# Forget It!

**Fangzhen Lin and Ray Reiter***
Department of Computer Science
University of Toronto
Toronto, Canada M5S 1A4
email: fl@ai.toronto.edu reiter@ai.toronto.edu

## Introduction

We encountered the need for a theory of forgetting in our work on Cognitive Robotics.[1] In our effort to control an autonomous robot, we imagine the robot has a knowledge base representing the initial world state. As the robot performs actions, she needs to bring the knowledge base up to date. To do so, she needs to forget about all the facts that are no longer true, and in such a way that this will not affect any of her possible future actions (Lin and Reiter (2; 3)).

This paper describes in general terms the particular forms of forgetting used in (Lin and Reiter (2; 3)). Specifically, we propose a logical theory to account for: forgetting about a fact (forget that John is a student), and forgetting about a relation (forget the *student* relation). We then apply our notion of forgetting in defining various notion of relevance.

## Forgetting - Basic Concepts

We assume a first-order language $\mathcal{L}$ with equality.

### Forgetting About a Fact

Given a theory $T$ and a ground atom $p$, we want to define what it means to forget about $p$ in $T$. Intuitively, the resulting theory should be weaker than the original one, but entail the same set of sentences that are "irrelevant" to $p$.

We shall give a semantic definition of this particular form of forgetting. To that end, we define an equivalence relation over structures of $\mathcal{L}$. Let $P(\vec{t})$ be a ground atom of $\mathcal{L}$, and $M_1$ and $M_2$ two first-order structures of $\mathcal{L}$. We define $M_1 \sim_{P(\vec{t})} M_2$ iff $M_1$ and $M_2$ agree on everything except possibly on the truth value of $P(\vec{t})$:

1. $M_1$ and $M_2$ have the same domain, and interpret every function the same.
2. For every predicate symbol $Q$ distinct from $P$, $M_1[Q] = M_2[Q]$.
3. Let $\vec{u} = M_1[\vec{t}]$, then for any tuple $\vec{d}$ of the elements in the domain that is distinct from $\vec{u}$, $\vec{d} \in M_1[P]$ iff $\vec{d} \in M_2[P]$.

**Definition 1** Let $T$ be a theory, and $p$ a ground atom. A theory $T'$ is a result of *forgetting about $p$ in $T$* iff for any structure $M$, $M$ is model of $T'$ iff there is a model $M'$ of $T$ such that $M \sim_p M'$.

It is clear that if both $T'$ and $T''$ are results of forgetting $p$ in $T$, then they are logically equivalent. In the following, we shall denote by $forget(T;p)$ the result of forgetting $p$ in $T$. We shall show below that $forget(T;p)$ always exists. But first, we show that $forget(T;p)$ has the right properties.

Intuitively, forgetting leaves a weaker theory. Formally, we have:

**Proposition 2** $T \models forget(T;p)$.

Intuitively, forgetting about $p$ should not affect other things. To formulate precisely this property, for any ground atom $P(\vec{t})$, and any formula $\varphi$, we denote by $\varphi \downarrow P(\vec{t})$ the result of replacing every occurrence of the form $P(\vec{t'})$ in $\varphi$ by

$$P(\vec{t'}) \wedge \vec{t'} \neq \vec{t}.^2$$

Informally, we can say that $P(\vec{t})$ is irrelevant to a sentence $\phi$ iff $\phi$ it is equivalent to a sentence of the form $\varphi \downarrow P(\vec{t})$. In particular, if $P$ is a 0-ary predicate, then $P$ is irrelevant to $\phi$ iff it is equivalent to a sentence that does not mention $P$.

**Proposition 3** For any sentence $\varphi$, $T \models \varphi \downarrow p$ iff $forget(T;p) \models \varphi \downarrow p$.

One of the main technical results of this paper is that for any finite theory $T$ and atom $p$, $forget(T;p)$ exists and can be obtained from $T$ and $p$ by simple syntactic manipulations. To formulate this result, we need to introduce some notation.

Let $\varphi$ be a formula, and $P(\vec{t})$ a ground atom. We denote by $\varphi[P(\vec{t})]$ the result of replacing every occurrence

---

*Fellow of the Canadian Institute for Advanced Research


[1] Joint project with Yves Lespérance, Hector Levesque, Daniel Marcu, and Rich Scherl.

