Forgetting in CTL to Compute Necessary and Sufficient Conditions

Written by AAAI Press Staff[1]*
AAAI Style Contributions by Pater Patel Schneider,
Sunil Issar, J. Scott Penberthy, George Ferguson, Hans Guesgen
[1]Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303
publications20@aaai.org

Abstract

This paper proved a method to computing the forgetting in CTL which has been submitted to IJCAI, from the resolution proposed by Zhang at all by extending the resolution rules.

Introduction

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 (Eiter and Kern-Isberner 2019), 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, Wang, and Zhang 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 2009; Su et al. 2009; Liu and Wen 2011; Fang, Liu, and Van Ditmarsch 2019). In application, forgetting has been used in planning (Lin 2003), conflict solving (Lang and Marquis 2010; Zhang, Foo, and Wang 2005), createing restricted views of ontologies (Zhao and Schmidt 2017), strongest and weakest definitions (Lang and Marquis 2008), SNC (WSC) (Lin 2001) and so on.

Though forgetting has been extensively investigated from various aspects of different logical systems. 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 2009), we research forgetting in CTL from the semantic forgetting point of view. And it is shown that our definition of forgetting satisfies those four postulates of forgetting.

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 the following several parts, we will introduce the structure we use for CTL, syntactic and semantic of CTL and the normal form $\text{SNF}^g_{\text{CTL}}$ (Separated Normal Form with Global Clauses for CTL) of CTL (Zhang, Hustadt, and Dixon 2009).

Model structure in CTL

In general, a transition system [1] is described as a *model structure* (or *Kripke structure*)(in this article, we treat transition system and model structure as the same thing), and a model structure is a triple $\mathcal{M} = (S, R, L)$ (Emerson 1990), where

- S is a set of states,
- $R \subseteq S \times S$ is a total binary relation over S, *i.e.* , for each state $s \in S$ there is a state $s' \in S$ such that $(s, s') \in R$, and
- L is an interpretation function $S \to 2^{\mathcal{A}}$ mapping every state to the set of atoms true at that state.

In this article, the same as (Browne, Clarke, and Grumberg 1988), all of our results apply only to finite Kripke structures. Besides, we restrict ourselves to model structure $\mathcal{M} = (S, R, L, s_0)$ (similar with that in (Zhang, Hustadt, and Dixon 2009)) such that

- there exists a state s_0, called the *initial state*, such that for every state $s \in S$ there is a path π_{s_0} s.t. $s \in \pi_{s_0}$.

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 ($i \leq j$), $(s_j, s_{j+1}) \in R$. By $s' \in \pi_{s_i}$ we mean that s' is a state in the path π_{s_i}.

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

*Primarily Mike Hamilton of the Live Oak Press, LLC, with help from the AAAI Publications Committee

[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) $\to \subseteq S \times \text{Act} \times S$ is a transition relation, (4) $I \subseteq S$ is a set of initial states, (5) AP is a set of atomic propositions, and (6) $L : S \to 2^{\text{AP}}$ is a labeling function.

- $\mathrm{Tr}_0(s)$ 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 $\mathrm{Tr}_n(s')$[2].

By s_n we mean the node at the nth level in tree $\mathrm{Tr}_m(s)$ ($m \geq n$).

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

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, Emerson, and Sistla 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:

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

where $p \in \mathcal{A}$. The formulas $\phi \wedge \psi$ and $\phi \rightarrow \psi$ are defined in a standard manner of propositional logic. The other form formulas of $\mathcal{L}$ are abbreviated using the forms of (1). Notice that, according to the above definition for formulas of CTL, each of the CTL *temporal connectives* has the form XY where $X \in \{\mathrm{A}, \mathrm{E}\}$ and $Y \in \{\mathrm{X}, \mathrm{F}, \mathrm{G}, \mathrm{U}, \mathrm{W}\}$. The priorities for the CTL connectives are assumed to be (from the highest to the lowest):

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

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 $(s_1 = s, s_2, \ldots)$ such that $(\mathcal{M}, s_i) \models \phi$ for each $i \geq 1$;
- $(\mathcal{M}, s) \models \mathrm{E}[\phi_1 \mathrm{U} \phi_2]$ iff $\mathcal{M}$ has a path $(s_1 = s, s_2, \ldots)$ such that, for some $i \geq 1$, $(\mathcal{M}, s_i) \models \phi_2$ and $(\mathcal{M}, s_j) \models \phi_1$ for each $j < i$.

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 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 both the underlying states in model structure and signatures are 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.

Let ϕ be a formula or set of formulas. By *Var*(ϕ) we mean the set of atoms occurring in ϕ. Let $V \subseteq \mathcal{A}$. The formula ϕ is V-*irrelevant*, written $\mathrm{IR}(\phi, V)$, if there is a formula ψ with *Var*$(\psi) \cap V = \emptyset$ such that $\phi \equiv \psi$.

The normal form of CTL

It has proved that any CTL formula φ can be transformed into a set T_φ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ (Separated Normal Form with Global Clauses for CTL) clauses in polynomial time such that φ is satisfiable iff T_φ is satisfiable (Zhang, Hustadt, and Dixon 2008). An important difference between CTL formulae and $\mathrm{SNF}^g_{\mathrm{CTL}}$ is that $\mathrm{SNF}^g_{\mathrm{CTL}}$ 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 $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses is 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 Ind; and (4) temporal operators: $\mathrm{E}_{\langle ind \rangle}\mathrm{X}$, $\mathrm{E}_{\langle ind \rangle}\mathrm{F}$, $\mathrm{E}_{\langle ind \rangle}\mathrm{G}$,$\mathrm{E}_{\langle ind \rangle}\mathrm{U}$ and $\mathrm{E}_{\langle ind \rangle}\mathrm{W}$.

