About the computation of forgetting symbols and literals

Yves Moinard
INRIA/IRISA, Campus de Beaulieu, 35042 RENNES-Cedex FRANCE
moinard@irisa.fr

Abstract

Recently, the old logical notion of forgetting propositional symbols (or reducing the logical vocabulary) has been generalized to a new notion: forgetting literals. The aim was to help the automatic computation of various formalisms which are currently used in knowledge representation, particularly for nonmonotonic reasoning. We develop here a further generalization, allowing propositional symbols to vary while forgetting literals. We describe the new notion, on the syntactical and the semantical side. We provide various manipulations over the basic definitions involved, including for the original version, which hopefully should help improving again the efficiency of the computation. This work concerns especially circumscription, since it is known that one way of computing circumscription uses the forgetting of literals.

Introduction

The well-known notion of forgetting propositional symbols, which is known at least since a 1854 paper by Boole under the name "elimination of middle terms", has been used for a long time in mathematical logic and in its applications for knowledge representation (see e.g. (Lin & Reiter 1994; Lin 2001; Su, Lv, & Zhang 2004)). It is a reduction of the vocabulary, thanks to the suppression of some propositional symbols. Let us consider the formula

$$(bird \wedge \neg exceptional \rightarrow flies) \wedge \neg exceptional.$$

We may want to "forget" the symbol $exceptional$, considered here as "auxiliary", then we get the formula

$$bird \rightarrow flies.$$

Recently, Lang et al. (Lang, Liberatore, & Marquis 2003) have extended this notion in a significant manner, by allowing the forgetting of *literals*. In above example, it happens that in fact what has been done is equivalent to forgetting the literal $\neg exceptional$. In the general case, forgetting a literal is more precise than forgetting its propositional symbol: we get a formula standing "somewhere between" the original formula and the formula obtained by forgetting the propositional symbol.

This new definition is a natural extension of the classical definition of forgetting propositional symbols. Lang et al. have shown that this new notion also is useful for knowledge representation and particularly for nonmonotonic reasoning. In some cases, this provides a simplification of the computations, and the authors provide various ways for computing the forgetting of literals, in order to obtain concrete examples of simplification of the computation of some already known formalism.

We extend the notion by allowing some propositional symbols to *vary* when forgetting literals. The new definitions are a simple and natural extension of the original ones, and they have the same kind of behavior

We describe various ways for computing these notions (including the original ones, without varying symbols), and we provide hints showing that the complexity of the new notion should be comparable to the complexity of the notion without variable symbols. This is of some importance in order to apply the results given in (Lang, Liberatore, & Marquis 2003) to the new notion, since this should simplify significantly the overall computation. The main application example of the interest of these methods for computing already known formalisms given in (Lang, Liberatore, & Marquis 2003) concerns circumscription, and (Moinard 2005) has shown how the new notion with varying symbols allows to reduce a two stage method to a single stage one.

Firstly, we give the preliminary notations and definitions. Then we remind the notion of propositional symbol forgetting, with a few more technical tools. Then we remind the notion of literal forgetting as introduced by Lang et al. Then we introduce our generalization, allowing symbols to vary when literals are forgotten. Finally, we detail yet another method for computing these notions.

Technical preliminaries

We work in a propositional language $\mathbf{PL}$. As usual, $\mathbf{PL}$ also denotes the set of all the formulas, and the *vocabulary of* $\mathbf{PL}$ is a set of *propositional symbols* denoted by $\mathcal{V}(\mathbf{PL})$. We restrict our attention to finite sets $\mathcal{V}(\mathbf{PL})$ in this text.

Letters φ, ψ denote formulas in $\mathbf{PL}$, $\top$ and $\perp$ denote respectively the true and the false formulas. *Interpretations for* $\mathbf{PL}$, identified with subsets of $\mathcal{V}(\mathbf{PL})$, are denoted by the letter ω. The notations $\omega \models \varphi$ and $\omega \models X$ for a set X

of formulas are defined classically. For a set E, $\mathcal{P}(E)$ denotes the set of the subsets of E. The set $\mathcal{P}(\mathcal{V}(\mathbf{PL}))$ of the interpretations for $\mathbf{PL}$ is denoted by **Mod**. A *model* of X is an interpretation ω such that $\omega \models X$, $\mathbf{Mod}(\varphi)$ and $\mathbf{Mod}(X)$ denote respectively the sets of the models of $\{\varphi\}$ and X.

A *literal* l is either a symbol p in $\mathcal{V}(\mathbf{PL})$ (*positive literal*) or its negation $\neg p$ (*negative literal*). If l is a literal, $\sim l$ denotes its *complementary literal*: $\sim \neg p = p$ and $\sim p = \neg p$. Similarly, we define $\sim \top = \bot$ and $\sim \bot = \top$.

A *clause* (respectively a *term*) is a disjunction (respectively a conjunction) of literals. Subsets of $\mathcal{V}(\mathbf{PL})$ are denoted by P, Q, V. P^+ (respectively P^-) denotes the set of the positive (respectively negative) literals built on P, and $P^\pm$ denotes the set $P^+ \cup P^-$ of all the literals built on P (P and P^+ can be assimilated). For any (finite) set X of formulas, $\bigwedge X$ (respectively $\bigvee X$) denotes the conjunction (respectively disjunction) of all the formulas in X. We get: $\bigwedge X \equiv X$, $\bigwedge \emptyset \equiv \top$ and $\bigvee \emptyset \equiv \bot$. $\mathcal{V}(X)$ denotes the set of the propositional symbols appearing in X.

A *disjunctive normal form* or DNF of φ is a disjunction of consistent terms which is equivalent to φ. A set L of literals in $V^\pm$ (and the term $\bigwedge L$) is *consistent and complete in* V if each propositional symbol of V appears once and only once in L; the clause $\bigvee L$ is then *non trivial and complete in* V. For any set L of literals, $\sim L$ denotes the set of the literals complementary to those in L (notice that $\sim P = P^-$).

We need the following notions and notations, many of them coming from (Lang, Liberatore, & Marquis 2003):

If φ is some formula and p is a propositional symbol in $\mathbf{PL}$, $\varphi_{p:\top}$ (respectively $\varphi_{p:\bot}$) is the formula obtained from φ by replacing each occurrence of p by $\top$ (respectively $\bot$). If $l = p$ is a positive literal, $\varphi_{l:i}$, denotes the formula $\varphi_{p:i}$[1]; if $l = \neg p$ is a negative literal, $\varphi_{l:i}$ denotes the formula $\varphi_{p:\sim i}$.

Notations 1 *1. If $v_1, \cdots, v_n$ are propositional symbols, $\varphi_{(v_1:\epsilon_1,\cdots,v_n:\epsilon_n)}$ with each $\epsilon_j \in \{\bot, \top\}$, denotes the formula $(\cdots((\varphi_{v_1:\epsilon_1})_{v_2:\epsilon_2})\cdots)_{v_n:\epsilon_n}$.*
If the v_j's in the list are all distinct, the order of the v_j's is without consequence for the final result. Thus, if V_1 and V_2 are disjoint subsets of V, we may define $\varphi_{[V_1:\top,V_2:\bot]}$ as $\varphi_{(v_1:\top,\cdots,v_n:\top,v_{n+1}:\bot,\cdots,v_{n+m}:\bot)}$, where $(v_1, \cdots, v_n)$ and $(v_{n+1}, \cdots, v_{n+m})$ are two orderings of all the elements of V_1 and V_2 respectively.