[2] $\vec{t} = \vec{t'}$ is a shorthand for $t_1 = t'_1 \wedge \cdots \wedge t_n = t'_n$. If $n = 0$, then $\vec{t} = \vec{t'}$ is *true*.

of the form $P(\vec{t'})$ in $\varphi$ by

$$[\vec{t} = \vec{t'} \wedge P(\vec{t})] \vee [\vec{t} \neq \vec{t'} \wedge P(\vec{t'})].$$

It is clear that $\varphi$ and $\varphi[P(\vec{t})]$ are equivalent. We denote by $\varphi^{+}_{P(\vec{t})}$ the result of replacing $P(\vec{t})$ by $true$ in $\varphi[P(\vec{t})]$, and symmetrically by $\varphi^{-}_{P(\vec{t})}$ the result of replacing $P(\vec{t})$ by $false$ in $\varphi[P(\vec{t})]$.

For instance, if $\varphi$ is

$$student(John) \vee student(Joe) \vee teacher(John),$$

and $p$ is $student(John)$, then $\varphi[p]$ is

$$\begin{aligned}
&John = John \wedge student(John) \vee \\
&John \neq John \wedge student(John) \vee \\
&John = Joe \wedge student(John) \vee \\
&John \neq Joe \wedge student(Joe) \vee \\
&teacher(John).
\end{aligned}$$

So $\varphi^{+}_{p}$ is

$$\begin{aligned}
&John = John \wedge true \vee \\
&John \neq John \wedge true \vee \\
&John = Joe \wedge true \vee \\
&John \neq Joe \wedge student(Joe) \vee \\
&teacher(John),
\end{aligned}$$

which is equivalent to $true$. It can be similarly shown that $\varphi^{-}_{p}$ is equivalent to

$$(John \neq Joe \wedge student(Joe)) \vee teacher(John).$$

We are now ready to state the theorem. Without loss of generality, we consider only singleton theories. It is easy to see that if $T = \{\varphi_1, ..., \varphi_n\}$, then for any atom $p$, $forget(T;p)$ is equivalent to $forget(T';p)$, where $T' = \{\varphi_1 \wedge \cdots \wedge \varphi_n\}$.

**Theorem 4** Let $T = \{\varphi\}$ be a theory, and $p$ a ground atom. Then

$$forget(T,p) = \{\varphi^{+}_{p} \vee \varphi^{-}_{p}\}.$$

**Proof:** Suppose that $\phi$ is a formula. By induction on the syntactic form of $\phi$, we can show that for any structure $M$, and any variable assignment $\sigma$:

1. $M, \sigma \models \phi^{+}_{p}$ iff $M', \sigma \models \phi$, where $M'$ is the unique structure satisfying the properties $M' \sim_p M$ and $M' \models p$.
2. $M, \sigma \models \phi^{-}_{p}$ iff $M', \sigma \models \phi$, where $M'$ is the unique structure satisfying the properties $M' \sim_p M$ and $M' \models \neg p$.

From these results, it is straightforward to obtain the theorem. ■

**Example 5** Let

$T_1 = \{student(John) \vee student(Joe) \vee teacher(John)\}$.

Then

$$forget(T_1; student(John)) = \{true\}.$$

Let $T_2 = \{\varphi\}$, where $\varphi = (\exists x)student(x)$. Since $\varphi[student(John)]$ is

$$(\exists x).(x = John \wedge student(John)) \vee (x \neq John \wedge student(x)).$$

Thus $\varphi^{+}_{student(John)}$ is equivalent to $true$. Therefore

$$forget(T_2; student(John)) = \{true\}.$$

Let $T_3 = \{\varphi\}$, where $\varphi = (\forall x)student(x)$. Since $\varphi[student(John)]$ is

$$(\forall x).(x = John \wedge student(John)) \vee (x \neq John \wedge student(x)).$$

Thus $\varphi^{+}_{student(John)}$ is equivalent to

$$(\forall x).x \neq John \supset student(x),$$

and $\varphi^{-}_{student(John)}$ is equivalent to

$$(\forall x).x \neq John \wedge student(x).$$

Therefore $forget(T_3; student(John))$ is

$$\{(\forall x).x \neq John \supset student(x)\}.$$

■

Notice that $T_2$ and $T_3$ in the example show the necessity of transforming a formula $\varphi$ into $\varphi[p]$ before substituting $p$ by *true* and *false*.

So far we have defined forgetting only for a single atom. Let $p_1, ..., p_n$ be a sequence of ground atoms, we define the *result of forgetting about* $p_1, ..., p_n$ *in* $T$, written $forget(T; p_1, ..., p_n)$, to be, inductively,

$$forget(forget(T; p_1, ..., p_{n-1}); p_n).$$

Semantically, a structure $M$ is a model of $forget(T; p_1, ..., p_n)$ iff it it agrees with a model of $T$ on everything except the truth values of $p_1, ..., p_n$. This means that the order of the atoms does not matter. Indeed, *forget* commutes:

**Proposition 6** For any theory $T$ and any ground atoms $p_1$ and $p_2$, $forget(forget(T; p_1); p_2)$ and $forget(forget(T; p_2); p_1)$ are logically equivalent.

