in other domains e.g., planning. Orthogonal to this, forgetting is the field of study which concerns removing information that is deemed irrelevant or obsolete from a knowledge base while guaranteeing certain properties.

In this paper, we present a resolution-based approach to perform *forgetting* in CTL. More specifically, we develop a calculus which extends earlier work (CTL with *index* i.e.,  $\text{SNF}_{\text{CTL}}^q$ ) with additional rules i.e., EF-implication which connects *next state* and *future state*. As tailoring a resolution calculus for forgetting is a non-trivial task, our technical contribution is manifold: (i) we provide a bisimulation between CTL formulae and  $\text{SNF}_{\text{CTL}}^q$  clauses; (ii) we introduce techniques for eliminating undesired atoms resulting from such transformation. Moreover, we show the soundness of our approach and analyse its computational complexity.

## 1 Introduction

Computation Tree Logic (CTL) (Clarke and Emerson 1981) is one of the central logical formalisms in computer science with a wide range of applications; it is used mostly in formal verification in the context of representing and reasoning about high-level system information (or *specification*), but also in other domains e.g., planning (Giunchiglia and Traverso 1999; Dal Lago, Pistore, and Traverso 2002; Akintunde 2017). Both in planning and verification updates are required where, e.g., some of the elements previously considered are no longer required. In these cases it is desirable to update the specifications such that they contain only the relevant vocabulary without the need to fully reproduce them. Moreover, with increasing size of the systems or planning domains the formalization becomes prohibitively large in size and complexity. As a consequence specifications become overly difficult to maintain, modify and re-use for later processing overly costly, even if only a specific part of a specification is of interest. Therefore, techniques and automated tools for obtaining sub-specification based on restricted vocabularies are required.

*Forgetting*, the task of distilling a reduced knowledge base that is relevant to a subset of the signature, addresses

these issues. As a logical notion, *forgetting* was first formally defined in propositional and first order-logics by Lin and Reiter (Lin and Reiter 1994). Over the last twenty years, not only have researchers developed forgetting notions and theories in propositional and first-order logic, but also in other logic systems (Eiter and Kern-Isberner 2019), such as in logic programs under answer set/stable model semantics (Zhang and Foo 2006; Eiter and Wang 2008; Wong 2009; Wang et al. 2012; Wang, Wang, and Zhang 2013), forgetting in description logic (Wang et al. 2010; Lutz and Wolter 2011; Zhao and Schmidt 2017a) and in modal logic (Zhang and Zhou 2009; Su et al. 2009; Liu and Wen 2011; Fang, Liu, and Van Ditmarsch 2019). Forgetting has been used in several application domains, such as planning (Lin 2003), conflict solving (Lang and Marquis 2010; Zhang, Foo, and Wang 2005), creating restricted views of ontologies (Zhao and Schmidt 2017a), strongest and weakest definitions (Lang and Marquis 2008), SNC (WSC) (Lin 2001) and others.

Although forgetting has been extensively investigated from various aspects of different logical systems, it is not directly applicable to CTL knowledge bases. There are a number of challenges. For instance, in propositional forgetting theory, forgetting atom  $q$  from  $\varphi$  is equivalent to a formula  $\varphi[q/\top] \vee \varphi[q/\perp]$ , where  $\varphi[q/X]$  is a formula obtained from  $\varphi$  by replacing each  $q$  with  $X$  ( $X \in \{\top, \perp\}$ ). This method cannot be extended to CTL. 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  $\psi$  by using the above method, we would have  $\psi[q/\top] \vee \psi[q/\perp] \equiv \perp$ . This is obviously not correct since after forgetting  $q$  this specification should not become inconsistent.

Another problem is that resolution based methods for propositional logic (Lin and Reiter 1994; Wang 2015) and Ackermann-based approach (second-order elimination) in description logic (Zhao and Schmidt 2017b) require a specific normal form which does not exist in this form in CTL. While any CTL formula can be transformed into a set of  $\text{SNF}_{\text{CTL}}^q$  clauses, which is a variant of CTL for which such a normal form does exist, it introduces *indices* and extra atoms. Both these two problems will have to be addressed.

In this paper, we give the definition of forgetting in CTL from the semantic forgetting point of view and explore a resolution-based method to compute it. In particular, we ex-

tend the Resolution Calculus in (Zhang, Hustadt, and Dixon 2014) by eliminating the atoms introduced in the transformation process by using a *binary bisimulation relation* (one on the set of atoms, one on indices). Such a bisimulation relation is an extension of the set-based bisimulation introduced in (Feng et al. 2020) by taking *index* into account.

In Section 2 we introduce the notation and technical preliminaries. In Section 3 we give a more precise definition of the forgetting problem. As key contributions, Section 4, introduces the resolution-based approach, as well as the proofs of soundness and complexity. We conclude the paper with some related and future work, as well as a brief discussion. Due to space restrictions and to avoid hindering the flow of content, some of the proves are moved to the supplementary material <sup>1</sup>.

## 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,
- $R \subseteq S \times S$  and, for each  $s \in S$ , there is  $s' \in S$  such that  $(s, s') \in R$ ,
- $L : S \rightarrow 2^{\mathcal{A}}$  is a labeling function.

Given a model structure  $\mathcal{M} = (S, R, L)$ , a *path*  $\pi_{s_i}$  starting 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  $\pi_{s_i}$ . A state  $s \in S$  is *initial* if for any state  $s' \in S$ , there is a path  $\pi_s$  s.t.  $s' \in \pi_s$ . If  $s_0$  is an initial state of  $\mathcal{M}$ , then we denote this model structure  $\mathcal{M}$  as  $(S, R, L, s_0)$ .

For a given model 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  $n \geq 0$  and is rooted at  $s$ , is recursively defined (Browne, Clarke, and Grumberg 1988) as follows:

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

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 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, 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:  $X, F, G, U$  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:

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

where  $p \in \mathcal{A}$ . The formulas  $\phi \wedge \psi$  and  $\phi \supset \psi$  are defined in the usual way. Other formulas in  $\mathcal{L}$  are abbreviated using the forms in (1).

Next, we define the semantics. Let  $\mathcal{M} = (S, R, L, s_0)$  be a model structure,  $s \in S$  and  $\phi \in \mathcal{L}$ . The *satisfiability* relationship between  $(\mathcal{M}, s)$  and  $\phi$ , written  $(\mathcal{M}, s) \models \phi$ , is inductively defined on the structure of  $\phi$  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  $s_1 \in S$  and  $(s, s_1) \in R$ ;
- $(\mathcal{M}, s) \models \text{EG}\phi$  iff  $\mathcal{M}$  has a path  $(s_1 = s, s_2, \dots)$  such that  $(\mathcal{M}, s_i) \models \phi$  for each  $i \geq 1$ ;
- $(\mathcal{M}, s) \models E[\phi_1 \text{ U } \phi_2]$  iff  $\mathcal{M}$  has a path  $(s_1 = s, s_2, \dots)$  such that, for some  $i \geq 1$ ,  $(\mathcal{M}, s_i) \models \phi_2$  and  $(\mathcal{M}, s_j) \models \phi_1$  for each  $1 \leq j < i$ .

Similar to the work in (Browne, Clarke, and Grumberg 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 (or set of formulas)  $\phi$  whenever  $\mathcal{K} \models \phi$  (or  $\mathcal{K} \models \psi$  for each  $\psi \in \varphi$ ). We denote  $\text{Mod}(\phi)$  the set of models of  $\phi$ .  $\phi$  is *satisfiable* if  $\text{Mod}(\phi) \neq \emptyset$ . Given two formulas (or sets of formulas)  $\phi_1$  and  $\phi_2$ ,  $\phi_1 \models \phi_2$  we mean  $\text{Mod}(\phi_1) \subseteq \text{Mod}(\phi_2)$ . And by  $\phi_1 \equiv \phi_2$ , we mean  $\phi_1 \models \phi_2$  and  $\phi_2 \models \phi_1$ . In this case,  $\phi_1$  is *equivalent* to  $\phi_2$ . The set of atoms occurring in  $\phi_1$  is denoted by  $\text{Var}(\phi_1)$ . The formula  $\phi_1$  is *irrelevant* to the atoms in a set  $V$  (or simply *V-irrelevant*), written  $\text{IR}(\phi_1, V)$ , if there is a formula  $\psi$  with  $\text{Var}(\psi) \cap V = \emptyset$  such that  $\phi_1 \equiv \psi$ . The *V-irrelevant* of a set of formulas can be defined similarly.

### 2.3 The Normal Form of CTL

It is known that any CTL formula  $\varphi$  can be transformed into a set  $T_\varphi$  of  $\text{SNF}_{\text{CTL}}^g$  (Separated Normal Form with Global Clauses for CTL) clauses in polynomial time such that  $\varphi$  is satisfiable iff  $T_\varphi$  is satisfiable (Zhang, Hustadt, and Dixon 2008). An important difference between CTL formulae and  $\text{SNF}_{\text{CTL}}^g$  is that  $\text{SNF}_{\text{CTL}}^g$  is an extension of the syntax of CTL to use indices. These indices can be used to preserve a particular path context. The language of  $\text{SNF}_{\text{CTL}}^g$  clauses is

<sup>1</sup><https://github.com/fengrenyan/Resolution-proof-CTL.git>

defined over an extension of CTL. That is, the language is based on: (1) the language of CTL; (2) a propositional constant **start**; (3) a countably infinite index set  $\text{Ind}$ ; and (4) temporal operators:  $E_{\langle \text{ind} \rangle}X$ ,  $E_{\langle \text{ind} \rangle}F$ ,  $E_{\langle \text{ind} \rangle}G$ , and  $E_{\langle \text{ind} \rangle}U$ .

We introduce the  $\text{SNF}_{\text{CTL}}^g$  clauses first, then we talk about its semantics. A  $\text{SNF}_{\text{CTL}}^g$  clause consists of formulae of the following forms.

$$\begin{aligned} \text{AG}(\mathbf{start} \supset \bigvee_{j=1}^k m_j) & \quad (\text{initial clause}) \\ \text{AG}(\mathbf{true} \supset \bigvee_{j=1}^k m_j) & \quad (\text{global clause}) \\ \text{AG}(\bigwedge_{i=1}^n l_i \supset \text{AX} \bigvee_{j=1}^k m_j) & \quad (\text{A - step clause}) \\ \text{AG}(\bigwedge_{i=1}^n l_i \supset E_{\langle \text{ind} \rangle}X \bigvee_{j=1}^k m_j) & \quad (\text{E - step clause}) \\ \text{AG}(\bigwedge_{i=1}^n l_i \supset \text{AF}l) & \quad (\text{A - sometime clause}) \\ \text{AG}(\bigwedge_{i=1}^n l_i \supset E_{\langle \text{ind} \rangle}Fl) & \quad (\text{E - sometime clause}). \end{aligned}$$

where  $k \geq 0$ ,  $n > 0$ , **start** is a propositional constant,  $l_i$  ( $1 \leq i \leq n$ ),  $m_j$  ( $1 \leq j \leq k$ ) and  $l$  are literals, that is, atomic propositions or their negation, and  $\text{ind} \in \text{Ind}$ . By a clause, we mean the classical clause or the  $\text{SNF}_{\text{CTL}}^g$  clause unless explicitly stated. As all clauses are of the form  $\text{AG}(P \supset D)$ , we often simply write  $P \supset D$  instead.