2. *If $L = (l_1, \cdots, l_n)$ is a list of literals, $\varphi_{(l_1:\epsilon_1\cdots l_n:\epsilon_n)}$ denotes the formula $(\cdots((\varphi_{l_1:\epsilon_1})_{l_2:\epsilon_2})\cdots)_{l_n:\epsilon_n}$.*
3. *Let $\mathcal{V}(\mathbf{PL})^\pm$ be ordered in some arbitrary way. If $L_1, \cdots, L_n$ are disjoint sets of literals, $\varphi_{\langle L_1:\epsilon_1,\cdots,L_n:\epsilon_n\rangle}$ denotes the formula $\varphi_{(l_1:\gamma_1,\cdots,l_n:\gamma_n)}$ where $(l_1, \cdots, l_n)$ is the enumeration of the set $L_1 \cup \cdots \cup L_n$ which respects the order chosen for the set of all the literals, and where, for each l_j, γ_j is equal to ϵ_r where $r \in \{1, \cdots, n\}$ is such that $l_j \in L_r$.*

[1]Notice that in (Lang, Liberatore, & Marquis 2003), "$\varphi_{l:\bot}$" (respectively "$\varphi_{l:\top}$") is denoted by "$\varphi_{l\leftarrow 0}$" (respectively "$\varphi_{l\leftarrow 1}$").

Forgetting propositional symbols

Let us remind a possible definition for this well known and old notion [2].

Definition 2 *If $V \subseteq \mathcal{V}(\mathbf{PL})$ and $\varphi \in \mathbf{PL}$, $ForgetV(\varphi, V)$ denotes a formula, in the propositional language $\mathbf{PL}_{\overline{V}}$ built on the vocabulary $\overline{V} = \mathcal{V}(\mathbf{PL}) - V$, which is equivalent to φ in this restricted language: $ForgetV(\varphi, V) \equiv Th(\varphi) \cap \mathbf{PL}_{\overline{V}}$ where $Th(\varphi) = \{\varphi' \in \mathbf{PL} / \varphi \models \varphi'\}$.*

For any $\psi \in \mathbf{PL}_{\overline{V}}$, $\varphi \models \psi$ iff $ForgetV(\varphi, V) \models \psi$.

Here are two known ways to get $ForgetV(\varphi, V)$:

1. In a DNF form of φ, for each term suppress all the literals in $V^\pm$ ("empty terms" being equivalent to $\top$ as usual).
2. For any formula φ, and any list V of propositional symbols, we get
   (a) $ForgetV(\varphi, \{v\} \cup V) = ForgetV(\varphi, V)_{v:\top} \vee ForgetV(\varphi, V)_{v:\bot}$,
   (b) $ForgetV(\varphi, \emptyset) = \varphi$.

The iterative point 2 applies to any formula, and shows that we can forget one symbol at a time. Also, the order is irrelevant: the final formulas are all equivalent when the order is modified. Here is the corresponding "global formulation" (cf Notations 1-1):

Definition 3 $ForgetV(\varphi, V) = \bigvee_{V' \subseteq V} \varphi_{[V':\top,(V-V'):\bot]}$.

Considering the formulation $ForgetV(\varphi, V) \equiv Th(\varphi) \cap \mathbf{PL}_{\overline{V}}$, the following obvious technical remark happens to be very useful:

Remark 4 *When considering a formula equivalent to a set $Th(\varphi) \cap X$, the set of formulas X can be replaced by any set Y having the same $\wedge$-closure: $\{\bigwedge X' / X' \subseteq X\} = \{\bigwedge X' / X' \subseteq Y\}$. Indeed, we have:*

- *If X and Y have the same $\wedge$-closure, then $Th(\varphi) \cap X \equiv Th(\varphi) \cap Y$.*
- *The converse is true*, provided that we assimilate equivalent formulas*: if $Th(\varphi) \cap X \equiv Th(\varphi) \cap Y$ for any $\varphi \in \mathbf{PL}$, then X and Y have the same $\wedge$-closure.*

Since we work in finite propositional languages, there exists a unique smallest (for set inclusion, and up to logical equivalence) possible set, the $\wedge$-reduct of X, equal to the set $X - \{\varphi \in X / \varphi$ is in the $\wedge$-closure of $X - \{\varphi\}\}$. Thus, X can be replaced by any set containing the $\wedge$-reduct of X and included in the $\wedge$-closure of X.

Thus, instead of considering the whole set $\mathbf{PL}_{\overline{V}}$ in $ForgetV(\varphi, V) \equiv Th(\varphi) \cap \mathbf{PL}_{\overline{V}}$ (Definition 2), we can consider the set of all the clauses built on $\overline{V}$, the smallest (for $\subseteq$) set that can be considered here being the set of these clauses which are non trivial and complete in $\overline{V}$.

[2]"V" in $ForgetV$ stands for "[propositional] variable", meaning "propositional symbol", and is in accordance with the notations of (Lang, Liberatore, & Marquis 2003), even if using term "variable" here could provoke confusions with the notions described later in this text.

On the semantical side, the set of the models of $ForgetV(\varphi, V)$ is the set of all the interpretations for $\mathbf{PL}$ which coincide with a model of φ for all the propositional symbols not in V: $\mathbf{Mod}(ForgetV(\varphi, V)) = \{\omega \in \mathbf{Mod} \ / \ \exists \omega', \ \omega' \models \varphi \text{ and } \omega \cap \overline{V} = \omega' \cap \overline{V}\}$.

These syntactical and semantical characterizations justify the name "$Forget$".

Example 1 *Here* $\mathcal{V}(\mathbf{PL}) = \{a, b, c, d\}$, *and* $\varphi = (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c \wedge \neg d)$.

DNF rule: $ForgetV(\varphi, \{b, c\}) \equiv (\neg a) \vee (a \wedge \neg d) \equiv \neg a \vee \neg d$.

iteratively: $ForgetV(\varphi, \{c\}) \equiv (\neg a \wedge b) \vee (a \wedge \neg b \wedge \neg d)$. $ForgetV(ForgetV(\varphi, \{c\}), \{b\}) \equiv ForgetV(\varphi, \{b, c\})$.

semantics:
Starting with $\mathbf{Mod}(\varphi) = \{\{a\}, \{b, c\}, \{b, c, d\}\}$, *we get the twelve models of* $ForgetV(\varphi, \{b, c\})$ *by adding all the interpretations varying on* $\{b, c\}$, *which gives the twelve interpretations:* $\emptyset \cup E, \{a\} \cup E, \{d\} \cup E$, *for any subset* E *of the set of the forgotten symbols* $\{b, c\}$.

Remind that for any formulas φ_1 and φ_2,
we get $ForgetV(\varphi_1 \vee \varphi_2, V) \equiv ForgetV(\varphi_1, V) \vee ForgetV(\varphi_2, V)$,
and $ForgetV(\varphi_1 \wedge \varphi_2, V) \models ForgetV(\varphi_1, V) \wedge ForgetV(\varphi_2, V)$.

Here is counter-example for the converse entailment: $\varphi_1 = a \vee \neg b, \varphi_2 = b$, thus $\varphi_1 \wedge \varphi_2 = a \wedge b$ and we get $ForgetV(\varphi_1, \{b\}) = ForgetV(\varphi_2, \{b\}) = \top$, while $ForgetV(\varphi_1 \wedge \varphi_2, \{b\}) = a$.

We need now another definition:

Definition 5 *(Lang, Liberatore, & Marquis 2003, pp. 396–397) Let* ω *be an interpretation for* $\mathbf{PL}$, p *be a propositional symbol in* $\mathbf{PL}$ *and* L *be a consistent set of literals in* $\mathbf{PL}$.
We define the interpretations
$Force(\omega, p) = \omega \cup \{p\}$ *and*
$Force(\omega, \neg p) = \omega - \{p\}$ *and more generally,*
$Force(\omega, L) = \omega \cup \{p / p \in \mathcal{V}(\mathbf{PL}), p \in L\} - \{p / p \in \mathcal{V}(\mathbf{PL}), \neg p \in L\}$.

Thus, $Force(\omega, L)$ is the interpretation for $\mathbf{PL}$ equal to ω for all the propositional symbols in $\mathcal{V}(\mathbf{PL}) - \mathcal{V}(L)$ and which satisfies all the literals of L.

An immediate consequence of the definition of $\varphi_{l:\top}$ is that we get: $\mathbf{Mod}(\varphi_{l:\top}) =$
$\{\omega / \omega \models \varphi, \omega \models l\} \cup \{Force(\omega, \sim l) / \omega \models \varphi, \omega \models l\} =$
$\{Force(\omega, l), Force(\omega, \sim l) / \omega \models \varphi, \omega \models l\}$ (Mod$_{l:\top}$).

