

# A Prime Implicates-based Formulae Forgetting

Dai Xu and Zuoquan Lin  
Department of Information Science, Peking University  
Beijing 100871, P. R. China  
Email: {xudai,lzq}@is.pku.edu.cn

**Abstract**—This paper proposes a novel operation called formulae forgetting by extending literals forgetting, which is popular in reasoning about knowledge, to ignore or hide those irrelevant formulae for given tasks. First, prime implicates are employed to capture variables forgetting and literals forgetting; and literals forgetting could be taken as particular cases of formulae forgetting. Clauses forgetting are then introduced to build a uniform framework for variables forgetting via prime implicates. Based on this uniform framework, a formula could be forgotten by using clauses forgetting on the resolution closure of all prime implicates of the formula. As an extension of variables forgetting, formulae forgetting might be more flexible in reasoning about knowledge since formulae are the most general forms of knowledge. In technology, our operator for formulae forgetting is based on prime implicates and clauses forgetting while variables forgetting is simply based on variables assignment. In addition, our operator could keep the syntax-independency which is an important semantic feature in representing knowledge.

**Keywords**—variables forgetting, literals forgetting, clauses forgetting, formula forgetting, prime implicate

## I. INTRODUCTION

As a useful operation in reasoning about knowledge, forgetting, firstly presented in [1], has attracted much attention in artificial intelligence (AI). The key idea of forgetting is ignoring or hiding those information that is irrelevant to given tasks in order to contract or narrow the search scopes of tasks with maintaining semantic soundness and completeness ([1], [3]). Forgetting also is widely applied in many other research areas such as resolving inconsistencies ([4]), logic programming ([6]), semantic web ([5]) and propositional independence ([2]). Forgetting are roughly classified three types, namely, variables forgetting, literals forgetting ([2], [7]) and formulae forgetting. Indeed, three types could be taken as three stages of studying forgetting since formulae are more expressive than literals and literals are more expressive than variables. That is to say, the basic idea of variables forgetting is planted to both literals forgetting and formulae forgetting.

However, variables forgetting is variable-oriented so that all prime implicates containing this variable are discarded. Taken in this sense, variables forgetting seems too rough. For instance, there is some information about **object A and B**. Formulae built from variables set  $V_1$  and  $V_2$  describe them respectively. If we forget  $V_1$ , then we know nothing about A. But in most cases, when we forget some information about A, we still know something else about it, especially its interaction with B. This example enlightens that a novel operation might be proposed to extend forgetting in formulae called *formulae forgetting*. Notice that there are often some one-se questions in a standardized examination. It means that only one item among several choices (usually four choices) is the right answer. Let

$a, b, c, d$  be four propositions representing four choices. Let  $\phi$  be a proposition that only one of four choices is right.  $\phi$  can be formalized as follows:  $\phi = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge d)$ . However, some persons might ignore this fact that these questions are one-choice so that they could not always make right judgements. Let  $\psi$  be a proposition that there does not exist two of four choices which are right.  $\psi$  can be formalized as follows:  $\psi = \neg((a \wedge b) \vee (a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d) \vee (c \wedge d))$ . We use  $ForgetFor(\phi, \psi)$  to denote an operation that  $\psi$  is relaxed or forgotten in  $\phi$ . Thus,  $ForgetFor(\phi, \psi) = a \vee b \vee c \vee d$ . That is to say, they **admit at least one choice is the right** among four. When two(or more) choices are indeterminate, they usually make mistake.

Before starting defining *ForgetFor* as a new operation to capture a fine consideration compared with current forgetting, we argue that *ForgetFor* should be provided with three features as follows: The first is that *ForgetFor* could be taken as a natural extension of *ForgetLit* which is used to denote literals forgetting in technology. The second is that the result of *ForgetFor* should be intuitive. The last is that *ForgetFor* should be syntax-independent, i.e., it does not depend on the form of formulae. However, it is not trial or not direct that *ForgetFor* is built from *ForgetLit* in technology. To smooth the technical difficulty, we fall back on norm forms of formulae.

In [1], the relationship between forgetting and prime implicates is investigated and we find that variables forgetting is, indeed, deleting prime implicates which include variables to be forgotten in prime implicates set (PI). The remaining clauses are the prime implicates of the result of forgetting formula. Notice that literals forgetting follows the similar principles. Prime implicates have a lot of applications ([9], [10]) and their computation has been explored further ([12], [11]). As a basic technique, the relationship between forgetting and prime implicates is used to define formulae forgetting. The operator is difficult, even for forgetting a clause (which is a special case of formula). Instead of using prime implicates set which does not fit for clauses, we consider its resolution closure *CPI* (presented later). When we forget a clause, we must discover clauses which are relevant to it in *CPI* and then delete them. It is reasonable that two clauses are relevant if one subsumes the other. Forgetting variables or literals is simple because we only remove the clauses including forgotten variables or literals in set *PI*. The rest can not resolve the deleted clauses. But it is not enough that we only delete clauses including or included by the forgotten clause because there may be other clauses that can resolve them. So we keep deleting.

In [8], they proposed to consider forgetting an arbitrary sentence in section conclusion and make it as their future work. We think that keeping the relation between forgetting and update is not necessary good for formula. So we attempt to define formula forgetting based on prime implicates.

## II. PRELIMINARIES

In this section, we briefly represent background knowledge of propositional language, variables forgetting and literals forgetting.