Formulae of  $\text{SNF}_{\text{CTL}}^g$  over  $\mathcal{A}$  are interpreted in Ind-model structure  $\mathcal{M} = (S, R, L, [\cdot], s_0)$ , where  $S, R, L$  and  $s_0$  is the same as our model structure talked above and  $[\cdot] : \text{Ind} \rightarrow 2^{(S \times S)}$  maps every index  $\text{ind} \in \text{Ind}$  to a successor function  $[\text{ind}]$  which is a functional relation on  $S$  and a subset of the binary accessibility relation  $R$ . That is, for every  $s \in S$  there exists exactly a state  $s' \in S$  such that  $(s, s') \in [\text{ind}]$  and  $(s, s') \in R$ . An infinite path  $\pi_{s_i}^{\langle \text{ind} \rangle}$  is an infinite sequence of states  $s_i, s_{i+1}, s_{i+2}, \dots$  such that for every  $j \geq i$ ,  $(s_j, s_{j+1}) \in [\text{ind}]$ .

Similarly, an *Ind-structure* (or *Ind-interpretation*) is an Ind-model structure  $\mathcal{M} = (S, R, L, [\cdot], s_0)$  associating with a state  $s \in S$ , which is simplified as  $(\mathcal{M}, s)$  for convenience in the following. In the case that  $s$  is an initial state of  $\mathcal{M}$ , the Ind-structure is *initial*.

The semantics of  $\text{SNF}_{\text{CTL}}^g$  is then defined as an extension of the semantics of CTL. Let  $\mathcal{M} = (S, R, L, [\cdot], s_0)$  be an Ind-model structure,  $s_i \in S$  and  $\psi$  a  $\text{SNF}_{\text{CTL}}^g$  formulae. The *satisfiability* relationship between  $(\mathcal{M}, s_i)$  and  $\psi$ , written  $(\mathcal{M}, s_i) \models \psi$ , is inductively defined on the structure of  $\psi$  as follows:

- $(\mathcal{M}, s_i) \models \mathbf{start}$  iff  $s_i = s_0$ ;
- $(\mathcal{M}, s_i) \models E_{\langle \text{ind} \rangle}X\psi$  iff for the path  $\pi_{s_i}^{\langle \text{ind} \rangle}$ ,  $(\mathcal{M}, s_{i+1}) \models \psi$ ;

- $(\mathcal{M}, s_i) \models E_{\langle \text{ind} \rangle}G\psi$  iff for every  $s_j \in \pi_{s_i}^{\langle \text{ind} \rangle}$ ,  $(\mathcal{M}, s_j) \models \psi$ ;
- $(\mathcal{M}, s_i) \models E_{\langle \text{ind} \rangle}[\psi_1 U \psi_2]$  iff there exists  $s_j \in \pi_{s_i}^{\langle \text{ind} \rangle}$  such that  $(\mathcal{M}, s_j) \models \psi_2$  and for every  $s_k \in \pi_{s_i}^{\langle \text{ind} \rangle}$ , if  $i \leq k < j$ , then  $(\mathcal{M}, s_k) \models \psi_1$ ;
- $(\mathcal{M}, s_i) \models E_{\langle \text{ind} \rangle}F\psi$  iff  $(\mathcal{M}, s_i) \models E_{\langle \text{ind} \rangle}[\top U \psi]$ .

The semantics of the remaining operators are analogous to that of CTL given previously. A  $\text{SNF}_{\text{CTL}}^g$  formula  $\varphi$  is satisfiable, iff for some Ind-model structure  $\mathcal{M} = (S, R, L, [\cdot], s_0)$ ,  $(\mathcal{M}, s_0) \models \varphi$ , and unsatisfiable otherwise. And if  $(\mathcal{M}, s_0) \models \varphi$  then  $(\mathcal{M}, s_0)$  is called an Ind-model of  $\varphi$ , and we say that  $(\mathcal{M}, s_0)$  satisfies  $\varphi$ . By  $T \wedge \varphi$  we mean  $\bigwedge_{\psi \in T} \psi \wedge \varphi$ , where  $T$  is a finite set of formulae. Other terminologies are similar with those in Section 2.2.

### 3 Problem Definition

The concept of semantic forgetting in CTL was introduced formally in (Feng et al. 2020). We recall the main results that are required for this paper. To present a formal definition of knowledge forgetting, we need the concepts of V-bisimulation on a given signature. It is (somehow) an extension of the classical bisimulation for CTL in (Baier and Katoen 2008).

For convenience, in the following we denote  $\mathcal{M} = (S, R, L, s_0)$ ,  $\mathcal{M}' = (S', R', L', s'_0)$ ,  $\mathcal{M}_i = (S_i, R_i, L_i, s_i^i)$  (or  $\mathcal{M} = (S, R, L, [\cdot], s_0)$ ,  $\mathcal{M}' = (S', R', L', [\cdot], s'_0)$ ,  $\mathcal{M}_i = (S_i, R_i, L_i, [\cdot], s_i^i)$ ) and  $\mathcal{K}_i = (\mathcal{M}_i, s_i)$  with  $s_i \in S_i$  and  $i \in \mathbb{N}$ .

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  $V \subseteq \mathcal{A}$ . Two K-structures (or Ind-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  $\pi_i = (s_{i,1}, s_{i,2}, \dots)$  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  $\mathcal{K}_{i,j} = (\mathcal{M}_i, s_{i,j})$ .*

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 (Ind-model structures) of states  $s_1$  and  $s_2$  are clear from the context.

**Lemma 1.** (Feng et al. 2020) *The relation  $\leftrightarrow_V$  is an equivalence relation.*

The following proposition shows that if a K-structure (or an Ind-structure) is  $V_1$  and  $V_2$ -bisimilar with the other two K-structures (or Ind-structures) respectively, then those

two  $\mathcal{K}$ -structures (or Ind-structures) are  $V_1 \cup V_2$ -bisimilar. This is important for forgetting since this laid the foundation of resolving atoms in  $V$  one by one in the resolution process later. Moreover, the  $V_1$ -bisimulation between two  $\mathcal{K}$ -structures (Ind-structures) implies that these two  $\mathcal{K}$ -structure (Ind-structures) are  $V_2$ -bisimilar for each  $V_2$  with  $V_1 \subseteq V_2 \subseteq \mathcal{A}$ . Formally,

**Proposition 1.** (Feng et al. 2020) Let  $i \in \{1, 2\}$ ,  $V_1, V_2 \subseteq \mathcal{A}$  and  $\mathcal{K}_i = (\mathcal{M}_i, s_i)$  ( $i = 1, 2, 3$ ) be  $\mathcal{K}$ -structures (Ind-structures) such that  $\mathcal{K}_1 \leftrightarrow_{V_1} \mathcal{K}_2$  and  $\mathcal{K}_2 \leftrightarrow_{V_2} \mathcal{K}_3$ . Then:

- (i)  $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$ ;
- (ii) If  $V_1 \subseteq V_2$  then  $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$ .

Intuitively, if two  $\mathcal{K}$ -structures are  $V$ -bisimilar, then they satisfy the same formula  $\varphi$  that does not contain any atoms in  $V$ , i.e.  $\text{IR}(\varphi, V)$ .

**Theorem 1.** (Feng et al. 2020) Let  $V \subseteq \mathcal{A}$ ,  $\mathcal{K}_i$  ( $i = 1, 2$ ) be two  $\mathcal{K}$ -structures such that  $\mathcal{K}_1 \leftrightarrow_V \mathcal{K}_2$  and  $\phi$  a formula with  $\text{IR}(\phi, V)$ . Then  $\mathcal{K}_1 \models \phi$  if and only if  $\mathcal{K}_2 \models \phi$ .

Based on our notion of  $V$ -bisimulation, we can now introduce forgetting in CTL from the semantic forgetting point view. This means that the result of forgetting the atoms in the set  $V$  of atoms from CTL formula  $\varphi$  is a formula which shares the same models as  $\varphi$  and models that are  $V$ -bisimilar to one of the models of  $\varphi$ .

**Definition 2** (Forgetting). (Feng et al. 2020) Let  $V \subseteq \mathcal{A}$  and  $\phi$  a CTL formula. A CTL formula  $\psi$  with  $\text{Var}(\psi) \cap V = \emptyset$  is a result of forgetting  $V$  from  $\phi$ , if

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

where  $\mathcal{K}$  and  $\mathcal{K}'$  are  $\mathcal{K}$ -structures.

Note that if both  $\psi$  and  $\psi'$  are results of forgetting  $V$  from  $\phi$  then  $\text{Mod}(\psi) = \text{Mod}(\psi')$ , i.e.  $\psi$  and  $\psi'$  have the same models. In other words, the forgetting result is unique (up to equivalence).

In order to bridge the gap between CTL and  $\text{SNF}_{\text{CTL}}^g$ , we define the  $\langle V, I \rangle$ -bisimulation between Ind-structures as follows:

**Definition 3** (Binary bisimulation relation). Let  $\mathcal{M}_i = (S_i, R_i, L_i, [-]_i, s_i^0)$  with  $i \in \{1, 2\}$  be two Ind-structures,  $V$  be a set of atoms and  $I \subseteq \text{Ind}$ . The  $\langle V, I \rangle$ -bisimulation  $\beta_{\langle V, I \rangle}$  between initial Ind-structures is a set that satisfy  $((\mathcal{M}_1, s_1^0), (\mathcal{M}_2, s_2^0)) \in \beta_{\langle V, I \rangle}$  if and only if  $(\mathcal{M}_1, s_1^0) \leftrightarrow_V (\mathcal{M}_2, s_2^0)$  and  $\forall j \notin I$  there is

- (i) for each  $(s, s_1) \in [j]_1$  there exists  $(s', s'_1) \in [j]_2$  such that  $s \leftrightarrow_V s'$  and  $s_1 \leftrightarrow_V s'_1$ , and
- (ii) for each  $(s', s'_1) \in [j]_2$  there exists  $(s, s_1) \in [j]_1$  such that  $s \leftrightarrow_V s'$  and  $s_1 \leftrightarrow_V s'_1$ .

We call this relation as *binary bisimulation relation*, also denoted as  $\leftrightarrow_{\langle V, I \rangle}$ . This definition is similar to our concept of  $V$ -bisimulation except that  $\langle V, I \rangle$ -bisimulation takes index into account. This new type of bisimulation will later be used to show the *equivalence* between a CTL formula and a  $\text{SNF}_{\text{CTL}}^g$  formula.