It is then interesting to relate $ForgetV(\varphi, v)$ [$v \in \mathcal{V}(\mathbf{PL})$] to the formulas $\varphi_{v:\top}$ and $\varphi_{v:\perp}$:

$$\begin{aligned} \varphi_{v:\top} &\equiv ForgetV(v \wedge \varphi, v); \\ \varphi_{v:\perp} &\equiv ForgetV(\neg v \wedge \varphi, v). \\ ForgetV(\varphi, v) &\equiv \varphi_{v:\top} \vee \varphi_{v:\perp}. \end{aligned}$$

Indeed, $ForgetV(\varphi, v) \equiv \varphi_{v:\top} \vee \varphi_{v:\perp}$ and $\varphi \equiv (v \wedge \varphi_{v:\top}) \vee (\neg v \wedge \varphi_{v:\perp})$ are obvious, while choosing $l = v$ in result (Mod$_{l:\top}$) gives: $\varphi_{v:\top} \equiv ForgetV(v \wedge \varphi, v)$.

Thus, we get, for each $\epsilon \in \{\perp, \top\}$:
$(\varphi \vee \psi)_{l:\epsilon} \equiv \varphi_{l:\epsilon} \vee \psi_{l:\epsilon}$, and also $(\varphi \wedge \psi)_{l:\epsilon} \equiv \varphi_{l:\epsilon} \wedge \psi_{l:\epsilon}$.

Remark 6 *Let* φ *be any formula and* l *some literal with* v_l *as its propositional symbol. Then, the following six formulas are all equivalent:*

$$\begin{array}{lll} ForgetV(l \wedge \varphi, v_l) \vee \varphi & & \equiv \\ (\neg l \wedge ForgetV(l \wedge \varphi, v_l)) \vee \varphi & & \equiv \\ \varphi_{l:\top} \vee (\neg l \wedge \varphi) \quad \equiv & \varphi_{l:\top} \vee (\neg l \wedge \varphi_{l:\perp}) & \equiv \\ \varphi_{l:\top} \vee \varphi \quad \equiv & \varphi \vee (\neg l \wedge \varphi_{l:\perp}). & \end{array}$$

Indeed, the set of the models of each of these formulas is $\{Force(\omega, l) / \omega \models \varphi, \omega \models l\} \cup \{Force(\omega, \sim l) / \omega \models \varphi, \omega \models l\} \cup \{Force(\omega, \sim l) / \omega \models \varphi, \omega \models \neg l\}$.

Forgetting literals

Variable forgetting as been generalized as detailed now, beginning with the semantical side.

Definition 7 (Literal forgetting) *(Lang, Liberatore, & Marquis 2003, Prop. 15) If* φ *is a formula and* L *a set of literals in* $\mathbf{PL}$, $ForgetLit(\varphi, L)$ *is a formula having for models the set of all the interpretations for* $\mathbf{PL}$ *which can be turned into a model of* φ *when forced by a consistent subset of* L*:*

$$\mathbf{Mod}(ForgetLit(\varphi, L)) = \{\omega / Force(\omega, L_1) \models \varphi \text{ and } L_1 \text{ is a consistent subset of } L\}.$$

Thus, the models of $ForgetLit(\varphi, L)$ are built from the models of φ by allowing to "negate" (or "complement") an arbitrary number of values of literals in L:

$$\mathbf{Mod}(ForgetLit(\varphi, L)) = \{Force(\omega', L_1') \ / \omega' \models \varphi \text{ and } L_1' \text{ is a consistent subset of } \sim L\}.$$

Let us consider the syntactical side now. One way is to start from a DNF formulation of φ:

Proposition 8 *(Lang, Liberatore, & Marquis 2003) If* $\varphi = t_1 \vee \cdots \vee t_n$ *is a DNF,* $ForgetLit(\varphi, L)$ *is equivalent to the formula* $t_1' \vee \cdots \vee t_n'$ *where* t_i' *is the term* t_i *without the literals in* L.

The similar method for obtaining $ForgetV(\varphi, V)$ when φ is a DNF has been reminded in point 1 following Definition 2. Similarly, the following syntactical definition, analogous to Definition 3, can be given:

Definition 9 *If* L *is a set of literals in* $\mathbf{PL}$, *then*

$$ForgetLit(\varphi, L) = \bigvee_{L' \subseteq L} \left(\left(\bigwedge \sim L' \right) \wedge \varphi_{\langle (L - L'):\top \rangle} \right).$$

This is a "global formulation", easily shown to be equivalent to the following ***iterative definition*** (Lang, Liberatore, & Marquis 2003, Definition 7):

1. $ForgetLit(\varphi, \emptyset) \quad = \quad \varphi$.

2. $ForgetLit(\varphi, \{l\}) = \varphi_{l:\top} \vee \varphi$.
3. $ForgetLit(\varphi, \{l\} \cup L) = ForgetLit(ForgetLit(\varphi, L), l)$.

We refer the reader to (Lang, Liberatore, & Marquis 2003) which shows the adequacy with Definition 7 and Proposition 8, and also that choosing any order of the literals does not modify the meaning of the final formula (cf Notations 1-3). It follows that this independence from the order of the literals also applies to the global formulation in Definition 9. The fact that, exactly as with the notion of forgetting symbols (cf Definition 2 and following comment), the notion of forgetting literals has such an iterative definition is important from a computational point of view (Lang, Liberatore, & Marquis 2003).

Notice that (Lang, Liberatore, & Marquis 2003) uses the formula $\varphi_{l:\top} \vee (\neg l \wedge \varphi)$ in point 2, and also the variant $\varphi_{l:\top} \vee (\neg l \wedge \varphi_{l:\top})$, instead of $\varphi_{l:\top} \vee \varphi$. Remark 6 shows that any of the six formulas given there could be used here, which could marginally simplify the computation, depending on the form in which φ appears.

The presence of $\left(\bigwedge_{l' \in L'} \neg l'\right)$ in Definition 9, which is what differentiates $ForgetLit(\varphi, ...)$ from $ForgetV(\varphi, ...)$, comes from the fact that here we forget $l \in L$ but we do not forget $l' \in \sim L$.

A proof in (Lang, Liberatore, & Marquis 2003), using Proposition 8, shows that we get $ForgetLit(\varphi, V^{\pm}) \equiv ForgetV(\varphi, V)$. This proof is easily extended to get the following result:

Remark 10 *Since any set of literals can be written as a disjoint union between a consistent set L' and a set $V^{\pm}$ of complementary literals, here is a useful formulation:*

$$ForgetLit(\varphi, L' \cup V^{\pm}) \equiv ForgetLit(ForgetV(\varphi, V), L').$$

Notice that we could also forget the literals first, i.e. consider the formula $ForgetV(ForgetLit(\varphi, L'), V)$, even if it seems likely that this is less interesting from a computational point of view.

This remark has the advantage of separating clearly the propositional symbols into three kinds. Let V' denote the set $\mathcal{V}(L')$ of the propositional symbols in L', and $V'' = \mathcal{V}(\mathbf{PL}) - V - V'$ be the set of the remaining symbols. Then we get:

1. The propositional symbols in V are forgotten.
2. The propositional symbols in V'' are *fixed*, since the literals in $V''^{\pm}$ are not forgotten.
3. The remaining symbols, in V', are neither forgotten nor fixed, since only the literals in L' are forgotten, but not the literals in $\sim L'$.

Thus, $ForgetLit(\varphi, L_1)$ can be defined as: *forgetting literals with some propositional symbols fixed*. It is then natural to generalize the notion, by allowing some propositional symbols to *vary* in the *forgetting* process.

Forgetting literals with varying symbols

As done with the original notion, let us begin with the semantical definitions.

