A Logic-Based Perspective on Agent Reconfiguration: Preliminary Report

Thomas Meyer
CSIR Meraka Institute and
School of Computer Science
University of KwaZulu-Natal
South Africa
tommie.meyer@meraka.org.za

Ivan Varzinczak
CSIR Meraka Institute and
School of Computer Science
University of KwaZulu-Natal
South Africa
ivan.varzinczak@meraka.org.za

***Abstract*—We investigate the problem of maintaining and reasoning with different configurations of a logic-based agent. Given specific contexts, there may be several possible usual configurations that the agent's knowledge base can be in, and that one may want to access at different times. This can happen due to foreseeable exceptional situations one has to cater for, or different environments in which the agent may have to operate, or simply due to upgrades of the agent's initial configuration. In all these cases, there is a need for a system capable of managing possibly conflicting versions of the knowledge base and allowing the agent to switch between any two given configurations at run time. Building on Franconi *et al*.'s framework for propositional knowledge base versioning, here we establish the logical foundations for a general semantic-based architecture of such a system. Central to our approach is the notion of logical difference, which allows us to determine the essential pieces of information on which two given configurations of an agent differ.**

I. Introduction

In logic-based Artificial Intelligence, the notion of knowledge base is a prominent one. Knowledge bases (KBs) are (usually finite) sets of statements specified in some formal, machine-processable, language. Such a language is usually a logic-based one, with a well defined formal semantics and an associated method for performing reasoning that can be automated in computer systems. This formalization is intended to capture or represent knowledge about some particular domain of interest in a concise, unambiguous and useful way. The type of knowledge that is represented can range from simple facts about a given scenario to agents' beliefs and knowledge in cognitive robotics and multi-agent systems.

The practical use of knowledge bases is at the very heart of the AI endeavor: autonomous systems designed to perform rationally in an environment will have associated with them a knowledge base of some kind. As an example, consider the following scenario depicting a nuclear power station (Figure 1). In a particular power plant there is an atomic pile and a cooling system, both of which can be either on or off. An agent is in charge of detecting hazardous situations and preventing the plant from malfunctioning.

Such an agent is equipped with a knowledge base which can be configured to formalize both the environment and the agent's behavior and beliefs. For instance, the agent's KB may contain statements formalizing the facts that "a situation in which the pile is on and the cooling off is a hazardous one", or "the effect of switching the pile off brings about a non-hazardous situation", or even "if the agent believes that there is danger, then he must perform the action of switching the pile off". Using this configuration of the knowledge about its domain, the agent can then draw conclusions, i.e., query the KB to derive implicit knowledge from the knowledge that is explicitly stated in its KB. Examples of inferences that are relevant to the agent are *planning* (how to safely shut the plant down), *introspecting* (what important facts the agent ignores) and *diagnosing* (what has switched the cooling system off).

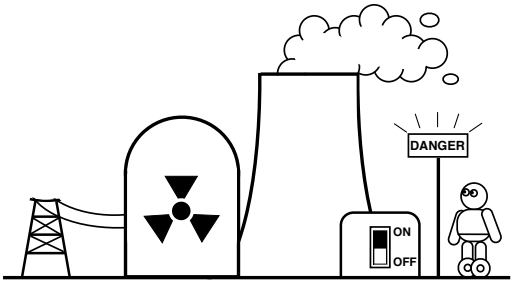

Fig. 1. The nuclear power station and its controlling agent [1].

It turns out that, depending on specific contexts or situations, there may be more than one possible (and desirable) configuration the agent's knowledge base can be in. For instance, in the event of a hazardous situation in the nuclear power plant, the agent may be forced to switch to a configuration whose rules override most of those of the normal configuration. Alternatively, when the power plant is in maintenance mode, the agent needs to be reconfigured to perform actions he is not allowed when in normal mode. One can also argue for the case where the knowledge engineer in charge of (re)configuring the agent wants to be able to determine which configuration is the most appropriate for the agent given a particular context. That is achieved by querying which configuration entails specific facts or a desirable behavior. One immediate consequence of this is that the most (if not all) usual configurations should then somehow always be available.