**Proposition 2.** Let  $i \in \{1, 2\}$ ,  $V_1, V_2 \subseteq \mathcal{A}$ ,  $I_1, I_2 \subseteq \text{Ind}$  and  $\mathcal{K}_i = (\mathcal{M}_i, s_i^0)$  ( $i = 1, 2, 3$ ) be initial Ind-structures such that  $\mathcal{K}_1 \leftrightarrow_{\langle V_1, I_1 \rangle} \mathcal{K}_2$  and  $\mathcal{K}_2 \leftrightarrow_{\langle V_2, I_2 \rangle} \mathcal{K}_3$ . Then:

- (i)  $\mathcal{K}_1 \leftrightarrow_{\langle V_1 \cup V_2, I_1 \cup I_2 \rangle} \mathcal{K}_3$ ;
- (ii) If  $V_1 \subseteq V_2$  and  $I_1 \subseteq I_2$  then  $\mathcal{K}_1 \leftrightarrow_{\langle V_2, I_2 \rangle} \mathcal{K}_2$ .

*Proof.* (i) By Proposition 1 we have  $\mathcal{K}_1 \leftrightarrow_{V_1 \cup V_2} \mathcal{K}_3$ . For (i) of Definition 3 we can prove it as follows: for all  $(s, s_1) \in [j]_1$  there is a  $(s', s'_1) \in [j]_2$  such that  $s \leftrightarrow_{V_1} s'$  and  $s_1 \leftrightarrow_{V_1} s'_1$  and there is a  $(s'', s''_1) \in [j]_3$  such that  $s' \leftrightarrow_{V_2} s''$  and  $s'_1 \leftrightarrow_{V_2} s''_1$ , then we have for all  $(s, s_1) \in [j]_1$  there is a  $(s'', s''_1) \in [j]_3$  such that  $s \leftrightarrow_{V_1 \cup V_2} s''$  and  $s_1 \leftrightarrow_{V_1 \cup V_2} s''_1$ . The (ii) of Definition 3 can be proved similarly.

(ii) This can be proved from (ii) of Proposition 1.  $\square$

Obviously, this proposition has the same meaning as Proposition 1, except that indices are now taken into account.

We are now ready to introduce our resolution calculus for forgetting.

## 4 The Calculus

*Resolution* in CTL is a method to decide the satisfiability of a CTL formula. In this section, we shall extend it for our purposes and explore its use to compute forgetting in CTL. We use the transformation rules Trans(1) to Trans(12) and resolution rules (SRES1), ..., (SRES8), RW1, RW2, (ERES1), (ERES2) in (Zhang, Hustadt, and Dixon 2009). Due to space restrictions, these rules are not listed here.

The key problems of this method include: (1) How to fill in the gap between CTL and  $\text{SNF}_{\text{CTL}}^g$  since there are indices for existential quantifiers in  $\text{SNF}_{\text{CTL}}^g$ ; and (2) How to eliminate the irrelevant atoms; both those we want to forget and those that introduced by these rules. We will resolve both problems by  $\langle V, I \rangle$ -bisimulation and a new *eliminate* operator, respectively. For convenience, we use  $V \subseteq \mathcal{A}$  to denote the set we want to forget, and  $V' \subseteq \mathcal{A}$ , with  $V \cap V' = \emptyset$ , to denote the set of atoms introduced in the computation. Moreover,  $\varphi$  be the CTL formula, and  $T_\varphi$  be the set of  $\text{SNF}_{\text{CTL}}^g$  clauses obtained from  $\varphi$  by applying transformation rules on it and  $\mathcal{M} = (S, R, L, [-], s_0)$  unless explicitly stated otherwise. Let  $T, T'$  be two formulae (or sets of formulae),  $I$  a set of indexes introduced in the transformation and  $V'' \subseteq \mathcal{A}$ , by  $T \equiv_{\langle V'', I \rangle} T'$  we mean that  $\forall (\mathcal{M}, s_0) \in \text{Mod}(T)$  there is a  $(\mathcal{M}', s'_0)$  such that  $(\mathcal{M}, s_0) \leftrightarrow_{\langle V'', I \rangle} (\mathcal{M}', s'_0)$  and  $(\mathcal{M}', s'_0) \models T'$  and vice versa.

Algorithm 1 computes forgetting in CTL. The main idea is, first, turning the CTL formula into a set of  $\text{SNF}_{\text{CTL}}^g$  clauses (the *Transform process*), and then computing all the possible resolutions on the specified set of atoms (the *Resolution process*). Third, eliminating all the irrelevant atoms, including *Instantiate*, *Connect* and *Removing atoms*. As a final step, in order to change the obtained result into a CTL formula, we need to go through three sub-processes: *Removing index* (removing the index in the formula), *Replacing atoms* (replacing the atoms in  $V'$  with a formula) and  $T_{\text{CTL}}$  (removing the **start** in  $T$ ). To describe our algorithm clearly, we illustrate it with the following example.

**Example 1.** Let  $\varphi = A((p \wedge q) \cup (f \vee m)) \wedge r$  and  $V = \{p\}$ .

In the following, we will show how to compute the  $F_{\text{CTL}}(\varphi, V)$  step by step using our algorithm.

**Input:** A CTL formula  $\varphi$  and a set  $V$  of atoms  
**Output:**  $ERes(\varphi, V)$

```

1  $T_\varphi \leftarrow \emptyset$  // the initial set of  $\text{SNF}_{\text{CTL}}^g$  clauses of  $\varphi$ ;
2  $V' \leftarrow \emptyset$  // the set of atoms introduced in the
   Transform process;
3  $T_\varphi, V' \leftarrow \text{Transform}(\varphi)$ ;
4  $Res \leftarrow \text{Resolution}(T_\varphi, V \cup V')$ ;
5  $\text{Inst}_{V'} \leftarrow \text{Instantiate}(Res, V')$ ;
6  $\text{Com}_{\text{EF}} \leftarrow \text{Connect}(\text{Inst}_{V'})$ ;
7  $\text{RemA} \leftarrow \text{Removing\_atoms}(\text{Com}_{\text{EF}}, \text{Inst}_{V'})$ ;
8  $\text{NI} \leftarrow \text{Removing\_index}(\text{RemA})$ ;
9  $\text{Rp} \leftarrow \text{Replacing\_atoms}(\text{NI})$ ;
10 return  $\bigwedge_{\psi \in \text{Rp}_{\text{CTL}}} \psi$ .
```

**Algorithm 1:** Computing forgetting - A resolution-based method

#### 4.1 The Transform Process

The *Transform* process, denoted as  $\text{Transform}(\varphi)$ , transforms the CTL formula into a set of  $\text{SNF}_{\text{CTL}}^g$  clauses by applying the rules Trans(1) to Trans(12) in (Zhang, Hustadt, and Dixon 2009)).

The transformation of any CTL formula  $\varphi$  into the set  $T_\varphi$  is a sequence  $T_0, T_1, \dots, T_n = T_\varphi$  of sets of formulae with  $T_0 = \{\text{AG}(\text{start} \supset p), \text{AG}(p \supset \text{simp}(\text{nnf}(\varphi)))\}$  such that for every  $i$  ( $0 \leq i < n$ ),  $T_{i+1} = (T_i \setminus \{\psi\}) \cup R_i$  (Zhang, Hustadt, and Dixon 2009)), where  $p$  is a new atom not appearing in  $\varphi$ ,  $\psi$  is a formula in  $T_i$  which is not in  $\text{SNF}_{\text{CTL}}^g$  clause and  $R_i$  is the result set of applying a matching transformation rule to  $\psi$ . Note that throughout the transformation, formulae are kept in negation normal form (nnf).

**Proposition 3.** Let  $\varphi$  be a CTL formula, then  $\varphi \equiv_{\langle V', I \rangle} T_\varphi$ .

*Proof.* (sketch) This can be proved from  $T_i$  to  $T_{i+1}$  ( $0 \leq i < n$ ) by using one transformation rule on  $T_i$ . We show  $\varphi \equiv_{\langle \{p\}, \emptyset \rangle} T_0$ . Other cases are similar.

First, for every  $(\mathcal{M}_1, s_1) \in \text{Mod}(\varphi)$ , i.e.  $(\mathcal{M}_1, s_1) \models \varphi$ . We can construct an initial Ind-model structure  $\mathcal{M}_2$  which is identical to  $\mathcal{M}_1$  except  $L_2(s_2) = L_1(s_1) \cup \{p\}$ . It is apparent that  $(\mathcal{M}_2, s_2) \models T_0$  and  $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset \rangle} (\mathcal{M}_2, s_2)$ .

Second, for all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_0)$ , it is apparent that  $(\mathcal{M}_1, s_1) \models \varphi$  by the semantic of **start**.  $\square$

**Example 2.** By the Transform process, the result  $T_\varphi$  of the Example 1 can be listed as follows:

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

Besides, the set of new atoms introduced in this process is  $V' = \{x, y, x, w\}$  with  $w$  is a new atom related to  $z \supset \text{AF}x$ .

**Input:** A CTL formula  $\varphi$   
**Output:** A set  $T_\varphi$  of  $\text{SNF}_{\text{CTL}}^g$  clauses and a set  $V'$  of atoms

```

1  $T_\varphi \leftarrow \emptyset$  // the initial set of  $\text{SNF}_{\text{CTL}}^g$  clauses of  $\varphi$ ;
2  $\text{OldT} \leftarrow \{\text{start} \supset z, z \supset \text{simp}(\text{nnf}(\varphi))\}$ ;
3  $V' \leftarrow \{z\}$ ;
4 while  $\text{OldT} \neq T_\varphi$  do
5    $\text{OldT} \leftarrow T_\varphi$ ;
6    $R \leftarrow \emptyset$ ;
7    $X \leftarrow \emptyset$ ;
8   if Chose a formula  $\psi \in \text{OldT}$  that does not a
       $\text{SNF}_{\text{CTL}}^g$  clause then
9     Using a match rule  $Rl$  to transform  $\psi$  into a
       set  $R$  of  $\text{SNF}_{\text{CTL}}^g$  clauses;
10     $X$  is the set of atoms introduced by using  $Rl$ ;
11     $V' \leftarrow V' \cup X$ ;
12     $T_\varphi \leftarrow \text{OldT} \setminus \{\psi\} \cup R$ ;
13  end
14 end
```

**Algorithm 2:**  $\text{Transform}(\varphi)$

#### 4.2 The Resolution Process

The *Resolution* process consists of computing all the possible resolutions of  $T_\varphi$  on  $V \cup V'$ , denoted as  $\text{Resolution}(T_\varphi, V \cup V')$ . A *derivation* on a set  $V \cup V'$  of atoms and  $T_\varphi$  is a sequence  $T_0 = T_\varphi, T_1, T_2, \dots, T_n = Res$  of sets of  $\text{SNF}_{\text{CTL}}^g$  clauses such that  $T_{i+1} = T_i \cup R_i$  for all  $0 \leq i < n$ , where  $R_i$  is a set of clauses obtained as the conclusion of the application of a resolution rule to premises in  $T_i$ . Note that all  $T_i$  ( $0 \leq i \leq n$ ) are set of  $\text{SNF}_{\text{CTL}}^g$  clauses. Besides, if there is a  $T_i$  containing **start**  $\supset \perp$  or  $\top \supset \perp$ , then we have  $F_{\text{CTL}}(\varphi, V) = \perp$ .