### A. Notation

In this paper, we only consider the language  $PS$  over a finite set of variables  $V$  in propositional logic. A literal is either an atom symbol or its negation. A clause is the disjunction of literals and a term is the conjunction of literals. Clause is also regarded as set of its composed literals.  $|C|$  represents the number of literals in  $C$ . We use  $\phi, \psi$  denoting the formulae in  $PS$ ,  $C$  denoting a clause,  $t$  denoting a term, and  $\omega$  denoting a model. we'll also use the subscripted symbols just mentioned.  $\top$  and  $\perp$  denote respectively true and false formulae. A clause  $C'$  subsumes a clause  $C$  if literals of  $C$  are all included in  $C'$ , namely,  $C \models C'$ . A model is a possible world that the truth value of every atom in  $PS$  is known.  $Mod(\phi)$  denotes the set of models for  $\phi$ .  $Var(\phi)$ ,  $Lit(\phi)$  respectively represent set of variables and literals in formula  $\phi$ .

For any set  $X$  of formulae,  $\wedge X$  (or  $\vee X$ ) denotes conjunction (or disjunction) of all the formulae in  $X$ . In this paper, we usually consider set  $X$  of clauses.  $\wedge X$  (or  $\vee X$ ) is a formula which is equivalent to conjunction (or disjunction) of clauses in  $X$ .

$T$  is a theory iff  $T$  is set of formulae and closed under classical logic deduction.  $Cn(T)$  denotes the closure.

$\phi$  is a formula in  $PS$ , a clause  $C$  is an implicate of  $\phi$  iff  $\phi \models C$ .

The *prime implicate* of a formula  $\phi$  is a clause  $C$  that satisfies the following conditions:

- $C$  is an implicate of  $\phi$ .
- There is no other implicate  $C'$  of  $\phi$  satisfies  $C' \models C$ .

The prime implicate is the strongest necessary condition clause by the definition. Let  $PI(\phi)$  denotes the set of prime implicates for  $\phi$ , then we have  $\phi \equiv \wedge PI(\phi)$ . For any two clauses  $C_1, C_2$  in  $PI$ , they don't infer each other, i.e.,  $C_1 \not\models C_2$  and  $C_2 \not\models C_1$ .

### B. Variables Forgetting and Literals Forgetting

$ForgetVar$ ,  $ForgetLit$  stand for variables forgetting and literals forgetting.

$\phi$  is a propositional formula in  $PS$ ,  $p$  is an atom and  $V \subset Var(PS)$ ,  $ForgetVar(\phi, V)$  denotes forgetting set of variables  $V$  in formula  $\phi$  which can be defined recursively as follows:

- $ForgetVar(\phi, \emptyset) \equiv \phi$ .
- $ForgetVar(\phi, \{p\}) \equiv \phi_{p \leftarrow 0} \vee \phi_{p \leftarrow 1}$ .
- $ForgetVar(\phi, V \cup \{p\}) \equiv ForgetVar(\phi, V)_{p \leftarrow 0} \vee ForgetVar(\phi, V)_{p \leftarrow 1}$

For any formula  $\phi$  and set of atoms  $V$ , we have  $ForgetVar(\phi, V) \equiv \wedge \{C \text{ is a clause} \mid C \in PI(\phi) \text{ and } Var(C) \cap V = \emptyset\}$ .

We get formula after variables forgetting by remove the clauses which include some forgotten variable in prime implicates set of original formula. Also the remaining set is the prime implicates set for the formula composed of them. Let  $S = \{C \text{ is a clause} \mid C \in PI(\phi) \text{ and } Var(C) \cap V = \emptyset\}$ , every clause in  $S$  is prime implicate for  $ForgetVar(\phi, V)$  because  $\phi \models ForgetVar(\phi, V)$ . Then we prove  $S$  contains all prime implicates of  $ForgetVar(\phi, V)$ . Let  $C$  is prime implicate of  $ForgetVar(\phi, V)$ ,  $C$  must not include variables from  $V$  and  $\phi \models C$ . There must be a clause  $C'$  in  $PI(\phi)$  which variables in  $V$  don't occur in satisfying  $C' \models C$ . So  $C' \in S$  and  $C' = C$ . Thus  $PI(\wedge S) = S$ . We not only obtain forgetting formula but also its prime implicates set by Theorem 1.

Literal forgetting are given in [2] and has the similar relation with PI.

$\phi$  is a propositional formula in  $PS$ ,  $l$  is a literal and  $L$  is a set of literals. then

- $ForgetLit(\phi, \emptyset) \equiv \phi$ .
- $ForgetLit(\phi, \{l\}) \equiv \phi_{l \leftarrow 1} \vee (\neg l \wedge \phi)$ .
- $ForgetLit(\phi, L \cup \{l\}) \equiv ForgetLit(ForgetLit(\phi, L), l)$

For any formula  $\phi$  and set of literals  $L$ , we have  $PI(ForgetLit(\phi, L)) = \{C \text{ is a clause} \mid C \in PI(\phi) \text{ and } Lit(C) \cap L = \emptyset\}$

The theorem above indicates that we get prime implicates set of the resultant formula after eliminating the clauses that involve some forgotten literal, i.e., any prime implicate of  $ForgetLit(\phi, L)$  must not contain some literal from  $L$ . So the resulting formula is equivalent to the conjunction of the remaining set.