These scenarios raise the question of how to manage different co-existing configurations or versions of an agent's knowledge base. Importantly: How to do that at *run time*.

One naïve approach would be to assume that the agent starts with an initial knowledge base and then let it evolve. Such is the approach extensively investigated in the literature on belief revision [2], [3] and belief update [4]. It works pretty well in circumstances like the one in which, say, the agent is moved from the nuclear power plant to a thermoelectric one and its knowledge base must be re-configured for it to become operable in the new environment. Nevertheless, theory revision is computationally expensive, in particular that of knowledge bases that are specified in more expressive languages [5], [6], and therefore not feasible at run time. A similar argument holds against *learning* (and actually one does not even want to let an agent fiddle around with a nuclear power plant to discover how it works). Moreover, revision cycles as studied in traditional belief change tend to 'forget' previous configurations: Putting the agent's knowledge base back to one of its previous configurations may require a thorough revision of the current one.

Another naïve way of approaching this problem would be to allow the agent to store more than one usual configuration of its knowledge base. This has the obvious problem that it can lead to logical inconsistency (two or more versions of the knowledge base may conflict), or it may give rise to the more general problem of undesirable implicit consequences [7], [8].

Finally, even if storing all possible configurations at the same time were logically harmless, the amount of space required to keep all that information would not be optimal.

All the issues raised above call for the need of a general system allowing the knowledge engineer to maintain and reason with different *configurations* of a *logic-based* agent. Yet, such a general and versatile system still does not exist.

A similar problem to the one we are interested in here has been addressed by Franconi *et al*. in the context of ontology versioning [9]. However, the referred approach is propositional in nature, whereas the design of agents requires languages that are more expressive than classical propositional logic. In this paper, we consider one of such logical formalisms. Our aim is then to apply Franconi *et al*.'s semantic constructions to the above motivated scenario of agent reconfiguration, thereby extending their method to logical formalisms other than classical propositional logic.

The remainder of the present paper is structured as follows: After presenting the logical formalism of a modal logic of actions in which one can formalize a particular agent's behavior (Section II), we recall Franconi *et al*.'s notion of logical difference recasting it in the more expressive language of multi-modal logic (Section III). This allows us to determine the difference between two given configurations of an agent's knowledge base. We then define a general architecture for logic-based agent reconfiguration (Section IV) in which the notion of logical difference plays a central role. After a discussion (Section V) and a comparison with related work (Section VI), we conclude the paper with a summary of the contributions and some open questions and threads for future investigation.

II. Logical Preliminaries and Notation

There is in the AI literature a fair number of logical languages for formalizing agents and their behavior. Most of them are essentially variants of modal logic [10] having the multi-modal logic K as backbone. Given that, in this work we shall consider multi-modal logic K as our underlying formalism. In particular, we are interested in the formalization of an agent's beliefs about the behavior of actions, their effects and preconditions. (All we shall say in Sections III and IV should then transfer smoothly to other modal-based formalisms like logics of action and belief, obligations, and combinations thereof.)

The language of our multi-modal logic of actions is built upon a (finite) set of *atomic propositions* $\mathfrak{Prop}$ (together with the distinguished atom $\bot$), and a (finite) set of *atomic actions* $\mathfrak{Act}$, using the logical connectives $\wedge$, $\neg$, and a set of modal operators $[a]$, one for each $a \in \mathfrak{Act}$. Propositions are denoted by $p, q, \ldots$, and formulas by $\alpha, \beta, \ldots$ These are constructed according to the rule:

$$\alpha ::= p \mid \bot \mid \alpha \wedge \alpha \mid \neg\alpha \mid [a]\alpha \tag{1}$$

The other connectives ($\vee$, $\rightarrow$, $\ldots$), the special atom $\top$, and the operator $\langle a \rangle$ are defined in terms of the others in the usual way. Formulas of the form $[a]\alpha$ are used to specify the *effects* of actions and they read as "after *every* execution of action a, the formula α holds". $\langle a \rangle$ is mostly used to specify the *executability* of actions: $\langle a \rangle \top$ reads as "there is a possible execution of action a".

