Semantic Diff as the Basis for Knowledge Base Versioning

Enrico Franconi
Free University of Bolzano
Bolzano, Italy
franconi@inf.unibz.it

Thomas Meyer
Meraka Institute, CSIR
Pretoria, South Africa
tommie.meyer@meraka.org.za

Ivan Varzinczak
Meraka Institute, CSIR
Pretoria, South Africa
ivan.varzinczak@meraka.org.za

Abstract

In this paper we investigate the problem of maintaining and reasoning with different *versions* of a knowledge base. We are interested in the scenario where a knowledge base (expressed in some logical formalism) might evolve over time and, as a consequence, different versions thereof have to be maintained simultaneously in a *parsimonious* way. Moreover, users of the knowledge base should be able to access, not only any specific version, but also the *differences* between two given versions of the knowledge base. We address this problem by proposing a *general* semantic framework for the maintenance of different versions of a knowledge base. It turns out that the notion of *semantic difference* between knowledge bases plays a central role in the framework. We show that an appropriate characterization produces a unique definition of semantic difference which is applicable to a large class of logic-based knowledge representation languages. We then proceed to restrict our attention to finitely generated propositional logics, and show that our semantic framework can be represented syntactically in a particular kind of normal form, referred to as *ordered complete conjunctive normal form* or oc-CNF. This is followed by a generalization in which we show that similar results can be obtained for any syntactic representation (in a finitely generated propositional logic) of the semantic framework. Of particular interest are representations of appropriately chosen normal forms. We expect that our constructions for the propositional case can be extended to more expressive languages, such as description logics (DLs). In that respect, our results add to the investigation of the versioning problem for DL-based ontologies.

Introduction

Consider the situation in which we have a knowledge base (expressed in some logic-based knowledge representation language) which might evolve over time. By evolution here we mean modifications (updates, refinements, etc.) made by a knowledge engineer (or a group of knowledge engineers working collaboratively). In situations such as these, it is frequently the case that different users need access to different versions of the knowledge base. Consequently, there is a need to maintain a number of different versions of a knowledge base. In addition to being able to access specific versions of the same knowledge base, users usually need to be able to distinguish between the different versions as well. More specifically, users need access to a system which is able to perform the following two reasoning tasks: (i) determine whether a given piece of information can be derived from a specific version of the knowledge base (access to a specific version); (ii) determine from which versions of the knowledge base it is possible to derive a given piece of information, and from which versions it is not (articulating the differences between versions). These requirements highlight the need for a system with the following characteristics:

- Different versions of the knowledge base have to be maintained simultaneously in a *parsimonious* way. I.e., one should not store *all* of them, but only some kind of *core* from which all can somehow be reconstructed;
- Since we are dealing with knowledge bases (and not data bases), it should be possible not only to identify the difference in syntax between versions, but also to determine the difference in *meaning* between them.

Besides being of interest in general, the problem of managing different versions of a knowledge base has interesting specific applications. A prominent example (and one of our main motivations in this work) is the problem of *ontology versioning* expressed in a suitable representational formalism such as RDF (Gutiérrez *et al.* 2004) or one of the numerous available description logics (Baader *et al.* 2003). When developing or maintaining an ontology collaboratively, different simultaneous (possibly conflicting) versions thereof might exist at the same time (see Figure 1).

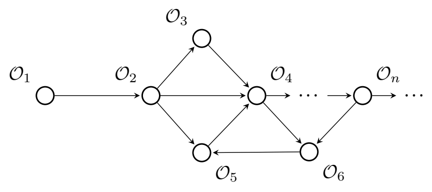

Figure 1: An initial ontology and its subsequent versions.

That can happen due to many reasons, such as different teams working on different modules of the same ontology in parallel, or different developers having different views of the domain, among others. Moreover, modifications performed on ontologies need not be incremental (monotone): information may be added and removed frequently, and it might well happen that the latest ontology is actually closer to one of its preliminary versions than its immediate predecessor. In that respect, in order for the ontology engineers (and users of the ontology) to be able to coordinate their work in an efficient way, they need tools allowing them to (i) keep track of all versions; (ii) determine to what extent two versions of an ontology differ; (iii) revert from one ontology to another (possibly previously agreed upon) one; and (iv) given a particular piece of information, to determine from which of the current versions of the ontology it can be inferred.

This problem is also relevant to other areas of logic-based knowledge representation, such as multi-agent systems, regardless of the underlying logical formalism. Because of that, our focus in this paper is not on a particular application, but rather on a general framework for maintaining different versions of a logical theory. In doing so we first investigate the notion of *semantic difference* between knowledge bases. It turns out that this is a crucial component of the framework for versioning that we propose.

Our results in this regard are quite broadly applicable. As we shall see, it holds (at least) for all logics with a Tarskian consequence relation. These results make it possible to define a semantic framework with the following structure: We suppose that there are n versions of a knowledge base that we need to maintain. We then store a *core knowledge base* (Figure 2), which may, or may not, be one of the n versions, together with the semantic differences between the core knowledge base and the different versions. We refer to this stored information as the *core*. We will show that from the core it is possible to generate (i) any one of the n versions of the knowledge base, and (ii) the semantic difference between any two of the n versions of the knowledge base. We shall refer to this information as the *required output*.

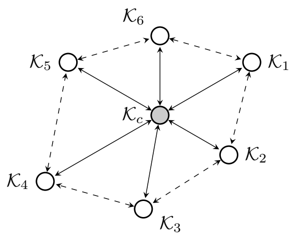

Figure 2: The core knowledge base, from which to access all different versions.

Although applicable to any Tarskian logic, our framework does not address the question from a computational perspective, where it becomes important to consider the specific syntactic representation of the knowledge bases and related information. To do so, we restrict our attention to finitely generated propositional logics, and present ways of storing the core, and generating the required output, in appropriate syntactic forms. We expect that these results can be used to find appropriate syntactic representations expressed in richer logic-based knowledge representation languages (such as description logic languages) as well.

The rest of the paper is organized as follows. After some logical preliminaries we define and investigate a notion of semantic difference. Following that, we present a general framework for knowledge base versioning which is based on our notion of diff. We then turn to a specific normal form with which we illustrate and investigate the issues associated with compact representations of knowledge bases and the diffs in propositional logic. This leads to the description of a general syntactic representation of knowledge bases and diffs. After a discussion of, and comparison with, related work, we conclude with future directions of investigation.