The priorities for the $\mathrm{SNF}^g_{\mathrm{CTL}}$ connectives are assumed to be (from the highest to the lowest):

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

Where the operators in the same brackets have the same priority.

Before talked about the sematic of this language, we introduce the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses at first. The $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses con-

[2] Though some nodes of the tree may have the same label, they are different nodes in the tree.

sists of formulae of the following forms.

$$\mathrm{AG}(\textbf{start} \supset \bigvee_{j=1}^{k} m_j) \qquad (\textit{initial clause})$$

$$\mathrm{AG}(true \supset \bigvee_{j=1}^{k} m_j) \qquad (\textit{global clause})$$

$$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{AX} \bigvee_{j=1}^{k} m_j) \qquad (\mathrm{A} - \textit{step clause})$$

$$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{E}_{\langle ind \rangle}\mathrm{X} \bigvee_{j=1}^{k} m_j) \qquad (\mathrm{E} - \textit{step clause})$$

$$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{AF}l) \qquad (\mathrm{A} - \textit{sometime clause})$$

$$\mathrm{AG}(\bigwedge_{i=1}^{n} l_i \supset \mathrm{E}_{\langle ind \rangle}\mathrm{F}l) \qquad (\mathrm{E} - \textit{sometime clause}).$$

where $k \geq 0$, $n > 0$, **start** is a propositional constant, l_i $(1 \leq i \leq n)$, m_j $(1 \leq j \leq k)$ and l are literals, that is atomic propositions or their negation and ind is an element of Ind (Ind is a countably infinite index set). By clause we mean the classical clause or the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause unless explicitly stated.

Formulae of $\mathrm{SNF}^g_{\mathrm{CTL}}$ over $\mathcal{A}$ are interpreted in Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$, where S, R, L and s_0 is the same as our model structure talked in 2.1 and $[\_] : \mathrm{Ind} \to 2^{(S*S)}$ maps every index $ind \in \mathrm{Ind}$ to a successor function $[ind]$ which is a functional relation on S and a subset of the binary accessibility relation R, such that for every $s \in S$ there exists exactly a state $s' \in S$ such that $(s, s') \in [ind]$ and $(s, s') \in R$. An infinite path $\pi_{s_i}^{\langle ind \rangle}$ is an infinite sequence of states $s_i, s_{i+1}, s_{i+2}, \ldots$ such that for every $j \geq i$, $(s_j, s_{j+1}) \in [ind]$.

Similarly, an *Ind-structure* (or *Ind-interpretation*) is a Ind-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 Ind-structure is *initial*.

The semantics of $\mathrm{SNF}^g_{\mathrm{CTL}}$ is an extension of the semantics of CTL defined in Section 2.2 except using the Ind-model structure $\mathcal{M} = (S, R, L, [\_], s_0)$ replace model structure, $(\mathcal{M}, s_i) \models$ **start** iff $s_i = s_0$ and for all $\mathrm{E}_{\langle ind \rangle}\Gamma$ are explained in the path $\pi_{s_i}^{\langle ind \rangle}$, where $\Gamma \in \{\mathrm{X}, \mathrm{G}, \mathrm{U}, \mathrm{W}\}$. The semantics of $\mathrm{SNF}^g_{\mathrm{CTL}}$ is then defined as shown next as an extension of the semantics of CTL defined in Section 2.2. Let φ and ψ be two $\mathrm{SNF}^g_{\mathrm{CTL}}$ formulae and $\mathcal{M} = (S, R, L, [\_], s_0)$ be an Ind-model structure, the relation "$\models$" between $\mathrm{SNF}^g_{\mathrm{CTL}}$ formulae and $\mathcal{M}$ is defined recursively as follows:

- $(\mathcal{M}, s_i) \models$ **start** iff $s_i = s_0$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{X}\psi$ iff for the path $\pi_{s_i}^{\langle ind \rangle}$, $(\mathcal{M}, s_{i+1}) \models \psi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\psi$ iff for every $s_j \in \pi_{s_i}^{\langle ind \rangle}$, $(\mathcal{M}, s_j) \models \psi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{U} \psi]$ iff there exists $s_j \in \pi_{s_i}^{\langle ind \rangle}$ such that $(\mathcal{M}, s_j) \models \psi$ and for every $s_k \in \pi_{s_i}^{\langle ind \rangle}$, if $i \leq k < j$, then $(\mathcal{M}, s_k) \models \varphi$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{F}\psi$ iff $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\top \mathrm{U} \psi]$;
- $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{W} \psi]$ iff $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}\mathrm{G}\varphi$ or $(\mathcal{M}, s_i) \models \mathrm{E}_{\langle ind \rangle}[\varphi \mathrm{U} \psi]$.

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

Problem Definition

In order to define our problem, *i.e.* forgetting in CTL, we review our definition of V-bisimulation (read ?? for more detials).

Definition 1 *Let $V \subseteq \mathcal{A}$ and $\mathcal{K}_i = (\mathcal{M}_i, s_i)$ $(i = 1, 2)$ be* K*-structures (Ind-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 $(s_1, s_1') \in R_1$

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

Proposition 1 *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 (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) $s_1' \leftrightarrow_{V_i} s_2'$ $(i = 1, 2)$ 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 $\mathcal{K}_1 \leftrightarrow_{V_2} \mathcal{K}_2$.

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

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

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

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

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).