With $\mathcal{L}$ we denote the set of all formulas of the language generated according to Rule (1).

As for the semantics we adopt the standard possible-world semantics from modal logic.

Definition 1: A Kripke model is a tuple $\mathscr{M} = \langle W, R, V \rangle$ where W is a set of possible worlds, $R = \langle R_{a_1}, \ldots, R_{a_n} \rangle$, where each $R_{a_i} \subseteq W \times W$ is an accessibility relation on W, $1 \leq i \leq |\mathfrak{Act}|$, and $V : W \mapsto 2^{\mathfrak{Prop}}$ is a valuation function.

Definition 2 (Satisfaction): Given $\mathscr{M} = \langle W, R, V \rangle$:

- $\mathscr{M}, w \not\Vdash \bot$ for every $w \in W$;
- $\mathscr{M}, w \Vdash p$ iff $p \in V(w)$;
- $\mathscr{M}, w \Vdash \alpha \wedge \beta$ iff $\mathscr{M}, w \Vdash \alpha$ and $\mathscr{M}, w \Vdash \beta$;
- $\mathscr{M}, w \Vdash \neg\alpha$ iff $\mathscr{M}, w \not\Vdash \alpha$;
- $\mathscr{M}, w \Vdash [a_i]\alpha$ iff $\mathscr{M}, w' \Vdash \alpha$ for all w' s.t. $(w, w') \in R_{a_i}$.

Given $\alpha \in \mathcal{L}$ and $\mathscr{M} = \langle W, R, V \rangle$, we say that $\mathscr{M}$ *satisfies* α if there is at least one $w \in W$ s.t. $\mathscr{M}, w \Vdash \alpha$. We say that $\mathscr{M}$ is a *model of* α iff $\mathscr{M}, w \Vdash \alpha$ for every $w \in W$. With $Mod(\alpha)$ we denote the set of all models of α.

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 $Mod(\alpha) \subseteq Mod(\beta)$. $\alpha \equiv \beta$ 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 $Mod(.)$, entailment and logical equivalence to knowledge bases in the usual way:

$Mod(\mathcal{K})$ is the set of all Kripke models $\mathscr{M}$ satisfying every formula in $\mathcal{K}$; $\mathcal{K} \models \alpha$ if and only if $Mod(\mathcal{K}) \subseteq Mod(\alpha)$.

Example 1: In our nuclear power plant scenario, we have the atoms $\mathfrak{Prop} = \{\mathsf{p}, \mathsf{c}, \mathsf{h}\}$, where p stands for "the atomic pile is on", c for "the cooling system is on", and h for "hazardous situation". As for the actions, we have $\mathfrak{Act} = \{\mathsf{f}, \mathsf{m}\}$, where f stands for "flipping the pile switch", and m for "malfunction". One possible specification of such a scenario is given by the following knowledge base:

$$\mathcal{K}_1 = \left\{ \begin{array}{c} (\mathsf{p} \wedge \neg\mathsf{c}) \leftrightarrow \mathsf{h}, \mathsf{h} \rightarrow \langle\mathsf{m}\rangle\top, \mathsf{p} \rightarrow [\mathsf{f}]\neg\mathsf{p}, \\ \langle\mathsf{f}\rangle\top, [\mathsf{m}](\neg\mathsf{p} \wedge \neg\mathsf{c}) \end{array} \right\}$$

For the sake of presentation we do not develop here a solution to the frame problem [11] of our own. Instead, we refer the reader to existing modal-based solutions to the frame problem [12] and the ramification problem [13], [14] which can be integrated into the present formalism in a straightforward way. Unless otherwise indicated, we shall assume that all relevant frame axioms are stated in the knowledge base.