Preliminaries

Given a set X, its power set (the set of all subsets of X) is denoted by $\mathscr{P}(X)$.

In the later parts of the paper we work in a propositional language $\mathcal{L}$ over a set of propositional *atoms* $\mathfrak{P}$, together with the two distinguished atoms $\top$ (*true*) and $\bot$ (*false*), and with the standard model-theoretic semantics. Atoms will be denoted by $p, q, \ldots$ A *literal* is an atom or the negation of an atom. We use $\alpha, \beta, \ldots$ to denote classical propositional formulas. They are recursively defined in the usual way, with connectives $\neg$, $\wedge$, $\vee$, $\rightarrow$ and $\leftrightarrow$.

Given a formula α, *atm*(α) denotes the set of atoms occurring in α. As an example, if $\alpha = p \rightarrow (p \rightarrow q \vee r) \rightarrow (p \rightarrow q \vee r)$, then *atm*$(\alpha) = \{p, q, r\}$.

We denote by $\mathcal{V}$ the set of all propositional valuations or interpretations $v : \mathfrak{P} \longrightarrow \{0, 1\}$, with 0 denoting falsity and 1 truth. With *Mod*(α) we denote the set of all *models* of α (propositional valuations satisfying α).

Classical logical consequence (semantic entailment) and logical equivalence are denoted by $\models$ and $\equiv$ respectively. Given sentences α and β, the meta-statement $\alpha \models \beta$ means *Mod*$(\alpha) \subseteq$ *Mod*(β). $\alpha \equiv \beta$ is an abbreviation (in the meta-language) of $\alpha \models \beta$ *and* $\beta \models \alpha$.

A *knowledge base* $\mathcal{K}$ is a (possibly infinite) set of formulas $\mathcal{K} \subseteq \mathcal{L}$. We extend the above notions of classical entailment and logical equivalence to knowledge bases in the usual way: $\mathcal{K} \models \alpha$ if and only if *Mod*$(\mathcal{K}) \subseteq$ *Mod*(α).

Given a knowledge base, the set of all logical consequences of $\mathcal{K}$ is defined as *Cn*$(\mathcal{K}) = \{\alpha \mid \mathcal{K} \models \alpha\}$. The consequence relation *Cn*$(.)$ associated with a logic is said to be *Tarskian* if and only if it satisfies the properties of *Inclusion*: $X \subseteq$ *Cn*(X); *Idempotence*: *Cn*$($*Cn*$(X)) \subseteq$ *Cn*(X); and *Monotonicity*: $X \subseteq Y$ implies *Cn*$(X) \subseteq$ *Cn*(Y).

It turns out that the following definitions will also be useful: $[\alpha] = \{\beta \mid \alpha \equiv \beta\}$, and $[\mathcal{K}] = \bigcup_{\alpha \in \mathcal{K}} [\alpha]$. For a set of sentences $\alpha_1, \ldots, \alpha_n$ we usually write $\{[\alpha_1], \ldots, [\alpha_n]\}$ for $[\alpha_1] \cup \ldots \cup [\alpha_n]$.

A *clause* is a disjunction of literals. Clauses will be denoted by $\chi, \chi_1, \ldots$ A clause χ is a *complete clause* if and only if each atom in $\mathfrak{P}$ appears exactly once in it. An *ordered complete clause* (alias *oc-clause*) is one in which the atoms appear in sequence according to some pre-established fixed enumeration. In our examples, in oc-clauses we use $p, q, \ldots$ with the obvious enumeration.

For any sentence α, the *ordered complete conjunctive normal form*, or *oc-CNF* of α is a set of oc-clauses X such that $\bigwedge X \equiv \alpha$. It is well-known that every α has an oc-CNF. By convention, the oc-CNF of α is the empty set if and only if α is a tautology.

Semantic Diff

Given two knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, the first step in the development of our framework is to define a notion of *semantic diff* between $\mathcal{K}$ and $\mathcal{K}'$. For the purposes of this section, we assume that the knowledge bases can be expressed in any logic with a Tarskian consequence relation, denoted by $Cn(.)$. In later sections we will restrict ourselves to finitely generated propositional logics.

There is an analogy here with the Unix `diff` command, but whereas `diff` distinguishes between *syntactically* different files, our *semantic diff* will highlight the difference in terms of the (logical) *meaning* between two knowledge bases. For example, although the (propositional) knowledge bases $\{p, q\}$ and $\{p, p \rightarrow q\}$ are syntactically different, they convey exactly the same meaning (they are logically equivalent), and therefore there should be no semantic difference between them. Hence our first requirement is that the knowledge bases $\mathcal{K}$ and $\mathcal{K}'$ are closed under logical consequence.

(P1) $\mathcal{K} = Cn(\mathcal{K})$ and $\mathcal{K}' = Cn(\mathcal{K}')$

We specify the semantic diff of $\mathcal{K}$ and $\mathcal{K}'$ as a pair of sets of sentences $\langle A, R\rangle$. The intuition is that A contains the sentences to be *added* to $\mathcal{K}$, and R the sentences to be *removed* from $\mathcal{K}$ to obtain $\mathcal{K}'$. A will therefore be referred to as the *add-set* of $(\mathcal{K}, \mathcal{K}')$, and R as the *remove-set* of $(\mathcal{K}, \mathcal{K}')$.

(P2) $\mathcal{K}' = (\mathcal{K} \cup A) \setminus R$

In order to avoid redundancy, and to comply with the principle of minimal change, we require that the sentences to be added to $\mathcal{K}$ to obtain $\mathcal{K}'$ should be contained in $\mathcal{K}'$.

(P3) $A \subseteq \mathcal{K}'$

Similarly, sentences to be removed from $\mathcal{K}$ to obtain $\mathcal{K}'$ should be in $\mathcal{K}$.

(P4) $R \subseteq \mathcal{K}$

We require the semantic diff to have a certain *duality* in the sense that it can be used to generate $\mathcal{K}'$ from $\mathcal{K}$, or to generate $\mathcal{K}$ from $\mathcal{K}'$.