Similar with the V-bisimulation between K-structures, we define the $\langle V, I \rangle$-bisimulation between Ind-structures as follows:

Definition 3 *($\langle V, I\rangle$-bisimulation) Let $\mathcal{M}_i = (S_i, R_i, L_i, [\_]_i, s_0^i)$ with $i \in \{1,2\}$ be two Ind-structures, V be a set of atoms and $I \subseteq Ind$. The $\langle V, I\rangle$-bisimulation $\beta_{\langle V,I\rangle}$ between initial Ind-structures is a set that satisfy $((\mathcal{M}_1, s_0^1), (\mathcal{M}_2, s_0^2)) \in \beta_{\langle V,I\rangle}$ if and only if $(\mathcal{M}_1, s_0^1) \leftrightarrow_V (\mathcal{M}_2, s_0^2)$ and $\forall j \notin I$ there is*

(i) $\forall (s, s_1) \in [j]_1$ there is $(s', s_1') \in [j]_2$ such that $s \leftrightarrow_V s'$ and $s_1 \leftrightarrow_V s_1'$, and

(ii) $\forall (s', s_1') \in [j]_2$ there is $(s, s_1) \in [j]_1$ such that $s \leftrightarrow_V s'$ and $s_1 \leftrightarrow_V s_1'$.

Apparently, this definition is similar with our concept V-bisimulation except that this $\langle V, I\rangle$-bisimulation has introduced the index.

Proposition 2 *Let $i \in \{1,2\}$, $V_1, V_2 \subseteq \mathcal{A}$, $I_1, I_2 \subseteq Ind$ and $\mathcal{K}_i = (\mathcal{M}_i, s_0^i)$ $(i = 1,2,3)$ be 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: $\forall (s, s_1) \in [j]_1$ 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 $s_1' \leftrightarrow_{V_2} s_1''$, and then we have $\forall (s, s_1) \in [j]_1$ 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 (i). ■

The Calculus

Resolution in CTL is a method to decide the satisfiability of a CTL formula. In this part, we will explore a resolution-based method 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).

The key problems of this method include (1) How to fill the gap between CTL and $\text{SNF}^g_{\text{CTL}}$ since there is index for existential quantifier in $\text{SNF}^g_{\text{CTL}}$; and (2) How to eliminate the irrelevant atoms, which we want to forget, in the formula. We will resolve these two problems by $\langle V, I\rangle$-bisimulation and *eliminate* operator respectively. For convenient, we use $V \subseteq \mathcal{A}$ denote the set we want to forget, $V' \subseteq \mathcal{A}$ with $V \cap V' = \emptyset$ the set of atoms introduced in the transformation process below, φ the CTL formula, T_φ be the set of $\text{SNF}^g_{\text{CTL}}$ clause obtained from φ by using transformation rules and $\mathcal{M} = (S, R, L, [\_], s_0)$ unless explicitly stated. Let T, T' be two sets of formulae, I a set of indexes and $V'' \subseteq \mathcal{A}$, by $T \equiv_{\langle V'', I\rangle} T'$ we mean that $\forall (\mathcal{M}, s_0) \in Mod(T)$ 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.

The algorithm of computing the forgetting in CTL is as Algorithm 1. The main idea of this algorithm is to change the CTL formula into a set of $\text{SNF}^g_{\text{CTL}}$ clauses at first (the Transform process), and then compute all the possible resolutions on the specified set of atoms (the Resolution process). Third, eliminating all the irrelevant atoms which dose not be eliminated by the resolution. We will describe this process, which include *Instantiate*, *Connect* and *Removing_atoms* sub-processes, in detail below. Changing the result obtained before into a CTL formula at last, this will include three sub-processes: *Removing_index* (removing the index in the formula), *Replacing_atoms* (replacing the atoms in V' with an formula) and T_{CTL} (removing the **start** in the formula). To describe our algorithm clearly, we illustrate it with the following example.

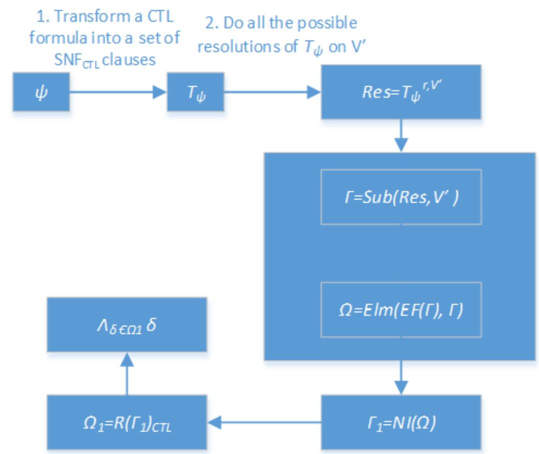

Figure 1: The block diagram of the algorithm

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

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

Input: A CTL formula φ and a set V of atoms
Output: $ERes(\varphi, V)$
$T_\varphi = \emptyset$ // the initial set of $\text{SNF}^g_{\text{CTL}}$ clauses of φ ;
$V' = \emptyset$ // the set of atoms introduced in the process of transforming φ into $\text{SNF}^g_{\text{CTL}}$ clauses;
$T_\varphi, V' \leftarrow Transform(\varphi, V)$;
$Res \leftarrow Resolution(T_\varphi, V')$;
$\text{Inst}_{V'} \leftarrow Instantiate(Res, V')$;
$\text{Com}_{\text{EF}} \leftarrow Connect(\text{Inst}_{V'})$;
$RemA \leftarrow Removing\_atoms(\text{Com}_{\text{EF}}, \text{Inst}_{V'})$;
$\text{NI} \leftarrow Removing\_index(RemA)$;
$\text{Rp} \leftarrow Replacing\_atoms(\text{NI})$;
return $\bigwedge_{\psi \in \text{Rp}_{\text{CTL}}} \psi$.