$\mathcal{K}_1$ basically says that "a hazardous situation is one in which the pile is on and the cooler off", "a hazardous situation may lead to a malfunction", "if the pile is on, then flipping switches it off", "one can always flip the pile switch", and finally "after a malfunction, there is no pile or cooling system whatsoever". Then we can conclude for instance $\mathcal{K}_1 \models \mathsf{p} \rightarrow [\mathsf{f}]\neg\mathsf{h}$, $\mathcal{K}_1 \models [\mathsf{m}]\bot \rightarrow (\neg\mathsf{p} \vee \mathsf{c})$, and $\mathcal{K}_1 \not\models [\mathsf{m}]\langle\mathsf{f}\rangle\neg\mathsf{p} \vee [\mathsf{f}]\mathsf{h}$. ■

Given a knowledge base $\mathcal{K}$, 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 our modal logic is a *Tarskian consequence relation* in the sense that it satisfies the following properties:

- $\mathcal{K} \subseteq Cn(\mathcal{K})$ (Inclusion)
- $Cn(Cn(\mathcal{K})) \subseteq Cn(\mathcal{K})$ (Idempotence)
- $\mathcal{K} \subseteq \mathcal{K}'$ implies $Cn(\mathcal{K}) \subseteq Cn(\mathcal{K}')$ (Monotonicity)

We finish this section with the following useful definitions: $\widehat{\alpha} := \{\beta \mid \alpha \equiv \beta\}$, and $\widehat{\mathcal{K}} = \bigcup_{\alpha \in \mathcal{K}} \widehat{\alpha}$. Given sentences $\alpha_1, \ldots, \alpha_n$ we shall write $\{\widehat{\alpha_1}, \ldots, \widehat{\alpha_n}\}$ for $\widehat{\alpha_1} \cup \ldots \cup \widehat{\alpha_n}$.

III. Logical Difference between Configurations

In this section we recast Franconi *et al.*'s notion of logical difference between propositional knowledge bases [9] in the context of our modal logic of actions.

We start by supposing that there are n different co-existing configurations (alias versions) of an agent's knowledge base that we need to maintain. One way to address the problems mentioned in the Introduction is to store a *core knowledge base* (Figure 2), which may, or may not, be one of the n versions, together with the differences (whatever that means) between the core knowledge base and the different versions. We refer to this stored information as the *core*. The goal is to give a suitable definition of the core in such a way that from the core it is possible to generate (*i*) any one of the n configurations, and (*ii*) the difference between any two of the n configurations.

Given two knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, the first step in the development of our framework is to define a notion of

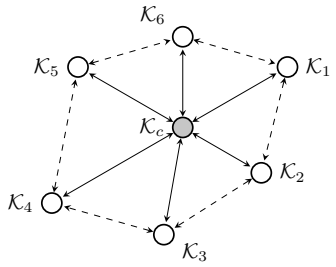

Fig. 2. The core knowledge base, from which to access all different versions.

logical diff between $\mathcal{K}$ and $\mathcal{K}'$. Such a notion of diff should be *syntax-independent*: For example, although the knowledge bases $\{[a](p \wedge q)\}$ and $\{[a]p, [a](p \rightarrow q)\}$ are syntactically different, they convey exactly the same meaning (they are logically equivalent), and therefore there should be no logical difference between them. Hence our logical diff should highlight the difference in terms of the (logical) *meaning* between two knowledge bases. The first step to achieve this is to require that the knowledge bases $\mathcal{K}$ and $\mathcal{K}'$ be closed under logical consequence (cf. end of Section II). This gives us the first postulate below:

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

Henceforth we specify the logical 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 $(\mathcal{K}, \mathcal{K}')$. These two basic notions give us the following postulate:

(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 logical 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 logical diff should provide for an 'undo' operation when moving from one configuration of a knowledge base to another: one should be able to roll back any modification performed.

With the above postulates we have a precise definition of logical diff between two agent's configurations.

Definition 3: Let $\mathcal{K}$ and $\mathcal{K}'$ be two knowledge bases, and let $- : 2^{\mathcal{L}} \times 2^{\mathcal{L}} \mapsto 2^{\mathcal{L}} \times 2^{\mathcal{L}}$ be an operator such that $\mathcal{K} - \mathcal{K}' = \langle A, R\rangle$, for A and R sets of sentences. We say that $-$ is *logical diff compliant* with respect to $(\mathcal{K}, \mathcal{K}')$ if and only if $\mathcal{K}$, $\mathcal{K}'$, A and R satisfy Postulates P1–P5.