(P5) $\mathcal{K} = (\mathcal{K}' \cup R) \setminus A$

In other words, the semantic diff should provide for an 'undo' operation when moving from one version of a knowledge base to another: one should be able to roll back any modification performed.

With the above postulates we can now provide a precise definition of semantic diff between two knowledge bases.

Definition 1 *Let $\mathcal{K}$ and $\mathcal{K}'$ be two knowledge bases, and let A and R be sets of sentences. Then $\langle A, R\rangle$ is* semantic diff compliant *with respect to $(\mathcal{K}, \mathcal{K}')$ if and only if $(\mathcal{K}, \mathcal{K}')$ and $\langle A, R\rangle$ satisfy Postulates P1–P5.*

Semantic diff compliance, as defined above, does not, of course, necessarily guarantee the existence of an operator which is semantic diff compliant. In the definition below we provide a specific construction for the semantic diff operator which we will show to be semantic diff compliant.

Definition 2 *Given two knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, the* ideal semantic diff *of $(\mathcal{K}, \mathcal{K}')$ is the pair $\langle A, R\rangle$, where $A = \mathcal{K}' \setminus \mathcal{K}$, and $R = \mathcal{K} \setminus \mathcal{K}'$.*

Note that neither A nor R are logically closed. To witness, consider the following example:

Example 1 *Let $\mathcal{K} = Cn(p \wedge q)$*[1] *and $\mathcal{K}' = Cn(\neg q)$, and let $\langle A, R\rangle$ be the ideal semantic diff of $(\mathcal{K}, \mathcal{K}')$. Then we have that $A = \{[\neg q], [\neg p \vee \neg q]\}$, and $R = \{[p \wedge q], [p], [q], [p \leftrightarrow q], [p \vee q], [\neg p \vee q]\}$. Clearly $p \vee \neg q \in Cn(A)$, and $p \vee \neg q \in Cn(R)$, but $p \vee \neg q \notin A$ and $p \vee \neg q \notin R$. In fact, for any ideal semantic diff $\langle A, R\rangle$, $\top \notin A$ and $\top \notin R$.*

We now show that the ideal semantic diff is the only operator that is semantic diff compliant with respect to a given pair of knowledge bases.

Theorem 1 *Let $\langle A, R\rangle$ be the ideal semantic diff of $\mathcal{K}$ and $\mathcal{K}'$. Then $\langle A, R\rangle$ is semantic diff compliant with respect to $(\mathcal{K}, \mathcal{K}')$. Moreover, $\langle A, R\rangle$ is the only pair of sets that is semantic diff compliant with respect to $(\mathcal{K}, \mathcal{K}')$.*

We have thus established that there is a unique ideal semantic diff associated with any two knowledge bases. An interesting consequence of the uniqueness of the semantic diff is that its two components are disjoint.

Corollary 1 *For the ideal semantic diff $\langle A, R\rangle$ of $(\mathcal{K}, \mathcal{K}')$, $A \cap R = \emptyset$.*

This is in line with the principle of minimal change, in the sense that one does not want to place a sentence in the add-set, only for it to be subsequently removed (by placing it in the remove-set) or vice versa. Observe also that the ideal semantic diff $\langle A, R\rangle$ of $\mathcal{K}$ and $\mathcal{K}'$ is closely related to their *symmetric difference*: $(\mathcal{K}' \setminus \mathcal{K}) \cup (\mathcal{K} \setminus \mathcal{K}')$. Indeed, it is easily seen that the symmetric difference of $\mathcal{K}$ and $\mathcal{K}'$ is simply the union $A \cup R$ of the add-set and the remove-set of $(\mathcal{K}, \mathcal{K}')$.

Observe also that, as expected, taking the semantic difference of a knowledge base with itself is the only case in which both the add-set and the remove-set are empty.

Corollary 2 *For the ideal semantic diff $\langle A, R\rangle$ of $(\mathcal{K}, \mathcal{K}')$, $\langle A, R\rangle = \langle\emptyset, \emptyset\rangle$ if and only if $\mathcal{K} = \mathcal{K}'$.*

A Framework for Knowledge Base Versioning

The results in the previous section allow us to present our framework for knowledge base versioning. We have a scenario in which there are n versions, $\mathcal{K}_1, \ldots, \mathcal{K}_n$, of a knowledge base that need to be stored, and a core knowledge base $\mathcal{K}_c$. For $1 \leq i, j \leq n$, we will refer to the ideal semantic diff of $(\mathcal{K}_i, \mathcal{K}_j)$ as $\langle D_{ij}, D_{ji}\rangle$ (and the semantic diff of $(\mathcal{K}_c, \mathcal{K}_i)$ as $\langle D_{ci}, D_{ic}\rangle$). Observe that this notation makes sense, primarily because of properties P2 and P5: The add-set D_{ij}

[1]For simplicity, we will write $Cn(\alpha)$ instead of $Cn(\{\alpha\})$.

of (K_i, K_j) is also the remove-set of (K_j, K_i), and the remove-set D_{ji} of (K_i, K_j) is also the add-set of (K_j, K_i).

In order to be able to access any version of the knowledge base, it is sufficient:

- To store the *core* knowledge base $\mathcal{K}_c$, and
- To store D_{ic} and D_{ci} for all $\mathcal{K}_i$ s.t. $1 \leq i \leq n$.

Given this information, we are able to access any version of the knowledge base. To see why, observe firstly that by Theorem 1, $\mathcal{K}_i = (\mathcal{K}_c \cup D_{ci}) \setminus D_{ic}$ for every i such that $1 \leq i \leq n$. Figure 3 depicts such a scenario.

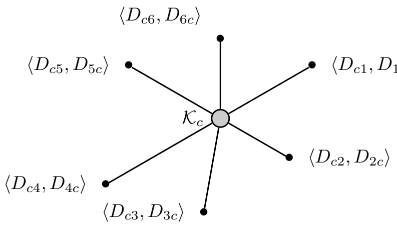
$\langle D_{c6}, D_{6c}\rangle$
$\langle D_{c5}, D_{5c}\rangle$
$\langle D_{c1}, D_{1c}\rangle$
$\mathcal{K}_c$
$\langle D_{c2}, D_{2c}\rangle$
$\langle D_{c4}, D_{4c}\rangle$
$\langle D_{c3}, D_{3c}\rangle$

Figure 3: Core knowledge base and diffs.