## Forgetting About a Relation

Sometimes it may be necessary for an agent to forget an entire relation instead of an instance of it. As we shall see, forgetting about a relation is a second-order notion. In the following, all theories and formulas are assumed to be in $\mathcal{L}^2$, the second-order extension of $\mathcal{L}$.

Let $P$ be a predicate, we can similarly define the relation $\sim_P$ as: for any structures $M_1$ and $M_2$, $M_1 \sim_P M_2$ iff $M_1$ and $M_2$ agree on everything except possibly on $P$.

**Definition 7** Let $T$ be a theory, and $P$ a predicate. A theory $T'$ is *a result of forgetting* $P$ *in* $T$ iff for any structure $M$, $M \models T'$ iff there is a model $M'$ of $T$ such that $M \sim_P M'$.

Again, if both $T'$ and $T''$ are results of forgetting $P$ in $T$, then $T'$ and $T''$ are logically equivalent. We'll also denote by $forget(T; P)$ the result of forgetting $P$ in $T$.

**Theorem 8** Suppose $T = \{\varphi\}$, and $P$ is an n-ary predicate. Then

$$forget(T; P) = \{(\exists R)\varphi(P/R)\},$$

where $R$ is an n-ary second-order predicate variable, and $\varphi(P/Q)$ is the result of replacing every occurrence of $P$ in $\varphi$ by $Q$.

**Example 9** Consider the theory

$T_1 = \{student(John) \vee student(Joe) \vee teacher(John)\}.$

By Theorem 8, $forget(T; student)$ is

$$\{(\exists R).R(John) \vee R(Joe) \vee teacher(John)\},$$

which is equivalent to *true* because $(\exists R)R(John)$ is valid in second-order logic.

Consider the theory

$T_2 = \{(student(John) \vee student(Joe)) \wedge teacher(John)\}.$

By Theorem 8, $forget(T; student)$ is

$$\{(\exists R).(R(John) \vee R(Joe)) \wedge teacher(John)\},$$

which is equivalent to $teacher(John)$. ■

For the two theories in the example, forgetting happens to be first-order definable. However, this is not true in general. For instance, a result from (Lin and Reiter (2)) implies that there is a finite first-order theory $T$, and a predicate $P$ such that $forget(T; P)$ is not first-order expressible.

Many properties of "forgetting about a fact" hold as well here:

**Proposition 10** Let $T$ be a theory, and $P$ and $P'$ two predicates.

1. $T \models forget(T; P)$.
2. For any formula $\varphi$ that does not mention $P$, $T \models \varphi$ iff $forget(T; P) \models \varphi$.
3. $forget(forget(T; P); P')$ and $forget(forget(T; P'); P)$ are equivalent.

We can also extend *forget* to a sequence of predicates: for a sequence of predicates $P_1, ..., P_n$, $forget(T; P_1, ..., P_n)$ is defined, inductively, to be $forget(forget(T; P_1, ..., P_{n-1}); P_n)$. Again, this definition is independent of the order of the predicates.

The dual of forgetting is remembering. Let $T$ be a theory, and $P_1, ..., P_k$ a sequence of predicates mentioned in $T$. Let $P_{k+1}, ..., P_n$ be the remaining predicates mentioned in $T$. We define the *result of remembering only* $P_1, ..., P_k$ *in* $T$, written $remember(T; P_1, ..., P_k)$, to be the result of forgetting the remaining predicates $P_{k+1}, ..., P_n$ in $T$, i.e.,

$$remember(T; P_1, ..., P_k) = forget(T; P_{k+1}, ..., P_n).$$

The following result relates "remember only" to *prime implicates* (Reiter and de Kleer (4)) for propositional theories. Notice that for any propositional theory $T$, and any primitive proposition $p$, $forget(T; p)$ can be understood in two ways: as "forgetting about a fact" with $p$ as a ground atom, and as "forgetting about a relation" with $p$ as a 0-ary predicate. Fortunately, these two readings are equivalent.

**Theorem 11** Suppose that $T$ is a finite set of propositional clauses. Then for any sequence of primitive propositions $p_1, ..., p_k$, $remember(T; p_1, ..., p_k)$ is equivalent to the set of prime implicates of $T$ that mention only primitive propositions in $\{p_1, ..., p_k\}$.