Logical diff compliance, as defined above, does not, of course, necessarily guarantee the existence of an operator which is logical diff compliant. The definition below provides a specific (in this case set theoretical) construction for the logical diff operator which we shall show to be logical diff compliant.

Definition 4: Given knowledge bases $\mathcal{K}$ and $\mathcal{K}'$, the *ideal logical diff* of $(\mathcal{K}, \mathcal{K}')$ is the operator $-_i$ defined as $\mathcal{K} -_i \mathcal{K}' = \langle A, R\rangle$, 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 2: Let $\mathcal{K} = Cn(\mathcal{K}_1)$, where $\mathcal{K}_1$ is the knowledge base from Example 1, and let $\mathcal{K}' = Cn(\mathcal{K}_2)$, where

$$\mathcal{K}_2 = \left\{ \begin{array}{c} (\mathsf{p} \wedge \neg \mathsf{c}) \rightarrow \mathsf{h}, \mathsf{h} \leftrightarrow \langle \mathsf{m}\rangle \top, \\ \mathsf{p} \rightarrow [\mathsf{f}]\neg \mathsf{p}, \neg \mathsf{p} \rightarrow [\mathsf{f}]\mathsf{p}, \langle \mathsf{f}\rangle \top, [\mathsf{m}]\mathsf{p} \end{array} \right\}$$

Let $\mathcal{K} -_i \mathcal{K}' = \langle A, R\rangle$. Then we have

$$A = \{\widehat{\langle \mathsf{m}\rangle \top \rightarrow \mathsf{h}}, \widehat{\neg \mathsf{p} \rightarrow [\mathsf{f}]\mathsf{p}}, \widehat{[\mathsf{m}]\mathsf{p}}, \widehat{[\mathsf{m}](\mathsf{p} \vee \mathsf{c})}\}$$

$$R = \left\{ \begin{array}{c} \mathsf{h} \rightarrow \widehat{(\mathsf{p} \wedge \neg \mathsf{c})}, \widehat{[\mathsf{m}](\neg \mathsf{p} \wedge \neg \mathsf{c})}, \widehat{[\mathsf{m}]\neg \mathsf{p}}, \widehat{[\mathsf{m}]\neg \mathsf{c}}, \\ \widehat{[\mathsf{m}]\neg \mathsf{p} \leftrightarrow \neg \mathsf{c}}, \widehat{[\mathsf{m}](\neg \mathsf{p} \vee \neg \mathsf{c})}, \widehat{[\mathsf{m}](\mathsf{p} \vee \neg \mathsf{c})} \end{array} \right\}$$

Clearly $[\mathsf{m}]\mathsf{p} \vee \neg \mathsf{c} \in Cn(A)$, but $[\mathsf{m}]\mathsf{p} \vee \neg \mathsf{c} \notin A$, and $[\mathsf{m}]\neg \mathsf{p} \vee \mathsf{c} \in Cn(R)$, but $[\mathsf{m}]\neg \mathsf{p} \vee \mathsf{c} \notin R$. In fact, one can see that for any $\langle A, R\rangle$ obtained from the ideal logical diff between $\mathcal{K}$ and $\mathcal{K}'$, $\top \notin A$ and $\top \notin R$. ∎

As shown by Franconi *et al.*, the ideal logical diff is the only operator that is logical diff compliant with respect to a given pair of knowledge bases.

Theorem 1: Let $-_i$ be the ideal logical diff of $\mathcal{K}$ and $\mathcal{K}'$. Then $-_i$ is logical diff compliant with respect to $(\mathcal{K}, \mathcal{K}')$. Moreover, $-_i$ is the only operator that is logical diff compliant with respect to $(\mathcal{K}, \mathcal{K}')$.