Furthermore, for $1 \leq i, j \leq n$, we are able to generate the ideal semantic difference $\langle D_{ij}, D_{ji}\rangle$ of $(\mathcal{K}_i, \mathcal{K}_j)$ directly from the stored information D_{ic}, D_{ci}, D_{cj} and D_{jc}, courtesy of the following result.

Proposition 1 *For* $1 \leq i, j \leq n$,

- $D_{ij} = (D_{cj} \setminus D_{ci}) \cup (D_{ic} \setminus D_{jc})$;
- $D_{ji} = (D_{ci} \setminus D_{cj}) \cup (D_{jc} \setminus D_{ic})$.

Figure 4 shows the overall picture of our framework.

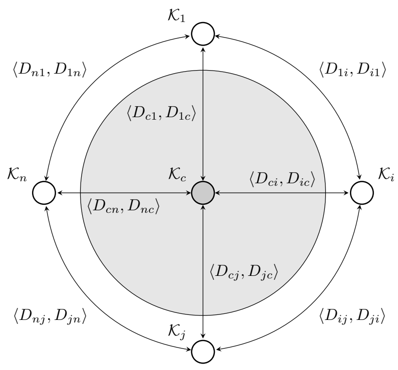
$\mathcal{K}_1$
$\langle D_{n1}, D_{1n}\rangle$
$\langle D_{1i}, D_{i1}\rangle$
$\langle D_{c1}, D_{1c}\rangle$
$\mathcal{K}_n$
$\mathcal{K}_c$
$\langle D_{ci}, D_{ic}\rangle$
$\mathcal{K}_i$
$\langle D_{cn}, D_{nc}\rangle$
$\langle D_{cj}, D_{jc}\rangle$
$\langle D_{nj}, D_{jn}\rangle$
$\langle D_{ij}, D_{ji}\rangle$
$\mathcal{K}_j$

Figure 4: Core knowledge base $\mathcal{K}_c$, the different versions and the respective diffs w.r.t. $\mathcal{K}_c$. The grey area depicts the information that is really stored: the core and the direct diffs.

The observant reader will have noticed that the core knowledge base $\mathcal{K}_c$ is assumed *not* to be one of $\mathcal{K}_1 \ldots \mathcal{K}_n$. The core knowledge base can, for example, be chosen as the 'average' of $\mathcal{K}_1, \ldots, \mathcal{K}_n$, i.e., a representation minimizing the overall semantic diff of $\mathcal{K}_c$ to each of the knowledge bases $\mathcal{K}_1, \ldots, \mathcal{K}_n$. Because such a computation can be carried out offline, it would not have a negative impact on the overall performance of the whole system.

On the other hand, there are good reasons to consider choosing one of $\mathcal{K}_1, \ldots, \mathcal{K}_n$ as the core knowledge base:

- If $\mathcal{K}_c = \mathcal{K}_i$ for some $1 \leq i \leq n$, whenever $\mathcal{K}_i$ has to be accessed there is no need to reconstruct it;
- By the principle of *temporal locality* (Denning 1970), it is reasonable to take $\mathcal{K}_c$ as one of the most recent versions (if not the most recent version);
- By the principle of *spatial locality* (Denning 1970), it is reasonable to choose $\mathcal{K}_c$ as one of the $\mathcal{K}_i$s that are closest (in terms of semantic diff) to the most accessed versions (if not the most accessed one).

All these issues (and consequences thereof) rely on the assumption that extra information of some kind is provided. An analysis of how to choose the core knowledge base and its impact on the efficiency of the versioning system is beyond the scope of this paper. Therefore, we do not develop this further here and we assume that $\mathcal{K}_c$ is *not* one of $\mathcal{K}_1, \ldots, \mathcal{K}_n$. Observe that this assumption does not involve any loss of generality since the basic framework remains essentially the same, regardless of whether the core knowledge base is one of the versions $\mathcal{K}_i$ of the knowledge base.

Compiled Representation

Although our characterisation of semantic diff is on the *knowledge level*, from a computational perspective it is necessary to be able to represent it in a compiled format. In particular, given appropriately compiled representations of the knowledge bases $\mathcal{K}_i$ and $\mathcal{K}_j$, whatever these representations may look like, we are interested in the specification of an *intermediate* representation of the ideal semantic diff which will enable us to do the following:

- From the knowledge base $\mathcal{K}_i$ together with this intermediate representation of the ideal semantic diff, generate the knowledge base $\mathcal{K}_j$;
- From this intermediate representation, possibly in conjunction with other information, generate the ideal semantic diff $\langle D_{ij}, D_{ji}\rangle$ (cf. Definition 2).

In order to do so, we restrict ourselves from now on to finitely generated propositional logics.

Ordered Complete Conjunctive Normal Form

Our initial choice for the representation of knowledge bases is in *ordered complete conjunctive normal form* (oc-CNF).

Definition 3 *For a knowledge base* $\mathcal{K}$, $F(\mathcal{K})$ *is defined as the ordered complete conjunctive normal form of* $\mathcal{K}$. *That is,* $F(\mathcal{K})$ *is a set of oc-clauses such that* $Cn(\bigwedge F(\mathcal{K})) = \mathcal{K}$.

Example 2 *Let* $\mathfrak{P} = \{p, q\}$, *and let* $\mathcal{K} = Cn(p \wedge q)$. *Then* $F(\mathcal{K}) = \{p \vee q, \neg p \vee q, p \vee \neg q\}$.

Our intermediate representation of the ideal semantic diff $\langle D_{ij}, D_{ji}\rangle$ is defined as follows:

Definition 4 *For* $1 \leq i, j \leq n$, *the* intermediate representation *of* $\langle D_{ij}, D_{ji}\rangle$ *is the pair* $\langle I(D_{ij}), I(D_{ji})\rangle$, *where* $I(D_{ij}) = F(\mathcal{K}_j) \setminus F(\mathcal{K}_i)$, *and* $I(D_{ji}) = F(\mathcal{K}_i) \setminus F(\mathcal{K}_j)$.