Let  $C$  be a clause and  $C'$  be a clause or set of clauses. If  $C$  and  $C'$  are resolvable, then  $\text{res}(C, C')$  is a set of  $\text{SNF}_{\text{CTL}}^g$  clauses, i.e., if there is a resolution rule using  $C$  and  $C'$  as the premises on some given atom. The pseudocode of *Resolution* process is shown in Algorithm 3.

**Proposition 4.** Let  $\varphi$  be a CTL formula, then  $T_\varphi \equiv_{\langle V \cup V', \emptyset \rangle} Res$ .

*Proof.* (sketch) This can be proved from  $T_i$  to  $T_{i+1}$  ( $0 \leq i < n$ ) by using one resolution rule on  $T_i$ . For instance, if we can use the resolution rule (SRES1) on  $\psi \subseteq T_i$  and obtain the result  $R$ , then we can prove  $T_i \equiv T_{i+1}$  with  $T_{i+1} = T_i \cup R$  as follows.

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

Proposition 3 and Proposition 4 mean that  $\varphi \equiv_{\langle V \cup V', I \rangle} Res$ , this resolves a part of the problem (1).

**Example 3.** The resolutions of  $T_\varphi$  obtained from Example 2

**Input:** A set  $T_\varphi$  of  $\text{SNF}_{\text{CTL}}^g$  clauses and a set  $V \cup V'$  of atoms

**Output:** A set  $\text{Res}$  of  $\text{SNF}_{\text{CTL}}^g$  clauses

```

1  $S \leftarrow \{C \mid C \in T_\varphi \text{ and } \text{Var}(C) \cap (V \cup V') = \emptyset\};$ 
2  $\Pi \leftarrow T \setminus S;$ 
3 for ( $p \in V \cup V'$ ) do
4    $\Pi' \leftarrow \{C \in \Pi \mid p \in \text{Var}(C)\};$ 
5    $\Sigma \leftarrow \Pi \setminus \Pi';$ 
6   for ( $C \in \Pi'$  s.t.  $p$  appearing in  $C$  positively) do
7     for ( $C' \in \Pi'$  s.t.  $p$  appearing in  $C'$  negatively and  $C, C'$  are resolvable) do
8        $\Sigma \leftarrow \Sigma \cup \text{res}(C, C');$ 
9        $\Pi' \leftarrow \Pi' \cup \{C'' \in \text{res}(C, C') \mid p \in \text{Var}(C'')\};$ 
10    end
11  end
12   $\Pi \leftarrow \Sigma;$ 
13 end
14  $\text{Res} \leftarrow \Pi \cup S;$ 

```

**Algorithm 3:**  $\text{Resolution}(T, V \cup V')$

on  $V \cup V'$  are listed as follows:

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

### 4.3 The Elimination Process

We say that an atom appears in a formula positively if it is preceded by an even number of negative connectives. For solving problem (2), we should pay focus on the following properties that are obtained from the transformation and resolution rules:

- **(GNA)** For each atom  $p \in \text{Var}(\varphi)$ ,  $p$  does not positively appear in the left hand of the  $\text{SNF}_{\text{CTL}}^g$  clause;
- **(PI)** For each atom  $p \in V'$ , if  $p$  appears in the left hand of a  $\text{SNF}_{\text{CTL}}^g$  clause, then  $p$  appears positively.

The *Elimination* process includes three sub-processes: *Instantiate*, *Connect* and *Removing atoms*. We describe them next.

**The Instantiation Process** An *instantiate formula* of a set  $V''$  of atoms is a formula  $\psi$  such that  $\text{Var}(\psi) \cap V'' = \emptyset$ . Given a formula of the form  $p \supset \psi$  with  $p$  is an atom not in  $V'' \cup \text{Var}(\psi)$ , if  $\psi$  is an instantiate formula of set  $V''$  then we say that  $p$  is instantiated by  $\psi$ . A key point to compute forgetting is eliminating those irrelevant atoms. For this purpose, we define the follow instantiation process.

**Definition 4** (Instantiation). Let  $V'' = V'$  and  $\Gamma = \text{Res}$ , then the process of instantiation is as follows:

- for each global clause  $C = \top \supset D \vee \neg p \in \Gamma$ , if there is one and on one atom  $p \in V'' \cap \text{Var}(C)$  and  $\text{Var}(D) \cap (V \cup V'') = \emptyset$  then let  $C = p \supset D$  and  $V'' := V'' \setminus \{p\}$ ;
- find out all the possible instantiate formulae  $\varphi_1, \dots, \varphi_m$  of  $V \cup V''$  with  $p \supset \varphi_i \in \Gamma$  ( $1 \leq i \leq m$ );
- if there is  $p \supset \varphi_i$  for some  $i \in \{1, \dots, m\}$ , then let  $V'' := V'' \setminus \{p\}$ ;
- for  $\bigwedge_{j=1}^n p_j \supset \varphi \in \Gamma$  ( $i \in \{1, \dots, n\}$ ), if there is  $\alpha \supset p_1, \dots, \alpha \supset p_n \in \Gamma$  and  $\varphi$  is an instantiate formula of  $V \cup V''$ , then let  $\Gamma_1 := \Gamma \cup \{\alpha \supset \varphi\}$ . if  $\Gamma_1 \neq \Gamma$  then let  $\Gamma := \Gamma_1$  go to step (i), else if  $V''$  has been changed before then go to (i) else return  $V \cup V''$ .

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

Intuitively, this process iteratively removes the atoms in  $V'$  that can be represented by the formula of  $\text{Var}(\varphi) \setminus (V'' \cup V)$ . We denote this process as  $\text{Instantiate}(\Gamma, V')$ . After this, we obtain a set of atoms that have not been instantiated by any instantiate formula of  $V \cup V''$ .

**Example 4.** By using the instantiation process on result of Example 3, we obtain that  $x$  is instantiated by  $f \vee m$  at first since there is  $\top \supset \neg x \vee f \vee m \in T_\varphi$  with  $x \in V'$  and  $\text{Var}(f \vee m) \cap (V \cup V') = \emptyset$ , then  $V'' = \{y, z\}$ .

Similarly, due to  $\top \supset \neg y \vee q \in T_\varphi$  and  $y \supset \text{AX}(q \vee f \vee m) \in T_\varphi$ ,  $y$  can be instantiated by  $q \wedge \text{AX}(q \vee f \vee m)$ . And  $z$  can be instantiated by  $r$ . Therefore  $V'' = \{w\}$ . That is  $\text{Instantiate}(\text{Res}, V') = V \cup \{w\}$ .

With the instantiation operator, we guarantee that those atoms in  $V \cup V''$  are indeed irrelevant i.e. should be forgotten.

**The Connect Process** Let  $P$  be a conjunction of literals,  $l, l_1$  be literals, in which  $\text{Var}(l_1) \in V \cup V'$ , and  $C_i$  ( $i \in \{2, 3, 4\}$ ) be classical clauses. Let  $A = \{\text{true} \supset \neg l \vee \neg l_1 \vee C_2, l \supset C_3 \vee C_2\}$ ,  $\alpha = P \supset ((\neg C_3 \wedge \neg C_2) \supset (E_{\langle \text{ind} \rangle} X(C_3 \wedge \neg(C_2 \vee C_4) \supset \text{AXAF}(C_3 \vee C_2))))$ ,  $\beta = P \supset ((\neg C_3 \wedge \neg C_2) \supset (\text{AX}(C_3 \wedge \neg(C_2 \vee C_4) \supset \text{AXAF}(C_3 \vee C_2))))$  and  $\gamma = P \supset ((\neg C_3 \wedge \neg C_2) \supset (E_{\langle \text{ind} \rangle} X(C_3 \wedge \neg(C_2 \vee C_4) \supset E_{\langle \text{ind} \rangle} X E_{\langle \text{ind} \rangle} F(C_3 \vee C_2))))$ , we add the following new rules,

we call it **EF-implication**.

- (EF1)  $\{P \supset \text{Afl}, P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(l_1 \vee C_4)\} \cup A \rightarrow \alpha$   
 (EF2)  $\{P \supset \text{Afl}, P \supset \text{AX}(l_1 \vee C_4)\} \cup A \rightarrow \beta$   
 (EF3)  $\{P \supset \text{E}_{\langle \text{ind} \rangle} \text{Fl}, P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(l_1 \vee C_4)\} \cup A \rightarrow \gamma$   
 (EF4)  $\{P \supset \text{E}_{\langle \text{ind} \rangle} \text{Fl}, P \supset \text{AX}(l_1 \vee C_4)\} \cup A \rightarrow \gamma$ .

By  $\text{Connect}(\text{Instantiate}(\text{Res}, V'))$  we mean using (EF1) to (EF4) on  $\text{Res}$  and replacing  $P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(\neg l \vee C_2 \vee C_4)$  with  $P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(\neg l \vee C_2 \vee C_4) \vee \alpha$  for rule (EF1), replacing  $P \supset \text{AX}(\neg l \vee C_2 \vee C_4)$  with  $P \supset \text{AX}(\neg l \vee C_2 \vee C_4) \vee \beta$  for rule (EF2) and replacing  $P \supset \text{AX}(\neg l \vee C_2 \vee C_4)$  with  $P \supset \text{AX}(\neg l \vee C_2 \vee C_4) \vee \gamma$  for other rules when  $l, C_2, C_3$  and  $C_4$  are instantiate formulae of  $\text{Sub}(\text{Res}, V')$  and  $\text{Var}(l_1) \in V \cup V'$ . The reason why we specify  $l, C_2, C_3$  and  $C_4$  are instantiate formulae of  $\text{Sub}(\text{Res}, V')$  will be explained later.

**Proposition 5.** Let  $\Gamma = \text{Res}$ , we have  $\Gamma \equiv_{\langle V', \emptyset \rangle} \text{Connect}(\text{Instantiate}(\Gamma, V'))$ .

*Proof.* It is obvious from the (EF1) to (EF4).