As the two theorems show, Forgetting symbols and literals are relevant to prime implicates closely, i.e., remove from prime implicates set of original formula the clauses that contain the forgotten symbols or literals. Our available set is *stable* because it consists of all prime implicates of formula formed by conjunction of the set itself. If  $S$  is such a set of clauses, then  $PI(\wedge S) = S$ . This simple relation motivate us extend forgetting from this direction and we'll show that this extension meets our intuition. As prime implicate is a clause, we'll first consider forgetting clauses, then to forget a general formula.

## III. CLAUSES FORGETTING

### A. Definition

It becomes difficult even if we consider forgetting a clause. As variable or literal forgetting, we should first discard clauses that subsumes the forgotten clause. But it's far from enough. For instance,  $C_1, C_2, C'$  are in the set satisfying  $C'$  is resolvent of  $C_1$  and  $C_2$ , if  $C \models C'$ , then  $C'$  must be deleted and  $C_1, C_2$  shouldn't be simultaneously reserved when we forget  $C$ . We'll give distance between two clauses to decide which one to be put off. However, for existing forgetting, it's enough to remove clauses containing the forgotten variables or literals because the remained ones must not resolute the deleted clauses. On the other hand, we must eliminate the clauses subsumed by

the forgotten ones because we don't remember the forgotten information after forgetting. Likewise, if a subset of clauses resolve  $C'$  and  $C' \models C$ , we must destroy it.

The next task is to give the distance between two clauses. Clauses have the same form and consists of some literals. So the quantity of different literals in two clauses measures it well. There are two extremes. The distance is maximal when two clauses don't share literals, on the other hand, it is minimal when one clause subsumes another.

Let  $C_1, C_2$  be two clauses,  $|C_1 - C_2|$  denotes number of literals which are in  $C_1$  and not in  $C_2$ .  $Dis(C_1, C_2)$  represents the distance between  $C_1$  and  $C_2$ . We define it as:

$$Dis(C_1, C_2) = \frac{|C_1 - C_2| + |C_2 - C_1|}{|C_1| + |C_2|}.$$

The pre-order  $\leq_C$  is defined as:  $C_1 \leq_C C_2$  iff  $Dis(C_1, C') \leq Dis(C_2, C')$ .

It's obvious that  $0 \leq Dis(C_1, C_2) \leq 1$ .  $Dis(C_1, C_2) = 0$  if and only if  $C_1$  subsumes  $C_2$  or vice versa.  $Dis(C_1, C_2) = 1$  iff  $C_1$  and  $C_2$  don't have common literals.

Instead of operating on set PI for forgetting of literals or variables, we consider its resolution closure.

**Definition 1:**  $X$  and  $Y$  are two clause sets,  $X$  is called a resolution closure of  $Y$  iff they satisfy the following conditions:

- (1)  $\forall C \in X$  iff  $C \in Y$  or  $C$  is resolvent of clauses in  $Y$ .
- (2)  $\forall C \in X, C \neq \top$ .

It is obvious that  $Y \subseteq X$ . We denotes **resolution closure of prime implicates set of formula  $\phi$  as  $CPI(\phi)$** . Generally speaking,  $PI(\phi) \neq CPI(\phi)$ .

**Example 1:** Let  $PI(\phi) = \{a \vee b \vee \neg c, c \vee d, a \vee d\}$ . The resolution closure of  $PI(\phi)$  is  $\{a \vee b \vee \neg c, c \vee d, a \vee d, a \vee b \vee d\}$ .  $a \vee b \vee d$  isn't in  $PI(\phi)$  because  $a \vee d$  is in it.

Condition (2) is necessary. Two clauses may resolve  $\top$  which gives us no information, so we must discard it. Let  $Y = \{a \vee b, \neg a \vee \neg b\}$ . These clauses resolve  $\top$  no matter whether resolution is based on variable  $a$  or  $b$ . So resolution closure set of  $Y$  is itself.

We can easily find the related clauses stepwise on CPI instead of PI, and variables or literals forgetting by PI and CPI are logical equivalent.

**Theorem 1:** Let  $\phi$  is a formula,  $V$  is set of variables and  $L$  is set of literals.  $PI_V$  and  $PI_L$  denote respectively the remaining sets after removing all clauses containing symbols in  $V$  and literals in  $L$  from  $PI(\phi)$ .  $CPI_V$  and  $CPI_L$  are similar. Then  $\wedge(PI_V) \equiv \wedge(CPI_V)$  and  $\wedge(PI_L) \equiv \wedge(CPI_L)$ .