Proof: To prove that $-_i$ is semantic diff compliant with respect to $(\mathcal{K}, \mathcal{K}')$ is trivial. To prove that it is unique, assume that $\mathcal{K}$, $\mathcal{K}'$, A' and R' also satisfy Postulates P1–P5. By P3, $A' \subseteq \mathcal{K}'$. If $A' \subset A$, then Postulate P2 is violated, regardless of what R' looks like. Suppose that A' contains an $\alpha \in \mathcal{K}' \setminus A$. Then $\alpha \in \mathcal{K}$, and so Postulate P5 is violated. It thus follows that $A' = A$. Similarly, if $R' \subset R$, then Postulate P5 is violated, regardless of what A' looks like. And suppose R' contains a $\beta \in \mathcal{K} \setminus R$. Then $\beta \in \mathcal{K}'$, and so P2 is violated. Hence $R' = R$. ∎

An interesting consequence of the uniqueness of the ideal logical diff is that its two components are disjoint.

Corollary 1: Let $\mathcal{K} -_i \mathcal{K}' = \langle A, R\rangle$. Then $A \cap R = \emptyset$.

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 logical 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 logical 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: Let $\mathcal{K} -_i \mathcal{K}' = \langle A, R\rangle$. Then $\langle A, R\rangle = \langle \emptyset, \emptyset\rangle$ if and only if $\mathcal{K} = \mathcal{K}'$.

IV. A Framework for Agent Reconfiguration

Based on the results of the previous section, here we present our framework for agent reconfiguration. We have a scenario in which there are n configurations, $\mathcal{K}_1, \ldots, \mathcal{K}_n$, of an agent's knowledge base that need to be stored, and a core configuration $\mathcal{K}_c$. The notion of difference is a symmetric one: Given Postulates P2 and P5, the add set of $(\mathcal{K}_i, \mathcal{K}_j)$ is also the remove-set of $(\mathcal{K}_j, \mathcal{K}_i)$, and the remove-set of $(\mathcal{K}_i, \mathcal{K}_j)$ is also the add-set of $(\mathcal{K}_j, \mathcal{K}_i)$. In that sense, from now on we shall refer to the ideal diff of $(\mathcal{K}_i, \mathcal{K}_j)$ as $\langle D_{ij}, D_{ji}\rangle$ (and the diff of $(\mathcal{K}_c, \mathcal{K}_i)$ as $\langle D_{ci}, D_{ic}\rangle$), for $1 \leq i, j \leq n$.

As we are going to see, in order to be able to access any specific configuration of the knowledge base, it is sufficient:

- To store the *core* configuration $\mathcal{K}_c$, 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 configuration 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.

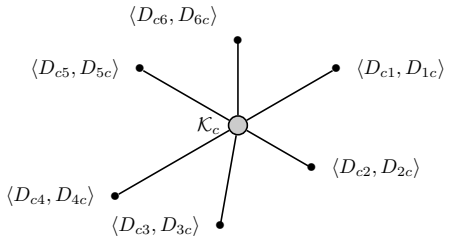

Fig. 3. Core configuration and diffs.

Furthermore, for $1 \leq i, j \leq n$, we are able to generate the ideal logical 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}, thanks to the following result.

Theorem 2: 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.

We conclude this section with a brief note on computational complexity. The reconstruction of a version $\mathcal{K}_i$ from the core $\mathcal{K}_c$ and the diff $\langle D_{ci}, D_{ic}\rangle$ is obviously *linear* in the size of the input, since it amounts to simple set union and set difference operations. The same holds in constructing the difference between two versions $\mathcal{K}_i$ and $\mathcal{K}_j$ via the core. On the other hand, computing each of the D_{ij}'s is a computationally expensive task and its complexity depends on that of the entailment problem of the underlying logic. For instance, in Franconi *et al.*'s framework, to determine whether a sentence

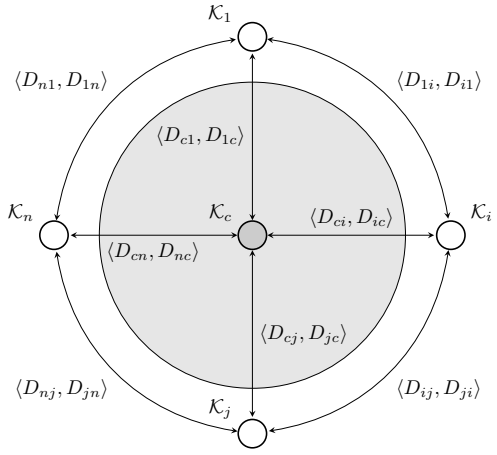

Fig. 4. Core configuration $\mathcal{K}_c$, the different co-existing alternative configurations and the respective diffs w.r.t. $\mathcal{K}_c$. The grey area depicts the information that is really stored: the core and the direct diffs.

is an element of D_{ij} or not requires two entailment tests in propositional logic, which is NP-COMPLETE. On the bright side, computing the diffs is a task that can be carried out offline and therefore does not jeopardize the overall performance of the system at run time.