We prove the (EF1), other rules can be proved similarly. Let  $T_{i+1} = T_i \cup \{\varphi\}$ , where  $\{\varphi\}$  is obtained from  $T_i$  by using rule (EF1) on  $T_i$ , i.e.  $\varphi = P \supset ((\neg C_3 \wedge \neg C_2) \supset (\text{E}_{\langle \text{ind} \rangle} \text{X}(C_3 \wedge \neg(C_2 \vee C_4)) \supset \text{AXAF}(C_3 \vee C_2)))$ . It is apparent that  $T_{i+1} \models T_i$  and  $T_i \models P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(\neg l \vee C_2 \vee C_4)$ . We will show that for all  $(\mathcal{M}, s_0) \in \text{Mod}(T_i)$  there is an initial Ind-structure  $(\mathcal{M}', s'_0)$  such that  $(\mathcal{M}', s'_0) \models T_{i+1}$  and  $(\mathcal{M}', s'_0) \leftrightarrow_{\langle V', \emptyset \rangle} (\mathcal{M}, s_0)$ .

$\forall (\mathcal{M}, s) \models T_i$  we suppose  $(\mathcal{M}, s) \models P \wedge \neg C_3 \wedge \neg C_2$  and  $(\mathcal{M}, s_1) \models C_3 \wedge \neg C_2 \wedge \neg C_4$  with  $(s, s_1) \in [\text{ind}]$  (due to other case can be proved easily). Then we have  $(\mathcal{M}, s) \not\models l$  (by  $(\mathcal{M}, s) \models l \supset C_3 \vee C_2$ ) and  $(\mathcal{M}, s_1) \models l_1$  (by  $(\mathcal{M}, s) \models P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(l_1 \vee C_4)$ ). If  $(\mathcal{M}, s_1) \not\models \text{AXAF}(C_3 \vee C_2)$  then we have  $(\mathcal{M}, s_1) \models l$  due to  $(\mathcal{M}, s) \models \text{AG}(l \supset C_3 \vee C_2)$  and  $(\mathcal{M}, s) \models \text{Afl}$ . And then  $(\mathcal{M}, s_1) \models \neg l_1$  by  $(\mathcal{M}, s) \models \text{AG}(l \supset \neg l_1 \vee C_2)$ . it is a contradiction. Then  $(\mathcal{M}, s_1) \models \text{AXAF}(C_3 \vee C_2)$ .  $\square$

**The Removing\_atoms process** For eliminating those irrelevant atoms, we define the following *Removing\_atoms* operator.

**Definition 5** (Removing\_atoms). Let  $T$  be a set of formulae,  $C \in T$  and  $V$  a set of atoms, then the *Removing\_atoms* operator is defined as:

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

Intuitively, if the formula  $C$  contains at least one of atoms in  $V$  then let  $\text{Removing\_atoms}(C, V)$  be true, else be  $C$  itself. For convenience, for any set  $T$  of formula we have  $\text{Removing\_atoms}(T, V) = \{\text{Removing\_atoms}(r, V) \mid r \in T\}$ .

**Proposition 6.** Let  $V'' = V \cup V'$ ,  $\Gamma = \text{Instantiate}(\text{Res}, V')$  and  $\Gamma_1 = \text{Removing\_atoms}(\text{Connect}(\Gamma), \Gamma)$ , then  $\Gamma_1 \equiv_{\langle V'', \emptyset \rangle} \text{Res}$  and  $\Gamma_1 \equiv_{\langle V'', I \rangle} \varphi$ .

*Proof.* (stretch) Note that for each clause  $C = T \supset H$  in  $\text{Connect}(\Gamma)$ , if  $\Gamma \cap \text{Var}(C) \neq \emptyset$  then there must be an atom  $p \in \Gamma \cap \text{Var}(H)$ . It is apparent that  $\text{Connect}(\Gamma) \models \Gamma_1$ . We show that for all  $(\mathcal{M}, s_0) \in \text{Mod}(\Gamma_1)$  there is a  $(\mathcal{M}', s_0)$  such that  $(\mathcal{M}', s_0) \models \text{Connect}(\Gamma)$  and  $(\mathcal{M}, s_0) \leftrightarrow_{\langle \Gamma, \emptyset \rangle} (\mathcal{M}', s_0)$ . Let  $C = T \supset H$  in  $\text{Connect}(\Gamma)$  with  $\Gamma \cap \text{Var}(C) \neq \emptyset$ , for all  $(\mathcal{M}, s_0) \in \text{Mod}(\Gamma_1)$  we construct  $(\mathcal{M}', s_0)$  s.t.  $(\mathcal{M}, s_0) \leftrightarrow_{\Gamma} (\mathcal{M}', s_0)$  and  $(\mathcal{M}', s_0) \models C$  by adding or deleting some atoms in  $\Gamma$  to the  $L'(s')$  with  $s' \in S'$ .  $\square$

**Example 5.** After removing the clauses that include atoms in  $V = \{p\}$ , the following clauses are left:

$$\begin{array}{ll} \text{start} \supset z & \top \supset \neg z \vee r \\ \top \supset \neg x \vee f \vee m & \top \supset \neg z \vee x \vee y \\ \top \supset \neg y \vee p & \top \supset \neg y \vee q \\ z \supset \text{Afl} & y \supset \text{AX}(x \vee y) \\ \text{start} \supset r & \text{start} \supset x \vee y \\ \top \supset \neg z \vee y \vee f \vee m & y \supset \text{AX}(f \vee m \vee y) \\ \top \supset \neg z \vee x \vee q & y \supset \text{AX}(x \vee q) \\ \text{start} \supset f \vee m \vee y & \text{start} \supset x \vee q \\ \top \supset q \vee \neg z \vee f \vee m & y \supset \text{AX}(q \vee f \vee m) \\ \text{start} \supset f \vee m \vee q & \end{array}$$

In this case, if we do not specify  $l, C_2, C_3$  and  $C_4$  are instantiate formulae of  $\text{Sub}(\text{Res}, V')$ , it is easy to check that all results including  $P \supset \text{E}_{\langle \text{ind} \rangle} \text{X}(\neg l \vee C_2 \vee C_4)$  and  $P \supset \text{AX}(\neg l \vee C_2 \vee C_4)$  obtained from the *Connect* process will be deleted in the *Removing\_atoms* process.

#### 4.4 Remove the Index and Start

The *Removing\_index*(*RemA*) process is to change the set *RemA* obtained above into a set of formulas without the index by using the equations in Proposition 7.

**Proposition 7.** Let  $P, P_i$  and  $\varphi_i$  be CTL formulas, then

- (i)  $\bigwedge_{i=1}^n (P \supset \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_i) \equiv_{\langle \emptyset, \{\text{ind}\} \rangle} P \supset \text{EX} \bigwedge_{i=1}^n \varphi_i$ ,
- (ii)  $\bigwedge_{i=1}^n (P_i \supset \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_i) \equiv_{\langle \emptyset, \{\text{ind}\} \rangle} \bigwedge_{e \in 2^{\{0, \dots, n\}} \setminus \{\emptyset\}} (\bigwedge_{i \in e} P_i \supset \text{EX}(\bigwedge_{i \in e} \varphi_i))$ ,
- (iii)  $\bigwedge_{i=1}^n (P \supset \text{E}_{\langle \text{ind} \rangle} \text{F} \varphi_i) \equiv_{\langle \emptyset, \{\text{ind}\} \rangle} P \supset \bigvee \text{EF}(\varphi_{j_1} \wedge \text{EF}(\varphi_{j_2} \wedge \text{EF}(\dots \wedge \text{EF} \varphi_{j_n})))$ , where  $(j_1, \dots, j_n)$  are sequences of all elements in  $\{0, \dots, n\}$ ,
- (iv)  $P \supset (C \vee \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_1) \wedge P \supset \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_2 \equiv_{\langle \emptyset, \{\text{ind}\} \rangle} P \supset ((C \wedge \text{EX} \varphi_2) \vee \text{EX}(\varphi_1 \wedge \varphi_2))$ ,
- (v)  $P \supset (C \vee \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_1) \vee P \supset \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_2 \equiv_{\langle \emptyset, \{\text{ind}\} \rangle} P \supset (C \vee \text{EX}(\varphi_1 \vee \varphi_2))$ .

*Proof.* (i) For all  $(\mathcal{M}, s_0) \in \text{Mod}(\bigwedge_{i=1}^n (P \supset \text{E}_{\langle \text{ind} \rangle} \text{X} \varphi_i))$  there exists  $(s_0, s_1) \in [\text{ind}]$  such that  $(\mathcal{M}, s_1) \models \varphi_1, \dots, (\mathcal{M}, s_1) \models \varphi_n$ , then there is  $(s_0, s_1) \in R$  s.t.  $(\mathcal{M}, s_1) \models \bigwedge_{i=1}^n \varphi_i$ , i.e.  $(\mathcal{M}, s_0) \models P \supset \text{EX} \bigwedge_{i=1}^n \varphi_i$ .

For each  $(\mathcal{M}, s_0) \in \text{Mod}(P \supset \text{EX} \bigwedge_{i=1}^n \varphi_i)$ , we suppose there is  $(s_0, s_1) \in R$  s.t.  $(\mathcal{M}, s_1) \models \bigwedge_{i=1}^n \varphi_i$ . It is easy to construct an initial Ind-model  $(\mathcal{M}', s_0)$  such that

$(\mathcal{M}', s_0)$  is identical to  $(\mathcal{M}, s_0)$  except the  $(s_0, s_1) \in [ind]$ , i.e.  $(\mathcal{M}, s_0) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}', s_0)$ .

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

(iii) (Only if part) For any model  $(\mathcal{M}, s_0)$  of the right side of the equation if there is  $(\mathcal{M}, s_0) \models P$  then there exists a path  $\pi_{s_0}$  such that  $\varphi_i \in \pi_{s_0}$  ( $1 \leq i \leq n$ ). This means we can construct an initial Ind-model  $(\mathcal{M}', s_0)$  such that  $(\mathcal{M}', s_0)$  is identical to  $(\mathcal{M}, s_0)$  except for each  $(s_j, s_{j+1})$  of  $\pi_{s_0}$  there is  $(s_j, s_{j+1}) \in [ind]$  ( $0 \leq j$ ). It is easy to check  $(\mathcal{M}', s_0) \models \bigwedge_{i=1}^n (P \supset E_{\langle ind \rangle} F \varphi_i)$  and  $(\mathcal{M}, s_0) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}', s_0)$ . The other side can be shown similarly as in (ii).

Other results can be proved similarly.  $\square$

The following proposition follows from Proposition 7.

**Proposition 8. (NI-BRemain)** Let  $I$  be the set of indexes appearing in  $RemA$ , then we have  $RemA \equiv_{\langle \emptyset, I \rangle} Removing\_index(RemA)$ .

In our Example 5 we do not need such process since there is no index in the set of formulae. Let  $T$  be a set of  $SNF_{CTL}^g$  clauses, then we define the following operator:

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

Then  $T \equiv T_{CTL}$  by  $\varphi \equiv AG(\mathbf{start} \supset \varphi)$  (Bolotov 2000).

The last step of our algorithm is to eliminate all the atoms in  $V'$  which has been introduced in the Transform process.