Definition 11 *Let φ be a formula, V a set of propositional symbols, and L a consistent set of literals, in $\mathbf{PL}$, with V and $\mathcal{V}(L)$ disjoint in $\mathcal{V}(\mathbf{PL})$. $ForgetLitVar(\varphi, L, V)$ is a formula having the following set of models:*

$$\mathbf{Mod}(ForgetLitVar(\varphi, L, V)) = \{\omega \;/ Force(\omega, L_1 \cup L_2) \models \varphi, L_1 \subseteq L, L_2 \subseteq V^{\pm}, L_2 \textit{ consistent}, \textit{ and } (\omega \not\models L_1 \textit{ or } L_2 = \emptyset)\}.$$

This is equivalent to:

$$\mathbf{Mod}(ForgetLitVar(\varphi, L, V)) = \mathbf{Mod}(\varphi) \cup \{Force(\omega, L_1 \cup L_2) \;/\; \omega \models \varphi, \omega \not\models L_1, L_1 \subseteq \sim L, \;\; L_2 \subseteq_{cons} V^{\pm}\}.$$

Notice the notation $\subseteq_{cons} V^{\pm}$ for "included in $V^{\pm}$ and consistent".

Since $\omega \models L_2$ iff $Force(\omega, L_2) = \omega$, the condition "*($\omega \not\models L_1$ or $L_2 = \emptyset$)*" can be replaced by "*($\omega \not\models L_1$ or $\omega \models L_2$)*", and then we can replace everywhere here "*L_2 consistent*" by "*L_2 consistent and complete in V*" (there are $3^{card(V)}$ consistent sets L_2 and "only" $2^{card(V)}$ consistent and complete sets).

We could be more general, by allowing to forget some propositional symbols, which amounts to allow non consistent sets L. This generalization does not present difficulties, however, since we have not found any application for it till now, we leave it for future work.

With respect to Definition 7, what happens here is that the non consistent part of the set of literals, which allowed to forget some set V of propositional symbols altogether, has been replaced by a set of varying propositional symbols.

Remark 12 *Since $ForgetLit(L_1, \varphi) \models ForgetLit(L_1 \cup L_2, \varphi)$ holds from (Lang, Liberatore, & Marquis 2003) (*"the more we forget, the less we know"*), we get:*

$$\varphi \models ForgetV(\varphi, V) \models ForgetLit(\varphi, L \cup V^{\pm}).$$

Similarly, it is clear that the new definition allows a finer (more cautious) forgetting than $ForgetLit$:

$$\varphi \models ForgetLitVar(\varphi, L, V) \models ForgetLit(\varphi, L \cup V^{\pm}).$$

Remind the motivations for introducing $ForgetLitVar$: we want to "forget" the literals in L, even at the price of modifying the literals in $V^{\pm}$: if we effectively forget at least one literal in L, then, we allow any modification for the literals in $V^{\pm}$. However, we do not want to modify the literals in $V^{\pm}$ "for nothing" our aim being to forget as many literals in L as possible. This justifies the appearance of the condition "$\omega \not\models L_1$" in the definition and in the alternative formulation.

The syntactical aspect is slightly more tricky, but it remains rather simple and it allows to revisit and improve already known results. As with the original notion (see Proposition 8), the simplest way is to start from a DNF.

Since L is consistent, without loss of generality and in order to simplify the notations, we can consider that L is a set of negative literals (otherwise, replace any $p \in \mathcal{V}(L)$ such that $p \in L$ by $\neg p'$, p' being a new propositional symbol, then after the computations, replace p' by $\neg p$). Thus, till the end of this section, we will consider two disjoint subsets P and V of $\mathcal{V}(\mathbf{PL})$, and $L = P^-$ with $Q = \mathcal{V}(\mathbf{PL}) - V - P$ denoting the set of the remaining propositional symbols.

Proposition 13 *(See proof in Appendix) Let $\varphi = t_1 \vee \cdots \vee t_n$ be a DNF, with*

$$t_i = (\bigwedge P_{i,1}) \wedge (\bigwedge \neg(P_{i,2})) \wedge (\bigwedge V_{i,l}) \wedge (\bigwedge Q_{i,l}),$$

where $P_{i,1} \subseteq P$, $P_{i,2} \subseteq P - P_{i,1}$, with $V_{i,l} \subseteq V^{\pm}$ and $Q_{i,l} \subseteq Q^{\pm}$ being consistent sets of literals. Then $ForgetLitVar(\varphi, P^-, V) \equiv t'_1 \vee \cdots \vee t'_n$ where

$$t'_i = (\bigwedge P_{i,1}) \wedge (\bigwedge Q_{i,l}) \wedge [(\bigvee (P - P_{i,1})) \vee (\bigwedge V_{i,l})], i.e.$$

$$t'_i = (\bigwedge P_{i,1}) \wedge (\bigwedge Q_{i,l}) \wedge [\bigwedge_{l \in V_{i,l}} (l \vee (\bigvee (P - P_{i,1})))].$$

Thus, t'_i is t_i except that the literals in P^- are suppressed while each literal in $V^{\pm}$ must appear in disjunction with the clause $\bigvee(P - P_1)$, this clause denoting the disjunction of all the literals in P^+ which do not appear (positively) in t_i. Naturally, the literals of $L = P^-$ appearing in t_i disappear. Moreover, it is important to notice that the literals from $P^{\pm} = L \cup \sim L$ in t_i which remain are those which do not appear positively in t_i. This means that t_i could be "completed in P" by the conjunction of all the $\neg p$ for each symbol $p \in P$ not appearing in t_i, without modifying the "forget" formula.

We have provided the semantical definition (in the lines of Definition 7) and a characterization from a DNF formulation (in the lines of Proposition 8). Let us provide now other characterizations, and a comparison with $ForgetLit$.

Proposition 14 *Let φ be a formula in $\mathbf{PL}$, and P, Q and V be three pairwise disjoint sets of propositional symbols such that $P \cup Q \cup V = \mathcal{V}(\mathbf{PL})$.*

1. *$ForgetLit(\varphi, P^- \cup V^{\pm})$ is equivalent to the set $Th(\varphi) \cap X$ where X is the set of the formulas in $\mathbf{PL}$ which are disjunctions of terms of the kind $(\bigwedge P_1) \wedge (\bigwedge Q_l)$ with $P_1 \subseteq P$ and $Q_l \subseteq Q^{\pm}$.*
2. *$ForgetLitVar(\varphi, P^-, V)$ is equivalent to the set $Th(\varphi) \cap X$ where X is the set of the formulas in $\mathbf{PL}$ which are disjunctions of terms of the kind $(\bigwedge P_1) \wedge (\bigwedge Q_l) \wedge [\bigwedge_{l \in V_l} (l \vee (\bigvee (P - P_1)))]$, where $P_1 \subseteq P$, $V_l \subseteq_{cons} V^{\pm}$ and $Q_l \subseteq Q^{\pm}$.*

(We can clearly consider consistent sets Q_l only.)

These two results are immediate consequences of Propositions 8 and 13 respectively. We get the following alternative possibilities for the sets X's, firstly by boolean duality from the preceding results, then by considering some set having the same $\wedge$-closure as X (Remark 4):

Proposition 14 (following)

1(a) *For $ForgetLit(\varphi, P^- \cup V^{\pm})$, X is the set of the conjunctions of the clauses of the kind $(\bigvee P_1) \vee (\bigvee Q_l)$ with $P_1 \subseteq P$ and $Q_l \subseteq Q^{\pm}$ (we can clearly consider consistent sets Q_l only).*

(b) *We can also consider the set X of the clauses $(\bigvee P_1) \vee (\bigvee Q_l)$ with $P_1 \subseteq P$ and $Q_l \subseteq Q^{\pm}$.*

(c) *The smallest set X possible is the set of the clauses $(\bigvee P_1) \vee (\bigvee Q_l)$ with $P_1 \subseteq P$, $Q_l \subseteq Q^{\pm}$, Q_l consistent and complete in Q.*

2(a) *For $ForgetLitVar(\varphi, P^-, V)$, X is the set of the conjunctions of the formulas $flv(P_1, Q_l, V_l) = (\bigvee P_1) \vee (\bigvee Q_l) \vee \bigvee_{l \in V_l} (l \wedge (\bigwedge (P - P_1)))$, where $P_1 \subseteq P$, $V_l \subseteq_{cons} V^{\pm}$ and $Q_l \subseteq_{cons} Q^{\pm}$.*

(b) *We can also consider the set X of all the formulas $flv(P_1, Q_l, V_l)$ of this kind.*

(c) *The smallest set X possible is the set of the formulas $flv(P_1, Q_l, V_l)$ with $P_1 \subseteq P$, Q_l and V_l being sets of literals consistent and complete in Q and V respectively.*

These results provide the analogous, for $ForgetLit$ and $ForgetLitVar$, of the results for $ForgetV$ reminded in Definition 2, and in Remark 4.