In Definition 4, the understanding is that $I(D_{ij})$ is the intermediate representation of the add-set D_{ij}, and $I(D_{ji})$ is the intermediate representation of the remove-set D_{ji}, with respect to knowledge bases $\mathcal{K}_i$ and $\mathcal{K}_j$.

Example 3 *Let $\mathfrak{P} = \{p, q\}$, and let $\mathcal{K}_i = Cn(p \wedge q)$ and $\mathcal{K}_j = Cn(\neg q)$. Then $F(\mathcal{K}_i) = \{p \vee q, \neg p \vee q, p \vee \neg q\}$, $F(\mathcal{K}_j) = \{p \vee \neg q, \neg p \vee \neg q\}$, and so $I(D_{ij}) = \{\neg p \vee \neg q\}$ and $I(D_{ji}) = \{p \vee q, \neg p \vee q\}$.*

As expected, this intermediate representation can be used to generate the (compiled representation of the) two knowledge bases from one another.

Theorem 2 *For $1 \leq i, j \leq n$, $F(\mathcal{K}_i) = (F(\mathcal{K}_j) \setminus I(D_{ij})) \cup I(D_{ji}) = (F(\mathcal{K}_j) \cup I(D_{ji})) \setminus I(D_{ij})$.*

Example 4 *Let $\mathfrak{P} = \{p, q\}$, and let $\mathcal{K}_i = Cn(p \wedge q)$ and $\mathcal{K}_j = Cn(\neg q)$. Then $F(\mathcal{K}_i) = \{p \vee q, \neg p \vee q, p \vee \neg q\}$, $I(D_{ji}) = \{p \vee q, \neg p \vee q\}$, and $I(D_{ij}) = \{\neg p \vee \neg q\}$. So $(F(\mathcal{K}_i) \setminus I(D_{ji})) \cup I(D_{ij}) = (\{p \vee q, \neg p \vee q, p \vee \neg q\} \setminus \{p \vee q, \neg p \vee q\}) \cup \{\neg p \vee \neg q\}$ which is equal to $F(\mathcal{K}_j) = \{p \vee \neg q, \neg p \vee \neg q\}$. Similarly, $(F(\mathcal{K}_j) \setminus I(D_{ij})) \cup I(D_{ji}) = (\{p \vee \neg q, \neg p \vee \neg q\} \setminus \{\neg p \vee \neg q\}) \cup \{p \vee q, \neg p \vee q\}$, which is equal to $F(\mathcal{K}_i) = \{p \vee q, \neg p \vee q, p \vee \neg q\}$.*

In addition, and very importantly, this intermediate representation can also be used to generate the *ideal* semantic diff (cf. Definition 2), which was the second of our stated aims. Before we show that this is the case, we need the following two definitions.

Definition 5 *Given sets X and Y, we define*

$$X \uplus Y = \{U \cup V \mid U \in \mathscr{P}(X), \emptyset \neq V, V \in \mathscr{P}(Y)\}$$

So $X \uplus Y$ is a set containing as elements the union of every subset of X with every non-empty subset of Y.

Example 5 *For $X = \{x_1, x_2\}$ and $Y = \{y_1, y_2\}$ we have that $X \uplus Y = \{\{y_1\}, \{y_2\}, \{y_1, y_2\}, \{x_1, y_1\}, \{x_1, y_2\}, \{x_1, y_1, y_2\}, \{x_2, y_1\}, \{x_2, y_2\}, \{x_2, y_1, y_2\}, \{x_1, x_2, y_1\}, \{x_1, x_2, y_2\}, \{x_1, x_2, y_1, y_2\}\}$.*

Definition 6 *For $X \subseteq \mathscr{P}(\mathcal{L})$, $\Delta(X) = \{\bigwedge x \mid x \in X\}$.*

Example 6 *For $X = \{\{\alpha\}, \{\beta, \gamma\}\}$, $\Delta(X) = \{\alpha, \beta \wedge \gamma\}$.*

We are now ready to show that the intermediate representation of the semantic diff can be used to generate a compact representation of the ideal semantic diff. (In Theorem 3 below, keep in mind that $[X] = \bigcup_{\alpha \in X}[\alpha]$.)

Theorem 3 *For $1 \leq i, j \leq n$,*

$$D_{ij} = [\Delta((F(\mathcal{K}_i) \setminus I(D_{ji})) \uplus I(D_{ij}))]$$

That is, $[\Delta((F(\mathcal{K}_i) \setminus I(D_{ji})) \uplus I(D_{ij}))]$ is exactly equal to D_{ij}, the add-set of $(\mathcal{K}_i, \mathcal{K}_j)$, while the set of sentences $[\Delta((F(\mathcal{K}_j) \setminus I(D_{ij})) \uplus I(D_{ji}))]$ is exactly equal to D_{ji}, the remove-set of $(\mathcal{K}_i, \mathcal{K}_j)$. Theorem 3 therefore provides a method for generating the ideal semantic diff $\langle D_{ij}, D_{ji} \rangle$ of $(\mathcal{K}_i, \mathcal{K}_j)$ from $\mathcal{K}_i$, and the intermediate representations $I(D_{ij})$ and $I(D_{ji})$ of D_{ij} and D_{ji}, respectively.

Example 7 *Continuing with Example 4, observe firstly that $F(\mathcal{K}) \setminus I(D_{ji}) = \{p \vee q, \neg p \vee q, p \vee \neg q\} \setminus \{p \vee q, \neg p \vee q\}$ which is equal to $\{p \vee \neg q\}$. Furthermore, $I(D_{ij}) = \{\neg p \vee \neg q\}$, and so $(F(\mathcal{K}) \setminus I(D_{ji})) \uplus I(D_{ij}) = \{p \vee \neg q\} \uplus \{\neg p \vee \neg q\}$, which is equal to $\{\{\neg p \vee \neg q\}, \{p \vee \neg q, \neg p \vee \neg q\}\}$. So $\Delta((F(\mathcal{K}) \setminus I(D_{ji})) \uplus I(D_{ij})) = \{\neg p \vee \neg q, (p \vee \neg q) \wedge (\neg p \vee \neg q)\}$, and therefore $[\Delta((F(\mathcal{K}) \setminus I(D_{ji})) \uplus I(D_{ij}))] = \{[\neg p \vee \neg q], [(p \vee \neg q) \wedge (\neg p \vee \neg q)]\}$, which is equal to $\{[\neg p \vee \neg q], [\neg q]\}$, and which, in turn, is equal to D_{ij}.*