Let  $\Gamma = Instantiate(Res, V')$  and  $\Gamma_1 = Removing\_atoms(Connect(\Gamma))$ , then  $Replacing\_atoms(Removing\_index(\Gamma_1))$  is obtained from  $Removing\_index(\Gamma_1)$  by doing the following three steps for each  $p \in (V' \setminus \Gamma)$ :

- replacing each  $p \supset \varphi_1 \vee \dots \vee p \supset \varphi_n$  with  $p \supset \bigvee_{i=1}^n \varphi_i$ ;
- replacing  $p \supset \varphi_1 \wedge \dots \wedge p \supset \varphi_m$  with  $\varphi_j$  are instantiate formulae of  $\Gamma$  ( $j \in \{1, \dots, m\}$ ) with  $p \supset \psi$ , where  $\psi = \bigwedge_{j=1}^m \varphi_j$  and  $p$  do not appear in  $\varphi_j$ .
- For any formula  $C \in \Gamma_1$ , replacing every  $p$  in  $C$  with  $\psi$ .

Recall that any atom in  $V'$  introduced in the Transform process is a name of the sub-formula of  $\varphi$  (Bolotov 2000). Apparently, this process is just a process of replacing each atom with an equivalent formula. Then we have:

**Proposition 9.** Let  $\Gamma_1 = Instantiate(Res, V')$ ,  $\Gamma_2 = Removing\_atoms(Connect(\Gamma_1, \Gamma_1))$  and  $\Gamma_3 = Replacing\_atoms(Removing\_index(\Gamma_2))$ , then  $\Gamma_2 \equiv_{\langle V' \setminus \Gamma_1, I \rangle} \Gamma_3$  and  $\varphi \equiv_{\langle V \cup V', \emptyset \rangle} (\Gamma_3)_{CTL}$ .

**Example 6.** By using the Replacing\_atoms process on result of Example 5 directly since there is no index in those clauses, we obtain that  $x$  is replaced by  $f \vee m$ . Then  $y$  is replaced by  $q \wedge AX(q \vee f \vee m)$  and  $z$  is replaced by  $r \wedge (f \vee m \vee q) \wedge (f \vee m \vee (q \wedge AX(f \vee m \vee q))) \wedge AF(f \vee m)$ .

## 4.5 An Example for the Connect Process

In order to show the necessity of the Connect process, we give the following example.

**Example 7.** Let  $\psi = AF(p \wedge q) \wedge EX \neg p$  and  $V = \{p\}$ . By the processes Transform and Resolution, we can obtain  $V' = \{f, z, w\}$  and the following set  $Res$  of  $SNF_{CTL}^g$  clauses.

$$\begin{array}{lll} \mathbf{start} \supset z & z \supset AFf & z \supset E_{\langle ind \rangle} X \neg p \\ \top \supset \neg f \vee p & \top \supset \neg f \vee q & z \supset E_{\langle ind \rangle} X \neg f \end{array}$$

According to our Algorithm 1, we have  $Instantiate(Res, V') = \{p, w\}$  since  $f$  can be instantiated by  $q$  and  $z$  can be instantiated by  $AFf$ .

On the one hand, in the Connect process, by using (EF1) rule on the  $Res$  we have  $\alpha = z \supset (\neg q \supset (E_{\langle ind \rangle} X(q \supset AXAFq)))$  and replace  $z \supset E_{\langle ind \rangle} X \neg f \in Res$  with  $z \supset E_{\langle ind \rangle} X \neg f \vee \alpha$  since  $l, C_2, C_3$  and  $C_4$ , which are  $f, \emptyset, q$  and  $\emptyset$  respectively ( $\emptyset$  express that there is not such clause), are instantiate formulae. Apparently,  $z \supset E_{\langle ind \rangle} X \neg f \vee \alpha \equiv z \supset q \vee E_{\langle ind \rangle} X(\neg f \vee \neg q \vee AXAFq)$ .

After the Removing\_atoms process, we have the following set  $RemA$  of formulae:

$$\begin{array}{ll} \mathbf{start} \supset z & z \supset AFf \\ \top \supset \neg f \vee q & z \supset q \vee E_{\langle ind \rangle} X(\neg f \vee \neg q \vee AXAFq) \end{array}$$

Removing the indexes appearing in the  $RemA$ , we obtain the following set  $NI$ :

$$\begin{array}{ll} \mathbf{start} \supset z & z \supset AFf \\ \top \supset \neg f \vee q & z \supset q \vee EX(\neg f \vee \neg q \vee AXAFq) \end{array}$$

Replacing the atoms in  $V'$  that have been instantiated, i.e.  $f$  is replaced with  $q$  and  $z$  is replaced with  $AFq \wedge (q \vee EX(\neg q \vee AXAFq))$ , we have

$$Rp = \{\mathbf{start} \supset AFq \wedge (q \vee EX(\neg q \vee AXAFq))\}.$$

As all the formulas  $\mathcal{F}$  in the  $T_\varphi$  are the form  $AG\mathcal{F}$ , hence we have:

$$Rp_{CTL} = \{AFq \wedge (q \vee EX(\neg q \vee AXAFq))\}$$

i.e.  $ERes(\varphi, V) = AFq \wedge (q \vee EX(\neg q \vee AXAFq))$ . In this case, we can easily check that  $ERes(\varphi, V) \equiv_{\langle V, \emptyset \rangle} \varphi$ .

On the other hand, if we do not using the Connect process, we can easily obtain the result of  $ERes$ , i.e.  $ERes(\varphi, V) = AFq \wedge EX(\neg q)$ . It is apparent that  $ERes(\varphi, V) \not\equiv_{\langle V, \emptyset \rangle} \varphi$ . This can proved by model  $(\mathcal{M}, s_0)$  as in Figure 1 since  $(\mathcal{M}, s_0) \models \varphi$  and  $(\mathcal{M}, s_0) \not\models ERes(\varphi, V)$ .

This example shows why we introduce the EF-implication rules. Intuitively, the result of replacing the atoms that have been instantiated in  $V'$  with an instantiate formula is stronger than our method, because by the *Removing\_atoms* process, we have removed some clauses, such as  $C = \top \supset \neg f \vee p$ , that contain  $f$ . The original one is  $f \supset p \wedge q$ , but after removing  $C$  we only obtain  $f \supset q$ . In this example, there is a clause  $z \supset EX \neg f \in Res$ , after replacing  $f$  with  $q$ , we obtain  $z \supset EX \neg q$ . However, if we do not remove  $C$  (i.e.  $f \supset p \wedge q$ ), then we have  $z \supset EX(\neg q \vee \neg p)$ , this is weaker than  $z \supset EX \neg q$ . In fact, for any model  $(\mathcal{M}, s_0)$  of  $\varphi$ , it might be the case that  $q \in L(s)$  for all next states  $s$  of  $s_0$  and if there is  $q \in L(s)$  for all next states  $s$ , then there must be a next state  $s$  of  $s_0$  with  $p \notin L(s)$  s.t. for all next state  $s'$  of  $s$  there is  $(\mathcal{M}, s') \models AFq$  (see Fig. 1).



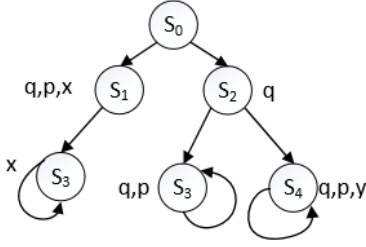


Figure 1: A model  $(\mathcal{M}, s_0)$  of  $\varphi$

#### 4.6 Soundness and Complexity of the Algorithm

If a formula contains no index for its existential quantifier and **start**, only initial K-structures are considered to be candidate models. Because in such a case, it is a CTL formula.

The soundness means that the result  $ERes(\varphi, V)$  obtained from our Algorithm is  $F_{CTL}(\varphi, V)$ , i.e. output the result of forgetting  $V$  from  $\varphi$  when input  $\varphi$  and  $V$  to Algorithm 1.

**Theorem 2** (Soundness). *Let  $V'' = V \cup V'$  and  $\Gamma_1 = ERes(\varphi, V)$ , then*

- (i)  $F_{CTL}(\varphi, V'') \equiv \Gamma_1$ ;
- (ii)  $F_{CTL}(\varphi, V) \equiv \Gamma_1$ .

*Proof.* (i)  $(\Rightarrow)$  For all  $(\mathcal{M}, s_0) \in Mod(F_{CTL}(\varphi, V''))$  there exists  $(\mathcal{M}', s'_0) \in Mod(\varphi)$  s.t.  $(\mathcal{M}, s_0) \leftrightarrow_{V''} (\mathcal{M}', s'_0)$  by the definition of  $F_{CTL}$ , and then there exists  $(\mathcal{M}_1, s_1) \in Mod(\Gamma_1)$  s.t.  $(\mathcal{M}_1, s_1) \leftrightarrow_{V''} (\mathcal{M}', s'_0)$  by Proposition 9. Hence,  $(\mathcal{M}, s_0) \leftrightarrow_{V''} (\mathcal{M}_1, s_1)$  since  $\leftrightarrow$  is an equivalence relation. Therefore,  $(\mathcal{M}, s_0) \models \Gamma_1$  since  $IR(\Gamma_1, V'')$  and Theorem 1.

$(\Leftarrow)$  For all  $(\mathcal{M}_1, s_1) \in Mod(\Gamma_1)$  there exists  $(\mathcal{M}', s'_0) \in Mod(\varphi)$  s.t.  $(\mathcal{M}_1, s_1) \leftrightarrow_{V''} (\mathcal{M}', s'_0)$  by Proposition 9. Hence,  $(\mathcal{M}_1, s_1) \models F_{CTL}(\varphi, V'')$  since  $IR(F_{CTL}(\varphi, V''), V'')$  and  $\varphi \models F_{CTL}(\varphi, V'')$ .

(ii) It is obtained from (i) since  $IR(\varphi, V')$ .  $\square$

Then we can obtain the result of forgetting of Example 4:

$$\begin{aligned} F_{CTL}(\varphi, \{p\}) &\equiv r \wedge (f \vee m \vee q) \wedge AF(f \vee m) \wedge \\ &(f \vee m \vee (q \wedge AX(f \vee m \vee q))) \wedge AG((q \wedge AX(f \vee m \vee q) \\ &\supset AX(f \vee m \vee (q \wedge AX(f \vee m \vee q))))). \end{aligned}$$

**Proposition 10.** *Let  $\varphi$  be a CTL formula and  $V \subseteq \mathcal{A}$ . The time and space complexity of Algorithm 1 are  $O((m+1)2^{4(n+n')})$  where  $|Var(\varphi)| = n$ ,  $|V'| = n'$  ( $V'$  is set of atoms introduced in transformation) and  $m$  is the number of indices introduced during transformation.*

Indeed,  $m$  is at most the number of temporal operators in  $\varphi$ . That is, the computational complexity of our algorithm only depends on the number of atoms and temporal operators in  $\varphi$ . Although it is exponential, it is more efficient than that of the model-based algorithm in (Feng et al. 2020), which not only depends on the number of atoms in  $\mathcal{A}$  but also the number of states.