The next definition is analogous to Definitions 3 and 9 (see appendix for a proof of the adequacy with Definition 11):

Definition 15 *If φ is a formula and P and V are two disjoint subsets of $\mathcal{V}(\mathbf{PL})$, then $ForgetLitVar(\varphi, P^-, V)$ is the formula*

$$\bigvee_{P_1 \subseteq P} \Big(\bigwedge P_1 \wedge (\varphi_{[P_1:\top,(P-P_1):\bot]} \vee (ForgetV(\varphi_{[P_1:\top,(P-P_1):\bot]}, V) \wedge (\bigvee (P - P_1)))) \Big).$$

Example 2 *Here $P = \{a, b\}, V = \{c\}, Q = \{d\}$, with $\varphi = (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c \wedge \neg d)$.*

Syntactical side:

Since φ is a DNF, the rules from a DNF after Definition 2 (for $ForgetV$), in Proposition 8 (for $ForgetLit$) and Proposition 13 (for $ForgetLitVar$), give the three results:

- $ForgetV(\varphi, V) \equiv (\neg a \wedge b) \vee (a \wedge \neg b \wedge \neg d)$.
- $ForgetLit(\varphi, P^- \cup V^{\pm}) \equiv b \vee (a \wedge \neg d)$.
- $ForgetLitVar(\varphi, P^-, V) \equiv$
  $(a \wedge b) \vee (a \wedge \neg c \wedge \neg d) \vee (b \wedge c)$. $\quad FLV1$

Definitions 9 and 15 can be used also, as shown now for Definition 15 where, in each case, $\psi = \varphi_{[P_1:\top,(P-P_1):\bot]}$:

$$P_1 = \emptyset : \psi \vee (ForgetV(\psi, c) \wedge (a \vee b)) \equiv \bot \vee (\bot \wedge (a \vee b)) \equiv \bot. \qquad (\varphi_1)$$

$$P_1 = \{a\} : a \wedge (\psi \vee (ForgetV(\psi, c) \wedge b)) \equiv a \wedge ((\neg c \wedge \neg d) \vee (\neg d \wedge b)). \qquad (\varphi_2)$$

$$P_1 = \{b\} : b \wedge (\psi \vee (ForgetV(\psi, c) \wedge a)) \equiv b \wedge (c \vee (\top \wedge a)). \quad (\varphi_3)$$

$$P_1 = \{a, b\} : a \wedge b \wedge (\psi \vee (ForgetV(\psi, c) \wedge \bot)) \equiv a \wedge b \wedge (\bot \vee \bot) \equiv \bot. \quad (\varphi_4)$$

The disjunction $\bigvee_{i=1}^{4} \varphi_i$ *is equivalent to* $FLV1$.

Semantical side:

We get $\mathbf{Mod}(\varphi) = \{\{a\}, \{b, c\}, \{b, c, d\}\}$.

- *The six models of* $ForgetV(\varphi, V)$ *are obtained by adding the three interpretations differing from the three models of* φ *by the value attributed to* c *(cf example 1):* $\{a, c\}, \{b\}$, *and* $\{b, d\}$.
- *The ten models of* $ForgetLit(\varphi, P^- \cup V^\pm)$ *are obtained by adding to the models of* φ *the seven interpretations differing from these models by adding any subset of* $\{a, b\}$ *and by either do nothing else or modify the value of* c *(adding* c *if it is not present and removing* c *if it is present). This gives the six models of* $ForgetV(\varphi, V)$ *plus the four interpretations including* $\{a, b\}$.
- *The seven models of* $ForgetLitVar(\varphi, P^-, V)$ *are obtained by adding to the three models of* φ *the four interpretations differing from these models by adding a* non empty *subset of* $\{a, b\}$ *and by either do nothing else or modify the value of* c*, which gives here the four interpretations including* $\{a, b\}$.

Here is a technical result which can be drawn from this example, and which may have a computational interest:

Remark 16 *1. For any formula* φ *we get:*
$ForgetV(\varphi, V) \vee ForgetLitVar(\varphi, P^-, V) \equiv ForgetLit(\varphi, P^- \cup V^\pm)$

2. For any formula φ *which is* uniquely defined in P*, we get:*
$ForgetV(\varphi, V) \wedge ForgetLitVar(\varphi, P^-, V) \equiv \varphi$.
By formula uniquely defined in P *we mean a formula which is equivalent to a conjunction* $\varphi_1 \wedge \varphi_2$*, where* φ_1 *is a term complete in* P *and* φ_2 *is without symbol of* P.

See the Appendix for a proof. This remark can be compared with Remark 12. Notice that in Example 2, the formula φ is uniquely defined in P [indeed, $\varphi \equiv (\neg a \wedge b) \wedge (c \vee (\neg c \wedge \neg d))$], thus points 1. and 2. of this Remark are satisfied. Here is a simple counter-example (where the important fact to notice is that φ is a term which is *not complete in* P, i.e. $P_{i,1} \cup P_{i,2} \neq P$) showing that the second equivalence does not hold for any formula.

Example 3 P, V, Q*, and* $\mathbf{PL}$ *as in example 2,* $\varphi = t = a \wedge c$. *We get:*

- $ForgetV(t, V) \equiv a$.
- $ForgetLit(t, P^- \cup V^\pm) \equiv a$.
- $ForgetLitVar(t, P^-, V) \equiv ForgetLitVar(a \wedge \neg b \wedge c, P^-, V) \equiv a \wedge (b \vee c)$.

Notice also that, once we have all the models of φ, the complexity of the construction of all the models of $ForgetLitVar(\varphi, P^-, V)$ is not greater than the complexity of the construction of all the models of $ForgetLit(\varphi, P^- \cup V^\pm)$.

More about the computation of these notions

On the syntactical side, we have the same kind of iterative definition than we had for $ForgetV$ and $ForgetLit$ (cf the two "iterative definitions", in Point 2 just before Definition 3 for $ForgetV$, and after Definition 9 for $ForgetLit$):

Remark 17 *Let us suppose that* V *is a set of propositional symbols and that* $L \cup \{l\}$ *is a consistent set of literals without symbol in* V *and such that* $l \notin L$.

1. $ForgetLitVar(\varphi, \emptyset, V) = \varphi$;
2. $ForgetLitVar(\varphi, \{l\}, V) = \varphi \vee ForgetV(\neg l \wedge ForgetV(l \wedge \varphi, v_l), V)$ *(where* v_l *denotes the symbol of* l*).*
3. $ForgetLitVar(\varphi, \{l\} \cup L, V) = ForgetLitVar(ForgetLitVar(\varphi, L, V), \{l\}, V)$.

We get equivalent formulas for each order of appearance of the literals in the iterative process. The complexity of the computation of $ForgetLitVar(\cdots, L, V)$ should be only slightly harder than for the computation of $ForgetLit$. Indeed, we have to "forget V" for each new literal, which introduces a rather small new complication, otherwise, computing $\neg l \wedge ForgetV(l \wedge ForgetLitVar(\varphi, L, V), v_l)$ is not harder than computing $ForgetLit(ForgetLit(\varphi, L), l)$.

See the appendix for the proof of the equivalence with Definition 15. Notice already that the formula $(\neg l \wedge ForgetV(l \wedge \Phi, v_l))$ has for models the models of Φ which are *actively forced by* $\neg l$ (l was true in the initial model, and l is forced to be false).

Formally, $\mathbf{Mod}(\neg l \wedge ForgetV(l \wedge \Phi, v_l)) = \{Force(\omega, \sim l)/\omega \models \Phi \wedge l\}$. (M¬lFVl)

It seems important, from a computational point of view, to describe an alternative syntactical way to compute this formula (besides the possibility of using the formulation in $ForgetV$ given above). Here is a syntactical method.