Algorithm 1: Computing forgetting - A resolution-based method

The Transform process

The *Transform* process, denoted as *Transform*(φ), is to transform the CTL formula into a set of $\text{SNF}^g_{\text{CTL}}$ clauses by using the rules Trans(1) to Trans(12) in (Zhang, Hustadt, and Dixon 2009)).

The transformation of an arbitrary CTL formula φ into the set T_φ is a sequence $T_0, T_1, \ldots, T_n = T_\varphi$ of sets of formulae with $T_0 = \{\text{AG}(\textbf{start} \supset p), \text{AG}(p \supset \textbf{simp}(\textbf{nnf}(\varphi)))\}$ 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 φ, ψ is a formula in T_i not in $\text{SNF}^g_{\text{CTL}}$ clause and R_i is the result set of applying a matching transformation rule to ψ. Note that throughout the transformation formulae are kept in negation normal form.

Proposition 3 *Let φ 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.

■

This means that φ has the same models with T_φ excepting that the atoms in V' and the relations $[i]$ with $i \in I$.

Input: A CTL formula φ
Output: A set T of $\text{SNF}^g_{\text{CTL}}$ clauses and a set V' of atoms
$T = \emptyset$ // the initial set of $\text{SNF}^g_{\text{CTL}}$ clauses of φ ;
$OldT = \{$**start** $\supset z, z \supset \varphi\}$;
$V' = \{z\}$;
while $OldT \neq T$ **do**
    $OldT = T$;
    $R = \emptyset$;
    $X = \emptyset$;
    if *Chose a formula $\psi \in OldT$ that dose not a* $\text{SNF}^g_{\text{CTL}}$ *clause* **then**
        Using a match rule Rl to transform ψ into a set R of $\text{SNF}^g_{\text{CTL}}$ clauses;
        X is the set of atoms introduced by using Rl;
        $V' = V' \cup X$;
        $T = OldT \setminus \{\psi\} \cup R$;
    end
end

Algorithm 2: *Transform*(φ)

Example 2 By the *Transform* process, the result T_φ of the Example 1 can be listed as follows:

1.**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 $V' = \{x, y, x\}$.

The Resolution process

The *Resolution* process is to compute all the possible resolutions of T_φ on $V \cup V'$, denoted as *Resolution*$(T_\varphi, V \cup V')$. A *derivation* on a set $V \cup V'$ of atoms and T_φ is a sequence $T_0, T_1, T_2, \ldots, T_n = Res$ of sets of $\text{SNF}^g_{\text{CTL}}$ clauses such that $T_0 = T_\varphi$ and $T_{i+1} = T_i \cup R_i$ 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 the T_i ($0 \leq i \leq n$) are set of $\text{SNF}^g_{\text{CTL}}$ clauses. Besides, if there is a T_i containing **start** $\supset \perp$ or $\top \supset \perp$, then we have $\text{F}_{\text{CTL}}(\varphi, V) = \perp$. Given two clauses C and C', we call C and C' are resolvable, the result denote as $res(C, C')$, if there is a resolution rule using C and C' as the premises on some given atom. And the pseudocode of algorithm *Resolution* is as Algorithm 3.

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

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.

■

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

Input: A set T of $\text{SNF}^g_{\text{CTL}}$ clauses and a set V' of atoms
Output: A set Res of $\text{SNF}^g_{\text{CTL}}$ clauses
$S = \{C | C \in T$ and $Var(C) \cap V = \emptyset\}$;
$\Pi = T \setminus S$;
for *($p \in V \cup V'$)* **do**
    $\Pi' = \{C \in \Pi | p \in Var(C)\}$;
    $\Sigma = \Pi \setminus \Pi'$;
    for *($C \in \Pi'$ s.t. p appearing in C positively)* **do**
        for *($C' \in \Pi'$ s.t. p appearing in C' negatively and C, C' are resolvable)* **do**
            $\Sigma = \Sigma \cup \{res(C, C')\}$;
            $\Pi' = \Pi' \cup \{C'' = res(C, C') | p \in Var(C'')\}$;
        end
    end
    $\Pi = \Sigma$;
end
$Res = \Pi \cup S$;

Algorithm 3: *Resolution*(T, V')

Example 3 The resolution of T_φ obtained from Example 2 on $V \cup V'$ is as follows:

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

The Elimination process

For resolving problem (2), we should pay attention to the following properties that obtained from the transformation and resolution rules at first:

- **(GNA)** for all atom p in *Var*(φ), p do not positively appear in the left hand of the $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause;
- **(PI)** for each atom $p \in V'$, if p appearing in the left hand of a $\mathrm{SNF}^g_{\mathrm{CTL}}$ clause, then p appear positively.

This *Elimination* process include three sub-processes: *Instantiate*, *Connect* and *Removing_atoms*. We will described those sub-processes carefully now.

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

Definition 4 *[instantiation] Let $V'' = V'$ and $\Gamma = Res$, then the process of instantiation is as follows:*

(i) for each global clause $C = \top \supset D \vee \neg p \in \Gamma$, if there is one and on one atom $p \in V'' \cap Var(C)$ and $Var(D) \cap (V \cup V'') = \emptyset$ then let $C = p \supset D$ and $V'' := V'' \setminus \{p\}$;

(ii) find out all the possible instantiate formulae $\varphi_1, ..., \varphi_m$ of $V \cup V''$ in the $p \supset \varphi_i \in \Gamma$ ($1 \leq i \leq m$);

(iii) if there is $p \supset \varphi_i$ for some $i \in \{1, \dots, m\}$, then let $V'' := V'' \setminus \{p\}$, which means p is a instantiate formula;

(iv) for $\bigwedge_{j=1}^m p_j \supset \varphi_i \in \Gamma$ ($i \in \{1, \dots, m\}$), if there is $\alpha \supset p_1, \dots, \alpha \supset p_m \in \Gamma$ then let $\Gamma_1 := \Gamma \cup \{\alpha \supset \varphi\}$. if $\Gamma_1 \neq \Gamma$ then let $\Gamma := \Gamma_1$ go to step (i), else return $V \cup V''$.