Although Theorem 3 is a useful result, it still leaves us somewhat short of the stated aim of being able to generate D_{ij} and D_{ji} directly from the stored information $F(\mathcal{K}_c)$, $I(D_{ic})$, $I(D_{ci})$, $I(D_{jc})$, and $I(D_{cj})$. More specifically, observe that D_{ij} and D_{ji} are currently being generated from $F(\mathcal{K}_i)$, $F(\mathcal{K}_j)$, $I(D_{ij})$ and $I(D_{ji})$, *none* of which are stored explicitly. Thanks to Theorem 2 we can generate $F(\mathcal{K}_i)$ and $F(\mathcal{K}_j)$ from $F(\mathcal{K}_c)$, $I(D_{ic})$, $I(D_{jc})$, $I(D_{ci})$, and $I(D_{cj})$. This still leaves $I(D_{ij})$, and $I(D_{ji})$ as information not being stored explicitly. The next result shows that $I(D_{ij})$ and $I(D_{ji})$ can be defined in terms of $I(D_{ic})$ and $I(D_{ci})$, $I(D_{jc})$ and $I(D_{cj})$; information that *is* stored explicitly as part of the core.

Theorem 4 *For $1 \leq i, j \leq n$,*

$$I(D_{ij}) = (I(D_{cj}) \setminus I(D_{ci})) \cup (I(D_{ic}) \setminus I(D_{jc}))$$

So Theorems 2, 3 and 4 allow us to generate D_{ij} and D_{ji} from information that is all being stored explicitly — at least in principle. In practice a direct application of these results yields definitions of D_{ij} and D_{ji} that are quite lengthy and unwieldy, and are omitted here due to space considerations. Fortunately, it is possible to simplify these definitions considerably, as the next result shows.

Theorem 5 *For $1 \leq i, j \leq n$, let*

$$\begin{aligned} X &= (F(\mathcal{K}_c) \setminus (I(D_{ic}) \cup I(D_{jc}))) \cup \\ &\quad (I(D_{ci}) \cap I(D_{cj})); \\ X_{ij} &= I(D_{cj}) \cup I(D_{ic}); \text{ and} \\ X_{ji} &= I(D_{jc}) \cup I(D_{ci}). \end{aligned}$$

Then $D_{ij} = [\Delta(X \uplus (X_{ij} \setminus X_{ji}))]$.

Example 8 *Let $\mathcal{K}_i = Cn(\{p \wedge q\})$, $\mathcal{K}_j = Cn(\neg q)$, and $\mathcal{K}_c = Cn(p)$. Then $F(\mathcal{K}_c) = \{p \vee q, p \vee \neg q\}$, $I(D_{ic}) = \emptyset$, $I(D_{jc}) = \{p \vee q\}$, $I(D_{ci}) = \{\neg p \vee q\}$, and $I(D_{cj}) = \{\neg p \vee \neg q\}$. Let $X = (F(\mathcal{K}_c) \setminus (I(D_{ic}) \cup I(D_{jc}))) \cup (I(D_{ci}) \cap I(D_{cj}))$, which is equal to $\{p \vee \neg q\}$, let $X_{ij} = I(D_{cj}) \cup I(D_{ic}) = \{\neg p \vee \neg q\}$, and let $X_{ji} = I(D_{jc}) \cup I(D_{ci}) = \{\neg p \vee q, p \vee q\}$. Then $X \uplus (X_{ij} \setminus X_{ji}) = \{p \vee \neg q\} \uplus \{\neg p \vee \neg q\} = \{\{\neg p \vee \neg q\}, \{p \vee \neg q, \neg p \vee \neg q\}\}$. So $[\Delta(X \uplus (X_{ij} \setminus X_{ji}))] = \{[\neg p \vee \neg q], [\neg q]\}$, which is our D_{ij}. Similarly, $X \uplus (X_{ji} \setminus X_{ij}) = \{p \vee \neg q\} \uplus \{\neg p \vee q, p \vee q\} = \{\{\neg p \vee q\}, \{p \vee q\}, \{p \vee \neg q, \neg p \vee q\}, \{p \vee \neg q, p \vee q\}, \{p \vee \neg q, \neg p \vee q, p \vee q\}\}$. So $[\Delta(X \uplus (X_{ij} \setminus X_{ji}))] = \{[\neg p \vee \neg q], [p \vee q], [q], [p \leftrightarrow q], [p], [p \wedge q]\}$, which is our D_{ji}.*

The ideal semantic diff $\langle D_{ij}, D_{ji} \rangle$ can thus be generated from X, X_{ij}, and X_{ji} as defined in Theorem 5.

A General Syntactic Representation

In the previous section we showed how knowledge base versioning can be represented in a specific normal form — ordered complete conjunctive normal form (oc-CNF). While oc-CNF is useful in gaining a proper understanding of the representation of versioning, it is not a compact form of representation, and may therefore not be as useful from a computational perspective. In this section we provide a more general syntactic representation of versioning. We do not investigate which normal forms are appropriate for versioning. This is left as future work. Rather, the results in this section can provide the basis for such an investigation.

Given a knowledge base $\mathcal{K}$, we let $\Phi(\mathcal{K})$ be a sentence representing $\mathcal{K}$. That is, $\mathit{Mod}(\Phi(\mathcal{K})) = \mathit{Mod}(\mathcal{K})$. In general $\Phi(\mathcal{K})$ can be any sentence representing $\mathcal{K}$, but in practice $\Phi(\mathcal{K})$ will be some normal form for $\mathcal{K}$. We shall therefore refer to $\Phi(\mathcal{K})$ as the *normal form for* $\mathcal{K}$. In what follows below, we frequently need to refer to $\Phi(I(D_{ij}))$, where $I(D_{ij})$ is a component of the intermediate representation $\langle I(D_{ij}), I(D_{ji})\rangle$ of a semantic diff (cf. Definition 4). For readability we abbreviate this to $\Phi(D_{ij})$, and we refer to it as the normal form for D_{ij}.