**Proof:** Let $T'$ be the set of prime implicates of $T$ that mention only primitive propositions in $\{p_1, ..., p_k\}$. By Theorem 8, $remember(T; p_1, ..., p_k)$ is equivalent to a sentence that mentions only primitive propositions in $\{p_1, ..., p_k\}$. Thus by the definition of prime implicates, it is easy to see that $T'$ entails $remember(T; p_1, ..., p_k)$.

Conversely, by Proposition 10, for any sentence $\varphi$ that mentions only primitive propositions in $\{p_1, ..., p_k\}$, we have that $T \models \varphi$ iff $remember(T; p_1, ..., p_k) \models \varphi$. But by the definition of prime implicates, $T \models T'$, therefore $remember(T; p_1, ..., p_k)$ entails $T'$ as well. ■

## Relevance and Irrelevance

What we have defined is a notion of forgetting. What we haven't addressed so far are strategies as what to forget. Given a theory and a class of queries, a natural strategy is to forget everything in the theory that is irrelevant for answering the queries. There are three key notions involved in this strategy: a notion of forgetting; a notion of irrelevance; and a notion of equivalence between two theories w.r.t. a given class of queries. These three notions are related. In fact, any one can be defined in terms of the other two. For instance, if we had a definition of irrelevance, we could then define that two theories are equivalent w.r.t. to a class of queries iff they are logically equivalent to a theory that is obtained from them by forgetting some irrelevant facts. In the following, however, we shall define some notion of irrelevance in terms of various notion of equivalence w.r.t. a class of queries. For simplicity, we consider only single query.

For any first-order theory, and any query which we assume to be a first-order sentence, there are three possible cases: the theory may entail the query or its negation or neither of them. It is then natural to define that two theories $T_1$ and $T_2$ are *equivalent w.r.t. the query* $q$ iff

$$T_1 \models q \text{ iff } T_2 \models q,$$

and

$$T_1 \models \neg q \text{ iff } T_2 \models \neg q.$$

So we define:

**Definition 12** Let $T$ be a theory, $q$ a query, and $p$ a ground atom. We say that $p$ in $T$ is irrelevant for answering $q$ iff $T$ and $forget(T; p)$ are equivalent w.r.t. $q$.

Consider the theory

$$T = \{student(John), student(John) \supset young(John)\},$$

and the query $young(John)$. We see that for this query the fact $student(John)$ is irrelevant, because $T$ and $forget(T; student(John))$ are equivalent w.r.t. $young(John)$. This may seem rather counter-intuitive. However, notice that theory $T$ is logically equivalent to the theory

$$\{student(John), young(John)\}.$$

More generally, by Proposition 3, we see that if the predicate in atom $p$ does not appear in the query, then $p$ will be irrelevant for answering $q$.

One can imagine many objections to the above notion of irrelevance. For example, consider the theory

$$\{student(John) \supset young(John)\}.$$

According to our definition, $student(John)$ is irrelevant for answering the query $young(John)$. One may object to this assertion on the ground that it is too strong, and that in general one should allow conditional relevance. For instance, if it is later learned that $student(John)$ holds, we would be able to answer the desired query positively. However, without any restriction on what can be learned in the future, the notion of relevance becomes trivial, for any irrelevant information can always become relevant once certain other relevant information is learned. Still, it seems intuitive to say that what counts as relevant now partly depends on what information may possibly be learned in the future. The following is a simple minded formalization of this intuition. Again, we define it through a suitable notion of equivalence.

Let $q$ be a query, and $\mathcal{P}$ a set of learnable sentences. We say that two theories $T_1$ and $T_2$ are *equivalent w.r.t. $q$ and $\mathcal{P}$* iff for any subset $\mathcal{P}'$ of $\mathcal{P}$, $T_1 \cup \mathcal{P}'$ and $T_2 \cup \mathcal{P}'$ are equivalent w.r.t. $q$ whenever both $T_1 \cup \mathcal{P}'$ and $T_2 \cup \mathcal{P}'$ are logically consistent. In other words, $T_1$ and $T_2$ are equivalent for answering $q$ regardless of what may be learned later.

**Definition 13** Let $T$ be a theory, $\mathcal{P}$ a set of learnable sentences, $q$ a query, and $p$ a ground atom. We say that $p$ in $T$ is irrelevant for answering $q$ iff $T$ and $forget(T; p)$ are equivalent w.r.t. $q$ and $\mathcal{P}$.