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

We denote this process as *Instantiate*(Γ, V'), which can be described as the following Algorithm 4. After this process we obtain a set of atoms that do not has been instantiated by any instantiate formula of $V \cup V''$ in this process.

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 *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 $y \supset \mathrm{AX}(q \vee f \vee m) \in T_\varphi$, then y can be instantiated by $q \wedge \mathrm{AX}(q \vee f \vee m)$. And z can be instantiated by r. Therefore $V'' = \emptyset$ That is *Instantiate*$(T_\varphi^{r,V\cup V'}, V') = V$, which means all the introduced atoms are instantiated.

By instantiation operator, we guarantee those atoms in $V \cup V''$ are really irrelevant, *i.e.* should be forgot.

Input: A set Γ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses φ and $V, V' \subseteq \mathcal{A}$
Output: A set of atoms
Let $V'' := V'$;
Let $V_1 = \emptyset$;
Let $\Gamma_1 := \emptyset$;
Let $\Gamma_2 := \Gamma$;
while *($\Gamma_1 \neq \Gamma_2$ or $V_1 \neq V''$)* **do**
  $\Gamma_1 := \Gamma_2$;
  $V_1 := V''$;
  for *($C \in \Gamma_2$)* **do**
    if *(C is a global clause)* **then**
      Let $C := D \vee \neg p$;
      if *($p \in V'' \cap Var(C)$ and $Var(D) \cap V == \emptyset$)* **then**
        $C := p \supset D$;
        $V'' := V'' \setminus \{p\}$;
      end
    end
  end
  for *($C \in \Gamma_2$)* **do**
    if *($C == p \supset \varphi$ and $p \in V''$ and $Var(\varphi) \cap V \cup V'' == \emptyset$)* **then**
      $V'' := V'' \setminus \{p\}$;
    end
  end
  for *($C \in \Gamma_2$)* **do**
    if *($C == \bigwedge_{j=1}^m p_j \supset \varphi$ and $Var(\varphi) \cap V \cup V'' == \emptyset$)* **then**
      if *(there is $\alpha \supset p_1, \dots, \alpha \supset p_m \in \Gamma_2$)* **then**
        $\Gamma_2 := \Gamma_2 \cup \{\alpha \supset \varphi\}$;
      end
    end
  end
end
return $V \cup V''$.

Algorithm 4: Computing *Instantiate*(Γ, V')

The Connect process Let P be a conjunction of literals, l, l_1 be literals, in which *Var*$(C_1) \cap V \cup V' = \emptyset$, and C_i ($i \in \{2, 3, 4\}$) be classical clauses. Let $A = \{l \supset \neg l_1 \vee C_2, l \supset C_3 \vee C_2\}$, $\alpha = P \supset ((\neg C_3 \wedge \neg C_2) \supset (\mathrm{E}_{\langle ind \rangle}\mathrm{X}(C_3 \wedge \neg(C_2 \vee C_4) \supset \mathrm{AXAF}(C_3 \vee C_2))))$ and $\beta = P \supset ((\neg C_3 \wedge \neg C_2) \supset (\mathrm{AX}(C_3 \wedge \neg(C_2 \vee C_4) \supset \mathrm{AXAF}(C_3 \vee C_2))))$,we add following new rules, we call it **EF** imply.

(EF1)$\{P \supset \mathrm{AF}l, P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{X}(l_1 \vee C_4)\} \cup A \to \alpha$

(EF2)$\{P \supset \mathrm{AF}l, P \supset \mathrm{AX}(l_1 \vee C_4)\} \cup A \to \beta$

(EF3)$\{P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{F}l, P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{X}(l_1 \vee C_4)\} \cup A \to \alpha$

(EF4)$\{P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{F}l, P \supset \mathrm{AX}(l_1 \vee C_4)\} \cup A \to \alpha$

By *Connect*(*Instantiate*$(T_\varphi^{r,V\cup V'}, V')$) we mean using (EF1) to (EF4) on $T_\varphi^{r,V\cup V'}$ and replacing $P \supset \mathrm{AX}(\neg l \vee C_2 \vee C_4)$ with $P \supset \mathrm{AX}(\neg l \vee C_2 \vee C_4) \vee \beta$ for rule (EF2) and replacing $P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{X}(\neg l \vee C_2 \vee C_4)$ with $P \supset \mathrm{E}_{\langle ind \rangle}\mathrm{X}(\neg l \vee C_2 \vee C_4) \vee \alpha$ for other rules when l, C_2,

C_3 and C_4 are instantiate formulae of Sub$(T_\varphi^{r,V\cup V'}, V')$ and *Var*$(l_1) \in V \cup V'$. This process can be described as Algorithm 5.