V. OUTLOOK: HOW TO GET THE CORE CONFIGURATION

The observant reader will have noticed that the core configuration $\mathcal{K}_c$ is assumed *not* to be one of $\mathcal{K}_1, \ldots, \mathcal{K}_n$. The core knowledge base can, for instance, be chosen as the 'average' of $\mathcal{K}_1, \ldots, \mathcal{K}_n$, i.e., a representation minimizing the overall logical diff of $\mathcal{K}_c$ to each of the alternative configurations $\mathcal{K}_1, \ldots, \mathcal{K}_n$. 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 configuration:

- 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* [15], 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* [15], it is reasonable to choose $\mathcal{K}_c$ as one of the $\mathcal{K}_i$s that are closest (in terms of logical diff) to the most accessed configuration (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 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.

VI. 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. Besides being of interest in its own right, the problem of managing different configurations (versions) of a knowledge base has other interesting specific applications. A prominent example is the problem of *ontology versioning*: When developing or maintaining an ontology collaboratively, different simultaneous (possibly conflicting) versions thereof might exist at the same time (Figure 5).

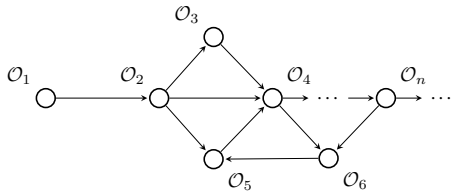

Fig. 5. An initial ontology and its subsequent versions.

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 to 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.

Given the well-know connection between modal logic K as we studied here and the description logic $\mathcal{ALC}$ [16], which is the backbone of most ontology representation languages, we believe that our constructions should transfer smoothly to the case of ontology versioning.

Kontchakov *et al*. [17] have investigated the problem of determining the logical (semantic) diff between DL ontologies. 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 3 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*. [18] investigate a notion of logical diff for lightweight

description logics and provide algorithms for determining the refinement of an ontology with respect to another.

Franconi *et al*. [9] define a general framework for knowledge base versioning on which we base our constructions. There the ultimate goal is the definition of a general versioning management system for DL ontologies. Using propositional logic as a stepping stone, Franconi *et al*. give the syntactic counterpart of the semantic constructions in terms of a specific normal form, namely ordered complete conjunctive normal form (oc-CNF).

VII. Conclusion and Future Work

The contributions of the present paper are as follows:

- We have given a logic-based perspective on the problem of agent reconfiguration relating it to the more general problem of knowledge base versioning.
- We have made the first steps in generalizing Franconi *et al*.'s propositional framework to logics that are more expressive than classical propositional logic. By investigating the case of a modal logic of actions, we provided evidence supporting Franconi *et al*.'s conjecture that their results hold for knowledge bases formalized in any logic with a Tarskian consequence relation.
- We have provided an intuitive, simple and computationally tractable framework with which to manage different co-existing configurations of a logical agent.

With our framework, one does not need to store all the information regarding all existing configurations 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 configurations of the knowledge base. They also show that the differences in meaning between any two given configurations in the system can be determined through the core, without direct access to any of the versions at all. Since all our definitions are on the *knowledge level*, this is the case irrespective of the underlying syntactic representation.

Logical languages that are richer than the propositional one have more structure — in the syntax and in the semantics. This may reflect in additional properties of the logical diff and its relationship with the core configuration beyond the ones investigated by Franconi *et al*.'s in the propositional case and that here we have recast in basic multi-modal logic K. What these properties are and the impact the choice of the underlying language can have on the overall system we leave for future investigation. Our results in this paper provide the basic framework with which to move from the propositional to the more expressive cases.