## 5 Related work

Deciding satisfiability with resolution calculus in Propositional Linear Temporal Logic (PLTL) was introduced in (Fisher 1991) and further discussed in (Fisher 1997; Fisher, Dixon, and Peim 2001). The main idea is to transform PLTL formulas into the a normal form, called Separated Normal Form (SNF) by introducing a new connective **start** that holds only at the beginning of time.

Later, resolution-based satisfiability in CTL was proposed by Bolotov in (Bolotov 2000) and then further refined by Zhang in (Zhang, Hustadt, and Dixon 2009; Zhang, Hustadt, and Dixon 2014). In those papers, the main idea is also to transform CTL formulas into a normal form  $SNF_{CTL}^g$ . But as CTL is branching time temporal logic, they introduce “indices” besides **start** for that purpose. All in all, a complete set of transformation and resolution rules had been proposed for both PLTL and CTL. It has been shown that this transformation is satisfiability-preserving, which also holds for the result obtained from using the resolution rules on the normal form.

Other resolution procedures which are similar to Second-order quantifier elimination, has been used to compute the forgetting or uniform interpretation in propositional logic (Wang 2015) and Modal logic (Herzig and Mengin 2008). In those case, the formula is required to be in a particular form-“CNF” (the definition of CNF in Modal logic can be found in (Herzig and Mengin 2008)).

As aforementioned, the normal form used here for resolution is an extension of CTL with **start** and “index”.

## 6 Conclusion and Future Work

This paper proposed a resolution-based algorithm to compute the forgetting in CTL. Our method extend the resolution calculus in (Zhang, Hustadt, and Dixon 2014) by adding processes that remove irrelevant atoms and which transforming the result back into CTL formula. For this purpose, a new type of binary bisimulation relation, called  $\langle V, I \rangle$ -bisimulation, has been defined to bridge the gap between CTL and  $SNF_{CTL}^g$ . Besides, for connecting the next state and future state we proposed four EF-implication rules. These rules yield the result of the original resolution, and our proposal of Replacing atoms also ensure that we obtain the correct result. Finally, we proved that our algorithm is sound, i.e., return the result of forgetting some set of atoms from a CTL formula. Examples show how to compute forgetting using our algorithm. Moreover, our resolution-based method is more efficient than that of model-base in (Feng et al. 2020).

As for the future research, a Prolog implementation is currently under development, and we are planning to evaluate its practical aspects.<sup>2</sup> Moreover, as for the theory, we are planning to carry out a parameterised complexity analysis on our resolution calculus.

<sup>2</sup><https://github.com/fengrenyan/Resolution-proof-CTL/blob/master/main-code.pl>.

## A Supplementary Material: Proof Appendix

**Proposition 3** Let  $\varphi$  be a CTL formula, then  $\varphi \equiv_{\langle V', I \rangle} T_\varphi$ .

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

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

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

( $\Leftarrow$ ) For all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_0)$ , it is apparent that  $(\mathcal{M}_1, s_1) \models \varphi$  by the semantic of **start**.

By  $\psi \rightarrow_t R_i$  we mean using transformation rules  $t$  on formula  $\psi$  (the formulae  $\psi$  as the premises of rule  $t$ ) and obtaining the set  $R_i$  of transformation results. Let  $X$  be a set of formulas, we will show  $T_i \equiv_{\langle V', I \rangle} T_{i+1}$  by using the transformation rule  $t$ . Where  $T_i = X \cup \{\psi\}$ ,  $T_{i+1} = X \cup R_i$ ,  $V'$  is the set of atoms introduced by  $t$  and  $I$  is the set of indexes introduced by  $t$ . (We will prove this result in  $t \in \{\text{Trans}(1), \text{Trans}(4), \text{Trans}(6)\}$ , other cases can be proved similarly.)

(2) For  $t = \text{Trans}(1)$ :

( $\Rightarrow$ ) For all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_i)$  i.e.  $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{EX}\varphi)$

$\Rightarrow (\mathcal{M}_1, s_1) \models X$  and for every  $\pi$  starting from  $s_1$  and every state  $s_1^j \in \pi$ ,  $(\mathcal{M}, s_1^j) \models \neg q$  or there exists a path  $\pi'$  starting from  $s_1^j$  such that  $(s_1^j, s_1^{j+1}) \in R_1$  and  $(\mathcal{M}, s_1^{j+1}) \models \varphi$ .

We can construct an Ind-model structure  $\mathcal{M}_2$  is identical to  $\mathcal{M}_1$  except  $[ind]_2 = \bigcup_{s \in S} R_s \cup R_y$ , where  $ind$  is the index introduced by using  $\text{Trans}(1)$  on clause  $\text{AG}(q \supset \text{EX}\varphi)$ ,  $R_{s_1^j} = \{(s_1^j, s_1^{j+1}), (s_1^{j+1}, s_1^{j+2}), \dots\}$  and  $R_y = \{(s_x, s_y) \mid \text{for all } s_x \in S \text{ if for all } (s'_1, s'_2) \in \bigcup_{s \in S} R_s, s'_1 \neq s_x \text{ then find exactly one state } s_y \in S \text{ such that } (s_x, s_y) \in R\}$ , which means for every  $s \in S$  there exists exactly a state  $s' \in S$  such that  $(s, s') \in [ind]$  and  $(s, s') \in R$ . It is apparent that  $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}_2, s_2)$  (let  $s_2 = s_1$ ).

$\Rightarrow$  for every path starting from  $s_1$  and every state  $s_1^j$  in this path,  $(\mathcal{M}_2, s_1^j) \models \neg q$  or  $(\mathcal{M}_2, s_1^j) \models \text{EX}\varphi_{\langle ind \rangle}$  (by the semantic of EX)

$\Rightarrow (\mathcal{M}_2, s_1) \models \text{AG}(q \supset \text{E}_{\langle ind \rangle} X\varphi)$

$\Rightarrow (\mathcal{M}_2, s_1) \models X \wedge \text{AG}(q \supset \text{E}_{\langle ind \rangle} X\varphi)$

( $\Leftarrow$ ) For all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_{i+1})$  i.e.  $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{E}_{\langle ind \rangle} X\varphi)$

$\Rightarrow (\mathcal{M}_1, s_1) \models X$  and  $(\mathcal{M}_1, s_1) \models \text{AG}(q \supset \text{E}_{\langle ind \rangle} X\varphi)$

$\Rightarrow$  for every path starting from  $s_1$  and every state  $s_1^j$  in this path,  $(\mathcal{M}_1, s_1^j) \models \neg q$  or there exists a state  $s'$  such that  $(s_1^j, s') \in [ind]_1$  and  $(\mathcal{M}_1, s') \models \varphi$  (by the semantic of  $\text{E}_{\langle ind \rangle} X$ )

$\Rightarrow$  for every path starting from  $s_1$  and every state  $s_1^j$  in this path,  $(\mathcal{M}_1, s_1^j) \models \neg q$  or  $(\mathcal{M}_1, s_1^j) \models \text{EX}\varphi$  (by the semantic of EX)

$\Rightarrow (\mathcal{M}_1, s_1) \models \text{AG}(q \supset \text{EX}\varphi)$

$\Rightarrow (\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{EX}\varphi)$

It is apparent that  $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \emptyset, \{ind\} \rangle} (\mathcal{M}_1, s_1)$ .

(3) For  $t = \text{Trans}(4)$ :

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

The we can construct an Ind-model structure  $\mathcal{M}_2$  as follows:  $\mathcal{M}_2$  is the same with  $\mathcal{M}_1$  except for each state  $s'_1$  if  $(\mathcal{M}_1, s'_1) \models \neg q$  then  $L_2(s'_1) = L_1(s'_1)$ , else if  $(\mathcal{M}_1, s'_1) \models \varphi_1$  then  $L_2(s'_1) = L_1(s'_1)$  else  $L_2(s'_1) = L_1(s'_1) \cup \{p\}$ . It is apparent that  $(\mathcal{M}_2, s'_1) \models (q \supset \varphi_1 \vee p) \wedge (p \supset \varphi_2)$  and  $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p\}, \emptyset \rangle} (\mathcal{M}_2, s_2)$ , then  $(\mathcal{M}_2, s_1) \models T_{i+1}$ .

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

(4) For  $t = \text{Trans}(6)$ :

We prove for  $\text{E}_{\langle ind \rangle} X$ , while for the AX can be proved similarly.

( $\Rightarrow$ ) For all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_i)$ , i.e.  $(\mathcal{M}_1, s_1) \models X \wedge \text{AG}(q \supset \text{E}_{\langle ind \rangle} X\varphi)$

$\Rightarrow (\mathcal{M}_1, s_1) \models X$  and for all  $s'_1 \in S, (\mathcal{M}_1, s'_1) \models q \supset \text{E}_{\langle ind \rangle} X\varphi$   
 $\Rightarrow (\mathcal{M}_1, s'_1) \models \neg q$  or there exists a state  $s'$  such that  $(s'_1, s') \in [ind]$  and  $(\mathcal{M}_1, s') \models \varphi$

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

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

**Proposition 4** Let  $\varphi$  be a CTL formula, then  $T_\varphi \equiv_{\langle V \cup V', \emptyset \rangle} \text{Res}$ .

*Proof.* (sketch) This can be proved from  $T_i$  to  $T_{i+1}$  ( $0 \leq i < n$ ) by using one resolution rule on  $T_i$ .

By  $\psi \rightarrow_r R_i$  we mean using resolution rules  $r$  on set  $\psi$  (the formulae in  $\psi$  as the premises of rule  $r$ ) and obtaining the set  $R_i$  of resolution results. we will show  $T_i \equiv_{\langle V, I \rangle} T_{i+1}$  by using the resolution rule  $r$ . Where  $T_i = X \cup \psi$ ,  $T_{i+1} = X \cup R_i$ ,  $X$  be a set of  $\text{SNF}_{\text{CTL}}^q$  clauses,  $p$  be the proposition corresponding with literal  $l$  used to do resolution in  $r$ .

(1) If  $\psi \rightarrow_r R_i$  by an application of  $r \in \{(\text{SRES1}), \dots, (\text{SRES8}), \text{RW1}, \text{RW2}\}$ , then  $T_i \equiv_{\langle \{p\}, \emptyset \rangle} T_{i+1}$ .

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

(2) If  $\psi \rightarrow_r R_i$  by an application of  $r = (\text{ERES1})$ , then  $T_i \equiv_{\langle \{l, w_{\neg l}^A\}, \emptyset \rangle} T_{i+1}$ .