Input: A set Γ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses, a set of A-step clauses and a set of E-step clauses
Output: A set of formulae

```
for (C ∈ A) do
    Let C == P ⊃ AFl;
    if (P ⊃ E⟨ind⟩X(l1 ∨ C4), l ⊃ ¬l1 ∨ C2, l ⊃ C3 ∨ C2 ∈ Γ and l, C2, C3, C4 are instantiate formulae) then
        Replacing P ⊃ E⟨ind⟩X(¬l ∨ C2 ∨ C4) with P ⊃ E⟨ind⟩X(¬l ∨ C2 ∨ C4) ∨ α;
    end
    if (P ⊃ AX(l1 ∨ C4), l ⊃ ¬l1 ∨ C2, l ⊃ C3 ∨ C2 ∈ Γ and l, C2, C3, C4 are instantiate formulae) then
        Replacing P ⊃ AX(¬l ∨ C2 ∨ C4) with P ⊃ AX(¬l ∨ C2 ∨ C4) ∨ β;
    end
end
for (C ∈ E) do
    Let C == P ⊃ E⟨ind⟩Fl;
    if (P ⊃ E⟨ind⟩X(l1 ∨ C4), l ⊃ ¬l1 ∨ C2, l ⊃ C3 ∨ C2 ∈ Γ or P ⊃ AX(l1 ∨ C4), l ⊃ ¬l1 ∨ C2, l ⊃ C3 ∨ C2 ∈ Γ and l, C2, C3, C4 are instantiate formulae) then
        Replacing P ⊃ E⟨ind⟩X(¬l ∨ C2 ∨ C4) with P ⊃ E⟨ind⟩X(¬l ∨ C2 ∨ C4) ∨ α;
    end
end
return Γ.
```

Algorithm 5: Computing *Connect*(Γ, V)

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

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

We prove the (EF1), for 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.* $\varphi = P \supset ((\neg C_3 \wedge \neg C_2) \supset (\mathrm{E}_{\langle ind\rangle}\mathrm{X}(C_3 \wedge \neg(C_2 \vee C_4) \supset \mathrm{AXAF}(C_3 \vee C_2))))$. It is apparent that $T_{i+1} \models T_i$ and $T_i \models P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}(\neg l \vee C_2 \vee C_4)$. We will show that $\forall(\mathcal{M}, s_0) \in Mod(T_i)$ 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 $(s, s_1) \in [ind]$ (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 $(\mathcal{M}, s) \models P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}(l_1 \vee C_4)$). If $(\mathcal{M}, s_1) \not\models \mathrm{AXAF}(C_3 \vee C_2)$ then we have $(\mathcal{M}, s_1) \models l$ due to $(\mathcal{M}, s) \models \mathrm{AG}(l \supset C_3 \vee C_2)$ and $(\mathcal{M}, s) \models \mathrm{AF}l$. And then $(\mathcal{M}, s_1) \models \neg l_1$ by $(\mathcal{M}, s) \models \mathrm{AG}(l \supset \neg l_1 \vee C_2)$. It is contract with our supposing. Then $(\mathcal{M}, s_1) \models \mathrm{AXAF}(C_3 \vee C_2)$.

$\blacksquare$

The Removing_atoms process For eliminate those irrelevant atoms, we can do the following elimination operator.

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

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

For convenience, for any set T of formula we have *Removing_atoms*$(T, V) = \{$*Removing_atoms*$(r, V) | r \in T\}$.

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

Proof: Note the fact that for each clause $C = T \supset H$ in *Connect*(Γ), if $\Gamma \cap Var(C) \neq \emptyset$ then there must be an atom $p \in \Gamma \cap Var(H)$. It is apparent that *Connect*$(\Gamma) \models \Gamma_1$, we will show $\forall(\mathcal{M}, s_0) \in Mod(\Gamma_1)$ there is a $(\mathcal{M}', s_0)$ such that $(\mathcal{M}', s_0) \models$ *Connect*(Γ) and $(\mathcal{M}, s_0) \leftrightarrow_{\langle \Gamma, \emptyset\rangle} (\mathcal{M}', s_0)$. Let $C = T \supset H$ in *Connect*(Γ) with $\Gamma \cap Var(C) \neq \emptyset$, $\forall(\mathcal{M}, s_0) \in Mod(\Gamma_1)$ 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 $p \in Var(H) \cap V$, 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 $L'(s_1) = L(s_1) \setminus \{p\}$ for (each (if C is an A-step or A-sometime clause)) $(s, s_1) \in R$, else if C is a global (or an initial) clause then let $L'(s) = L(s) \cup \{p\}$ else let $L'(s_1) = L(s_1) \cup \{p\}$ for (each (if C is a A-step or A-sometime clause)) $(s, s_1) \in R$.

(iii) for other clause $C = Q \supset H$ with $p \in Var(H) \cap \Gamma$, we can do it as (ii).

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

(1) For (ii) talked-above, we show it from the form of $\mathrm{SNF}^g_{\mathrm{CTL}}$ clauses. Supposing C_1 and C_2 are instantiate formula of Γ:

   (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 $C' = \top \supset \neg p \vee C_2 \in$ *Connect*(Γ), then there is $\top \supset C_1 \vee C_2 \in$ *Connect*(Γ) 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 $C = T \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}(p \vee C_1)$. If there is a $C' = T' \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}(\neg p \vee C_2) \in$ *Connect*(Γ), then there is $T \wedge T' \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}(C_1 \vee C_2) \in$ *Connect*(Γ) by the resolution ($(\mathcal{M}, s) \models \mathrm{E}_{\langle ind\rangle}\mathrm{X}C_2$ 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.

(2) (iii) can be proved as (ii) due to the fact we point at the beginning.