From (M¬lFVl), we get

$$\neg l \wedge ForgetV(l \wedge \Phi, v_l) \equiv \neg l \wedge [l \wedge \Phi]_{l:\top}. \quad \text{(F¬lFVl)}$$

An interesting point in the proof of the equivalence between Remark 17 and Definition 15 is that it shows how to improve the computation a bit. Indeed, once a model has been modified by some $l \in L$, the set of all its variant in V (i.e. the set $\{Force(\omega, L_2)/L_2 \subseteq_{cons} V\pm\}$) is already computed. Thus, for such a model, it is useless to compute again all the variants in V, since they are already present, and forgetting one more literal in L will have no consequence to that respect: since we had already all the variants in V, modifying a new symbol brings only one more model (at most, it was not already present) without the need to consider again all the variants in V for this model.

This gives rise to the following iterative process:

1. $ForgetLitVar(\varphi, \emptyset, V) = \varphi$;
2. $ForgetLitVar(\varphi, \{l\} \cup L, V) =$
$$\Phi \vee \Phi_{l:\top} \vee ForgetV(\neg l \wedge [l \wedge \varphi]_{l:\top}, V)$$
where $\Phi = ForgetLitVar(\varphi, L, V)$.

Remind that $\neg l \wedge [l \wedge \varphi]_{l:\top}$ can be replaced by $\neg l \wedge ForgetV(l \wedge \varphi, v_l)$ (see formula (F¬lFVl).

The simplification with respect to Remark 17 comes from the fact that only the "fixed" formula φ is considered when forgetting the symbols in V, instead of the "moving" formula $ForgetLitVar(\varphi, L, V)$. This can be interesting, since φ can be simplified before the computations, which will then be facilitated.

Let us apply this improved iterative method to Example 2:

Example 4 *cf Example 2:* $P = \{a, b\}, V = \{c\}, Q = \{d\}$, *with* $\varphi = (\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c \wedge \neg d)$.

- *We compute* $ForgetLitVar(\varphi, P, V)$ *again:*

1. $\Phi^0 = ForgetLitVar(\varphi, \emptyset, \{c\}) = \varphi$;
2. $\Phi^1 = \Phi^0 \vee \Phi^0_{\neg a:\top} \vee ForgetV(a \wedge [\neg a \wedge \varphi]_{\neg a:\top}, c) \equiv$
$((\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c \wedge \neg d)) \vee$
$(b \wedge c) \vee ForgetV(a \wedge (b \wedge c), \{c\}) \equiv$
$(\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge \neg c \wedge \neg d)) \vee (b \wedge c) \vee (a \wedge b) \equiv$
$(a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (a \wedge b) \vee (b \wedge c)$;
3. $ForgetLitVar(\varphi, P, V) = \Phi^2 =$
$\Phi^1 \vee \Phi^1_{\neg b:\top} \vee ForgetV(b \wedge [\neg b \wedge \varphi]_{\neg b:\top}, c)) \equiv$
$((a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (a \wedge b) \vee (b \wedge c)) \vee (a \wedge \neg c \wedge \neg d) \vee$
$ForgetV(b \wedge (a \wedge \neg c \wedge \neg d), \{c\}) \equiv$
$(a \wedge \neg b \wedge \neg c \wedge \neg d) \vee (a \wedge b) \vee (b \wedge c) \vee (a \wedge \neg c \wedge \neg d) \vee$
$(a \wedge b \wedge \neg d) \equiv$
$(a \wedge b) \vee (b \wedge c) \vee (a \wedge \neg c \wedge \neg d)$ *(cf Example 2).*

Conclusion and perspectives

Why could this work be useful:

The notion of forgetting literals consists in small manipulations of propositional formulas. This notion can help the effective computation of various useful already known knowledge representation formalisms. As shown in (Lang, Liberatore, & Marquis 2003), we cannot hope that this will solve all the problems, but it should help in providing significant practical improvements. And the introduction of varying symbols while forgetting literals should enhance these improvements in a significant way. However, the present text has not developed this applicative matter. Let us just remind a few indications on this subject now [see (Lang, Liberatore, & Marquis 2003; Moinard 2005) for more details]. Various knowledge representation formalisms are known to be concerned, we will only evoke circumscription.

Circumscription (McCarthy 1986) is a formalism aimed at *minimizing* some set of propositional symbols. For instance, circumscribing the symbol $exceptional$ in the sub-formula $bird \wedge \neg exceptional \rightarrow flies$ of our introductory example would conclude $\neg exceptional$ since it is compatible with the sub-formula that "no exception" happens. Notice that even on this simple example a complication appears: we cannot "circumscribe" $exceptional$ alone, if we want the expected minimization to hold here. Instead, we must also allow at least one other symbol to *vary* during the circumscription (e.g. we could allow $flies$ to *vary* while $exceptional$ is *circumscribed*). Circumscription is used in action languages and other formalizations of common sense reasoning, but a key and limiting issue is the efficient computation. The notion of forgetting literals provides a (limited, but real) progress on the subject. The main result is the following one:

$$Circ(P, Q, V)(\varphi) \models \psi \quad \text{iff}$$
$$\varphi \models ForgetLitVar(\varphi \wedge \psi, P^-, V).$$

The propositional symbols in P, V, Q are respectively *circumscribed*, *varying*, and *fixed* in the "circumscription of the formula φ" here.

This result is known to improve (from a computational perspective) previously known results, mainly a result from (Przymusinski 1989). The notion of varying symbols allows some simplification with respect to Przymunsinski's method and even with respect to the computational improvements of this method discovered by (Lang, Liberatore, & Marquis 2003).

What has been done here:

We have provided the semantical and several syntactical characterizations for a new notion, extending the notion of literal forgetting introduced in (Lang, Liberatore, & Marquis 2003) to the cases where some propositional symbols are allowed to vary. These results show that the new notion is not significantly harder than literal forgetting without varying symbols. The various characterizations provide effective ways for computing the results, depending on the form in which the formulas appear. These different ways for computing the notions introduced should help the effective computation in many cases. This is why we have provided several equivalent formulas for the main formulas introduced here, and also for some important auxiliary formulas involved in the definitions. This kind of work is absolutely necessary when coming to the effective computation. Indeed, as shown in (Lang, Liberatore, & Marquis 2003), no formulation can be considered as the best one in any case.

Hopefully, the various ways of defining the formulas and notions introduced here could also help getting a better grasp of these notions, since they are not very well known till now.

What remains to be done:

Various knowledge representation formalisms are known to be concerned (Lang, Liberatore, & Marquis 2003). Moreover, it is highly probable that these notions of forgetting literals for themselves can give rise to new useful formalizations of old problems in knowledge representation. It seems even likely that new knowledge representation formalisms could emerge from these enhanced notions of "forgetting".

More concretely, the notion of "forgetting" can still be generalized: we could directly "forget formulas" (instead of just "literals"), in the lines of what has been done with formula circumscription with respect to predicate circumscription.

Again more concretely, the present work [after the initiating work of (Lang, Liberatore, & Marquis 2003)] has given a preliminary idea on what kind technical work can be done for simplifying the effective computation of the formulas involved in the forgetting process. It is clear that a lot of important work should still be done on the subject.

Also, the complexity results, which have been described in (Lang, Liberatore, & Marquis 2003), should be extended to the new notion, and to the new methods of computation. This is far from simple since, as shown in (Lang, Liberatore, & Marquis 2003), it seems useless to hope for a general decrease of complexity with respect to the already known methods. So, the methods should be examined one by one, and for each method, its range of utility (the particular formulations for a given formula φ for which the method is interesting) should be discovered and discussed.