*Proof:* . Let  $PI(\phi) = X_1 \cup X_2$ ,  $X_1$  denotes the set of clauses that include literals from  $L$  and the clauses in  $X_2$  don't.  $CPI(\phi) = X_1 \cup X_2 \cup X_3 \cup X_4$ ,  $X_i (i = 1, \dots, 4)$  don't intersect with each other. Clauses in  $X_3 \cup X_4$  are resolvent of subsets of  $PI(\phi)$ . Clauses in  $X_3$  include literals of  $L$  and clauses in  $X_4$  don't. We need to prove  $\wedge X_2 \equiv \wedge(X_2 \cup X_4)$ .  $\forall C \in X_4$ , there must be an clause  $C' \in PI(\phi)$  satisfying  $C' \models C$  because  $\phi \models C$ .  $C' \in X_2$  due that  $C$  doesn't contain literals of  $L$ . So  $\wedge(PI_V) \equiv \wedge(CPI_V)$ . Analogously,  $\wedge(PI_L) \equiv \wedge(CPI_L)$  holds. ■

Now we are ready to give process of deletion for clause forgetting. Suppose we'll forget a clause  $C$  in formula  $\phi$ . According to description above, we consider operation on set  $CPI(\phi)$ . First we delete clause  $C'$  if  $C' \models C$  or  $C \models C'$ . We

put all these clauses  $C'$  into set  $FS_1$ . Second, if  $C' \in FS_1$  is resolvent of two clauses  $C_1, C_2$  in the rest set, then we must delete at least one of  $C_1$  and  $C_2$  according to pre-order  $\leq_C$ . We don't need to consider  $C' \in S_1$  is resolvent of more than two clauses because these resolution relations can be destroyed later or by other deletion. In this step, we put clauses selected into  $FS_2$  to delete. Subsequently, we continue find two clauses resolve  $C''$  for every  $C'' \in FS_2$  and then put into  $FS_3$ . Because  $CPI(\phi)$  is finite, there must be an integer  $n$  satisfying  $FS_{n+1} = \emptyset$ . So  $FS = FS_1 \cup FS_2 \dots \cup FS_n$  is set of all clauses which are going to be discarded.

**Definition 2:** The *forgetting set* for a clause  $C$  in  $CPI(\phi)$  is  $FS(\phi, C)$  which can be defined recursively until it is stable:  $FS_1 = \{B | B \in CPI(\phi), Dis(B, C) = 0\}$   
 $FS_{i+1} = \cup_{B \in FS_i} \{C' | \exists C_1, C_2 \in (CPI(\phi) - \cup_{k=1}^i FS_k), B \text{ is resolvent of } C_1, C_2, C' \text{ is the less one of } \{C_1, C_2\} \text{ for } \leq_C\}$   
 $i = 1, 2, \dots$

If  $FS_{n+1} = \emptyset$  for some integer  $n$ , then  $FS(\phi, C) = \cup_{i=1}^n FS_i$ .

If  $FS_1 = \emptyset$  then  $FS(\phi, C) = \emptyset$ . Because we remove at least one clause at each step and  $CPI(\phi)$  is finite, the process terminates at most  $|CPI(\phi)|$  step. Since we have the notion of forgetting set, forgetting a clause is natural.

**Definition 3:** For any formula  $\phi$  and a clause  $C$ , Forgetting a clause is denoted by  **$ForgetCla(\phi, C)$** , then  $ForgetCla(\phi, C) = \wedge(CPI(\phi) - FS(\phi, C))$ .

**Example 2:** Let  $\phi = a \wedge b \wedge c$ ,  $C = a \vee b$ , then  $CPI(\phi) = PI(\phi) = \{a, b, c\}$ . Because  $a \models C, b \models C$ ,  $forgetCla(\phi, C) = c$ .

**Example 3:** Let  $\phi = a \vee b \vee c$ ,  $C = a \vee b$ , then  $CPI(\phi) = PI(\phi) = \{a \vee b \vee c\}$ , because  $C \models a \vee b \vee c$ , we have  $ForgetCla(\phi, C) = \top$ .

We can regard  $a \vee b$  as entirety in formula  $a \vee b \vee c$  and then apply literal forgetting. The result is also  $\top$ .

## B. Properties

$ForgetCla(\phi, C) = \wedge(CPI(\phi) - FS(\phi, C))$  holds by definition. In fact, set  $CPI(\phi) - FS(\phi, C)$  subsumes all prime implicates of  $ForgetCla(\phi, C)$ .

**Theorem 2:** For any formula  $\phi$  and clause  $C$ ,  $PI(ForgetCla(\phi, C)) \subseteq CPI(\phi) - FS(\phi, C)$ .

*Proof:* Let  $X = CPI(\phi) - FS(\phi, C)$ , we first point set  $X$  is resolution closed.  $\forall C_1, C_2 \in X$ , if  $C'$  is a resolvent of  $C_1, C_2$ , then  $C'$  must be in  $X$ . Otherwise, according to definition, one of  $C_1$  and  $C_2$  mustn't be in  $X$ . It is impossible. Let's consider resolvent of any subset of  $X$ .  $\forall Y \subseteq X$ , the resolvent of  $Y$  can be gained by multi steps resolution and so it is in  $X$ . Next we'll prove  $\forall C' \in PI(ForgetCla(\phi, C))$ , it holds that  $C' \in CPI(\phi) - FS(\phi, C)$ . If there is  $C'' \in CPI(\phi) - FS(\phi, C)$  satisfies  $C'' \models C'$  then  $C'' = C'$  because  $C'$  is prime implicate of  $ForgetCla(\phi, C)$ . Else if there is subset  $X' \subseteq X$  satisfies  $X' \models C'$ .  $C'$  must be a resolution result of  $X'$  (if  $C'$  isn't a resolution, then there must be a resolution  $C''$  of  $X'$  satisfying  $C'' \models C'$ . It's impossible because  $C'$  is a prime implicate.) and  $C' \in X$  because  $X$  is resolution closed. ■

The next theorem shows the nature of forgetting.