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

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

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

start $\supset z$	$\top \supset \neg z \vee r$	$\top \supset \neg x \vee f \vee m$
$\top \supset \neg z \vee x \vee y$	$\top \supset \neg y \vee p$	$\top \supset \neg y \vee q$
$z \supset \mathrm{AF}x$	$y \supset \mathrm{AX}(x \vee y)$	**start** $\supset r$
start $\supset x \vee y$	$\top \supset \neg z \vee y \vee f \vee m$	$y \supset \mathrm{AX}(f \vee m \vee y)$
$\top \supset \neg z \vee x \vee q$	$y \supset \mathrm{AX}(x \vee q)$	**start** $\supset f \vee m \vee y$
start $\supset x \vee q$	$\top \supset q \vee \neg z \vee f \vee m$	$y \supset \mathrm{AX}(q \vee f \vee m)$
start $\supset f \vee m \vee q$		

Remove the Index and start

The *Removing_index*(Γ) process is to change the set Γ of $\mathrm{SNF}^g_{\mathrm{CTL}}$ into a set of formulas without the index by using the equations in Proposition 7. 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 φ_i be CTL formulas, then*

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

(ii) $\bigwedge_{i=1}^n P_i \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_i \equiv_{\langle\emptyset,\{ind\}\rangle} \bigwedge_{e\in 2^{\{0,\dots,n\}}\setminus\{\emptyset\}}(\bigwedge_{i\in e} P_i \supset \mathrm{EX}(\bigwedge_{i\in e}\varphi_i))$,

(iii) $\bigwedge_{i=1}^n P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{F}\varphi_i \equiv_{\langle\emptyset,\{ind\}\rangle} P \supset \bigvee \mathrm{EF}(\varphi_{j_1} \wedge \mathrm{EF}(\varphi_{j_2} \wedge \mathrm{EF}(\cdots \wedge \mathrm{EF}\varphi_{j_n})))$, *where $(j_1,\dots,j_n)$ are sequences of all elements in* $\{0,\dots,n\}$,

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

(v) $P \supset (C \vee \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_1) \vee P \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\varphi_2 \equiv_{\langle\emptyset,\{ind\}\rangle} P \supset (C \vee \mathrm{EX}(\varphi_1 \vee \varphi_2))$.

Proof: It is easy to check. ■

Lemma 1 ***(NI-BRemain)*** *Let I is the set of indexes in* RemA*, we have* RemA $\equiv_{\langle\emptyset,I\rangle}$ Removing_index(RemA).

Proof: It is easy checking that from the definition of *Removing_index*. ■

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

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

Then $T \equiv T_{\mathrm{CTL}}$ by $\varphi \equiv \mathrm{AG}(\textbf{start} \supset \varphi)$ (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 $V'' = V \cup V'$, Γ = *Instantiate*(Res, V') and Γ_1 = *Removing_atoms*(*Connect*(Γ)), then *Replacing_atoms*(*Removing_index*(Γ_1)) is obtained from *Removing_index*(Γ_1) by doing the following two steps for each $p \in (V' \setminus \Gamma)$:

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

Apparently, this process is just a process of replacing each atom with an equivalent formula. Then we have:

Proposition 8 *Let* Γ_1 = Instantiate(Res, V'), Γ_2 = Removing_atoms (Connect$(\Gamma_1), \Gamma_1$) *and* Γ_3 = Replacing_atoms(Removing_index(Γ_2))*, then* $\Gamma_2 \equiv_{\langle V'\setminus\Gamma_1,I\rangle} \Gamma_3$ *and* $\varphi \equiv_{\langle V\cup V',\emptyset\rangle} (\Gamma_3)_{CTL}$.

Proof: For each p talked above is a name of the formula ψ, *i.e.* $p \leftrightarrow \psi$. Then $\Gamma_2 \equiv_{\langle (V'\setminus\Gamma_1),\emptyset\rangle} \Gamma_3$, and then $\Gamma_2 \equiv_{\langle V\cup V',I\rangle} \Gamma_3$ by (V) of Proposition 1.

Therefore, $\varphi \equiv_{\langle V\cup V',\emptyset\rangle} (\Gamma_3)_{CTL}$ by Proposition 6 and the definitions of *Removing_index* and T_{CTL}. ■

Example 6 By using the *Removing_atoms* process on result of Example 5 directly since there is not index in those clauses, we obtain that x is replaced by $f \vee m$ at first, then y is replaced by $q \wedge \mathrm{AX}(q \vee f \vee m)$ and z is replaced by $r \wedge (f \vee m \vee q) \wedge (f \vee m \vee (q \wedge \mathrm{AX}(f \vee m \vee q))) \wedge \mathrm{AF}(f \vee m)$.

An example for Connect process

In order to show the necessity of the Connect process, we see the following example at first.

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

start $\supset z$	$z \supset \mathrm{AF}f$	$z \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\neg p$
$\top \supset \neg f \vee p$	$\top \supset \neg f \vee q$	$z \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\neg f$

On the one hand, according to our Algorithm 1, we have *Instantiate*$(Res, V') = V$ since f can be instantiated by q and z can be instantiated by $\mathrm{AF}f$.

In the *Connect* process, by using **EF1** rule on the Res we have $\alpha = z \supset (\neg q \supset (\mathrm{E}_{\langle ind\rangle}\mathrm{X}(q \supset \mathrm{AXAF}q)))$ and replace $z \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\neg f \in Res$ with $z \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\neg f \vee \alpha$ since l, C_2, C_3 and C_4 are instantiate formulae. Apparently, $z \supset \mathrm{E}_{\langle ind\rangle}\mathrm{X}\neg f \vee \alpha \equiv z \supset q \vee \mathrm{E}_{\langle ind\rangle}\mathrm{X}(\neg f \vee \neg q \vee \mathrm{AXAF}q)$.