Appendix

Proof of Proposition 13:

Let us consider complete terms first, such as

$$t_i = t = (\bigwedge P_1) \wedge (\bigwedge \neg(P - P_1)) \wedge (\bigwedge V_l) \wedge (\bigwedge Q_l),$$

where $P_1 \subseteq P$, V_l and Q_l being consistent and complete sets of literals in V and Q respectively. t corresponds to an interpretation ω. The set $F(\omega) = \{Force(\omega, L_1 \cup L_2) \ / \ L_1 \subseteq P,\ L_2 \subseteq V^{\pm}, L_2$ consistent and complete in V, and $\omega \not\models L_1$ or $\omega \models L_2\}$ is the set of the models of the formula $t^1 \wedge t^2$ where $t^1 = (\bigwedge P_1) \wedge (\bigwedge Q_l)$ and $t^2 = \neg(\bigwedge \neg(P - P_1)) \vee (\bigwedge V_l)$, i.e. $t^2 \equiv (\bigvee(P - P_1)) \vee (\bigwedge V_l))$.

Indeed, for each $\omega' \in F(\omega)$, t^1 holds since it holds in ω, and the symbols in $P - P_1$ and V can take any value satisfying the condition $\omega \not\models L_1$ or $\omega \models L_2$. Since $\omega \models t$, this means $L_1 \cap (P - P_1) \neq \emptyset$ or $L_2 \subseteq V_l$, which is equivalent to $\omega' \models t_2$. Conversely, any model ω'' of $t_1 \wedge t_2$ is easily seen to be in $F(\omega)$.

The same result holds for any (consistent) term $t = t_i = (\bigwedge P_1) \wedge (\bigwedge \neg(P_2)) \wedge (\bigwedge V_l) \wedge (\bigwedge Q_l)$, where $P_1 \subseteq P$, $P_2 \subseteq P - P_1$, V_l and Q_l being consistent subsets of $V^{\pm}$ and $Q^{\pm}$ respectively: Let us first consider separately the cases where some symbols in P are missing, then symbols in V, then symbols in Q.

(1) If $p \in P$ does not appear in t, for any model ω' of t, $\omega'' = Force(\omega', \{\neg p\})$ and $Force(\omega'', \{p\})$ are two models of t (one of these is ω'). By considering all the missing p's, we get that the set $\{Force(\omega', L_1 \cup L_2)/\omega' \models t,\ L_1 \subseteq P^-, L_2 \subseteq_{cons} V^{\pm}, \omega' \not\models L_1$ or $L_2 = \emptyset\}$ is included in the set $\{Force(\omega'', L_1 \cup L_2)/\omega'' \models t \wedge \bigwedge \neg(P - P_1),\ L_1 \subseteq P^-, L_2 \subseteq_{cons} V^{\pm}, \omega'' \not\models L_1$ or $L_2 = \emptyset\}$. Thus any missing p in t behaves as if the negative literal $\neg p$ was present: we get a term "completed in P" satisfying $ForgetLitVar(t, P^-, V) \equiv ForgetLitVar(t \wedge \neg(P - P_1), P^-, V)$.

(2) The reasoning for a missing q in t ($q \in Q$) is simpler yet: if some $q \in Q$ does not appear in t, it can be interpreted as false or true for any model of $ForgetLitVar(t, L, Q)$, which means that we keep the part $\bigwedge Q_l$ unmodified, exactly as in the case where Q_l is complete in Q.

(3) The case for V is similar (the disjunction of all the formulas with all the possibilities for the missing symbols gives the formula where these symbols are missing): If some $v \in V$ is missing in t, then any model ω' of t has its counterpart where the value for v is modified. Let us call V_m the set of the symbols in V which are absent in t. By considering the disjunctions of all the possibilities, we get the formula $\bigvee_{V_l' \in \mathcal{L}_m}((\bigwedge P_1) \wedge (\bigwedge Q_l) \wedge ((\bigvee(P - P_1)) \vee (\bigwedge V_l \wedge \bigwedge V_l')))$, where $\mathcal{L}_m$ is the set of all the sets of literals consistent and complete in V_m. This is equivalent to the formula $(\bigwedge P_1) \wedge (\bigwedge Q_l) \wedge ((\bigvee(P - P_1)) \vee (\bigwedge V_l))$.

Combining "the three incompleteness" (1)–(3) gives:
$ForgetLitVar(t_i, P^-, V) \equiv (\bigwedge P_1) \wedge (\bigwedge Q_l) \wedge ((\bigvee(P - P_1)) \vee (\bigwedge V_l))$.

The disjunction for all the t_i's gives the result. □

Proof of the adequacy of Definition 15 with Definition 11:

Each model ω of φ gives rise to the following models of $ForgetLitVar(\varphi, P^-, V)$:

- ω itself, model of $\psi_1 = \bigwedge P_1 \wedge \bigwedge \neg(P - P_1) \wedge \varphi_{[P_1:\top,\ (P-P_1):\bot]}$ where $P_1 = \omega \cap P$, together with
- all the interpretations differing from ω in that they have at least one more $p \in P$, and no constraint holds for the symbols in V; this set of interpretations being the set of models of the formula $\psi_2 = \bigwedge P_1 \wedge ForgetV(\varphi_{[P_1:\top,\ (P-P_1):0]}, V) \wedge \bigvee(P - P_1)$.

Since $\varphi_{[P_1:\top,(P-P_1):\bot]} \models ForgetV(\varphi_{[P_1:\top,(P-P_1):\bot]}, V)$ and $\bigwedge \neg(P - P_1) \equiv \neg(\bigvee(P - P_1))$, when considering the disjunction $\psi_1 \vee \psi_2$, we can suppress $\wedge \bigwedge \neg(P - P_1)$ in ψ_1. The disjunction of all these formulas $\psi_1 \vee \psi_2$ for each model ω of φ, gives the formula as written in this definition. □

Proof of Remark 16:

1. For any formula φ, $\mathbf{Mod}(ForgetV(\varphi, V)) = \{Force(\omega, L_2)/L_2 \subseteq_{cons} V^{\pm}\} = \{Force(\omega, L_1 \cup L_2)/L_1 \subseteq P^-, L_2 \subseteq_{cons} V^{\pm}, \omega \models L_1\}$ and $\mathbf{Mod}(ForgetLitVar(\varphi, P^-, V)) = \{Force(\omega, L_1 \cup L_2)/L_1 \subseteq P^-, L_2 \subseteq_{cons} V^{\pm}, [\omega \not\models L_1$ or $L_2 = \emptyset]\}$. Thus, $\mathbf{Mod}(ForgetV(\varphi, V) \vee ForgetLitVar(\varphi, P^-, V)) = \mathbf{Mod}(ForgetV(\varphi, V)) \cup \mathbf{Mod}(ForgetLitVar(\varphi, P^-, V)) = \{Force(\omega, L_1 \cup L_2)/L_1 \subseteq P^-, L_2 \subseteq_{cons} V^{\pm}\} = \mathbf{Mod}(ForgetLit(\varphi, P^- \cup V^{\pm}))$.
2. We get $\mathbf{Mod}(ForgetV(\varphi, V) \wedge ForgetLitVar(\varphi, P^-, V)) = \mathbf{Mod}(ForgetV(\varphi, V)) \cap \mathbf{Mod}(ForgetLitVar(\varphi, P^-, V))$. Let us suppose now that φ is a formula uniquely defined in P. This means that the set $\mathbf{Mod}(\varphi) \cap P$ is a singleton. Then, if $L_1 \subseteq P^-$, $\omega \models \varphi$ and $\omega \not\models L_1$, we get $Force(\omega, L_1) \notin \mathbf{Mod}(\varphi)$,

and also, for any $\omega' \in \mathbf{Mod}(\varphi)$ and any consistent subsets L_2, L'_2 of $V^{\pm}$, $Force(\omega, L_1 \cup L_2) \neq Force(\omega', L'_2)$. Thus, for any element $Force(\omega, L_1 \cup L_2)$ of $\mathbf{Mod}(ForgetLitVar(\varphi, P^-, V))$ which is also in $\mathbf{Mod}(ForgetV(\varphi, V)$, we get $\omega \models L_1$, thus also $L2 = \emptyset$, thus $Force(\omega, L_1 \cup L_2) = \omega$, thus this element is in $\mathbf{Mod}(\varphi)$. Thus we get $ForgetV(\varphi, V) \wedge ForgetLitVar(\varphi, P^-, V) \models \varphi$, and, by Remark 12, $ForgetV(\varphi, V) \wedge ForgetLitVar(\varphi, P^-, V) \equiv \varphi$. □

Proof of the adequacy of Remark 17 with Definition 15:

Let V be a set of propositional symbols and $L \cup \{l\}$ be a consistent set of literals without symbol in V such that $l \notin L$.