Here we restrict ourselves to multi-modal logic K because it is the backbone of most logics of action, beliefs, and combinations thereof commonly found in the AI literature. In that sense, we believe that our constructions can be smoothly generalized to the case of other logics for agent design extending K. We are currently investigating extensions of our framework to the description logic $\mathcal{ALC}$.

Although our characterization of logical diff is on the knowledge level and 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. We plan to pursue further work by investigating which normal forms are more appropriate as syntactical representations for knowledge base versioning. Franconi *et al*.'s results for oc-CNF provide us with a basis for such an investigation.

References

[1] K. Britz, T. Meyer, and I. Varzinczak, "Preferential reasoning for modal logic," Submitted, 2011.

[2] C. Alchourrón, P. Gärdenfors, and D. Makinson, "On the logic of theory change: Partial meet contraction and revision functions," *Journal of Symbolic Logic*, vol. 50, pp. 510–530, 1985.

[3] S. Hansson, *A Textbook of Belief Dynamics: Theory Change and Database Updating*. Kluwer Academic Publishers, 1999.

[4] H. Katsuno and A. Mendelzon, "On the difference between updating a knowledge base and revising it," in *Belief revision*, P. Gärdenfors, Ed. Cambridge University Press, 1992, pp. 183–203.

[5] I. Varzinczak, "Action theory contraction and minimal change," in *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, J. Lang and G. Brewka, Eds. AAAI Press/MIT Press, 2008, pp. 651–661.

[6] ——, "On action theory change," *Journal of Artificial Intelligence Research*, vol. 37, pp. 189–246, 2010.

[7] A. Herzig and I. Varzinczak, "Cohesion, coupling and the meta-theory of actions," in *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, L. Kaelbling and A. Saffiotti, Eds. Morgan Kaufmann Publishers, 2005, pp. 442–447.

[8] ——, "Metatheory of actions: beyond consistency," *Artificial Intelligence*, vol. 171, pp. 951–984, 2007.

[9] E. Franconi, T. Meyer, and I. Varzinczak, "Semantic diff as the basis for knowledge base versioning," in *13th International Workshop on Nonmonotonic Reasoning (NMR)*, 2010.

[10] P. Blackburn, J. van Benthem, and F. Wolter, *Handbook of Modal Logic*. Elsevier North-Holland, 2006.

[11] J. McCarthy and P. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Machine Intelligence*, B. Meltzer and D. Mitchie, Eds. Edinburgh University Press, 1969, vol. 4, pp. 463–502.

[12] R. Demolombe, A. Herzig, and I. Varzinczak, "Regression in modal logic," *Journal of Applied Non-Classical Logic*, vol. 13, no. 2, pp. 165–185, 2003.

[13] D. Zhang and N. Foo, "EPDL: A logic for causal reasoning," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, B. Nebel, Ed. Morgan Kaufmann Publishers, 2001, pp. 131–138.

[14] M. Castilho, A. Herzig, and I. Varzinczak, "It depends on the context! A decidable logic of actions and plans based on a ternary dependence relation," in *9th International Workshop on Nonmonotonic Reasoning (NMR)*, 2002.

[15] P. Denning, "Virtual memory," *ACM Computing Surveys*, vol. 2, no. 3, pp. 153–189, 1970.

[16] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, Eds., *The Description Logic Handbook: Theory, Implementation and Applications*, 2nd ed. Cambridge University Press, 2007.

[17] R. Kontchakov, F. Wolter, and M. Zakharyaschev, "Can you tell the difference between DL-Lite ontologies?" in *Proceedings of the 11th International Conference on Principles of Knowledge Representation and Reasoning (KR)*, J. Lang and G. Brewka, Eds. AAAI Press/MIT Press, 2008, pp. 285–295.

[18] B. Konev, D. Walther, and F. Wolter, "The logical difference problem for description logic terminologies," in *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR)*, ser. LNAI, A. Armando, P. Baumgartner, and G. Dowek, Eds., no. 5195. Springer-Verlag, 2008, pp. 259–274.