After the *Removing_atoms* process, we have the following set *RemA* of formulae:

start $\supset z$	$z \supset \mathrm{AF}f$
$\top \supset \neg f \vee q$	$z \supset q \vee \mathrm{E}_{\langle ind\rangle}\mathrm{X}(\neg f \vee \neg q \vee \mathrm{AXAF}q)$

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

start $\supset z$	$z \supset \mathrm{AF}f$
$\top \supset \neg f \vee q$	$z \supset q \vee \mathrm{EX}(\neg f \vee \neg q \vee \mathrm{AXAF}q)$

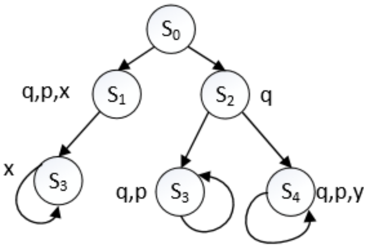

Figure 2: A model $(\mathcal{M}, s_0)$ of φ

Replacing the atoms in V' that have been instantiated, we have

$$Rp = \{\textbf{start} \supset \mathrm{AF}q \wedge (q \vee \mathrm{EX}(\neg q \vee \mathrm{AXAF}q))\}.$$

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

$$Rp_{\mathrm{CTL}} = \{\mathrm{AF}q \wedge (q \vee \mathrm{EX}(\neg q \vee \mathrm{AXAF}q))\}.$$

i.e. $ERes(\varphi, V) = \mathrm{AF}q \wedge (q \vee \mathrm{EX}(\neg q \vee \mathrm{AXAF}q))$. 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. $ERes(\varphi, V) = \mathrm{AF}q \wedge \mathrm{EX}(\neg q)$. 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 2 since $(\mathcal{M}, s_0) \models \varphi$ and $(\mathcal{M}, s_0) \not\models ERes(\varphi, V)$.

This example shows that why we introduce the **EF** imply rules. Intuitively, the result of replacing the atoms that have been instantiated in V' with an instantiate formula is more stronger than our method, because by the *Removing_atoms* process, we have removing 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 that $f \supset q$. In this example, there is a clauses $z \supset \mathrm{EX}\neg f \in Res$, after replacing f with q, we obtain $z \supset \mathrm{EX}\neg q$. However, if we do not removing C (i.e. $f \supset p \wedge q$), then we have $z \supset \mathrm{EX}(\neg q \vee \neg p)$, this is weaker than $z \supset \mathrm{EX}\neg q$. In fact, for any model $(\mathcal{M}, s_0)$ of φ there is not necessary $q \notin L(s)$ for some next state s and if there is q for all next states, then there must be a next state s with $p \notin L(s)$ s.t. for all next state s' of s there is $(\mathcal{M}, s') \models \mathrm{AF}q$ (see Fig. 2).

The Correction and Complexity of the Algorithm

In the case that formula dose not include index, we use model structure $\mathcal{M} = (S, R, L, s_0)$ to interpret formula instead of Ind-model structure.

Theorem 1 (Resolution-based CTL-forgetting) *Let $V'' = V \cup V'$ and $\Gamma_1 = \mathrm{ERes}(\varphi, V)$, then*

(i) $\mathrm{F}_{\mathrm{CTL}}(\varphi, V'') \equiv \Gamma_1$;

(ii) $\mathrm{F}_{\mathrm{CTL}}(\varphi, V) \equiv \bigwedge_{\psi \in \Gamma_1} \psi$.

Proof: (i) ($\Rightarrow$) $\forall(\mathcal{M}, s_0) \in Mod(\mathrm{F}_{\mathrm{CTL}}(\varphi, V''))$
$\Rightarrow \exists(\mathcal{M}', s_0') \in Mod(\varphi)$ s.t. $(\mathcal{M}, s_0) \leftrightarrow_{V''} (\mathcal{M}', s_0')$
$\Rightarrow \exists(\mathcal{M}_1, s_1) \in Mod(\Gamma_1)$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{V''} (\mathcal{M}', s_0')$ from Proposition 8
$\Rightarrow (\mathcal{M}, s_0) \leftrightarrow_{V''} (\mathcal{M}_1, s_1)$
$\Rightarrow (\mathcal{M}, s_0) \models \Gamma_1$ (IR(Γ_1, V''))
($\Leftarrow$)$\forall(\mathcal{M}_1, s_1) \in Mod(\Gamma_1)$
$\Rightarrow \exists(\mathcal{M}', s_0') \in Mod(\varphi)$ s.t. $(\mathcal{M}_1, s_1) \leftrightarrow_{V''} (\mathcal{M}', s_0')$
$\Rightarrow (\mathcal{M}_1, s_1) \models \mathrm{F}_{\mathrm{CTL}}(\varphi, V'')$ (IR($\mathrm{F}_{\mathrm{CTL}}(\varphi, V''), V''$) and $\varphi \models \mathrm{F}_{\mathrm{CTL}}(\varphi, V'')$)
(ii) It is obtained from (i) since IR(φ, V'). ∎

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

$\mathrm{F}_{\mathrm{CTL}}(\varphi, \{p\}) \equiv r \wedge (f \vee m \vee q) \wedge \mathrm{AF}(f \vee m) \wedge$
$(f \vee m \vee (q \wedge \mathrm{AX}(f \vee m \vee q))) \wedge \mathrm{AG}((q \wedge \mathrm{AX}(f \vee m \vee q)) \supset \mathrm{AX}(f \vee m \vee (q \wedge \mathrm{AX}(f \vee m \vee q)))))$.

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

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

Related work

Resolution-based satisfiability of CTL

the paper Form 2000 to 2009, include LTL and CTL.

Using Resolution Computing forgetting

in Propositional Logic, In Model Logic.

Conclusion and Future Work

References

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

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

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.; 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.

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.

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

Fang, L.; Liu, Y.; and Van Ditmarsch, H. 2019. Forgetting in multi-agent modal logics. *Artificial Intelligence* 266:51–80.

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. *Annuals 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.

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 ctl. Technical report, Citeseer.

Zhang, L.; Hustadt, U.; and Dixon, C. 2009. A refined resolution calculus for ctl. In *International Conference on Automated Deduction*, 245–260. Springer.

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