For any formula Φ, we have $\mathbf{Mod}(\neg l \wedge ForgetV(l \wedge \Phi, v_l)) = \{Force(\omega, \sim l)/\omega \models \Phi, \omega \models l\}$.
This is the set of all the models of Φ *actively forced by* $\sim l$: l was satisfied by ω while $Force(\omega, \sim l)$ differs from ω in that it satisfies $\neg l$. Then we get
$\mathbf{Mod}(ForgetV(\neg l \wedge ForgetV(l \wedge \Phi, v_l), V)) =$
$\{Force(Force(\omega, \sim l), L_2)/\omega \models \Phi, \omega \models l, L_2 \subseteq_{cons} V\pm\} =$
$\{Force(\omega, \{\sim l\} \cup L_2)/\omega \models \Phi, \omega \models l, L_2 \subseteq_{cons} V\pm\}$.

Thus, from Definition 11, wet get
$\mathbf{Mod}(ForgetLitVar(\varphi, L, V)) = \mathbf{Mod}_1 \cup \mathbf{Mod}_2$ and
$\mathbf{Mod}(ForgetV(\neg l \wedge ForgetV(l \wedge ForgetLitVar(\varphi, L, V), v_l), V)) = \mathbf{Mod}_3 \cup \mathbf{Mod}_4$
where

1. $\mathbf{Mod}_1 = \{\omega/\omega \models \varphi\}$;
2. $\mathbf{Mod}_2 = \{Force(\omega, L_1 \cup L_2)/\omega \models \varphi, \omega \not\models L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm\}$;
3. $\mathbf{Mod}_3 = \{Force(\omega, \{\sim l\} \cup L_2)/\omega \models \varphi, \omega \models l, L_2 \subseteq_{cons} V\pm\}$;
4. $\mathbf{Mod}_4 = \{Force(Force(\omega, L_1 \cup L_2), \{\sim l\} \cup L'_2)/\omega \models \varphi, \omega \models l, \omega \not\models L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm, L'_2 \subseteq_{cons} V\pm\}$.

Notice that we get: $v_l \notin L, v_l \notin V$ and $\mathcal{V}(L \cup \{l\}) \cap V = \emptyset$. Thus we get
$\mathbf{Mod}_4 =$
$\{Force(\omega, \{\neg l\} \cup L_1 \cup L'_2 \cup (L_2 - \sim L'_2))/\omega \models \varphi, \omega \models l, \omega \not\models L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm, L'_2 \subseteq_{cons} V\pm\}$.

When the sets L_2 and L'_2 run over the set of the consistent subsets of $V^{\pm}$, the set $L''_2 = L'_2 \cup (L_2 - \sim L'_2)$ also runs over the same set and we get:

$$\mathbf{Mod}_4 = \{Force(\omega, \{\sim l\} \cup L_1 \cup L''_2)/\omega \models \varphi, \omega \models l, \omega \not\models L_1, L_1 \subseteq \sim L, L''_2 \subseteq_{cons} V\pm\}.$$

If $L_1 \subseteq \sim L$ and $\omega \models L_1$, we get
$Force(\omega, \{\sim l\} \cup L_2) = Force(\omega, \{\sim l\} \cup L_1 \cup L_2)$.

Thus we get $\mathbf{Mod}_3 \cup \mathbf{Mod}_4 = \mathbf{Mod}_{34} =$
$\{Force(\omega, \{\sim l\} \cup L_1 \cup L_2)/\omega \models \varphi, \omega \models l, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm\}$.

Similarly, if $\omega \not\models l$ (i.e. $\omega \models \neg l$), we get
$Force(\omega, L_1 \cup L_2) = Force(\omega, \{\sim l\} \cup L_1 \cup L_2)$.
Thus we get $\mathbf{Mod}_2 = \mathbf{Mod}_{2a} \cup \mathbf{Mod}_{2b}$ where:
$\mathbf{Mod}_{2a} = \{Force(\omega, \{\sim l\} \cup L_1 \cup L_2)/\omega \models \varphi, \omega \not\models l, \omega \not\models L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm\}$ and
$\mathbf{Mod}_{2b} = \{Force(\omega, L_1 \cup L_2)/\omega \models \varphi, \omega \not\models L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm\} =$
$\{Force(\omega, L'_1 \cup L_2)/\omega \models \varphi, \omega \not\models L'_1, \neg l \notin L'1, L'_1 \subseteq \{\sim l\} \cup \sim L, L_2 \subseteq_{cons} V\pm\}$.

Since $\omega \not\models \{l\} \cup L_1$ iff $\omega \not\models l$ or $\omega \not\models \cup L_1$, we get:
$\mathbf{Mod}_{2a} \cup \mathbf{Mod}_{34} = \mathbf{Mod}_{2a34} =$
$\{Force(\omega, \{\sim l\} \cup L_1 \cup L_2)/\omega \models \varphi, \omega \not\models \{\sim l\} \cup L_1, L_1 \subseteq \sim L, L_2 \subseteq_{cons} V\pm\} =$
$\{Force(\omega, L'_1 \cup L_2)/\omega \models \varphi, \omega \not\models L'_1, L'_1 \subseteq \{\sim l\} \cup \sim L, \sim l \in L'_1, L_2 \subseteq_{cons} V\pm\}$.

Thus we get $\mathbf{Mod}_{2a34} \cup \mathbf{Mod}_{2b} =$
$\mathbf{Mod}_{234} = \{Force(\omega, L_1 \cup L_2)/\omega \models \varphi, \omega \not\models L_1, L_1 \subseteq \{\sim l\} \cup \sim L, L_2 \subseteq_{cons} V\pm\}$.

Finally we get the result which achieves the proof:
$\mathbf{Mod}(ForgetLitVar(\varphi, L, V) \vee ForgetV(\neg l \wedge ForgetV(l \wedge ForgetLitVar(\varphi, L, V), v_l), V)) =$
$\mathbf{Mod}_1 \cup \mathbf{Mod}_2 \cup \mathbf{Mod}_3 \cup \mathbf{Mod}_4 = \mathbf{Mod}_1 \cup \mathbf{Mod}_{234} =$
$\mathbf{Mod}(ForgetLitVar(\varphi, \{l\} \cup L, V))$.

Thus, we have shown:
$ForgetLitVar(\varphi, \{l\} \cup L, V) = ForgetLitVar(\varphi, L, V) \vee ForgetV(\neg l \wedge ForgetV(l \wedge ForgetLitVar(\varphi, L, V), v_l), V)$. □

References

Lang, J.; Liberatore, P.; and Marquis, P. 2003. Propositional Independence - Formula-Variable Independence and Forgetting. *(Electronic) Journal of Artificial Intelligence Research* 18:391–443. `http://WWW.JAIR.ORG/`.

Lin, F., and Reiter, R. 1994. Forget it! In Mellish, C. S., ed., *AAAI Fall Symposium on Relevance*, 1985–1991. New Orleans, USA: Morgan Kaufmann.

Lin, F. 2001. On strongest necessary and weakest sufficient conditions. *Artficial Intelligence* 128(1–2):143–159.

McCarthy, J. 1986. Application of circumscription to formalizing common sense knowledge. *Artificial Intelligence* 28(1):89–116.

Moinard, Y. 2005. Forgetting literals with varying propositional symbols. In McIlraith, S.; Peppas, P.; and Thielscher, M., eds., 7^{th} *Int. Symposium on Logical Formalizations of Common Sense Reasoning*, 169–176.

Przymusinski, T. C. 1989. An Algorithm to Compute Circumscription. *Artificial Intelligence* 38(1):49–73.

Su, K.; Lv, G.; and Zhang, Y. 2004. Reasoning about Knowledge by Variable Forgetting. In Dubois, D.; Welty, C. A.; and Williams, M.-A., eds., *KR'04*, 576–586. AAAI Press.