**Theorem 3:** For any formula  $\phi$  and clause  $C$ ,  $ForgetCla(\phi, C) \not\models C$  and  $C \not\models ForgetCla(\phi, C)$ .

*Proof:* . If  $\text{ForgetCla}(\phi, C) \models C$ , then there is  $C' \in \text{PI}(\text{ForgetCla}(\phi, C))$  satisfying  $C' \models C$ . Of course it may be  $C = C'$ . It holds that  $C' \in \text{CPI}(\phi) - \text{FS}(\phi, C)$  by the theorem above. But  $C' \in \text{FS}(\phi, C)$  because  $\text{Dis}(C', C) = 0$ . It's impossible. We put every clause  $C'$  satisfying  $C \models C'$  into set  $\text{FS}(\phi, C)$ . So it's obvious that  $C \not\models \text{ForgetCla}(\phi, C)$ . ■

A literal can be treated as a special clause. theorem 1 shows that clause forgetting include literal forgetting.

**Corollary 1:** For any formula  $\phi$  and literal  $l$ ,  $\text{ForgetCla}(\phi, l) \equiv \text{ForgetLit}(\phi, l)$ .

We forget some clause by removing some clauses in  $\text{CPI}$  of the original formula. The following theorem shows that the resulting formula is weaker than the original one. It's intuitive that we know less information if we forget something.

**Theorem 4:**  $\phi \models \text{ForgetCla}(\phi, C)$

We define  $\text{ForgetCla}(\phi, C)$  by discarding some clauses in  $\text{CPI}(\phi)$  and  $\phi \equiv \wedge \text{CPI}(\phi)$ . So  $\text{ForgetCla}(\phi, C)$  is weaker than  $\phi$ .

Forgetting something might not work in some special case.


**Theorem 5:** If  $\text{Lit}(C) \cap \text{Lit}(\phi) = \emptyset$ , then  $\phi \equiv \text{ForgetCla}(\phi, C)$ .

*Proof:*  $\forall C' \in \text{CPI}(\phi)$ , we have  $\text{Lit}(C') \subseteq \text{Lit}(\phi)$ . And we get  $\text{Lit}(C) \cap \text{Lit}(C') = \emptyset$  according to the condition. So  $\text{Dis}(C, C') = 1$ . Each clause in  $\text{CPI}(\phi)$  should be kept, i.e.,  $\phi \equiv \text{ForgetCla}(\phi, C)$ . ■

But when  $\text{Lit}(\phi) \cap \text{Lit}(C) \neq \emptyset$ , it might hold too. Let  $\text{CPI}(\phi) = \{a \vee b, \neg b \vee c, a \vee c\}$ ,  $\psi = \neg a \vee b$ , then  $\text{ForgetCla}(\phi, C) \equiv \phi$ .

The  $\text{ForgetCla}$  operator is based on  $\text{CPI}$  of formula. All equivalent formulae have the same  $\text{PI}$  and  $\text{CPI}$ . The theorem below is a natural conclusion.

**Theorem 6:** If  $\phi_1 \equiv \phi_2$ , then  $\text{ForgetCla}(\phi_1, C) \equiv \text{ForgetCla}(\phi_2, C)$ .

After we forget  we know nothing about it. It remains when we continue forgetting the same content. The forgetting operator reach a stable state.

**Theorem 7:**  $\text{ForgetCla}(\text{ForgetCla}(\phi, C), C) \equiv \text{ForgetCla}(\phi, C)$ .

Our forgetting is not to forget each literal in forgotten clause. It's stronger than the latter. We concentrate on level of clauses, not literals.

**Theorem 8:**  $\text{ForgetCla}(\phi, C) \models \text{ForgetCla}(\phi, \text{Lit}(C))$

*Proof:* We only need to prove each clause in  $\text{FS}(\phi, C)$  in Definition 4 subsumes literals in  $\text{Lit}(C)$ . If  $\text{FS}(\phi, C) = \emptyset$ , obviously it holds. Otherwise, let  $C'$  in  $\text{FS}_i(i > 1)$  be the first clause not subsuming these literals. So  $\text{Dis}(C', C) = 1$  holds, but  $\forall C'' \in \text{FS}_k(k < i)$ ,  $\text{Dis}(C'', C) < 1$ . Then  $\exists C'' \in \text{FS}_{i-1}$  and  $C'_1$ ,  $C''$  is the resolvent of  $C'$  and  $C'_1$ . it holds  $\text{Dis}(C'_1, C) = 1$ . So  $C''$  doesn't subsume literals in  $\text{Lit}(C)$ . It's impossible. ■

#### IV. FORMULA FORGETTING

We need to define **clauses set forgetting** in order to define **formula forgetting**. But forgetting clauses can't be defined recursively because that it's sensitive to the order of clauses. For example, Let  $\text{CPI}(\phi) = \{a \vee b \vee c, a \vee b \vee \neg d, c \vee d\}$ , then  $\text{ForgetCla}(\text{ForgetCla}(\phi, a \vee b \vee c), c \vee d) = \top$ ,  $\text{ForgetCla}(\text{ForgetCla}(\phi, c \vee d), a \vee b \vee c) = a \vee b \vee \neg d$ .

This is due that the forgotten clauses can affect each other. Forgetting set  $L$  of literals is equivalent to discard clauses subsuming literals in  $L$ . When we forget several clauses, it's intuitive to remove clauses that correlate with some of them.