In this setting, therefore, the *core*, i.e., the information that we store explicitly, consists of the normal form for $\mathcal{K}_c$ together with the normal forms of the semantic diff components, D_{ci} and D_{ic} for $i = 1, \ldots, n$.

We now proceed to show that all the information we are interested in can be generated from the core. Firstly, the normal form of any version $\mathcal{K}_i$ of the knowledge base can be generated from the core.

Theorem 6 *For $i = 1, \ldots, n$,*

$$\Phi(\mathcal{K}_i) \equiv (\Phi(\mathcal{K}_c) \vee \neg\Phi(D_{ic})) \wedge \Phi(D_{ci}).$$

Example 9 *Continuing from our Example 8, let $\Phi(\mathcal{K}_i) = p \wedge q$, $\Phi(\mathcal{K}_j) = \neg q$, $\Phi(\mathcal{K}_c) = p$, $\Phi(D_{ic}) = \top$, $\Phi(D_{jc}) = p \vee q$, $\Phi(D_{ci}) = \neg p \vee q$, and $\Phi(D_{cj}) = \neg p \vee \neg q$. Then $(\Phi(\mathcal{K}_c) \vee \neg\Phi(D_{ic})) \wedge \Phi(D_{ci}) = (p \vee \neg(\top)) \wedge (\neg p \vee q)$, which is logically equivalent to $\Phi(\mathcal{K}_i)$. Similarly, $(\Phi(\mathcal{K}_c) \vee \neg\Phi(D_{jc})) \wedge \Phi(D_{cj}) = (p \vee \neg(p \vee q)) \wedge (\neg p \vee \neg q)$, which is logically equivalent to $\Phi(\mathcal{K}_j)$.*

Next we show that the ideal semantic diff of any two knowledge bases $\mathcal{K}_i$ and $\mathcal{K}_j$ can be generated from the core. Recall from Theorem 5 that, in order to do so using oc-CNF, we first generate the sets X, X_{ij}, and X_{ji}. We first show that we can generate appropriate normal forms for these sets.

Proposition 2 *For $1 \leq i, j \leq n$, let X and X_{ij} be defined as in Theorem 5. Then*

$$\begin{aligned}\Phi(X) &\equiv (\Phi(K_c) \vee \neg(\Phi(D_{ic}) \wedge \Phi(D_{jc}))) \wedge \\ &\quad\ (\Phi(D_{ci}) \vee \Phi(D_{cj})); and \\ \Phi(X_{ij}) &\equiv \Phi(D_{ic}) \wedge \Phi(D_{cj}).\end{aligned}$$

Example 10 *Continuing from Examples 8 and 9, observe that $(\Phi(K_c) \vee \neg(\Phi(D_{ic}) \wedge \Phi(D_{jc}))) \wedge (\Phi(D_{ci}) \vee \Phi(D_{cj}))$ $= (p \vee \neg(\top \wedge (p \vee q))) \wedge ((\neg p \vee q) \vee (\neg p \vee \neg q))$, which is logically equivalent to $p \vee \neg q$, and therefore to $\Phi(X)$. Also, $\Phi(D_{ic}) \wedge \Phi(D_{cj}) = \top \wedge (\neg p \vee \neg q)$, which is logically equivalent to $\neg p \vee \neg q$, and therefore to $\Phi(X_{ij})$. Similarly, $\Phi(D_{jc}) \wedge \Phi(D_{ci}) = (p \vee q) \wedge (\neg p \vee q)$, which is logically equivalent to q, and therefore to $\Phi(X_{ji})$.*

This puts us in a position to show how $\langle D_{ij}, D_{ji}\rangle$ can be generated from the normal forms of X, X_{ij} and X_{ji}.

Theorem 7 *For $i \leq i, j \leq n$, let $\Phi(X)$, $\Phi(X_{ij})$ (and $\Phi(X_{ji})$) be generated as in Proposition 2. Then*

$$D_{ij} = \mathit{Cn}(\Phi(X) \wedge (\Phi(X_{ij}) \vee \neg\Phi(X_{ji}))) \setminus \mathit{Cn}(\Phi(X)).$$

Thus to determine if a sentence is an element of D_{ij}, we need to do two entailment checks: We need to check whether it follows from $\Phi(X) \wedge (\Phi(X_{ij}) \vee \neg\Phi(X_{ji}))$ but does not follow from $\Phi(X)$. (And similarly for D_{ji}, of course.)

Example 11 *Continuing from Examples 8 and 10, observe that $\Phi(X) \wedge (\Phi(X_{ij}) \vee \neg\Phi(X_{ji})) \equiv (p \vee \neg q) \wedge ((\neg p \vee \neg q) \vee \neg q) \equiv \neg q$. Recall also that $\Phi(X) \equiv p \vee \neg q$. Therefore $\mathit{Cn}(\neg q) \setminus \mathit{Cn}(p \vee \neg q) = \{[\neg q], [\neg p \vee \neg q]\}$, which is our D_{ij}. Also, observe that $\Phi(X) \wedge (\Phi(X_{ji}) \vee \neg\Phi(X_{ij})) \equiv (p \vee \neg q) \wedge (q \vee \neg(\neg p \vee \neg q)) \equiv p \wedge q$. And since $\Phi(X) \equiv p \vee \neg q$, we have $\mathit{Cn}(p \wedge q) \setminus \mathit{Cn}(p \vee \neg q) = \{[p \wedge q], [p], [q], [p \rightarrow q], [\neg p \vee \neg q], [p \vee q]\}$, which is equal to D_{ji}.*

Related Work

To the best of our knowledge, the problem of determining the difference between two (logical) representations of a given domain is a quite recent topic of investigation. Originally some work has been done on a *syntax-based* notion of diff between ontologies (Noy and Musen 2002). Here our focus is different: despite the role that syntax might play in the evolution of a knowledge base, our primary focus is on the semantics, i.e., the difference in *meaning* between two versions of a knowledge base.