Consider the theory

$$T = \{student(John) \supset young(John), \\ immortal(John) \supset young(John)\}.$$

Suppose the query is $young(John)$, and the set of learnable sentences is

$$\{student(John), \neg student(John)\}.$$

Then $student(John)$ is relevant, but $immortal(John)$ is not. Notice that $forget(T; immortal(John))$ is

$$\{student(John) \supset young(John)\}.$$

In general, if a predicate does not appear in the query, nor in the learnable sentences, then any ground atom constructed from the predicate will be irrelevant for answering the query. However, the converse is in general not true. Consider the theory

$$\{student(John) \supset young(John), \\ student(John) \supset people(John)\}.$$

Suppose the set of learnable sentences is

$$\{student(John), \neg student(John), young(John), \\ \neg young(John), people(John), \neg people(John)\}.$$

Then for query $young(John)$, $student(John)$ is relevant, but $people(John)$ is not. This implies that the agent in this case should actively seek the truth value of $student(John)$, but not that of $people(John)$.

As an extended example, consider again our motivating problem of progressing a database (Lin and Reiter (2)). Suppose $\mathcal{D}$ is a situation calculus theory of actions, and $\mathcal{D}_{S_0}$ an initial database. To compute the new database after action $\alpha$ has been performed, compute first the projection of $\mathcal{D}$ onto the states $S_0$ (the initial state) and $do(\alpha, S_0)$ (the state resulting from performing $\alpha$ in $S_0$), then forget all sentences mentioning the initial state $S_0$ in the union of the projection and $\mathcal{D}_{S_0}$. This strategy of forgetting about $S_0$ is justified because we can show that for all future queries, the initial state is irrelevant regardless of what new knowledge the robot may obtain through her sensors. For sensor readings are always about the current state. In particular, knowledge about the past can never be directly obtained from perception.

## Related Work

In the propositional case, Weber (7) has proposed $\varphi_p^+ \vee \varphi_p^-$ as a form of forgetting, and used it to formalize database updates. In the first-order case, Vladimir Lifschitz[3] has proposed $(\exists Q)\varphi(P/Q)$ as a formalization of forgetting about $P$ in $\varphi$. Independently, Gabbay and Ohlbach (1) have implemented a resolution algorithm for eliminating second-order quantifiers in formulas of the form $(\exists P_1, ..., P_n)\varphi$. However, they were not interested in the notion of forgetting per se, rather they applied their algorithm to compute circumscription, modal correspondence theory, and others that involve second-order universal quantifications. We consider our main contributions to be in proposing a coherent semantics, and showing that it has the desired properties.

A radically different approach is taken by Subramanian (6). A fundamental difference between our definition and Subramanian's is that hers does not in general preserve logical equivalence. It is to a large degree syntax oriented in the sense that the result of forgetting depends on the syntactic forms of theories. It will be interesting to see the tradeoff between syntax oriented approaches and semantics oriented ones.

## Conclusions

We have defined a notion of forgetting, and showed that it can be used to define various notion of irrelevance.

As for future research, we are particularly interested in identifying some special classes of theories for which our notion of forgetting can be efficiently computed. In general, computing $forget(T; p_1, ..., p_n)$ requires doing exponential number (in the size of $n$) of substitutions. It's even worse for $forget(T; P_1, ..., P_n)$: it may not even be first-order definable. In the framework of the situation calculus, we have found some useful special cases for which progression (thus forgetting) can be done efficiently. We are currently investigating if there are similar special cases in the more general context. It is particularly interesting to look for such cases in the context of logic programming. For instance, forgetting about a fact in a logic program is like partially evaluating the involving predicate.

We are also investigating some promising applications of our notion of forgetting. In model-based diagnosis, given a system description $T$ and a set $O$ of observations, it turns out that remembering only the abnormality predicate $ab$ in $T \cup O$ is precisely the *maximally abstract diagnosis* (Saraswat and de Kleer and Raiman (5)) of $T$ and $O$. Our notion of forgetting is also closely related to so-called *hidden or intermediate states*: $T$ can be considered to compute $forget(T; P)$ by using $P$ as an intermediate predicate.

## Acknowledgements

We have benefited much during the course of this work from discussions with and comments by Russ Greiner, Vladimir Lifschitz, and the other members of the University of Toronto Cognitive Robotics Group: Yves Lespérance, Hector Levesque, Bill Millar, Daniel Marcu, and Richard Scherl. This research was funded by the Government of Canada National Sciences and Engineering Research Council, and the Institute for Robotics and Intelligent Systems.

[3] Personal communication.