It has been proved that  $\psi \models R_i$  in (Bolotov 2000), then there is  $T_{i+1} = T_i \cup \Lambda_{\neg l}^A$  and then for all  $(\mathcal{M}_1, s_1) \in \text{Mod}(T_i = X \cup \psi)$  there is a  $(\mathcal{M}_2, s_2) \in \text{Mod}(T_{i+1} = T_i \cup \Lambda_{\neg l}^A)$  s.t.  $(\mathcal{M}_1, s_1) \leftrightarrow_{\langle \{p, w_{\neg l}^A\}, \emptyset \rangle} (\mathcal{M}_2, s_2)$  and vice versa by Proposition 3.

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

**Proposition 6** Let  $V'' = V \cup V'$ ,  $\Gamma = \text{Instantiate}(\text{Res}, V')$  and  $\Gamma_1 = \text{Removing\_atoms}(\text{Connect}(\Gamma), \Gamma)$ , then  $\Gamma_1 \equiv_{\langle V'', \emptyset \rangle} \text{Res}$  and  $\Gamma_1 \equiv_{\langle V'', I \rangle} \varphi$ .

*Proof.* Take note the fact that for each clause  $C = T \supset H$  in  $\text{Connect}(\Gamma)$ , if  $\Gamma \cap \text{Var}(C) \neq \emptyset$  then there must be an atom  $p \in \Gamma \cap \text{Var}(H)$ . It is apparent that  $\text{Connect}(\Gamma) \models \Gamma_1$ , we will show for all  $(\mathcal{M}, s_0) \in \text{Mod}(\Gamma_1)$  there is a  $(\mathcal{M}', s_0)$  such that  $(\mathcal{M}', s_0) \models \text{Connect}(\Gamma)$  and  $(\mathcal{M}, s_0) \leftrightarrow_{\langle \Gamma, \emptyset \rangle} (\mathcal{M}', s_0)$ . Let  $C = T \supset H$  in  $\text{Connect}(\Gamma)$  with  $\Gamma \cap \text{Var}(C) \neq \emptyset$ , for all  $(\mathcal{M}, s_0) \in \text{Mod}(\Gamma_1)$  we construct  $(\mathcal{M}', s_0)$  as  $(\mathcal{M}, s_0)$  except for each  $s \in S$ , if  $(\mathcal{M}, s) \not\models T$  then  $L'(s) = L(s)$ , else:

- (i) if  $(\mathcal{M}, s) \models H$ , then  $L'(s) = L(s)$ ;
- (ii) else if  $(\mathcal{M}, s) \models T$  with  $p \in \text{Var}(H) \cap \Gamma$ , then if  $p$  appearing in  $H$  negatively, then if  $C$  is a global (or an initial) clause then let  $L'(s) = L(s) \setminus \{p\}$  else let  $L'(s^*) = L(s^*) \setminus \{p\}$  for (each (if  $C$  is an A-step or A-sometime clause))  $s^* \in \pi_s$ , else if  $C$  is a global (or an initial) clause then let  $L'(s) = L(s) \cup \{p\}$  else let  $L'(s^*) = L(s^*) \cup \{p\}$  for (each (if  $C$  is a A-step or A-sometime clause))  $s^* \in \pi_s$ . Where  $s^*$  is a next or future state of  $s$  (it depends on the type of the clause: if the clause is a  $X$ -step ( $X \in \{A, E\}$ ) clause then  $s^*$  is the next state, else if the clause is a  $X$ -sometime clause then  $s^*$  is a future state).

It is apparent that  $(\mathcal{M}, s_0) \leftrightarrow_{\langle \Gamma, \emptyset \rangle} (\mathcal{M}', s_0)$ , we will show that  $(\mathcal{M}', s_0) \models \text{Connect}(\Gamma)$  from the following two points:

- (1) For (i), it is apparent  $(\mathcal{M}', s_0) \models C$ ;
- (2) For (ii) talked-above, we show it from the form of  $\text{SNF}_{\text{CTL}}^g$  clauses. Supposing  $C_1$  and  $C_2$  are instantiate formula of  $\Gamma$ :
  - (a) If  $C$  is a global clause, i.e.  $C = \top \supset p \vee C_1$  with  $C_1$  is a disjunction of literals (we suppose  $p$  appearing in  $C$  positively). If there is a  $C' = \top \supset \neg p \vee C_2 \in \text{Connect}(\Gamma)$ , then there is  $\top \supset C_1 \vee C_2 \in \text{Connect}(\Gamma)$  by the resolution  $((\mathcal{M}, s) \models C_2$  due to we have suppose  $(\mathcal{M}, s) \not\models C$ ). It is apparent that  $(\mathcal{M}', s_0) \models C \wedge C'$ .
  - (b) If  $C = T \supset E_{\langle \text{ind} \rangle} X(p \vee C_1)$ . If there is a  $C' = T' \supset E_{\langle \text{ind} \rangle} X(\neg p \vee C_2) \in \text{Connect}(\Gamma)$ , then there is  $T \wedge T' \supset E_{\langle \text{ind} \rangle} X(C_1 \vee C_2) \in \text{Connect}(\Gamma)$  by the resolution  $((\mathcal{M}, s) \models E_{\langle \text{ind} \rangle} X C_2$  due to we have suppose  $(\mathcal{M}, s) \not\models C$ ). It is apparent that  $(\mathcal{M}', s_0) \models C \wedge C'$ .
  - (c) Other cases can be proved similarly.

Therefore, we have  $\Gamma_1 \equiv_{\langle V'', \emptyset \rangle} \text{Res}$  by Proposition 2 and Proposition 5.

And then  $\Gamma_1 \equiv_{\langle V'', I \rangle} \varphi$  follows.  $\square$

**proposition 10** Let  $\varphi$  be a CTL formula and  $V \subseteq \mathcal{A}$ . The time and space complexity of Algorithm 1 are  $O((m + 1)2^{4(n+n')})$ . Where  $|\text{Var}(\varphi)| = n$ ,  $|V'| = n'$  ( $V'$  is set of atoms introduced in transformation) and  $m$  is the number of indices introduced during transformation.

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

## References

- Akintunde, M. E. 2017. Planning for  $\text{ctl}^*$ -specified temporally extended goals via model checking.
- Baier, C., and Katoen, J. 2008. *Principles of Model Checking*. The MIT Press.
- Bolotov, A. 1999. A clausal resolution method for  $\text{ctl}$  branching-time temporal logic. *Journal of Experimental & Theoretical Artificial Intelligence* 11(1):77–93.
- Bolotov, A. 2000. *Clausal resolution for branching-time temporal logic*. Ph.D. Dissertation, Manchester Metropolitan University.
- Browne, M. C.; Clarke, E. M.; and Grumberg, O. 1988. Characterizing finite kripke structures in propositional temporal logic. *Theor. Comput. Sci.* 59:115–131.
- 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. Springer.
- 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.
- Dal Lago, U.; Pistore, M.; and Traverso, P. 2002. Planning with a language for extended goals. In *AAAI/IAAI*, 447–454.
- 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*. Elsevier Science Publishers Ltd.
- Fang, L.; Liu, Y.; and Van Ditmarsch, H. 2019. Forgetting in multi-agent modal logics. *Artificial Intelligence* 266:51–80.
- Feng, R.; Acar, E.; Schlobach, S.; and Wang, Y. 2020. On Sufficient and Necessary Conditions in Bounded CTL. *arXiv e-prints*.
- Fisher, M.; Dixon, C.; and Peim, M. 2001. Clausal temporal resolution. *ACM Transactions on Computational Logic (TOCL)* 2(1):12–56.
- Fisher, M. 1991. A resolution method for temporal logic. In *Ijcai*, volume 91, 99–104. Citeseer.
- Fisher, M. 1997. A normal form for temporal logics and its applications in theorem-proving and execution. *Journal of Logic and Computation* 7(4):429–456.
- Giunchiglia, F., and Traverso, P. 1999. Planning as model checking. In *European Conference on Planning*, 1–20. Springer.
- Herzig, A., and Mengin, J. 2008. Uniform interpolation by resolution in modal logic. In *European Workshop on Logics in Artificial Intelligence*, 219–231. Springer.
- Lang, J., and Marquis, P. 2008. On propositional definability. *Artificial Intelligence* 172(8):991–1017.
- Lang, J., and Marquis, P. 2010. *Reasoning under inconsistency: a forgetting-based approach*. Elsevier Science Publishers Ltd.
- 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. *Artif. Intell.* 128(1-2):143–159.
- Lin, F. 2003. Compiling causal theories to successor state axioms and strips-like systems. *Journal of Artificial Intelligence Research* 19:279–314.
- Liu, Y., and Wen, X. 2011. On the progression of knowledge in the situation calculus. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 976–982. Barcelona, Catalonia, Spain: IJCAI/AAAI.
- Lutz, C., and Wolter, F. 2011. Foundations for uniform interpolation and forgetting in expressive description logics. In *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 989–995. Barcelona, Catalonia, Spain: IJCAI/AAAI.
- Su, K.; Sattar, A.; Lv, G.; and Zhang, Y. 2009. Variable forgetting in reasoning about knowledge. *Journal of Artificial Intelligence Research* 35:677–716.
- Wang, Z.; Wang, K.; Topor, R. W.; and Pan, J. Z. 2010. Forgetting for knowledge bases in DL-Lite. *Annals of Mathematics and Artificial Intelligence* 58(1-2):117–151.
- Wang, Y.; Zhang, Y.; Zhou, Y.; and Zhang, M. 2012. Forgetting in logic programs under strong equivalence. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Thirteenth International Conference*, 643–647. Rome, Italy: AAAI Press.
- Wang, Y.; Wang, K.; and Zhang, M. 2013. Forgetting for answer set programs revisited. In *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, 1162–1168. Beijing, China: IJCAI/AAAI.
- Wang, Y. 2015. On forgetting in tractable propositional fragments. <http://arxiv.org/abs/1502.02799>.
- Wong, K.-S. 2009. *Forgetting in Logic Programs*. Ph.D. Dissertation, The University of New South Wales.
- 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.; Foo, N. Y.; and Wang, K. 2005. 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, 627–634.
- Zhang, L.; Hustadt, U.; and Dixon, C. 2008. First-order resolution for  $\text{ctl}$ . Technical report, Citeseer.
- Zhang, L.; Hustadt, U.; and Dixon, C. 2009. A refined resolution calculus for  $\text{ctl}$ . In *International Conference on Automated Deduction*, 245–260. Springer.
- Zhang, L.; Hustadt, U.; and Dixon, C. 2014. A resolution calculus for the branching-time temporal logic  $\text{ctl}$ . *ACM Transactions on Computational Logic (TOCL)* 15(1):1–38.

Zhao, Y., and Schmidt, R. A. 2017a. Role forgetting for  $\text{alcoqh}(\delta)$ -ontologies using an ackermann-based approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 1354–1361. AAAI Press.

Zhao, Y., and Schmidt, R. A. 2017b. Role forgetting for  $\text{alcoqh}(\Delta)$ -ontologies using an ackermann-based approach. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 1354–1361. AAAI Press.