The problem of determining the logical (semantic) diff between knowledge bases was originally investigated by Kontchakov *et al.* in the context of DL ontologies (Kontchakov *et al.* 2008). There the need for a semantic-driven notion of diff, in contrast to a simple syntax-based one, is motivated, and variants thereof are presented for the lightweight description logic DL-Lite. Their notion of diff, however, differs from ours in that (i) there the difference between ontologies $\mathcal{O}_1$ and $\mathcal{O}_2$ is defined with respect to their shared *signature*, i.e., the set of symbols of the language that are common to both $\mathcal{O}_1$ and $\mathcal{O}_2$; and (ii) they define diff between $\mathcal{O}_1$ and $\mathcal{O}_2$ as a *refinement* of $\mathcal{O}_1$ with respect to $\mathcal{O}_2$, i.e., what we call the add-set of $\mathcal{O}_1$ to get $\mathcal{O}_2$. It can be checked that our Definition 1 encompasses both (i) and (ii), making the symmetry of diff explicit and also showing the properties expected from such an operation.

In the same lines of Kontchakov *et al.*'s work, Konev *et al.* (Konev *et al.* 2008) investigate the problem of logical diff for another lightweight description logic, namely $\mathcal{EL}$ (Baader 2003), and provide algorithms for determining the refinement of an ontology with respect to another.

The two aforementioned approaches are specific to a particular family of DLs. Our general definition of semantic diff (Definition 1) holds for any logic which has a Tarskian consequence relation. Moreover, our framework in terms of compact representations remains the same for more expressive formalisms, relying essentially on the existence of appropriate normal forms for the respective logic. In that sense, the approach by Kontchakov *et al.* and Konev *et al.* can be seen as a special case of ours.

Jiménez-Ruiz *et al.* address the problem of maintaining multiple versions of an ontology by adapting the Concurrent Versioning paradigm from software engineering to the ontology versioning case (Jiménez-Ruiz *et al.* 2009). Although their motivations and ours overlap to some extent, they focus on the definition of a high-level architecture for versioning, which is based upon the notions of semantic diff defined by Kontchakov *et al.* and Konev *et al.*. For that reason, the work by Jiménez-Ruiz *et al.* is not directly comparable to ours. Nevertheless it is worth mentioning that the crucial difference between their architecture and ours is that there *all* versions of a given ontology are stored in a server's shared repository, whereas with our general framework we keep only the core information which is sufficient to reconstruct the entire set of versions (cf. Figure 4).

Observe that the topic investigated here is not the same as the one usually addressed in the belief change community, either from a belief *revision* (Gärdenfors 1988) or a belief *update* (Katsuno and Mendelzon 1992) perspective. Rather knowledge base versioning complements it. In traditional belief change, one has a source knowledge base, a new given piece of information, and the main task is to determine what the *target* knowledge base looks like. On the other hand, in knowledge base versioning one has a set of knowledge bases (pairwise different), and the focus resides in determining the piece of information on which they (pairwise) differ. So the operators associated with belief versioning (generating one version from another, and determining the difference between two versions) can only be applied *after* any belief change has already taken place.

Also, although there are similarities between the notion of knowledge base versioning presented here and *truth maintenance systems* (Doyle 1979), they are largely superficial. In the case of truth maintenance systems, the goal is to provide systems enriched with *justifications* indicating why certain beliefs are (or are not) held, whereas in our case the add- and remove-sets fulfill quite a different purpose—that of encoding the semantic difference between knowledge bases.

Conclusion and Future Work

We have laid the groundwork for a knowledge base versioning system built up on a notion of semantic difference which is intuitive, simple, and at the same time general: our results hold for any knowledge bases formalized in a logic with a Tarskian consequence relation.

With our framework, one does not need to store all the information regarding all existing versions of a knowledge base, but rather only part of it, namely, the *core*. Our results show that the core corresponds precisely to the sufficient piece of information required to reconstruct *any* of the versions of the knowledge base. They also show that the differences in meaning between any two given versions in the system can be determined through the core, without direct access to any of the versions at all. This is the case irrespective of the underlying syntactic representation.

We plan to pursue further work by investigating which normal forms are more appropriate as syntactical representations for knowledge base versioning. Our results for oc-CNF provide us with a basis for such an investigation.

Finally, since our semantic constructions also apply to more expressive logics (than propositional logic), we are currently investigating extensions of our framework to the description logic $\mathcal{ALC}$.

References

[Baader *et al.* 2003] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *Description Logic Handbook*. Cambridge University Press, 2003.

[Baader 2003] F. Baader. Terminological cycles in a description logic with existential restrictions. In V. Sorge, S. Colton, M. Fisher, and J. Gow, editors, *Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 325–330. Morgan Kaufmann Publishers, 2003.

[Denning 1970] P.J. Denning. Virtual memory. *ACM Computing Surveys*, 2(3):153–189, 1970.

[Doyle 1979] Jon Doyle. A truth maintenance system. *Artif. Intell.*, 12(3):231–272, 1979.

[Gärdenfors 1988] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. MIT Press, 1988.

[Gutiérrez *et al.* 2004] C. Gutiérrez, C.A. Hurtado, and A.O. Mendelzon. Foundations of semantic web databases. In *Proceedings of the 23rd ACM Symposium on Principles of Database Systems*, pages 95–106. ACM Press, 2004.

[Jiménez-Ruiz *et al.* 2009] E. Jiménez-Ruiz, B. Cuenca Grau, I. Horrocks, and R. Berlanga. Building ontologies collaboratively using content CVS. In *22nd International Workshop on Description Logics*, 2009.

[Katsuno and Mendelzon 1992] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In P. Gärdenfors, editor, *Belief revision*, pages 183–203. Cambridge University Press, 1992.

[Konev *et al.* 2008] B. Konev, D. Walther, and F. Wolter. The logical difference problem for description logic terminologies. In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proceedings of IJCAR*, number 5195 in LNAI, pages 259–274. Springer-Verlag, 2008.

[Kontchakov *et al.* 2008] R. Kontchakov, F. Wolter, and M. Zakharyaschev. Can you tell the difference between DL-Lite ontologies? In J. Lang and G. Brewka, editors, *Proceedings of KR*, pages 285–295. AAAI Press/MIT Press, 2008.

[Noy and Musen 2002] N. Noy and M. Musen. PromptDiff: A fixed-point algorithm for comparing ontology versions. In R. Dechter, M. Kearns, and R. Sutton, editors, *Proceedings of AAAI*, pages 744–750. AAAI Press/MIT Press, 2002.