**Definition 4:** Let  $\phi$  be a formula,  $S$  be a set of clauses,  $C$  be a clause, then we can define **clauses set forgetting** as follows:  $\text{ForgetClas}(\phi, S) \equiv \bigwedge \bigcap_{C \in S} (\text{CPI}(\phi) - \text{FS}(\phi, C))$ .

Corollary 1 can be extended to set of literals.

**Corollary 2:** For any set of literals  $L$  and formula  $\phi$ ,  $\text{ForgetClas}(\phi, L) \equiv \text{ForgetLit}(\phi, L)$ .

Every prime implicate of formula is a clause and the formula is equivalent to the conjunction of its prime implicates set. We define forgetting of formula by forgetting its prime implicates set.

**Definition 5:** For any two formulae  $\phi, \psi$ , we use  $\text{ForgetFor}(\phi, \psi)$  as forgetting formula  $\psi$  in  $\phi$ , then  $\text{ForgetFor}(\phi, \psi) = \text{ForgetClas}(\phi, \text{PI}(\psi))$ .

**Example 4:** Let  $\phi = a \wedge b \wedge c$ ,  $\psi = a \wedge b$ , then we have  $\text{ForgetFor}(\phi, \psi) = c$ .

We again consider an example in introduction.  $\phi = (a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge c \wedge \neg d) \vee (\neg a \wedge \neg b \wedge c \wedge d)$ , then  $\text{CPI}(\phi) = \{a \vee b \vee c \vee d, \neg a \vee \neg b, \neg a \vee \neg c, \neg a \vee \neg d, \neg b \vee \neg c, \neg b \vee \neg d, \neg c \vee \neg d\}$ .  $\text{PI}(\psi) = \{\neg a \vee \neg b, \neg a \vee \neg c, \neg a \vee \neg d, \neg b \vee \neg c, \neg b \vee \neg d, \neg c \vee \neg d\}$ . So we get  $\text{ForgetFor}(\phi, \psi) = a \vee b \vee c \vee d$ .

Of course, clause forgetting is a special case of formula forgetting.

**Theorem 9:** For any formula  $\phi$  and clause  $C$ ,  $\text{ForgetFor}(\phi, C) = \text{ForgetCla}(\phi, C)$ .

Sometimes we might regard the forgotten formula as a whole so as to apply literal forgetting.

**Theorem 10:** For any two formula  $\phi, \psi$ ,  $\phi(t/\psi)$  denotes the formula replacing every occurrence of  $\psi$  by  $t$ , if  $\phi$  can be defined by  $\psi$  and some symbols not in  $\text{Var}(\psi)$ , then  $\text{ForgetFor}(\phi, \psi) = \text{ForgetLit}(\phi(t/\psi), t)$ .

*Proof:* Let  $\psi = t$ ,  $\text{PI}(\phi(t/\psi)) = S_1 \cup S_2 \cup S_3 = \{t \vee C_1, \dots, t \vee C_m, \neg t \vee C'_1, \dots, \neg t \vee C'_n, C''_1, \dots, C''_k\}$ .  $S_1$  denotes the set of clauses including literal  $t$ ,  $S_2$  denotes the set of clauses including literal  $\neg t$ ,  $S_3$  denotes the set of rest clauses. Let  $\text{PI}(\psi) = \{D_1, \dots, D_s\}$  and  $\text{PI}(\neg\psi) = \{D'_1, \dots, D'_t\}$ . We replace each  $t$  in  $S_1$  with  $\wedge \text{PI}(\psi)$  and  $\neg t$  in  $S_2$  with  $\wedge \text{PI}(\neg\psi)$ , then we spread form of clauses.  $S_1$  is supposed to be  $S'_1 = \{D_1 \vee C_1, \dots, D_s \vee C_1, \dots, D_1 \vee C_m, \dots, D_s \vee C_m\}$ ,  $S_2$  be  $S'_2 = \{D'_1 \vee C'_1, \dots, D'_t \vee C'_1, \dots, D'_1 \vee C'_n, \dots, D'_t \vee C'_n\}$ . Now we first prove  $\text{PI}(\phi) = S'_1 \cup S'_2 \cup S_3$ . On the one hand, for any implicate  $C$  of  $\phi$ ,  $\phi(t/\psi) \models C$ . Then we have  $\exists C' \in \text{PI}(\phi(t/\psi))$  satisfies  $C' \models C$ . If  $C' = t \vee C_i$ , ( $1 \leq i \leq m$ ), then  $C$  can be divided in two clauses  $C_1$  and  $C_2$  satisfying  $t \models C_1$ ,  $C_i \models C_2$ . So there must be  $D_j$ , ( $1 \leq j \leq s$ ) satisfying  $D_j \models C_1$  and  $D_j \vee C_i \in S'_1$ . The proof is same for  $C' \in S_2$ . If  $C' \in S_3$ , then  $C' \in \text{PI}(\phi)$ . On the other hand, we need to prove  $\forall C_0, C'_0 \in S'_1 \cup S'_2 \cup S_3$ ,  $C_0 \not\models C'_0$  and  $C'_0 \not\models C_0$ . If  $C_0 \in S'_1, C'_0 \in S'_i (i = 1, 2, 3)$  or  $C_0 \in S'_1, C'_0 \in S_3$  or  $C_0 \in S'_2, C'_0 \in S_3$ , it holds because any one clause can't subsume another one in prime implicates set  $\text{PI}(\phi(t/\psi))$ . Let's investigate last case:  $C_0 \in S'_1, C'_0 \in S'_2$ . If  $D_i \in \text{PI}(\psi), D'_j \in \text{PI}(\neg\psi)$ , then one of  $D_i$  and  $D'_j$  doesn't subsume the other. Otherwise, suppose  $D_i \models D'_j$ . We

have  $\psi \models D'_j$  and  $\neg\psi \models D'_j$ . then  $\psi \vee \neg\psi \models D'_j$ . That is,  $D'_j \equiv \top$ . This is impossible. So the property holds also for last condition. thus we proved  $PI(\phi) = S'_1 \cup S'_2 \cup S_3$ .

Next we compute  $CPI(\phi)$  which is closure of  $PI(\phi)$ . We call clauses in  $\{C_1, \dots, C_m, C'_1, \dots, C'_n, C''_1, \dots, C''_k\}$   $C$  - clauses and clauses in  $\{D_1, \dots, D_s, D'_1, \dots, D'_t\}$   $D$  - clauses. Then  $C$  - clauses and  $D$  - clauses don't share variables. We get  $CPI(\phi)$  by  $C$  - clauses or  $D$  - clauses resolutions. The new generated clauses by resolutions are clarified into 7 types:

- (1) generated by resolutions of clauses in  $S'_1$ . We denote set of these clauses as  $S_{1,1}$ .
- (2) generated from  $S'_2$ . This kinds of set is  $S_{2,2}$ .
- (3) generated from  $S_3$ . The set is  $S_{3,3}$ .
- (4) generated from  $S'_1$  and  $S'_2$ . The set is  $S_{1,2}$ .
- (5) generated from  $S'_1$  and  $S_3$ . The set is  $S_{1,3}$ .
- (6) generated from  $S'_2$  and  $S_3$ . The set is  $S_{2,3}$ .
- (7) generated from  $S'_1, S'_2$  and  $S_3$ . The set is  $S_{1,2,3}$ .

We perform our forgetting operator aiming at 7 sets above plus 3 sets  $S'_1, S'_2, S_3$ . We first compute  $FS_1$ .  $S'_1 \subseteq FS_1$  because each clause in  $S'_1$  includes some clause in  $PI(\psi)$  as its part. It's also easy to see that clauses in  $S'_2$  and  $S_3$  don't belong to  $FS_1$ . We get clauses in  $S_{1,1}$  by  $C$  - clauses resolutions and  $D$  - clauses resolutions. Each of them subsumes some clause in  $PI(\psi)$ . So  $S_{1,1} \subseteq FS_1$ .  $S_{2,2}$  and  $S_{3,3}$  don't intersect with  $FS_1$ . Clauses in these two sets are all entailed by  $\wedge(S'_2 \cup S_3)$ . Now we'll prove every  $C$  - clauses resolves  $\top$  For case (4). For any  $1 \leq i \leq l$  and  $1 \leq j \leq s$ ,  $\neg D_i \models \neg\psi$  because  $\psi \models D_i$  and  $\neg D'_j \models \psi$  because  $\neg\psi \models D'_j$ .  $\neg D_i \wedge \neg D'_j \models \psi \wedge \neg\psi$ . So  $\neg D_i \wedge \neg D'_j \equiv \perp$ . Then we prove  $D_i \vee D'_j \equiv \top$ . So  $D$  - clauses resolutions participate in producing clauses in  $S_{1,2}$ . Thus each clause in  $S_{1,2}$  subsume some clause in  $S_3$  because resolvent of two clauses in  $S_1$  and  $S_2$  is some clause in  $S_3$  or subsume it. Some clauses of them may be discarded. For (5) and (6),  $S_{1,3} \subseteq FS_1$  and  $S_{2,3} \wedge FS_1 = \emptyset$ . (7) is similar to (4),  $D$  - clauses resolutions of  $S'_1$  and  $S'_2$  must occur. These clauses are all entailed by clauses in  $S_3$ . Every clause in  $FS_1$  is either in  $S'_1$  or resolved from clauses in  $S'_1$  and other sets. The remained set  $S$  don't resolve discarded clauses and they are all entailed by clauses in  $\{S'_2 \cup S_3\}$ . So  $FS_2 = \emptyset$ . And  $\wedge S \equiv \wedge(S'_2 \cup S_3) \equiv ForgetLit(\phi(t/\psi), t)$ .

There are two special cases. One is  $PI(\phi(t/\psi)) = \{t, \neg t \vee C'_1, \dots, \neg t \vee C'_n, C''_1, \dots, C''_k\}$ . The other is  $PI(\phi(t/\psi)) = \{t \vee C_1, \dots, t \vee C_m, \neg t, C''_1, \dots, C''_k\}$ . These are all easy to prove. ■

For a non-trivial term  $t = l_1 \wedge l_2 \wedge \dots \wedge l_n$  and formula  $\phi$ , we have  $ForgetFor(\phi, t) = ForgetLit(\phi, \{l_1, \dots, l_n\})$ . In general case,  $ForgetFor(\phi, \psi_1 \wedge \psi_2) \neq ForgetFor(\phi, \{\psi_1, \psi_2\})$ . For instance,  $CPI(\phi) = \{a \vee b \neg c, c \vee d, a \vee b \vee d\}$ ,  $CPI(\psi_1) = \{a \vee b \vee \neg c\}$ ,  $CPI(\psi_2) = \{c \vee d\}$ . So  $CPI(\psi_1 \wedge \psi_2) = CPI(\phi)$ .  $ForgetFor(\phi, \psi_1) = \{c \vee d, a \vee b \vee d\}$  and  $ForgetFor(\phi, \psi_2) = \{a \vee b \vee \neg c, a \vee b \vee d\}$ . So  $ForgetFor(\phi, \{\psi_1, \psi_2\}) = a \vee b \vee d$ . But  $ForgetFor(\phi, \psi_1 \wedge \psi_2) = \top$ .

**Example 5:**  $\phi = (\neg\psi \wedge b) \vee (\psi \wedge c)$ ,  $\psi = d \wedge e$ . Then  $\phi = [(\neg d \vee \neg e) \wedge b] \vee (c \wedge d \wedge e)$ ,  $PI(\phi) = \{c \vee \neg d \vee \neg e, b \vee c, b \vee d, b \vee e\}$ . So  $ForgetFor(\phi, \psi) = \wedge\{c \vee \neg d \vee \neg e, b \vee c\}$ .

And  $ForgetCla(\phi(t/\psi), t) = \wedge\{\neg t \vee c, b \vee c\} = \wedge\{c \vee \neg d \vee \neg e, b \vee c\}$ .

Formula forgetting satisfies all properties for clause forgetting. Theorem 6 can be reinforced: If  $\phi_1 \equiv \phi_2$  and  $\psi_1 \equiv \psi_2$  then  $Forget(\phi_1, \psi_1) \equiv Forget(\phi_2, \psi_2)$ . And we have the following triviality in extreme cases:

**Theorem 11:** (triviality)

$$\begin{aligned} ForgetFormula(\phi, \phi) &= \top \\ ForgetFormula(\top, \psi) &= \top \\ ForgetFormula(\perp, \psi) &= \perp \\ ForgetFormula(\phi, \perp) &= \phi \end{aligned}$$

*Proof:* We only prove the first because others are easier.  $ForgetFormula(\phi, \phi) = ForgetCla(\phi, PI(\phi))$ . It's enough to prove  $\forall C \in CPI(\phi), \exists C' \in PI(\phi)$ , we have  $Dis(C', C) = 0$ . If  $C \in PI(\phi)$ , take  $C' = C$ ; else there must be  $C'' \in PI(\phi)$  satisfying  $C'' \models C$ . So  $Dis(C'', C) = 0$ . ■

## V. CONCLUSION

This paper proposed a uniform framework for literals forgetting by using prime implicates. Then we employed this framework to define clauses forgetting and formulae forgetting. Compared with variables forgetting rough treatment, formulae forgetting is formula-oriented so that our forgetting operation could treat knowledge more carefully and exquisite. we will generalize the forgetting approach and possibly arouse some interesting topics to be researched in the future. We will consider our future works from two aspects. One is to study their application, especially in handling inconsistencies; and the other is to give its semantic features, that is, what is the relationship between operation on CPI and changes in models of formulae.

## ACKNOWLEDGMENT

This work is supported by NSFC under grant number 60973003, 60496322.

## REFERENCES

- [1] Fangzhen Lin and Reiter, R., *Forget it*, Working Notes of AAAI Fall Symposium on Relevance, 1994.
- [2] Jérôme Lang and Paolo Liberatore and Pierre Marquis, *Propositional Independence: Formula-Variable Independence and Forgetting*, J. Artif. Intell. Res. (JAIR), 391-443, 2003.
- [3] Kaile Su and Guanfeng Lv and Yan Zhang, *Reasoning about Knowledge by Variable Forgetting*, KR, 576-586, 2004.
- [4] Jérôme Lang and Pierre Marquis, *Resolving Inconsistencies by Variable Forgetting*, 239-250, KR, 2002.
- [5] Zhe Wang and Kewen Wang and Rodney W. Topor and Jeff Z. Pan, *Forgetting Concepts in DL-Lite*, ESWC, 245-257, 2008.
- [6] Yan Zhang and Norman Y. Foo, *Solving logic program conflict through strong and weak forgettings*, 739-778, Vol. 170, Artif. Intell., 2006.
- [7] Yves Moinard, *Forgetting Literals with Varying Propositional Symbols*, 955-982, Vol.17, J. Log. Comput., 2007.
- [8] Abhaya C. Nayak and Yin Chen and Fangzhen Lin, *Forgetting and Knowledge Update*, 131-140, Australian Conference on Artificial Intelligence, 2006.
- [9] Marquis, P., *Knowledge compilation using theory prime implicates*, International Joint Conference on Artificial Intelligence, 837-845, Vol.14, 1995.
- [10] Rajaratnam, D. and Pagnucco, M., *Prime Implicates for Approximate Reasoning*, Lecture Notes in Computer Science, Springer, Vol.4798, 2007.
- [11] Ramesh, A. and Becker, G. and Murray, N.V., *On Computing Prime Implicates and Prime Implicates*, Tableaux, 73-75, 1992.
- [12] Kean, A. and Tsiknis, G., *An incremental method for generating prime implicates/implicates*, Journal of Symbolic Computation, 185-206, Vol.9, Academic Press, 1990.